International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Duplicate Question Detection Using Natural Processing

Mrs.Rizwana[1], Korangi Praveen[2], Pasagada Manaswini[3], Sunkara Veera Lakshmi[4], Perabathula Jagadesh Satyanarayana[5], Mushini Vuttej Sri Venkata Rama Kumar[6]

[1]Assistant Professor, Cse Department, Vsm College Of Engineering, Ramachandrapuram, Dr.B.R.Ambhedkar Konaseema Dt, Andhra Pradesh - 533255
[2,3,4,5,6]Students, Cse Department, Vsm College Of Engineering, Ramachandrapuram, Dr.B.R.Ambhedkar Konaseema Dt, Andhra Pradesh - 533255

ABSTRACT

With the exponential increase in user-generated content on Q&A platforms such as Quora, Stack Overflow, and Reddit, detecting duplicate questions has become crucial for maintaining data quality and enhancing the user experience. Duplicate questions lead to redundancy, cluttered interfaces, and inefficient information retrieval. This project addresses this issue using a machine learning-based system built on Natural Language Processing (NLP) techniques.

The system incorporates a robust pipeline beginning with text preprocessing, which includes normalization techniques like tokenization, stop word removal, and stemming to clean and standardize the text. Then, Term Frequency-Inverse Document Frequency (TF-IDF) vectorization transforms the text into numerical feature vectors. Cosine similarity is applied to measure the semantic similarity between question pairs. These features are used to train classification models—Logistic Regression and Random Forest—to predict whether a pair of questions are duplicates. To overcome the problem of class imbalance in the dataset, resampling techniques such as up-sampling, under-sampling, and SMOTE (Synthetic Minority Over-sampling Technique) are employed. The proposed system effectively identifies semantically similar questions, helping to consolidate redundant content and streamline user interaction on digital platforms.

INTRODUCTION

In today's digital era, community-driven Q&A platforms such as Quora and Stack Overflow are inundated with user-submitted queries. While this content is invaluable, it often includes numerous duplicate questions phrased differently. These redundancies hamper the user experience by cluttering search results and making it more difficult for users to find relevant and consolidated answers. As the volume of submissions continues to grow, so does the need for automated mechanisms to identify and filter out such duplicates.

Traditional duplicate detection methods largely rely on keyword matching and manual moderation. Keyword-based systems fail to detect semantically similar questions that use different wording, while manual moderation is labour-intensive and cannot scale with the growth of user-generated content. This creates a pressing demand for more intelligent, automated approaches that can interpret the context and meaning of questions.

Natural Language Processing (NLP) and Machine Learning (ML) technologies offer promising solutions to this problem. NLP enables systems to process and analyse textual data by performing operations such as tokenization, stop word removal, and stemming. These preprocessing steps transform raw text into a normalized format that can be used effectively by ML algorithms.

Once the text is pre-processed, feature extraction techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) are applied to convert the text into numerical vectors. Cosine similarity is then used to quantify the semantic similarity between pairs of questions. These features are critical for building models that can detect duplicates accurately.

This project utilizes two supervised ML classifiers—Logistic Regression and Random Forest—to predict whether question pairs are duplicates. To ensure fair model training despite the dataset's class imbalance, the system incorporates balancing techniques like SMOTE, up-sampling, and under-sampling. The combination of NLP, ML, and data balancing strategies results in a reliable, scalable solution for duplicate question detection.

EXISTING SYSTEM

Existing systems primarily rely on keyword matching, regular expressions, or human reviewers to identify duplicate questions. These systems can only detect exact matches or closely related terms, failing to identify paraphrased or semantically similar content. Manual moderation, while more accurate, is time-consuming, costly, and inconsistent at scale. These limitations make traditional methods inadequate for modern platforms where understanding the contextual and semantic nature of questions is essential**.**

PROBLEM STATEMENT

Online Q&A platforms face an ongoing challenge of managing vast amounts of content submitted by users daily. Among these entries, many questions are duplicates—posed in different ways but reflecting the same intent. These redundancies clutter the interface, mislead users, and create confusion by spreading answers across multiple threads. Consequently, users often waste time navigating through similar content instead of finding consolidated and accurate answers efficiently.

