UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

Deep Learning Techniques for Hate Speech Detection on Twitter During the 2022 National Strike in Ecuador.

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingenieria de Tecnologías de la Información y la Comunicación

Autor:

Menoscal Saltos Wilter José

Tutor:

Phd. Cuenca Pauta Erick Eduardo

Co-Tutora:

Mgtr. Escobar Cordova Silvana Karina

Urcuquí, Abril 2024

Autoría

Yo, **WILTER JOSE MENOSCAL SALTOS**, con cédula de identidad **1250948617**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Abril 2024.

Wilter Jose Menoscal Saltos

CI: 1250948617

Autorización de publicación

Yo, **WILTER JOSE MENOSCAL SALTOS**, con cédula de identidad 1250948617, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, Abril 2024.

Firmado electrónicamente por:
WILTER JOSE
MENOSCAL SALTOS

Wilter José Menoscal Saltos

CI: 1250948617

Dedication

To my family, which has been an indispensable support in my studies, this work is for them and for my girlfriend and friends, who have been a fundamental pillar in this path.
This work is also thanks to my professors, who have given all their effort and knowledge to the students, creating great professionals.

Wilter José Menoscal Saltos

Acknowledgment

I am very grateful to my family, in especially my mother Ingrid and my brother Diego, for being my unwavering support during my stay at the University. For their support, for believing in me and for their patience, which were great reasons to finish my studies. I thank my girlfriend Nathalia Ortega, for being with me several sleepless nights doing the thesis, for giving me her love throughout this journey, for being a strong pillar in this process. I thank my friends, especially Nelly, Stefany, Ricardo and Leo, for being my greatest friends throughout this journey, for giving me their friendship and knowledge all this time. Finally, a mention to my tutor Erick and co-tutor Silvana, for their constant interest and work for this degree work.

Wilter José Menoscal Saltos

Resumen

Ecuador pasó por un paro nacional en el año 2022, esto se dio por varias razones políticas, donde, dada la movilización por parte de las comunidades indígenas, se pasó una crisis alimentaria afectando a la mayoría parte del país. A esto la gente empezó a generar odio hacia estas comunidades. Este estudio está enfocado en desarrollar varios modelos de detección de discurso de odio, en diferentes ramas de la inteligencia artificial, tanto en el campo de machine learning y deep learning. Esto con el fin de crear un modelo que sea capaz de detectar discurso de odio sobre las comunidades indígenas en Ecuador. Se hizo un estudio en profundidad del rendimiento de todos los modelos, para finalmente tomar el mejor modelo de todos los planteados, con el fin de poder usar este modelo en cualquier plataforma y así detectar y persuadir a estos discursos de odio.

Palabras Clave:
Sequential Neural Network(SNN), Logistic Regression (LR), Support Vector Machine(SVM), Random Forest(RF), Naive Bayes (NB), Bidirectional Encoder Representations from Transformers (BERT).

Abstract

Ecuador went through a national strike in 2022, which occurred for several political reasons. Given the mobilization by indigenous communities, a food crisis affected most of the country. This led people to generate hatred towards these communities. This study focuses on developing various hate speech detection models in different branches of artificial intelligence, both in the fields of machine learning and deep learning. The aim is to create a model capable of detecting hate speech against indigenous communities in Ecuador. A thorough study of the performance of all models was conducted to finally select the best model among them, in order to use this model on any platform to detect and counteract such hate speech.

Keywords:
Sequential Neural Network(SNN), Logistic Regression (LR), Support Vector Machine(SVM), Random Forest(RF), Naive Bayes (NB), Bidirectional Encoder Representations from Transformers (BERT).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Social media has evolved since its inception, transforming how we communicate, interact, and share information online. Its impact spans from personal to professional realms, with undeniable influence on society and culture [4].

The use of social media has exponentially grown over the years, with billions of active users on platforms such as Facebook, Twitter, Instagram, and LinkedIn, among others [5]. These platforms offer various services, from connecting with friends and family to professional networking, entertainment, education, and social activism.

Among these platforms, Twitter stands out as a microblogging social network, where users can post short messages called "tweets" to express their thoughts, ideas, news, or share content [6]. With over 556 million monthly active users, Twitter offers unique features that distinguish it from other social networks. Its concise format of 280 characters per tweet fosters brevity and clarity in communication, making it an ideal stage for sentiment analysis [7]. Additionally, hashtags facilitate the organization and search of specific topics, while mentions allow direct interaction between users.

Sentiment analysis (SA) is a crucial discipline in social networks. It involves analyzing and understanding users' opinions, emotions, and attitudes through their online posts. SA utilizes various technologies such as natural language processing (NLP), machine learning algorithms, and data mining techniques to extract meaningful information from large volumes of data generated on social networks [8].

SA on social networks serves a wide range of purposes. It enables companies and organizations to monitor public perception of their products, services, or marketing campaigns, providing valuable insights for decision-making. Furthermore, researchers use sentiment analysis on Twitter to study social trends, assess the impact of online events, and better understand human behavior in the digital world [9].

One of the prominent aspects of sentiment analysis on social networks is the detection and study of hate speech, also known as "hate speech." This phenomenon refers to the expression of messages or comments that promote hatred, discrimination, or violence towards individuals or groups based on characteristics such as race, religion, and sexual orientation, among others [10]. Identifying and mitigating hate speech on Twitter and other social platforms has become a significant challenge, with significant implications for online security and the protection of human rights.

In this context, during 2022, Ecuador was embroiled in a political and insecurity crisis, leading to a national mobilization led by the Confederation of Indigenous Nationalities of Ecuador (CONAIE) from June 13 to June 30.

The mobilization was motivated by the imposition of neoliberal policies by the government, which led to the worsening of poverty, the reduction of the overall state budget in Health and Education, the decrease in social policies for social equality, the increase in insecurity and violence in several areas of the country, the aggressive imposition of extractive policies and activities (mining and oil), the violation of collective rights of indigenous and Afro-descendant peoples; and overall, the lack of guarantees for the exercise of economic, social, and cultural rights of the population [11].

National food scarcity was generated during this mobilization due to road closures in critical locations, such as the passage from one province to another. There was also a negative impact on the country's economy due to the national paralysis; businesses and large industries lost significant money[12]. All of this generated discontent among the population, leading to a series of discussions on the Twitter platform, where there were many people both in favor and against the strike, generating hate speeches towards the entities involved in the mobilization, namely, the government and indigenous communities.

1.2 Problem statement

For years, various NLP technologies for hate speech detection have been evolving. Among these, the most prominent are machine learning and deep learning techniques [13].
Machine learning (ML) and deep learning (DL) are pivotal technologies in sentiment analysis, enabling the development of robust hate speech detection systems [14]. ML algorithms learn from data to identify patterns and make predictions, while DL, a subset of ML, utilizes neural networks with multiple layers to process complex data [15]. These technologies offer sophisticated solutions for analyzing text data and detecting subtle linguistic cues indicative of hate speech.

Several models of these two techniques will be used to detect hate generated toward indigenous communities during the event in Ecuador in 2022. This aims to analyze and produce a model that detects hateful texts towards these communities. The models to be used for the ML technique are support vector machine, logistic regression, random forest, and naive Bayes. The models to be used for the DL technique are based on a sequential neural network and Bidirectional Encoder Representations from Transformers. We aim to explore various approaches to identify and mitigate hate speech targeting these marginalized communities effectively. This comprehensive approach will not only improve our understanding of the prevalence and nature of hate expressed in social networks, but it will also contribute to developing more accurate and reliable detection models to address the problems of online hate speech in diverse linguistic and cultural contexts.

1.3 Objectives

1.3.1 General Objective

To analyze and detect the hatred generated towards indigenous communities on Twitter during the 2022 national strike in Ecuador.

1.3.2 Specific Objectives

- Manually label the obtained tweets in “hate” and “no-hate”.
- Implement different hate speech detection models, such as, Regression logistic, Support

Vector Machine, Random Forest, Naive Bayes, Sequential Neural Network, and Bidirectional Encoder Representations from Transformers.

- Select and test the model with the lowest error value.

Chapter 2

Theoretical Framework

This section delves into various fundamental concepts, encompassing hate speech and natural language processing. To fully grasp the essence of these concepts, it is imperative to delve into their origins. Therefore, a concise exploration of the underpinnings from which these models spring will be presented. The discourse will briefly touch upon artificial intelligence and its subsequent fields to establish a more comprehensive understanding, particularly in diverse applications.

Moving forward, the crucial data preprocessing stage lies within natural language processing (NLP). Furthermore, this section furnishes theoretical foundations on distinct learning models, encompassing machine and deep learning, which have extensive applications in NLP. Each of these models possesses specific criteria that delineate their characteristics. A brief differentiation between machine learning and deep learning models will be elucidated. Lastly, the section will establish diverse methodologies for evaluating each model, providing a comprehensive framework for understanding their effectiveness and nuances.

2.1 Artificial Intelligence

Artificial intelligence (AI) is a field of computer science that focuses on creating systems and programs that can perform tasks that require intelligence when performed by humans. These tasks include learning, reasoning, perception, natural language processing, and decision-making [16]. AI seeks to replicate or simulate the cognitive capacity of humans in machines and computer systems. Its beginnings date back to the 1950s, with pioneers such as Alan Turing and the Dartmouth Conference in 1956 [17].

AI has experienced ups and downs over the decades, from rule-based systems to the current rise of machine learning and big data processing [18]. Today, AI has transformed technology, enabling machines to learn and perform speech recognition, computer vision, and autonomous decision-making tasks.

Several branches derive from artificial intelligence, among which are:

- **Machine Learning:** Machine Learning is one of the most prominent branches of AI. It focuses on developing algorithms and models that enable machines to learn patterns and perform tasks without direct human intervention. It encompasses approaches such as supervised, unsupervised, and reinforcement learning [19].

- **Artificial Neural Networks and Deep Learning:** Within machine learning, Deep Learning stands out by using artificial neural networks with multiple layers (deep neural networks) to learn complex data representations [20]. This approach has driven significant advances in natural language processing, computer vision, and more.

- **Natural Language Processing:** NLP deals with the interaction between computers and human language. It seeks to understand and generate natural language, enabling machines to interpret and respond effectively to texts and conversations [21].

- **Computer Vision:** This branch focuses on training machines to interpret and understand visual information [22], mimicking the human ability to recognize patterns and objects in images and videos.

- **Robotics:** AI in robotics aims to develop autonomous machines capable of performing complex physical and cognitive tasks [23]. It includes path planning, environment perception, and motion control.

- **Expert Systems:** These systems employ rules and expert knowledge to make decisions in specific areas [24]. They are instrumental in fields like medical diagnosis and technical support.

- **Pattern Recognition:** It focuses on identifying patterns and relationships in complex datasets, essential in facial recognition and data mining [25].

2.2 Machine Learning

Machine Learning (ML) involves systematically exploring algorithms and statistical models employed by computer systems to execute specific tasks without the need for explicit programming [26]. Learning algorithms play a pivotal role in numerous applications we encounter daily. For instance, when utilizing a web search engine like Google to navigate the internet, its impressive performance is partly attributed to a learning algorithm adept at prioritizing web page rankings. These algorithms find utility across diverse domains, including data mining, image manipulation, and predictive analysis [27]. A key advantage of harnessing machine learning lies in its capacity to autonomously execute tasks once it has comprehended how to process data. It is essential to mention that the machine learning algorithm's general purpose is to learn critical input data characteristics. Machine learning is divided into several branches [28], and each unit has different algorithms that are most used in this branch, as shown in Figure 2.1.

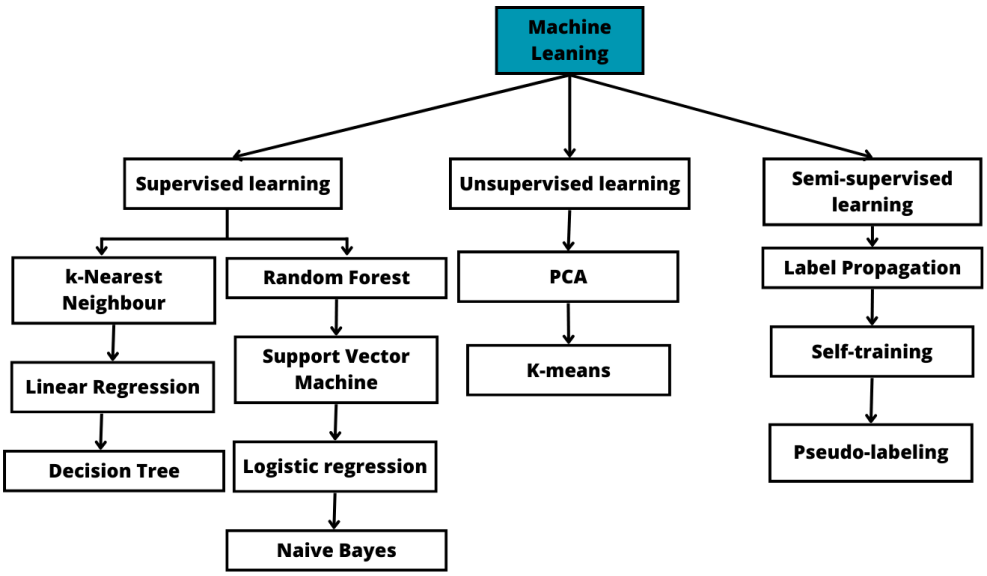

Figure 2.1: Machine Learning branches with some different algorithms

2.2.1 Supervised Learning

Supervised learning is an approach in machine learning where a model is trained to make predictions or decisions based on previously labeled examples. In this approach, a training dataset is provided to the model, containing input examples along with corresponding

correct outputs [29].

The training process in supervised learning involves adjusting the model's parameters so that it can learn to map inputs to outputs appropriately[30]. The model aims to generalize from the training examples and accurately predict new and unseen data. Supervised learning is divided into two main categories:

1. **Classification:** In this category, the model is trained to predict membership in one or multiple classes . For example, they classify emails as "spam" or "not spam" or identify behaviors of cats or dogs. Such as the work of Franzoni, Valentina, and Milani [31].

2. **Regression:** In this case, the model is trained to predict a continuous numerical value. For instance, indicating the price of a house based on features like size and location or estimating the time it will take to complete a task based on specific variables. For example, Manasa, J, and Gupta's work predicts a house's price using different features [32].

It is essential to be clear that the data to be used should be divided into three parts:

- **Train dataset:** The training dataset is a subset of data used for training a machine learning model. It is typically the most significant data used to teach the model to make predictions or classifications [33].

- **Test dataset:** The testing dataset is a separate subset of data used to evaluate the performance of a trained machine-learning model. It is not seen by the model during training and is used to assess how well the model generalizes to new, unseen data [34].

- **Validation dataset:** The validation dataset is an additional subset of data used during model development. It helps to tune hyperparameters and prevent overfitting by objectively evaluating the model's performance during training [35].

Then, the typical process of supervised learning involves the following steps:

1. Collecting labeled training data.

2. Splitting the data into training and testing sets.

3. Selecting an appropriate model for the task.

4. Training the model using the training set.

5. Tuning the model's parameters to minimize prediction errors.

6. Evaluating the model using the test set to measure its performance on unseen data.

Supervised learning finds extensive applications in various domains, including speech recognition, natural language processing, computer vision, medicine, and more, where labeled data allows models to learn and perform specific tasks with high accuracy[36].

After knowing how this works, defining the different models shown in Figure 2.1 is essential.

Model: Decision Trees

A decision tree is a predictive modeling technique in machine learning and data mining. It's a tree-like structure where internal nodes represent features, branches represent decisions, and leaf nodes represent outcomes [37].

It is important to know how this model works, as mentioned in the paper by Charbuty and Abdulazeez [38]:

1. **Tree Construction**: The decision tree is built recursively by splitting the dataset into subsets based on the values of input features. The splitting process aims to maximize the homogeneity of the target variable in the resulting subsets.

2. **Decision Making**: A decision is made based on a feature's value at each internal node. This decision determines which branch to follow down the tree.

3. **Leaf Nodes**: When a leaf node is reached, it represents the predicted outcome or class label.

