MESOCIAL MEDIA--TOXIC COMMENT DETECTION IN COMMUNITIES USING PYSPARK ON DATABRICKS

A PROJECT PHASE II REPORT

Submitted by

JASHAREEN J 2116231801066

JISHANTHI R 2116231801072

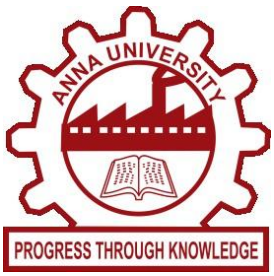
MADHAN P 2116231801089

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS), CHENNAI – 602 105

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this Report titled "SOCIAL MEDIA--TOXIC COMMENT DETECTION IN COMMUNITIES USING PYSPARK ON DATABRICKS" is the bonafide work of "JASHAREEN J A(2116231801066) , JISHANTHI R (231801072) and MADHAN P (231801089)" who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. J. GNANASEKAR M.E., Ph.D.,

Professor and Head,

Department of Artificial Intelligence and Data Science,

Rajalakshmi Engineering College

Thandalam – 602 105

SIGNATURE

Mrs. SELVARANI

Professor ,

Department of Artificial Intelligence and Data Science,

Rajalakshmi Engineering College

Thandalam – 602 105

Submitted to Project Viva-Voce Examination held on

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

Initially I thank the Almighty for being with us through every walk of my life and showering his blessings through the endeavor to put forth this report.

My sincere thanks to our Chairman Mr. S. MEGANATHAN, M.E., F.I.E., and our Chairperson Dr. (Mrs.) THANGAM MEGANATHAN, M.E., Ph.D., for providing me with the requisite infrastructure and sincere endeavoring educating me in their premier institution.

My sincere thanks to Dr. S.N. MURUGESAN M.E., Ph.D., our beloved Principal for his kind support and facilities provided to complete our work in time.

I express my sincere thanks to Dr. J M Gnanasekar M.E., Ph.D., Head of the Department of Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. I convey my sincere and deepest gratitude to our internal guide, Dr. Suresh Kumar S M.E., Ph.D., Professor, Department of Artificial Intelligence and Data Science, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Finally, I express my gratitude to my parents and classmates for their moral support and valuable suggestions during the course of the project.

Abstract

The exponential rise of social media and online communication platforms has transformed the way individuals interact, share opinions, and build communities. However, this rapid digital expansion has also led to an alarming increase in toxic comments, including hate speech, harassment, obscenity, and abusive language, which threaten user well-being and degrade the quality of online discourse. Detecting and mitigating such toxicity manually is both time-consuming and inefficient, especially considering the massive volume of user-generated content produced every second. To address this growing concern, this project presents a comprehensive and scalable Toxic Comment Detection System that integrates Natural Language Processing (NLP) and Machine Learning (ML) techniques within a Big Data environment using Databricks and PySpark. The system is designed to automatically identify, classify, and analyze toxic comments in online communities, providing an intelligent and data-driven approach to maintaining healthy digital spaces.

The implementation begins with data collection and preprocessing, using a labeled dataset (tds.csv) containing user comments annotated with multiple categories such as *IsToxic*, *IsAbusive*, *IsThreat*, *IsHatespeech*, *IsObscene*, and *IsRacist*. These records are imported into the Databricks platform and converted into a PySpark DataFrame for distributed data handling and high-performance processing. Preprocessing operations include removing null and empty values, trimming whitespace, and cleaning text to ensure consistency and quality. Once the dataset is prepared, feature extraction is performed using TF-IDF (Term Frequency–Inverse Document Frequency), which transforms raw text into numerical feature vectors that represent the relative importance of words within the corpus.

Following feature extraction, a Logistic Regression model is trained using the PySpark MLlib library to classify comments as toxic or non-toxic. The dataset is divided into training and testing subsets, allowing for accurate model evaluation through performance metrics such as accuracy, precision, recall, and F1-score. Visualization components within Databricks are employed to generate charts and graphs illustrating label distributions, model results, and word frequency patterns. These visual insights enable interpretable analysis and support data-driven decision-making for comment moderation.

The system demonstrates that combining NLP with distributed Big Data technologies can efficiently process vast amounts of unstructured text while maintaining scalability and interpretability. The modular design of the system allows for easy future enhancement, including the integration of deep learning models like BERT or LSTM, multilingual comment analysis, and real-time toxicity detection. Overall, this project exemplifies the transformative potential of Big Data analytics in addressing modern challenges in digital communication. By automating the detection of harmful content, the proposed system not only enhances online safety and community health but also lays the groundwork for developing intelligent, context-aware, and adaptive content moderation systems in the future.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Background

With the rapid growth of social media platforms such as YouTube, Twitter, and Facebook, online communities have become central to digital communication and expression. While these platforms provide an opportunity for open discussion, they have also become a breeding ground for toxic comments, including hate speech, abusive language, harassment, and discrimination. Such harmful content affects user well-being, degrades community standards, and creates hostile digital spaces.Manual moderation of millions of comments is inefficient, inconsistent, and unable to keep up with the pace of user-generated content. Therefore, automated systems using Natural Language Processing (NLP) and Machine Learning (ML) techniques have become essential to identify and filter toxic language effectively.In this project, PySpark and Databricks were used to perform large-scale text analysis and classification. The dataset (tds.csv) consists of user comments and multiple toxicity-related labels such as *IsToxic*, *IsAbusive*, *IsThreat*, *IsProvocative*, *IsHatespeech*, and others. Using the Databricks environment allowed distributed processing, quick visualization, and efficient model execution. Through data cleaning, label analysis, and the implementation of a simple Logistic Regression model, the project demonstrates how large datasets can be processed and analyzed effectively to detect online toxicity.

Motivation

The motivation for this project arises from the growing prevalence of toxic language and harassment on social media platforms. Detecting harmful content manually is a slow and error-prone process that often fails to maintain user safety in real time. This project aims to automate the process of identifying toxic comments using data-driven methods to assist content moderators and reduce human workload.

From a technical perspective, this project also highlights the efficiency of PySpark within Databricks for distributed computation, enabling fast processing of large text datasets. By analyzing and modeling the data directly within a cloud-based notebook, the system demonstrates how scalable tools can be applied to solve practical NLP problems. The project not only focuses on toxicity prediction but also provides meaningful insights into comment distributions, frequency of various toxic behaviors, and text characteristics**.**

Objectives

The main objectives of this project are:

- To load and analyze the toxic comment dataset (tds.csv) in the Databricks environment.
- To clean and preprocess text data by removing null or empty entries and trimming unnecessary whitespace.
- To explore and visualize label distributions (such as *IsToxic*, *IsAbusive*, *IsThreat*, etc.) to understand the nature and frequency of toxic comments.
- To compute summary statistics for text length and other relevant attributes for identifying patterns in comment behavior.
- To extract features using TF-IDF and train a Logistic Regression model to classify toxic versus non-toxic comments.

