

IMBALANCED TEXT CLASSIFICATION WITH ABSTRACT FEATURE EXTRACTION

A major project report submitted in partial fulfillment of the requirement
for the award of degree of

Bachelor of Technology

in

Computer Science and Engineering/Information Technology

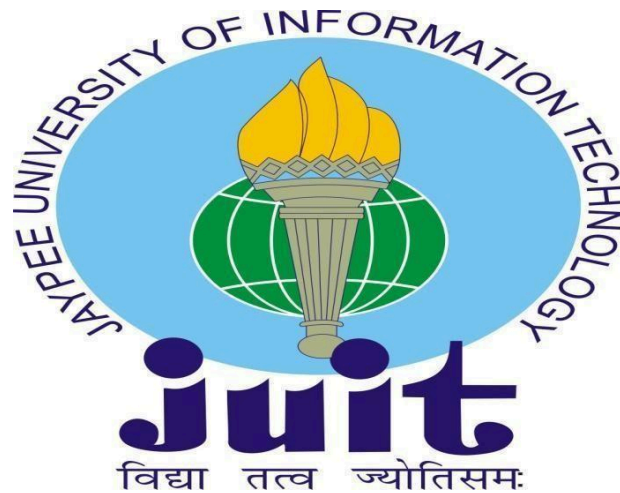
Submitted by

Ridhi Sood (201416)

Karanveer Singh (201185)

Under the guidance & supervision of

Dr. Rakesh Kanji



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Wagnaghat,

Solan-173234(India)

CERTIFICATE

This is to certify that the work which is being presented in the project report entitled “**Imbalanced text classification with abstract feature extraction**” in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Wagnaghat is an authentic record of work carried out by “Ridhi Sood(201416) and Karanveer Singh(201185).” during the period from August 2023 to December 2023 under the supervision of Dr. Rakesh Kanji, Assistant Professor (SG), Department of Computer Science and Engineering, Jaypee University of Information Technology, Wagnaghat.

Ridhi Sood (201416)

Karanveer Singh (201185)

The above statement made is correct to the best of our knowledge.

Dr. Rakesh Kanji

Assistant Professor (SG)

Computer Science and Engineering/Information Technology Department

Dated: May 15, 2024

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled “ **Imbalanced text classification with abstract feature extraction**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2024 to May 2024 under the supervision of **Dr. Rakesh Kanji, Assistant Professor (SG) in Computer Science and Engineering/Information Technology Department.**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Ridhi Sood

201416

Karanveer Singh

201185

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Rakesh Kanji

Assistant Professor (SG)

Computer Science and Engineering/Information Technology Department

Dated: May 15, 2024

ACKNOWLEDGEMENT

To begin, I would like to express my heartfelt gratitude to almighty God for his heavenly grace, which enabled us to successfully complete the project work.

I am extremely grateful and wish to express my deep gratitude to Supervisor **Dr. Rakesh Kanji, Asst. Prof. Senior Grade**, Department of CSE & IT Jaypee University of Information Technology, Waknaghat. His never-ending patience, intellectual direction, persistent encouragement, constant and vigorous supervision, constructive criticism, helpful suggestions, and reading numerous poor versions and revising them at all stages allowed this project to be completed. I would like to express my heartiest gratitude to Dr. Rakesh Kanji, Department of CSE & IT, for his kind help to finish my project.

I would also like to express my gratitude to everyone who has assisted me in making this project a success, whether directly or indirectly. In this unusual scenario, I'd like to express my gratitude to the different staff members, both teaching and non-teaching, who have provided me with valuable assistance and assisted my project. Finally, I must express my gratitude for my parents' unwavering support and patience.

Ridhi Sood (201416)

Karanveer Singh (20118)

TABLE OF CONTENTS

S. No.	Chapters	Page No.
1.	Chapter 1: Introduction	
	1.1 Introduction	1-2
	1.2 Problem Statement	2-3
	1.3 Objectives	3-4
	1.4 Significance and Motivation of the Project Work	4-5
	1.5 Organization of Project Report	6
2.	Chapter 2: Literature Survey	
	2.1 Overview of Relevant Literature	7-17
	2.2 Key Gaps in the Literature	17-18
3.	Chapter 3: System Development	
	3.1 Requirements and Analysis	19-20
	3.2 Project Design and Architecture	20-22
	3.3 Data Preparation	22-24
	3.4 Implementation	24-37
	3.5 Key Challenges	38-39
4.	Chapter 4: Testing	
	4.1 Testing Strategy	40-41
	4.2 Test Cases and Outcomes	41-42
5.	Chapter 5: Results and Evaluation	
	5.1 Results	43-44
6.	Chapter 6: Conclusions and Future Scope	
	6.1 Conclusion	45-47
	6.2 Future Scope	47-48
7.	References	
		49-51

LIST OF FIGURES

Figure No.	Label	Page No.
1	Project Design	20
2	Dataset Used	23-24
3	Formulae of TF-IDF	26
4	Importing Modules and Dataset	26
5	Data Collection	27
6	Data Pre-Processing	27
7	Implementing Word cloud to verify data pre-processing	28
8	Lemmatization of the data for the TF-IDF Model	29
9	Solving Class Imbalance Problem with Augmentation	30
10	Data Before Augmentation	31
11	Data After Augmentation	31
12	Applying TF-IDF Model	33-34
13	Implementing TF-IDF combined with Random Forest Classification	36-37
14	Precision, Recall, F-1 Score and Accuracy of the Model	44

LIST OF TABLES

Table No.	Label	Page No.
1	Literature Survey of Research Papers	12

LIST OF ABBREVIATIONS

Sr No.	Abbreviation	Meaning
1	LDA	Latent Dirichlet Allocation
2	TF-IDF	Term Frequency-Inverse Document Frequency
3	NLP	Natural Language Processing
4	CORPUS	Collection of text documents
5	DOC	Document
6	AUC	Area under precision-recall Curve

ABSTRACT

Imbalanced text classification is an important problem for NLP due to bias stems from classes with low representation of models. This paper presents a new technique based on abstract features for tackling the problem of imbalanced text categorization. The purpose is to improve classification performance of underrepresented classes while preserving the accuracy for a well-represented class.

The proposed methodology involves combining traditional text representation and advanced abstract feature extraction techniques including word embeddings, sub word embeddings, as well as contextual embeddings. Semantic and contextual indicators such as hidden marks and features play a significant role in counteracting the effect of class bias and improving the performance of the models in recognizing minority groups.

This Project uses TF-IDF

To demonstrate the effectiveness of the proposed approach, experimental tests using benchmark datasets with unequal class frequencies are performed. Comparison with traditional text classification techniques reveals that abstract feature extraction improves performance. This includes measures such as precision, recall, F1 score, and Area under precision-recall Curve (AUC), to assess the Overall model in balancing tasks of imbalanced Text Classification.

This finding leads to the conclusion that abstract feature extraction techniques significantly enhance the model's generalization ability when there is an unequal distribution of classes leading to fairer and stronger text classification. This project is a contribution in addressing the challenges posed with regard to class imbalance in natural language processing.

Chapter 01: INTRODUCTION

1.1 INTRODUCTION

Text classification is a basic NLP task that has many use cases such as sentiment analysis or categorizing topics. On the other hand, building an equitable and correct classification model in text data sets is quite thorny owing to imbalanced class distributions. In this regard, if not all classes are present or represented adequately, then the models end up being biased and fail to recognize patterns of minority classes. This paper deals with a complex issue of imbalanced text classification through combined abstract feature extraction methods.

The problem of class imbalance affects a text classification model's ability to effectively generalize beyond the classes with a higher relative frequency and make appropriate predictions for other lower frequency classes. Although traditional text representation methods like bag-of-words and TF-IDF capture basic linguistic features, they may be unable to deal with complications related to unbalanced datasets. Since then, the research focus shifts towards developing sophisticated feature extraction mechanisms capable of encompassing delicate semantic details and environmental connotations entailed by the text data.

A method called Term Frequency-Inverse Documents Frequencies or TF-IDF, is a quantitative representation used in natural language processes and information retrieval for measuring term importance in documents. This method combines two crucial metrics: TF-IDF. Frequency of occurrence of a term within a particular document forms the basis for TF which is important in determining significant terms in that document. In fact, IDF assigns the significance of rarity of a term in relation to the whole set of documents and diminishes the significance of common terms existing in various documents. A numerical value which reflects term relevance to the documents against the entire corpus in term frequency multiplied by inverse document frequency or simply called as $TF \times IDF$ equation. The technique that TF-IDF entails has wide applications that include information

retrieval, document categorization, and extracting keywords from texts.

In our study, we investigate using abstract feature extraction methods for improving imbalanced text classification model performance. By abstract features we should understand representation beyond conventional lexicon syntagmatic aspects leading to the semantic and extralinguistic ones. Some of these features are word embeddings for the semantic relations between words; sub word embeddings for handling the out – of – vocabulary words and morphological variations; and finally, contextual embeddings where it considers the surrounding context of words in a specific document or paragraph.

Abstract feature extraction is utilized as a form of compensation due it minimizes the effects of class imbalance. Using such refined technologies, we intend to help the model detect small features that relate to majority as well as minority categories for enhanced precision and equity among diverse clusters of individuals. Later we shall expound upon the employed method and how the abstract feature extraction techniques are integrated in the text categorization pipeline. We will look at how these approaches enrich the traditional methods of text representation and help us to present the full picture of textual data. Moreover, our experimental results are based on benchmark datasets that compare our proposed approach with traditional text classification approaches.

1.2 PROBLEM STATEMENT

Text classification especially introduces a great problem for machine learners. Imbalanced text classification describes situations where there is an imbalance in data distributions across different classes, one having many more data points than other classes. When dealing with it, we refer to the big class and small class respectively. The one which has a higher number of instances is the former and the latter refers to the one having a lower number of instances. This means that one class has more influence compared to other classes and when applied on machine learning algorithms it may lead to poor results.

This Project addresses the main issue of an imbalanced class because of disproportionate distribution between classes. This issue is widespread in the domain of text classification creating a significant challenge for any model that seeks to identify patterns contained within either bulk or minority groups. Class imbalance does not only imply statistical

asymmetry but it also leads to biasing machine learning models towards higher-sample classes.

Text data is very subtle, which constitutes the essence of the issue. Textual data has a unique structure different from that of structured datasets whereby each data point is usually characterized by a specified set of features. Consequently, orthodox categorization schemes fail to represent specific complexities within data, especially when it comes to small sections of text. With the majority of instances belonging to the majority category than the less-represented ones, it will lead to biased perspective because during learning the subtlety and underlying patterns present in the less represented categories will be overshadowed due to abundance of majority group instances. This result has many significant consequences. With respect to the class imbalance problem, machine learning algorithms display biases towards the majority class if they are inappropriately addressed. As a result, such models might give higher accuracy for most classes but fail in minor classes. This form of bias is particularly detrimental within the context of text classification since critical content may be overlooked, some significant instances may be misclassified, and false predictions are generated for rare ones.

1.3 OBJECTIVES

This project takes into consideration the continued imbalance problem in text categorization and employs an approach that includes abstract feature extraction. One the main task is using contemporary abstraction extracting techniques instead of outdated words in the bag for the class unbalanced solution for text information. Abstract feature extraction is another form of transformation which translates the textual information into large dimensioned, meaningful representations. These representations move beyond their limits to portray the hidden semantics, contextual data and built in structural elements residing in the text. The aim of this project is to use abstract features so that the model will be able to spot fine lines and other intricacies in both majority and minority groups.

The goals of this project are multi-faceted:

Building a resilient machine learning model for imbalanced data is the goal of this project. In most cases, traditional models suffer from bias towards the majority class outcome;

therefore, we emphasize on the creation of a class-imbalance-resilient model that is equally efficient on each category.

The project will utilize the abstraction feature extraction method for the extraction of most salient and relevant characteristics. It attempts to mitigate these shortcomings and equip the model to efficiently discriminate among different classes, like less represented categories.

The project will evaluate the effectiveness of the proposed methodology using real-world data sets. Therefore, the performance metrics will go beyond traditional accuracy and consider precision, recall, F1 score, and the area under the precision-recall curve. However, this evaluation process will demonstrate the practicality of the model and its capability to work with uneven class distribution in a wide range.