4. **Training and Pruning**: Decision trees can become overly complex and prone to overfitting, especially with noisy data. Pruning techniques are often applied to simplify the tree and improve its generalization performance.

Applications

1. **Classification**: Decision trees are commonly used for classification tasks, where the goal is to predict the class label of a sample based on its features.

2. **Regression**: Decision trees can also be used for regression tasks, where the goal is to predict a continuous value.

3. **Risk Assessment**: Decision trees are used in various fields for risk assessment and decision-making.

4. **Medical Diagnosis**: Decision trees are used in healthcare for medical diagnosis and treatment planning.

5. **Customer Relationship Management (CRM)**: Decision trees are used in CRM systems to segment customers based on their characteristics and behaviors.

Decision trees are versatile and interpretable models widely used in various domains due to their simplicity, ease of understanding, and ability to handle both categorical and numerical data.

Model: Random Forest

The Random Forest model is an ensemble learning method that combines multiple decision trees to create a more robust and accurate predictive model. Each decision tree in the ensemble is built independently and makes its predictions, and the final prediction is determined by aggregating the predictions of all the individual trees [39].

Given a dataset with input features and corresponding labels, as shown in Figure 2.2, a Random Forest is constructed by random data selection, random feature selection, decision tree construction, and voting for prediction[40].

- **Random Selection of Data:** A random subset of the dataset is selected for each tree using techniques like bootstrapping (sampling with replacement). This ensures diversity among the trees.

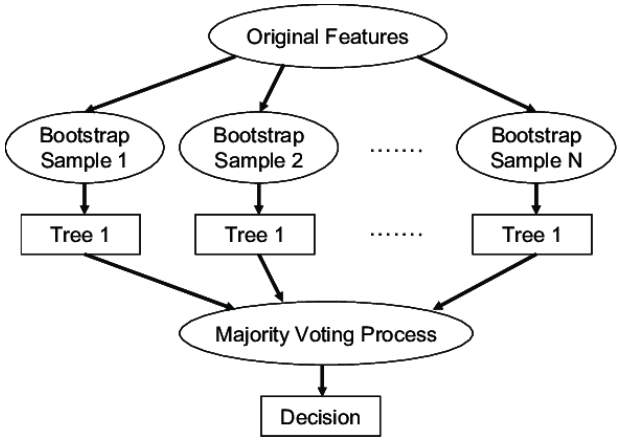

Figure 2.2: Random Forest [1]

- **Random Feature Selection:** At each decision tree node, only a random subset of features is considered for splitting. This helps decorate the trees and improve the model's overall performance.
- **Decision Tree Construction:** Each tree is constructed by recursively partitioning the data based on the selected features until a predefined stopping criterion is met, such as a maximum depth or a minimum number of samples in a leaf node.
- **Voting for Prediction:** During prediction, each tree in the Random Forest makes its prediction. The projections are averaged for regression tasks, and for classification tasks, a majority voting scheme is applied to determine the final class.

Using randomness in data and feature selection helps create a diverse set of trees, leading to a more robust model that is less prone to overfitting. Random Forests are known for their high predictive accuracy and generalization to new, unseen data.

Applications of Random Forest

- Image classification and object recognition.
- Fraud detection in financial transactions.
- Predictive maintenance in manufacturing.
- Bioinformatics for gene expression analysis.

The Random Forest model's ability to handle various data types, feature importance ranking, and resistance to overfitting makes it a popular and versatile choice in machine learning applications.

Model: Support Vector Machine (SVM)

Support Vector Machine is a supervised learning model for classification and Regression. Its primary goal is to find a hyperplane in the feature space that best separates data points from different classes. The hyperplane, known as support vectors, is selected to maximize the distance between the closest data points from different classes [41].

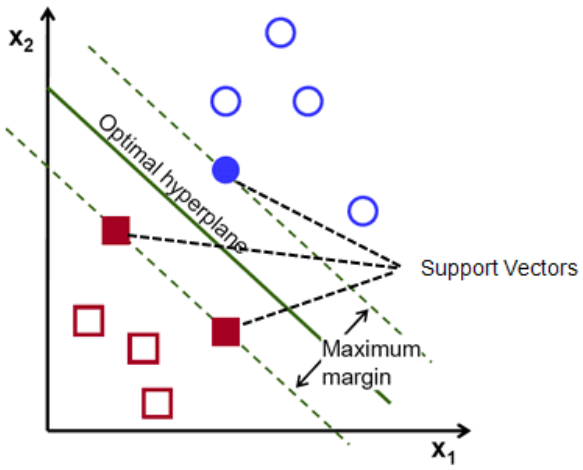

Figure 2.3: Support Vector Machine [2]

It is important to know how SVM works, as mention in the paper by Campbell and Ying [42]:

1. **Problem Formulation:**
    - SVM is primarily used for classification and regression problems. In the context of binary classification, it aims to find a hyperplane that effectively separates two classes in the feature space.
2. **Optimal Hyperplane Selection:**
    - SVM seeks the hyperplane that maximizes the margin between classes. The margin is the perpendicular distance from the hyperplane to the closest points of both classes (support vectors).

3. **Support Vectors:**

    - Support vectors are the data points closest to the hyperplane. These points are crucial in determining the optimal hyperplane and, thus, the model's predictive capacity.

4. **Hinge Loss Function:**

    - SVM uses the hinge loss function to penalize misclassifications. The hinge function measures the distance between a point and the margin, penalizing classification errors.

5. **Optimization:**

    - The goal of SVM is to minimize the sum of hinge loss penalties and maximize the margin. This leads to a quadratic optimization problem, which is solved to obtain the weights w and bias b of the hyperplane.

6. **Kernel Trick:**

    - The kernel trick is employed to extend SVM to higher-dimensional feature spaces without explicitly performing the transformation. This allows handling non-linear problems by introducing a kernel function that computes the dot product in the transformed feature space.

7. **Classification and Prediction:**

    - Once the optimal hyperplane is found during training, SVM classifies new data points based on their position relative to the hyperplane. For regression problems, SVM predicts continuous values rather than discrete classes.

Applications of SVM

- Text classification and sentiment analysis.
- Handwriting and character recognition.
- Medical diagnosis, such as disease detection.

- Spam detection in emails.

SVM is known for its effectiveness in handling high-dimensional datasets and its efficiency in binary classification problems and some regression problems. The kernel trick expands its applicability to non-linear problems.

Model: Logistic Regression

The logistic regression model transforms a linear combination of input features using the logistic function [43]. The logistic function maps any input value to a value between 0 and 1, as shown in Figure 2.4, which can be interpreted as the probability of the input belonging to the positive class.

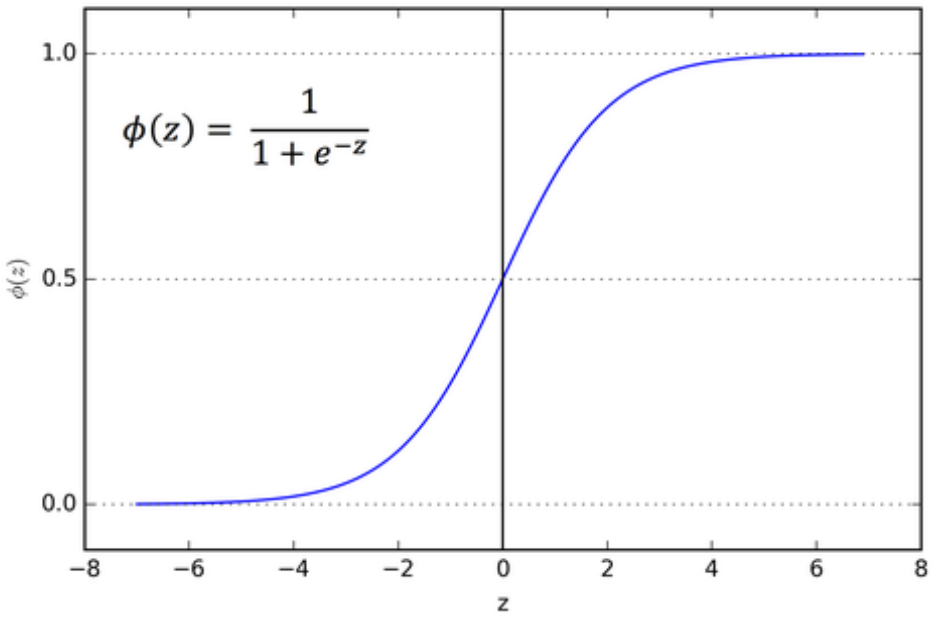

Figure 2.4: Logistic Regression [3]

Given an input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and corresponding weights $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ [44], the linear combination is calculated as:

$$z = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

The logistic function then transforms the linear combination into a probability:

$$p(\mathbf{x}) = \frac{1}{1 + e^{-z}}$$

Where e is the base of the natural logarithm.

During training, the model adjusts the weights $\mathbf{w}$ to minimize the difference between the predicted probabilities and the actual class labels. This is typically done using an optimization algorithm like gradient descent [45].

Applications of Logistic Regression

Logistic Regression has various applications, including:

- Medical Prediction of whether a patient has a specific disease. For example, they determine the probability of an individual developing diabetes based on certain risk factors.
- Sentiment Analysis, classification of sentiment in a text as positive or negative. This is used in social media, online product reviews, movie reviews, etc., to assess overall perception.
- Credit risk assessment, determining if a loan applicant is likely to default on payments. Logistic Regression can analyze various financial and credit history factors to predict the risk of default.

Logistic Regression is a powerful and widely used model for binary classification tasks. It transforms input features using the logistic function to estimate the probability of an input belonging to a specific class, making it a versatile tool for various applications.

Model: Naive Bayes

The naive Bayes(NB) Classifier is a supervised learning model based on Bayes' theorem. Despite its name including "naive," this model has proven effective in many applications and is particularly popular for text classification tasks.

Bayes' Theorem:

The Naive Bayes classifier is built on Bayes' theorem, which establishes the relationship between the conditional probability of an event given another event and the prior probabilities of related events. [46]

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

In the context of the Naive Bayes classifier, we make the "naive" assumption of conditional independence between each pair of features given the classes. This assumption simplifies calculations and facilitates the implementation of the model.

It is essential to know how NB works, as mentioned in the paper "Improved naive Bayes classification algorithm for traffic risk management"[47].

1. **Prior Probabilities:**
    - Calculate the prior probabilities of each class in the training set. This involves computing the probability that a random data point belongs to each class before considering the specific features of the data point.
2. **Conditional Probabilities:**
    - Calculate the conditional probabilities for each feature given in each class. These are the probabilities of observing a specific feature given that we know the class to which it belongs.
3. **Classification:**
    - Given a new data point, the Naive Bayes classifier calculates the probability of it belonging to each class using the prior and conditional probabilities. It then classifies the data point into the class with the highest probability.

Types of Naive Bayes Classifiers:

1. **Gaussian Naive Bayes:**
    - Assumes that features follow a Gaussian (normal) distribution.

2. **Multinomial Naive Bayes:**
    - Suitable for discrete features describing counts, such as word frequencies in documents.
3. **Bernoulli Naive Bayes:**
    - Used for binary features, where only the presence or absence of a feature is considered.

Applications:

- Document and email classification as spam or non-spam.
- Sentiment analysis in comments and reviews.
- Medical diagnosis based on symptoms.

The Naive Bayes classifier is efficient, easy to implement, and often performs well on large and sparse datasets, especially in text classification tasks. The model's simplicity and speed make it popular for practical applications.

2.2.2 Unsupervised Learning

Unsupervised learning is an approach in machine learning where the objective is to discover hidden patterns, structures, or relationships in data without needing pre-existing labels or known outputs[48]. In contrast to supervised learning, where the model is provided with labeled examples to learn how to make predictions, unsupervised learning allows the model to explore the data on its own to uncover helpful information.

In unsupervised learning, two main types of tasks are commonly used:

1. **Clustering:** In this task, the goal is to divide the data into groups or clusters, where elements within the same set are similar but different from elements in other collections. Clustering algorithms aim to identify natural structures in the data, such as groups of users with similar behaviors, market segments, etc.

2. **Dimensionality Reduction:** In this task, the objective is to reduce the number of variables in the data while preserving as much relevant information as possible. Dimensionality reduction helps visualize data in lower-dimensional plots, eliminating noise and redundancy and speeding up algorithm processing. A typical example is Principal Component Analysis (PCA).

In unsupervised learning, the training process focuses on discovering intrinsic patterns in the data without external guidance. Even without labels, these models can uncover exciting and unknown structures with applications in various fields, such as text analysis, image segmentation, product recommendation, and more.

Table 2.1: Categories of different unsupervised learning models

Algorithm	Clustering	Dimensionality Reduction
Principal Component Analysis		X
K-means	X	

2.2.3 Semi-supervised Learning

Semi-supervised learning is a machine learning approach that combines unsupervised and supervised learning elements. It is used when you have a dataset that contains a small amount of labeled data and a more significant amount of unlabeled data. This setting is familiar in real-world applications, as acquiring labeled data can be expensive and time-consuming.

In semi-supervised learning, the model leverages the available labeled data, where the relationships between inputs and outputs are known, to learn patterns and structures within the entire dataset, including the unlabeled data[49]. Several characteristics are essential to semi-supervised learning, among which are:

1. **Combines Labeled and Unlabeled Data:** In semi-supervised learning, you can access labeled data with known outputs and unlabeled data. The combination of labeled and unlabeled data allows the model to harness the benefits of both types of information.

2. **Tasks in semi-supervised Learning:** In semi-supervised learning, some tasks are similar to those in unsupervised learning, such as clustering and dimensionality reduction. The model can still uncover hidden patterns and structures within the unlabeled data. However, it does so with the guidance of the labeled data, which can improve the quality of the learned representations.

3. **Increased Efficiency:** Semi-supervised learning is beneficial when obtaining labeled data is costly or impractical. By using a small amount of labeled data to guide the learning process, you can make the most of the available resources while benefiting from the large amount of unlabeled data.

4. **Applications:** Semi-supervised learning is applied in various fields, including natural language processing, computer vision, and speech recognition. Semi-supervised learning can help improve the model's performance by incorporating both data types.

2.3 Deep Learning

Deep learning is a subset of machine learning that uses artificial neural networks to solve complex problems[50]. These networks, inspired by the structure and function of the human brain, consist of interconnected layers of artificial neurons. Deep learning has gained significant attention and popularity in recent years due to its remarkable performance in various applications[51]. Here is an in-depth look at the critical aspects of Deep Learning:

2.3.1 Neural Networks

At the heart of Deep Learning are artificial neural networks. These networks comprise layers of interconnected neurons, where each neuron processes information and passes it to the next layer. The three fundamental types of layers in a neural network are:

1. **Input Layer:** This layer receives the initial data, such as images, text, or other structured or unstructured information forms.

2. **Hidden Layers:** These layers, which can be multiple, process and transform the input data through various mathematical operations, such as weighted sums and activation functions.

3. **Output Layer:** The final layer provides the model's prediction or output, depending on the specific task, whether image classification, language translation, or any other application.

2.3.2 Training and Learning

Deep Learning models learn from data through a process known as training. The model optimizes its internal parameters (weights and biases) during training to minimize the difference between its predictions and target values in the training data [52]. The key components of training include:

1. **Loss Function:** A loss function quantifies the difference between the predicted output and the actual target. The goal is to minimize this loss [53].

2. **Backpropagation:** This is an algorithm that computes gradients of the loss concerning the model's parameters. These gradients guide the parameter updates, improving the model's predictions.

3. **Optimization Algorithms:** Various optimization algorithms, such as stochastic gradient descent (SGD), adjust the model's parameters efficiently during training.

4. **Regularization Techniques:** Regularization methods like dropout and L1/L2 regularization help prevent overfitting, where the model memorizes the training data rather than generalizes it.

2.3.3 Architectures and Applications

Deep learning encompasses a wide range of neural network architectures and applications [54]. Some notable architectures include:

1. **Convolutional Neural Networks (CNNs):** Designed for image and video analysis, CNNs can identify features and patterns in visual data [55].