Manual identification of such duplicates is not only time-consuming but also impractical, especially as the scale of data grows exponentially. Existing systems that rely on keyword matching often fail to catch questions that are semantically identical but syntactically different. Therefore, a machine learning-based solution that integrates Natural Language Processing techniques is required to automatically understand the context and semantics of user-submitted questions, thereby enabling accurate and efficient duplicate detection.

PROPOSED SYSTEM

The proposed system aims to automate the identification of duplicate questions by leveraging both Natural Language Processing (NLP) and supervised Machine Learning (ML) algorithms. The primary goal is to determine whether a pair of questions shares similar semantic meaning, regardless of the way they are worded. This is achieved by transforming raw text data into numerical features that machine learning models can understand and classify.

The process begins with comprehensive text preprocessing, including converting text to lowercase, removing punctuation and special characters, tokenizing words, removing stop words, and applying stemming. These steps ensure consistency and reduce textual noise. After preprocessing, Term Frequency-

Inverse Document Frequency (TF-IDF) is used to convert the cleaned text into numerical vectors. Cosine similarity is then calculated between these vectors to assess semantic closeness.

Two machine learning classifiers—Logistic Regression and Random Forest—are trained using these similarity-based features. Logistic Regression offers a fast and interpretable baseline, while Random Forest brings robustness through its ensemble structure. To address dataset imbalance, techniques like up-sampling, under-sampling, and SMOTE are applied, enabling the model to better learn from minority class samples and improve prediction accuracy. Overall, the system is designed to be efficient, scalable, and effective in reducing content redundancy on Q&A platforms.

IMPLEMENTED ALGORITHM

(a)TF-IDF Vectorizer:

TF-IDF (Term Frequency-Inverse Document Frequency) is a text vectorization technique that reflects the importance of a word in a document relative to a collection of documents. Words that occur frequently in a document but rarely across all documents receive higher weights, making them more meaningful for comparison. In this project, TF-IDF transforms the cleaned and pre-processed questions into numerical vectors that represent their content, enabling machine learning models to process and compare them.

(b) Cosine Similarity:

Cosine Similarity is a metric that evaluates how similar two vectors are by measuring the cosine of the angle between them. It ranges from -1 to 1, with 1 indicating perfect similarity. This technique is highly effective for sparse, high-dimensional datasets like TF-IDF vectors. In the system, cosine similarity is used to measure how closely related two questions are after being vectorized, serving as a key input feature for the classification models.

(c) Logistic Regression:

Logistic Regression is a linear classification algorithm commonly used for binary classification tasks. It estimates the probability that a given input belongs to a certain class using the logistic (sigmoid) function. It is simple, interpretable, and efficient, making it suitable for establishing a reliable baseline for duplicate detection in this project.

(d) Random Forest Classifier:

Random Forest is an ensemble learning algorithm that constructs multiple decision trees during training and merges their outputs to enhance prediction accuracy. It is robust to overfitting and handles both linear and non-linear relationships effectively. In this system, it is used to classify question pairs as duplicates or non-duplicates with improved performance and generalization.

(e) SMOTE, Up-sampling, and Under-sampling:

Class imbalance in the dataset can lead to biased models. To address this, various resampling techniques are applied. Up-sampling involves replicating instances from the minority class to match the majority class size. Under-sampling reduces the number of majority class samples. SMOTE (Synthetic Minority Over-sampling Technique) creates synthetic samples of the minority class based on existing data points. These techniques help balance the training data, enabling the models to better learn and generalize from both classes.

LIBRARIES USED

(a)Pandas:

Pandas is a powerful library for data manipulation and analysis. In this project, it is used to load and

handle large datasets of question pairs, filter out missing values, merge datasets, and create new columns for feature engineering. Its DataFrame structure simplifies operations like row filtering, aggregation, and grouping, which are essential for preparing the data before feeding it into the machine learning pipeline.