1.4 SIGNIFICANCE AND MOTIVATION OF PROJECT WORK

Our Project “Imbalanced Text Classification with Abstract Feature Extraction” addresses vital problems in NLP and machine learning. Here, we dig into the profound reasons behind undertaking this research endeavor:

SIGNIFICANCE:

Class imbalances in real world datasets are always common. In text classification, tackling this issue directly affects many applications such as sentiment analysis, spam filtering, among others. It also shows how the project’s outcomes could increase the model’s reliability and fairness that will be used in real life settings.

Predictive systems are often unbiased when it comes to imbalanced classes, which is mostly the case. It is important to mitigate this bias to achieve a level playing field as far as the automated decision-making system or algorithm is concerned. The project aims at text classification on an unbalanced dataset thereby promoting development of fairer machine learning algorithms. Such models of conventional text classification may find it difficult to capture even slight patterns belonging to minority groups. The objective of this project is to improve on the model and make it more accurate in predicting even minority categories by using abstract feature extraction techniques. This has some bearing on decisions made from such information

Abstract feature extraction, compared to bag-of-words as well as TF-IDF techniques, symbolizes progression. This is a critical project because it helps in building new ways to analyze and interpret the intricate nature and meaning in text data. The techniques applied in this project can be applied to the different fields of the data set. More importantly, this generalizability is important as it determines if the proposed techniques can solve many practical issues with impact all over the world.

MOTIVATION:

Most existing text classification models struggle with imbalance data problems, which negatively affects their accuracy. Motivated by the desire for enhanced techniques for balancing real-world, skewed textual data.

This is motivated by the idea that overcoming biased text categorization via extractive abstraction is a feasible and practical approach. The objective is to create mechanisms that work successfully in settings having variable population stratification and disease prevalence rates.

For automated decision support systems to be effective in numerous sectors, it is critical to make them trustworthy and unbiased. Therefore, the study intends to contribute towards the development of trustworthy model generation techniques, especially where the textual information is slightly unbalanced.

This is motivated by the need to go beyond the limits of feature extraction methods. The project's motivation is the desire to investigate whether abstract feature extraction can make any progress in text classification.

Therefore, project aims to provide a solution for imbalanced text classification. It should contribute to creation of more reliable, fair and precise machine learning mechanisms used in specific everyday reality situations.

1.5 ORGANIZATION OF PROJECT REPORT

Chapter 1: is an overview of the project, introducing the context, relevance, and key components of the research. It sets the stage for understanding the problem addressed by the project, objectives, and the significance of the research. It also outlines the motivation behind the project and previews the organization of the project report.

Chapter 2: The section is a summary of the body of literature concerning a project that involves consulting sources such as books, journals, websites, technical papers, and so forth. Additionally, it outlines important deficiencies found in existing studies that this project is designed to rectify and establishes the distinctiveness of the intended study.

Chapter 3: It starts from requirements and analysis, describing the demands and limitations of the project. This is followed by a project design and architecture that gives an overview look into the system. Data preparation appears in the last part specifying how data was pre-processed and implementation part describes coding, algorithms, and utilized techniques and tools. Further, issues experienced in development and construction are discussed.

Chapter 4: The testing chapter starts by the adopted test strategy and the testing tools used. Then it gives individual test examples and their results, revealing discovered troubles during testing and explaining ways of their solving.

Chapter 5: The results in this chapter represent an interpretation of findings coupled with a discussion on implications. In addition, it should be compared to other available solutions where possible, pointing out the benefits and originality.

Chapter 6: Finally, a summary of main findings, limitations, and contributions is provided in the last chapter. The conclusions provide details about the project's effectiveness and identify possible areas that may require refinement or extension.

Chapter 02: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

The term frequency (TF) determines how often a term features in a particular document. The ratio of occurrence of a term in a document to the total number of terms contained in the document. TF focuses on the importance of a term in a specific document. Assuming that the repeated words are more representative of the subject matter.

IDF measures how infrequent a word is in the whole database. This is obtained by taking a logarithm of the total number of documents within the corpus and divided by the documents that contain the term. For IDF to lower the impact of frequent terms, such frequencies must first be calculated and added up across the whole corpus. The IDF will award more points for rare terms that could be used as discriminating factors.

To compute the TF-IDF score of a term in a document, it becomes necessary to multiply the former with the latter. Therefore, the formula is $TF-IDF = TF \times IDF$.

High TF-IDF score means that the term occurs often on the document but rarely on the entire collection.

Applications:

Search engines often use TF-IDF as a measure of how relevant different documents are to a user's query.

Documents are vectorized using TF-IDF so that we can train machine learning algorithms for tasks such as text categorization. TF-IDF is useful in identifying crucial terms in a particular document and helps in summarization and content analysis.

Some Research Papers on TF-IDF studied to understand the working and applications of TF-IDF:

[1] Research of Text Classification Based on Improved TF-IDF Algorithm by Cai-zhi Liu¹ , Yan-xiu Sheng¹ , Zhi-qiang Wei¹ And Yong-Quan Yang

With the recent and quick development of Text data via Internet technology is expanding daily at a rapid rate. Users must sort through a vast amount of data to find what they need,

quantity of text. Consequently, text classification that is automatic Users of technology can find information more easily. To ensure that solve issues like missing contextual semantic links, for example and varied vocabulary significance in conventional writing categorization methods, this study offers a vector representation of deep learning-based feature words Word2vec tool, and the feature words' weight is computed using the enhanced TF-IDF algorithm. Through doubling the vector, the word weight, and the word weight the word's representation is achieved. Ultimately, every text is expressed by adding up each word's vectors.

[2] Text Classification Using Novel Term Weighting Scheme-Based Improved TF-IDF for Internet Media Reports by Zhiying Jiang, Bo Gao, Yanlin He, Yongming Han

Owing to the advancement in Internet technology, it is possible to collect large volumes of internet textual data. Text Classification (TC) technology is widely used for processing large scale text data and its output largely depends on the quality of term weighting system in TC. The TF- IDF strategy fails for internet media reports because it was originally designed for IR and is not efficient for TC when dealing with unbalanced text data distribution. Thus, it is proposed that the ADF should improve the capability for processing text with an unbalanced distribution of terms by measuring the variance between the DF value of a specific term relative to all others, that is, the ADF. The normal TF-IDF algorithm is then slightly modified by the proposed ADF to process a set of unbalanced texts in four ways, which are known as TF-IADF, TF-IDF+, TF-IDF norm and TF-IDF+ norm Therefore, it will be possible to create a viable blueprint for the TC task using internet media reports. The performances of the proposed methods are evaluated through a set of simulations. Simulation results indicate that the proposed methods confirm effectiveness and feasibility compare to state-of-the-art classification algorithms based on TF-IDF.

[3] Dealing with Data Imbalance in Text Classification by Cristian Padurariu a b, Mihaela Elena Breaban

Many real-world datasets do not offer enough training input for regular classifiers: some of the classes are better represented than other ones. This results into difficulties in Machine learning where it becomes challenging to predict an outcome because of lack of data for learning. In our research, the unit of classification is a field of HR data (short descriptions of experiences) that need to be assigned into several highly unbalanced classes denoting job types. We conduct an extensive experimentation exercise using different representations of

textual data, a number of classification algorithms and balancing techniques to arrive at the best performing model in terms of accuracy and recall and the contribution is twofold.

While Term Frequency-Inverse Document Frequency (TF-IDF) is a widely used and effective method for feature extraction in text analysis, it does have some limitations and disadvantages-IDF do not capture semantics among and between the terms. Instead, it considers each term by itself without any attention to document meaning in terms of word relationships (co-occurrences). Such lack has an effect on its ability to understand text with full meaning. TF-IDF is sensitive to document length because long documents will usually have higher overall term frequencies. However, more lengthy papers can unwittingly be attributed a weight for each term which may result in skewed results. In many real world applications, especially ones involving large document corpora that contain a lot of dimensions or variables, the resulting TF-IDF matrix ends up being a sparse one that contains relatively few nonzero values. Therefore, this sparsity may influence the computational efficiency of algorithms and models using TF-IDF representations. However, TF – IDF is not effective in dealing with synonyms and polysemy (multiplicity of meaning). This can lead to different terms having similar meanings being regarded as independent resulting in lower than optimum representations that are important when capturing semantic equivalence.

A probabilistic generative model, known as Latent Dirichlet Allocation (LDA), widely applied in NLP and machine learning for topic modelling. Developed by David Blei, Andrew Ng, and Michael Jordan in 2003, LDA discovers latent topics within document collection. Latent Dirichlet Allocation continues being a strong tool enabling detection of hidden topics among texts and that application is relevant not only in the area where it's important to reveal the contents of texts concerning the same.

Whether it is LDA or TF-IDF is a factor that should depend on the specific goals and needs of NLP or text analysis operation. However, each approach comes with unique advantages and disadvantages. In some occasions therefore, one may be superior compared to another. Hidden thematic structures and the semantic relationships are captured by LDA in a corpus. This creates an elaborate representation of the documents as spreads over topics, allowing for a better illustration in context. In particular LDA was developed with specific emphasis on application in domain such as topic modelling; hence its appropriateness for tasks that necessitate extraction of underlying subjects from group of papers and other literary pieces. In such cases, it is vital that this approach can prove useful when information about the thematic content is unknown. Generally, in comparison with LDA, this term tends to be

more inclined to noisy or non-informative terms. It also reduces the weight of words that are abundant but irrelevant as they relate to an article's topic. LDA reveals the relationship among words in documents and offers an overall view on the information. This approach considers how words occur together within the topic and improves the overall representation of the documents' content.

Some Research Papers on LDA to understanding its working, advantages and disadvantages are: -

[1] An Improved LDA Algorithm for Text Classification by Dexin Zhaol, Jinqun He1, Jin Liu2

The model latent Dirichlet allocation can derive a concept known as latent topic from a large body of data. This model presumes that any token in relevance to a theme must originate from relevance to a document. The document bears relation to that topic on them. In this paper, a new topic text classification algorithm labelled as, "G-LDA" topic-category distribution parameter to LDA, which will generate a document out of the most appropriate category. Approximate inference on the other hand is done using the Gibbs sampling the effectiveness of different datasets in providing means and experiment results.

[2] Research on text categorization model based on LDA – KNN by Weihua Chen, Xian Zhang

Similarity is a crucial component in the text categorization but the existing classification methods only consider the similarity between feature words and categories and does not involve the semantic similarity between feature words. A new classification model – LDA, LDA – KNN was proposed. Solving the problem of semantic similarity is done by using an LDA. In order to differentiate the sample, it uses KNN classification method in space. The experiment was conducted using a MATLAB software. The data set was sampled from Fudan Chinese corpus. The high precision classification result came in when university averaged value of 0.933 which was obtained." LDA-KNN model is MI (Mutual Information)-KNN Model Compared with LSI (Latent Semantic Index)-KNN model. The results show that Classification performance of LDA-KNN model is better than automatic text categorization.

[3] Text classification method based on self-training and LDA topic models by Miha Pavlinek , Vili Podgorelec

When supervised text classification techniques can pick up knowledge from labelled sets of a reasonable size, they perform well. Yet, semi-supervised techniques are more suitable when there is a limited number of labelled documents available. Because the foundation of these techniques is the comparison of distributions between labelled and unlabelled instances, attention must be paid to the representation and its capacity for discrimination. The ST LDA method, which uses topic model-based representations for semi-supervised text classification, is presented in this paper. A model that chooses parameter values for each new document collection and a semi-supervised text classification algorithm based on self-training make up the suggested approach. Self-training uses information from unlabelled data to enlarge the small initial labelled set. This study investigates to what extent topic-based representation influences model's prediction accuracy comparing the performances of the NBMN and SVM classification algorithms on an extended labelled set together with the TF/IDF representation.