2. **Recurrent Neural Networks (RNNs):** RNNs are used for sequential data, such as natural language processing and time series prediction, due to their ability to capture temporal dependencies.

3. **Transformers:** These models have revolutionized natural language processing tasks and are known for their self-attention mechanisms. That means that such a model can work on different data areas simultaneously, thus capturing long-term relationships. We have several models based on this architecture, including BERT, GPT, and XLNet.

4. **Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM):** These architectures improve the handling of vanishing gradient problems in RNNs.

5. **Generative Adversarial Networks (GANs):** GANs are used for generating synthetic data, image-to-image translation, and artistic style transfer.

Deep learning has found applications in various domains, including:

1. **Computer Vision:** Image and video analysis, object detection, facial recognition, and autonomous driving.

2. **Natural Language Processing (NLP):** Machine translation, sentiment analysis, chatbots, and text generation.

3. **Speech Recognition:** Voice assistants, transcription services, and speaker identification.

4. **Healthcare:** Medical image analysis, disease diagnosis, and drug discovery.

5. **Finance:** Predictive modeling, fraud detection, and algorithmic trading.

6. **Autonomous Systems:** Robotics, drones, and self-driving cars.

2.3.4 Model: Sequential Neural Networks (SNN)

Sequential Neural Networks (SNN) are a Recurrent Neural Networks (RNN) subset tailored for sequential data processing. SNN is a type of neural network architecture designed to process sequential data and has applications in various machine learning tasks and sequential data processing [56].

How Sequential Neural Networks Work

They consist of layers of connected neurons, each fully connected to the next.

- **Dense Layers:** SNNs employ densely connected layers, where each neuron in one layer is connected to all neurons in the next layer.
- **Activation Functions:** SNN layers commonly use activation functions like ReLU (Rectified Linear Unit), sigmoid, or tanh to introduce nonlinearities into the model.
- **Dropout:** Using the Dropout technique helps prevent overfitting by randomly turning off a percentage of neurons during each training step.
- **Output:** The final layer of the SNN typically has neurons representing the output classes or categories. For classification problems, the common activation is softmax.

SNNs process data sequentially through these layers, with each layer progressively transforming the input information until reaching the output layer.

Applications of Sequential Neural Networks

Sequential Neural Networks (SNNs) have found applications in various domains:

- Image Classification and Object Recognition: SNNs can learn complex features and make accurate classifications in tasks such as image and object recognition.
- Time Series Analysis: They forecast future values in sequential data, such as finances, traffic, or weather conditions, being useful for the analysis and prediction of temporal trends.
- Tabular Data Processing: They assist in classification or regression problems on structured datasets, facilitating tabular data processing.

In summary, Sequential Neural Networks offer flexibility and versatility that make them suitable for addressing the challenge of detecting hate speech, especially when analyzing the sequential structure and linguistic complexities associated with this phenomenon.

2.3.5 Model: Bidirectional Encoder Representations from Transformers(BERT)

BERT is a pre-trained language model that uses the transformer architecture, a block-based attention architecture [57]. It is part of the family of NLP transformers, similar to RoBERTa and other models, which are widely used for NLP tasks and sentiment analysis [58]. Its key innovation is its ability to capture the bidirectional context of words in a sentence. Instead of processing words in a single direction, BERT examines the entire surrounding context, both to the left and right of each word.

How BERT works

- **Pretraining:** BERT is trained on large amounts of unlabeled text. During this process, the model learns to predict masked words in a sentence and understand the relationship between different text parts.
- **Bidirectional Architecture:** BERT uses a bidirectional architecture of transformers. Each layer of transformers has multiple attention heads, allowing the model to capture complex relationships and dependencies throughout the word sequence.
- **Stacked Layers:** BERT consists of multiple stacked layers of transformers. Each layer refines word representations based on the bidirectional context learned during pretraining.
- **Fine-Tuning:** After pretraining, BERT can be finely tuned for specific tasks such as text classification, entity extraction, or question answering. During fine-tuning, the model adapts its learned representations to the particular task.

Applications

BERT has demonstrated exceptional performance across various NLP tasks, including text classification, sentiment analysis, information extraction, and translation. Its ability to understand complex contexts and capture bidirectional dependencies has led to significant advancements in natural language understanding by machines.

2.4 Model Evaluation

Model evaluation is a crucial step in assessing the performance and effectiveness of machine learning models. This section provides insights into how to evaluate machine learning models and deep learning models.

2.4.1 Evaluating Machine Learning Models

Evaluating machine learning models is essential to measure their performance and determine their effectiveness in specific tasks. Here are key aspects of evaluating machine learning models:

Evaluation Metrics

Metrics are quantitative indicators used to assess a model's performance [59]. Common metrics in machine learning evaluation include:

Classification Metrics:

- **Confusion Matrix:** A table representing the performance of a classification model by comparing actual and predicted classes. The confusion matrix contains four cells showing the number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions [60]. TP represents the number of instances that were correctly predicted as positive by the model. FP represents the number of instances that were incorrectly predicted as positive by the model. TN represents the number of instances that were correctly predicted as negative by the model. FN represents the number of instances that were incorrectly predicted as negative by the model. All this is represented in Figure 2.5.

- **Accuracy:** Measures the proportion of correct predictions out of the total predictions. It provides an overall assessment of the model's correctness. Accuracy is particularly useful when the classes in the dataset are well-balanced [61]. It is calculated using the formula:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total predictions}}$$

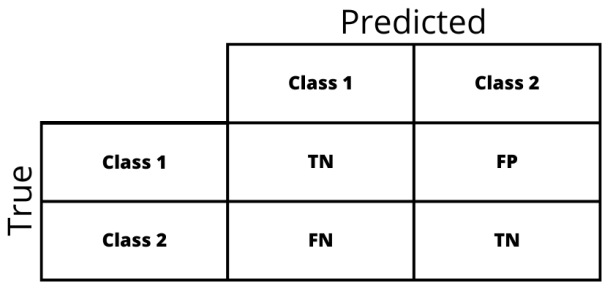

Figure 2.5: General confusion matrix

- **Precision:** Calculates the ratio of accurate positive predictions to the total optimistic predictions [62]. It is helpful in binary classification problems. Precision is calculated using the formula:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall:** Computes the ratio of accurate optimistic predictions to actual positive instances. Recall, also known as Sensitivity or True Positive Rate (TPR), measures the model's ability to identify positive instances correctly [62]. It is essential when detecting all positive instances is paramount, such as in disease screening tests. It is calculated using the formula:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F1-Score:** The harmonic mean of precision and recall provides a balanced measure of a model's accuracy [62].F1-score is useful when there is an uneven class distribution or when false positives and false negatives carry different costs. The F1-score is calculated using the formula:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **ROC-AUC:** Area under the Receiver Operating Characteristic curve measures the model's ability to discriminate between positive and negative classes across different thresholds. The area under the curve (AUC) is calculated by plotting the true positive

rate against the false positive rate at various threshold settings [63].

- **Log Loss:** Measures the performance of a classification model where the prediction output is a probability value between 0 and 1 [64]. Log Loss is calculated using the formula:

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right)$$

where N is the number of samples, y_i is the true label of the i-th sample, and $\hat{y}_i$ is the predicted probability of the positive class for the i-th sample. y_i takes the values of 0 if the sample does not belong to the target class and 1 if it belongs.

Regression Metrics:

- **Mean Squared Error (MSE):** Used in regression problems, it measures the average squared differences between predictions and actual values [65]. MSE is calculated using the formula:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- **Mean Absolute Error (MAE):** Measures the average absolute differences between predictions and actual values [66]. MAE is calculated using the formula:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

- **R-squared (R2):** Represents the proportion of the variance in the dependent variable that is predictable from the independent variables [66]. R-squared is calculated using the formula:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where RSS is the residual sum of squares and TSS is the total sum of squares.

- **Root Mean Squared Error (RMSE):** Similar to MSE but provides a measure of the standard deviation of the residuals. RMSE is calculated as the square root of the mean squared error (MSE) [65].

2.4.2 Evaluating Deep Learning Models

Evaluating deep learning models shares similarities with machine learning evaluation but includes specific considerations for deep neural networks. Here are key aspects:

Evaluation Metrics

Evaluation metrics used in deep learning models are similar to machine learning, including accuracy, precision, recall, F1-score,confusion matrix, and MSE. However, deep learning often employs task-specific and custom metrics.

Model Size and Capacity

The size and capacity of a deep learning model can impact performance. Larger models may be prone to overfitting, while petite models may underfit. Choosing the right model size is a critical aspect of deep learning.

Regularization Techniques

To prevent overfitting, deep neural networks often require regularization techniques, such as dropout and L1/L2 regularization. These techniques help control the model's complexity.

Hyperparameter Tuning

Hyperparameter tuning is a crucial part of deep learning model evaluation. This includes adjusting parameters like learning rate, network architecture, the number of layers, and the number of units in each layer.

Transfer Learning

In deep learning, transfer learning involves using pre-trained models on massive datasets, such as language models or convolutional neural networks trained on large image datasets. This technique can accelerate training and improve performance.

In summary, evaluating deep learning models encompasses various metrics, the confusion matrix for classification tasks, consideration for model size and capacity, regularization techniques, hyperparameter tuning, and transfer learning to enhance performance.

2.5 NLP: Natural Language Processing

Natural Language Processing (NLP) is an interdisciplinary field of artificial intelligence and computational linguistics that focuses on the interaction between computers and human language as spoken and written [67]. The main objective of NLP is to allow machines to understand, interpret, and generate human language in a natural way [68]. The use of Natural Language Processing involves a certain variety of tasks, among which are:

- Data Collection and Preprocessing
- Text Representation
- Modeling and Training
- Evaluation and Fine-Tuning

Figure 2.6 shows the general structure of an NLP model, where there are two phases, one for training and one for testing. In the first phase, all the data processing and characteristics extraction are done to train the model. In contrast, in the second phase, the features of a tweet that has not been used in the model are extracted, and later, it is sent to the model to the classifier, which is generated once the model has been trained. The model gives us the class to which the text belongs.

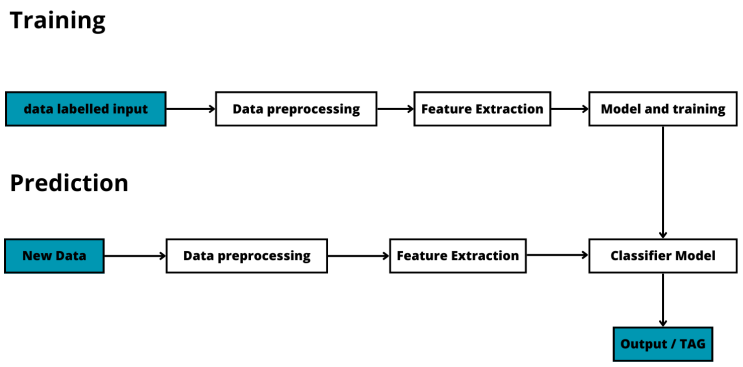

Figure 2.6: General structure of NLP

2.5.1 Data Preprocessing

Data preprocessing is a pivotal phase in analyzing textual data, such as tweets, as it encompasses a series of systematic transformations to enhance data quality, ensure

compatibility with analysis techniques, and extract meaningful insights. Sequentially, the process involves several essential components, such as data annotation and others displayed in Figure 2.7. It is vital to describe each one of the steps to preprocess the data [69].

Moreover, it is expected to encounter numerous repetitions in dealing with a substantial amount of data. Therefore, it becomes necessary to eliminate all duplicated data entries in the dataset. This ensures a more streamlined and efficient analysis, reducing redundancy and enhancing the overall quality of the processed data.

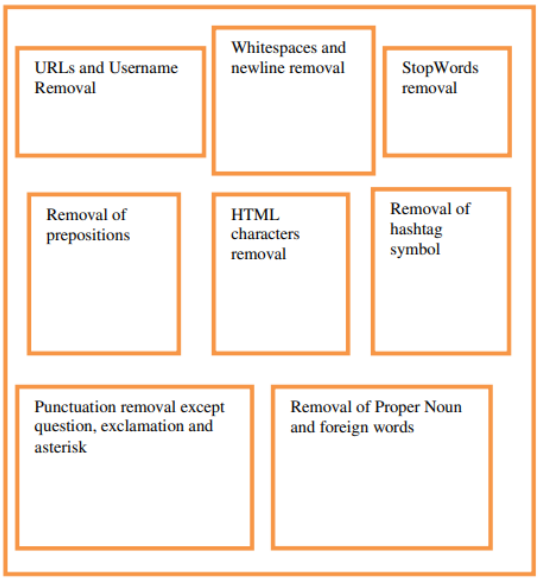

Figure 2.7: Basic preprocessing of tweets

Given a tweet in Figure 2.8, the step-by-step preprocessing of this data is demonstrated to have a clearer idea of how each step is associated with processing this type of data.

Figure 2.8: Tweet example

Data Annotation and Labeling

Data annotation is crucial for its critical role in model training and evaluation. These models can be machine learning algorithms and deep learning techniques. Labeling the data makes it much easier for the model to recognize specific patterns of the different classes. In the context of hate speech, it is essential to label the data since we will only have a certain number of given types. To facilitate a correct annotation in this context, several tools and APIs allow us to examine the level of Toxicity of the attached data; among these is the Perspective APIs [1] which is described in the following section.

Text Cleaning

Remove special characters, punctuation marks, links, and mentions. This can be done using regular expressions. Convert the text to lowercase to ensure that uppercase and lowercase words are treated similarly.

Continuing with the previous example 2.4, the tweet after the text cleaning looks like this: "With the right person, even the simplest thing becomes fun: visit us."

Stopword Removal

This technique is a process that involves the removal of stop words from a text; this process is done before analyzing or processing it in any way. These are familiar words, which, most of the time, do not provide a relevant meaning to the content of the text. The types of words include pronouns, prepositions, conjunctions, and other terms such as "he," "she," and "in," among others.

The use of stopwords elimination has several objectives, among which are:

- **Noise Reduction:** Removing stop words reduces noise in the text and highlights the most significant keywords and terms.
- **Efficiency improvement:** By removing stop words, the efficiency of word processing and analysis algorithms can be improved by working with a more focused set of words.
- **Retrench of space:** Removing stop words can reduce the total number of words in a text, which can help speed up processing and analysis.

[1] https://perspectiveapi.com

Applying tokenization and stopword removal to example 2.8 results in “right person simpler come back fun visit”.

Feature Extraction

This involves transforming text into numerical or categorical features that machine learning algorithms can process. Techniques such as bag-of-words and word embeddings can be used.

- **Bag-of-Words**

  The bag of words (BoW) technique is fundamental when representing text documents; in this context, they are tweets representing numerical vectors [70], which can be understood by different Machine learning models. The bag of words technique is an outstanding tool in natural language processing. The operation of the pack of words is briefly introduced:

  1. **Tokenization:** The first step is to break the text into smaller units called “tokens.” In most cases, tokens are words but can also be subwords (n-grams), characters, or emojis.
  2. **Vocabulary Creation:** Next, a vocabulary is created from all the unique tokens in the corpus of tweets. Each word becomes a dimension in a vector space.
  3. **Frequency Counting:** For each tweet, the occurrences of each word in the vocabulary are counted. This creates a vector representing the frequency of each word in that tweet. Words that don’t appear in the tweet will have a count of zero.
  4. **Vectorization:** Finally, each tweet is represented as a numerical vector, where each dimension corresponds to a word from the vocabulary, and the value in that dimension is the frequency of that word in the tweet.
  5. **Document-Term Matrix:** By applying BoW to all tweets, a tweet-term matrix is created, where each row represents a tweet and each column represents a word from the vocabulary. The values in the matrix are the frequencies of the words in the tweets.

6. **Use in Machine Learning Models:** This numerical matrix can be used as input for Machine Learning algorithms, such as classifiers. Each tweet has been converted into a numerical vector the model can process.

For example, if we have a vocabulary of limited size with words like Figure 2.8, "with", "the", "right", "person", "even", "simplest", "thing", "becomes", "fun", "visit", and "us", and the text is "With the right person, even the simplest thing becomes fun: visit us.", the resulting vector might look something like this: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], where each value represents the frequency of each word in the text in the order of occurrence in the vocabulary.