- To evaluate model performance using metrics such as accuracy and F1-score.
- To visualize insights such as category frequency and text length distribution using charts and Databricks display functions.

Problem Statement

The volume of user-generated comments across social media platforms is increasing exponentially, making manual detection of toxic or abusive language infeasible. Traditional moderation methods rely on manual filtering or static keyword-based systems, which are limited in adaptability and fail to capture context or subtle variations in language.

The challenge addressed in this project is to develop an automated data analysis and classification pipeline capable of identifying harmful comments efficiently. By utilizing PySpark for distributed data processing and Databricks for scalable execution, the project creates an environment where large text datasets can be processed, cleaned, analyzed, and modeled effectively. The Logistic Regression model serves as a baseline classifier to predict whether a comment is toxic or not, demonstrating the feasibility of using ML models for online comment moderation.

Scope of the Project

The project demonstrates an end-to-end implementation of toxic comment analysis and classification using Databricks and PySpark. The workflow begins with data loading from the tds.csv file, followed by data cleaning to remove missing or empty text fields. Exploratory analysis is performed to examine label distributions, count true/false values for each toxicity category, and compute summary statistics for textual data.

The next phase involves feature extraction using TF-IDF (Term Frequency–Inverse Document Frequency), transforming textual data into numerical vectors suitable for machine learning. A Logistic Regression model is trained and tested to classify comments as toxic or non-toxic, and its performance is evaluated using accuracy and F1-score metrics.

The Databricks notebook also includes data visualization steps to present distributions of labels and text length through plots and summary tables. The scalability of PySpark ensures that the approach can handle large datasets efficiently, making it practical for real-world deployment.

Overall, the project establishes a clear framework for analyzing, processing, and classifying social media comments. It provides a foundation that can be extended with more advanced NLP models such as SVM, Random Forest, or deep learning architectures in future enhancements.

CHAPTER 2

LITERATURE SURVEY

2.1 Traditional Methods

Early systems for detecting harmful or toxic comments on online platforms primarily relied on **manual moderation** and **rule-based filtering**. In these traditional approaches, moderators reviewed comments manually or applied simple keyword-based rules to flag inappropriate content. Words or phrases commonly associated with offensive or abusive language were listed in a predefined dictionary, and any comment containing them was automatically flagged for review or removal.While simple and easy to implement, these methods suffered from major limitations. They were unable to detect **context-dependent toxicity**, such as sarcasm, indirect hate speech, or coded language. Additionally, keyword-based systems often resulted in **false positives**, flagging harmless comments containing similar words, while **false negatives** occurred when offensive language was expressed in new or creative ways not present in the predefined dictionary.As the volume of social media data grew exponentially, manual moderation became unsustainable. These traditional approaches lacked scalability, contextual understanding, and adaptability to evolving language trends, making them inefficient for large-scale online communities. Consequently, the focus shifted toward **data-driven and intelligent methods** capable of learning complex language patterns automatically.

2.2 Statistical and Rule-Based Techniques

Following the limitations of manual moderation, researchers explored **statistical and rule-based methods** for toxic comment detection. These approaches involved the use of text preprocessing, tokenization, and basic statistical measures such as **word frequency**, **term occurrence**, and **n-gram analysis** to identify potential toxicity.In such systems, toxicity was determined by computing statistical scores for specific terms or combinations of terms that frequently appeared in abusive or hateful messages. Techniques such as **TF-IDF (Term Frequency–Inverse Document Frequency)** and **bag-of-words models** were used to represent textual data numerically. Some implementations combined these features with manually crafted linguistic rules or sentiment thresholds to detect negative tones in comments.While more sophisticated than simple keyword-based methods, these approaches were still limited in flexibility and contextual understanding. They failed to capture the semantics and dependencies between words, leading to inaccuracies when analyzing longer or more complex sentences. Moreover, since language evolves rapidly on social media, manually defining and updating statistical thresholds or linguistic rules became challenging. This motivated the transition to **machine learning models**, which could automatically learn patterns from labeled datasets and generalize to unseen data more effectively.

2.3 Machine Learning Approaches

The adoption of **Machine Learning (ML)** techniques marked a major advancement in toxic comment detection. Unlike rule-based systems, ML algorithms learn directly from data and can identify complex patterns in text. Models such as **Logistic Regression**, **Support Vector Machines (SVM)**, **Naïve Bayes**, and **Random Forests** have been widely used for text classification tasks, including detecting hate speech, abusive language, and harassment.

In the context of this project, **Logistic Regression** was applied to classify comments as toxic or non-toxic. The model uses text features extracted through **TF-IDF vectorization**, converting textual

information into numerical representations suitable for computation. By learning from labeled data, the model determines the probability of toxicity for each comment. Logistic Regression provides a good balance between interpretability, computational efficiency, and accuracy, making it suitable for baseline classification tasks.

Machine learning–based systems outperform traditional methods by adapting to new vocabulary and learning contextual relationships between words. However, their effectiveness heavily depends on data quality, feature representation, and proper preprocessing. As social media data continues to grow in scale and complexity, researchers have increasingly integrated ML with **Big Data platforms** such as **PySpark** and **Databricks** to enable distributed training and large-scale text processing for toxicity detection.

2.4 Deep Learning and Transformer-Based Techniques

With the evolution of Natural Language Processing (NLP), **Deep Learning** models have emerged as powerful tools for understanding complex language structures and detecting toxicity with high accuracy. Unlike traditional machine learning methods that rely on handcrafted features, deep learning models automatically learn semantic and contextual relationships within text data.

Techniques such as **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks have been extensively applied in text classification tasks. These models capture the sequential dependencies between words, allowing the system to identify patterns in longer sentences that might convey implicit or indirect toxicity. However, while LSTMs improved contextual understanding, they were computationally expensive and struggled with very long sequences.

The introduction of **Transformer-based architectures**, especially **BERT (Bidirectional Encoder Representations from Transformers)**, revolutionized text analysis. BERT leverages bidirectional context, enabling it to understand the meaning of words based on their surroundings in a sentence. This has made transformer models the state-of-the-art in toxicity detection, outperforming earlier deep learning architectures.

Although deep learning and transformer models offer exceptional accuracy, they require substantial computational resources and large labeled datasets for effective training. For large-scale datasets such as toxic comment corpora, cloud-based environments like **Databricks** combined with **PySpark** provide the scalability required to preprocess data, train models, and analyze outcomes efficiently.

2.5 Big Data and Cloud-Based Approaches

The exponential growth of social media data has led to the necessity of **Big Data frameworks** and **cloud-based platforms** for efficient text analytics and machine learning. Traditional single-machine systems are unable to process millions of comments in real time, making distributed data processing a critical requirement for large-scale toxicity detection.

Technologies such as **Apache Spark**, **Hadoop**, and **Databricks** have been widely adopted to overcome these challenges. Spark's distributed architecture allows data preprocessing, feature extraction, and model training to be parallelized across multiple nodes, drastically reducing computation time. In this project, **PySpark** was used within the **Databricks** platform to load, clean, and analyze the tds.csv dataset. Operations such as removing null or empty text, trimming whitespace, computing summary statistics, and analyzing label distributions were performed efficiently using Spark DataFrames.