LITERATURE REVIEW:

Given Below are the other research papers we went through to grasp the understanding on how we will pursue with the making of this project:

S. No	Paper Title	Journal/Conference (Year)	Tools/Technique/Dataset	Results	Limitations
1.	Text Classification Algorithm Based on TF-IDF and BERT	IEEE/2022	RNN KNN TF-IDF BERT	TF-IDF features are used to optimize the proper noun vectors, and uses BERT to extract the feature representation vectors of sentences and important proper nouns. Ultimately, the proper noun vectors are classified based on text features combined with information about the proper nouns and their contextual utterances.	Large models make challenging to deploy on resource-constrained devices or in environments with limited storage capacity. This can limit their applicability in certain real-time
2.	imbalance d text classification by Murat Okkalıoğlu, Burcu Demirelli Okkalıoğlu	Springer Nature/2022	NLP, SVM, KNN, AFE	AFE is a powerful tool in text classification that can serve as a feature extraction, dimension reduction method.	Data Availability, Overfitting

3.	A TF-IDF-Based Method for Imbalanced Text Classification Using a Weighted Cost-Sensitive Support Vector Machine by Wang, Y., Zhang, S., & Li, J.	Expert Systems, Volume 39, Issue 3, e12905, 2022, Wiley Online Library	TF-IDF Weighted Cost-Sensitive Support Vector Machine (SVM) Imbalanced text classification Dataset	The proposed method outperformed baseline methods on several imbalanced text classification datasets, including the Amazon Review dataset, the Yahoo! Answers dataset, and the SST-2 dataset.	The proposed method was only evaluated on a few imbalanced text classification datasets. It is possible that the performance of the method may vary on other datasets.
4.	Research on text categorization model based on LDA – KNN Weihua Chen, Xian Zhang	IEEE (2022)	LDA – KNN KNN MI- KNN LDA - NB	The LDA-KNN model presented in this paper combines the speed and simplicity of KNN with the similarity metric of LDA and shrinks feature space to enhance text categorization performance. Building a corpus, pre-processing samples, LDA modelling, and using the KNN algorithm for training and	LDA-KNN is a two-step process wherein topic modelling is done with LDA and classification is done with KNN. Though the effectiveness of LDA and KNN separately does not guarantee their synergy in

				testing were the methods used in the experiments.	combination, this approach might not guarantee a globally optimal solution.
5.	An Improved LDA Algorithm for Text Classification by Dexin Zhao ¹ , Jinqun He ¹ , Jin Liu ²	IEEE/2021	LDA G-LDA PLSI	G-LDA has a more efficient modelling process for word-sense disambiguation by employing a Dirichlet forest prior over topics in a category. The model indicates the topics which should have high probability in a category and helps us improve the quality of predictions on text data	G-LDA adds more complexity, which could make it more difficult to interpret. The model's outputs may become more difficult to interpret and explain as a result of the increased complexity.
6.	Research of Text Classification Based on Improved TF-IDF Algorithm	IEEE/2021	Word2Vec TF - IDF	This paper updates the TF-IDF algorithm and adds the weighting factor because the traditional algorithm is unable to accurately represent the distribution of feature words across different categories.	Improved IDF measures may introduce increased complexity compared to traditional IDF.

7.	Subjective Answers Evaluation Using Machine Learning and Natural Language Processing by MUHAMMAD FARRUK H BASHIR	IEEE/2021	Wordnet, Word2vec TF-IDF LDA WMD	This paper presented a novel method for evaluating subjective responses that is based on natural language processing and machine learning techniques. There are two suggested score prediction algorithms that yield scores that are up to 88% accurate.	In terms of future improvements, the word2vec model can be trained especially for subjective answers evaluation of a particular domain, and with large data sets, the number of classes or grades in the model can be significantly increased.
8.	A new TF-IDF-based method for imbalanced text classification by Jiang, Y., Zhang, L., Yang, Y., & Chen, Z.	Knowledge-Based Systems, Volume 193, 105430, 2020, Elsevier	TF-IDF SVM Imbalanced Text Classification Dataset	The proposed method outperformed baseline methods on several imbalanced text classification datasets, including the 20 Newsgroups dataset, the Reuters dataset, and the WebKB dataset.	The proposed method was only evaluated on a few imbalanced text classification datasets. It is possible that the performance of the method may vary on other datasets.

9.	Dealing with Data Imbalance in Text Classification Cristian Padurariu, Mihaela Elena Breaban	Elsevier B.V/2019	TF-IDF Word2Vec SVM	interactions between classification algorithms, text vectorization choices and the schemes to deal with degrees of imbalance	The paper focuses on a limited number of imbalanced text classification algorithms.
10.	Handling Imbalanced Data in Text Classification by Y. Li, et al.	2019	DNN	focusing on models from traditional models to deep learning	Data Dependency, Lack of Causality
11.	Learning from Imbalanced Data by Haibo He, Edwardo A.	IEEE/2019	Random Forest, AdaBoost, or XGBoost	comprehensive review of the development of research in learning from imbalanced data	Oversampling, Cost-Sensitive Learning

12	Text classification method based on self-training and LDA topic models Miha Pavlinek*, Vili Podgorelec	IEEE/2017	TF-IDF NBMN ST LDA	On very small labelled sets, the ST LDA method can improve classification results. A range of data sets with varying imbalanced ratios and initial labelled set ratios were used to test the method. It was found that ST LDA with an NBMN classifier performs better than other variations and compared methods.	Contextual information found in longer documents is frequently absent from short texts. Because short texts have limited context, ST LDA may find it difficult to capture subtle relationships between words or nuanced topics.
----	---	-----------	--------------------------	---	---

Table 1 : Literature Survey of Research Papers

2.1 Key Gaps in the Literature

TF-IDF (Term Frequency-Inverse Document Frequency) and LDA (Latent Dirichlet Allocation) are both methods employed in natural language processing and text analysis. However, they have different aims and the advantages they bring are diverse. TF-IDF is a rather simple method that is easy to master and apply. It's based on the simple concept of term frequency and inverse document frequency, hence, it is so easy for users to understand its mechanics. TF-IDF produces a simple feature representation of documents, where each document is recorded by a vector of TF-IDF scores of each term in the vocabulary. This representation is very helpful for many other tasks such as classification, clustering and information retrieval. The weights that TF-IDF gives to the terms used in the documents are the indicators of their importance. High TF-IDF scores mean terms that are both the most frequently used within a document and the least popular ones in the whole corpora, thus they are likely to have a special semantic meaning. Tf-Idf can be easily computed, particularly in sparse matrix representations, which makes it good for big-scale text analysis with limited computational power as well. TF-IDF enables the customization by the

modification of parameters like smoothing techniques, tokenization or simply IDF weighting schemes in order to meet the specific needs or features of the corpus. TF-IDF is quite language-independent and it can be used to any language without the necessity of intensive language-specific preprocessing or modeling adjustments. LDA, on the other hand, is a generative probabilistic model used for topic modeling. It assumes that documents are generated by a mixture of topics, where each topic is characterized by a distribution over words. LDA aims to discover these latent topics in a corpus and the distribution of words within each topic. It's commonly used for tasks such as document clustering, topic extraction, and content recommendation. TF-IDF is a statistical measure used to evaluate the importance of a term within a document relative to a corpus. It reflects how often a term appears in a document while also considering how unique the term is across the entire corpus. TF-IDF is typically used for tasks such as information retrieval, text mining, and document classification.

Chapter-3 SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS:

In order to properly use TF-IDF (Term Frequency-Inverse Document Frequency), a specific set of requirements and a systematic analysis process is a must. The first step to TF-IDF implementation is to set up the objectives of the technique, whether it is for keyword extraction, document similarity evaluation, or information retrieval. The analysis of the data characteristics is vital; this requires the inspection of the body of documents, which of course, consists of the size and the type of the documents. The most common sources are journal articles, papers or any more special features like language or domain. The preprocessing phases, which include the tokenization, lowercasing, and stop-word removal must be scrutinized to guarantee the quality of the data. The investigation of the way terms are distributed within documents, which is represented by term frequency (TF), and the way terms are distributed across the corpus, which is represented by inverse document frequency (IDF), require careful examination. For instance, parameters like weights and IDF calculation should be tested to find out the best results. Assessment procedures, both qualitative and quantitative, are required to evaluate the efficiency of TF-IDF for the aforementioned task. Techniques of NLP integration, for example compatibility with other NLP techniques and computational efficiency, should be taken into consideration. Lastly, the whole process and the results should be thoroughly documented in order to be used as a reference for future purposes and decision making. Thus, TF-IDF can be practically used in many natural language processing tasks by means of this organized Method. The TF-IDF effectiveness is the subject of a rigorous evaluation, which encompasses both the qualitative and quantitative assessments. Qualitative review of the best words and their meaningful contexts is a part of the quantitative evaluation, which is usually done by comparing the TF-IDF-based documents ranking to the ground truth relevance or the performance benchmarks.

The mind of the integration issues is big as TF-IDF is the intersection of NLP work and the workflows or applications. The above-mentioned are the factors which have to be thoughtfully considered. Lastly, the entire process documentation, which summarizes the whole process from the requirement analysis to the optimization of the parameters and finally the evaluation outcome, is a good reference for future purposes and knowledge sharing. The methodical approach to the TF-IDF system allows it to be seen not only as a tool but also as a strategic asset in the analysis of the natural language data.

3.2 PROJECT DESIGN AND ARCHITECTURE

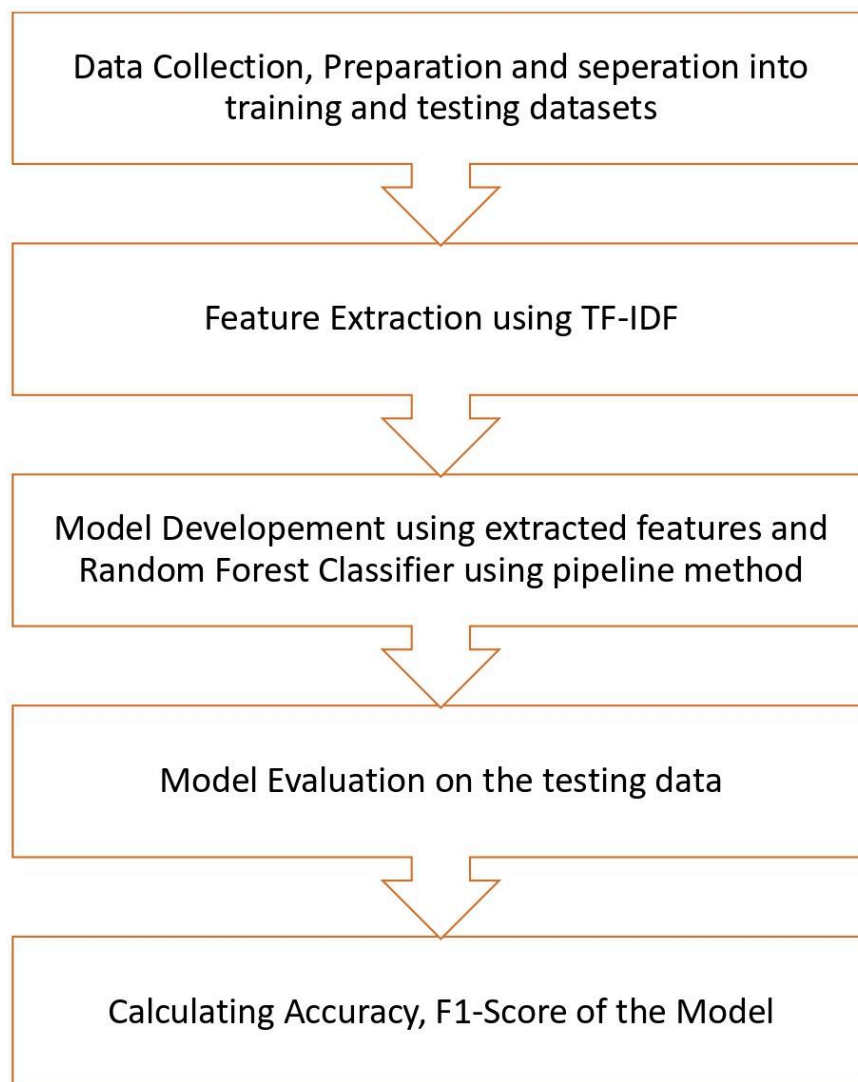


FIG 1: Project Design

The data preparation phase involves gathering and manipulating appropriate textual datasets. This entails pinpointing data that is prone to skewness; class imbalance has become an everyday occurrence in text classification. Some preprocessing tasks that come into play involve dealing with missing values, cleaning textual data, and class balance by using oversampling or under-sampling. This goal aims at having a tidy and balanced dataset which is ideal for subsequent feature extraction and model creation.