- **Word Embeddings**

  In this technique, several pre-trained models, such as Word2Vec or GloVe, represent words in a continuous vector space, capturing semantic and contextual relationships.

  – **Word2Vec: Advanced Word Representation Algorithm**

    Word2Vec is an advanced word representation algorithm in natural language processing. Its primary goal is to transform words into numerical vectors in a high-dimensional vector space, such that the semantic and contextual relationships between terms are reflected in the distance and relative orientation of the vectors[71]. Word2Vec operates through two main approaches: Skip-gram and Continuous Bag of Words (CBOW).

    Skip-gram In the Skip-gram approach, Word2Vec takes each word in a text corpus and attempts to predict the surrounding words (context) based on that word. The idea is that words sharing similar contexts will have similar vectors in the vector space. The process can be summarized in the following steps:

    1. Select a target word from the text.
    2. Create a vector for the target word (initially random).
    3. Use the vector to predict the surrounding words (context).
    4. Gradually adjust the target vector to improve the prediction of surrounding words.

5. Repeat this process for all words in the corpus and adjust the vectors based on multiple context instances.

Then, using the example 2.8, a hypothetical example of Word Embedding rendering is shown for some words in the text using Word2Vec.

```
‘‘right’’: [0.2, 0.5, -0.3, 0.8, -0.1, ...]
‘‘person’’: [-0.1, 0.3, 0.6, -0.2, 0.7, ...]
‘‘simplest’’: [0.4, -0.2, 0.1, 0.9, -0.5, ...]
‘‘fun’’: [-0.3, 0.7, -0.4, 0.6, 0.2, ...]
‘‘visit’’: [0.6, -0.4, 0.2, 0.5, -0.6, ...]
‘‘us’’: [-0.5, 0.2, -0.7, 0.3, 0.1, ...]
```

Continuous Bag of Words (CBOW)

In the CBOW approach, Word2Vec does the opposite. It takes the context (surrounding words) and aims to predict the target word. The process is similar to the Skip-gram approach but reversed:

1. Select a context of surrounding words.
2. Average the word vectors in the context to create a context vector.
3. Use the context vector to predict the target word.
4. Gradually adjust the vectors of surrounding words to improve the prediction of the target word.

– **GloVe: Global Vectors for Word Representation**

GloVe, short for Global Vectors for Word Representation, is another influential algorithm in natural language processing focusing on word representation. Its primary goal is to capture global statistical information about the co-occurrence of words in a given corpus. The fundamental concept behind GloVe is to create word vectors that encode semantic relationships by analyzing patterns of expression co-occurrences across the entire dataset[72]. Here's a breakdown of how GloVe functions:

Co-occurrence Matrix

GloVe initiates by constructing a co-occurrence matrix based on the frequency of word pairs appearing together in the corpus. Each cell within the matrix denotes the frequency of one word appearing within the context of another word. This matrix effectively captures the statistical relationships between terms.

Word Vector Initialization

GloVe initializes word vectors for each vocabulary word. These vectors begin with random values.

Objective Function

GloVe defines an objective function that represents the relationship between the dot product of word vectors and the logarithm of the observed co-occurrence values. The objective is to learn word vectors that produce similar dot products as the logarithms of the co-occurrence values.

Optimization

GloVe employs an iterative optimization process to adjust the word vectors to minimize the discrepancy between the predicted dot products (derived from the phrase vectors) and the actual co-occurrence values. This process entails updating the word vectors iteratively to find the most suitable representation that fulfills the specified objective function.

Vector Space Representation

The resulting word vectors are embedded within a continuous vector space upon optimization. Within this space, words with similar meanings or usage contexts are situated closer to each other. The vectors effectively encode semantic relationships between terms based on their co-occurrence patterns.

Application

The word vectors derived from GloVe can be applied to various natural language processing tasks, including measuring similarity, word analogies, sentiment analysis, and more. Comments sharing similar meanings or contextual usages will have corresponding vector representations close in proximity.

Lastly, GloVe differs from Word2Vec in that it leverages global statistics and co-occurrence patterns to create word vectors, whereas Word2Vec focuses on predicting nearby words (Skip-gram) or a target word from its context (CBOW). GloVe and Word2Vec are potent tools for numerically representing words, enabling machines to comprehend and process human language more effectively.

2.6 Hate Speech

In recent years, technological advancements and the proliferation of social media have provided access to many individuals, but they are not always used appropriately. There has always been a division of opinions, often leading to hostile, prejudiced, or discriminatory attitudes among individuals. This extends to views and discrimination against various groups, such as those based on race, religion, gender, and sexual orientation, among others [73]. The emergence of new social media has exacerbated the prevalence of hate speech, which entails verbally expressed animosity directed towards groups or individuals in various contexts. Recognizing this hate speech provides us with a solid foundation to address the issue and develop effective strategies for prevention and mitigation.

The connection between technology, freedom of expression, and hate speech highlights the need to consider the ethical and social implications of online communication. While freedom of expression is fundamental to democracy and individual autonomy, it is also crucial to establish limits that prevent the abuse of that freedom to propagate hatred and violence. Effective regulation and public awareness are essential to counteract the negative impact of hate speech and ensure that technology and online platforms are used responsibly and constructively.

As technology advances, a handy tool has been developed to detect and classify tweets according to different kinds of hate. This tool is perspective API, which allows us to analyze

how much hate the message to be analyzed.

Perspective APIs

Perspective is an API created by Jigsaw and Google's Counter Abuse Technology team in Conversation-AI. Conversation AI is a collaborative research effort exploring ML as a tool for better discussions online. The API uses machine learning models to score the Toxicity of an input text, where toxic is defined as a rude, disrespectful, or unreasonable comment that is likely to make one leave a discussion [74].

Perspective models provide scores for several different attributes. In addition to the flagship Toxicity attribute, here are some of the other attributes Perspective can provide scores for:

- Severe Toxicity
- Insult
- Profanity
- Identity attack
- Threat
- Sexually explicit

To evaluate the Toxicity of language in each comment, raters use the following definitions:

Toxic level	Description
Very Toxic	A comment that is hateful, aggressive, or otherwise very likely to make a user leave a discussion.
Toxic	A comment that makes a user leave a discussion.
Not Toxic	A even nice comment is unlikely to discourage the conversation.

Table 2.2: Toxicity of language on Perspective API

So, to finish understanding the operation of this API, an example of the perspective toxic detection system using phrases provided by API.

Original Phrase(Toxicity Score)
Climate change is happening, and it's not changing in our favor. If you think differently, you're an **idiot**. (84%)
They're **stupid**, it's getting warmer, we should enjoy it while it lasts (86%)
They are liberal **idiots** who are **uneducated** (90%)
idiots. backward-thinking people. **nationalists.** not accepting facts, susceptible to **lies**. (80%)
They are **stupid** and **ignorant** with no class (91%)
It's **stupid** and wrong (89%)
If they voted for Hilary the are **idiots** (90%)
Anyone who voted for Trump is a **moron** (80%)
Screw you trump supporters (79%)

Table 2.3: Example of perspective API toxic detection system

Chapter 3

State of the Art

In this chapter, several Machine Learning and deep learning models were analyzed in the context of Hate speech. Twelve research articles published from 2015 to 2022 are presented in a summary. The methodologies vary between different models of Machine learning and deep learning. In the case of machine learning, there are different models, such as logistic regression, support vector machine, random forest, and decision tree. Meanwhile, techniques such as RNN, and transformers are analyzed in deep learning.

Ibrahim A., Maria H., Neveen H., Hossam F., Raneem Q. , Bassam H., Mohammad A. & Mohammad A.

The authors present a study on detecting cyber hate speech on social media sites, particularly Twitter [75], using machine learning algorithms. The machine learning algorithms used in this paper are Support Vector Machine (SVM), Naive Bayes (NB), Decision Tree (DT), and Random Forest (RF). These algorithms were implemented using Python and the Scikit-learn machine learning package. The authors used different combinations of three features (words, profile, and emotional features) to train and test the models. The authors created a dataset for capturing expressions of hatred. They conducted several text vectorization techniques and machine learning algorithms on the dataset to measure its performance and capability in detecting cyber hate speech. The study also includes a feature importance analysis on the collected dataset to determine the most significant informative features. The evaluation measures used in this paper are accuracy, precision, recall, and g-mean. The study concludes

that the Random Forest algorithm outperforms others in detecting cyber hate speech on Twitter.

Oriola O. & E. Kotzé

In this work, O. Oriola and E. Kotzé [76] present an evaluation of machine learning techniques for detecting offensive and hate speech in South African tweets. The authors used a dataset of 25,000 tweets collected from Twitter using keywords related to hate speech and offensive language. Human annotators annotated the dataset into three classes: hate speech, offensive speech, and free speech.

Then, they used several machine learning algorithms to build models for detecting offensive and hate speech in South African tweets. The algorithms used include Logistic Regression (LogReg), Support Vector Machine (SVM), Random Forest (RF), and Gradient Boosting (GB). The authors also used ensemble models such as weighted, majority voting, and meta-learning to evaluate the performance of the different features. The authors optimized the models using a grid search approach. They assessed the performance of the models on the test set using various metrics such as True Positive Rate (TPR), accuracy, precision, recall, and F1-score. After that, they found that the multi-tier meta-learning model with three meta-features comprising word n-gram, character n-gram, and syntactic meta-features recorded the best TPR of 0.858 and 0.887 for hate speech and offensive speech, respectively. Specifically, SVM, RF, and GB had the best TPR for hate speech, while LogReg had the best TPR for offensive speech. The authors also found that despite the effectiveness of word and character n-gram features in the detecting hate speech and offensive speech, the excellent performance of one led to the poor performance of the other.

Din A. , Rabani S., Khan Q. & Mali S.

In this work, the authors aim to detect hate speech on social media during the COVID-19 era using machine learning algorithms [77]. The authors used a dataset of 4,093 tweets labeled into two classes: “Hate” and “Normal.” The paper used various machine and ensemble learning algorithms to perform the classification task, including Decision Tree, Stochastic Gradient Boosting, Multinomial Naive Bayes, and Support Vector Machine. The model was

evaluated using several metrics: precision, recall, F1-score, and accuracy. The precision of the model was 98%, the recall was 97%, and the accuracy was 97.96%. These metrics were calculated using a fivefold cross-validation approach. The results showed that the Decision Tree and Stochastic Gradient Boosting algorithms outperformed all other algorithms in accuracy. The paper concludes that machine learning algorithms can effectively detect hate speech on social media during COVID-19.

Abro S., Khand Z., Shaikh S., Ali Z., Khan S. & Mujtaba G.

In this work, the authors present a comparative study of various feature engineering techniques and machine learning algorithms to detect hate speech messages on social media platforms [78]. The authors used a publicly available dataset of tweets that were labeled as hate speech, offensive but not hate speech, or not offensive.

Then, they used three types of features: n-gram (bigram) with TFIDF, Word2vec, and Doc2vec. They applied eight machine learning algorithms to create master feature vectors: NB, SVM, KNN, DT, RF, AdaBoost, MLP, and LR. In total, 24 analyses were evaluated to check the effectiveness of classification models.

The results showed that the highest recall, precision, accuracy, and F-measure were obtained by SVM using TFIDF features representation with bigram features. The lowest precision, recall, accuracy, and F-measure were found in MLP and KNN classifiers using TFIDF features representation with bigram features.

Overall, this study provides insights into the effectiveness of different feature engineering techniques and machine learning algorithms in detecting hate speech messages on social media platforms. The findings of this study can be used to develop more accurate and efficient hate speech detection systems, which can help address the issue of hate speech on social media platforms.

Badjatiya P., Gupta S., Gupta M. & Varma V.

In this work, the authors [79] investigate the application of deep learning methods for the task of hate speech detection on Twitter. They explore various tweet semantic embeddings like char n-grams, word Term Frequency-Inverse Document Frequency (TF-IDF) values, Bag of Words Vectors (BoWV) over Global Vectors for Word Representation (GloVe), and task-specific embeddings learned using FastText, CNNs and LSTMs.

The authors experimented with a dataset of 16K annotated tweets made available by the authors. Of the 16K tweets, 3383 are labeled as sexist, 1972 as racist, and the remaining are marked as neither sexist nor racist. They performed 10-fold cross-validation and calculated weighted macro precision, recall, and F1-scores. They used 'adam' for CNN, ' STM, and 'RMS-Prop' for FastText as their optimizer. They performed training in batches of size 128 for CNN & LSTM and 64 for FastText.

The authors experimented with multiple word embedding sizes for their task. They observed similar results with different sizes, and hence, due to lack of space, they reported results using embedding size=200. They experimented with multiple models, including CNN, LSTM, and FastText. They also experimented with various classifiers like SVMs and GBDTs as the learning method.

The author's methods beat state-of-the-art methods by a large margin (18 F1 points better). They also showed that embeddings learned using DNNs clearly show the "racist" or "sexist" bias for various words.

Paul C. & Bora P.

In this work, Chayan Paul and Pronami Bora [80] discuss the problem of hate speech on social media and propose a deep learning model for classifying social media content as either hateful or usual. The authors collected a dataset from Kaggle containing tweets from American users and built two deep learning models, LSTM and Bi-LSTM, to classify the tweets. The authors evaluated the performance of the models using several metrics, including accuracy, precision, recall, and F1 score. The results showed that the Bi-LSTM model outperformed the LSTM model regarding accuracy and F1 score. The authors mention several limitations of the study, including the limited size of the dataset, which

may not be representative of all types of hate speech on social media. The study also focused on English-tweets, which may not apply to other languages. The authors also noted that the models may not detect subtle forms of hate speech, such as sarcasm or irony.

Hind S., Areej M. & Kawthar M.

In this work, Hind S., Areej M., and Kawthar M [81] investigate the feasibility of automatically detecting white supremacist hate speech on Twitter using deep learning models and domain-specific word embeddings. The paper begins with an introduction that highlights the negative impact of white supremacist hate speech on society and the limitations of traditional methods of detecting such speech on social media.

The paper provides background information on word embeddings and language models, including word2vec, Glove, and BERT. The authors propose a deep learning model that utilizes domain-specific word embeddings to capture whitsemanticsacist slang and coded slang semantics. The proposed model is trained and tested on a balanced subset of the Stormfront dataset, which consists of about 2k sentences collected from the Stormfront forum.

After that, the proposed model is a deep learning model that utilizes domain-specific word embeddings to capture the semantics of white supremacist slang and coded words. The model is a Bidirectional Long Short-Term Memory (BiLSTM) neural network that takes as input a sequence of words and outputs a binary classification of whether the input sequence contains white supremacist hate speech or not. The model is trained and tested on a balanced subset of the Stormfront dataset and evaluated on the Twitter White Supremacy dataset. The results show that the proposed model achieves an accuracy of 0.68, outperforming the baseline models.

Steven Z., Chris F. & Udo K.

In this work, the authors focus on improving hate speech detection with deep learning ensembles [82]. The authors used a publicly available dataset of tweets labeled as hate speech or not hate speech. They experimented with different deep learning models, including Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks,

and ensemble methods of these models. They also explored different weight initialization methods, batch sizes, and several epochs to optimize the models.

The results showed that the ensemble method of CNNs and LSTMs with a specific weight initialization method outperformed the other models and achieved a nearly 5% improvement in F-measure compared to previous work. The authors also discussed the challenges of reproducibility in deep learning methods and the need for more detailed evaluation information in published work.

Mozafari, Marzieh & Farahbakhsh

The article aims to present a transfer learning approach for hate speech detection on social media using the BERT model [83], along with a bias mitigation mechanism to address biases in datasets and trained classifiers. Two public datasets were used: the Waseem dataset, which contains 19,697 tweets labeled as racist, sexist, or none of the above, and the Davidson dataset, which includes 24,783 tweets labeled as hate speech, offensive language, or none. The transfer learning approach and the bias mitigation mechanism were evaluated using the F1-macro measure. The results showed that the proposed approach outperformed previous methods in terms of F1-macro and successfully mitigated racial bias in pre-trained classifiers.