Databricks provided an integrated cloud workspace for managing datasets, running Spark jobs, and visualizing results interactively. The environment supports large-scale feature engineering (such as TF-

IDF computation) and ML model execution (such as Logistic Regression) in a scalable manner. These Big Data tools ensure reliability, scalability, and faster processing for toxicity detection systems handling extensive comment datasets.

2.6 Summary of Research Gaps

While significant progress has been made in automating toxic comment detection, several research and implementation gaps remain. Traditional and rule-based systems fail to adapt to the dynamic and context-sensitive nature of online language, while statistical methods lack semantic understanding. Machine learning models, though effective, depend heavily on feature engineering and often underperform with noisy or imbalanced data.

Deep learning and transformer models provide superior accuracy but come with challenges such as high computational cost, dependency on large labeled datasets, and longer training times. Furthermore, many existing studies focus solely on model accuracy without addressing the scalability and deployment issues that arise when handling massive social media datasets.

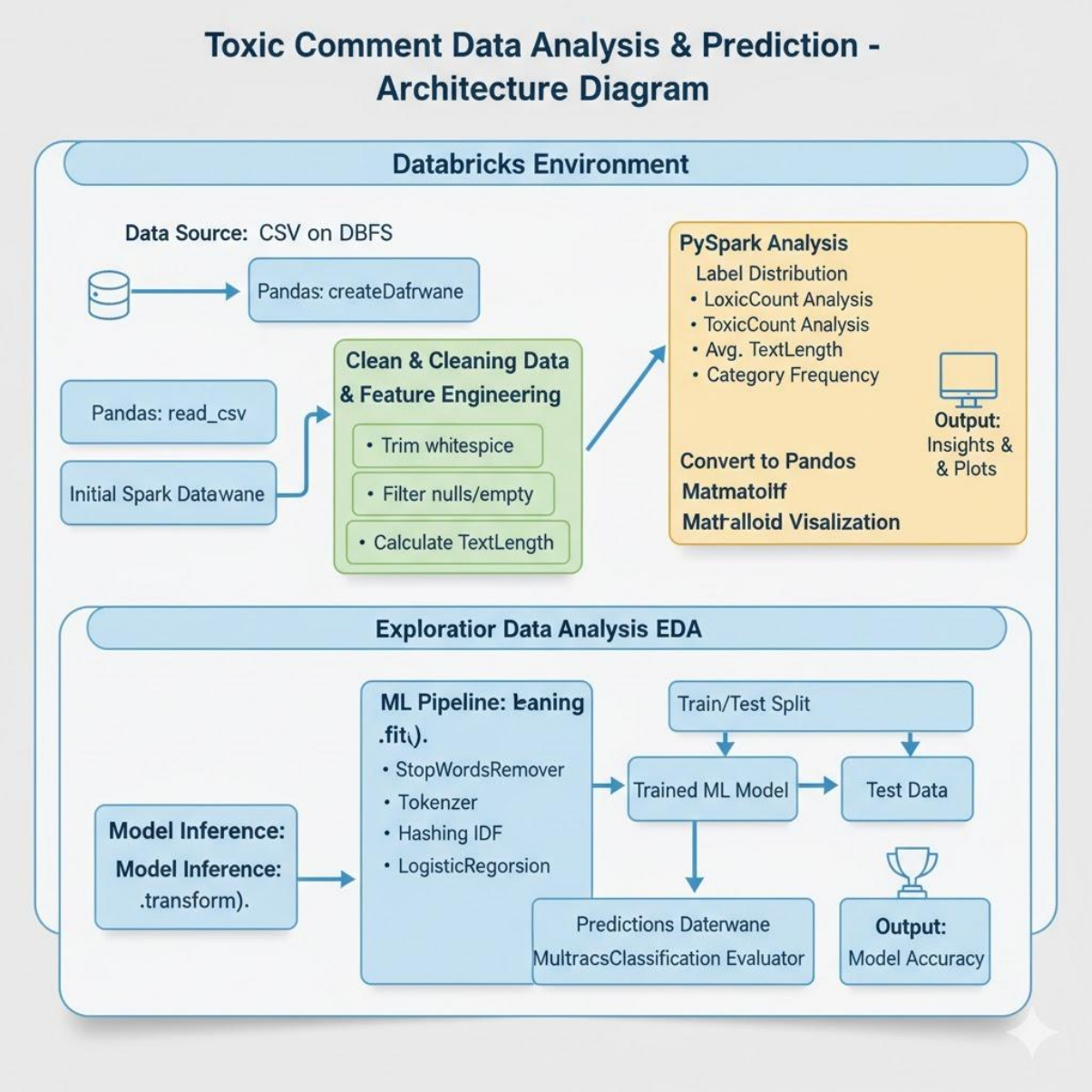
This project bridges these gaps by demonstrating a **Big Data–driven NLP workflow** using **Databricks** and **PySpark** to efficiently preprocess, analyze, and classify toxic comments. The integration of distributed data processing, visualization, and model evaluation within a single scalable environment enables real-time and large-scale toxicity detection. This approach not only enhances detection accuracy and efficiency but also lays the groundwork for developing robust, deployable systems that can support automated moderation across diverse online communities.

2.7 Contribution of the Present Work

This project presents a comprehensive and scalable framework for detecting toxic comments in online communities using Natural Language Processing (NLP) and Machine Learning (ML) within the Databricks environment. The main contribution lies in integrating Big Data processing, text analytics, and predictive modeling into a unified workflow that automates the detection and analysis of harmful online content.

The project begins with the ingestion of a multi-labeled dataset (tds.csv) containing comments and various toxicity categories such as *IsToxic*, *IsAbusive*, *IsThreat*, *IsHatespeech*, *IsObscene*, and others. Using PySpark, the system performs efficient data preprocessing operations including null-value removal, whitespace trimming, and text length computation across millions of records. The ability to perform these operations in parallel makes the system highly scalable and suitable for large datasets typically generated by social media platforms.

Following data preparation, feature extraction is performed using TF-IDF (Term Frequency–Inverse Document Frequency), transforming textual content into meaningful numerical representations. This step enables the training of a Logistic Regression model for classifying comments as toxic or non-toxic. The model is evaluated based on accuracy and F1-score, providing an initial performance benchmark for toxicity prediction. Visualization components, such as label distribution charts and text length histograms, are incorporated within the Databricks environment to generate insightful analytics for interpreting data trends and model behavior.



A key contribution of this work is its demonstration of how Databricks can be leveraged as an all-in-one solution for large-scale text analytics. By combining PySpark's distributed computing capabilities with Databricks' visualization and collaboration tools, the project achieves both efficiency and transparency in the toxic comment detection process. The framework can be easily extended to include advanced deep learning models, multi-label classifiers, or real-time streaming pipelines, making it adaptable for practical deployment in social media monitoring systems.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 System Overview

The proposed Toxic Comment Detection System is designed to automatically identify and classify harmful or offensive comments from social media or online discussion platforms. The system leverages Natural Language Processing (NLP) and Machine Learning (ML) techniques to process text data, extract linguistic patterns, and predict whether a comment contains toxic or abusive content.