Feature extraction with TF-IDF is a fundamental step in the process of transforming raw textual data into a format that computational algorithms can efficiently process and analyze. This process is based on the idea of capturing the terms which are important in documents depending on the context of the whole corpus. The first step in the process of turning text into data is tokenization, which is the breaking down of the raw text into individual units, usually words or phrases, so that the text can be further analyzed. Every document's term frequency (TF) is obtained after which its value is computed which indicates how many times each term appears within the document in relation to the total number of terms in the document. The stage is very important in this process as it points out the significance of certain terms in the document. At the same time, the IDF for each term is derived from all the documents in the corpus. IDF suggests that the uniqueness of the terms in the documents is the main feature to identify them; the terms that appear only in one document and are not common in the whole corpus are defined as the most important. Through combining TF and IDF, the TF-IDF score for each term in each document is calculated. This score acts as the gauge of a term's importance in a text compared to its frequency in the corpus. Hence, terms that are common within a document but not found anywhere else in the corpus, receive higher TF-IDF scores, this means that they are crucial in describing the document. These TF-IDF scores are saved and used as feature values, thus, each document is converted into a numerical vector representation. In this vector space, each dimension represents a particular term, and the value of each dimension is the TF-IDF score of the particular term that is in the document. This vectorization process ultimately leads to the creation of a high-dimensional sparse matrix, in which documents are represented as points in this multi-dimensional space. This conversion doesn't only help the further analysis but also helps in the comparison and similarity calculations of the documents based on their content. The heart of the project lies in the development of the

classification model using a Random Forest Classifier within a pipeline framework. This approach offers several advantages, including modularity, scalability, and ease of reproducibility. The pipeline seamlessly integrates the TF-IDF vectorization process with the model training phase, streamlining the workflow and reducing the risk of errors or inconsistencies. Hyperparameters of the Random Forest Classifier, such as the number of trees, maximum depth, and minimum samples per leaf, are carefully tuned to optimize model performance and generalization ability.

With the model trained, the next step involves evaluating its performance on the testing dataset. This evaluation encompasses a range of metrics, including accuracy, precision, recall, and F1-score, which collectively provide a comprehensive assessment of the model's effectiveness in classifying text instances. Additionally, confusion matrices are employed to visualize the distribution of true positive, false positive, true negative, and false negative predictions, offering valuable insights into the model's strengths and weaknesses.

3.3 DATA COLLECTION

A Dataset created using data about the sorting techniques which are imbalanced and our aim is to work on this imbalanced dataset. Sorting algorithms involves generating text data that reflects the characteristics of different sorting algorithms. Every text sample serves as an explanation or description of a particular sorting algorithm. Collected data about incorporating well-known sorting algorithms such as Selection Sort, Counting Sort, Insertion Sort, and shell Sort and assigned labels to each text sample based on the sorting algorithm it describes. Introduced class imbalance by varying the number of samples for each sorting algorithm and ensured that one or more algorithms have significantly fewer samples, simulating a real-world scenario of imbalanced class distribution. Included additional features that might contribute to classifying the algorithms. These could be technical terms, key characteristics, or complexities associated with each sorting algorithm. These features will be used for abstract feature extraction using TF-IDF. The goal is to have a dataset that reflects both the technical details of sorting algorithms and the imbalanced distribution typical of real-world scenarios.

The purpose of using a dataset pertaining to sorting techniques is to demonstrate how abstract feature extraction can be used in a situation where purposeful class imbalance is

introduced. Specifically, sorting algorithms are well-defined with their own properties. As a result, this helps give textual descriptions that make sense to every algorithm, which in turn will lead to the formation of a labelled set. Sorting algorithms are ranked according to their respective time complexity, space complexity and effectiveness as a whole entity. This context can be enhanced with the introduction of class imbalance, which creates an artificial situation in which certain algorithms are more prevalent and harder to classify than other ones. Sorting algorithms are characterized by a technical language that is commonly used in algorithm discussions. This creates an appropriate platform for unravelling how abstraction using TF-IDF catches and classifies technical words in the text. Some specific sorting methods do not necessarily have any direct connection with text classification, but the experience learned with class imbalance and feature extraction methods are transferable to more sophisticated and realistic text classification tasks. Sorting algorithm's inherent structure and properties might favour feature extraction mechanisms such as TF-IDF. The algorithms involved herein have their own unique nomenclature and distinct stages that can be well represented by abstract features.

It's crucial to remember that the project objectives and the features of the text data you want to categorize will determine which dataset you use. In practice, the dataset would be selected according to the particular application or domain that calls for imbalanced text classification. For illustrative purposes, the sorting techniques dataset is used here; you can modify the approach to a domain more pertinent to your particular use case.

TEXT	TEXT TAG	TEXT SUBTAG
1 Bucket sort is also known as bin sort. It works by distributing the element into the array also called buckets. In this sorting algorithms, Buckets are sorted individually by using different sorting algorithm.	SORTING ALGORITHM	BUCKET SORT
2 Bucket sort, also known as bin sort, is a sorting algorithm that partitions the input data into a number of buckets. Each bucket is then sorted using another sorting algorithm, such as insertion sort or merge sort. The sorted buckets are then combined to produce the final sorted output.	SORTING ALGORITHM	BUCKET SORT
3 Bucket sort is an efficient sorting algorithm for data that is distributed relatively uniformly within a known range. It is particularly useful when the input data is already partially sorted or when dealing with large datasets.	SORTING ALGORITHM	BUCKET SORT
4 Steps of the Bucket Sort Algorithm include determining the number of buckets and the range of values for each bucket, then distributing each element of the input data into the appropriate bucket based on its value followed by sorting each bucket individually using another sorting algorithm followed by combining the sorted buckets to create the final sorted output.	SORTING ALGORITHM	BUCKET SORT
5 Bucket sort is a valuable sorting algorithm that offers efficient sorting for data with a known range and relatively uniform distribution. Its simplicity, intuitiveness, and scalability make it a popular choice for various applications. However, it is important to consider the sensitivity to input data distribution and the additional sorting step when evaluating its suitability for specific scenarios.	SORTING ALGORITHM	BUCKET SORT
6 Bucket sort, also known as bin sort, works by dividing the input data into a set of buckets based on their values. Each bucket represents a range of values, and elements are assigned to their respective buckets. Once the elements are distributed into buckets, they are sorted individually using a different sorting algorithm, such as insertion sort or merge sort. The sorted buckets are then combined to form the final sorted output.	SORTING ALGORITHM	BUCKET SORT
Advantages include efficient for uniformly distributed data: Bucket sort excels at sorting data that is evenly distributed within a known range. It is Scalable as it can handle large datasets efficiently,		

105	Comb sort demonstrates adaptability to partially sorted or presorted data. The algorithm's dynamic gap reduction allows it to quickly finalize the sorting process when presented with data that is already partially ordered, contributing to its efficiency in specific scenarios.	SORTING ALGORITHM	COMB SORT	
106	While not always the primary choice for large datasets, comb sort excels in sorting small arrays. Its simplicity and adaptability make it a suitable candidate for scenarios where the input size is limited.	SORTING ALGORITHM	COMB SORT	
107	Comb sort's performance in practice often surpasses its theoretical worst-case time complexity. Empirical studies have shown that the algorithm performs well on average, making it a practical choice for certain applications.	SORTING ALGORITHM	COMB SORT	
108	The step-by-step nature of comb sort's gap reduction process makes it visually intuitive. Visual representations of the sorting process help learners and developers grasp the algorithm's inner workings and understand how the gap reduction enhances efficiency.	SORTING ALGORITHM	COMB SORT	
109	The choice of the initial gap size can impact the overall performance of comb sort. Smaller initial gaps may expedite the sorting process for datasets with varying degrees of disorder, while larger initial gaps may be more suitable for already partially ordered data.	SORTING ALGORITHM	COMB SORT	
110	Comb sort shares similarities with other quadratic sorting algorithms like bubble sort and insertion sort. Comparing these algorithms sheds light on the nuances of their respective approaches and highlights the specific advantages of comb sort in certain contexts.	SORTING ALGORITHM	COMB SORT	
111	Similar to bubble sort, comb sort is a stable sorting algorithm. It preserves the relative order of equal elements, making it suitable for applications where maintaining the original order is crucial.	SORTING ALGORITHM	COMB SORT	
112	Parallelizing comb sort poses challenges due to its sequential nature. However, researchers have explored techniques to parallelize certain aspects, such as comparing and swapping elements within the comb, to leverage parallel processing capabilities.	SORTING ALGORITHM	COMB SORT	
113	Comb sort handles duplicate elements in a manner similar to bubble sort. When equal elements are encountered during comparisons, their relative order is maintained, ensuring the stability of the algorithm.	SORTING ALGORITHM	COMB SORT	

FIG 2: Dataset Used

3.4 IMPLEMENTATION

TOOL AND TECHNOLOGIES USED:

Python: Python is an easy-to-use computer language that is often used in machine learning and natural language processing. It has a lot of tools and frameworks that make working with data, building models, and evaluating them easier.

Pandas: The Python library Pandas is a strong tool for working with and pre-processing large amounts of data.

NumPy: NumPy is an important package for science computing in Python. It gives strength to large, multidimensional arrays and matrices, which are needed to solve mathematical operations.

NLTK (Natural Language Toolkit): NLTK, which stands for "Natural Language Toolkit," is a library for natural language processing jobs such as part-of-speech tagging, tokenization, lemmatization, stemming, etc

Scikit-learn: Scikit-learn is a powerful machine learning package provided by python which helps in Clustering, classification, regression, etc. It also includes utilities for balancing unbalanced datasets..

TensorFlow or PyTorch: TensorFlow or PyTorch can be used to build neural network architectures for people who want to learn more about deep learning methods. This might be helpful for adding neural topic models or complicated embeddings.

Matplotlib and Seaborn: Matplotlib and Seaborn are well-known Python tools for making graphs and charts. They are necessary for seeing how data is distributed, how well a model is doing, and how abstract features are represented.

Jupyter Notebooks or Google Colab: Jupyter Notebooks let you work on code, explore data, and see visualisations in a dynamic space. Google Colab is an online tool that makes it easy to work together on Jupyter Notebooks that have GPU resources available.

Git and GitHub: Git is a distributed version control system, and GitHub is a place where Git projects can be hosted and people can work together on them. They are very important for keeping track of different versions of code, working together on projects, and documenting them.

TF-IDF:TF-IDF, which means Term Frequency-Inverse Document Frequency, is a statistical measure used in natural language processing (NLP) to calculate the importance of a term in a document in relation to the corpus.

Term Frequency (TF): This assesses the number of times a term is used in a document. It is computed by counting the number of times a term is used in a document and then normalizing it by the total number of terms in the document. The theory is that the words that are used more often in a document are the ones that are the most important for that document's content.

Inverse Document Frequency (IDF): This counts the uniqueness of a term in the whole corpus. It is the quotient of the total number of documents in the corpus and the number of documents containing the term, and then the logarithm of that quotient is calculated. The reason is that terms that happen in fewer documents are more unique and therefore more helpful for differentiating between documents.