Dowlagar, Suman & Mamidi, Radhika

The paper aims to address the automated detection of hate speech and offensive content on social media, focusing on the use of pre-trained BERT and multilingual BERT models [84]. They utilize datasets provided by the shared tasks HASOC FIRE-2019 and FIRE-2020, including English, German, and Hindi tweets. The methods employed revolve around the BERT transformer model for hate speech and offensive language detection. Evaluated metrics include F1 macro score and precision. The results demonstrate that the use of pre-trained BERT and multilingual BERT, fine-tuned for text classification tasks, enhances performance compared to traditional machine learning approaches based on word representations.

Discussion

On the other hand, in deep learning, the study conducted by [79] underscores the superiority of deep learning methods in hate speech detection on Twitter. Through semantic embeddings and deep models such as Convolutional Neural Networks (CNN) and Long Short-Term Memory networks (LSTM), they significantly improved F1 scores compared to traditional methods. This suggests that deep learning models can capture intricate relationships and patterns and are particularly well-suited for addressing precise hate speech detection on social media platforms.

In summary, while logistic regression algorithms have proven effective in machine learning, deep learning models such as sequential neural networks,transformers, and LSTM have emerged as leaders in hate speech detection within the deep learning domain. Both methodologies have advantages and challenges, and the most suitable model will depend on various factors, including the dataset, problem complexity, and available resources.

Summary table

The table with the summary of each work studied is shown in the Appendix 1.

Chapter 4

Methodology

4.1 Experimental Design

Given our problem, six models will be developed to evaluate which suits our requirements better. Treating our data to contain relevant information for our models is crucial. Figure 4.1 shows the steps to follow, in which two models will be trained, and then we will evaluate their performance with specific metrics.

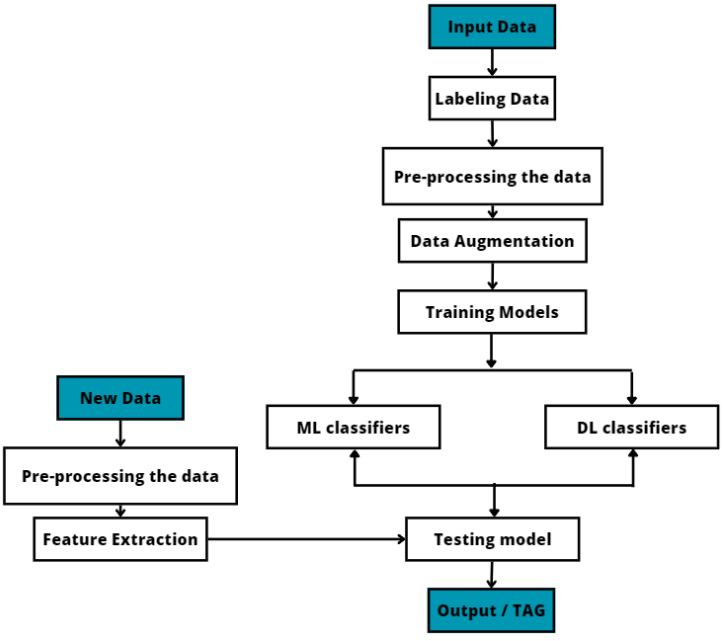

Figure 4.1: General methodology

4.2 Data Collection

The data was collected during the year 2022 from June 13th to June 30th. These dates were estimated because the national strike in Ecuador lasted two weeks. Data was collected through the Twitter API [1], using various filters such as hashtags. Table 4.1 mentions several of them, which were the most relevant when discussing our topic.

Hashtags		
#LassoEcuadorNoTeAguantaMas	#LassoEsUnFracaso	#LassoRenuncia
#ElParoNoPara	#ViolenciaEstadoEC	#ElParoSigue
#DondeEstasLasso	#DestituciónPresidencial	#SinDignidadNoHayPaz
#IzaSomosTodos	#GraciasIza	#VictimasParoEc2022
#IzaTerrorista	#IzaTerroristaCorreaFinancista	#EcuadorQuierePaz
#IzaGolpista	#ConaieTerrorista	#NoAlParo
#YoNoParoYoTrabajo	#IzaSoloBuscaElCaos	#MarchaPorLaPaz
#QueremosTrabajar	#IzaNoMeRepresenta	#SiAlTrabajo
#YoSiTrabajo	#ParoNacionalEC2022	#ParoNacionalEcuador
#ParoNacionalEC	#ParoNacional2022Ec	#ParoNacional2022
#ParoEcuador	#EcuadorSOS	#SOSEcuador
#FueraLassoFuera	#ParenLaMasacre	#BajenLasArmas
#GraciasHermanosIndigenas	#NoPodemosParar	#IzaPreso
#NoALaViolencia	#ParoNacional	#DiálogoNacional
#ParoEcuador	#LeonidasIza	#AcuerdoDePaz

Table 4.1: Hashtags used to filter tweets

During these days, 263.457 data points associated with the 2022 national strike in Ecuador were collected.

Due to the large data, a simple random daily sample of 33,512 data points was taken. The collected information has the following structure, as shown in Table 4.2:

[1] `https://developer.twitter.com/en/docs/twitter-api`

Name	Description
Id	This column refers to the tweet id.
Text	This column refers to the text in the tweet, such as the person's opinion.
Create at	Time tweet was created
Like count	Number of likes the tweet has had
Quote count	Number of quotes the tweet has had
Reply count	Number of replies the tweet has had
Retweet count	Number of retweets the tweet has had

Table 4.2: Data structure

Finally, in our context, we only need the tweet information, i.e., the text, which will be used to train the established models. However, these data need to be processed.

4.3 Data Annotation

Firstly, it is essential to note that a model needs at least two classes to infer and predict correctly. Therefore, a correct labeling of the data is required. The labeling was done manually with the help of a group of 5 undergraduate students, who were supervised by experts on the problem of hate speech.f The data was labeled into two classes: 1 if it was hate towards the indigenous community, and 0 if it was not hate.

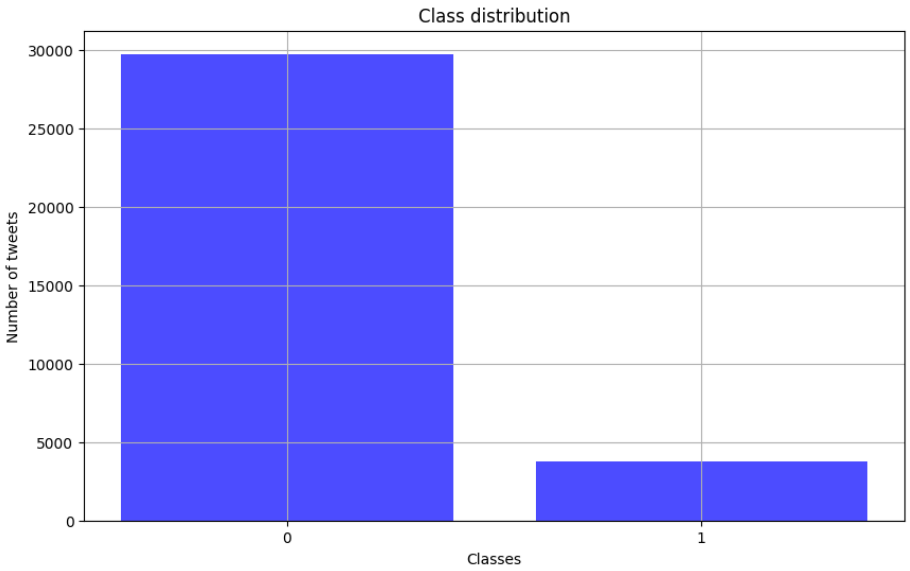

Figure 4.2: Data distribution

As seen in Figure 4.2, after labeling our data, we obtained 29,733 data points labeled as

No hate (Class 0) and only 3,779 data points labeled hate (Class 1).

4.4 Data Verification

To ensure that our manually labeled data is correct, we used a tool previously explained, the Google Perspective API. This tool allows us to measure the level of hate intensity in different areas, as described in the following table:

Areas	Description
Toxicity	Negative, hostile, or harmful behavior or speech towards others, online or offline.
Severe Toxicity	Multiple forms or manifestations of toxic or harmful behavior.
Identity Attack	Targeted attack on someone's identity, such as discrimination based on race, gender, sexual orientation, etc.
Insult	Offensive or derogatory comment directed at someone to belittle or hurt their feelings.
Profanity	Use of vulgar, rude, or coarse language that may be considered offensive or inappropriate.
Threat	A statement or action indicating an intent to cause physical, emotional, or otherwise harm to someone or something.

Table 4.3: Areas of Perspective API

In our study, the most important area to investigate is identity attack, as it is based on the discrimination of a group of people based on their identity, in our case, indigenous.

Once our data is labeled, we check by analyzing all those labeled as hate and comparing them with the expected value from the Google Perspective API, To establish whether it is hate or not, a threshold of 50% was established, where if an identity attack is higher than this value, it is classified as hate, and if it is lower, it is classified as not hate.

Now, we explain brief examples of our tagging, which are hate and are recognized by the perspective API as hate, and likewise when they are not recognized as hate by the API. As shown in the table 4.4, it is observed that in the first example, the hate is directly identified due to the word "indios vagos", while in the second example, the API does not recognize it as hate.

Table 4.4: Example of data verification with Perspective API

Description	Manual annotation	Perspective API
Los vagos Vera incluyendo el Junior @Polificcion adorando a los indígenas en tiempos de Correa, hoy? Indios vagos.	Hate	Hate
#ParoNacional no se confundan, los indígenas no son gente que piensen en el país, mucho menos el líder Iza, si no me creen adivinen quienes son los primeros en especular con los precios...	Hate	No hate

The results are as follows.

Amount	Annotations
3779	Handwritten annotations
1405	Perspective API

Table 4.5: Verify Hate with Perspective API

Given this, we can deduce that the Perspective API recognizes 38% of the data labeled hate as hate towards identity. This helps us verify how accurate our labeling is.

4.5 Data Preprocessing

Once our data is labeled, it is essential to process it to obtain the necessary and relevant features to distinguish one class from another. Therefore, it is crucial to clean the data, apply tokenization and stopword removal, and remove similar tweets to avoid redundant data in our models.

4.5.1 Data Cleaning

In essence, non-relevant data in our context is removed, such as:

- Removal of URLs.
- Removal of mentions.
- Removal of hashtags.
- Removal of non-alphabetic characters, including emoticons and characters, and eliminating white spaces.

Finally, the text must be converted to lowercase to avoid confusion for the model when identifying patterns, as each character is associated with numbers. However, the values of the characters “F” and “f” are not equal due to their numerical representation.

4.5.2 Tokenization and Stopwords Removal

In this part, it is essential to clarify what tokenization means: essentially, it divides the text into tokens or individual words using different methods. It is crucial to mention that these methods must be in Spanish to avoid incorrect text division. The last step is to remove stopwords. Stopwords are empty words, meaning they have no relevance in our context. These words are eliminated because they do not contribute significantly to the meaning of a sentence, i.e., they do not provide specific or distinctive information about its content. Some examples of stopwords are “el,” “la,” “de,” “en,” “y,” “o,” “un,” and “una,” among others. These words are characterized as articles, prepositions, conjunctions, and pronouns, which are generally necessary for the gra mmatical structure of the language but do not provide relevant information when analyzing the semantic content or meaning of the text.

4.5.3 Removal of Similar Tweets

The main focus of this step is to eliminate similar data to avoid duplicate data in our dataset. In our context, it removes identical tweets. For this, the TF-IDF approach and cosine similarity were used.

- **Term Frequency-Inverse Document Frequency (TF-IDF):** Term Frequency (TF) and Inverse Document Frequency (IDF) are defined as follows:

  Term Frequency

  Term Frequency (TF) measures how often a specific term appears in a document relative to the total number of terms in that document. In other words, TF quantifies the relative importance of a term in a document. TF is calculated with the following formula:

$$\mathrm{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in } d}$$

For example, consider two documents:

- Document 1: "The cat caught a mouse."
- Document 2: "The dog chased the cat."

We calculate the term frequency for the word "cat" in each document:

- For Document 1: $\text{TF}(\text{“}cat\text{”}, \text{Document 1}) = \frac{1}{5} = 0.2$
- For Document 2: $\text{TF}(\text{“}cat\text{”}, \text{Document 2}) = \frac{1}{6} \approx 0.167$

Inverse Document Frequency

Inverse Document Frequency (IDF) measures the importance of a term across the entire corpus, i.e., across all documents. Terms that appear in many documents will have a lower IDF, while terms that are rare will have a higher IDF. We calculate IDF with the following formula:

$$\text{IDF}(t, D) = \log \frac{\text{Total number of documents in corpus } D}{\text{Number of documents containing term } t + 1}$$

For example, suppose we have a corpus of 1000 documents and the term "cat" appears in 100 documents. We calculate IDF for "cat":

$\text{IDF}(\text{“}cat\text{”}, \text{Corpus}) = \log\left(\frac{1000}{100+1}\right) \approx \log 9.09 \approx 2.21$

TF-IDF

TF-IDF combines term frequency (TF) and inverse document frequency (IDF) to calculate a weight indicating the importance of a term in a document relative to the entire corpus. A high TF-IDF value means the term is important for that document, while a low value means the term is common across many documents.

The TF-IDF value is calculated as:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

For example, using the previous calculations:

- For Document 1: TF-IDF("*cat*", Document 1, Corpus) $\approx 0.2 \times 2.21 \approx 0.442$
- For Document 2: TF-IDF("*cat*", Document 2, Corpus) $\approx 0.167 \times 2.21 \approx 0.369$

These values indicate the relative importance of the term "cat" in each document in the context of the entire corpus.

- **Cosine Similarity:** The cosine similarity formula between two vectors A and B is expressed as:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Where:

- $A \cdot B$ is the dot product between vectors A and B.

- $\|A\|$ and $\|B\|$ are the magnitudes of vectors A and B, respectively.

Cosine similarity returns values between -1 and 1, where 1 indicates total similarity, and -1 indicates total dissimilarity. Values closer to 1 indicate higher similarity between documents.

In the context of identifying similar tweets, TF-IDF vectors are used to calculate cosine similarity between pairs of tweets. A similarity threshold is set to identify those pairs with a similarity greater than or equal to that value, which are considered sufficiently similar to be removed and reduce redundancy in the data. The established similarity threshold is 0.8.

So, after applying all the steps above, 3344 data points were removed, indicating that this entire quantity consisted of repeated or often reposted tweets. Subsequently, the remaining data quantities for both classes are presented after removing similar tweets.

Class	Amount
No Hate (0)	26900
Hate (1)	3202

Table 4.6: Total Number of data divided into two classes

4.6 Text augmentation

As shown in Table 4.6, the proportion of both classes is disproportionate, possibly due to the volume of informative tweets that do not express an opinion on whether it is hate or not and do not contribute anything relevant to our context. It is estimated that around 20% of the dataset contains these tweets, this was estimated manually, filtering the tweets from lowest to highest percentage of toxicity, detected with the Perspective API, and looking at the associated hastags in the tweets, given that these tweets have very little or no amount of hate, it will provide us with the hastags associated with them, then we filtered the tweets using these hastags, among which are #ATENCION, #URGENTE, #ULTIMAHORA #NOTICIAS ,as well as filtering by words, among which are "Info", "Información", "Noticia", and "Dialogo" . Thus, the "hate" class is only an eighth of the "no hate" class. Due to the immense data imbalance, this is unsuitable for adapting to a model. To manage this large imbalance, there are several options, among which are:

- Adjust parameters or metrics of the algorithm itself to try to balance the minority class.
- Generate synthetic data using different techniques such as GPT, GANs, and Synthetic Minority Over-sampling Technique (SMOTE).

Due to the large class imbalance and the number of models implemented, it is difficult to consider option 1, because each model will have different input parameters, so it was decided to generate synthetic data for a more level playing field when training and evaluating our models.

Here, the technique that stood out among the others is SMOTE due to its free access and data generation over the vector space.