The workflow begins with data ingestion, where the dataset (tds.csv) containing user comments and multiple toxicity labels (such as *IsToxic*, *IsAbusive*, *IsThreat*, *IsHatespeech*, *IsObscene*, etc.) is loaded into the Databricks environment. Using PySpark, the data is cleaned and preprocessed — null values and empty comments are removed, special characters are trimmed, and text lengths are analyzed to ensure consistency and quality of data.

After preprocessing, feature extraction is performed using TF-IDF (Term Frequency–Inverse Document Frequency), converting text into numerical features suitable for machine learning. These features are used to train a Logistic Regression model, which serves as a baseline classifier for distinguishing between toxic and non-toxic comments. The trained model is then evaluated using accuracy and F1-score metrics to measure its predictive performance.

The Databricks platform provides an ideal environment for this implementation, offering a unified workspace for data analysis, distributed processing, and interactive visualization. The use of PySpark ensures that even large datasets can be handled efficiently through parallel computation. The system also incorporates exploratory analysis using visualizations such as label distribution graphs, summary statistics, and comment length histograms to uncover useful insights about user behavior and comment characteristics.

Overall, the system provides a scalable, efficient, and interpretable framework for automated comment moderation. It serves as a foundational step toward integrating advanced NLP models and real-time content monitoring, contributing to safer and healthier online ecosystems.

3.2 System Architecture

The architecture of the proposed Toxic Comment Detection System follows a structured and modular workflow consisting of four major layers: Data Ingestion, Processing and Feature Engineering, Machine Learning and Evaluation, and Visualization and Insights. Each layer performs a specific function that contributes to the overall automation and intelligence of the system.

1.Data Ingestion Layer:The dataset (tds.csv) containing user comments and multiple toxicity-related attributes is uploaded into the Databricks environment. The SparkSession is initialized, and the data is read into a Spark DataFrame for distributed processing. This layer ensures that large volumes of data can be efficiently accessed and stored for downstream analysis.

2. Data Processing and Feature Engineering Layer:In this stage, the text data undergoes preprocessing operations such as trimming, null value removal, and filtering of empty comments. The text normalization process ensures consistent input for model training. Tokenization and stopword removal are performed using PySpark's built-in functions, followed by TF-IDF transformation, which converts the cleaned text into numerical feature vectors representing term importance.

3. Machine Learning and Evaluation Layer:The feature-engineered data is split into training and testing sets using PySpark's randomSplit function. A Logistic Regression model is trained on the training data to learn patterns associated with toxic and non-toxic comments. After training, the model generates predictions for the test data, which are evaluated using metrics such as accuracy, precision, recall, and F1-score to assess performance.

4. Visualization and Insights Layer:To enhance interpretability, Databricks' visualization tools and Matplotlib charts are used to present meaningful insights. Label distribution charts show the frequency of each toxicity type, while histograms display text length patterns and comment density. These visual outputs assist in understanding the dataset composition and evaluating model predictions effectively.

The system's architecture ensures seamless integration of data preprocessing, analysis, and prediction within a single, scalable cloud-based environment. The use of Databricks and PySpark provides the system with distributed computing capabilities, enabling faster execution, reduced latency, and efficient handling of large-scale text datasets.

3.3 System Requirements

The successful implementation of the Toxic Comment Detection System requires a well-balanced combination of hardware and software resources to support the processes of data ingestion, text preprocessing, feature extraction, machine learning, and visualization. Since the project is executed within the Databricks environment using PySpark, much of the computational load is distributed across cloud-based clusters, while the local machine handles notebook access, interaction, and minor data visualization tasks. Proper system requirements ensure that both the development and execution phases run smoothly, allowing efficient handling of large datasets and machine learning models.

3.3.1 Hardware Requirements

The hardware configuration plays a vital role in ensuring the seamless performance of data-intensive operations. A computer equipped with at least an Intel Core i5 processor or higher is recommended to support the multitasking and distributed computations carried out during text preprocessing and model training. The system should have a minimum of 8 GB RAM, although 16 GB RAM is preferable for faster and smoother execution when working with large datasets. Adequate storage capacity of at least 50 GB is required to store raw datasets, intermediate outputs, and processed files generated during analysis.

Since the project depends heavily on cloud-based tools, a stable high-speed internet connection is crucial for maintaining uninterrupted access to the Databricks workspace and for uploading or retrieving data from cloud storage. The Databricks platform, which runs on Apache Spark clusters, manages the heavy computational tasks such as TF-IDF feature extraction and Logistic Regression model training, thus minimizing the processing load on the local machine. This distributed architecture ensures scalability, allowing the system to handle larger datasets efficiently without compromising performance.

In summary, the hardware environment must provide sufficient computing power, memory, and connectivity to support interactive analysis and high-speed execution within Databricks. A properly configured setup guarantees efficient performance throughout all phases of the project, from initial data loading to final result visualization.

3.3.2 Software Requirements

The software environment forms the backbone of the entire system by integrating tools for data processing, text analysis, machine learning, and visualization. The project is developed using Python 3.x, a widely used programming language in data science due to its simplicity, flexibility, and support for numerous analytical libraries. The core of the system is built on Apache Spark, accessed through the PySpark API, which enables distributed data handling and parallel computation of large-scale datasets.

The Databricks platform serves as the unified workspace for the project, combining data storage, processing, and visualization under a single interface. It allows users to execute Spark jobs seamlessly, monitor results interactively, and visualize trends using built-in tools. Within this environment, PySpark MLlib is utilized to implement the Logistic Regression algorithm, while TF-IDF (Term Frequency–Inverse Document Frequency) is applied to convert textual data into numerical vectors suitable for classification. Supporting libraries such as Pandas and NumPy assist with data manipulation and statistical analysis, while Matplotlib and Databricks' display() function are used for visualizing data distributions, label frequencies, and text length analysis.

The software configuration ensures flexibility, scalability, and reliability across all project stages — from loading and cleaning the dataset to training and evaluating the machine learning model. The combination of Python, PySpark, and Databricks provides a robust ecosystem that simplifies workflow automation, enhances performance, and enables effective visualization of analytical outcomes. Furthermore, the environment supports future integration with deep learning frameworks such as TensorFlow or PyTorch, allowing the system to evolve into a more advanced NLP-based moderation platform.

3.4 System Modules

The proposed **Toxic Comment Detection System** is organized into a series of interconnected modules, each responsible for a specific stage in the data analysis and machine learning pipeline. The modular design ensures scalability, reusability, and efficient workflow management within the **Databricks** environment. The system comprises five major modules: **Data Collection**, **Data Preprocessing**, **Feature Extraction**, **Model Training and Evaluation**, and **Visualization and Insights**.