The TF-IDF score for a term in a document is the product of its Term Frequency (TF) and its Inverse Document Frequency (IDF). Thus, each term in each document receives a numerical score, where a high score shows that the term is more important in the document than in the whole corpus. TF-IDF is usually employed for a number of NLP tasks such as information retrieval, text mining, document classification, and content recommendation. It is useful in the process of finding significant terms in documents, enhancing the search relevance, and extracting the meaningful features for the machine learning models. In general, TF-IDF is a straightforward but effective approach for measuring the importance of the terms in the documents, thus, it is a useful tool for the analysis and processing of the text data.

$$TF = \frac{\text{Number of times a word "X" appears in a Document}}{\text{Number of words present in a Document}}$$

$$IDF = \log \left(\frac{\text{Number of Documents present in a Corpus}}{\text{Number of Documents where word "X" has appeared}} \right)$$

$$TF\ IDF = TF * IDF$$

FIG 3: Formulae of TF-IDF

```
[5] from google.colab import files
    ds=files.upload()
```

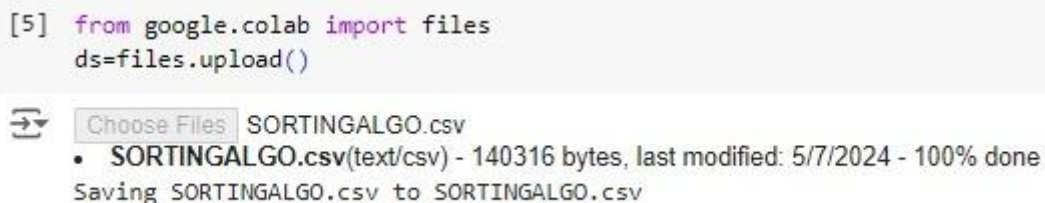
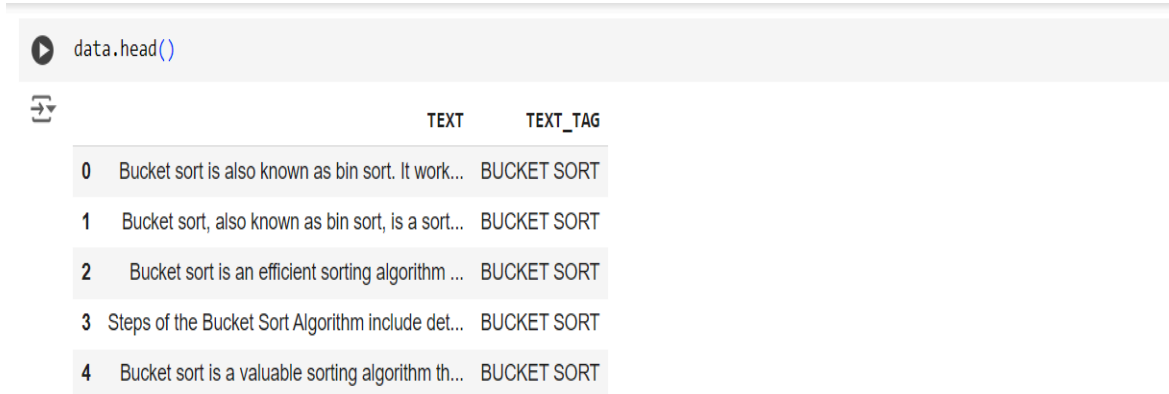


FIG 4: Importing Modules and the Dataset

DATA CLEANING AND PREPROCESSING:

Since the goal of this analysis is to perform topic modelling, we focus only on the text data and drop other metadata columns. So, we drop the column named “TEXT TAG” which is not helpful in performing the overall analysis.



```
data.head()
```

	TEXT	TEXT_TAG
0	Bucket sort is also known as bin sort. It work...	BUCKET SORT
1	Bucket sort, also known as bin sort, is a sort...	BUCKET SORT
2	Bucket sort is an efficient sorting algorithm ...	BUCKET SORT
3	Steps of the Bucket Sort Algorithm include det...	BUCKET SORT
4	Bucket sort is a valuable sorting algorithm th...	BUCKET SORT

FIG 5: Data Cleaning

REMOVE THE PUNCTUATIONS AND LOWERCASE:

As the next phase, we do some easy pre-processing on the content of the column to make it easier to analyse and get stronger results. One way to do that is to use a regular expression to get rid of every punctuation mark and then lowercase the text.

```
[ ] # Load the regular expression library
import re
# Remove punctuation
data['TEXT'] = \
data['TEXT'].map(lambda x: re.sub('[,\.!?!]', '', x))
data['TEXT_TAG'] = \
data['TEXT_TAG'].map(lambda x: re.sub('[,\.!?!]', '', x))
# Convert the titles to lowercase
data['TEXT'] = \
data['TEXT'].map(lambda x: x.lower())
data['TEXT_TAG'] = \
data['TEXT_TAG'].map(lambda x: x.lower())
# Print out the first rows of papers

data.head()
```

FIG 6: Data Pre-processing

EXPLORATORY ANALYSIS

To see if the preprocessing worked, we use the word cloud package to make a word cloud that shows the most popular words. It is important to understand the data and find out if the data needs to be cleaned up any further before the model is trained.


```

✓ [19] ### utility function for pre-processing the text
7s import spacy

# load english language model and create nlp object from it
nlp = spacy.load("en_core_web_sm")

def preprocess(processed_text):
    # remove stop words and lemmatize the text
    doc = nlp(processed_text)
    filtered_tokens = []
    for token in doc:
        if token.is_stop or token.is_punct:
            continue
        filtered_tokens.append(token.lemma_)

    return " ".join(filtered_tokens)

✓ [20] processed_data['processed_text'] = processed_data['processed_text'].apply(preprocess)
7s

✓ [21] processed_data.head()
0s

```

	processed_text	processed_text_tag
0	bucket sort know bin sort work distribute elem...	bucket sort
1	bucket sort know bin sort sort algorithm parti...	bucket sort
2	bucket sort efficient sort algorithm data dist...	bucket sort
3	step bucket sort algorithm include determine n...	bucket sort
4	bucket sort valuable sort algorithm offer effi...	bucket sort

Next steps: [View recommended plots](#)

FIG 8: Lemmatization of the data for the TF-IDF Model

SOLVING CLASS IMBALANCE PROBLEM WITH AUGMENTATION:

Addressing dataset imbalance involves several key steps. Firstly, it's crucial to understand the extent of the imbalance by analyzing the distribution of class labels. Next, select suitable augmentation techniques tailored to the dataset's characteristics and imbalance nature. Techniques such as data augmentation, SMOTE, or paraphrasing can be employed to create synthetic samples for minority classes. Implement these techniques to augment the dataset, ensuring that the integrity and quality of the data are preserved throughout the process.

Following augmentation, train the machine learning model on the augmented dataset to learn from a more representative sample of all classes. Evaluate the model's performance using appropriate metrics such as accuracy, precision, recall, and F1-score to assess its

effectiveness in handling the imbalance. Fine-tune the model based on performance evaluation results, adjusting augmentation parameters or exploring different techniques to further enhance performance.

Continuous monitoring of the model's performance is essential, especially in real-world scenarios where data distributions may change over time. Iterate on the augmentation process as needed to maintain the model's effectiveness and robustness. By following this comprehensive approach, dataset imbalance can be effectively mitigated, leading to the development of more accurate and reliable machine learning models.

```
[40] import nlpaug.augmenter.word.context_word_embs as aug

[41] sample_text = processed_data['processed_text'].iloc[100]

[42] sample_text
↳ 'comb sort relatively simple efficient sort algorithm devise 1980 computer scientist włodzim
```

```
[43] augmenter = aug.ContextualWordEmbsAug(model_path='bert-base-uncased', action="insert")

↳ tokenizer_config.json: 100% ██████████ 48.0/48.0 [00:00<00:00, 919B/s]
   config.json: 100% ██████████ 570/570 [00:00<00:00, 8.57kB/s]
   vocab.txt: 100% ██████████ 232k/232k [00:00<00:00, 1.63MB/s]
   tokenizer.json: 100% ██████████ 466k/466k [00:00<00:00, 4.27MB/s]
   model.safetensors: 100% ██████████ 440M/440M [00:05<00:00, 98.5MB/s]
```

```

def augmentMyData(processed_data, augmenter, repetitions=1, samples=200):
    augmented_texts = []
    # select only the minority class samples
    spam_df = processed_data[processed_data['processed_text_tag'] == 10].reset_index(drop=True) # removes unnecessary index column
    for i in tqdm(np.random.randint(0, len(spam_df), samples)):
        # generating 'n_samples' augmented texts
        for _ in range(repetitions):
            augmented_text = augmenter.augment(spam_df['processed_text'].iloc[i])
            if not isinstance(augmented_text, str):
                augmented_text = str(augmented_text)
            augmented_texts.append(augmented_text)

    data1 = {
        'processed_text_tag': 10,
        'processed_text': augmented_texts
    }
    aug_df = pd.DataFrame(data1)
    processed_data = pd.concat([processed_data, aug_df], ignore_index=True)
    return processed_data

[ ] aug_df = augmentMyData(processed_data, augmenter, samples=25)

↳ 100% ██████████ 25/25 [01:13<00:00, 3.24s/it]
```

FIG 9: Solving Class Imbalance Problem with Augmentation

DATASET BEFORE AUGMENTATION:

```
▶ processed_data['processed_text_tag'].value_counts()

↔ processed_text_tag
0      50
1      50
2      50
3      50
4      50
5      50
6      50
7      50
8      50
9      25
10     25
Name: count, dtype: int64
```

FIG 10: Data Before Augmentation

DATASET AFTER AUGMENTATION:

```
[ ] aug_df['processed_text_tag'].value_counts()

↔ processed_text_tag
0      50
1      50
2      50
3      50
4      50
5      50
6      50
7      50
8      50
9      50
10     50
Name: count, dtype: int64
```

FIG 11: Data After Augmentation

APPLYING TF-IDF

Term Frequency measures the frequency of a term (word) in a document. It's calculated by dividing the number of times a term appears in a document by the total number of terms in

that document. The idea is that the more times a term appears in a document, the more important it might be to that document.

IDF measures the importance of a term across a collection of documents. It's calculated by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient. The purpose of IDF is to give higher weight to terms that are rare across the entire document collection but common within individual documents.

To calculate the TF-IDF score for a term in a document, you simply multiply its TF by its IDF. This results in a numerical value that represents the significance of the term in that particular document. By considering both the frequency of a term within a document (TF) and its rarity across the entire corpus (IDF), TF-IDF can effectively highlight terms that are both relevant to a specific document and distinctive within the collection as a whole. This approach helps in various natural language processing tasks such as information retrieval, document classification, and text summarization, enabling systems to better understand and process textual data. Implementing TF-IDF involves straightforward calculations, making it widely applicable in diverse domains where textual analysis is crucial.

APPLYING TF-IDF (contd.)

APPLYING TF-IDF

```
[56] grouped_data = aug_df.groupby('processed_text_tag')['processed_text'].apply(lambda x: ' '.join(x)).reset_index()
```

```
[57] grouped_data['tokenized_text'] = grouped_data['processed_text'].apply(lambda x: word_tokenize(x))
```

Calculating Term Frequency

```
[59] tfidf_vectorizer = TfidfVectorizer()  
X_tfidf = tfidf_vectorizer.fit_transform([' '.join(tokens) for tokens in grouped_data['tokenized_text']])  
print(tfidf_vectorizer.vocabulary_)
```

```
{'bucket': 137, 'sort': 1134, 'know': 665, 'bin': 128, 'work': 1322, 'distribute': 341, 'element': 376, 'array': 85,
```

```
[60] tfidf_df = pd.DataFrame(X_tfidf.toarray(), columns=tfidf_vectorizer.get_feature_names_out())  
tfidf_df['processed_text_tag'] = grouped_data['processed_text_tag']
```

Calculating Inverse Document Frequency

```
[61] for word in tfidf_df:  
  
    #let's get the index in the vocabulary  
    indx = tfidf_vectorizer.vocabulary_.get(word)  
  
    #get the score  
    idf_score = tfidf_vectorizer.idf_[indx]  
  
    print(f"{word} : {idf_score}")
```

```
10 : 2.791759469228055  
13 : 2.791759469228055  
1945 : 2.791759469228055  
1956 : 2.791759469228055  
1960 : 2.791759469228055  
1963 : 2.791759469228055  
1964 : 2.791759469228055  
1965 : 2.791759469228055  
1980 : 2.791759469228055  
19th : 2.791759469228055  
ability : 1.1823215567939547  
about : 2.791759469228055  
academic : 2.791759469228055  
accelerate : 2.791759469228055  
access : 1.6931471805599454  
accessible : 1.6931471805599454  
accommodate : 2.386294361119891  
accomplish : 2.791759469228055  
accord : 2.791759469228055  
accordingly : 2.791759469228055  
accuracy : 2.791759469228055  
accurate : 1.538996500732687  
achieve : 2.09861228866811  
action : 2.386294361119891  
actual : 2.791759469228055  
actuary : 2.791759469228055  
adapt : 1.0870113769896297  
adaptability : 1.0870113769896297  
adaptable : 2.791759469228055  
adaptation : 1.538996500732687
```

```

print(tfidf_df)