(b)NumPy:

NumPy is essential for handling numerical operations. It supports mathematical and logical operations on arrays and matrices. In the context of this project, it is used during data preprocessing, vector computations, and while handling TF-IDF matrix operations. NumPy also supports efficient handling of sparse matrix formats and is crucial for computing cosine similarities.

(c)Scikit-learn:

Scikit-learn is the primary machine learning library used in the project. It provides implementations for TF-IDF vectorization, cosine similarity, logistic regression, and random forest classification. Additionally, it includes tools for model evaluation (like accuracy, recall, F1-score), data splitting, and preprocessing. Its consistency and ease of use make it suitable for developing and testing various ML models.

(d)NLTK (Natural Language Toolkit):

NLTK is a fundamental toolkit for Natural Language Processing. It provides essential utilities such as tokenizers, stemmers (e.g., Snow-ball-Stemmer), and lists of stop words. These tools are used extensively in the text preprocessing phase to clean and normalize question data before feature extraction.

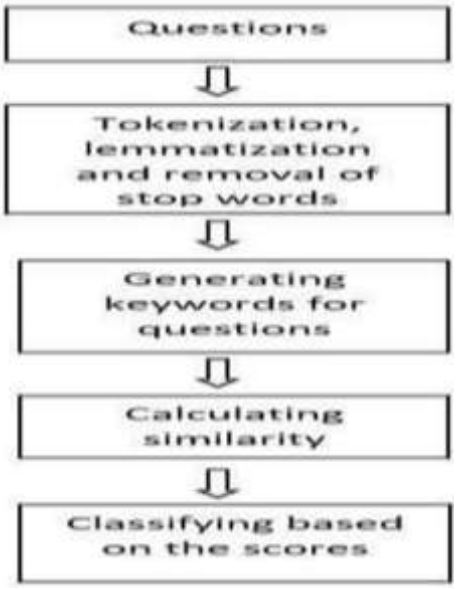
(e)Seaborn and Matplotlib:

These libraries are used for data visualization. Matplotlib provides basic plotting functions, while Seaborn builds on it with a more user-friendly interface and enhanced visualization styles. In the project, these libraries help plot class distribution, model performance metrics, and trends in similarity scores. Visualizations provide insights into data balance, model outputs, and clustering behaviour.

(f)re and string:

The re (regular expressions) library is used for cleaning raw text by removing unwanted characters like punctuation and digits. The string module helps with string formatting and basic operations like changing text to lowercase or removing specific characters. Both are used during text preprocessing to standardize question formatting for consistent processing.

SYSTEM ARCHITECTURE



1.Input:

The system begins with accepting pairs of questions from the dataset. These may come from Q&A platforms like Quora, and each row typically contains two text fields for analysis. This raw input serves as the foundation for subsequent processing.

2. Text Preprocessing:

This phase includes converting text to lowercase for consistency, removing punctuation and special characters using regular expressions, tokenizing text into words, eliminating stop-words (common words that add little meaning), and stemming to reduce words to their base forms. This helps normalize the input text and prepare it for vectorization.

3. Feature Extraction:

Cleaned text is converted into numerical representations using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. This helps quantify the significance of words in each question while reducing the impact of commonly used terms across the dataset.

4. Similarity Calculation:

Cosine similarity is calculated between the TF-IDF vectors of each question pair. This gives a similarity score between 0 (completely dissimilar) and 1 (identical), which is used as an important feature for model training.

5. Data Balancing:

To address the imbalance between duplicate and non-duplicate labels, techniques such as SMOTE, up-sampling, and under-sampling are applied. These methods improve the model's ability to learn patterns from the minority class (duplicate questions).