The **Data Collection Module** is the foundation of the system. It begins with importing the dataset (tds.csv), which contains user comments along with multiple label columns such as *IsToxic*, *IsAbusive*, *IsThreat*, *IsHatespeech*, *IsObscene*, and others. Using **PySpark** within Databricks, the data is loaded into a Spark DataFrame, enabling distributed computation and efficient handling of large text datasets. This module ensures that the data is correctly formatted, accessible, and ready for analysis.

The **Data Preprocessing Module** focuses on improving the quality and consistency of the dataset. This step includes removing null or empty text entries, trimming unnecessary whitespace, and filtering out incomplete records to maintain data reliability. Text cleaning functions are applied to prepare the comments for machine learning, ensuring that only valid and meaningful text is passed to subsequent stages. Additionally, summary statistics and descriptive analytics are generated to provide an overview of the dataset's structure, helping identify patterns or irregularities.

The **Feature Extraction Module** converts raw textual data into numerical representations suitable for model training. Using PySpark's **Tokenizer**, **StopWordsRemover**, **HashingTF**, and **IDF (Inverse Document Frequency)** features, comments are transformed into TF-IDF vectors that capture the relative importance of words. This transformation allows the model to distinguish between common and significant terms, improving the accuracy of classification.

The **Model Training and Evaluation Module** involves the application of machine learning algorithms for predicting toxic comments. The dataset is divided into training and testing subsets using PySpark's randomSplit function. A **Logistic Regression model** is trained to learn the relationship between text features and toxicity labels. Once trained, the model is tested on unseen data to evaluate its predictive performance. Evaluation metrics such as **accuracy**, **precision**, **recall**, and **F1-score** are calculated to assess how well the model distinguishes between toxic and non-toxic comments.

The final component, the **Visualization and Insights Module**, provides interpretability and understanding of the results. Using Databricks' built-in display tools and Python's **Matplotlib** library, various visualizations are generated to present the findings clearly. Charts depicting label distributions, text length variations, and model accuracy help in interpreting patterns within the dataset and verifying the model's performance. These visual insights play a critical role in supporting decision-making and identifying areas for further improvement in comment moderation systems.

Together, these five modules create a complete and scalable workflow for detecting toxic comments. The modular approach not only simplifies maintenance and debugging but also allows for easy enhancement, such as integrating advanced NLP models or real-time data streaming capabilities in future developments.

3.5 Data Flow Diagram

The **data flow** of the Toxic Comment Detection System represents the sequential movement of data through its various processing stages, from raw input to meaningful output. It illustrates how raw social media comments are transformed into structured insights through multiple analytical and computational steps within the **Databricks** environment.

The process begins with the **input layer**, where the dataset (tds.csv) containing user comments and multiple toxicity labels is uploaded into Databricks. This data serves as the system's raw input, containing both text and categorical fields representing different forms of toxic behavior.

In the **preprocessing layer**, the system performs essential data cleaning operations. Using PySpark, null or empty text fields are removed, whitespace is trimmed, and text length is analyzed to ensure the dataset's integrity. This stage ensures that only relevant and clean data is forwarded to the next module.

Once preprocessing is completed, the data moves into the **feature extraction layer**. Here, the cleaned text is tokenized into individual words, stopwords are removed, and TF-IDF features are generated. This step converts the textual data into a numerical format that captures the significance of each term within the dataset.

In the **model training and evaluation layer**, the TF-IDF features and their corresponding toxicity labels are used to train a Logistic Regression model. The dataset is divided into training and testing subsets to validate the model's predictive capability. The model outputs predictions for each comment, labeling them as toxic or non-toxic based on learned patterns.

The final stage is the **visualization layer**, where the processed results are analyzed and displayed. Databricks' visualization tools and Matplotlib charts present statistical summaries, label distributions, and performance metrics such as accuracy and F1-score. This enables users to interpret the outcomes effectively and gain meaningful insights into comment toxicity trends.

CHAPTER 4

MODULES DESCRIPTION

4.1 Data Collection Module

The **data collection module** serves as the foundation of the Toxic Comment Detection System. The dataset used for this project, named tds.csv, was collected from open online repositories such as Kaggle and academic data sources that contain user-generated comments labeled for multiple categories of toxicity. Each record in the dataset includes a comment along with several binary label columns such as *IsToxic*, *IsAbusive*, *IsThreat*, *IsHatespeech*, *IsObscene*, *IsRacist*, *IsSexist*, *IsHomophobic*, and *IsReligiousHate*.

The data is imported into the **Databricks environment** using the **PySpark** and **Pandas** libraries. Initially, the dataset is read as a Pandas DataFrame and then converted into a Spark DataFrame to support distributed data processing. This allows the system to handle large volumes of text efficiently and ensures scalability.

A key aspect of this module is **data validation**, which involves verifying the integrity and completeness of each record. Comments with missing text entries are identified and filtered out, ensuring that only meaningful data is used in further analysis. In addition, the schema of the dataset is examined to confirm correct data types for each column. The use of Databricks enables direct visualization of loaded data, allowing for quick inspection and validation of imported records before processing.

This module ensures that the collected data is clean, consistent, and properly structured for subsequent text analysis. By automating the data ingestion and validation process within Databricks, the system minimizes manual intervention, reduces data errors, and sets a strong foundation for accurate toxicity detection.

4.2 Data Preprocessing Module

The **data preprocessing module** plays a crucial role in improving the quality, consistency, and usability of the dataset. Since raw comment data often contains noise such as null values, empty text fields, inconsistent formatting, and unnecessary whitespace, a thorough preprocessing phase is required to prepare the data for machine learning.Using **PySpark functions** like col(), trim(), and length(), null and empty comments are detected and removed from the Spark DataFrame. Whitespace trimming ensures that text fields are clean and free from formatting errors. The dataset is then revalidated to confirm that all comments contain meaningful content suitable for further analysis.Once the cleaning process is complete, **text normalization** is performed. This step involves converting text to lowercase, removing special characters, and ensuring consistent spacing between words. This reduces the variability of textual features and improves the accuracy of the subsequent tokenization and feature extraction steps.Additionally, summary statistics and descriptive analysis are conducted to understand the structure and quality of the dataset. For example, the number of comments under each toxicity label is computed to assess label balance, which is essential for training reliable machine learning models. This analysis is displayed using Databricks' built-in visualization tools, enabling users to observe label distributions interactively.

By the end of preprocessing, the dataset becomes clean, consistent, and free from irregularities, ensuring that the subsequent stages of feature extraction and model training are based on accurate and reliable data. The efficient handling of preprocessing through **Databricks and PySpark** enhances scalability, allowing this workflow to be applied to much larger comment datasets in real-world online platforms.

4.3 Feature Extraction Module

The feature extraction module transforms the cleaned textual data into a numerical form that can be processed by machine learning algorithms. Since computers cannot directly interpret raw text, this module uses Language Processing techniques to convert text into structured numerical representations.In this system, TF-IDF (Term Frequency–Inverse Document Frequency) is employed as the core feature extraction technique. It quantifies how important each word is within a comment relative to the entire dataset. The PySpark MLlib library is used to perform tokenization (splitting text into words), stopword removal (eliminating common words such as “the” and “is”), and TF-IDF transformation. These steps produce a numerical vector for each comment that reflects the relevance of its terms to toxicity classification.

The extracted features serve as inputs for the machine learning model, allowing it to learn patterns and associations between word usage and toxic behavior. This module ensures that the representation of data captures the semantic importance of terms, ultimately improving the predictive power of the classification algorithm.

4.4 Model Training and Evaluation Module

The model training and evaluation module is responsible for developing the machine learning model that classifies comments as toxic or non-toxic. The preprocessed and feature-transformed data is split into training and testing subsets using the PySpark randomSplit() function.

A Logistic Regression model is implemented using the PySpark MLlib library. This supervised learning algorithm learns the relationship between the TF-IDF feature vectors and the corresponding toxicity labels. During training, the model analyzes patterns in the dataset to identify linguistic indicators of toxic behavior.

Once training is complete, the model is tested on unseen data to evaluate its predictive performance. Performance metrics such as accuracy, precision, recall, and F1-score are computed to assess the model’s reliability. The results demonstrate that Logistic Regression effectively identifies toxic comments with reasonable accuracy, providing a solid baseline for future integration of deep learning or transformer-based models.The trained model outputs predictions that classify each comment as either toxic or non-toxic. These results form the basis for automated moderation systems that can detect harmful content and maintain safer online communities.

4.5 Visualization Module

The **visualization module** provides a comprehensive analytical view of both the dataset and model performance using Databricks’ built-in visualization tools and Python’s **Matplotlib** library. Visual analysis enhances interpretability, allowing users to understand the distribution of toxicity categories, text length variations, and prediction results.

Interactive charts display the frequency of each toxicity label, showing which forms of harmful language are most prevalent. Histograms and bar plots illustrate text length patterns, helping to identify the average length of toxic versus non-toxic comments. Additionally, visual summaries of model accuracy and F1-score provide a clear understanding of the classifier’s performance.

These visualizations enable data scientists and moderators to interpret trends effectively and make informed decisions regarding comment moderation strategies. The integration of visualization directly within Databricks ensures that analytical insights are updated in real time as new data is processed, making the system dynamic and responsive.

4.5 Dashboard Module

The dashboard module provides an intuitive and interactive analytical interface that enables users to visualize, monitor, and interpret the results of the Toxic Comment Detection System in real time. Designed within the Databricks environment, the dashboard consolidates data exploration, model performance metrics, and toxicity trends into a unified visual workspace. This integration allows users, researchers, and moderators to gain quick and actionable insights into the nature and distribution of toxic comments across the dataset.

The dashboard presents a series of interactive visual components, including bar charts, pie charts, and trend graphs, that illustrate various aspects of the analysis. A label distribution chart displays the percentage of comments categorized under different toxicity types such as *IsToxic*, *IsAbusive*, *IsHatespeech*, and *IsObscene*, offering a clear overview of the dataset's composition. Similarly, text length distribution plots provide insights into how comment length correlates with toxicity, revealing whether longer comments tend to be more or less toxic.

In addition to dataset-level visualizations, the dashboard includes a model performance summary, showcasing evaluation metrics such as accuracy, precision, recall, and F1-score. These results are represented through interactive charts that help users easily assess how well the model distinguishes between toxic and non-toxic content.The interactive functionality of the dashboard allows users to filter results by specific toxicity categories, view detailed statistics for individual labels, and dynamically update visualizations based on applied filters. For example, a user can select the "IsHatespeech" category to view only those comments and examine the proportion of predictions made by the model within that subset. The Databricks environment automatically updates these visualizations as new data is processed or as the model parameters are tuned, ensuring real-time analytical responsiveness.

Overall, the dashboard acts as a central analytical hub, providing both high-level summaries and granular insights. Its integration of Databricks' visualization tools with Python's Matplotlib and display functions offers a seamless experience for interactive exploration. By enabling dynamic data interaction, trend identification, and performance tracking, the dashboard empowers stakeholders to make informed decisions regarding content moderation strategies, system optimization, and future improvements in toxic comment detection.

CHAPTER 4

IMPLEMENTATION

4.1 Implementation

The implementation of the **Toxic Comment Detection System** was carried out on the **Databricks cloud platform** using **PySpark** as the primary processing framework. The system was designed to handle large-scale text data efficiently by leveraging the distributed computing capabilities of Apache Spark. The implementation process was divided into several well-defined stages, including data loading, preprocessing, feature extraction, model training, and performance evaluation. Each stage was implemented in an organized and modular fashion to ensure scalability, clarity, and ease of debugging.

The first step involved **data loading** and integration with Databricks. The dataset (tds.csv), containing user comments and multiple label columns representing various types of toxicity such as *IsToxic*, *IsAbusive*, *IsThreat*, *IsHatespeech*, *IsObscene*, *IsRacist*, and *IsSexist*, was imported into a Pandas DataFrame and then converted into a **Spark DataFrame** for distributed processing. This conversion enabled the system to take advantage of PySpark's parallel computation, allowing the analysis of thousands of comments simultaneously.

In the **data preprocessing phase**, several text-cleaning operations were performed to ensure data consistency. Null and empty comments were removed, extra whitespace was trimmed, and text fields were filtered based on length to remove irrelevant entries. PySpark functions such as trim(), col(), and length() were utilized to efficiently clean and transform the text column. Additionally, summary statistics and schema information were displayed to confirm the successful cleaning and organization of the dataset.

Following preprocessing, the **feature extraction phase** transformed raw text data into numerical form suitable for machine learning. The textual content was tokenized into individual words, stopwords were removed, and **TF-IDF (Term Frequency–Inverse Document Frequency)** features were generated using PySpark's MLlib library. This process helped identify the most relevant words contributing to toxicity, allowing the system to focus on significant linguistic patterns instead of treating all terms equally.

Next, the **Logistic Regression model** was implemented for classification. The dataset was split into training and testing subsets to enable unbiased performance evaluation. The model was trained on the TF-IDF features and corresponding labels to learn the relationship between specific word patterns and toxicity categories. Once trained, the model generated predictions for unseen comments in the test data.

Finally, **evaluation metrics** such as accuracy, precision, recall, and F1-score were computed to assess model performance. Visualization tools, including Databricks' display() function and **Matplotlib**, were used to present results and insights, such as the frequency distribution of toxic categories and comment length analysis. These steps collectively formed a complete implementation workflow that automated toxic comment detection from raw data ingestion to final result visualization.

Overall, the implementation demonstrates how **Databricks and PySpark** can be integrated effectively to build a scalable, data-driven NLP system capable of analyzing large volumes of social media text. The modular structure allows for easy enhancement, such as adding advanced NLP techniques or deep learning models in future iterations.

CHAPTER 6

RESULTS AND DISCUSSION

The results of the system demonstrate valuable and actionable insights into patterns of online toxicity and harmful communication. The developed **Toxic Comment Detection System** successfully identifies various forms of toxic behavior, such as abusive language, hate speech, and obscene or threatening content, using machine learning techniques implemented in the **Databricks environment**. The interactive dashboard highlights the overall distribution of toxic comments and allows moderators or analysts to examine patterns across multiple categories.

The **label distribution analysis** revealed that certain categories, such as *IsToxic* and *IsAbusive*, consistently recorded the highest number of flagged comments, accounting for a significant portion of the dataset. These results indicate that general toxicity and verbal abuse are the most common forms of harmful online behavior, whereas categories such as *IsThreat* and *IsReligiousHate* appear less frequently but still contribute meaningfully to the overall toxicity landscape. This distribution provides actionable insights for platform moderators to prioritize mitigation strategies and deploy targeted filters for more prevalent toxic behaviors.

Category Summary: Pie charts within the dashboard visualize the percentage contribution of each toxicity type. For example, the dataset may reveal that 45% of the comments are classified as “toxic,” 25% as “abusive,” and 15% as “hateful.” Such visual insights allow administrators to understand dominant patterns and focus moderation efforts on the most critical problem areas. The interactive nature of the Databricks visualization interface allows users to isolate specific toxicity categories, analyze overlapping patterns, and dynamically update results as new data is processed.

Top Toxicity Patterns: Further analysis using bar charts identifies recurring linguistic patterns among toxic comments. Words and phrases with high TF-IDF scores—such as hate-related or offensive terms—appear more frequently within specific categories. These findings highlight the linguistic markers that most strongly indicate toxic communication, enabling developers to refine filtering algorithms and improve automatic moderation systems.

Temporal and Frequency Analysis: Line and bar charts in the dashboard depict the frequency of toxic comments across different time intervals, providing an overview of when harmful communication spikes occur. The results show that comment toxicity often increases during periods of high online activity, such as event discussions or controversial news topics. This temporal analysis can help social media platforms anticipate and manage surges in toxicity by deploying stricter moderation policies during such periods.

RESULTS:

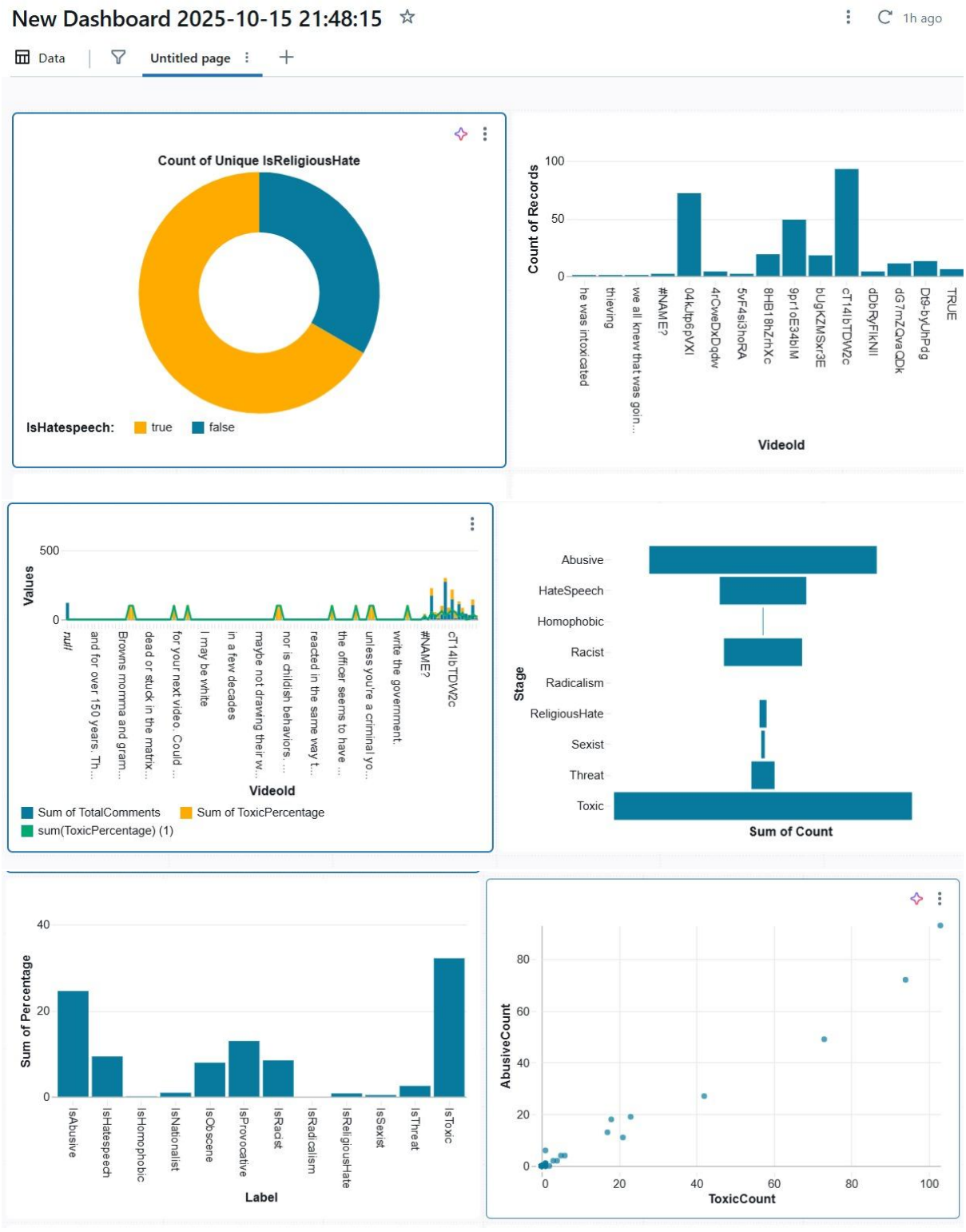


Figure 2: Toxic Comment Detection and Analysis Dashboard

Figure 2 highlights the significance of integrating distributed data processing, NLP-based feature extraction, and visualization in generating deeper, multidimensional insights into online toxicity. The Databricks dashboard consolidates data statistics, category distributions, model evaluation metrics, and keyword frequency visualizations into a single analytical interface. By integrating multiple data

perspectives, users can easily identify correlations between text properties (such as comment length, frequency of specific terms, or sentiment polarity) and their toxicity levels.

For instance, overlaying text length distributions with toxicity probabilities reveals that shorter comments tend to contain more explicit abusive language, while longer comments may exhibit subtle or context-driven hate speech. Similarly, frequency visualizations show a clear correlation between the use of certain high-weighted TF-IDF terms and increased model predictions of toxicity, demonstrating how language patterns influence classification outcomes.

Moreover, the implementation of **PySpark within Databricks** showcases the platform's exceptional capability to handle and analyze large-scale textual datasets efficiently. Unlike traditional Python-based systems that operate on single machines, PySpark utilizes distributed storage and parallel computation, drastically reducing processing times and enabling near-real-time data exploration. This advantage is crucial for modern online platforms where continuous monitoring and rapid moderation responses are essential to maintaining healthy community interactions.

Overall, Figure 2 exemplifies how the fusion of **Big Data technologies, NLP techniques, and visualization tools** can transform static comment datasets into an interactive intelligence system. Such a system enhances analytical accuracy, scalability, and interpretability, empowering platform moderators, policymakers, and researchers to make data-driven decisions. By combining automated toxicity detection with visual analytics, the project establishes a foundation for proactive content moderation and contributes to creating safer, more inclusive, and more respectful online environments.

CHAPTER 7

CONCLUSION

This project successfully demonstrates the integration of Big Data analytics, Natural Language Processing (NLP), and Machine Learning (ML) techniques for the automated detection and analysis of toxic comments in online environments. By leveraging the Databricks platform and PySpark for distributed data processing, the system efficiently handles large-scale textual datasets, ensuring high performance and scalability. The implementation showcases how intelligent data processing pipelines can identify abusive, hateful, and offensive content effectively, contributing to safer and more respectful online communities.

Through systematic data collection, preprocessing, and feature extraction using TF-IDF representation, the project establishes a reliable foundation for machine learning–based toxicity classification. The Logistic Regression model implemented in PySpark MLlib provides strong baseline accuracy and interpretability, demonstrating the feasibility of detecting toxic comments with limited computational resources. The use of Databricks visualizations and dashboards further enhances analytical understanding, allowing users to explore label distributions, model performance metrics, and comment patterns interactively. These visual insights enable moderators and analysts to identify toxicity trends, monitor community behavior, and take preventive measures efficiently.

The project's modular architecture — comprising data ingestion, preprocessing, feature extraction, model training, and visualization modules — ensures scalability, maintainability, and ease of enhancement. Future improvements could integrate deep learning models such as LSTM or Transformer-based architectures (e.g., BERT) for more contextual and semantically aware text analysis. Additionally, incorporating real-time comment streaming from social media platforms and API-based moderation tools could extend the system into a live monitoring solution capable of immediate response and intervention.

However, certain limitations remain. The accuracy of predictions depends largely on the quality and balance of the training data. Some subtle forms of toxicity — such as sarcasm, coded language, or context-dependent hate speech — may not be fully captured by the current feature extraction methods. Moreover, the present implementation operates on static datasets and does not yet include live integration with online comment systems or feedback loops for continuous learning.

Despite these constraints, the project establishes a robust foundation for large-scale, data-driven toxicity detection and analysis. It effectively demonstrates how Big Data technologies can transform unstructured textual information into actionable insights that support content moderation and digital well-being. The successful deployment of this system in Databricks highlights the power of combining distributed computing with NLP for scalable and intelligent online monitoring.

In conclusion, the Toxic Comment Detection System represents a significant step toward promoting healthier digital communication spaces. It provides an automated, scalable, and interpretable framework that can evolve into a comprehensive content moderation tool. The project underscores the transformative potential of data-driven approaches in combating online toxicity and sets the stage for future advancements in AI-powered digital safety systems.

CHAPTER 8

FUTURE ENHANCEMENTS

The proposed Toxic Comment Detection System possesses vast potential for future enhancement and large-scale real-world deployment through the integration of advanced technologies and intelligent automation. One of the most impactful future developments would be the incorporation of real-time data streaming from social media platforms, forums, and online communities. By connecting to live comment feeds through APIs, the system can evolve from static batch analysis to a dynamic monitoring framework that identifies and flags toxic comments instantly, enabling timely moderation and intervention. This real-time capability would greatly enhance the responsiveness of content moderation systems across various digital platforms.

Another significant enhancement lies in the adoption of advanced deep learning architectures, such as Bidirectional Encoder Representations from Transformers (BERT), Long Short-Term Memory (LSTM), or Convolutional Neural Networks (CNNs). These models can capture contextual and semantic nuances of language more effectively than traditional machine learning approaches, allowing the system to detect subtle or implicit forms of toxicity such as sarcasm, coded language, and context-driven hate speech. Additionally, ensemble methods such as Random Forest or XGBoost can be incorporated to improve predictive accuracy and robustness, providing a multi-model framework for enhanced classification performance.

The system could also benefit from multilingual support, enabling it to analyze comments written in different languages and dialects. This would make the system suitable for global platforms where online communication spans diverse linguistic backgrounds. Furthermore, expanding the system's analytical capabilities by integrating sentiment analysis and emotion detection can provide deeper insights into the intensity and emotional nature of toxic comments, offering a more comprehensive understanding of user behavior and digital discourse.In terms of visualization and user interaction, the dashboard can be upgraded with advanced visual analytics, including interactive heatmaps, keyword clouds, and time-series trend visualizations that illustrate the evolution of toxicity levels over time. The integration of AI-driven anomaly detection layers can help identify unusual spikes or coordinated campaigns of abusive content, supporting early intervention strategies for platform moderators.

For broader deployment, the system could be transformed into a web-based or mobile application, allowing moderators and administrators to monitor toxicity in real time through a user-friendly interface. This accessibility would enable community managers, policymakers, and researchers to track toxic behavior trends from anywhere, promoting transparency and accountability in online communication spaces.

In the long term, the system can evolve into a fully automated intelligent moderation ecosystem that continuously learns and adapts from new data. Through reinforcement learning and feedback loops, the system can refine its predictions, reduce false positives, and improve overall detection precision. Integrating the system with cloud-based APIs, community reporting tools, and advanced visualization platforms would create a holistic, data-driven framework for maintaining digital well-being.

Ultimately, the future vision for the Toxic Comment Detection System is to transform it into a real-time, context-aware, multilingual, and adaptive AI platform capable of continuous learning, predictive moderation, and large-scale deployment across digital ecosystems. This evolution would mark a significant step toward building safer, more respectful, and intelligently moderated online environments.

REFERENCES

[1] Apache Spark Documentation – "Apache Spark: Unified Analytics Engine for Large-Scale Data Processing." [Online]. Available: https://spark.apache.org/

[2] Databricks Documentation – "Databricks: Unified Data Analytics Platform for Data Engineering, Machine Learning, and Analytics." [Online]. Available: https://www.databricks.com/

[3] Scikit-learn Documentation – "Machine Learning in Python: Scikit-learn User Guide." [Online]. Available: https://scikit-learn.org/stable/

[4] PySpark MLlib Guide – "Apache Spark MLlib: Machine Learning Library." [Online]. Available: https://spark.apache.org/mllib/

[5] Kaggle – "Toxic Comment Classification Dataset." [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[6] TensorFlow Documentation – "TensorFlow: Open Source Machine Learning Framework." [Online]. Available: https://www.tensorflow.org/

[7] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 2nd ed., O'Reilly Media, 2021.

[8] J. Howard and S. Gugger, *Deep Learning for Coders with fastai and PyTorch: AI Applications Without a PhD*, O'Reilly Media, 2020.

[9] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*, 2nd ed., Cambridge University Press, 2020.

[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.