```

	10	13	1945	1956	1960	1963	1964	\	
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.010612	0.000000		
1	0.000000	0.000000	0.000000	0.014131	0.000000	0.000000	0.000000		
2	0.000000	0.011389	0.000000	0.000000	0.000000	0.000000	0.000000		
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000		
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.009923		
5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000		
6	0.000000	0.000000	0.013305	0.000000	0.000000	0.000000	0.000000		
7	0.000000	0.000000	0.000000	0.000000	0.018632	0.000000	0.000000		
8	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000		
9	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000		
10	0.010873	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000		
	1965	1980	19th	...	work	workflow	working	workload	\
0	0.000000	0.000000	0.000000	...	0.012872	0.000000	0.000000	0.015954	
1	0.000000	0.000000	0.000000	...	0.008570	0.000000	0.010623	0.000000	
2	0.000000	0.011389	0.000000	...	0.000000	0.007651	0.008561	0.008561	
3	0.000000	0.000000	0.000000	...	0.012443	0.000000	0.000000	0.000000	
4	0.009923	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.007460	
5	0.000000	0.000000	0.000000	...	0.008861	0.000000	0.010983	0.000000	
6	0.000000	0.000000	0.000000	...	0.000000	0.008938	0.000000	0.000000	
7	0.000000	0.000000	0.000000	...	0.000000	0.012517	0.000000	0.000000	
8	0.000000	0.000000	0.012657	...	0.000000	0.000000	0.000000	0.000000	
9	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	
10	0.000000	0.000000	0.000000	...	0.006595	0.014609	0.000000	0.000000	
	world	write	wrong	włodzimierz	year	processed_text_tag			
0	0.010685	0.000000	0.000000	0.000000	0.010612			0	
1	0.042684	0.014131	0.042393	0.000000	0.000000			1	
2	0.000000	0.000000	0.000000	0.011389	0.000000			2	
3	0.005164	0.000000	0.000000	0.000000	0.000000			3	
4	0.000000	0.000000	0.000000	0.000000	0.000000			4	
5	0.007355	0.000000	0.000000	0.000000	0.000000			5	
6	0.006698	0.000000	0.000000	0.000000	0.000000			6	
7	0.009380	0.000000	0.000000	0.000000	0.000000			7	
8	0.000000	0.000000	0.000000	0.000000	0.000000			8	
9	0.028065	0.000000	0.000000	0.000000	0.000000			9	
10	0.000000	0.000000	0.000000	0.000000	0.000000			10	

[11 rows x 1332 columns]

FIG 12: Applying TF-IDF Model

COMBINING TF-IDF WITH RANDOM FOREST CLASSIFIER TO CREATE A CLASSIFICATION MODEL USING PIPELINE METHOD:

Combining TF-IDF with a Random Forest Classifier using the pipeline method offers a streamlined approach for text classification tasks. TF-IDF, a widely used technique in natural language processing, evaluates the importance of words in a document relative to a corpus. It calculates weights based on both the term frequency (TF) and the inverse document frequency (IDF), providing a numerical representation of text data. Random Forest Classifier, on the other hand, is an ensemble learning method known for its robustness and ability to handle high-dimensional data effectively. The pipeline method allows for the seamless integration of TF-IDF vectorization and Random Forest Classifier into a single workflow. This simplifies the process by encapsulating multiple steps, from data preprocessing to model training and evaluation, into a single entity. In this approach, the text data is first preprocessed, including steps like cleaning and splitting into training and testing sets. Then, a pipeline is constructed, consisting of TF-IDF vectorization followed by Random Forest Classifier. The pipeline is trained on the training data, where both vectorization and model fitting occur simultaneously. Once trained, the model's performance is evaluated on the testing set using appropriate evaluation metrics. By combining TF-IDF with Random Forest Classifier in a pipeline, we create a robust text classification model capable of handling high-dimensional text data while providing accurate predictions. This approach is widely used in various natural language processing applications, offering efficiency, scalability, and effectiveness in text classification tasks.

```
[68] from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test, z_train, z_test = train_test_split(
    aug_df.processed_text,
    aug_df.processed_text_tag,
    aug_df.label,
    test_size=0.2, # 20% samples will go to test dataset
    random_state=2022,
    stratify=aug_df.processed_text_tag
)
```

```
[69] print("Shape of X_train: ", X_train.shape)
print("Shape of X_test: ", X_test.shape)
```

```
⇒ Shape of X_train: (440,)
Shape of X_test: (110,)
```

```
[70] X_train.head()
```

```
⇒ 213    heap sort connection heap datum structure alig...
151    count sort non comparative integer sort algori...
302    merge phase merge sort key component sort subl...
139    comb sort find application cryptographic algor...
529    ['unique selection sort instability check aspe...
Name: processed_text, dtype: object
```

```
[71] y_train.value_counts()
```

```
⇒ processed_text_tag
4    40
3    40
6    40
2    40
9    40
8    40
1    40
10   40
5    40
0    40
7    40
Name: count, dtype: int64
```

```
[72] y_test.value_counts()
```

```
⇒ processed_text_tag
3    10
6    10
8    10
0    10
1    10
5    10
10   10
9    10
7    10
2    10
4    10
```

```
[73] clf = Pipeline([
    ('vectorizer_tfidf',TfidfVectorizer()),          #using the ngram
    ('Random Forest', RandomForestClassifier())
])
```

```
[74] print(X_train[:5])
```

```
↳ 213 heap sort connection heap datum structure align...
151 count sort non comparative integer sort algorithm...
302 merge phase merge sort key component sort sublin...
139 comb sort find application cryptographic algorithm...
529 ['unique selection sort instability check aspect...
Name: processed_text, dtype: object
```

```
[75] print(X_test[:5])
```

```
↳ 150 count sort linear time sort algorithm operate ...
185 count sort adapt external sort scenario dataset...
330 merge sort stability efficient handling partial...
409 scenario involve variable length key radix sort...
427 scenario involve real time system radix sort a...
Name: processed_text, dtype: object
```

```
[76] #2. fit with X_train and y_train
      clf.fit(X_train, y_train)

      #3. get the predictions for X_test and store it in y_pred
      y_pred = clf.predict(X_test)

      #4. print the classification report
      print(classification_report(y_test, y_pred))
```

```
↳
```

	precision	recall	f1-score	support
0	0.91	1.00	0.95	10
1	1.00	1.00	1.00	10
2	1.00	1.00	1.00	10
3	1.00	1.00	1.00	10
4	1.00	1.00	1.00	10
5	1.00	1.00	1.00	10
6	1.00	0.90	0.95	10
7	1.00	1.00	1.00	10
8	0.90	0.90	0.90	10
9	0.91	1.00	0.95	10
10	1.00	0.90	0.95	10
accuracy			0.97	110
macro avg	0.97	0.97	0.97	110
weighted avg	0.97	0.97	0.97	110

FIG 13: Implementing TF-IDF combined with Random Forest Classification

3.5 KEY CHALLENGES

The "Imbalanced Text Classification with Abstract Feature Extraction using TF-IDF" project introduces several challenges that warrant careful consideration and mitigation strategies. Firstly, addressing the inherent class imbalance in the dataset poses a significant challenge. Text data describing sorting algorithms or any technical domain often results in uneven class distributions, where some algorithms may have substantially fewer instances than others. This imbalance can lead to biased model training, with the model favouring the majority class and potentially neglecting the minority classes. To overcome this, employing effective resampling techniques such as oversampling or under sampling, or utilizing advanced methods like synthetic data generation, becomes crucial. Balancing the class distribution ensures that the model generalizes well across all classes and accurately captures the intricacies of each sorting algorithm. TF-IDF, although a basic technique of natural language processing, has lots of problems in the many stages of its implementation. First of all, at the beginning of data preprocessing, difficulties are encountered in tokenization, for instance, in languages with complex morphology or words that can have different grammar structures. Besides, the process of spotting and getting rid of stop words, which are everywhere but have no meaning, needs to be done in a meaningful way, since their importance depends on the situation or the field of the text. Moreover, the vegetation of words through stemming or lemmatization has its own group of problems, as different languages and dialects may require specific methods to be used to get the word variations accurately. Relocating from the general rule of documentation, TF-IDF usually provides a high-dimensional sparse matrix because of the large number of terms and documents. This feature is the main source of the memory and the computational problems, especially when we talk about the large corpora, as the storage and the processing of such matrices are very expensive. The efficiency of TF-IDF is determined by many factors, for instance, the preprocessing steps, the document collection that is being used as reference, and the parameters settings like the use of frequent terms or the selection of normalization techniques. Therefore, the refinement and testing are crucial to the improvement of the performance and the generalization of the TF-IDF-based models on various tasks and fields. In general, there are some obstacles to be overcome in order to deal with these issues which demand domain expertise, algorithmic advancement, and computational efficiency.

Taking the hurdles and problems into account, we use the TF-IDF/Random Forest Classification technique to get the information from the textual data and to make progress in the related fields of natural language processing and other sciences.

Chapter 4: TESTING

4.1 TESTING STRATEGY

TF-IDF (Term Frequency-Inverse Document Frequency) testing methods are very important in order to verify their effectiveness and reliability in different natural language processing (NLP) tasks. The strategies are a set of methods that will involve the evaluation of the quality of the TF-IDF features, the assessment of their effect on downstream applications, and the validation of their performance across different datasets and scenarios. One basic testing technique is the examination of the quality of the TF-IDF features that are obtained from the documents. This refers to the process of evaluating the highest-ranked terms and their significance to the main documents or topics through the qualitative methods. Through the visual inspection of the extracted features, the experts can evaluate the efficiency of the TF-IDF method in identifying the important terms and eliminating the noise and the irrelevant ones. Moreover, the quantitative evaluation, for instance, comparing TF-IDF ranking with the annotations of experts or gold standard datasets, gives objective evidence of the feature quality and helps to find the discrepancies and the deficiencies of the extracted features. Besides, testing methods for TF-IDF implementations are also related to the assessment of their effect on the downstream applications, which include document classification, clustering, or information retrieval. In these cases, TF-IDF features are input to machine learning models or similarity algorithms affecting the overall performance of these applications. Testing is the process of testing how different TF-IDF parameters, like different weighting strategies or normalization methods, affect the performance of the downstream tasks. By performing the experiments and analysis, the professionals will be able to find the best TF-IDF settings that will make the applications work the most. In addition to the mentioned, cross-validation techniques are also used to check the reliability and the generalizability of TF-IDF implementations over different datasets and domains. The data is divided into several subsets and the TF-IDF models are trained and tested on different subsets to determine the consistency of the features in different data samples, thus, the stability of the extracted features is assessed. Cross-validation is useful for reducing overfitting problems and thus, it offers the insights on the application of TF-IDF-based models in real-world cases, thus, the models can be

deployed in the real world with confidence. In addition, comparing with the existing NLP tools or libraries which already have TF-IDF functionality can give ways to understand the performance and efficiency of the custom TF-IDF code. The comparison of computational metrics such as runtime, memory usage, and scalability will be of great help to the practitioners in the assessment of the competitiveness of their implementations and the identification of the areas in which they can optimize or improve. The evaluation of testing techniques for TF-IDF implementations to be both qualitative and quantitative. That is, the quality of the features has to be measured and the impact on the downstream applications is assessed. Besides, one has to cross-validate the TF-IDFs on different datasets to ensure their generality and compare them to the existing tools to make sure that they are better. Through taking the whole testing method, the experts can be sure that the TF-IDF-based solution is really working, is good and can be used in the different NLP tasks and applications.

4.2 TEST CASES AND OUTCOME

When testing TF-IDF (Term Frequency-Inverse Document Frequency) implementations, it's essential to ensure that the algorithm accurately calculates term weights based on their frequency in documents and their importance across the corpus. Here are some test cases and expected outcomes for TF-IDF:

Test Case 1: Dataset Preparation

Input: Raw text dataset with imbalanced class distribution.

Expected Outcome: After augmentation, the dataset should be balanced, with an equal number of samples for each class.

Test Case 2: Data Splitting

Input: Augmented dataset.

Expected Outcome: The dataset is split into training and testing subsets, maintaining class proportions in each subset.