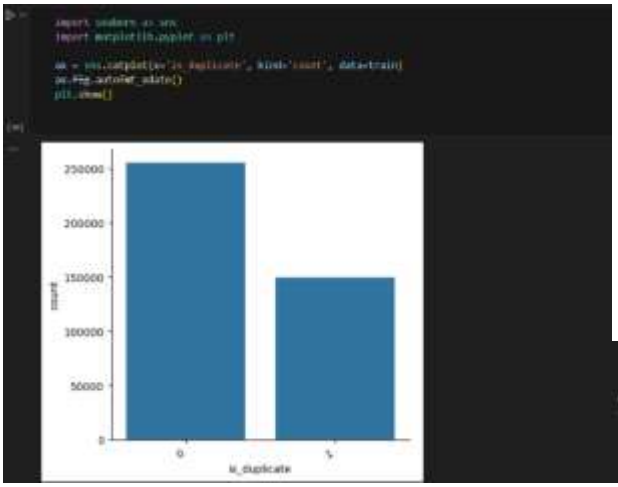
6. Model Training:

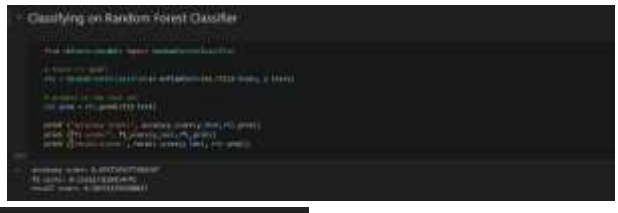
Features including similarity scores are used to train machine learning models. Logistic Regression serves as a simple and interpretable baseline, while Random Forest offers a more robust and flexible solution by combining multiple decision trees.

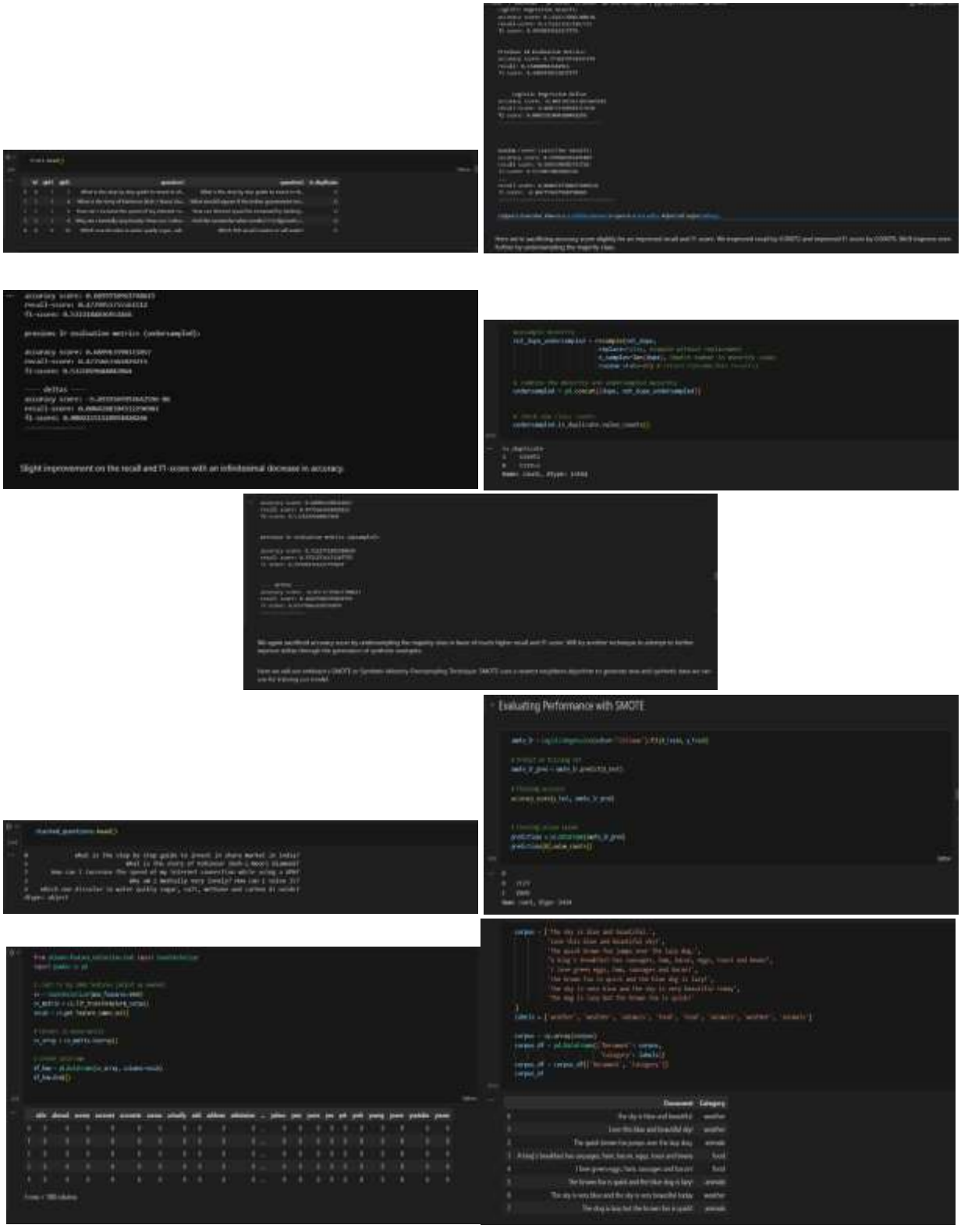
7. Prediction:

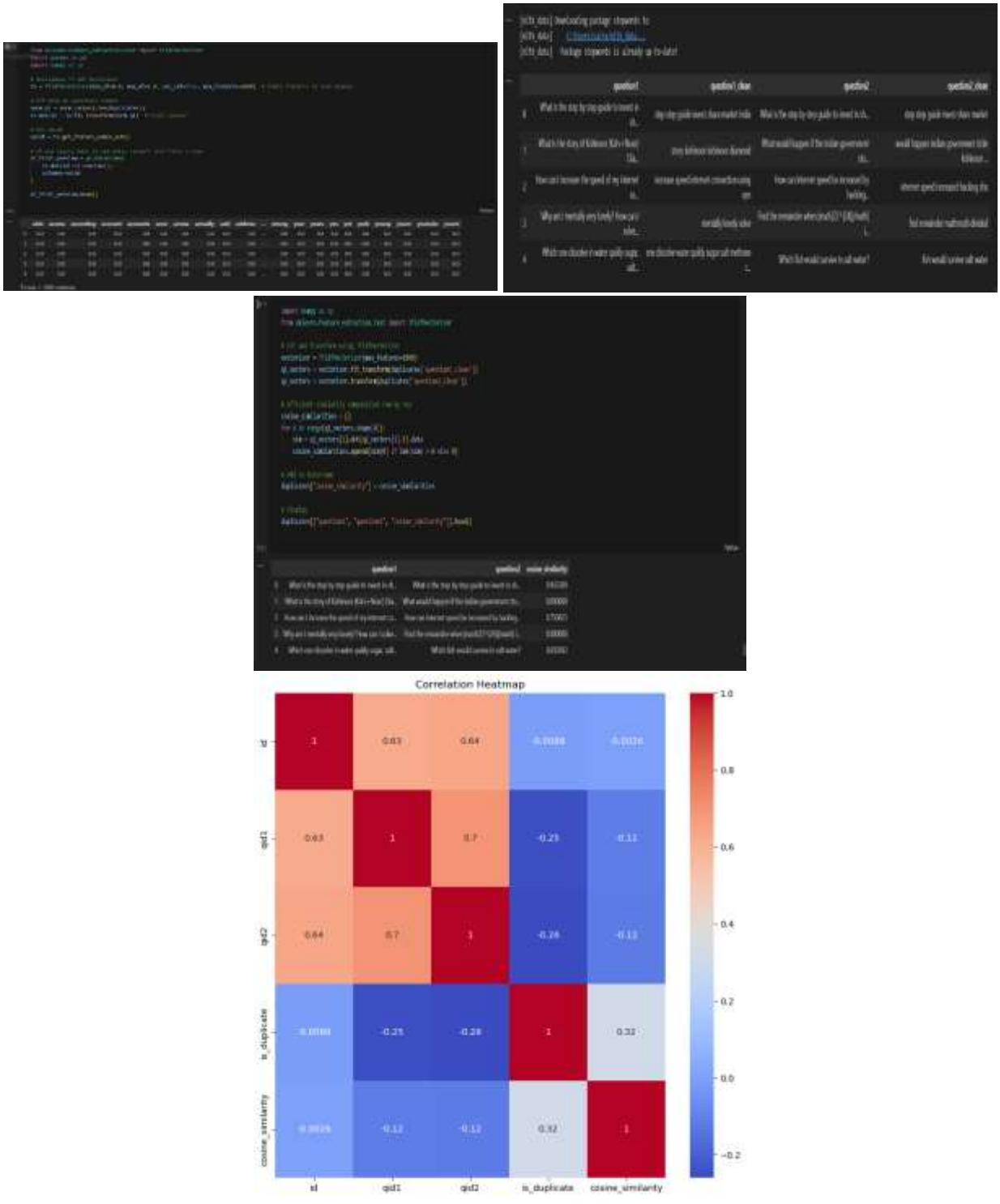
Once trained, the models predict whether a new pair of questions is a duplicate (label 1) or not (label 0). These predictions can be integrated into real-time systems to reduce redundancy on Q&A platforms and improve information retrieval.

RESULT ANALYSIS


250000
200000
150000
100000
50000
0
count
0
1
is_duplicate


Classifying on Random Forest Classifier


Slight improvement on the recall and f1-score with an infinitesimal decrease in accuracy
Evaluating Performance with SMOTE



CONCLUSION

This study presents a robust approach to duplicate question detection using Natural Language Processing and machine learning. Through TF-IDF vectorization, cosine similarity, and classifiers such as Logistic Regression and Random Forest, the system successfully identifies semantically similar questions. Incorporating resampling techniques further improves performance, especially on imbalanced datasets. This methodology can greatly benefit platforms dealing with large volumes of user-generated content. Future enhancements may include deep learning techniques such as BERT, LSTM, or Siamese networks

for improved semantic understanding.

REFERENCES

1. Shaeela Ayesha, Muhammad Kashif Hanif, and Ramzan Talib. Overview and comparative study of dimensionality reduction techniques for high dimensional data. Information Fusion, 59:44–58, 2020. 1
2. Aneesha Bakharia. Quick semantic search using siamese bert networks, 2023. 1
3. Jason Brownlee. How to stop training deep neural networks at the right time using early stopping, 2023. 4
4. CampusX. Duplicate question pairs, 2022. 2 [5] Priyanka Dandale. Quora question pairs similarity problem, 2021. 3
5. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. 2
6. Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pre trained language models: Weight initializations, data orders, and early stopping. arXiv preprint arXiv:2002.06305, 2020. 2
7. Hugging Face. Distilbert — transformers 4.10.0 documenta tion, 2023. 7
8. Lynette Gao and Yujing Zhang. Sentence embeddings for quora question similarity, 2023. 4 [10] Kaggle. Quora question pairs. 2017. 2
9. YuhuaLi,DavidMcLean,ZuhairABandar,JamesDO'shea, and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. IEEE transactions on knowledge and data engineering, 18(8):1138–1150, 2006. 7
10. Saleha Muzammil. How to deal with contractions in nlp, 2023. 4
11. PyPI. clean-text 0.6.0, 2023. 4
12. Ben Rodrawangpai and Witawat Daungjaiboon. Improving text classification with transformers and layer normalization. Machine Learning with Applications, 10:100403, 2022. 5 [15] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.