Synthetic Minority Over-sampling Technique

SMOTE is a popular technique used to address the class imbalance in machine learning datasets[85]. It works by generating synthetic samples for the minority class, thereby balancing the class distribution. It is important to know how it works.

1. **Identify Minority Class:** First, we identify the minority class in the dataset. In this context, the "hate" class is considered the minority class.

2. **Selecting Samples:** SMOTE selects individual samples from the minority class that are close in the feature space.

3. **Generating Synthetic Samples:** For each selected sample, SMOTE generates synthetic samples by interpolating between the selected sample and its nearest neighbors.

4. **Balancing Classes:** By generating synthetic samples for the minority class, SMOTE helps balance the class distribution, making it more suitable for training machine learning models.

Once we are clear about our data generation technique, we propose two ways of approaching the issue of data generation on our dataset, among which are:

1. **Generate synthetic data on the overall dataset**

   It's important to mention that if we generate synthetic data to balance both classes, this data should only be generated on the dataset used to train the model. This is because synthetic data has similarities to the original data. Suppose we generate synthetic data on the entire dataset. In that case, the testing dataset will have characteristics of the training dataset due to the large amount of data generated in the "hate" class. Figure 4.3 illustrates the distribution and class imbalance in the training set.

   When applying SMOTE only on the training set, class "Hate" will generate 18906 data points, which is incorrect since it creates many data points from only 2614 data points. This would significantly compromise the credibility of our model.

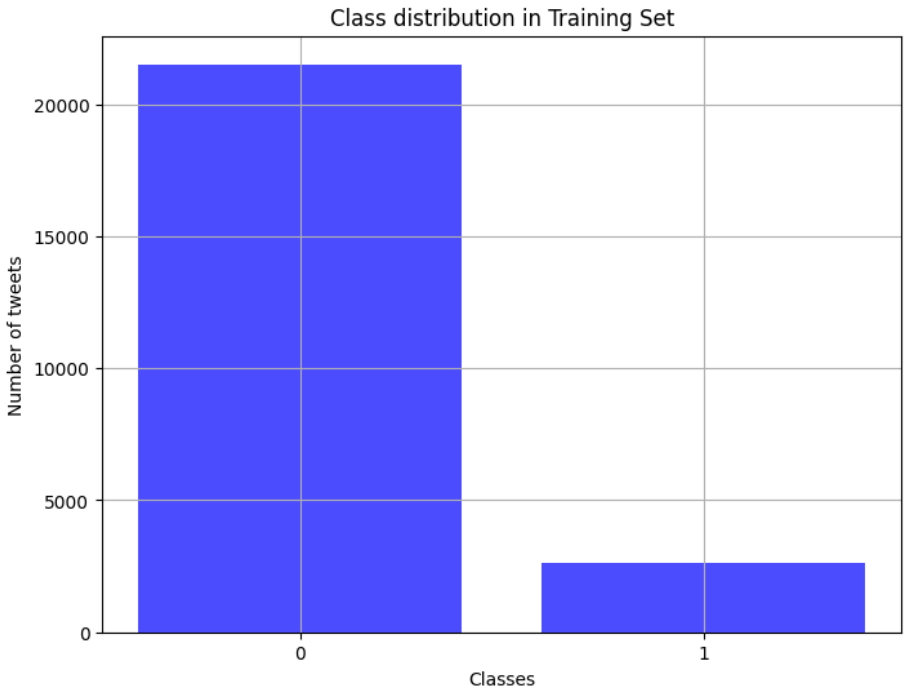

Figure 4.3: Data distribution of training set

2. **Reduce the size of the majority class and generate synthetic data**

   If we reduce the amount of data to a similar quantity as the minority class, the number of generated data points is much lower. This is often the most feasible approach due to the limited synthetic data. To achieve this, we reduce the number of data points in the majority class to 5000, as shown in Figure 4.4. This was done by filtering our data from highest to lowest, using the column that has ”identity attack” values, and selecting the first 5000 data.

In conclusion, after several experiments, the optimal approach is to use a dataset limited to 5000 due to the small amount of data generated. This is to prevent overfitting of the model, given a large number of similar generated data.

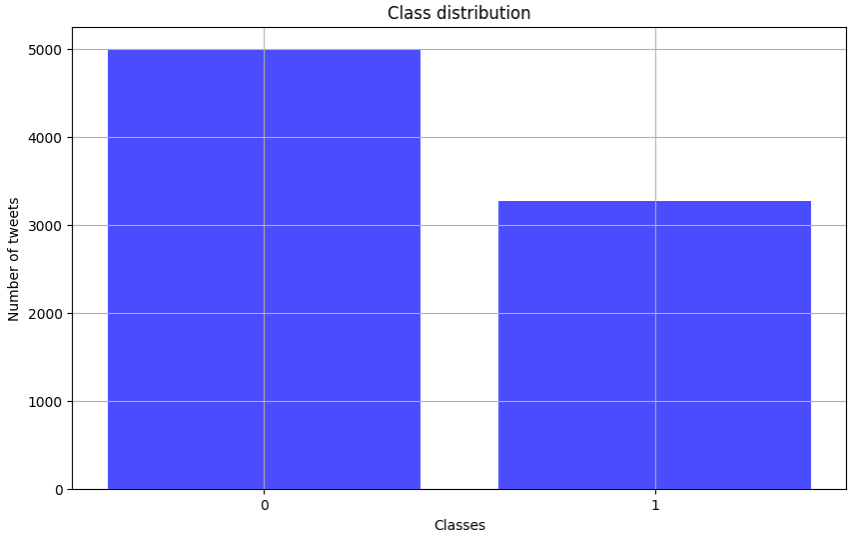

Figure 4.4: Wholesale class limited to 5000 data

4.7 Model design

To design our models, we must consider which method we will use for the proper data allocation. Thus, the second method was chosen, trimming the wholesale set to 5000 data and generating the missing data from the retail set, as depicted in Figure 4.4. The processing of the input data varies depending on the model used.

Several experiments were performed for machine learning techniques, and different classifiers were evaluated, among which are:

- Random Forest
- Support Vector Machine
- Logistic Regression
- Naive Bayes

The following models were used for deep learning:

- Sequential Neural Network(SNN).
- Pre-trained Bidirectional Encoder Representations from Transformers (BERT).

Sequential neural network architecture

A sequential artificial neural network (ANN) model with several dense layers. Uses ReLU activation in the hidden layers and Softmax in the output layer for binary classification. Dropout layers are incorporated to reduce overfitting when turning off neurons during training. It consists of an input layer with two neurons, followed by six dense layers with decreasing neuron sizes (256, 128, 64, 32, 16). The final layer has two neurons with Softmax activation for the output of two classes.

Layer Type	Neurons	Activation Function
Dense (Fully Connected)	256	ReLU
Dropout	-	Rate 0.5
Dense	128	ReLU
Dropout	-	Rate 0.5
Dense	64	ReLU
Dropout	-	Rate 0.5
Dense	32	ReLU
Dropout	-	Rate 0.5
Dense	2	Softmax

Table 4.7: Sequential Neural Network Architecture

4.7.1 Data processing

It is important to process the data after entry in different models, for both machine learning and deep learning models, the data for the models will be exactly the same.

Data processing for machine learning classifiers

For these models, not much data processing is required for the models' inputs. Essentially, after applying SMOTE in the data, the data is converted into a matrix of TF-IDF feature vectors, known as Bag of Words. This is done because the classifiers used take matrix inputs, learning from the features associated with the input vectors of the matrix through matrix operations. Having inputs as vectors is beneficial when training different models since the operations performed are linear and take less time in calculations than models that perform more matrix operations. The computation time greatly depends on the data structure.

Figure 4.5 illustrates the data split for training and testing our models, the number of data is 6562 for training, and 1641 for test.

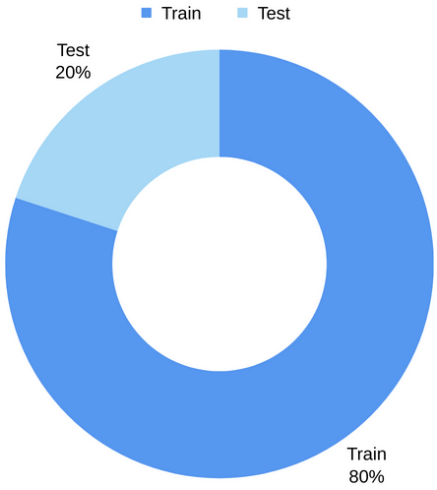

Figure 4.5: Split of the data

Data processing for deep learning

In this case, the same logic as data processing for ML models is used; however, there are significant variations when entering our Sequential Neural Network. The additional modifications that must be made include:

- After applying SMOTE to the training set, it is necessary to reorder the resulting matrix, because SMOTE generates new synthetic instances, and they are added to the end of the original data matrix and if the matrix is not reorganized, the synthetic instances may not be ordered according to their original indices.
- Subsequently, it is necessary to categorize the labels of our data into binary vector such as $[1, 0]$ and $[0, 1]$. This is done so that our model requires the labels to be in a format that it can process. This is common in deep learning models when working with texts. The labels must be modified to a format that deep learning models can process.

For our pre-trained BERT model, we used Hate-speech-CNERG/dehatebert-mono-spanish.

The trained BERT model's data processing is slightly different because it uses its tokenizer. The processing was as follows:

- Once our data is split, the text augmentation is performed on the training set, after that the tokenizer needs to be applied to convert the text sequences into encodings to feed the model.
- Then, the numerical representation of the encodings is obtained, structuring the input data.
- Finally, this input can be used to train the model, and later tested.

4.7.2 Experimentation

Taking this into consideration, it is crucial to highlight that the identical methodology was applied to all four distinct machine learning classifiers and the two deep learning models. After splitting our data, a substantial data imbalance becomes apparent in the training set:

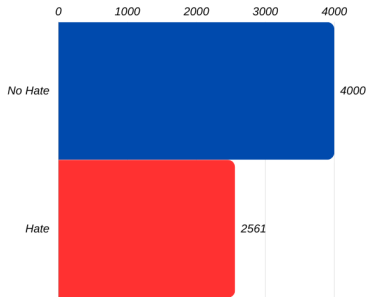

Figure 4.6: Train dataset

Therefore, the SMOTE technique was applied to generate synthetic data in class 1 on the training dataset, as shown in Figure 4.6 , thus avoiding any impact on the test set. Refraining from this approach might confuse the model since the generated data shares similarities with the original data, leading to a testing scenario with comparable datasets.

As a result, 1439 synthetic data points were generated for the hated class, considering it is the minority class. The logistic regression classifier was utilized to train our model, yielding significant outcomes. However, it is crucial to highlight the percentage of data used for testing our model, as depicted in Figure 4.7.

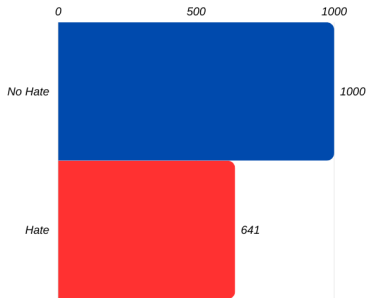

Figure 4.7: Test dataset

Subsequently, the four different ML classifiers were employed to determine which one adapted better to the data. Cross-validations were utilized to assess the specific performance of our models.

For our Deep Learning model, training was conducted with a batch size of 64 and 7 epochs. These values were established after several experiments with variations in these parameters.

Chapter 5

Results and Discussion

Before delving into the results, it is essential to recall the methodology employed in this research. The goal is to detect the amount of hate speech directed towards indigenous communities in texts, in this case, tweets. To achieve this, developing a learning model is necessary, for which various models were proposed based on different areas of artificial intelligence. Four of these models belong to the machine learning branch, and two belong to the deep learning branch.

Once this is clear, we must designate an evaluation method for our models and choose the best from each branch for a final comparison.

Quantitative results will be assessed by comparing different algorithms within the same branch and selecting the most optimal one. Once the models with the highest performance are identified, an exhaustive evaluation of both quantitative and qualitative results will be conducted. The most influential metrics in our hate speech context will be assessed using quantitative results: accuracy, precision, recall, F1-score, and AUC-ROC. The overall performance of each model will depend on various factors, such as the amount of data and hyperparameter settings associated with each model. Regarding qualitative results, a general example will be considered and tested with the two best-selected models, one from ML and one from deep learning.

The following Table 5.1 describes the results obtained for each machine learning model used and analyzes which one fits our data better. It is crucial to mention that the results were obtained after conducting a 10-fold cross-validation, each validation is represented by V_n, as described in Table 5.1. First, the overall performance of the models will be analyzed,

followed by a discussion of more detailed metrics.

Nº	SVM	LR	RF	NB
V1	0.846	0.815	0.834	0.798
V2	0.846	0.798	0.83	0.825
V3	0.848	0.78	0.811	0.816
V4	0.837	0.802	0.827	0.822
V5	0.83	0.776	0.82	0.796
V6	0.834	0.808	0.825	0.805
V7	0.815	0.792	0.813	0.81
V8	0.8425	0.786	0.815	0.798
V9	0.825	0.802	0.822	0.796
V10	0.8575	0.795	0.815	0.812
Average	0.838	0.795	0.821	0.808
General	0.78	0.79	0.76	0.77

Table 5.1: Accuracy of machine learning models

Upon analyzing this table, we can observe that the model that achieved the highest average of accuracy over cross validation was the support vector machine(SVM). In general terms, the model demonstrates high performance on the training set; however, these results might not be entirely reliable as there could be overfitting on the training set. Now, the last row refers to the evaluation of the model with test dataset, and we can see that there is no significant variation in precision between each model. Nevertheless, the logistic regression model had the highest performance with 79%.

Let's delve deeper into the performance of these four models by evaluating the most relevant metrics.

The following tables 5.2, 5.3, 5.4, and 5.5, presents the evaluation of each model with its different metrics over testing set, which are more relevant in our context. Classification reports for the four different machine learning models using various classifiers are displayed.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.85	0.80	0.82	1000
1	0.71	0.79	0.75	641
Accuracy	0.79			1641
Macro Avg	0.78	0.79	0.79	1641
Weighted Avg	0.80	0.79	0.79	1641

Table 5.2: Classification Report of Logistic Regression

Class	Precision	Recall	F1-Score	Support
0	0.78	0.89	0.83	1000
1	0.78	0.61	0.68	641
Accuracy		0.78		1641
Macro Avg	0.78	0.74	0.75	1641
Weighted Avg	0.78	0.78	0.77	1641

Table 5.3: Classification Report of Support Vector Machine

Class	Precision	Recall	F1-Score	Support
0	0.81	0.81	0.81	1000
1	0.70	0.70	0.70	641
Accuracy		0.77		1641
Macro Avg	0.76	0.76	0.76	1641
Weighted Avg	0.77	0.77	0.77	1641

Table 5.4: Classification Report of Random Forest

Class	Precision	Recall	F1-Score	Support
0	0.87	0.72	0.79	1000
1	0.66	0.84	0.74	641
Accuracy		0.77		1641
Macro Avg	0.77	0.78	0.76	1641
Weighted Avg	0.79	0.77	0.77	1641

Table 5.5: Classification Report of Naive Bayes

Table 5.2 presents the results of evaluating our model using the logistic regression classifier. There is coherence in the obtained metrics, where it is expected that class "0" would have significantly higher results than class "1," given the disparity in the evaluated data. However, it would be considered an anomaly if the results for class "1" were higher than those for class "0," as this could indicate the model to the data.

For the model presented in Table 5.3, the support vector machine (SVM) classifier was employed. Noticeable variations in the results of each metric are highlighted, suggesting that the model fails to adjust or learn important parameters from our text sequences, resulting in generally low performance.

Regarding the Random Forest classifier, represented in Table 5.4, substantial variability between classes and all evaluated metrics is observed again. This is typical for classifiers of this type, as they often require a large amount of data to achieve more accurate predictions.

Finally, Table 5.5 presents the results of the Naive Bayes classifier. Significant variation

in metric values between classes is noticed, indicating that the model performs poorly in detecting both classes. This could be attributed to the assumption of conditional independence between features given the courses by the classifier, meaning that the presence or absence of one word is considered independent of the presence or absence of other words in the document.