Test Case 3: TF-IDF Feature Extraction

Input: Training and testing subsets.

Expected Outcome: TF-IDF features are extracted from the text data, generating numerical representations that capture the importance of words in each document.

Test Case 4: Model Training

Input: TF-IDF features, Random Forest Classifier.

Expected Outcome: The pipeline method combines TF-IDF vectorization and Random Forest Classification, resulting in a trained model capable of classifying text data.

Test Case 5: Model Evaluation

Input: Testing subset, trained model.

Expected Outcome: The model accurately classifies text instances in the testing subset, with high precision, recall, and F1-score across all classes.

Test Case 6: Accuracy Calculation

Input: Predicted labels, ground truth labels.

Expected Outcome: The accuracy of the model is calculated by comparing the predicted labels with the ground truth labels, yielding a high accuracy score indicating the model's effectiveness in classification.

Test Case 7: F1-Score Calculation

Input: Predicted labels, ground truth labels.

Expected Outcome: The F1-score of the model is computed, providing a balanced measure of precision and recall across all classes and indicating the model's ability to handle imbalanced datasets.

Overall, the expected outcome of each test case is a well-performing text classification model that effectively handles imbalanced data using augmentation and TF-IDF combined with Random Forest Classification using the pipeline method. The model should demonstrate high accuracy and F1-score, providing reliable predictions for text classification tasks.

Chapter 05: RESULTS

TF-IDF / RANDOM FOREST CLASSIFIER MODEL RESULTS:

The results of a TF-IDF / Random Forest Classifier model using the pipeline method typically include performance metrics that assess the model's effectiveness in classifying text data. These metrics commonly include accuracy, precision, recall, F1-score, and possibly others depending on the specific requirements of the classification task. Here's a breakdown of these metrics and what they signify:

Accuracy: The proportion of correctly classified instances out of the total instances. It provides an overall measure of the model's correctness.

Precision: The proportion of true positive predictions out of all positive predictions made by the model. It indicates the model's ability to avoid false positives.

Recall (Sensitivity): The proportion of true positive predictions out of all actual positive instances in the dataset. It measures the model's ability to capture all positive instances.

F1-score: The harmonic mean of precision and recall, providing a balance between the two metrics. It is especially useful when there is an imbalance between the classes.

These metrics collectively provide insights into different aspects of the model's performance, such as its accuracy, ability to avoid false alarms (precision), ability to capture all relevant instances (recall), and overall balance between precision and recall (F1-score). In the context of a TF-IDF / Random Forest Classifier model using the pipeline method, these metrics would be computed after training and evaluating the model on a separate test dataset. The results would indicate how well the model generalizes to unseen data and its effectiveness in classifying text instances into their respective classes.

Here we obtained the overall accuracy of 97% with application of TF-IDF/Random Forest Classification Model:

```
[76] #2. fit with X_train and y_train
      clf.fit(X_train, y_train)

      #3. get the predictions for X_test and store it in y_pred
      y_pred = clf.predict(X_test)

      #4. print the classification report
      print(classification_report(y_test, y_pred))
```

```

⇌
      precision    recall  f1-score   support

 0         0.91      1.00      0.95         10
 1         1.00      1.00      1.00         10
 2         1.00      1.00      1.00         10
 3         1.00      1.00      1.00         10
 4         1.00      1.00      1.00         10
 5         1.00      1.00      1.00         10
 6         1.00      0.90      0.95         10
 7         1.00      1.00      1.00         10
 8         0.90      0.90      0.90         10
 9         0.91      1.00      0.95         10
10         1.00      0.90      0.95         10

 accuracy          0.97         110
 macro avg         0.97         0.97         0.97         110
 weighted avg     0.97         0.97         0.97         110
```

FIG 14: Precision, Recall, F-1 Score and Accuracy of the Model

Chapter 06: CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSIONS

In conclusion, the research project focusing on "Imbalanced text classification using abstract feature extraction" represents a significant endeavor in the field of natural language processing (NLP). The project addresses a pervasive challenge encountered in various NLP applications, wherein imbalanced datasets hinder the performance of traditional classification models. Through the innovative integration of augmentation techniques and advanced classification methodologies, particularly TF-IDF combined with Random Forest Classification using the pipeline method, this project offers a comprehensive solution to the problem of imbalanced text classification.

The initial phase of the project involved a thorough exploration and analysis of the dataset, revealing significant class imbalances that could potentially bias the learning process of machine learning models. To mitigate this imbalance, augmentation techniques were employed to generate synthetic samples for the minority classes, thereby achieving a more equitable distribution across all classes. This step is pivotal as it ensures that the model learns from a representative dataset, thereby improving its generalization ability and overall performance.

The integration of TF-IDF with Random Forest Classification within a pipeline framework exemplifies a sophisticated approach to feature extraction and classification in NLP tasks. TF-IDF, a widely adopted technique in text mining, effectively captures the importance of words in documents by weighing them based on their frequency and inverse document frequency. This abstract feature representation is then leveraged by the Random Forest Classifier, known for its robustness and ability to handle high-dimensional data, to make accurate predictions.

The utilization of the pipeline method streamlines the entire workflow, from data preprocessing to model evaluation, ensuring efficiency and reproducibility. The pipeline encapsulates multiple steps, including data preprocessing, TF-IDF vectorization, and model

training, into a single entity, simplifying the implementation and maintenance of the classification system.

Evaluation of the model's performance using a diverse range of metrics, such as accuracy, precision, recall, and F1-score, provides comprehensive insights into its efficacy in classifying text instances. These metrics serve as valuable benchmarks for assessing the model's performance across different classes and highlight its strengths and areas for improvement.

In summary, the successful execution of this project underscores the importance of leveraging advanced techniques in NLP to address complex challenges such as imbalanced text classification. By combining augmentation with TF-IDF and Random Forest Classification, the project not only offers a practical solution to imbalanced datasets but also contributes to the advancement of research in NLP. The findings and methodologies developed in this project have broad implications for various real-world applications, including sentiment analysis, spam detection, and content categorization, underscoring its significance in the field of artificial intelligence and machine learning.

APPLICATIONS:

1. **Medical Text Classification:** Medical text categorization encompasses classifying electronic health records and medical papers into categories such as disease diagnosis, treatment plans, and specific fields of medicine. For optimal information searching through text retrieval by healthcare professionals this has to be enhanced in improving the accuracy of medical text classification.
2. **Legal Document Classification:** Classifying legal documents according to classes that include different contract types, legal opinions, and case summaries. Efficient legal document management, including search and proper classification of numerous types of legal texts.
3. **Sentiment Analysis in Customer Reviews:** Sentiment polarity analysis of customer reviews, particularly in unbalanced situations when the number of negative sentiments is low. Developing models of sentiment analysis to pinpoint relevant negative sentiments for businesses so that they resolve customers' grievances efficiently.

4. **Fraud Detection in Financial Texts:** Fraud detection in financial documents, text classification (legitimate vs. fraudulent). Detecting fraud in imbalanced data sets and for more accurate identification of suspicious financial activities.
5. **News Article Classification:** News categorization into different topics or themes and some topics might appear fewer than others. To enhance classification of news for personalized content suggestions and effective search of rare news topics.
7. **Spam Email Detection:** Detecting spam emails among imbalanced datasets, when legitimate emails are highly populated. Improving the effectiveness of spam detection models and reducing false positives through improved email filtering systems.
8. **Social Media Content Moderation:** Classification of posts on the social media platforms as offensive, non-offensive, or potentially harmful for moderation. Accurately locating and dealing with inappropriate information on social media supports social security.

6.2 FUTURE SCOPE

The project on "Imbalanced text classification using abstract feature extraction" opens up several avenues for future research and development in the field of natural language processing (NLP) and machine learning. Some potential areas for future exploration and enhancement include:

Advanced Augmentation Techniques: Further investigation into augmentation techniques tailored specifically for text data could lead to the development of more sophisticated methods for generating synthetic samples. Techniques such as back-translation, contextual augmentation, or adversarial training could be explored to improve the diversity and quality of augmented data.

Feature Engineering: Experimentation with alternative feature extraction methods beyond TF-IDF, such as word embeddings (e.g., Word2Vec, GloVe) or contextual embeddings (e.g., BERT, GPT), could offer new insights into capturing semantic information in text data. These embeddings may provide richer representations of text documents, potentially enhancing the performance of classification models.

Ensemble Learning Approaches: Investigating ensemble learning techniques, such as stacking or boosting, for combining multiple classification models could lead to further improvements in classification accuracy and robustness. Ensemble methods have the potential to harness the strengths of different models and mitigate their individual weaknesses, resulting in more robust and reliable classification systems.

Deep Learning Architectures: Exploring deep learning architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), for text classification tasks could offer new opportunities for capturing complex patterns and relationships in text data. These architectures may be particularly effective for handling sequential or contextual information in text documents.

Transfer Learning: Leveraging pre-trained language models and transfer learning techniques could facilitate the development of more efficient and effective text classification models, especially in scenarios with limited labeled data. Fine-tuning pre-trained models on domain-specific datasets or utilizing techniques like domain adaptation could help adapt these models to specific classification tasks.

Model Interpretability and Explainability: Enhancing the interpretability and explainability of classification models could improve their trustworthiness and usability in real-world applications. Techniques for interpreting model predictions, identifying influential features, or generating explanations for model decisions could be explored to provide insights into the classification process.

Real-world Applications: Extending the application of the developed classification model to real-world scenarios and domains, such as social media analysis, healthcare, finance, or cybersecurity, could demonstrate its practical utility and impact. Customizing the model to specific domains and evaluating its performance in diverse contexts could provide valuable insights for deployment in real-world settings. Overall, the project lays the foundation for future research and innovation in imbalanced text classification and NLP, offering numerous opportunities for further exploration and advancement. By addressing these future research directions, researchers and practitioners can continue to push the boundaries of text classification technology and contribute to the development of more intelligent and adaptive systems.

REFERENCES

- [1] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining Text Data*, Springer, Boston, 2012, pp. 163–222, doi: 10.1007/978-1-4614-3223-4_6.
- [2] H. Gao et al., "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst Appl*, vol. 73, pp. 220–239, 2017, doi: 10.1016/j.eswa.2016.12.035.
- [3] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans Knowl Data Eng*, vol. 21, no. 9, pp. 1263–1284, 2009, doi: 10.1109/TKDE.2008.239.
- [4] M. J. Kowsari et al., "Text classification algorithms: A survey," *Inf (Basel)*, vol. 10, no. 4, p. 150, 2019, doi: 10.3390/info10040150.
- [5] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 40, no. 3, pp. 211–218, 2006.
- [6] T. Dogan and A. K. Uysal, "Improved inverse gravity moment term weighting for text classification," *Expert Syst Appl*, vol. 130, pp. 45–59, 2019, doi: 10.1016/j.eswa.2019.04.015.
- [7] J. Ortigosa-Hernandez, I. Inza, and J. A. Lozano, "Measuring the class-imbalance extent of multi-class problems," *Lett*, vol. 98, pp. 32–38, 2017, doi: 10.1016/j.patrec.2017.08.002.
- [8] T. Dogan and A. K. Uysal, "Improved inverse gravity moment term weighting for text classification," *Expert Syst Appl*, vol. 130, pp. 45–59, 2019, doi: 10.1016/j.eswa.2019.04.015.
- [9] J. Wang and Y. Dong, "Measurement of text similarity: A survey," *Information*, vol. 11, no. 9, p. 421, Aug. 2020, doi: 10.3390/info11090421.