In summary, after a detailed analysis of the performance of each machine learning model, we can conclude that the model using the **logistic regression classifier** is the best choice for our hate speech detection problem, el modelo que . This is because logistic regression, as a linear model, effectively adapts to the relationship between features extracted through TF-IDF and the target variable, especially when this relationship is approximately linear. The linearity of the model and its ability to handle regularization, interpretability, and computational efficiency make logistic regression a solid choice in this specific context.

Now, discussing the results obtained by different deep learning models, one being a Sequential Neural Network (SNN) and the other a pre-trained BERT model for text classification in Spanish. Table 5.6 presents the results of evaluating both models.

	SNN	BERT
Accuracy	0.80	0.79
Precision	0.78	0.76
Recall	0.78	0.74
F1-score	0.78	0.75

Table 5.6: Results of deep learning models

Both models show promising results; however, the SNN model performs significantly better overall and shows minimal variation in the evaluated metrics. Now, let's examine the confusion matrix generated by each model.

Analyzing both confusion matrices, it is observed that the pre-trained BERT model outperforms in predicting hate speech directed towards indigenous communities. On the other hand, the other model based on a sequential neural network shows higher precision in classifying the "No hate" class. It is important to note that variations in both matrices are minimal.

Given the excellent performance of both models on the amount of training data, we have decided to select both the pre-trained BERT model and the model based on a sequential

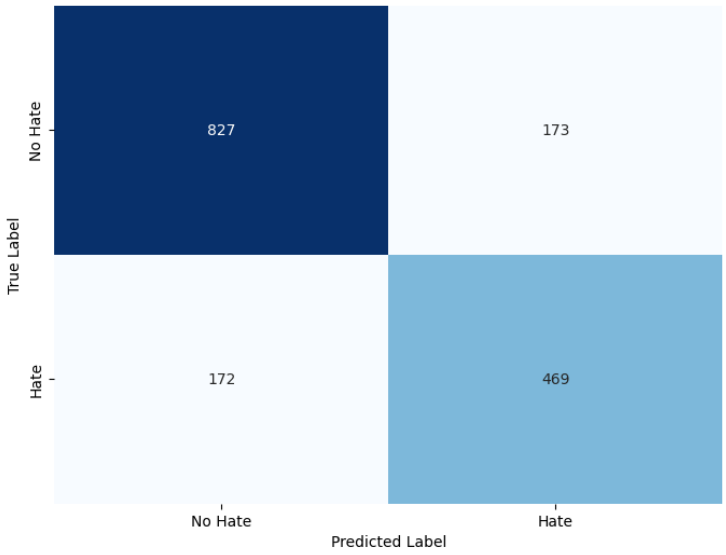

Figure 5.1: SNN model confusion matrix for test

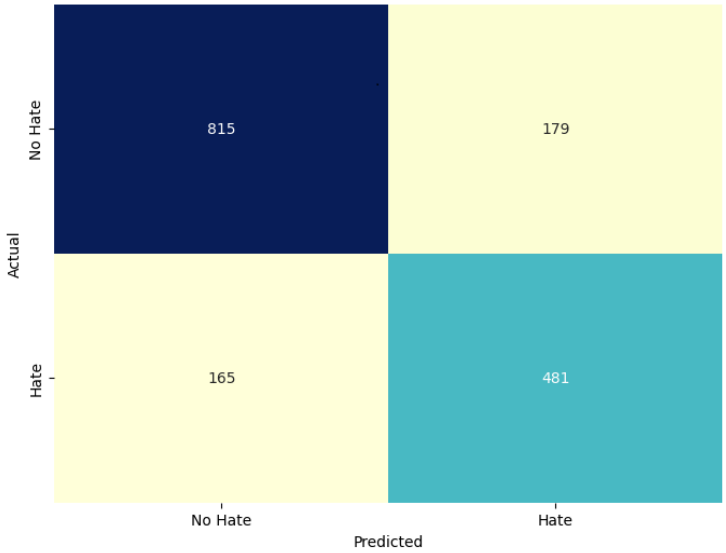

Figure 5.2: BERT model confusion matrix for test

neural network, along with the machine learning model using a logistic regression classifier, for the final testing.

To complement our evaluation, we will use an additional metric: the Area Under the Curve (AUC-ROC). This metric is handy for assessing the discriminative ability of binary classification models, especially in cases of class imbalance. A higher AUC-ROC value generally indicates better model performance.

Additionally, we will present the confusion matrix of the model using the logistic regression classifier as part of our final response.

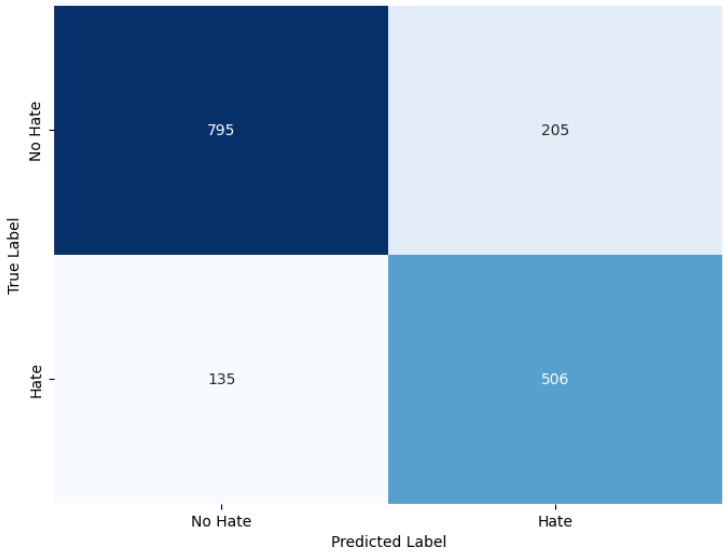

Figure 5.3: Logistic regression model confusion matrix for test

To conclude, both deep learning models outperform the performance shown in Table 5.3. However, let's analyze the AUC-ROC metric.

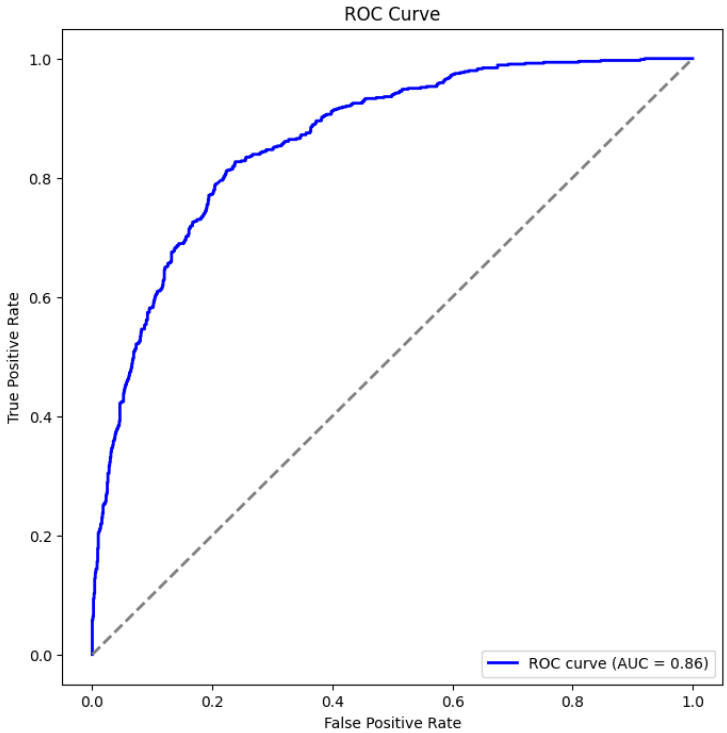

Figure 5.4: Logistic regression (AUC-ROC)

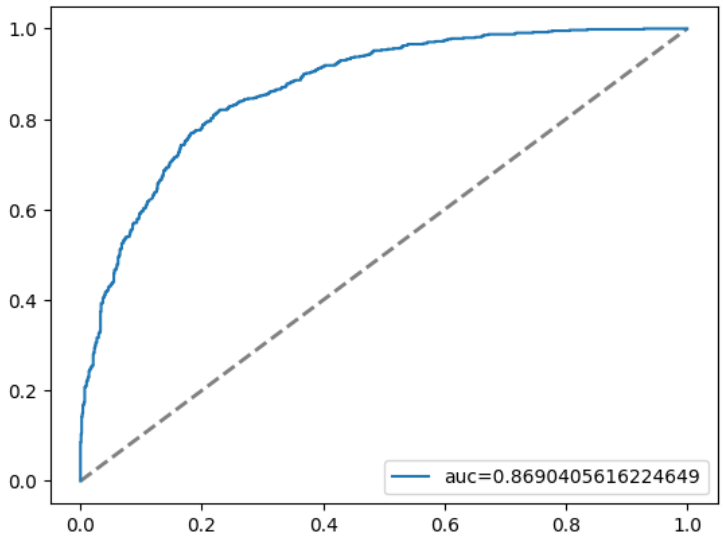

Figure 5.5: SNN (AUC-ROC)

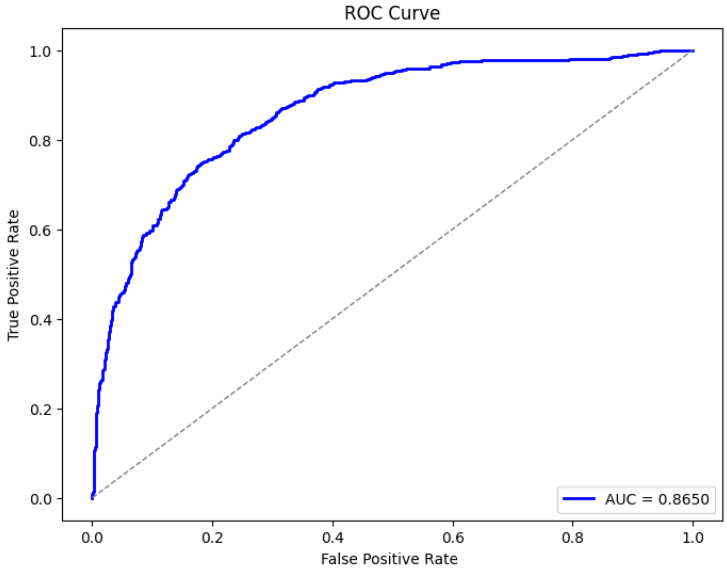

Figure 5.6: BERT (AUC-ROC)

Analyzing all the AUC-ROC values obtained by different models, it is evident that the **SNN** outperforms the pre-trained BERT model, which, in turn, surpasses the model using logistic regression.

Finally, let's examine the qualitative results of these three models. This is done using three examples of proposed texts. The first text contains no hate speech towards indigenous communities. The second text may be confusing as it contains some hate speech towards indigenous communities but also includes parts that are not directed toward them. The

last text contains strong language directed at indigenous communities.

	No hate			Hate		
Text	LR	SNN	Bert	LR	SNN	Bert
El pais está en una gran crisis, debido a las decisiones incompetente del presidente.	0.59	0.80	0.98	0.41	0.20	0.02
Los malditos indigenas salieron a destruir todo alrededor, debido al gobierno corrupto y la mala administración del presidente	0.59	0.77	0.66	0.41	0.23	0.33
Indigenas retrasa pueblos, todo el ecuador los odia #IzaTerrosista .	0.45	0.09	0.14	0.55	0.91	0.86

Table 5.7: Predictions of the three models with different texts.

Table 5.7 presents the models' predictions for various texts. The **machine learning model** exhibits the lowest precision when differentiating between classes. Its performance with entirely new data in our context is below 60%, indicating a tendency to confusion when trying to discern between categories.

In contrast, the **SNN model** demonstrates higher precision in predictions. It is notably more accurate than the **BERT model**. This model makes more precise predictions about tweets containing hate speech, effectively distinguishing the associated textual context. For example, in Text 2, both models accurately identify a percentage of hate speech. This text poses some complexity for prediction, as it indirectly contains a negative tone towards indigenous people due to the president's poor decisions. Here, the numerical values are consistent since, upon analyzing the message's semantics, the critical weight lies in government decisions, and only 0.33% of harmful content towards indigenous people can be discerned.

In conclusion, the sequential neural network (SNN) model is the best-developed model. This superior performance is reflected in metrics such as Accuracy, precision, recall, F1-score, and AUC-ROC, all of which were evaluated.

Chapter 6

Conclusions

In the fascinating technological progress that has defined our era, the proliferation of platforms for public expression has given rise to a concerning shadow: hate speech. This phenomenon, amplified by the immediacy of digital communications, poses urgent challenges regarding regulation and control. In this study, we have explored different models to address hate speech, from the tenacity of the classic logistic regression classifier to modern innovations such as BERT and sequential neural networks (SNN). The results reveal an eloquent panorama: the outstanding effectiveness of deep learning models, with the SNN particularly standing out.

The intrinsic ability of the SNN to discern linguistic complexities, adapt to nuances in discourse, and manage class imbalances provides a compelling response to the problem of hate speech. While older technologies, such as the machine learning model based on logistic regression, show signs of obsolescence in the face of the current data deluge, deep learning models point the way to a more robust and adaptive solution.

However, this study marks a milestone in the evaluated models' effectiveness and opens a window to future research. The possibilities for improvement and refinement in hate speech detection are vast. Future work can focus on exploring even more advanced model architectures, integrating more sophisticated natural language processing techniques, and continuously adapting to the changing dynamics of digital language. The promising path toward safer online discourse involves a continuous commitment to innovation and the constant evolution of our detection tools. This study represents a current achievement and a platform for future development in mitigating hate speech in cyberspace.

Bibliography

[1] B.-S. Yang, X. Di, and T. Han, "Random forests classifier for machine fault diagnosis," *Journal of mechanical science and technology*, vol. 22, pp. 1716–1725, 2008.

[2] D. A. V. Carvajal, J. E. O. Isaza, J. N. C. Puerto, O. S. N. Hemelberg, M. A. G. Alarcón, M. Jinete, and J. A. G. Alfonso, "Algoritmos para el procesamiento de imágenes implementados en el robot humanoide inmoov," *Revista EIA*, vol. 18, no. 36, pp. 12–12, 2021.

[3] D. P. Perez, R. S. Bustillos, M. Botto-Tobar, C. M. Mora *et al.*, "Análisis de imágenes de rayos x por medio de redes neuronales artificiales," *Ecuadorian Science Journal*, vol. 5, no. 1, pp. 55–60, 2021.

[4] A. Del Prete and S. Redon Pantoja, "Las redes sociales on-line: Espacios de socialización y definición de identidad," *Psicoperspectivas*, vol. 19, no. 1, pp. 86–96, 2020.

[5] E. Calvo and N. Aruguete, *Fake news, trolls y otros encantos: Cómo funcionan (para bien y para mal) las redes sociales*. Siglo XXI Editores, 2020.

[6] J. Burgess and N. K. Baym, *Twitter: A biography*. NYU Press, 2022.

[7] M. Blasco-Duatis and G. Coenders, "Análisis de sentimiento de la agenda de los partidos políticos españoles en twitter durante la moción de censura de 2018. un enfoque de datos composicionales," *Revista Mediterránea de Comunicación*, vol. 11, no. 2, pp. 185–198, 2020.

[8] M. Y. L. Vázquez, R. E. H. Cevallos, and J. E. Ricardo, "Análisis de sentimientos: herramienta para estudiar datos cualitativos en la investigación jurídica," *Universidad Y Sociedad*, vol. 13, no. S3, pp. 262–266, 2021.

[9] V. Rojo, M. F. Pollo Cattaneo, and P. V. Britos, "Análisis de sentimientos en twitter: desarrollo de recursos en el español rioplatense de argentina," in *XXII Workshop de Investigadores en Ciencias de la Computación (WICC 2020, El Calafate, Santa Cruz).*, 2020.

[10] C. Arcila-Calderón, P. Sánchez-Holgado, C. Quintana-Moreno, J. J. Amores, and D. Blanco-Herrero, "Hate speech and social acceptance of migrants in europe: analysis of tweets with geolocation//discurso de odio y aceptación social hacia migrantes en europa: análisis de tuits con geolocalización," *Comunicar*, vol. 30, no. 71, pp. 21–35, 2022.

[11] S. C. Malliquinga, "Los pueblos indígenas del ecuador desde la perspectiva del estado: un análisis crítico a partir del paro nacional de 2022," *Revista Científica Arbitrada Multidisciplinaria PENTACIENCIAS*, vol. 4, no. 5, pp. 519–538, 2022.

[12] R. U. Lara, "Paro nacional indígena y movilización social en ecuador. el trayecto de octubre 2019 a junio 2022." *Nº 34Los movimientos indígenas de Ecuador*, p. 56, 2022.

[13] A. Tontodimamma, E. Nissi, A. Sarra, and L. Fontanella, "Thirty years of research into hate speech: topics of interest and their evolution," *Scientometrics*, vol. 126, pp. 157–179, 2021.

[14] M. S. Jahan and M. Oussalah, "A systematic review of hate speech automatic detection using natural language processing." *Neurocomputing*, p. 126232, 2023.

[15] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, 2021.

[16] A. Masood and K. Ahmad, "A review on emerging artificial intelligence (ai) techniques for air pollution forecasting: Fundamentals, application and performance," *Journal of Cleaner Production*, vol. 322, p. 129072, 2021.

[17] V. Kaul, S. Enslin, and S. A. Gross, "History of artificial intelligence in medicine," *Gastrointestinal endoscopy*, vol. 92, no. 4, pp. 807–812, 2020.

[18] I. H. Sarker, "Ai-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems," *SN Computer Science*, vol. 3, no. 2, p. 158, 2022.

[19] R. K. Dhanaraj, K. Rajkumar, and U. Hariharan, "Enterprise iot modeling: supervised, unsupervised, and reinforcement learning," *Business Intelligence for Enterprise Internet of Things*, pp. 55–79, 2020.

[20] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An introductory review of deep learning for prediction models with big data," *Frontiers in Artificial Intelligence*, vol. 3, p. 4, 2020.

[21] P. Suta, X. Lan, B. Wu, P. Mongkolnam, and J. H. Chan, "An overview of machine learning in chatbots," *International Journal of Mechanical Engineering and Robotics Research*, vol. 9, no. 4, pp. 502–510, 2020.

[22] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, "Attention mechanisms in computer vision: A survey," *Computational visual media*, vol. 8, no. 3, pp. 331–368, 2022.

[23] M. Nagy, G. Lăzăroiu, and K. Valaskova, "Machine intelligence and autonomous robotic technologies in the corporate context of smes: Deep learning and virtual simulation algorithms, cyber-physical production networks, and industry 4.0-based manufacturing systems," *Applied Sciences*, vol. 13, no. 3, p. 1681, 2023.

[24] D. Janjanam, B. Ganesh, and L. Manjunatha, "Design of an expert system architecture: An overview," in *Journal of Physics: Conference Series*, vol. 1767, no. 1. IOP Publishing, 2021, p. 012036.

[25] W. Ali, W. Tian, S. U. Din, D. Iradukunda, and A. A. Khan, "Classical and modern face recognition approaches: a complete review," *Multimedia tools and applications*, vol. 80, pp. 4825–4880, 2021.

[26] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[27] H. S. Munawar, A. W. Hammad, and S. T. Waller, "A review on flood management technologies related to image processing and machine learning," *Automation in Construction*, vol. 132, p. 103916, 2021.

[28] D. Dhall, R. Kaur, and M. Juneja, "Machine learning: a review of the algorithms and its applications," *Proceedings of ICRIC 2019: Recent Innovations in Computing*, pp. 47–63, 2020.

[29] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia: case studies on organization and retrieval.* Springer, 2008, pp. 21–49.

[30] V. Nasteski, "An overview of the supervised machine learning methods," *Horizons. b*, vol. 4, pp. 51–62, 2017.

[31] V. Franzoni, A. Milani, G. Biondi, and F. Micheli, "A preliminary work on dog emotion recognition," in *IEEE/WIC/ACM International Conference on Web Intelligence-Companion Volume*, 2019, pp. 91–96.

[32] J. Manasa, R. Gupta, and N. Narahari, "Machine learning based predicting house prices using regression techniques," in *2020 2nd International conference on innovative mechanisms for industry applications (ICIMIA).* IEEE, 2020, pp. 624–630.

[33] X. Yang, X. He, J. Zhao, Y. Zhang, S. Zhang, and P. Xie, "Covid-ct-dataset: a ct scan dataset about covid-19," *arXiv preprint arXiv:2003.13865*, 2020.

[34] C. K. Reddy, V. Gopal, R. Cutler, E. Beyrami, R. Cheng, H. Dubey, S. Matusevych, R. Aichner, A. Aazami, S. Braun *et al.*, "The interspeech 2020 deep noise suppression challenge: Datasets, subjective testing framework, and challenge results," *arXiv preprint arXiv:2005.13981*, 2020.

[35] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, pp. 1–27, 2021.

[36] T. Hong, Z. Wang, X. Luo, and W. Zhang, "State-of-the-art on research and applications of machine learning in the building life cycle," *Energy and Buildings*, vol. 212, p. 109831, 2020.

[37] A. N. Elmachtoub, J. C. N. Liang, and R. McNellis, "Decision trees for decision-making under the predict-then-optimize framework," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2858–2867.

[38] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, 2021.

[39] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," *Ensemble machine learning: Methods and applications*, pp. 157–175, 2012.

[40] R. Genuer, J.-M. Poggi, R. Genuer, and J.-M. Poggi, *Random forests*. Springer, 2020.

[41] A. Mammone, M. Turchi, and N. Cristianini, "Support vector machines," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 3, pp. 283–289, 2009.

[42] C. Campbell and Y. Ying, *Learning with support vector machines*. Springer Nature, 2022.

[43] S. Nusinovici, Y. C. Tham, M. Y. C. Yan, D. S. W. Ting, J. Li, C. Sabanayagam, T. Y. Wong, and C.-Y. Cheng, "Logistic regression was as good as machine learning for predicting major chronic diseases," *Journal of clinical epidemiology*, vol. 122, pp. 56–69, 2020.

[44] X. Zou, Y. Hu, Z. Tian, and K. Shen, "Logistic regression model optimization and case analysis," in *2019 IEEE 7th international conference on computer science and network technology (ICCSNT)*. IEEE, 2019, pp. 135–139.

[45] G. Zhao, T. Wang, Y. Li, Y. Jin, C. Lang, and H. Ling, "The cascaded forward algorithm for neural network training," *arXiv preprint arXiv:2303.09728*, 2023.

[46] G. I. Webb, E. Keogh, and R. Miikkulainen, "Naïve bayes." *Encyclopedia of machine learning*, vol. 15, no. 1, pp. 713–714, 2010.

[47] H. Chen, S. Hu, R. Hua, and X. Zhao, "Improved naive bayes classification algorithm for traffic risk management," *EURASIP Journal on Advances in Signal Processing*, vol. 2021, no. 1, pp. 1–12, 2021.

[48] H. U. Dike, Y. Zhou, K. K. Deveerasetty, and Q. Wu, "Unsupervised learning based on artificial neural network: A review," in *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*. IEEE, 2018, pp. 322–327.

[49] S.-S. Learning, "Semi-supervised learning," *CSZ2006. html*, 2006.

[50] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational vision science & technology*, vol. 9, no. 2, pp. 14–14, 2020.

[51] L. Aziz, M. S. B. H. Salam, U. U. Sheikh, and S. Ayub, "Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review," *IEEE Access*, vol. 8, pp. 170 461–170 495, 2020.

[52] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[53] L. Ramos, E. Casas, C. Romero, F. Rivas-Echeverría, and M. E. Morocho-Cayamcela, "A study of convnext architectures for enhanced image captioning," *IEEE Access*, 2024.

[54] L. Koumakis, "Deep learning models in genomics; are we there yet?" *Computational and Structural Biotechnology Journal*, vol. 18, pp. 1466–1473, 2020.

[55] R. Castro, L. Ramos, S. Román, M. Bermeo, A. Crespo, and E. Cuenca, "U-net vs. transunet: Performance comparison in medical image segmentation," in *International Conference on Applied Technologies*. Springer, 2022, pp. 212–226.

[56] J. C.-W. Lin, Y. Shao, Y. Djenouri, and U. Yun, "Asrnn: A recurrent neural network with an attention model for sequence labeling," *Knowledge-Based Systems*, vol. 212, p. 106548, 2021.

[57] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[58] L. Ramos and O. Chang, "Sentiment analysis of russia-ukraine conflict tweets using roberta," *Uniciencia*, vol. 37, no. 1, pp. 421–431, 2023.

[59] M. Vergara, L. Ramos, N. D. Rivera-Campoverde, and F. Rivas-Echeverría, "Enginefaultdb: A novel dataset for automotive engine fault classification and baseline results," *IEEE Access*, vol. 11, pp. 126 155–126 171, 2023.

[60] D. Krstinić, M. Braović, L. Šerić, and D. Božić-Štulić, "Multi-label classifier performance evaluation with confusion matrix," *Computer Science & Information Technology*, vol. 1, 2020.

[61] D. Müller, I. Soto-Rey, and F. Kramer, "Towards a guideline for evaluation metrics in medical image segmentation," *BMC Research Notes*, vol. 15, no. 1, p. 210, 2022.

[62] E. Casas, L. Ramos, E. Bendek, and F. Rivas-Echeverría, "Assessing the effectiveness of yolo architectures for smoke and wildfire detection," *IEEE Access*, 2023.

[63] D. Chicco and G. Jurman, "The matthews correlation coefficient (mcc) should replace the roc auc as the standard metric for assessing binary classification," *BioData Mining*, vol. 16, no. 1, pp. 1–23, 2023.

[64] A. Aggarwal, Z. Xu, O. Feyisetan, and N. Teissier, "On primes, log-loss scores and (no) privacy," *arXiv preprint arXiv:2009.08559*, 2020.

[65] T. O. Hodson, "Root-mean-square error (rmse) or mean absolute error (mae): When to use them or not," *Geoscientific Model Development*, vol. 15, no. 14, pp. 5481–5487, 2022.

[66] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "On mean absolute error for deep neural network based vector-to-vector regression," *IEEE Signal Processing Letters*, vol. 27, pp. 1485–1489, 2020.

[67] Y. Kang, Z. Cai, C.-W. Tan, Q. Huang, and H. Liu, "Natural language processing (nlp) in management research: A literature review," *Journal of Management Analytics*, vol. 7, no. 2, pp. 139–172, 2020.

[68] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: an introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 2011.

[69] A. Tabassum and R. R. Patil, "A survey on text pre-processing & feature extraction techniques in natural language processing," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 06, pp. 4864–4867, 2020.

[70] W. A. Qader, M. M. Ameen, and B. I. Ahmed, "An overview of bag of words; importance, implementation, applications, and challenges," in *2019 international engineering conference (IEC)*. IEEE, 2019, pp. 200–204.

[71] V. K. Ayyadevara and V. K. Ayyadevara, "Word2vec," *Pro Machine Learning Algorithms: A Hands-On Approach to Implementing Algorithms in Python and R*, pp. 167–178, 2018.

[72] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[73] M. A. Paz, J. Montero-Díaz, and A. Moreno-Delgado, "Hate speech: A systematized review," *Sage Open*, vol. 10, no. 4, p. 2158244020973022, 2020.

[74] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, "Deceiving google's perspective api built for detecting toxic comments," *arXiv preprint arXiv:1702.08138*, 2017.

[75] I. Aljarah, M. Habib, N. Hijazi, H. Faris, R. Qaddoura, B. Hammo, M. Abushariah, and M. Alfawareh, "Intelligent detection of hate speech in arabic social network: A machine learning approach," *Journal of Information Science*, vol. 47, no. 4, pp. 483–501, 2021.

[76] O. Oriola and E. Kotzé, "Evaluating machine learning techniques for detecting offensive and hate speech in south african tweets," *IEEE Access*, vol. 8, pp. 21 496–21 509, 2020.

[77] A. M. U. D. Khanday, S. T. Rabani, Q. R. Khan, and S. H. Malik, "Detecting twitter hate speech in covid-19 era using machine learning and ensemble learning techniques," *International Journal of Information Management Data Insights*, vol. 2, no. 2, p. 100120, 2022.

[78] S. Abro, S. Shaikh, Z. H. Khand, A. Zafar, S. Khan, and G. Mujtaba, "Automatic hate speech detection using machine learning: A comparative study," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 8, 2020.

[79] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proceedings of the 26th international conference on World Wide Web companion*, 2017, pp. 759–760.

[80] C. Paul and P. Bora, "Detecting hate speech using deep learning techniques," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, no. 2, pp. 619–623, 2021.

[81] H. S. Alatawi, A. M. Alhothali, and K. M. Moria, "Detecting white supremacist hate speech using domain specific word embedding with deep learning and bert," *IEEE Access*, vol. 9, pp. 106 363–106 374, 2021.

[82] S. Zimmerman, U. Kruschwitz, and C. Fox, "Improving hate speech detection with deep learning ensembles," in *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.

[83] M. Mozafari, R. Farahbakhsh, and N. Crespi, "Hate speech detection and racial bias mitigation in social media based on bert model," *PloS one*, vol. 15, no. 8, p. e0237861, 2020.

[84] S. Dowlagar and R. Mamidi, "Hasocone@ fire-hasoc2020: Using bert and multilingual bert models for hate speech detection," *arXiv preprint arXiv:2101.09007*, 2021.

[85] D. Dablain, B. Krawczyk, and N. V. Chawla, "Deepsmote: Fusing deep learning and smote for imbalanced data," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Appendices

Table 1: Research papers

Authors	Overview	Input	Classes	Model	Dataset	Obser.	Metrics
Chayan Paul, Pronami Sora [80]	Detecting Hate Speech using Deep Learning Techniques	text	2	LSTM, Bi-LSTM	Kaggle	31962 tweets	Accuracy, precision, recall, F1-score
Steven Z., Chris F. and Udo K. [82]	Improving Hate Speech Detection with Deep Learning Ensembles	text	2	CNNs, LSTM	Hate Speech dataset	24802 Tweets	F1-measure
Hind S., Areej M. and Kawthar M. [81]	Detecting White Supremacist Hate Speech Using Domain Specific Word Embedding with Deep Learning and BERT	text	2	SVM, CNN, LSTM	Stormfront Supremacy	2000 tweets	Accuracy, precision, recall, F1-score
Pinkesh Badjatiya, Shashank Gupta. et al. [79]	Deep Learning for Hate Speech Detection in Tweets	Text	3	LR, RRF, SVMs, GBDTs, DNNs	Twitter	16000 tweets	Precision, recall, F1score
Sindhu Abro, Sarang Shaikh. et al. [78]	Automatic Hate Speech Detection Using Machine Learning: A Comparative Study	Text	3	SVM, KNN, DT, RF, MLP, LR	dataset is compiled and labeled by Crowd Flower	14509 tweets	Precision, recall, F-mesaure, Accuracy
Ibrahim Aljarah, Maria Habib, et.al [75]	Intelligent Detection of Hate Speech in Arabic Social Network: A Machine Learning Approach	Text	2	SVM, NB, DT, RF	own dataset collected on Twitter	3696 tweets	Accuracy, precision, recall, g-mean
Oluwafemi Oriola, Eduan Kotzé [76]	Evaluating Machine Learning Techniques for Detecting Offensive and Hate Speech in South African Tweets	Text	3	LogReg, SVM, RF, GB	own dataset collected on Twitter	21350 tweets	True positive Rate, Accuracy, Precision, Recall, F1-score
Mozafari, Marzieh & Farahbakhsh [83]	Hate speech detection and racial bias mitigation in social media based on BERT mode	Text	3	BERT	Waseem Davidson	44480 tweets	F1-score
Dowlagar, Suman & Mamidi, Radhika [84]	HASOCOne@FIRE-HASOC2020: Using BERT and Multilingual BERT models for Hate Speech Detection	Text	3	BERT	own dataset	-	Precision, F1-score