- [10] S. Aryal, K. M. Ting, T. Washio, and GHaffari, "A new simple and effective measure for bag-of-word inter-document similarity measurement," 2019, arXiv:1902.03402. doi:10.48550/arXiv.1902.03402
- [11] G. Grefenstette, "Tokenization," in *Syntactic Wordclass Tagging*, Springer, 1999, pp. 117–133, doi: 10.1017/S1351324997001502.
- [12] Vishwanath Bijalwan et al., "KNN based Machine Learning Approach for Text and Document Mining," *International Journal of Database Theory and Application*, vol. 1, no. 7, pp. 61–70, 2014, doi: 10.48550/arXiv.1406.1580.
- [13] Albitar S, Fournier S, Espinasse B. An Effective TF/IDF-Based Text-to-Text Semantic Similarity Measure for Text Classification[M]// *Web Information Systems Engineering – WISE 2014*. Springer International Publishing, 2014:105-114 dio: 10.1007/978-3-319-11749-2_8
- [14] Zhengdong Lu 'Cane Wing-ki Leung' Qiang Yang 'Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining 'Chicago' August 11 – 14' 2013[C]' Chicago:KDD2013' 2013'
- [15] Lauren R ' Biggers ' Cecylia Bococich ' Riley Capshaw ' et al ' Configuring latent Dirichlet allocation based feature location [J] 'Empirical Software Engineering' 2014' 3(19) .465-500'
- [16] Yuan S, Qian Z. Tibetan-Chinese Cross Language Text Similarity Calculation Based on LDA Topic Model[J]. *Open Cybernetics & Systemics Journal*, 2015, 9(1):2911-2919.
- [17] Mathmoud Mejdoub' Chokri Ben Amar. Classification improvement of local feature vectors over the KNN algorithm[J] 'Springer Link' 2013' 5(64) .197-218'
- [18] Wang Meng' Lin Lanfen' Wang Jing' et al 'Improving Short Text Classification Using Public Search Engines[J] 'Integrated Uncertainty in Knowledge Modelling and Decision Making' 2013' vol 8032. 157-166'
- [19] Tang Hong' Shen Li' Qi Yinfeng' et al 'A Multiscale Latent Dirichlet Allocation Model for Object-Oriented Clustering of VHR Panchromatic Satellite Images[J] 'IEEE Transactions on Geoscience and Remote Sensing' 2013, 51(3).1680-1692'

- [20] Ni C, Leung C C. Investigation of using different Chinese word segmentation standards and algorithms for automatic speech recognition[C]// International Symposium on Chinese Spoken Language Processing. IEEE, 2014:44-48
- [21]Zhang X,Zhao J,LeCun Y.Character-level convolutional networks for text classification[J].In Advances in neural information processing systems,2015,arXiv:1502-01710.
- [22] Devlin J,Chang M W,Lee K,et al.Bert:Pretraining of deep bidirectional transformers for language understanding[J]. arXiv preprint,2018: 1810-04805.
- [23] Chen Baixue, song Peiyan. TF-IDF Assisted Indexing Based on user natural annotation Algorithm and empirical research [J]. Library and information work, 2018,62 (1): 132-139.
- [24] Wang Xing, Liu Wei. Automatic indexing method of Chinese academic documents based on Citation Research[J]. Library and information work, 2014,58 (3): 106-110105.
- [25] Liu Siqin, Feng XURUI. Text emotion analysis based on Bert [J]. Information security research,

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at..... (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none">• All Preliminary Pages• Bibliography/Images/Quotes• 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Major 24

ORIGINALITY REPORT

18%

SIMILARITY INDEX

12%

INTERNET SOURCES

14%

PUBLICATIONS

7%

STUDENT PAPERS

PRIMARY SOURCES

- 1** Weihua Chen, Xian Zhang. "Research on text categorization model based on LDA — KNN", 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2017
Publication 1%
- 2** [ijmtst.com](#)
Internet Source 1%
- 3** [medium.com](#)
Internet Source 1%
- 4** Submitted to University of North Texas
Student Paper 1%
- 5** Cristian Padurariu, Mihaela Elena Breaban. "Dealing with Data Imbalance in Text Classification", Procedia Computer Science, 2019
Publication 1%
- 6** Zhao, Dexin, Jinqun He, and Jin Liu. "An improved LDA algorithm for text classification", 2014 International Conference 1%

on Information Science Electronics and
Electrical Engineering, 2014.

Publication

7	Submitted to University of Sunderland Student Paper	1 %
8	Cai-zhi Liu, Yan-xiu Sheng, Zhi-qiang Wei, Yong-Quan Yang. "Research of Text Classification Based on Improved TF-IDF Algorithm", 2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE), 2018 Publication	1 %
9	coek.info Internet Source	1 %
10	www.mdpi.com Internet Source	1 %
11	Submitted to Notre Dame of Marbel University Student Paper	<1 %
12	Submitted to Mepco Schlenk Engineering college Student Paper	<1 %
13	Muhammad Farrukh Bashir, Hamza Arshad, Abdul Rehman Javed, Natalia Kryvinska, Shahab S. Band. "Subjective Answers Evaluation Using Machine Learning and	<1 %

Natural Language Processing", IEEE Access, 2021

Publication

14 www.researchgate.net <1 %
Internet Source

15 Jian-Wei Sun, Jia-Qi Bao, Li-Ping Bu. "Text Classification Algorithm Based on TF-IDF and BERT", 2022 11th International Conference of Information and Communication Technology (ICTech)), 2022 <1 %
Publication

16 assets.researchsquare.com <1 %
Internet Source

17 www.scpe.org <1 %
Internet Source

18 ourspace.uregina.ca <1 %
Internet Source

19 Submitted to Adama Science and Technology University <1 %
Student Paper

20 Submitted to South Bank University <1 %
Student Paper

21 dergipark.org.tr <1 %
Internet Source

22 www.coursehero.com <1 %
Internet Source

23	www.ijraset.com Internet Source	<1 %
24	elibrary.stipram.ac.id Internet Source	<1 %
25	Submitted to University of Essex Student Paper	<1 %
26	journals.nubip.edu.ua Internet Source	<1 %
27	www.frontiersin.org Internet Source	<1 %
28	Murat Okkalioglu, Burcu Demirelli Okkalioglu. "AFE-MERT: imbalanced text classification with abstract feature extraction", Applied Intelligence, 2022 Publication	<1 %
29	Submitted to Xiamen University Student Paper	<1 %
30	"Mobile Radio Communications and 5G Networks", Springer Science and Business Media LLC, 2024 Publication	<1 %
31	onlineresource.ucsy.edu.mm Internet Source	<1 %
32	technodocbox.com Internet Source	<1 %

33	Kwan Yi. "A semantic similarity approach to predicting Library of Congress subject headings for social tags", Journal of the American Society for Information Science and Technology, 2010 Publication	<1 %
34	journals.uran.ua Internet Source	<1 %
35	link.springer.com Internet Source	<1 %
36	Submitted to Nanyang Polytechnic Student Paper	<1 %
37	cdn.techscience.cn Internet Source	<1 %
38	dokumen.pub Internet Source	<1 %
39	www.researchsquare.com Internet Source	<1 %
40	www2.mdpi.com Internet Source	<1 %
41	Lecture Notes in Computer Science, 2014. Publication	<1 %
42	Submitted to Liverpool John Moores University Student Paper	<1 %

43	liu.diva-portal.org Internet Source	<1 %
44	"Natural Language Understanding and Intelligent Applications", Springer Nature, 2016 Publication	<1 %
45	Submitted to De Montfort University Student Paper	<1 %
46	Submitted to Heriot-Watt University Student Paper	<1 %
47	Submitted to La Trobe University Student Paper	<1 %
48	Submitted to Rivier University Student Paper	<1 %
49	Submitted to Turun yliopisto Student Paper	<1 %
50	www.ijert.org Internet Source	<1 %
51	Submitted to AUT University Student Paper	<1 %
52	Submitted to University of Bristol Student Paper	<1 %
53	Submitted to University of Salford Student Paper	<1 %

54	ijsred.com Internet Source	<1 %
55	repository.ju.edu.et Internet Source	<1 %
56	www.grafiati.com Internet Source	<1 %
57	ds.inflibnet.ac.in Internet Source	<1 %
58	fastercapital.com Internet Source	<1 %
59	"Speech and Computer", Springer Science and Business Media LLC, 2016 Publication	<1 %
60	Submitted to Federal University of Technology-Nigeria Student Paper	<1 %
61	article.sciencepublishinggroup.com Internet Source	<1 %
62	www.appinio.com Internet Source	<1 %
63	www.speech.sri.com Internet Source	<1 %
64	Submitted to Arts, Sciences & Technology University In Lebanon Student Paper	<1 %

65

Babeş-Bolyai University

Publication

<1 %

66

Frank P. DiMaio. "Spherical-harmonic decomposition for molecular recognition in electron-density maps", International Journal of Data Mining and Bioinformatics, 2009

Publication

<1 %

67

Ksenia Lagutina, Nadezhda Lagutina. "A Survey of Models for Constructing Text Features to Classify Texts in Natural Language", 2021 29th Conference of Open Innovations Association (FRUCT), 2021

Publication

<1 %

68

docplayer.net

Internet Source

<1 %

69

iq.opengenus.org

Internet Source

<1 %

70

journals.plos.org

Internet Source

<1 %

71

"Image Analysis and Recognition", Springer Science and Business Media LLC, 2017

Publication

<1 %

72

Gene M. Ko, A. Srinivas Reddy, Sunil Kumar, Barbara A. Bailey, Rajni Garg. "Classification of HIV-1 protease crystal structures using Random Forest, linear discriminant analysis

<1 %

and logistic regression", 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, 2010

Publication

73

Habibe Karayiğit, Çiğdem İnan Acı, Ali Akdağlı. "Detecting abusive Instagram comments in Turkish using convolutional Neural network and machine learning methods", Expert Systems with Applications, 2021

Publication

<1 %

74

Lecture Notes in Computer Science, 2015.

Publication

<1 %

75

Lecture Notes in Computer Science, 2016.

Publication

<1 %

76

Lucia Liu, Ameth Guevara, Javier E. Sanchez-Galan. "Identification and Classification of Road Traffic Incidents in Panama City Through the Analysis of a Social Media Stream and Machine Learning", Intelligent Systems with Applications, 2022

Publication

<1 %

77

Shenghan Zhou, Chaofan Wei, Pan Li, Anying Liu, Wenbing Chang, Yiyong Xiao. "A Text-Driven Aircraft Fault Diagnosis Model Based on Word2vec and Stacking Ensemble Learning", Aerospace, 2021

Publication

<1 %

78 Wenli Shan. "Optimal pricing strategies and decision-making systems in e-commerce using integrated fuzzy multi-criteria method", *Expert Systems*, 2023
Publication <1 %

79 api.repository.cam.ac.uk
Internet Source <1 %

80 bora.uib.no
Internet Source <1 %

81 export.arxiv.org
Internet Source <1 %

82 hdl.handle.net
Internet Source <1 %

83 ijercse.com
Internet Source <1 %

84 mdpi-res.com
Internet Source <1 %

85 peerj.com
Internet Source <1 %

86 projet.liris.cnrs.fr
Internet Source <1 %

87 swathivegirajumth522.sites.umassd.edu
Internet Source <1 %

88 www.ijeat.org
Internet Source <1 %

89

www.scirp.org

Internet Source

<1 %

90

"Emerging Technologies in Data Mining and Information Security", Springer Science and Business Media LLC, 2019

Publication

<1 %

91

"Machine Learning and Knowledge Discovery in Databases", Springer Science and Business Media LLC, 2020

Publication

<1 %

92

Aarav Mulinti, Guillermo Goldsztein. "Sentiment Analysis to Identify Consumer Criticism of Artificial Intelligence: A ChatGPT Case Study", Journal of Student Research, 2023

Publication

<1 %

93

Advances in Intelligent Systems and Computing, 2016.

Publication

<1 %

94

Jaikishan Jaikumar, Mohana, Pavankumar Suresh. "Privacy-Preserving Personal Identifiable Information (PII) Label Detection Using Machine Learning", 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2023

Publication

<1 %

95

Jantima Polpinij, Khanista Namee. "Improving of Imbalanced Data in Multiclass Classification for Sentiment Analysis using Supervised Term Weighting", 2021 Research, Invention, and Innovation Congress: Innovation Electricals and Electronics (RI2C), 2021

Publication

<1 %

96

Miha Pavlinek, Vili Podgorelec. "Text classification method based on self-training and LDA topic models", Expert Systems with Applications, 2017

Publication

<1 %

97

Zhiying Jiang, Bo Gao, Yanlin He, Yongming Han, Paul Doyle, Qunxiong Zhu. "Text Classification Using Novel Term Weighting Scheme-Based Improved TF-IDF for Internet Media Reports", Mathematical Problems in Engineering, 2021

Publication

<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography On