University of Piraeus
School of Information and Communication Technologies
Department of Digital Systems

Postgraduate Program of Studies
MSc Digital Systems Security

MSc Dissertation

Cyberbullying Detection through NLP & Machine Learning

Supervisor Professor: Christos Xenakis

Ioannis Bakomichalis ioannis.bakomichalis@ssl-unipi.gr mte2117

Piraeus
17/03/2023

Acknowledgements

I would like to use this opportunity to express my gratitude to my family for their unwavering support during the entirety of my studies. Moreover, I want to thank all my friends that they supported me during my whole journey in academic education.
Finally, I want to express my sincere gratitude to my supervisor Dr. Christos Xenakis and PhD Candidate & Researcher Mr. Aristeidis Farao for their patience and continuous support for both my MSc dissertation as well as postgraduate program “Digital Systems Security”.

Abstract

Due to a misunderstanding of the idea of freedom speech, the growth of social media generates several problems. One of these problems is cyberbullying, a serious global problem that has an impact on both victims as well as society in general. Cyberbullying can be defined as a form of bullying that occurs across social media platforms. There have been numerous attempts to stop, prevent or lessen cyberbullying in the literature, but because they depend on the interaction of the victims, they are workable. Consequently, it's essential to identify cyberbullying without the victim's participation. The purpose of this thesis is to analyze existing machine learning models in the cyberbullying detection field and to demonstrate our approach for cyberbullying detection. Firstly, we will make an introduction into cyberbullying, natural language processing, machine learning and its algorithms. Secondly, we will focus on our approach both theoretical as well as technological and especially in our dataset, machine learning algorithms and results and generally in the whole process of our machine learning model. Finally, we will present our web application CBDA for predicting and recognizing text as cyberbullying or not and its functionality.

Keywords: Cyberbullying, Cyberbullying Detection, Cyberbullying Dataset, Machine Learning, NLP, Classification, Algorithms

Table of Content

List of Figures

List of Plots & Tables

Acronyms

Acronym	***Full Name***
NLP	Natural Language Processing
ML	Machine Learning
NLTK	Natural Language Toolkit
FN	False Negative
FP	False Positive
TN	True Negative
TP	True Positive
LR	Logistic Regression
DT	Decision Tree
RF	Random Forest
XGB	XGBoost
NB	Naïve Bayes
SVM	Support Vector Machine
BaggingDT	Bagging Decision Tree
BoostingDT	Boosting Decision Tree
TF-IDF	Term Frequency- Inverse Document Frequency
CV	Counter Vectorizer
CBD.ML	Cyber Bullying Detection Machine Learning Model
CBDA	Cyber Bullying Detection Application

1. Introduction

Cyberbullying is a growing social problem that affects millions of people worldwide. It refers to the use of electronic communication technologies, such as social media, instant messaging, or email, to harass, intimidate , or humiliate someone. Cyberbullying can take many forms, including spreading rumors, sharing private photos or videos, or making derogatory comments online. It can have serious consequences for the victims, including depression, anxiety, and even suicide. The usage of inappropriate information on social media has drawn the attention of organizations, online communities, and social media platforms. Because of the big number of posts that are made every day, it is becoming hard to use outdated methods to detect offensive materials online. One of the most common approaches to address the issue is to train software systems that can identify offensive contents and remove them without human interruption. Consequently, an approach is to use Natural Language Processing (NLP) and Machine Learning(ML). NLP is a field of artificial intelligence that focuses on the interaction between computers and human language, while ML refers to the use of algorithms that can learn from data without being explicitly programmed.

1.1 Problem Statement

Despite the potential of Machine Learning and NLP, there are still many challenges that need to be addressed. For instance, collecting and annotating large number of datasets for cyberbullying is challenging. Furthermore, detection is some time crucial due to the topic's abstractness, especially for negative sentences and special linguistic cues. In our approach to avoid it and achieve better results, we collected a variety of datasets that they examined, edited, and cleaned with the purpose of having clean text and right linguistic cues. Moreover, we test a combination of our datasets with different NLP features and Machine Algorithms to find out the best results. Finally, we developed a Web application to justify our system with user input to examine his provided text for cyberbullying, giving information and plots for our dataset, NLP features and Machine Learning Algorithms.

1.2 Objectives

The main objective of this thesis is to develop and evaluate Machine Learning models that can automatically detect cyberbullying using NLP techniques. Specifically, we aim to:

- Find datasets for cyberbullying detection. Edit them to have the same encoding, column naming and same data type. After it, concatenate them to develop the final dataset.
- Explore different NLP techniques for feature extraction and more accurate context-based text.
- Develop and compare different Machine Learning models for cyberbullying detection, including traditional classifiers and Ensemble Learning.
- Evaluate the performance of the developed models using evaluation metrics and classification report.
- Develop a web application for cyberbullying detection using the most accurate developed model, in which the user will provide text input and the application will analyze it and predict if the context is cyberbullying or not.

1.3 Outline

The dissertation is composed of 6 different chapters. Initially, in our first chapter, we briefly introduced the problem, our motivation and objectives for cyberbullying detection through dataset selection, NLP, and Machine Learning.

The second chapter contains the comparison between the related work in cyberbullying detection. There, provided information about the research papers that have been used in the analysis and development of this thesis.

The third chapter contains all the theoretical background for cyberbullying, NLP, and Machine Learning. Initially, we present both theoretical as well as technological aspects and focus on NLP, Machine Learning, and its algorithms in the data science aspect.

In the fourth chapter, we analyze the process of the developed model, which include dataset development and selection, NLP process and ML algorithms

classification. Moreover, we provide the results, the comparison, and the evaluation of models.

In the fifth chapter, we present the CBDA web application. Initially, we describe the idea, the technologies that they used in development, as well as its functionality.

The sixth chapter summarizes the whole thesis and presents the possible future work on cyberbullying detection.

2. Related Work

There is a variety of studies in the field of cyberbullying detection, mostly aiming to provide a new approach in detection techniques or providing a comparison between the same dataset for different algorithms. We have focused on papers from 2019 to 2022 for cyberbullying detection research area.

In [1], the authors provide a guide on steps for cyberbullying detection and a whole summary of research and papers on this field with a comparison table. Alotaibi M. et al. [2] proposing a deep learning system for cyberbullying detection using Bi-GRU, transformer block and CNN with accuracy score 87.99%. In [3], authors present a study for cyberbullying detection using CNN. Muneer A. et al. [4] introduce their approach using a dataset of 37373 tweets, having accuracy 90.57% in Logistic Regression classifier. In [5], authors provide a machine learning system using BERT with accuracy 71%. Raj C. et al. [6] propose a Hybrid machine learning model using Bi-GRU and GloVe with accuracy 96.98%. In [7], authors present a model using SVM and TF-IDF with accuracy 81%. Rahman et al. [8] proposes a model for cyberbullying detection using a dataset of 26835 tweets, having accuracy 94% in Logistic Regression. In [9], authors presents their approach using XGBoost with accuracy 96.4%. Raza et al. [10] introduce his model using Voting Classifier with accuracy score 84.4%. In [11], authors provide an approach using BERT model with accuracy 91.90%. Mounir et al. [12] presents his approach using Neural Networks and accuracy 92.8%.

3. Theoretical Background

3.1 Cyberbullying

Cyberbullying, which occurs via the use of computers, mobile phones, and other electronic devices, has rapidly expanded in recent years particularly among the youth population, owing mostly to advancements in computerized technology.

This situation is not only harmful as a personal hazard, but it may also cause significant social and financial costs to businesses. According to research, 10-40% of young people have experienced it, either as a victim or as an intimidator, where they utilized the internet and electronic gadgets to harass, bully, degrade, or otherwise upset their peers. Numerous web pages, including audio, video, picture, and social media profiles, have been established to harass people. According to a ScanSafe research, up to 80% of online blogs contain improper elements, with 74% of them containing pornography in the form of video, picture, or filthy language. The number of open and private online chat platforms and forums has expanded substantially the spread of cyberbullying cases.[13]

Not only may cyberbullying texts and photos be posted anonymously and quickly circulated to a large audience, but it can also happen at any time of day or night. Furthermore, unlike physical or conventional bullying, cyberbullying may linger for centuries on the internet and cause continued harm to the victim for a long period. Another difficulty is tracing and removing the source, and eliminating inappropriate or abusive messages, texts, and images is sometimes hard after they have been uploaded or transmitted. Even unintended behaviour can spread so quickly and become viral that it frequently becomes tremendous harassment for the victim. Cyberbullies can remain anonymous in chat rooms. Most forums and chat rooms events do not require a real name to be registered or used as a guest user. As a result, the most prevalent elements that worsen this social hazard are anonymity and a lack of real monitoring on the internet.[14]

3.1.1 Cyberbullying Definition

Cyberbullying is a combination of two words, cyber and bully[15], which meaning grammatically is the activity of using the internet to harm or frighten another person, especially by sending them unpleasant messages.

Figure 1 Cyberbullying-WordCloud

Cyberbullying is bullying that takes place over digital devices like cell phones, computers, and tablets. Cyberbullying can occur through SMS, Text, and apps, or online in social media, forums, or gaming where people can view, participate in, or share content. Cyberbullying includes sending, posting, or sharing negative, harmful, false, or mean content about someone else. It can include sharing personal or private information about someone else causing embarrassment or humiliation. Some cyberbullying crosses the line into unlawful or criminal behavior.[16] The most common places where cyberbullying occurs are:

- Social Media, such as Facebook, Instagram, Snapchat, and Tik Tok.
- Text messaging and messaging apps on mobile or tablet devices.
- Instant messaging, direct messaging, and online chatting over the internet.
- Online forums, chat rooms, and message boards, such as Reddit.
- Email.
- Online gaming communities.

With the prevalence of social media and digital forums, comments, photos, posts, and content shared by individuals can often be viewed by strangers as well as acquaintances.

The content an individual shares online – both their personal content as well as any negative, mean, or hurtful content – creates a kind of permanent public record of their views, activities, and behaviour. This public record can be thought of as an online reputation, which may be accessible to schools, employers, colleges, clubs, and others who may be researching an individual now or in the future. Cyberbullying can harm the online reputations of everyone involved – not just the person being bullied, but those doing the bullying or participating in it. Cyberbullying has unique concerns in that it can be:

- Persistent – Digital devices offer an ability to immediately and continuously communicate 24 hours a day, so it can be difficult for children experiencing cyberbullying to find relief.
- Permanent – Most information communicated electronically is permanent and public, if not reported and removed. A negative online reputation, including for those who bully, can impact college admissions, employment, and other areas of life.
- Hard to Notice – Because teachers and parents may not overhear or see cyberbullying taking place, it is harder to recognize.

3.1.2 History of Cyberbullying

With the launch of inexpensive personal computers in the 1990s, traditional bullying made its way onto the internet. Children and teenagers were cyberbullied by peers and even strangers in public chat rooms and private messaging systems. The anonymity of the internet offers the ideal cover for a user to harass or threaten others without consequence. While numerous US states have passed legislation in recent years to control teen cyberbullying, the long-term consequences can be devastating, and we should be more aware of them and take proactive actions to combat them.[17]

Governments began to implement anti-bullying legislation in reaction to the 1999 Columbine school tragedy. Some of these laws made cyberbullying a crime, while many did not. As many young suicides were caused by internet abuse, cyberbullying

was introduced into the public. One of the earlier examples happened in 2007 when 13-year-old Megan Meier died by herself after neighbors made a phony Myspace page under the name “Josh Evans” to torment her. The culprits were convicted guilty of conspiracy and unlawful computer usage by a federal grand jury, but they were eventually acquitted. Meier’s case prompted her native state of Missouri to enact an anti-harassment statute that included cyberbullying. [18]

3.1.3 Why Study Cyberbullying?

Cyberbullying has a significant impact on and undermines the targeted person’s right to equality and freedom. While this is sufficient motivation to fight it, it has demonstrated that its long-term consequences are potentially catastrophic if no action is taken. Cyberbullying fosters prejudice and hatred and has the potential to shake the foundations of societies by creating gaps between social groups, potentially leading to deep fractures in social cohesion. The popularity and ongoing growth of online communities has also contributed to an increase in hateful behavior. The ability to post and interact, mostly anonymously or without providing much personal information, acts as an inducement to express unpopular and hostile opinions without fear of repercussions. Governments, particularly social media platforms, have been attempting to devise effective solutions to combat cyberbullying; however, the lack of studies and research automatically identifying and detecting these behaviors makes it difficult to achieve significant results. As a result, it is critical to contribute solutions for automatic Cyberbullying detection in text.[19]

Furthermore, this has had a negative impact on organizations and the economy as a whole, putting additional strain on security officers. For a variety of reasons, the latter is facing increasing challenges. For example, cyberbullying can happen all day or all year and reach a child when they are alone.

3.2 Natural Language Processing (NLP)

Everything we say (verbally or in writing) contains a wealth of information. Everything we do, from the topic we choose to our tone and word choice, adds some type of information that can be interpreted, and value extracted from it. In theory, we can use that data to understand and even predict human behavior. However, there is a problem: in a declaration, one person may generate hundreds or thousands of words, each sentence with its own level of complexity. When it comes to scaling and analyzing hundreds, thousands, or millions of people or declarations in a given geography, the situation becomes unmanageable.[21]

Unstructured data can be generated by conversations, declarations, or even tweets. Unstructured data, which accounts for the vast majority of data available in the real world, does not fit neatly into the traditional row and column structure of relational databases. It's disorganized and difficult to manipulate. Nonetheless, advances in disciplines such as machine learning are causing a major revolution in this field. Nowadays, it is more important to understand the meaning behind the words than to try to interpret a text or speech based on its keywords (the old-fashioned mechanical way) (the cognitive way). This allows for the detection of figures of speech such as irony, as well as sentiment analysis.

Natural language Processing (NLP) is all about making computers learn, process, and manipulate natural languages. Natural Language Processing is a popular branch of AI that assists Data Scientists in extracting insights from textual data. Natural Language Processing professionals are in high demand as a result of this. Everything we say or express contains valuable information that can be used to make wise decisions. However, extracting this information with a machine is difficult because humans can use multiple languages, words, tones, and so on. All of the data we generate in our daily lives through our conversations is highly unstructured. However, modern data science and Natural Language Processing methods have made it possible for machines to interact with people in meaningful ways.[20]

3.2.1 Definition of NLP

Natural language processing (NLP) is a branch of computer science that is concerned with giving computers the ability to understand text and spoken words in the same way that humans do. NLP combines computational linguistics (human language rule-based modeling) with statistical, machine learning, and deep learning models. These technologies, when combined, allow computers to process human language in the form of text or voice data and 'understand' its full meaning, complete with the speaker's or writer's intent and sentiment.[22]

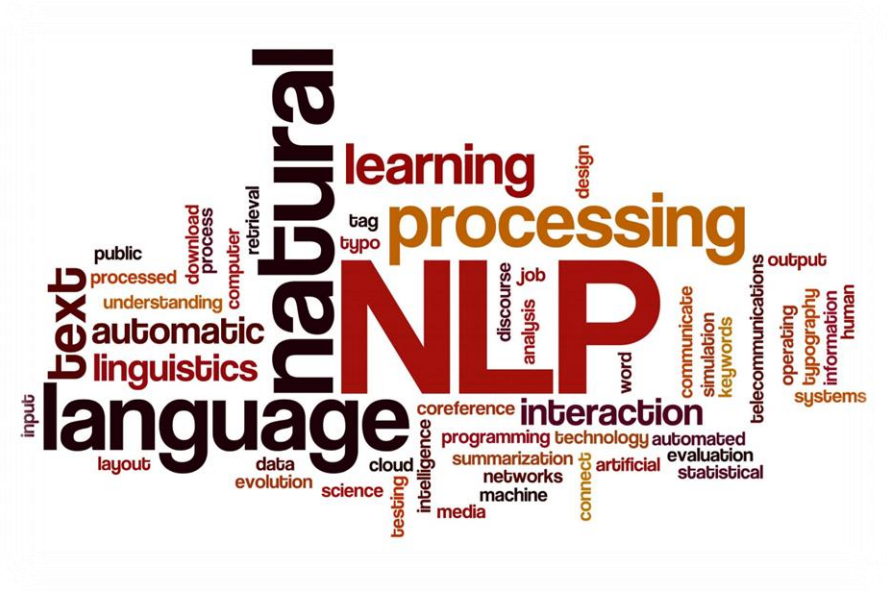


Figure 2 NLP-WordCloud

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. You've probably encountered NLP in the form of voice-activated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. However, NLP is increasingly being used in enterprise solutions to help streamline business operations, boost employee productivity, and simplify mission-critical business processes.

3.2.2 History of NLP

Natural language processing has its origins in the late 1940s. It is generally agreed that Weaver's memorandum (Shannon and Weaver, 1949) brought the idea of the first computer-based application related to natural language: machine translation (MT). It subsequently inspired many projects, notably the Georgetown experiment (Dostert, 1955), a joint project between IBM and Georgetown University that successfully demonstrated the machine translation of more than 60 Russian sentences into English. The researchers accomplished this feat by using hand-coded language rules, but the system failed to scale up to general translation. In fact, early work in MT was very simple: most systems used dictionary-lookup of appropriate words for translation and reordered the words after translation to fit the word-order rules of the target language. The researchers gradually realized that the task was far more difficult than they had anticipated, and that they required a more adequate theory of language. It took until 1957 for Chomsky (Chomsky, 1957) to introduce the concept of generative grammar, a rule-based system of syntactic structures that provided insight into how mainstream linguistics could aid machine translation.[23]

The 1950s were flooded with over-enthusiasm due to the development of syntactic theory of language and parsing algorithms. People expected fully automatic high quality translation systems to produce results that were indistinguishable from those produced by human translators, and that such systems would be operational within a few years. Given the available linguistic knowledge and computer systems at the time, this thought was completely implausible. After more than a decade of research and millions of dollars, machine translations were still more expensive than manual human translations in 1966, and there were no computers capable of carrying on a basic conversation. That same year, the ALPAC issued a report (Pierce et al., 1966) concluding that MT was not immediately achievable and recommending that the research community discontinue funding for it. This had the effect of significantly slowing down not only MT research, but also the majority of work in other NLP applications. [24]

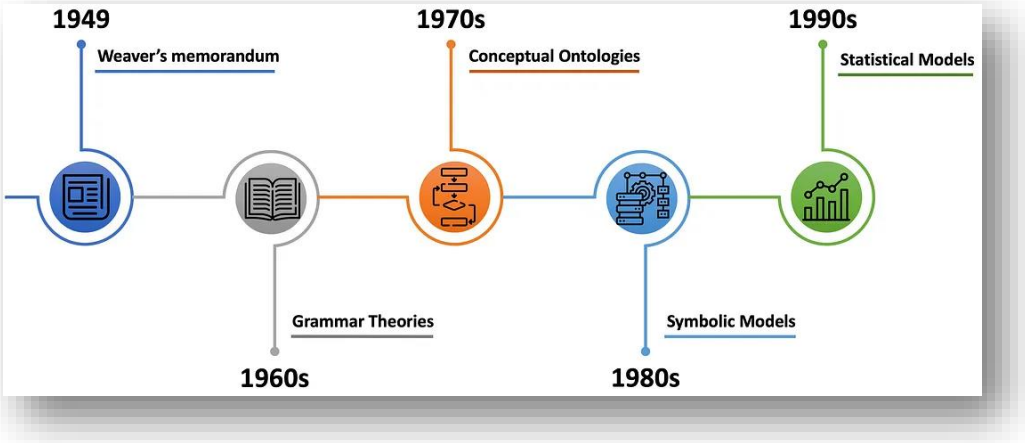


Figure 3NLP History 1949-1999

Despite this significant slowdown, some interesting developments were born during the years following the ALPAC report, both in theoretical issues and in construction of prototype systems. Theoretical work in the late 1960s and early 1970s mainly focused on how to represent meaning. Researchers developed new theories of grammar that were computationally tractable for the first time, particularly after the introduction of transformational generative grammars (Chomsky, 1965), which were criticized for being too syntactically oriented and not lending themselves easily to computational implementation. As a result, many new theories appeared to explain syntactic anomalies and provide semantic representations, such as case grammar (Fillmore, 1968), semantic networks (Collins et al., 1969), augmented transition networks (Woods, 1970), and conceptual dependency theory (Schank, 1972). Alongside theoretical development, this period of time also saw the birth of many interesting prototype systems. ELIZA (Weizenbaum, 1966) was built to replicate the conversation between a psychologist and a patient, simply by permuting or echoing the user input. SHRDLU (Winograd, 1971) was a simulated robot that used natural language to query and manipulate objects inside a very simple virtual micro-world consisting of a number of color blocks and pyramids. LUNAR (Woods et al., 1972) was developed as an interface system to a database that consisted of information about lunar rock samples using augmented transition network. Lastly, PARRY (Colby, 1974)

attempted to simulate a person with paranoid schizophrenia based on concepts, conceptualizations, and beliefs.

The 1970s brought new ideas into NLP, such as building conceptual ontologies which structured real-world information into computer-understandable data. Examples are MARGIE (Schank and Abelson, 1975), TaleSpin (Meehan, 1976), QUALM (Lehnert, 1977), SAM (Cullingford, 1978), PAM (Schank and Wilensky, 1978) and Politics (Carbonell, 1979).

Many significant problems in NLP were addressed in the 1980s using symbolic approaches, i.e., complex hard-coded rules and grammars to parse language (Charniak, 1983; Dyer, 1983; Riesbeck and Martin, 1986; Grosz et al., 1987; Hirst, 1987). Text was practically segmented into meaningless tokens (words and punctuation). After that, manual representations were created by assigning meanings to these tokens and their mutual relationships using well-known knowledge representation schemes and associated algorithms. Eventually, these representations were used to conduct in-depth analyses of linguistic phenomena.

Statistical models did not arrive as a revolution in NLP until the late 1980s and early 1990s (Bahl et al., 1989; Brill et al., 1990; Chitrao and Grishman, 1990; Brown et al., 1991), replacing most natural language processing systems based on complex sets of hand-written rules. This advancement was made possible by both the continuous increase in computational power and the shift to machine learning algorithms. While some of the earliest-used machine learning algorithms, such as decision trees (Tanaka, 1994; Allmuallim et al., 1994), produced systems that performed similarly to old-school hand-written rules, statistical models broke through the complexity barrier of hand-coded rules by creating them through automatic learning, leading to an increased focus on these models in research. These statistical models were capable of making soft, probabilistic decisions at the time.

Beginning in the 2000s, neural networks are used for language modeling, which is the task of predicting the next word in a text given the previous words. Bengio et al. proposed the first neural language model in 2003, which is a one-hidden layer feed-forward neural network. They were also among the first to introduce word embedding, or a real-valued word feature vector in Rd. More specifically, their model used vector representations of the n previous words as input, which were then looked up in a table learned alongside the model. The vectors were fed into a hidden layer, the output of which was then fed into a SoftMax layer, which predicted the next word in the

sequence. Although classic feed-forward neural networks have been gradually replaced by recurrent neural networks (Mikolov et al., 2010) and long short-term memory networks (Graves, 2013) for language modeling, they remain competitive with recurrent architectures in some settings, the latter being influenced by "catastrophic forgetting" (Daniluk et al., 2017). Furthermore, Bengio et al.'s general network's building blocks can still be found in most neural language and word embedding models today. [25]

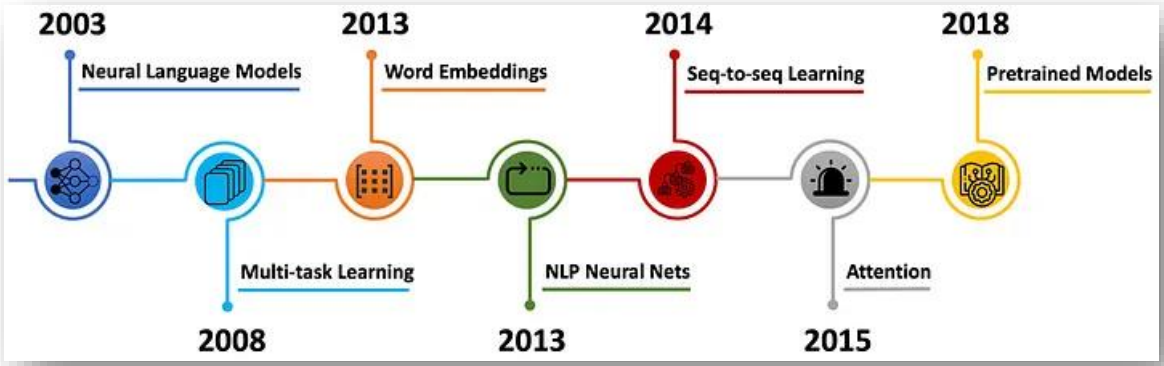


Figure 4 NLP History 2000-Today

In 2008, Collobert and Weston applied multi-task learning, a sub-field of machine learning in which multiple learning tasks are solved at the same time, to neural networks for NLP. They used a single convolutional neural network architecture (CNN; LeCun et al., 1999) that, given a sentence, was able to output many language processing predictions such as part-of-speech tags, named entity tags and semantic roles. The entire network was trained jointly on all the tasks using weight-sharing of the look-up tables, which enabled the different models to collaborate and share general low-level information in the word embedding matrix. As models are being increasingly evaluated on multiple tasks to gauge their generalization ability, multi-task learning has gained in importance and is now used across a wide range of NLP tasks. Also, their paper turned out to be a discovery that went beyond multi-task learning. It spearheaded ideas such as pre-training word embeddings and using CNNs for text, which have only been widely adopted in the last years.

In 2013, Mikolov et al. introduced Word2Vec, arguably the most popular word embedding model. Although dense vector representations of words were used as early as 2003 (Bengio et al.), the main innovation proposed in their paper was an efficient improvement of the training procedure by removing the hidden layer and approximating the objective. This year also, neural network models were also adopted in NLP, specifically three well-defined types of neural networks: recurrent neural networks (RNNs; Elman, 1990), convolutional neural networks (CNNs), and recursive neural networks (Socher et al., 2013). RNNs became popular for dealing with the dynamic input sequences that are common in NLP due to their architecture.

In 2014, Sutskever et al. proposed sequence-to-sequence learning, a general end-to-end approach for mapping one sequence to another using a neural network. In their method, an encoder neural network processes a sentence symbol by symbol and compresses it into a vector representation. Then, a decoder neural network predicts the output sequence symbol by symbol based on the encoder state and the previously predicted symbols that are taken as input at every step. Encoders and decoders for sequences are typically based on RNNs, but other architectures have also emerged.

In 2015, Bahdanau et al. introduced the principle of attention, which is one of the core innovations in neural machine translation (NMT) and the key idea that enabled NMT models to outperform classic sentence-based MT systems. It essentially removes the main bottleneck of sequence-to-sequence learning, which is the need to compress the entire content of the source sequence into a fixed-size vector.

Large pretrained language models are without a doubt the most recent major innovation in the world of NLP. While they were first proposed in 2015 (Dai and Le), they were only recently shown to provide a significant improvement over state-of-the-art methods across a wide range of tasks. Pre-trained language model embeddings can be used as features in a target model (Peters et al., 2018), or a pre-trained language model can be fine-tuned on target task data (Devlin et al., 2018; Howard and Ruder, 2018; Radford et al., 2019; Yang et al., 2019), allowing for efficient learning with significantly less data (Peters et al., 2018).

3.2.3 Tasks of NLP

The ambiguities in human language make it extremely difficult to write software that accurately determines the intended meaning of text or voice data. Homonyms, homophones, sarcasm, idioms, metaphors, grammar and usage exceptions, sentence structure variations—these are just a few of the human language irregularities that take humans years to learn, but that programmers must teach natural language-driven applications to recognize and understand accurately from the start if those applications are to be useful.[21] Several NLP tasks deconstruct human text and voice data to assist the computer in making sense of what it is ingesting. Among these tasks are the following: [22]

- **Speech Recognition**, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that responds to voice commands or questions. The way people talk makes speech recognition especially difficult, slurring words together, with varying emphasis and intonation, in different accents, and frequently using incorrect grammar.
- **Part of speech tagging**, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies ‘make’ as a verb in ‘I can make a paper plane,’ and as a noun in ‘What make of car do you own?’
- **Word sense disambiguation** is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in ‘make the grade’ (achieve) vs. ‘make a bet’ (place).
- **Named entity recognition**, or NEM, identifies words or phrases as useful entities. NEM identifies ‘Kentucky’ as a location or ‘Fred’ as a man's name.
- **Co-reference resolution** is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., ‘she’ = ‘Mary’), but

it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).

- **Sentiment analysis** attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.
- **Natural language generation** is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

3.2.4 Use Cases of NLP

Natural language processing is the driving force behind machine intelligence in many modern real-world applications.[22] Here are a few examples:

- **Spam detection**: You may not think of spam detection as an NLP solution, but the best spam detection technologies use NLP's text classification capabilities to scan emails for language that often indicates spam or phishing. These indicators can include overuse of financial terms, characteristic bad grammar, threatening language, inappropriate urgency, misspelled company names, and more. Spam detection is one of a handful of NLP problems that experts consider 'mostly solved' (although you may argue that this doesn't match your email experience).
- **Machine translation**: Google Translate is an example of widely available NLP technology at work. Truly useful machine translation involves more than replacing words in one language with words of another. Effective translation has to capture accurately the meaning and tone of the input language and translate it to text with the same meaning and desired impact in the output language. Machine translation tools are making good progress in terms of accuracy. A great way to test any machine translation tool is to translate text to one language and then back to the original. An oft-cited classic example: Not long ago, translating "The spirit is willing but the flesh is weak" from English to Russian and back yielded "The vodka is good but the meat is rotten." Today,

the result is “The spirit desires, but the flesh is weak,” which isn’t perfect, but inspires much more confidence in the English-to-Russian translation.

- **Virtual agents and chatbots**: Virtual agents such as Apple's Siri and Amazon's Alexa use speech recognition to recognize patterns in voice commands and natural language generation to respond with appropriate action or helpful comments. Chatbots perform the same magic in response to typed text entries. The best of these also learn to recognize contextual clues about human requests and use them to provide even better responses or options over time. The next enhancement for these applications is question answering, the ability to respond to our questions—anticipated or not—with relevant and helpful answers in their own words.
- **Social media sentiment analysis**: NLP has become an essential business tool for uncovering hidden data insights from social media channels. Sentiment analysis can analyze language used in social media posts, responses, reviews, and more to extract attitudes and emotions in response to products, promotions, and events–information companies can use in product designs, advertising campaigns, and more.
- **Text summarization**: Text summarization uses NLP techniques to digest huge volumes of digital text and create summaries and synopses for indexes, research databases, or busy readers who don't have time to read full text. The best text summarization applications use semantic reasoning and natural language generation (NLG) to add useful context and conclusions to summaries.

3.2.5 NLP in Data Science

There are many NLP techniques, that are used on Data Science field. Some of them are:

- **Bag of Words** is a commonly used model that allows you to count all words in a piece of text. Basically, it creates an occurrence matrix for the sentence or document, disregarding grammar, and word order. These word frequencies or occurrences are then used as features for training a classifier. [20]

	words	rain	a	paper	they	slip	the	universe	...
Words are flowing out like endless rain into a paper cup,	1	1	1	1	0	0	0	0	...
They slither while they pass, they slip away across the universe	0	0	0	0	3	1	1	1	...

Figure 5 NLP - Bag of Words

- **Tokenization** is one of the NLP techniques that segments the entire text into sentences and words. In other words, we can say that it is a process of dividing the text into segments called tokens. This process discards certain characters like punctuation, hyphens, etc. The main purpose of tokenization is to convert the text into a format that is more convenient for analysis.

Figure 6 NLP – Tokenization

- **Stop Words Removal** includes getting rid of common language articles, pronouns, and prepositions such as “and”, “the” or “to” in English. In this process some very common words that appear to provide little or no value to the NLP objective are filtered and excluded from the text to be processed, hence removing widespread and frequent terms that are not informative about the corresponding text. Stop words can be safely ignored by carrying out a lookup in a pre-defined list of keywords, freeing up database space and improving processing time. [21]

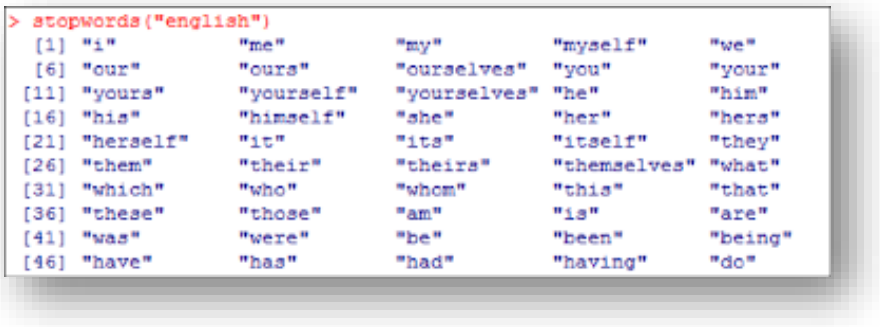


Figure 7 NLP - Stop Words

- **Stemming** is the process of reducing words to their root form. The stemming technique operates on the premise that words with slightly different spellings, but the same meaning should be placed in the same token. For efficient processing, the affixes are removed during stemming.[26]

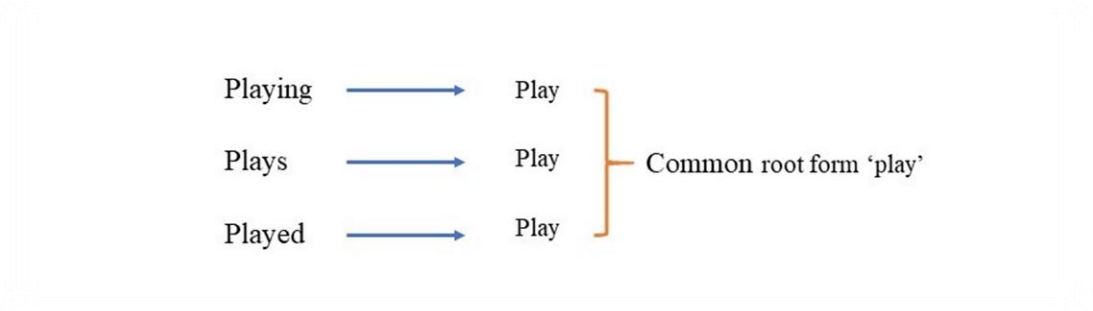


Figure 8 NLP – Stemming

- **Lemmatization** is the process of converting the words into lemma which is the dictionary form of the word. For example, “Hates”, and “hating” are forms of the word “hate”. So “hate” will be the lemma for these words. The Lemmatization technique aims at converting the different forms of a word to their root form and grouping them together. The aim of stemming and lemmatization is quite similar, but the approaches are different.

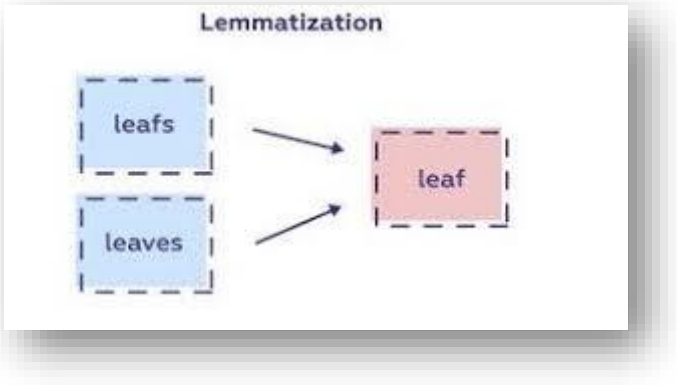


Figure 9 NLP – Lemmatization

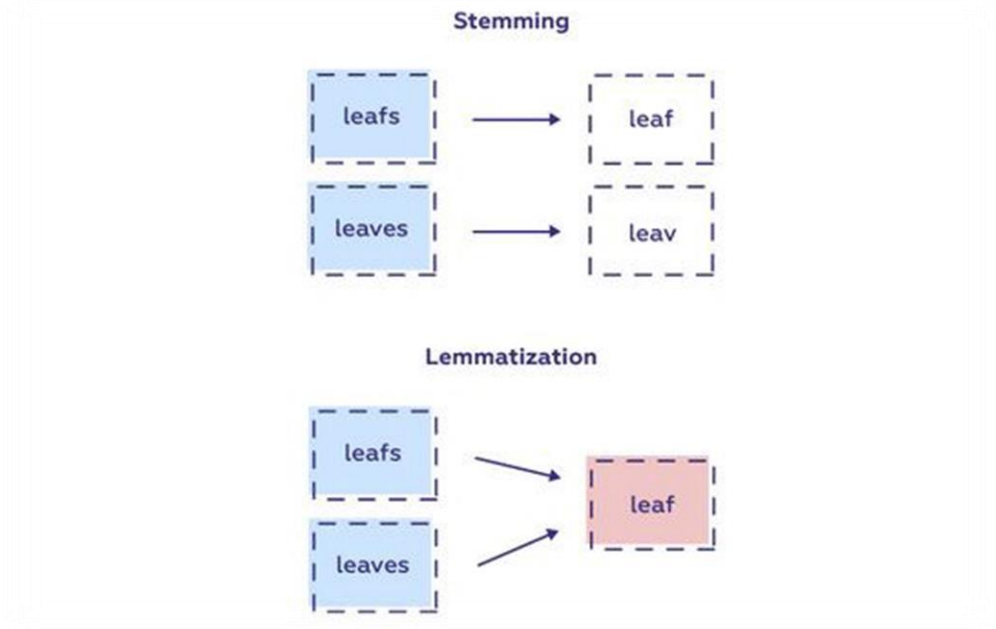


Figure 10 NLP - Stemming vs Lemmatization

- **Keyword Extraction** is a text analysis technique that automatically extracts the most used and most important words and expressions from a given text. It helps summarize the content of texts and recognize the main topics discussed.

Figure 11 NLP - Keywords Extraction

- **Word Embeddings** is a technique of representing the words of a document in the form of numbers. It should be represented in a way that similar words have a similar representation. It is a technique where individual words of a domain or language are represented as real-valued vectors in a lower dimensional space. It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.

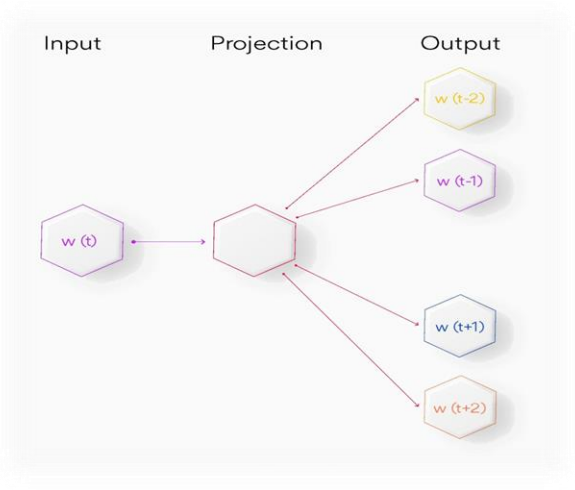


Figure 12 NLP - Word Embeddings

3.2.6 NLP Tools in Python

The Python programming language includes a plethora of tools and libraries for solving specific NLP tasks.[28] The most well-known NLP tools are described below:

- **Natural Language Toolkit (NLTK)** with Python is a leading tool for building NLP models. NLTK, which focuses on NLP research and education, is supported by an active community, as well as a variety of tutorials for language processing, sample datasets, and resources such as a comprehensive Language Processing and Python handbook.[27] Although mastering this library takes time, it is regarded as an excellent playground for gaining hands-on NLP experience. NLTK's modular structure allows it to provide many components for NLP tasks such as tokenization, tagging, stemming, parsing, and classification.[29]

Figure 13 NLP – NLTK

- **TextBlob** is a must-have for Python developers who are just getting started with NLP and want to make the most of their first encounter with NLTK. It essentially provides a simple interface for beginners to learn most basic NLP tasks such as sentiment analysis, pos-tagging, and noun phrase extraction.[30]

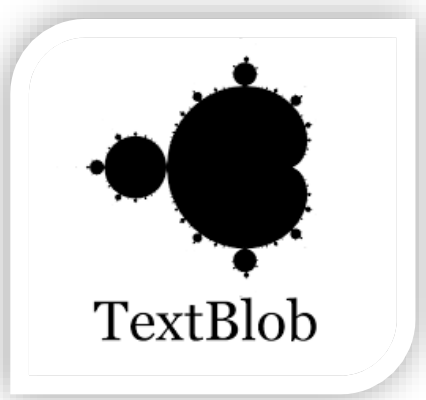


Figure 14 NLP – TextBlob

- **CoreNLP** is a Java-written library, which was created at Stanford University. Nonetheless, it comes with wrappers for a variety of languages, including Python. It can therefore be helpful for developers who want to experiment with Python's NLP capabilities. The main advantage of CoreNLP is that operates effectively in contexts for product development and is extremely quick. Moreover, various CoreNLP components can be coupled with NLTK, which is sure to increase the latter's effectiveness. [31]

Figure 15 NLP – CoreNLP

- **Gensim** is a Python library that specializes in identifying semantic similarity between two documents through vector space modeling and topic modeling toolkit. It can handle large text corpora with the help of efficiency data streaming and incremental algorithms, which is more than we can say about other packages that only target batch and in-memory processing.[32]

Figure 16 NLP – Gensim

- **SpaCy** is a relatively young library was designed for production usage. That's why it's so much more accessible than other Python NLP libraries like NLTK. SpaCy offers the fastest syntactic parser available on the market today. Moreover, since the toolkit is written in Python, it's also really speedy and efficient. [33]

Figure 17 NLP – SpaCy

- **Scikit-learn** is handy NLP library provides developers with a wide range of algorithms for building machine learning models. It offers many functions for using the bag-of-words method of creating features to tackle text classification problems. The strength of this library is the intuitive classes methods. Also, scikit-learn has an excellent documentation that helps developers make the most of its features. [34]

Figure 18 NLP - Scikit- learn

- **Polyglot** library offers a broad range of analysis and impressive language coverage. Thanks to NumPy, it also works really fast. Using polyglot is similar to spaCy – it's very efficient, straightforward, and basically an excellent choice for projects involving a language spaCy doesn't support. The library stands out from the crowd also because it requests the usage of a dedicated command in the command line through the pipeline mechanisms. [35]

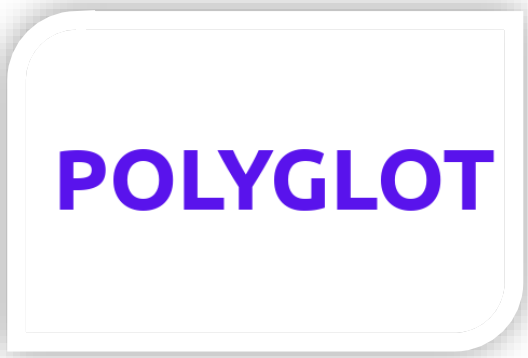


Figure 19 NLP - Polyglot

3.3 Machine Learning

Machine learning is now everywhere, whether you realize it or not: automated translation, image recognition, voice search technology, self-driving cars, and more. Machine learning has truly taken off in recent years, fueled by advances in statistics and computer science, as well as better datasets and the growth of neural networks.

A computer engineer writes a series of instructions that instruct a computer on how to transform input data into a desired output in traditional programming. The majority of instructions follow an IF-THEN structure: when certain conditions are met, the program performs a specific action. Machine learning, on the other hand, is a computer-assisted process that allows machines to solve problems with little or no human intervention and take actions based on previous observations.[36]

While the terms artificial intelligence and machine learning are frequently used interchangeably, they are not the same thing. AI is a broader concept that refers to machines making decisions, learning new skills, and solving problems in the same way that humans do, whereas machine learning is a subset of AI that allows intelligent systems to learn new things on their own.

Machine learning (ML) is a branch of artificial intelligence (AI) that enables computers to "self-learn" from training data and improve over time, without being explicitly programmed. Machine learning algorithms are able to detect patterns in data and learn from them, in order to make their own predictions. In short, machine learning algorithms and models learn through experience.

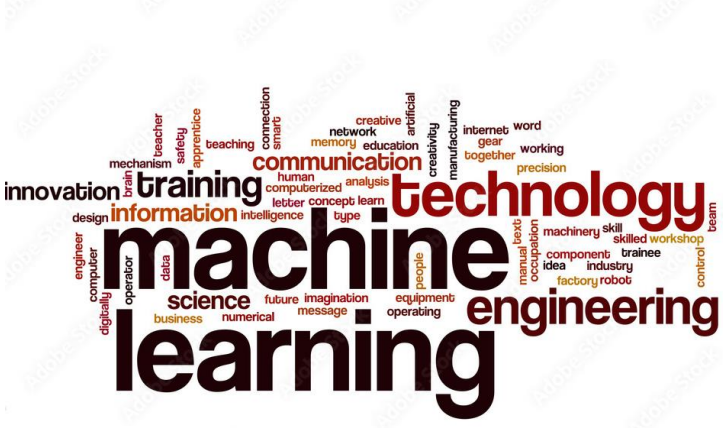


Figure 20 Machine Learning - Wordcloud

3.3.1 Definition of Machine Learning

The term machine learning was first coined in the 1950s when Artificial Intelligence pioneer Arthur Samuel built the first self-learning system for playing checkers. He noticed that the more the system played, the better it performed. He defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed ". However, there is no universally accepted definition for machine learning. Different authors define the term differently. We give below two more definitions.[37]

Machine learning is programming computers to optimize a performance criterion using example data or past experience . We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data. As a definition, Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. [38]

Machine learning is classified into three types:

- **Supervised Learning** is the process of training a model on labeled data.
- **Unsupervised Learning** is the process of training a model on unlabeled data.
- **Reinforcement Learning** involves training a model through trial and error.[39]

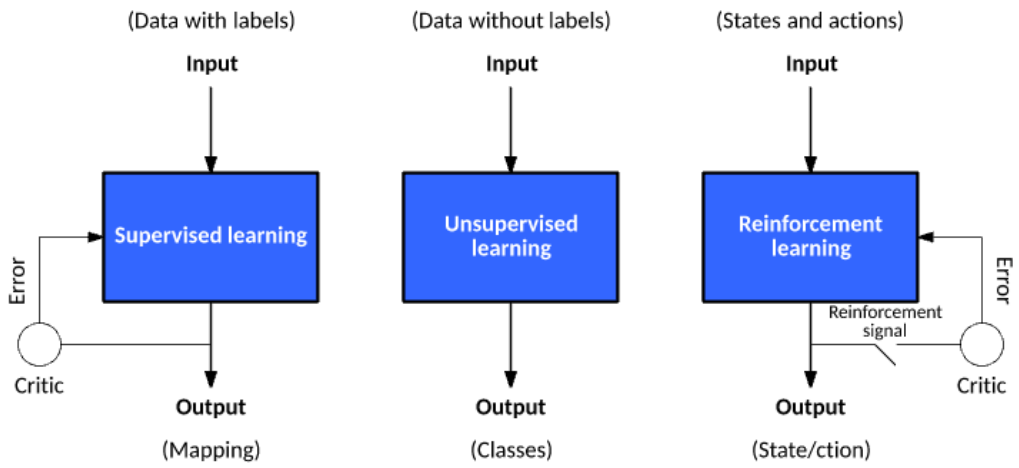


Figure 21 Supervised vs Unsupervised vs Reinforcement Learning

3.3.2 History of Machine Learning

Machine Learning is not a new concept; it was invented many years ago. Machine learning is benefiting us every day, from voice-activated home appliances to self-driving cars, online marketing, and healthcare. Mathematicians, great thinkers, and scholars all contributed to the foundation of how to begin machine learning.[40]

- **1943 – The First Mathematical Model of a Biological Neuron**

  Walter Pitts and Warren McCulloch created the first mathematical model of neural networks in 1943. Their scientific paper, "A Logical Calculus of the Ideas Immanent in Nervous Activity," was used to create algorithms that mimic human thought processes. This McCulloch Pitts Neuron has very limited capability and has no learning mechanism. Yet it was the real starting point for the modern discipline of machine learning and later led the way for deep learning and quantum machine learning.[41]

- **1949 – The Hebb Synapse**

  Canadian psychologist Donald O. Hebb published his book "The Organization of Behavior: A Neuropsychological Theory." Here, Hebb theorizes on neuron excitement and communication between neurons that influenced how psychologists view stimulus processing in the mind.
  The first use of the concept was in studying how brains learn. It also paved the way for the development of computational machines mimicking natural neurological processes, such as machine learning.

- **1950 – The Turing Test**

  Alan Turing, an English computer scientist, proposed the Turing Test as a measure of a computer's intelligence in 1950. It is a method of measuring artificial intelligence. If a person cannot tell whether they are speaking to another person or a computer, the computer is considered intelligent. The Turing

test has been criticized because it is difficult to create a fair and accurate test, as well as because it does not adequately measure intelligence. It is, however, an important milestone in the history of artificial intelligence research.

- **1952 – Machine Learning and the Game of Checkers**

Arthur Samuel, an English mathematician, created a computer learning program for playing championship-level computer checkers on the IBM 701. He started alpha-beta pruning, a design that calculates each side's chances of winning. This computer program determines its next move by employing a minimax algorithm, which determines the best possible move for a player in a game by minimizing the opponent's maximum gain and maximizing the player's minimum gain. Arthur Samuel is credited with coining and popularizing the term "Machine Learning."

- **1958 – The Perceptron**

Frank Rosenblatt, a psychologist, attempted to create "the first machine capable of producing an original idea," and later designed the Perceptron, the first neural network ever created. Donald Hebb's model of brain cell interaction was combined with Arthur Samuel's machine learning efforts. It was fed a series of punch cards and, after 50 attempts, learned to distinguish between cards with markings on the left and cards with markings on the right.

Despite its promise, the perceptron was unable to recognize a wide range of visual patterns, leaving researchers frustrated. It would be several years before the frustrations of investors and funding agencies subsided.

- **1963 – A Game of Tic Tac Toe**

Computer Scientist Donald Michel designed Machine Educable Noughts And Crosses Engine (MENACE), a large pile of matchboxes that contained several beads and used reinforcement learning to play tic-tac-toe.

MENACE works a little like a neural network. It is randomly optimized

initially, but after playing a few games, it adjusts to favor winning strategies in each situation.

- **1967 – The Nearest Neighbor Algorithm**

Thomas Cover and Peter Hart published his "Nearest Neighbor Pattern Classification" in 1967. It laid a foundation for recognizing patterns and regression in machine learning. The Nearest Neighbor algorithm is a method for very basic pattern recognition that was developed to allow computers to conduct rudimentary pattern detection. It works by comparing existing data and classifying it as the nearest neighbor, which means the most similar item in memory, which can help travel salesmen in a random city.

- **1979 – Neocognitron and The Stanford Cart**

Japanese computer scientist Kunihiko Fukushima publishes his work on Neocognitron, a hierarchical multilayered network used to detect patterns and inspire convolutional neural networks used for analyzing images. It sparked a revolution in what we now call AI.

In the same year, a group of researchers from Stanford University created a robot called the Cart. It was a decades-long endeavor that evolved in various forms from 1960 to 1980. Created initially as a remote-controlled television-equipped mobile robot, it became a radio-linked machine to a large mainframe computer that can independently navigate obstacles in a room.
The invention was state-of-the-art at the time, and machine learning shifted as a probable tool to create and eventually revitalize an autonomous vehicle.

- **1981 – Explanation Based-Learning**

Machine learning has come a long way since its inception in 1981. That year, Gerald Dejong introduced the concept of Explanation Based Learning (EBL),

in which a computer analyses training data and creates a general rule it can follow by discarding unimportant data. For example, if the software is instructed to concentrate on the queen in chess, it will discard all non-immediate-effect pieces. This laid the foundation for modern supervised learning techniques.

- **1989 – Boosting for Machine Learning**

The concept of boosting was first presented in a 1990 paper titled "The Strength of Weak Learnability" by Robert Schapire and Yoav Freund. It marked a necessary development for the evolution of machine learning. As Schapire states, "A set of weak learners can create a single strong learner." It simply translates to producing numerous weaker models and combines their predictions to convert them into single powerful model

- **1991 – The Vanishing Gradient Problem**

Although the start of the 1990s popularised methods such as support vector machines, there were still challenges found along the way. The vanishing gradient problem was first identified by Sepp Hochreiter. It was a challenge in machine learning development, specifically with deep neural networks.

As the number of layers in a network increases, the value of the derivative decreases until it eventually vanishes altogether. This can make the learning process extremely slow and difficult to manage.

- **1997 – Deep Blue and the Milestone of LSTM**

In 1997, IBM's Deep Blue became the first computer chess-playing system to defeat a reigning world chess champion when it beat Garry Kasparov.

It's also the year when Sepp Hochreiter and Jürgen Schmidhuber published a groundbreaking paper on "Long Short-Term Memory" (LSTM). It's a recurrent neural network architecture that will revolutionize deep learning in future decades.

- **2002 – The Release of Torch**

  In 2002, the open-source machine learning library Torch was released. This library allowed for more flexibility and customizability than other libraries at the time and quickly became popular among researchers.

- **2006 – Deep Belief Network**

  This year marks a remarkable time in the history of machine learning because Geoffrey Hinton created fast-learning algorithms to explain new algorithms that help computers distinguish objects and text in images and videos. Together with Ruslan Salakhutdinov, Osindero, and Teh, they published the paper "A fast learning algorithm for deep belief nets," in which they stacked multiple RBMs together in layers and called them Deep Belief Networks. The training process is much more efficient for large amounts of data.

- **2009 – ImageNet**

  Fei-Fei Li, a professor at Stanford, launched ImageNet, a database of 14 million labeled images in 2009. It would be a benchmark for deep learning researchers participating in ImageNet competitions (ILSVRC) every year. The Economist described the creation of this database as an exceptional event for popularizing AI throughout the tech community, marking a new era of deep learning history.

- **2010 – Microsoft's Kinect**

  A remarkable year for machine learning history is the release of Kinect, a motion-sensing input device for the Xbox 360 gaming console. It can track 20 different human features at the rate of 30 times per second.

- **2011 – IBM's Watson and Google Brain**

  Watson is a cognitive system powered by artificial intelligence and natural language processing developed by IBM. In 2011, Watson competed on the game show Jeopardy! against two human competitors and won. This made it the first computer system ever to win a quiz show against humans.

- **2012 – ImageNet Classification**

  In 2012, Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever published a research paper detailing a model that can reduce the error rate in image recognition systems by leaps and bounds. AlexNet, a GPU-based CNN model created by Alex Krizhevsky, won Imagenet's image classification contest with an accuracy of 84%. It significantly improved over the 75 percent success rate of prior models. This victory starts a deep learning revolution that will span the globe.

Machine learning is one of today's cutting-edge technologies that has aided us in improving not just industrial and professional procedures but also day-to-day life. This branch of machine learning uses statistical methods to create intelligent computer systems capable of learning from data sources accessible to it. Organizations use machine learning to gain insight into consumer trends and operational patterns, as well as the creation of new products. Many of today's top businesses incorporate machine learning into their daily operations. For many businesses, machine learning has become a significant competitive differentiator.

3.3.3 Tasks of Machine Learning

A Machine Learning task is a type of prediction or inference made based on the problem or question and the available data. The classification task, for example, assigns data to categories, whereas the clustering task groups data based on similarity. Rather than being explicitly programmed, machine learning tasks rely on patterns in data. After determining which task is appropriate for your scenario, you must select the best algorithm to train your model.[42]

- **Binary classification**

  A supervised Machine Learning task that is used to predict which of two classes (categories) an instance of data belongs to. The input of a classification algorithm is a set of labeled examples, where each label is an integer of either 0 or 1. The output of a binary classification algorithm is a classifier, which you can use to predict the class of new unlabeled instances.[45] Examples of binary classification scenarios include:

  - Understanding sentiment of Twitter comments as either "positive" or "negative".
  - Diagnosing whether a patient has a certain disease or not.
  - Making a decision to mark an email as "spam" or not.
  - Making a decision to mark a tweet as "cyberbullying" or not.

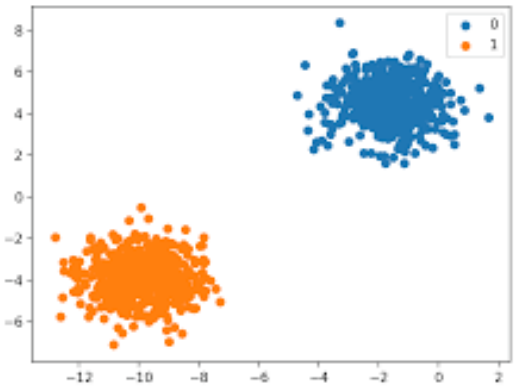


Figure 22 Machine Learning - Binary Classification

- **Multiclass Classification**

  A supervised Machine Learning task that is used to predict the class (category) of an instance of data. The input of a classification algorithm is a set of labeled examples. Each label normally starts as text. It is then run through the TermTransform, which converts it to the Key (numeric) type. The output of a classification algorithm is a classifier, which you can use to predict the class of new unlabeled instances.[42] Examples of multi-class classification scenarios include:

  - Categorizing flights as "early", "on time", or "late".
  - Understanding movie reviews as "positive", "neutral", or "negative".
  - Categorizing hotel reviews as "location", "price", "cleanliness", etc.

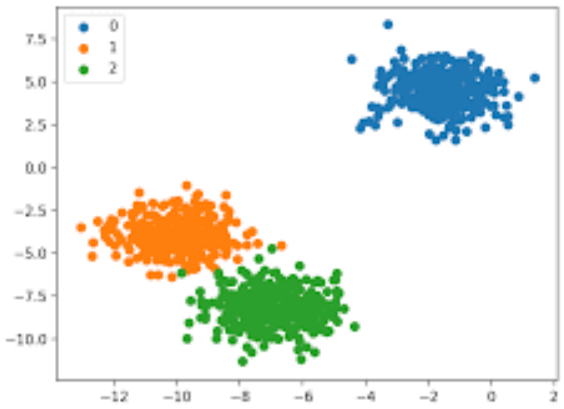


Figure 23 Machine Learning - Multiclass Classification

- **Regression**

  Regression tasks mainly deal with the estimation of numerical values (continuous variables). Regression task in Machine Learning is a supervised machine learning technique used to predict the values of a given target variable based on the input of one or more independent variables. It is a task of fitting a mathematical model to observed data points, where the objective is to minimize the sum of squared errors between the observed data and the predicted values. In regression tasks, we use linear and non-linear models to build our predictive models. Linear models have a basic assumption that there exists a linear

relationship between the input and output variables, while non-linear models do not rely on any such assumptions.[44]

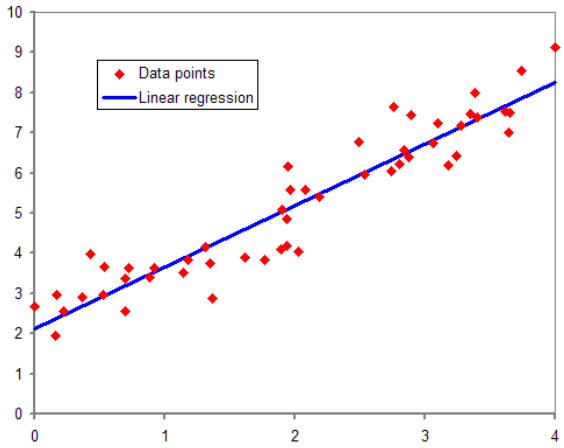


Figure 24 Machine Learning – Regression

- **Clustering**

  An unsupervised machine learning task used to group data instances into clusters with similar characteristics. Clustering can also be used to identify relationships in a dataset that browsing or simple observation would not logically reveal. The methodology used determines the inputs and outputs of a clustering algorithm. You have the option of using a distribution, centroid, connectivity, or density-based approach.[43]

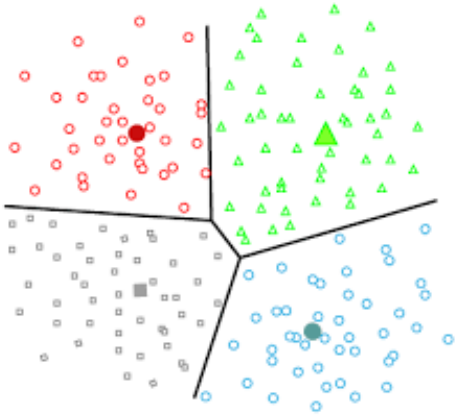


Figure 25 Machine Learning - Clustering

3.3.4 Use Cases of Machine Learning

In 2024, the Machine Learning market is expected to be worth $30.6 billion. The Internet of Things (IoT) and Artificially Intelligent (AI) solutions are increasingly driving the world. Machine Learning is critical in the design and development of these solutions. Machine learning is all around us. We live in an era dominated by machine learning applications, such as our smartphones' voice assistants, the Face Unlock feature, surge pricing on ride-hailing apps, email filtering, and many others. There could be several machine learning applications that you are using in your daily life without even realizing it.[46]

- **Image Recognition**

  One of the most notable machine learning applications is image recognition, which is a method for cataloging and detecting an object or feature in a digital image. In addition, this technique is used for further analysis, such as pattern recognition, face detection, and face recognition.[47]

- **Speech Recognition**

  Machine Learning software can measure spoken words using a set of numbers that represent the speech signal. Popular speech recognition applications include Amazon's Alexa, Apple's Siri, and Google Maps.[47]

- **Predict Traffic Patterns**

  When we enter our location on the map, the application collects massive amounts of data about current traffic in order to generate predictions about upcoming traffic and identify the quickest route to our destination, as Google Maps.[46]

- **E-commerce Product Recommendations**

  Product recommendation is a prominent feature of almost any e-commerce website, and it involves the sophisticated use of machine learning algorithms. Websites track customer behavior based on previous purchases, browsing habits, and cart history, and then recommend products based on machine learning and artificial intelligence.[46]

- **Self-Driving Cars**

  Self-driving cars use an unsupervised learning algorithm that heavily relies on machine learning techniques. This algorithm allows the vehicle to collect information about its surroundings from cameras and sensors, understand it, and decide what actions to take.[47]

- **Email Spam**

  One of the most well-known applications of machine learning is in the detection of email spam. Spam filters are built into email service providers' applications, and they use an ML algorithm to classify incoming emails as spam and route them to the spam folder.[46]

- **Catching Malware**

  There are two basic stages to using machine learning (ML) to detect malware. First, analyze suspicious activities in an Android environment to generate a suitable collection of features. Second, train the system to detect future cyberattacks in such environments using machine and deep learning (DL) techniques on the generated features.[47]

3.3.5 Machine Learning in Data Science

Many organizations and industries now emphasize the use of data to improve their products and services. If we only talk about data science, we are only talking about data analysis using Machine Learning Operations(MLOps) machine learning. Machine Learning and data science must work in tandem. Engineers must use Machine Learning and Data Science to make better and more appropriate decisions.[50]

Data Science is a field that studies data and how to extract meaning from it, whereas Machine Learning is a field devoted to understanding and building methods that utilize data to improve performance or inform predictions. Machine Learning is a branch of artificial intelligence.[48]

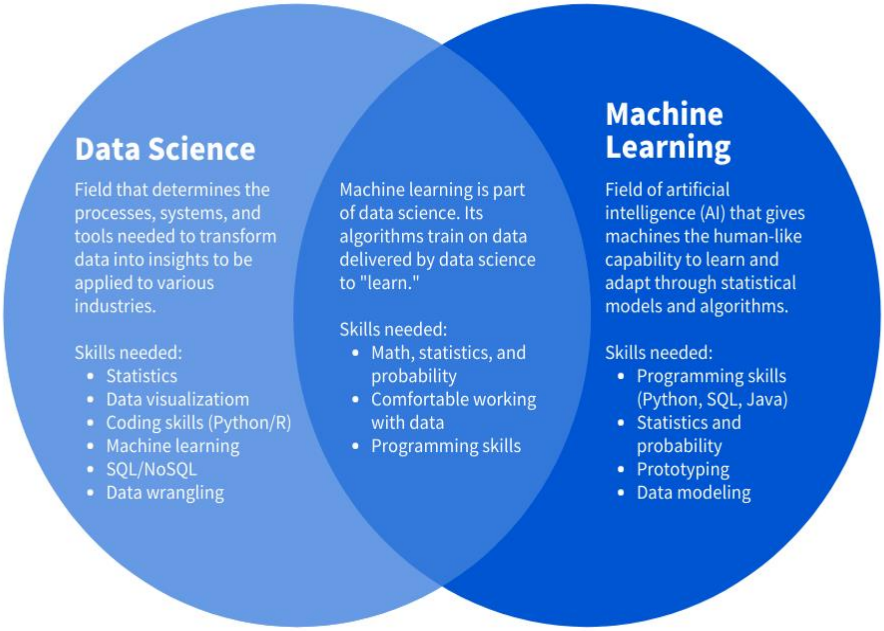


Figure 26 Data Science vs Machine Learning

Especially, Data Science is a field that processes and extracts data from semi-structured data and structured data. It needs an entire analytics universe and take cares of data processing. Moreover, it focuses on algorithms. On the other hand, Machine Learning is a field that offers systems the ability to learn without being programmed explicitly. It combine machine and Data Science. Finally, it focuses on algorithms statistics. [49]

The role of Machine Learning in Data Science takes place in 5 stages.

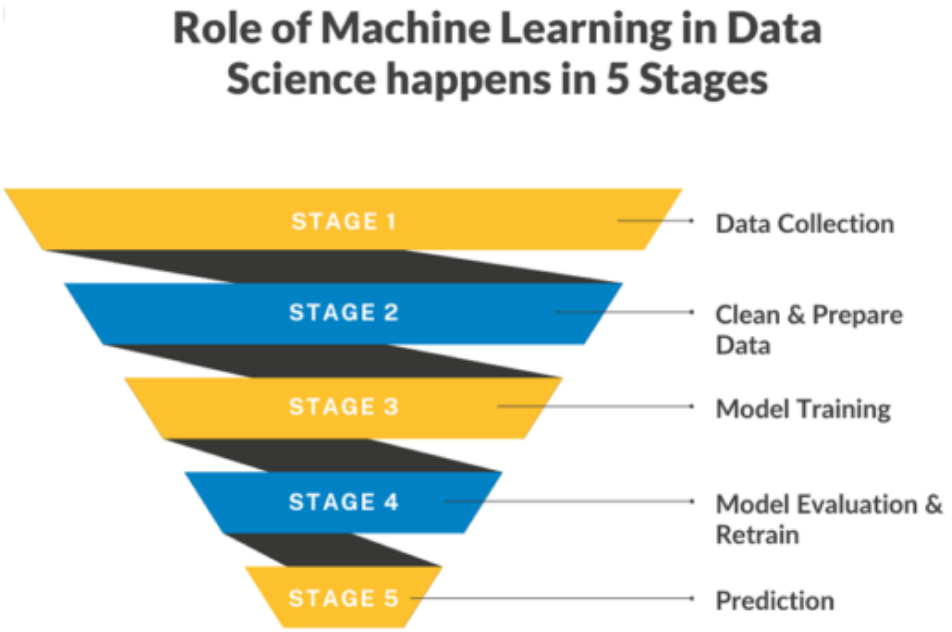


Figure 27 Machine Learning in Data Science

- **Data Collection**

  The first step in the machine learning process is data collection. As per the business problem, machine learning helps collect and analyze structured, unstructured, and semi-structured data from any database across systems. It could be a CSV file, a pdf file, a document, an image, or a handwritten form.

- **Data Preparation and Cleaning**

  Machine learning technology aids in data preparation by analyzing data and preparing features related to business problems. When clearly defined, ML systems understand the features and relationships between them. It should be noted that features are the foundation of machine learning and any data science

project. Once the data has been prepared, it must be cleansed because data in the real world is dirty and corrupted with inconsistencies, noise, incomplete information, and missing values. With the help of machine learning, we can find out the missing data and do data imputation, encode the categorical columns, remove the outliers, duplicate rows, and null values much faster in an automated fashion.

- **Model Training**

Model training depends on both the quality of the training data and the choice of the machine learning algorithm. An ML algorithm is selected based on end-user needs. Additionally, you need to consider the model algorithm complexity, performance, interpretability, computer resource requirements, and speed for better model accuracy. Once the right machine learning algorithm is selected, the training data set is divided into two parts for training and testing. This is done to determine the bias and variance of the ML model. As a result of model training, you will achieve a working model that can be further validated, tested, and deployed. Once your model has been trained, you can evaluate it using various metrics. Note that selecting a metric is completely dependent on the model type and implementation strategy. Although the model has been trained and evaluated, it is not yet ready to solve your business problems. By fine-tuning the parameters, any model can be improved for better accuracy.

- **Model Prediction**

Understanding prediction errors is critical whenever we discuss model prediction (bias and variance). Gaining a thorough understanding of these errors will assist you in developing accurate models and avoiding the errors of overfitting and underfitting the model. For a successful data science project, you can further reduce prediction errors by finding a good balance between bias and variance.

3.3.6 Machine Learning Algorithms

Our daily lives have been significantly impacted by machine learning. From smart assistants scheduling appointments, playing songs, and notifying users based on calendar events to NLP-based voice assistants, machine learning is everywhere. All of these intelligent systems use machine learning algorithms. Each machine learning algorithm in data science addresses a specific problem. In some cases, professionals will use a combination of these algorithms because one algorithm may be unable to solve a specific problem.[51]

- **Linear Regression**

  Linear regression describes the relationship between an input variable (x) and an output variable (y), also known as independent and dependent variables.
  A clear and simple algorithm approximates a linear relationship between one or more explanatory variables and a continuous numerical output variable. It is easier to train than other machine learning algorithms. Its most significant advantage is its ability to explain and interpret model predictions. It is a regression algorithm that is used to forecast outcomes such as customer lifecycle value, housing prices, and stock prices.[49]

- **Logistic Regression**

  Logistic Regression is a method for estimating discrete values (typically binary values such as 0/1) from a set of independent variables. It predicts the likelihood of an event by fitting data to a logit function. In logistic regression, the dependent variable is binary. This type of regression analysis describes and explains data by describing the relationship between one dichotomous variable and one or more independent variables. Logistic regression is used in predictive analysis where pertinent data predict an event probability to a logit function. As a result, it is also known as logit regression.[52]

- **Decision Tree**

  A decision tree allows you to visualize the map of potential outcomes for a series of decisions. It enables businesses to compare potential outcomes and then make an informed decision based on parameters such as advantages and probabilities that are advantageous to them. Decision tree algorithms have the potential to predict the best option based on a mathematical construct and can also be useful when brainstorming over a specific decision. The tree begins with a root node (decision node) and branches into sub-nodes that represent possible outcomes. Each outcome can generate child nodes, which can lead to new possibilities. The algorithm creates a tree-like structure that can be used to solve classification problems.[53]

- **Support Vector Machine (SVM)**

  Both classification and regression tasks are accomplished using Support Vector Machine algorithms. These are supervised machine learning algorithms that plot each piece of data in n-dimensional space, where n denotes the number of features. Each feature value is associated with a coordinate value, making plotting the features easier. Furthermore, classification is carried out by clearly determining the hyperplane that separates the two sets of support vectors or classes. A good separation ensures accurate classification of the plotted data points.[51]

- **Naïve Bayes**

  Naive Bayes is a probabilistic machine learning algorithm that is used to solve classification problems and is based on the Bayesian probability model. The algorithm's fundamental assumption is that the features under consideration are independent of one another, and that a change in the value of one does not affect the value of the other. A Naive Bayesian approach is simple to develop and apply. It is capable of handling massive datasets and is useful for making real-time predictions. Spam filtering, sentiment analysis and prediction, document classification, and other applications are among its uses.[52]

- **K-Nearest Neighbors (KNN)**

  The K Nearest Neighbors (KNN) algorithm is used for both classification and regression problems. It stores all the known use cases and classifies new use cases (or data points) by segregating them into different classes. This classification is accomplished based on the similarity score of the recent use cases to the available ones. KNN is a supervised machine learning algorithm, wherein 'K' refers to the number of neighboring points we consider while classifying and segregating the known n groups. The algorithm learns at each step and iteration, thereby eliminating the need for any specific learning phase. The classification is based on the neighbor's majority vote.[51]

- **K-Means**

  K-Means is a distance-based unsupervised machine learning algorithm used for clustering. In this algorithm, you classify datasets into clusters (K clusters) where the data points within one set remain homogenous, and the data points from two different clusters remain heterogeneous.[49]

- **Random Forest**

  It is arguably one of the most popular algorithms and builds upon the drawbacks of overfitting prominently seen in the decision tree models. Overfitting is when algorithms are trained on the training data a bit too well, and where they fail to generalize or provide accurate predictions on unseen data. Random forest solves the problem of overfitting by building multiple decision trees on randomly selected samples from the data. The final outcome in the form of the best prediction is derived from the majority voting of all the trees in the forest.[52]

3.3.7 Machine Learning Tools in Python

A Python framework is an interface or tool that allows developers to easily build ML models without looking deeper into the underlying algorithms. Python libraries are pre-written code files that can be imported into your code base using Python's import feature. This increases the reusability of your code. A Python framework can be defined as a collection of libraries designed to make it simple to build a model (e.g., machine learning) without having to understand the underlying algorithms. An ML developer, on the other hand, must understand how the algorithms work in order to know what results to expect and how to validate them.

- **Matplotlib**

  Matplotlib is a two-dimensional plotting library that is interactive and cross-platform. It can generate high-quality graphs, charts, and plots in a variety of hardcopy formats.[56]

Figure 28 Machine Learning – Matplotlib

- **NumPy**

  NumPy extends Python with multidimensional array and matrix processing, as well as a large collection of high-level mathematical functions. It is widely used in scientific computing, making it one of the most popular Python Packages for machine learning. [57]

Figure 29 Machine Learning – NumPy

- **SciPy**

  With machine learning growing at supersonic speed, many Python developers were creating python libraries for machine learning, especially for scientific and analytical computing. Travis Oliphant, Eric Jones, and Pearu Peterson in 2001 decided to merge most of these bits and pieces codes and standardize it. The resulting library was then named as SciPy library. SciPy is a very popular ML library with different modules for optimization, linear algebra, integration and statistics.[54]

Figure 30 Machine Learning - SciPy

- **Seaborn**

Seaborn is a library for making statistical graphs in Python. It is built on top of Matplotlib and also integrated with Pandas' data structures. Seaborn gives more attractive graphs and built-in plots than Matplotlib's. [55]

Figure 31 Machine Learning – Seaborn

- **Pandas**

Pandas is quickly becoming the most popular Python data analysis library, with support for fast, flexible, and expressive data structures designed to work on both "relational" and "labeled" data. Pandas today is an inevitable library for solving practical, real-world data analysis in Python. Pandas is extremely stable and provides highly optimized performance. The backend code is written entirely in C or Python.[56]

Figure 32 Machine Learning – Pandas

- **Keras**

  Keras is a popular Python Machine Learning library. It is a high-level neural network API that can be used with TensorFlow or Theano. It can run on both the CPU and the GPU at the same time. Keras makes it possible for ML novices to build and design a Neural Network. One of the best features of Keras is that it allows for quick and easy prototyping.[57]

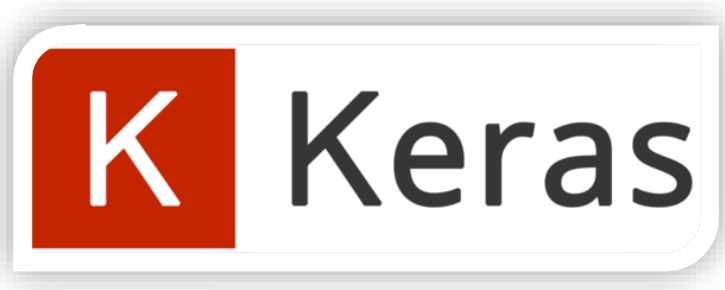


Figure 33 Machine Learning – Keras

- **PyTorch**

  PyTorch is a popular open-source Machine Learning library for Python that is based on Torch, an open-source Machine Learning library written in C with a Lua wrapper. It includes a wide range of tools and libraries that support Computer Vision, Natural Language Processing (NLP), and a variety of other ML programs. It enables developers to perform Tensor computations with GPU acceleration and also aids in the creation of computational graphs.[55]

Figure 34 Machine Learning - PyTorch

- **TensorFlow**

TensorFlow is a state-of-the-art Python framework for machine learning which carries out deep ML algorithms. Google Brain Team created it as a second-generation, open source-based system. TensorFlow is unique in that it can generate ML models for both computers and smartphones. It enables data to be distributed seamlessly across multiple GPU and CPU cores. TensorFlow is compatible with a variety of programming languages, including C++, Python, and Java. It uses tensors, which are n-dimensional data storages with liner operations. [54]

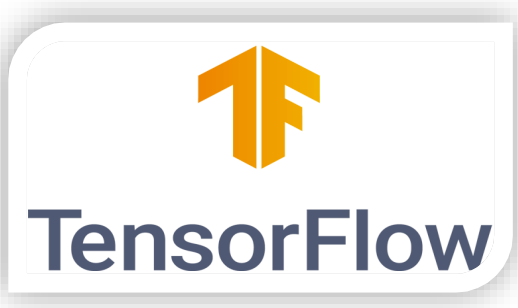


Figure 35 Machine Learning – TensorFlow

4. Machine Learning Model

A machine learning model is a program that can find patterns or make decisions from a previously unseen dataset. For example, in natural language processing, machine learning models can parse and correctly recognize the intent behind previously unheard sentences or combinations of words. In image recognition, a machine learning model can be taught to recognize objects. A machine learning model can perform such tasks by having it 'trained' with a large dataset. During training, the machine learning algorithm is optimized to find certain patterns or outputs from the dataset, depending on the task. [59] Once the model has been trained, you can use it to analyze previously unexplored data and make predictions about it. The output of this process - often a computer program with specific rules and data structures - is called a machine learning model. [58] The Machine Learning model development requires some basic steps that cannot skipped. [60]

- Problem Definition
- Data Collection
- Data Analysis & Preparation
- Model Training
- Model Evaluation
- Model Deployment
- Model Retrain

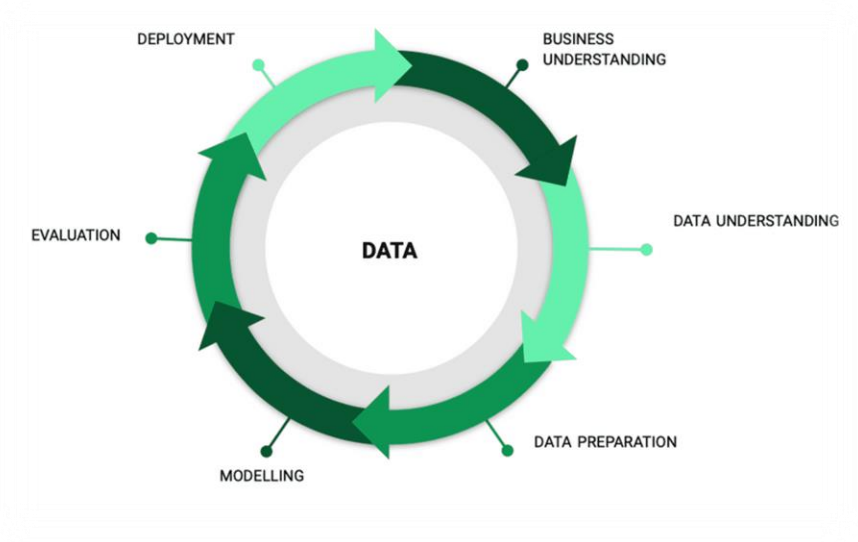


Figure 36 Machine Learning Model Development

4.1 Approach

Figure 37 Machine Learning Model – CBD.ML

In this chapter, it will be presented our Machine Learning Model. Cyberbullying detection is a binary classification problem, in which datasets consist of two columns. In particular, first column has the text or tweet, and the second has the annotation of cyberbullying or not. Generally, our approach is to find the suitable datasets for cyberbullying detection that after preprocessing of data and using the appropriate Machine Learning algorithms will give us high accuracy and precision.

Firstly, we will present and review our selected datasets. Moreover, we will provide information for dataset length, columns, rows and dataset preparation

Secondly, we will provide the technologies that will be used in model development, both programming language as well as libraries, techniques and tools.

Thirdly, we will introduce the Machine Learning algorithms which we select to examine more specifically. Furthermore, we will provide information both mathematical and theoretical and also the suitable hyper-parameters for better results.

Fourth, we will present all the process for the Machine Learning Model development along side with code and explanation.

Finally, we provide the results for all metrics and a comparison of all Machine Learning techniques and algorithms, in order to select the suitable Machine Learning model for cyberbullying detection problem.

4.2 Datasets

Our research for finding a suitable dataset for cyberbullying detection was a hard process due to the fact that there is not enough both documentation as well as open-source datasets in this research field. Besides, we manage to find datasets, which match perfect our binary classification problem for cyberbullying detection.

Dataset selection is the most important stage of creating a Machine Learning Model. More specifically, data have impact not only on Machine Learning Algorithms but also in Machine Learning model evaluation. Therefore, it is best practise to find accurate and large-scale dataset in binary classification problems for any prediction.

In this subchapter, we will present the datasets, which will use in our Machine Learning Model.

- Cyber Bullying Types Dataset [61]
- Cyber Troll Dataset [62]
- Classified Tweets Dataset [63]
- Cyberbullying Classification Dataset [64]

4.2.1 Cyber Bullying Types Dataset

Cyber Bullying Types Datasets was developed from Dr. N. Anathi[61] and contains 2140 entries. It is consisted of 2 columns and 5 cyberbullying categories such as Sexual Harassment, Doxing, Cyberstalking, Revenge Porn, and Slut Shaming. First columns is text and second column is cyberbullying category. It was downloaded from IEEE Dataport and it is a balanced dataset.

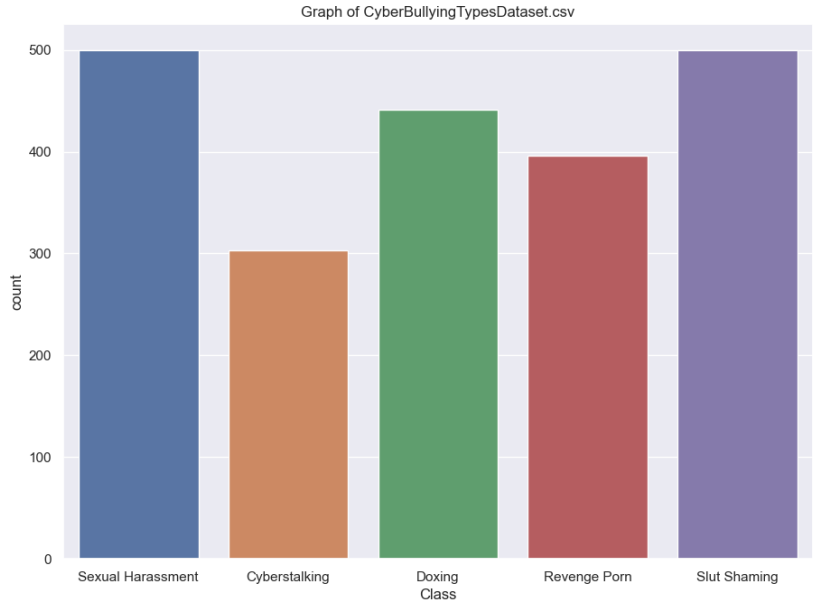


Table 1 Cyber Bullying Types Dataset

4.2.2 Cyber Troll Dataset

Cyber Troll datasets was developed from Saima Sadiq[62] and contains 20001 entries. It is consisted of 2 columns; first column is content, which is text, and second column is annotation, which is 1 or 0 for cyberbullying or not. It was downloaded from Zenodo and it is an imbalanced dataset.

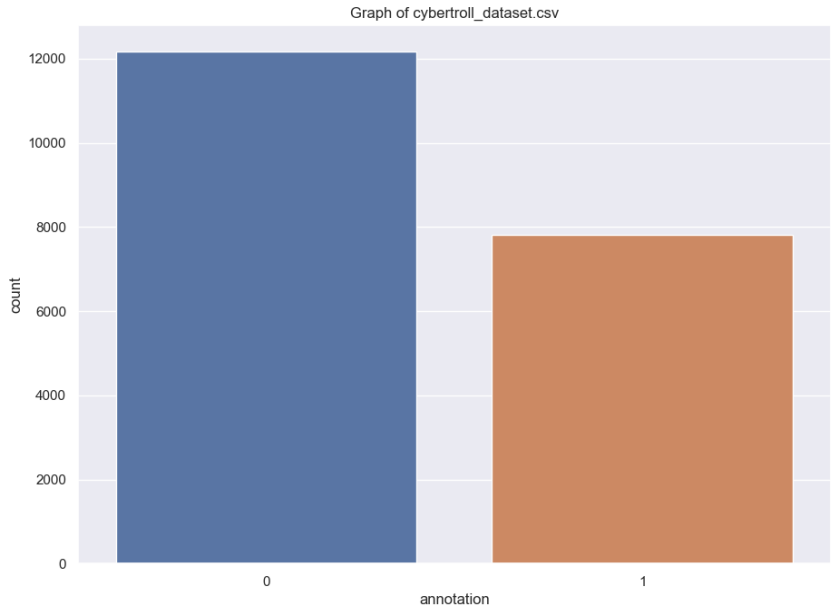


Table 2 Cyber Troll Dataset

4.2.3 Classified Tweets Dataset

Cyber Troll datasets was developed from Munki Albright[63] and contains 19934 entries. It is consisted of 5 columns.

- **text** = Tweets
- **suspicious** = 1 if suspicious and 0 if otherwise
- **cyberbullying** = 1 for racism, 2 for sexism and 0 if neither
- **hate** = 1 for hate text and 0 if otherwise
- **suicidal** = 1 for text with suicidal intent and 0 if otherwise

It was downloaded from Kaggle and it is an imbalanced dataset.

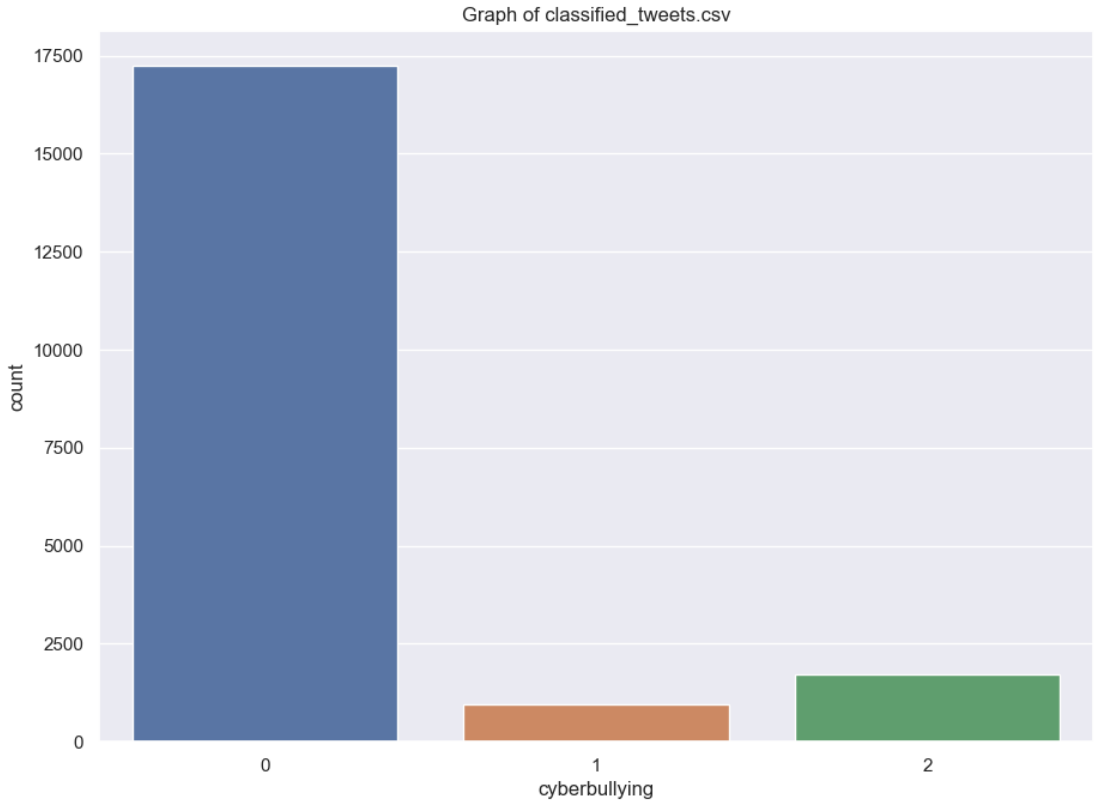


Table 3 Classified tweets

4.2.4 Cyberbullying Classification Dataset

Cyberbullying Classification was developed from J. Wang, K. Fu, C.T. Lu[64] [65] and contains 47692 entries. It is consisted of 2 columns and contains 6 cyberbullying categories.

- **tweet_text** = Tweets
- **cyberbullying_type** = categories of cyberbullying
    - **not_cyberbullying**
    - **gender**
    - **religion**
    - **other_cyberbullying**
    - **age**
    - **ethnicity**

It was downloaded from Kaggle and it is a balanced dataset.

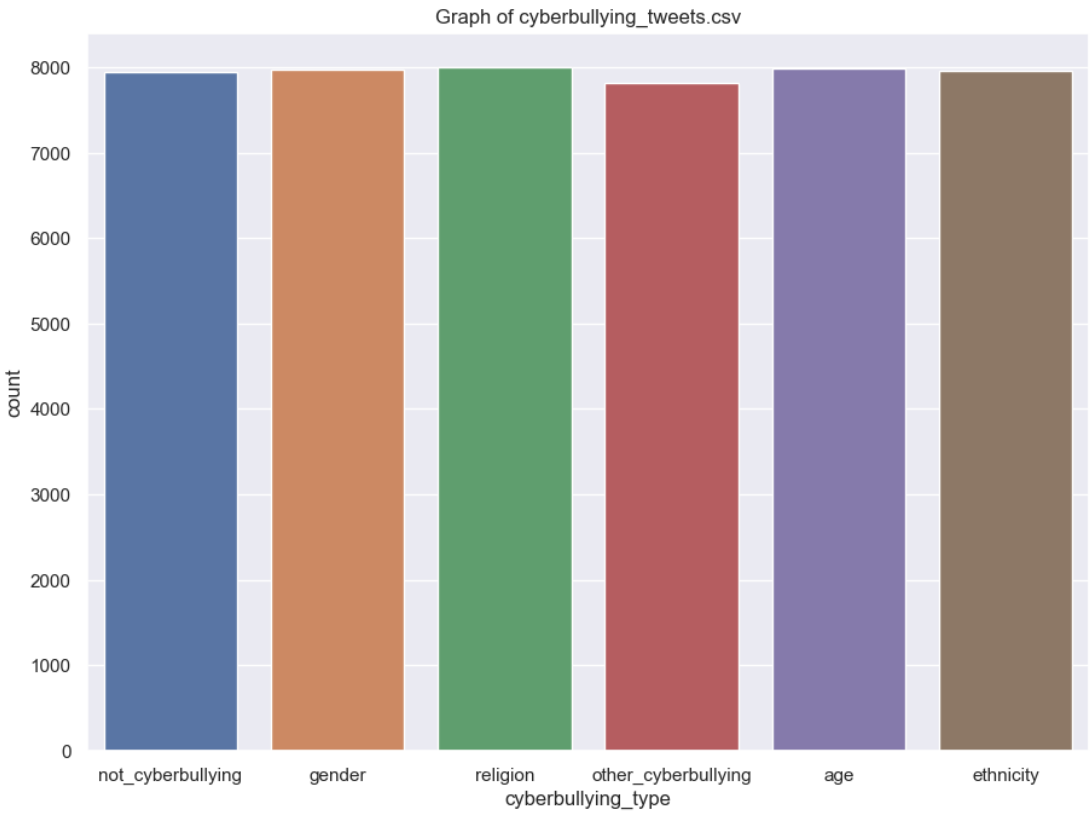


Table 4 Cyberbullying Classification

4.3 Technologies and Tools

We used many technologies, libraries, and tools, which help us to develop our Machine Learning Model for cyberbullying detection. We split our Machine Learning Model into 4 aspects: business overview , data gathering, model development and version control aspect.

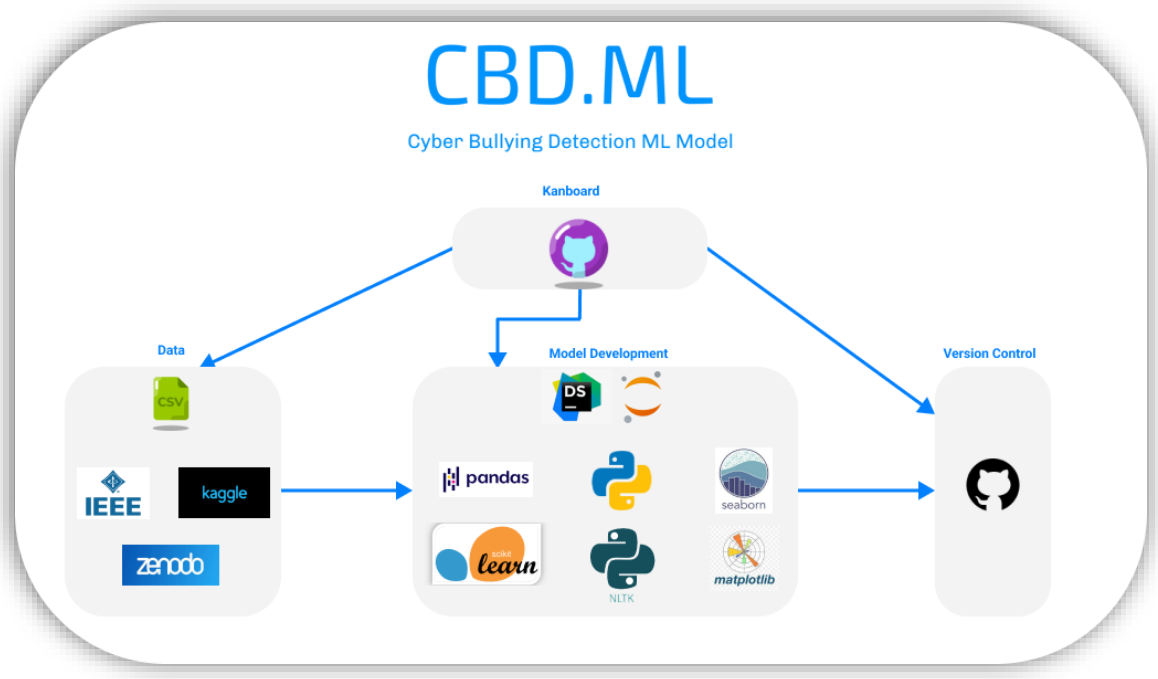


Figure 38 Machine Learning model - CBD.ML Technologies & Tools

- **Python**

  Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0 . Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2. Finally, Python consistently ranks as one of the most popular programming languages.[66]

Python can serve as a scripting language for web applications, e.g., via mod_wsgi for the Apache webserver.[191] With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, Streamlit, web2py, Tornado, Flask, Bottle, and Zope support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications.

Libraries such as NumPy, SciPy, and Matplotlib allow the effective use of Python in scientific computing, with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a computer algebra system with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. OpenCV has Python bindings with a rich set of features for computer vision and image processing.

Python is commonly used in artificial intelligence projects and machine learning projects with the help of libraries like TensorFlow, Keras, Pytorch, and scikit-learn. As a scripting language with a modular architecture, simple syntax, and rich text processing tools, Python is often used for natural language processing.

Figure 39 Python

- **Pandas**

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. Its name “Pandas” has a reference to both “Panel Data”, and “Python Data Analysis” and was created by Wes McKinney in 2008. Pandas allows to analyze big data and make conclusions based on statistical theories[67]

Pandas is mainly used for data analysis and associated manipulation of tabular data in DataFrames. Pandas allows importing data from various file formats such as comma-separated values, JSON, Parquet, SQL database tables or queries, and Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features. The development of pandas introduced into Python many comparable features of working with DataFrames that were established in the R programming language. The Pandas library is built upon another library NumPy, which is oriented to efficiently working with arrays instead of the features of working on DataFrames.

Developer Wes McKinney started working on pandas in 2008 while at AQR Capital Management out of the need for a high performance, flexible tool to perform quantitative analysis on financial data. Before leaving AQR he was able to convince management to allow him to open source the library.[69]
Another AQR employee, Chang She, joined the effort in 2012 as the second major contributor to the library. In 2015, pandas signed on as a fiscally sponsored project of NumFOCUS, a nonprofit charity in the United States.

- 2008: Development of pandas started
- 2009: pandas becomes open source
- 2012: First edition of Python for Data Analysis is published
- 2015: pandas becomes a NumFOCUS sponsored project
- 2018: First in-person core developer sprint

- **Sci-kit Learn**

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.[68]

The scikit-learn project started as scikits.learn, a Google Summer of Code project by French data scientist David Cournapeau. The name of the project stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers. In 2010, contributors Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort and Vincent Michel, from the French Institute for Research in Computer Science and Automation in Saclay, France, took leadership of the project and released the first public version of the library on February 1, 2010. In November 2012, scikit-learn as well as scikit-image, were described as two of the "well-maintained and popular" scikits libraries. In 2019, it was noted that scikit-learn is one of the most popular machine learning libraries on GitHub.

Scikit-learn is largely written in Python and uses NumPy extensively for high-performance linear algebra and array operations. Furthermore, some core algorithms are written in Cython to improve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR. In such cases, extending these methods with Python may not be possible.

It integrates well with many other Python libraries, such as Matplotlib and Plotly for plotting, NumPy for array vectorization, Pandas DataFrames, SciPy, and many more.

- **Matplotlib**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.[70]

Matplotlib was originally written by John D. Hunter. Since then, it has had an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and was further joined by Thomas Caswell. Matplotlib is a NumFOCUS fiscally sponsored project.

Matplotlib 2.0.x supports Python versions 2.7 through 3.10. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged not to support Python 2 past 2020 by signing the Python 3 Statement.

- **Seaborn**

Seaborn is one of an amazing library for visualization of the graphical statistical plotting in Python. It provides many color palettes and defaults beautiful styles to make the creation of many statistical plots in Python more attractive.

Seaborn library aims to make a more attractive visualization of the central part of understanding and exploring data. It is built on the core of the matplotlib library and also provides dataset-oriented APIs. Finally, Seaborn integrate with the Panda's data structures, and with this, we can easily jump between the various different visual representations for a given variable to better understand the provided dataset. [71]

- **NLTK**

  The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems.[72]

- **DataSpell**

  JetBrains DataSpell is a new IDE from JetBrains specifically for data scientists. It combines the interactivity of Jupyter Notebooks with the intelligent coding assistance of PyCharm, all within one IDE. In his talk, Andrey, PyCharm Product Manager, will explain how this new IDE is different from PyCharm and Jupyter Notebooks, and what it brings for those who are doing exploratory data analysis and prototyping ML models. [73]

Figure 40 JetBrains DataSpell

- **Jupyter Notebook**

  The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. This article will walk you

through how to use Jupyter Notebooks for data science projects and how to set it up on your local machine. [74]

A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it's a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.

Figure 41 Jupyter Notebook

- **GitHub**

GitHub is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project. Headquartered in California, it has been a subsidiary of Microsoft since 2018. It is commonly used to host open-source software development projects. As of January 2023, GitHub reported having over 100 million developers and more than 372 million repositories, including at least 28 million public repositories. Finally, it is the largest source code host.[75]

Figure 42 GitHub

4.4 Machine Learning Algorithms

Binary classification problems can be solved by a variety of machine learning algorithms ranging from Naive Bayes to deep learning networks. In our Machine Learning Model Development for cyberbullying detection, we use 8 algorithms for binary classification, which will be presented below.

4.4.1 Logistic Regression (LR)

Logistic Regression is a fundamental and widely used classification algorithm. It is named ‘Logistic Regression’ because its underlying technique is quite the same as Linear Regression. The term "Logistic" is obtained from the Logit function, which is used in this classification method. [76]

Logistic Regression to deal with the outlier uses the Sigmoid function. An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is a Sigmoid function, which takes any real value between zero and one. It is defined as:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

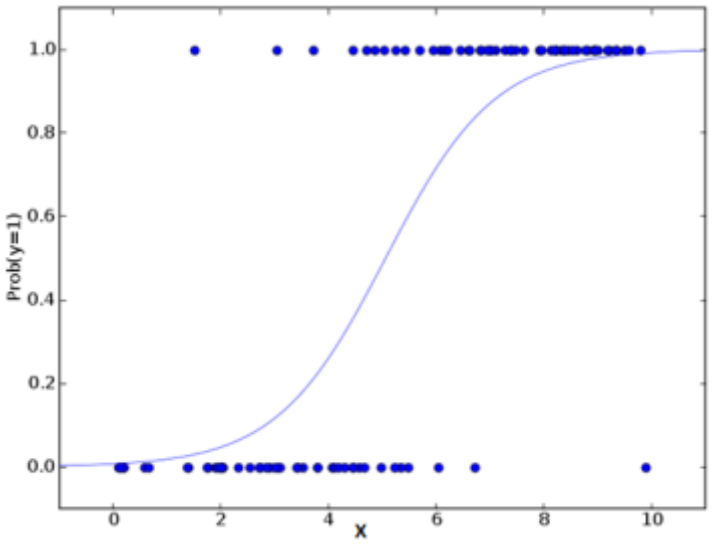


Table 5 Logistic Regression Plot

Let’s consider t as a linear function in a univariate regression model.

$$t = \beta_0 + \beta_1 x$$

So, the Logistic Equation will become

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Now, when the logistic regression model comes across an outlier, it will take care of it.

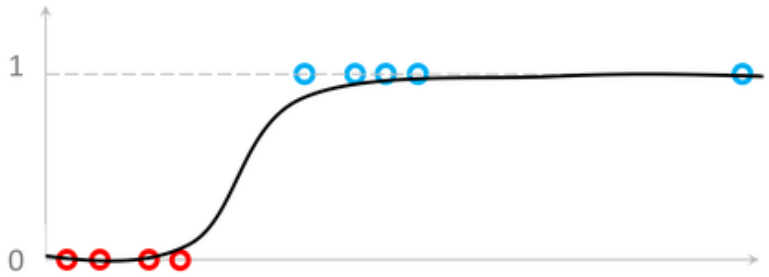


Table 6 Logistic Regression Outlier

4.4.2 Decision Tree (DT)

A decision tree is a flowchart-like tree structure where an internal node represents a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.

The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in a recursive manner called recursive partitioning. This flowchart-like structure helps you in decision-making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.[77]

A decision tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as with a neural network. Its training time is faster compared to the neural network algorithm.

The time complexity of decision trees is a function of the number of records and attributes in the given data. The decision tree is a distribution-free or non-parametric method which does not depend upon probability distribution assumptions. Decision trees can handle high-dimensional data with good accuracy.

The basic idea behind any decision tree algorithm is as follows:

- Select the best attribute using Attribute Selection Measures (ASM) to split the records.
- Make that attribute a decision node and breaks the dataset into smaller subsets.
- Start tree building by repeating this process recursively for each child until one of the conditions will match:
    - All the tuples belong to the same attribute value.
    - There are no more remaining attributes.
    - There are no more instances.

Attribute selection measure is a heuristic for selecting the splitting criterion that partitions data in the best possible manner. It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node. ASM provides a rank to each feature (or attribute) by explaining the given dataset. The best score attribute will be selected as a splitting attribute (Source). In the case of a continuous-valued attribute, split points for branches also need to define. The most popular selection measures are Information Gain, Gain Ratio, and Gini Index.

- **Information Gain**

  Info(D) is the average amount of information needed to identify the class label of a tuple in *D*.

$$\text{Info(D)} = -\sum_{i=1}^{m} pi \log_2 pi$$

Where, *Pi* is the probability that an arbitrary tuple in *D* belongs to class *Ci*.

InfoA(D) is the expected information required to classify a tuple from *D* based on the partitioning by *A*.

$$\text{Info}_A(D)=\sum_{j=1}^{V}\frac{|Dj|}{|D|}\ \text{X Info}(D_j)$$

Where, $|Dj|/|D|$ acts as the weight of the *jth* partition.

$$\text{Gain}(A)=\text{Info}(D)-\text{Info}_A(D)$$

The attribute *A* with the highest information gain, *Gain(A),* is chosen as the splitting attribute at node *N().*

- Gain Ratio

  Gain ratio handles the issue of bias by normalizing the information gain using Split Info.

$$SplitInfo_A(D)=-\sum_{j=1}^{v}\frac{|D_j|}{|D|}\times\log_2\left(\frac{|D_j|}{|D|}\right)$$

Where, $|Dj|/|D|$ acts as the weight of the jth partition and *v* is the number of discrete values in attribute *A*.

$$GainRatio(A)=\frac{Gain(A)}{SplitInfo_A(D)}$$

The attribute with the highest gain ratio is chosen as the splitting attribute (Source).

- Gini Index

Another decision tree algorithm CART (Classification and Regression Tree) uses the Gini method to create split points.

$$\text{Gini}(D)=1-\sum_{i=1}^{m} P_i^2$$

Where, *pi* is the probability that a tuple in *D* belongs to class *Ci*.

The Gini Index considers a binary split for each attribute. You can compute a weighted sum of the impurity of each partition. If a binary split on attribute A partitions data D into D1 and D2, the Gini index of D is:

$$\text{Gini}_A(D)=\frac{|D1|}{|D|}\text{Gini}(D_1)+\frac{|D2|}{|D|}\text{Gini}(D_2)$$

In the case of a discrete-valued attribute, the subset that gives the minimum gini index for that chosen is selected as a splitting attribute. In the case of continuous-valued attributes, the strategy is to select each pair of adjacent values as a possible split point, and a point with a smaller gini index is chosen as the splitting point.

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

The attribute with the minimum Gini index is chosen as the splitting attribute.

4.4.3 Random Forest (RF)

The Random Forest Algorithm is another frequently used ensemble learning classifier which uses multiple decision trees. The Random Forest classifier is basically a modified bagging algorithm of a Decision Tree that selects the subsets differently.[78] This model use two(2) key concepts that gives it the name random:

- **Random sampling of training data points when building trees**

When training, each tree in a random forest learns from a random sample of the data points. The samples are drawn with replacement, known as bootstrapping, which means that some samples will be used multiple times in a single tree. The idea is that by training each tree on different samples, although each tree might have high variance with respect to a particular set of the training data, overall, the entire forest will have lower variance but not at the cost of increasing the bias.

At test time, predictions are made by averaging the predictions of each decision tree. This procedure of training each individual learner on different bootstrapped subsets of the data and then averaging the predictions is known as bagging, short for bootstrap aggregating.

- **Random subsets of features considered when splitting nodes**

The other main concept in the random forest is that only a subset of all the features are considered for splitting each node in each decision tree. Generally, this is set to sqrt(n_features) for classification meaning that if there are 16 features, at each node in each tree, only 4 random features will be considered for splitting the node.

4.4.4 XGBoost

XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for “Extreme Gradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many Machine Learning tasks such as classification and regression.[79]

One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.

XGBoost can be used in a variety of applications, including Kaggle competitions, recommendation systems, and click-through rate prediction, among others. It is also highly customizable and allows for fine-tuning of various model parameters to optimize performance.

XGBoost stands for Extreme Gradient Boosting, which was proposed by the researchers at the University of Washington. It is a library written in C++ which optimizes the training for Gradient Boosting.

Finally, in this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

4.4.5 Multinomial Naïve Bayes

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. It is mainly used in text classification that includes a high-dimensional training dataset.[80]

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes. It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. It is called Bayes because it depends on the principle of Bayes' Theorem.

- **Bayes' Theorem**

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula for Bayes' theorem is given as:

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

Where, P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B. P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true. P(A) is Prior Probability: Probability of hypothesis before observing the evidence. P(B) is Marginal Probability: Probability of Evidence.

The Naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that's most probable; this is known as the maximum a posteriori or MAP decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label for some k as follows:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

MultinomialNB implements the naive Bayes algorithm for multinomially distributed data and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although TF-IDF vectors are also known to work well in practice). Finally, it is used for discrete counts.

4.4.6 Support Vector Machine (SVM)

The Support Vector Machine is a simple algorithm for classification and regression tasks. It can provide high accuracy with less computation power very fast. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.[81]

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input

features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions look as below.

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

When there is no misclassification, i.e our model correctly predicts the class of our data point, we only have to update the gradient from the regularization parameter.

When there is a misclassification, i.e our model makes a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform gradient update.

4.4.7 Bagging Decision Tree

Bagging, an acronym for bootstrap aggregation, creates and replaces samples from the data-set. In other words, each selected instance can be repeated several times in the same sample. We seem to increase our training data with bootstraps, which are each created and then used to create a classifier model. The final prediction is the average of all predictive models. [82]

The most popular bagging algorithm commonly used by data scientist is the random forest based on the decision tree algorithm. Another useful algorithm is the pocket filling of the neighboring subspace closest to K (KNN), where basic students are based on the closest neighbor algorithm to k. We will discuss these algorithms in detail in the future. We can understand the crisis with the following example.

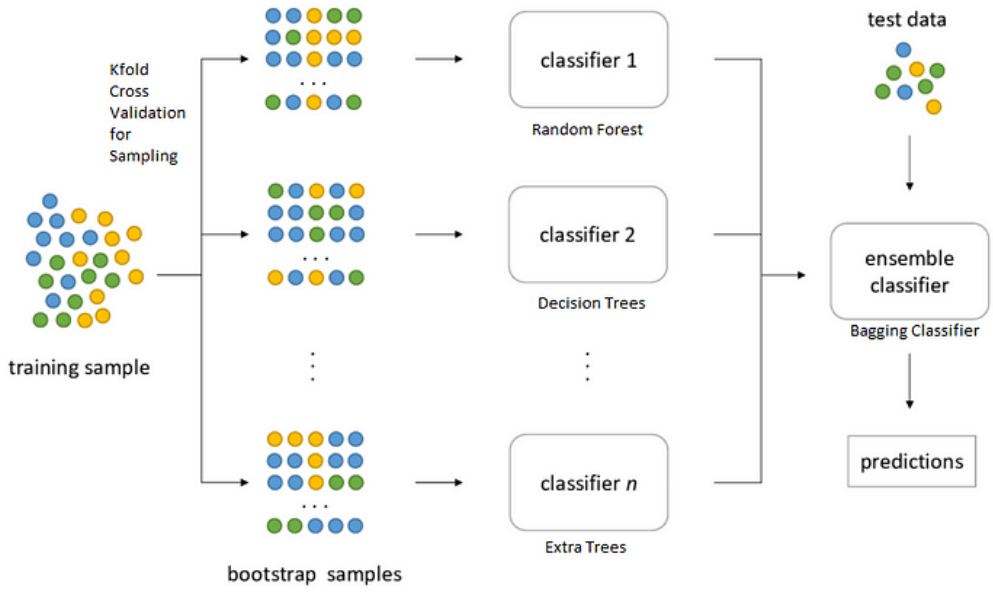


Figure 43 Bagging Classifier Process Flow

4.4.8 Boosting Decision Tree

Boosting, is a provide the strength of weak learner and help machine learning model to prevent under-fitting and over-fitting. It is a process that multiple weak learners(machine learning models) train and combine their output to create strong learner from it. It is used to prevent under-when single machine learning model is not working well and also used to prevent over-fitting when machine learning model is not working well on the validation dataset. [83]

Boosting is supervised learning algorithms which is a kind of frameworks that comprise many weak learners in cascade. It is training model continue sly on training dataset and updates their weights based on the variations between predicted and actual results. The weight updates based on algorithm we used for updates. In the case of boost by majority algorithms, mis-classified learner gets a weight gain and a learner with true classification loses weight. So, the classifier gives less attention to the true classification, which helps in faster convergence of the classifier.

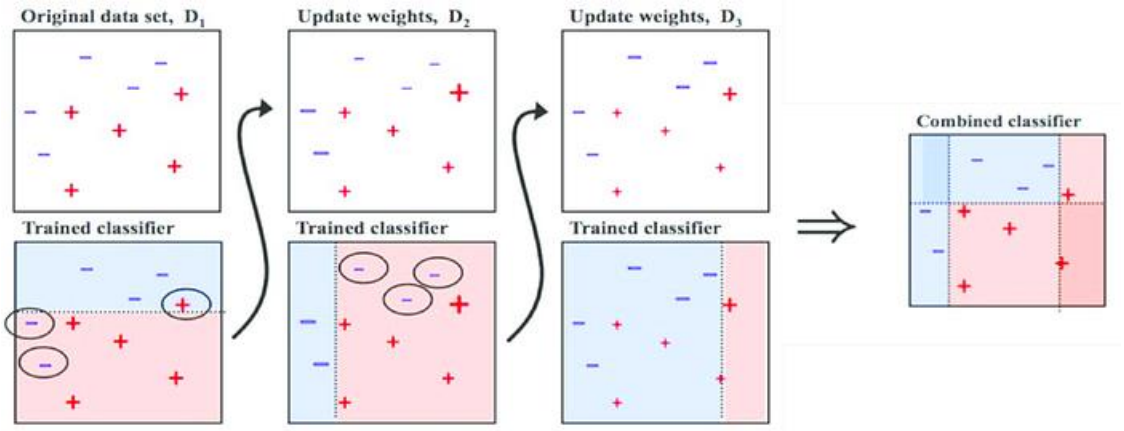


Figure 44 Boosting Classifier Process Flow

4.5 Machine Learning Model Development

In this subchapter, we will present the whole development process of our Machine Learning Model for cyberbullying detection. After gathering our datasets, we split the process into (8) phases, as they provide below.

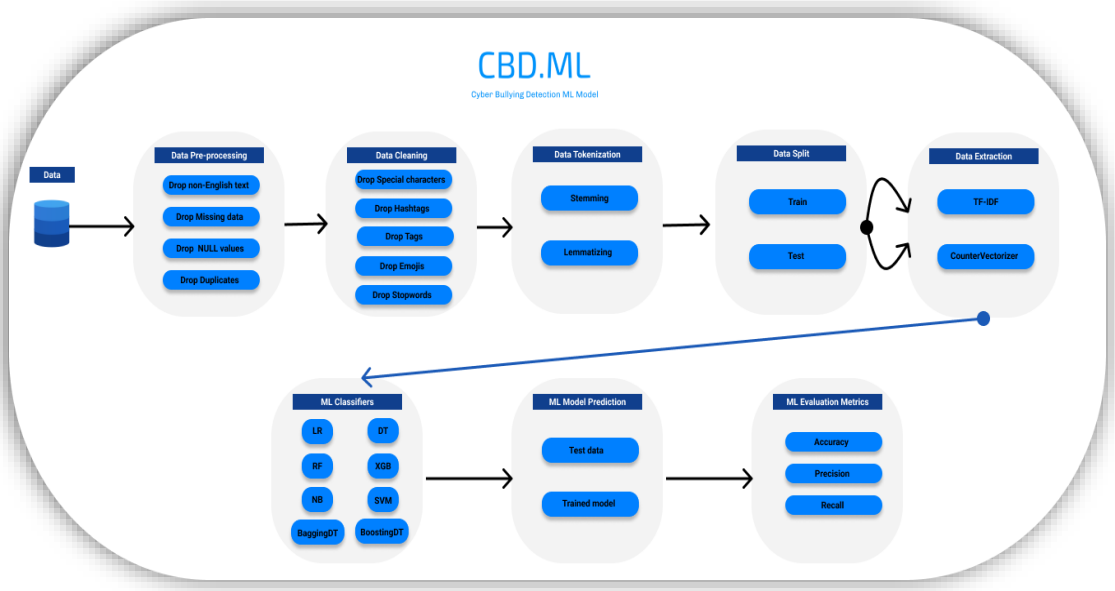


Figure 45 Machine Learning Model Development - CBD.ML

- **Data Preprocessing**

  The Data Preprocessing step is essential in cyberbullying detection. It consists of the dataset's initial cleaning. In our model, it has been applied to remove and clean unwanted noise in text, such as drop non-English characters, missing data, null values and duplicate text.

- **Data Cleaning**

The second step is Data Cleaning, which is the cleaning of the text. It is one of the most important procedure in cyberbullying detection for accurate results. It is consisted of special characters, hashtags, tags , URLs, emoji's and stopwords removal. As stopwords, they assumed the most common words in English, that do not add much information to the text. [89]

- **Data Tokenization**

Then, it is the time for data tokenization, such stemming, lemmatizing or both. Stemming is the process of reducing words to their stem or root. From the other hand, lemmatizing is the process of grouping together the inflected forms of a word, so it can be analyzed as a single term, identified by word's lemma. The difference is that stemming is faster, but lemmatizing is more accurate. [90]

- **Data Split**

Data split is an important aspect of data science, particularly for creating models based on data. This technique helps ensure the creation of data models and processes that use data models, such as machine learning, are accurate. The training data set is used to train and develop models. Training sets are commonly used to estimate different parameters or to compare different model performance.[91] The testing data set is used after the training is done. The training and test data are compared to check that the final model works correctly. In our model, the train data are 80% and test data is 20%.

- **Feature Extraction**

Feature Extraction is a crucial step for text classification in cyberbullying. In our model, we have used TF-IDF and CountVectorizer techniques for feature extraction.

TF-IDF is a combination of TF and IDF (term frequency-inverse document frequency), and this algorithm is based on word statistics for text feature extraction. Generally, TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. Therefore, TF-IDF is one of the most commonly used feature extraction techniques in text detection. [92]

$$w_{i,j} = \mathrm{tf}_{i,j} \times \log\left(\frac{N}{\mathrm{df}_i}\right)$$

$\mathrm{tf}_{i,j}$ = number of times i occurs in j divided by total number of terms in j
df_i = number of documents containing i
N = total number of documents

CountVectorizer is a great tool provided by the scikit-learn library in Python, that generally convert text to numerical data. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. CountVectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample.[93]

- **Machine Learning Classifiers**

In this study, various classifiers have been used to classify whether the text is cyberbullying or non-cyberbullying. The classifier models constructed are Logistic Regression(LR), Decision Tree(DT), Random Forest(RF), XGBoost(XGB), Naïve Bayes(NB), Support Vector Machine(SVM), Bagging Decision Tree(BaggingDT), Boosting Decision Tree(BoostingDT).

- **Machine Learning Model Prediction**

After Machine Learning Classifiers, next step is Machine Learning model prediction. This is the most crucial step of our model, where we test how accurate is our model for cyberbullying detection.

- **Machine Learning Evaluation Metrics**

The potential of any model can be evaluated using few metrics which helps in determining the ability of a model to differentiate texts as cyberbullying or not. To analyze the performance of models, it is important to examine the assessment metrics. The evaluation of models were performed based on various parameters such as Accuracy, Precision, Recall and F1-measure from the confusion matrix[94]

Confusion matrix can be used to measure the performance of any machine learning classification problem. The thesis focuses on a binary classification problem and categorizes the data into two classes, i.e., cyberbullying or non-cyberbullying. Thus, there would be four combinations of actual and predicted values. The confusion matrix indicates the TP,TN,FP and FN values.

This sections details the performance parameters, classification reports of the models developed, and plots generated for analysis of the models.

The Accuracy of a model can be defined as the ratio of the number of correct predictions against the total number of predictions made.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The Precision of a model is determined as the proportion of predicted positive cases to the total predicted positives. It helps us to calculate the ratio of relevant data among true positive (TP) and false positive (FP) data belonging to a specific class, i.e., either 0 or 1 .

$$Precision = \frac{TP}{TP + FP}$$

Recall can be defined as the proportion of Real Positive cases that are correctly Predicted Positive.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score is the weighted average of Precision and Recall. F1 Score is calculated using below formula. F1-score helps to combine precision and recall into a single measure that captures both properties.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

4.6 Results, Evaluation and Comparison

This section presents the results of the experiments and discusses their significance. Firstly, it is presented each classifier performance per Feature Extraction(TF-IDF, CountVectorizer) technique for our four datasets and combinations of them. Secondly, it is provided the best classifiers per dataset. Thirdly, it is introduced the best accuracy of any classifier. Finally, it is chosen the best classifier for our model and presented with its confusion matrix.

4.6.1 Results

- Cyber Bullying Types Dataset

As we can notice in the Table 7, the best classifier performance using TF-IDF technique is LR with 91% (90.88%). Moreover, RF and SVM have the second-best performance with 90% (89.71%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	TF-IDF	**90.88**%	**0.99**	0.87	0.92
DT	TF-IDF	87.38%	0.83	0.94	0.88
RF	TF-IDF	89.71%	0.91	0.90	0.91
XGB	TF-IDF	87.38%	0.91	0.87	0.89
NB	TF-IDF	88.78%	0.98	0.84	0.91
SVM	TF-IDF	89.71%	0.94	0.88	0.91
BaggingDT	TF-IDF	88.08%	0.87	0.91	0.89
BoostingDT	TF-IDF	87.38%	0.88	0.89	0.89

Table 7 Cyber Bullying Types Dataset - Evaluation metrics for TF-IDF

As we can notice in the Table 8, the best classifier performance using CountVectorizer technique are NB and SVM with 91% (90.88%). Moreover, LR has the second-best performance with 91% (90.65%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	CountVectorizer	90.65%	0.93	0.90	0.92
DT	CountVectorizer	89.71%	0.88	0.93	0.91
RF	CountVectorizer	90.18%	0.92	0.90	0.91
XGB	CountVectorizer	89.95%	0.94	0.89	0.91
NB	CountVectorizer	**90.88**%	**0.95**	0.89	0.92
SVM	CountVectorizer	90.88%	0.92	0.91	0.92
BaggingDT	CountVectorizer	89.25%	0.93	0.88	0.91
BoostingDT	CountVectorizer	89.01%	0.89	0.91	0.90

Table 8 Cyber Bullying Types Dataset - Evaluation metrics for CountVectorizer

As we can notice in the Table 9, the best classifier performance for Cyber Bullying Types Dataset using TF-IDF/CountVectorizer technique are LR, NB and SVM with 91% (90.88%).

Classifier	Feature Extraction	Accuracy	Rounding
LR	TF-IDF	**90.88**%	91%
DT	CountVectorizer	**89.71**%	90%
RF	CountVectorizer	**90.18**%	90%
XGB	CountVectorizer	**89.95**%	90%
NB	CountVectorizer	**90.88**%	91%
SVM	CountVectorizer	**90.88**%	91%
BaggingDT	CountVectorizer	**89.25**%	89%
BoostingDT	CountVectorizer	**89.01**%	89%

Table 9 Cyber Bullying Types Dataset - Best Accuracy of Algorithms

- Cyber Troll Dataset

As we can notice in the Table 10, the best classifier performance using TF-IDF technique is RF with 92% (91.70%). Moreover, BoostingDT has the second-best performance with 91% (91.22%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	TF-IDF	76.08%	0.64	0.73	0.68
DT	TF-IDF	85.92%	0.83	0.94	0.88
RF	TF-IDF	**91.70**%	**0.94**	0.86	0.90
XGB	TF-IDF	76.00%	0.67	0.71	0.69
NB	TF-IDF	78.20%	0.59	0.81	0.69
SVM	TF-IDF	80.50%	0.80	0.74	0.77
BaggingDT	TF-IDF	84.87%	0.92	0.75	0.83
BoostingDT	TF-IDF	91.22%	0.94	0.85	0.90

Table 10 Cyber Troll Dataset - Evaluation metrics for TF-IDF

As we can notice in the Table 11, the best classifier performance using CountVectorizer technique is BoostingDT with 90% (89.35%). Moreover, RF has the second-best performance with 87% (86.47%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	CountVectorizer	81.57%	0.81	0.75	0.78
DT	CountVectorizer	84.60%	0.96	0.74	0.83
RF	CountVectorizer	86.47%	0.95	0.77	0.85
XGB	CountVectorizer	73.48%	0.57	0.71	0.63
NB	CountVectorizer	79.73%	0.76	0.74	0.75
SVM	CountVectorizer	83.72%	0.88	0.75	0.81
BaggingDT	CountVectorizer	81.75%	0.93	0.71	0.80
BoostingDT	CountVectorizer	**89.35**%	**0.94**	0.82	0.88

Table 11 Cyber Troll Dataset - Evaluation metrics for CountVectorizer

As we can notice in the Table 12, the best classifier performance for Cyber Troll Dataset using TF-IDF technique is RF with 91% (91.47%).

Classifier	Feature Extraction	Accuracy	Rounding
LR	CountVectorizer	**81.57**%	82%
DT	TF-IDF	**85.92**%	86%
RF	TF-IDF	**91.70**%	92%
XGB	TF-IDF	**76.00**%	76%
NB	CountVectorizer	**79.73**%	80%
SVM	CountVectorizer	**83.72**%	84%
BaggingDT	TF-IDF	**84.87**%	85%
BoostingDT	TF-IDF	**91.22**%	91%

Table 12 Cyber Troll Dataset - Best Accuracy of Algorithms

- Classified Tweets Dataset

As we can notice in the Table 13, the best classifier performance using TF-IDF technique is RF with 91% (91.45%). Moreover, XGB has the second-best performance with 91% (90.95%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	TF-IDF	90.52%	0.98	0.91	0.95
DT	TF-IDF	89.10%	0.93	0.94	0.94
RF	TF-IDF	**91.45**%	**0.98**	0.93	0.95
XGB	TF-IDF	90.95%	0.98	0.92	0.95
NB	TF-IDF	87.89%	1.00	0.88	0.93
SVM	TF-IDF	91.30%	0.98	0.93	0.95
BaggingDT	TF-IDF	91.17%	0.96	0.94	0.95
BoostingDT	TF-IDF	90.24%	0.97	0.93	0.95

Table 13 Classified Tweets Dataset - Evaluation metrics for TF-IDF

As we can notice in the Table 14, the best classifier performance using CountVectorizer technique is XGB with 91% (91.38%). Moreover, LR has the second-best performance with 91% (91.07%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	CountVectorizer	91.07%	0.97	0.93	0.95
DT	CountVectorizer	88.90%	0.94	0.93	0.94
RF	CountVectorizer	90.92%	0.97	0.93	0.95
XGB	CountVectorizer	**91.38**%	**0.98**	0.93	0.95
NB	CountVectorizer	90.39%	0.97	0.93	0.95
SVM	CountVectorizer	90.24%	0.95	0.94	0.94
BaggingDT	CountVectorizer	90.95%	0.96	0.94	0.95
BoostingDT	CountVectorizer	89.40%	0.96	0.92	0.94

Table 14 Classified Tweets Dataset - Evaluation metrics for CountVectorizer

As we can notice in the Table 15, the best classifier performance for Classified Tweets Dataset using TF-IDF technique is RF with 91% (91.45%).

Classifier	Feature Extraction	Accuracy	Rounding
LR	CountVectorizer	**91.07**%	91%
DT	TF-IDF	**89.10**%	89%
RF	TF-IDF	**91.45**%	91%
XGB	CountVectorizer	**91.38**%	91%
NB	CountVectorizer	**90.39**%	90%
SVM	TF-IDF	**91.30**%	91%
BaggingDT	TF-IDF	**91.17**%	91%
BoostingDT	TF-IDF	**90.24**%	90%

Table 15 Classified Tweets Dataset - Best Accuracy of Algorithms

- Cyberbullying Classification Dataset

As we can notice in the Table 16, the best classifier performance using TF-IDF technique is SVM with 86% (86.27 %). Moreover, XGB has the second-best performance with 86% (86.26%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	TF-IDF	86.10%	0.98	0.87	0.92
DT	TF-IDF	82.23%	0.90	0.89	0.89
RF	TF-IDF	84.12%	0.95	0.887	0.91
XGB	TF-IDF	86.26%	0.98	0.87	0.92
NB	TF-IDF	84.67%	1.00	0.85	0.92
SVM	TF-IDF	**86.27**%	**0.97**	0.88	0.92
BaggingDT	TF-IDF	84.61%	0.93	0.89	0.91
BoostingDT	TF-IDF	83.54%	0.93	0.88	0.90

Table 16 Cyberbullying Classification Dataset - Evaluation metrics for TF-IDF

As we can notice in the Table 17, the best classifier performance using CountVectorizer technique is SVM with 86% (86.27 %). Moreover, XGB has the second-best performance with 86% (86.26%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	CountVectorizer	86.10%	0.98	0.87	0.92
DT	CountVectorizer	82.07%	0.90	0.89	0.89
RF	CountVectorizer	84.43%	0.95	0.87	0.91
XGB	CountVectorizer	86.26%	0.98	0.87	0.92
NB	CountVectorizer	84.67%	1.00	0.85	0.92
SVM	CountVectorizer	**86.27**%	**0.97**	0.88	0.92
BaggingDT	CountVectorizer	84.19%	0.93	0.89	0.91
BoostingDT	CountVectorizer	83.55%	0.93	0.88	0.90

Table 17 Cyberbullying Classification Dataset - Evaluation metrics for CountVectorizer

As we can notice in the Table 18, the best classifier performance for Cyberbullying Classification Dataset using TF-IDF/CountVectorizer technique is SVM with 86% (86.27%).

Classifier	Feature Extraction	Accuracy	Rounding
LR	TF-IDF/CountVectorizer	**86.10**%	86%
DT	TF-IDF	**82.23**%	82%
RF	CountVectorizer	**84.43**%	83%
XGB	TF-IDF/CountVectorizer	**86.26**%	86%
NB	TF-IDF/CountVectorizer	**84.67**%	85%
SVM	TF-IDF/CountVectorizer	**86.27**%	86%
BaggingDT	TF-IDF	**84.61**%	85%
BoostingDT	CountVectorizer	**83.55**%	84%

Table 18 Cyberbullying Classification Dataset - Best Accuracy of Algorithms

- Combination of Datasets

As we can notice in the Table 19, the best classifier performance using TF-IDF technique is SVM with 85% (84.72%). Moreover, LR has the second-best performance with 85% (84.57%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	TF-IDF	84.57%	0.89	0.88	0.89
DT	TF-IDF	80.03%	0.84	0.85	0.85
RF	TF-IDF	81.77%	0.87	0.86	0.86
XGB	TF-IDF	84.50%	0.82	0.94	0.88
NB	TF-IDF	74.90%	0.98	0.74	0.84
SVM	TF-IDF	**84.72**%	**0.88**	0.89	0.88
BaggingDT	TF-IDF	82.69%	0.86	0.88	0.87
BoostingDT	TF-IDF	80.65%	0.86	0.85	0.86

Table 19 Combination of Datasets - Evaluation metrics for TF-IDF

As we can notice in the Table 20, the best classifier performance using CountVectorizer technique is SVM with 85% (84.72%). Moreover, LR has the second-best performance with 85% (84.57%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	CountVectorizer	84.57%	0.89	0.88	0.89
DT	CountVectorizer	80.11%	0.84	0.86	0.84
RF	CountVectorizer	82.03%	0.87	0.86	0.87
XGB	CountVectorizer	84.50%	0.82	0.94	0.88
NB	CountVectorizer	74.90%	0.98	0.74	0.84
SVM	CountVectorizer	**84.72**%	**0.88**	0.89	0.88
BaggingDT	CountVectorizer	82.65%	0.86	0.88	0.87
BoostingDT	CountVectorizer	80.48%	0.86	0.85	0.85

Table 20 Combination of Datasets - Evaluation metrics for CountVectorizer

As we can notice in the Table 21, the best classifier performance for Combination of Datasets using TF-IDF/ CountVectorizer technique is SVM with 85% (84.72 %).

Classifier	Feature Extraction	Accuracy	Rounding
LR	TF-IDF/CountVectorizer	**84.57**%	85%
DT	CountVectorizer	**80.11**%	80%
RF	CountVectorizer	82.03%	82%
XGB	TF-IDF/CountVectorizer	**84.50**%	85%
NB	TF-IDF/CountVectorizer	**74.90**%	75%
SVM	TF-IDF/CountVectorizer	**84.72**%	85%
BaggingDT	TF-IDF	**82.69**%	83%
BoostingDT	TF-IDF	**80.65**%	81%

Table 21 Combination of Datasets - Best Accuracy of Algorithms

- Cyber Bullying Types Dataset & Cyber Troll Dataset

As we can notice in the Table 22, the best classifier performance using TF-IDF technique is RF with 90% (90.29%). Moreover, BoostingDT has the second-best performance with 90% (90.06%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	TF-IDF	78.12%	0.81	0.80	0.80
DT	TF-IDF	85.45%	0.93	0.79	0.85
RF	TF-IDF	**90.29**%	**0.92**	0.87	0.89
XGB	TF-IDF	75.52%	0.82	0.76	0.79
NB	TF-IDF	79.95%	0.85	0.80	0.82
SVM	TF-IDF	80.60%	0.82	0.76	0.79
BaggingDT	TF-IDF	84.84%	0.92	0.78	0.84
BoostingDT	TF-IDF	90.06%	0.92	0.87	0.89

Table 22 Cyber Bullying Types Dataset & Cyber Troll Dataset - Evaluation of TF-DF

As we can notice in the Table 23, the best classifier performance using CountVectorizer technique is BoostingDT with 89% (89.16%). Moreover, RF has the second-best performance with 86% (85.52%).

Classifier	Feature Extraction	Accuracy	Precision	Recall	F1
LR	CountVectorizer	81.89%	0.82	0.78	0.80
DT	CountVectorizer	83.60%	0.93	0.76	0.83
RF	CountVectorizer	85.52%	0.92	0.79	0.85
XGB	CountVectorizer	73.51%	0.83	0.73	0.78
NB	CountVectorizer	79.58%	0.82	0.81	0.82
SVM	CountVectorizer	83.24%	0.87	0.78	0.82
BaggingDT	CountVectorizer	82.45%	0.91	0.75	0.82
BoostingDT	CountVectorizer	**89.16**%	**0.91**	0.86	0.88

Table 23 Cyber Bullying Types Dataset & Cyber Troll Dataset - Evaluation of CountVectorizer

As we can notice in the Table 24, the best classifier performance for Cyber Bullying Types Dataset & Cyber Troll Dataset using TF-IDF technique is RF with 90% (90.29%).

Classifier	Feature Extraction	Accuracy	Rounding
LR	CountVectorizer	81.89%	82%
DT	TF-IDF	85.45%	85%
RF	TF-IDF	**90.29**%	90%
XGB	TF-IDF	75.52%	75%
NB	TF-IDF	79.95%	80%
SVM	CountVectorizer	83.24%	83%
BaggingDT	TF-IDF	84.84%	85%
BoostingDT	TF-IDF	90.06%	90%

Table 24 Cyber Bullying Types Dataset & Cyber Troll Dataset - Best Accuracy of Algorithms

4.6.2 Model Selection

As we can notice in the Table 25, the best classifier performance for all datasets and combinations of datasets is **RF** using **TF-IDF** from **Cyber Troll Dataset** with 92% (91.70%). The second-best classifier is **XGB** using **CountVectorizer** from **Classified Tweets Dataset** with 91% (91.38%). The third-best classifier is **SVM** using **TF-IDF** from **Classified Tweets Dataset** with 91% (91.30%).

Classifier	Dataset	Feature Extraction	Accuracy	Precision	Rounding
LR	Classified Tweets Dataset	CountVectorizer	91.07%	0.97	91%
DT	Cyber Bullying Types Dataset	CountVectorizer	89.71%	0.88	90%
RF	**Cyber Troll Dataset**	**TF-IDF**	**91.70%**	**0.94**	**92%**
XGB	**Classified Tweets Dataset**	**CountVectorizer**	**91.38%**	**0.98**	**91%**
NB	Cyber Bullying Types Dataset	CountVectorizer	90.88%	0.95	91%
SVM	**Classified Tweets Dataset**	**TF-IDF**	**91.30%**	**0.98**	**91%**
BaggingDT	Classified Tweets Dataset	TF-IDF	91.17%	0.96	91%
BoostingDT	Cyber Troll Dataset	TF-IDF	91.22%	0.94	91%

Table 25 Best Model per Algorithm

Finally, our best model is **RF** using **TF-IDF** from **Cyber Troll Dataset** with detection accuracy of 92% (91.70%). Below, it is presented the Confusion Matrix of our selected model.

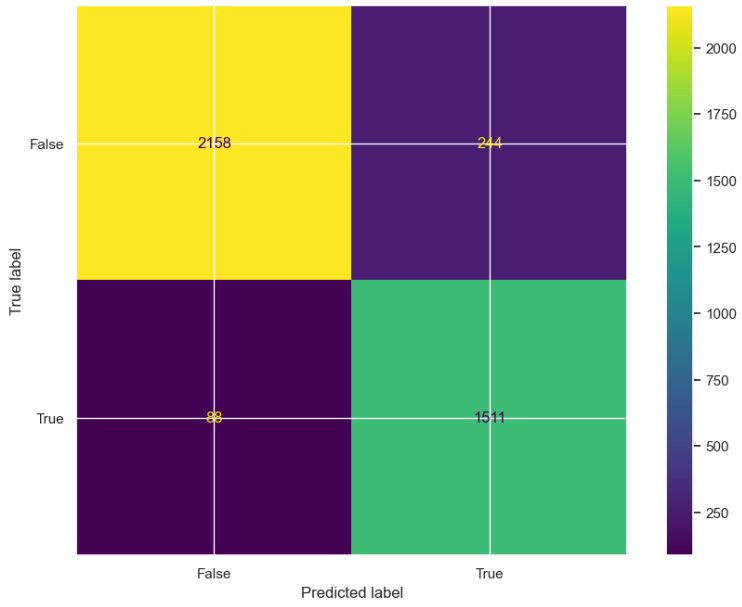


Table 26 Confusion Matrix of Best Model

5.Cyber Bullying Detection Application (CBDA)

In this chapter, we will present Cyber Bullying Detection Application that is a web application for cyberbullying detection online. CBDA has developed using Python programming language and Streamlit framework, which is a web framework for Machine Learning model deployment. The basic functionality of CBDA is to predict if the submitted text is cyberbullying or not.

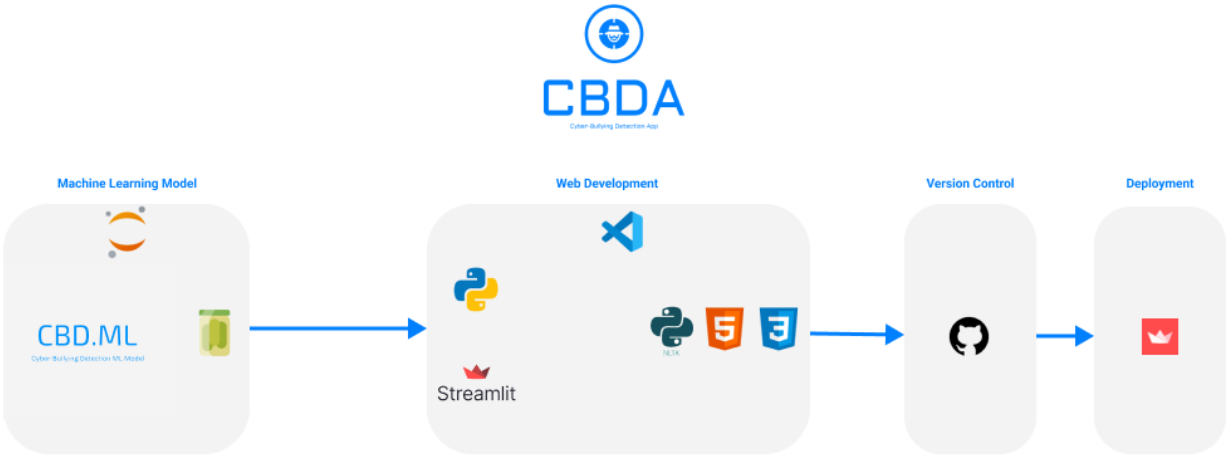
5.1 Idea

After creating our Machine Learning Model, we made research to find some extra patterns in cyberbullying detection. As a result, we decide to deepen into the field of cyberbullying detection and deploy our model as web application, which any user of world wide web can access.

The basic functionality of the app is to predict if the user input text is cyberbullying or not. However, we wanted to develop an application more user-friendly. For this reason, we decide to add some more complex features, as dataset and algorithms page, with purpose to use all the benefits of our Machine Learning model development.

5.2 Technologies and Tools

For the development of CBDA, we use many technologies and tools that help us to achieve our goal and final version of our application. We split the development process in three(3) phases; Machine Learning model export, Web development and Deployment, as it presented below.



- Pickle

  Pickle is a Python standard library for object serialization and deserialization. Pickle serialization saves objects into a file in byte format while deserialization is the reverse of serialization.[84] If you try to serialize an object into a pickle file without first specifying the byte/binary format, you will encounter an error. Several objects can be pickled/serialized such as Machine Learning models.

- Streamlit

  Streamlit is an open-source framework for Machine Learning and Data Science web applications. It is user-friendly and allows you to turn data scripts into web apps using Python.[85]

- Html5

  HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and final major HTML version that is a World Wide Web Consortium (W3C) recommendation. The current specification is known as the HTML Living Standard. [86]

- Css3

  Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.[87]

- Streamlit Cloud

  Streamlit Cloud is an open and free platform for the community to deploy, discover, and share Streamlit apps and code with each other. It is very user-friendly and easy to deploy your app using a GitHub account with your repository code.[88]

5.3 Functionality

As we introduce above, the main functionality of CBDA application is to predict if the user input (text) is cyberbullying or not. Let's discover together the full functionality of our web application for cyberbullying detection.

- **CBDA – Home Page**

  Home page of CBDA presents the menu of our application as Cyberbullying Detection, Datasets, Algorithms and About us page.

Figure 46 CBDA - Home Page

- **CBDA – Cyberbullying Detection Page**

In Cyberbullying Detection Page, user can write his text, which want to determine if it is cyberbullying or not. In analysis procedure, first of all we predict the result, cyberbullying or not. Secondly, we provide information for the prediction process as original text, cleaned text, transformed text, binary prediction and model accuracy.

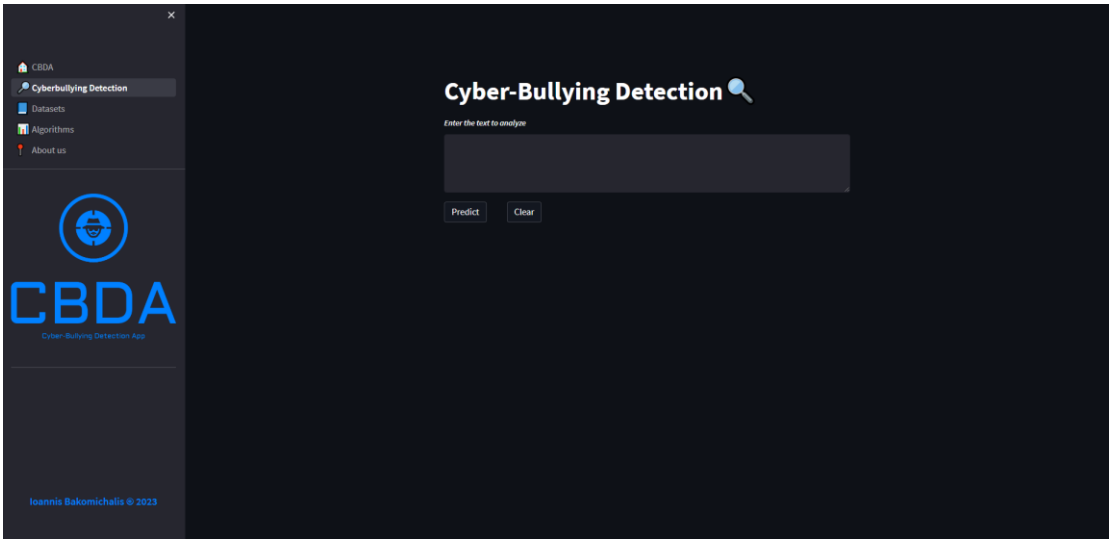


Figure 47 CBDA - Cyberbullying Detection

- Cyberbullying text

User gives as input "*you are SUCH A fucking dork*", which seems like cyberbullying text, and press the "*Predict*" button. Let's discover the whole analysis process.

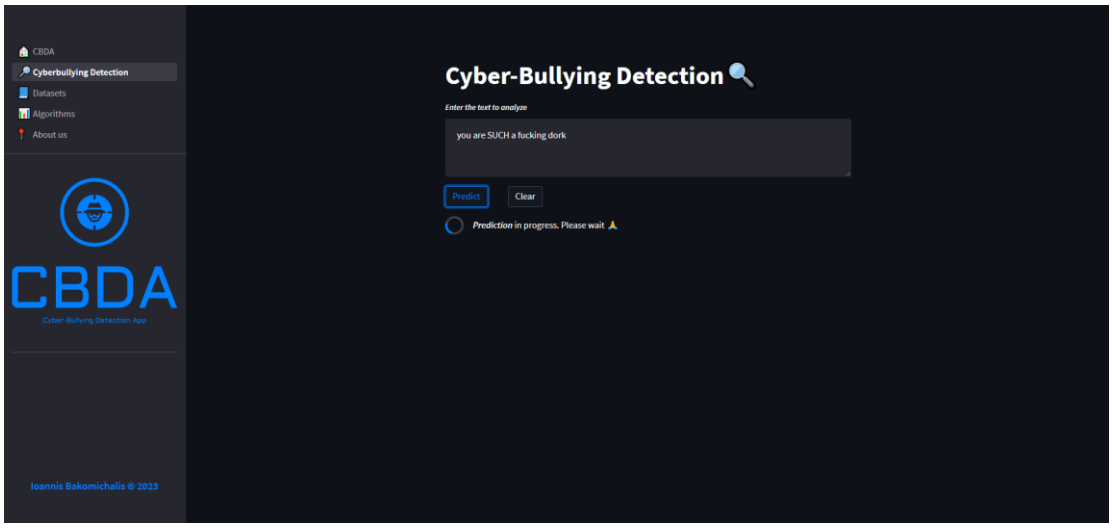


Figure 48 CBDA - Cyberbullying Detection - Cyberbullying Text

First step is the **Result**, in which the user input has predicted as **Cyberbullying**.

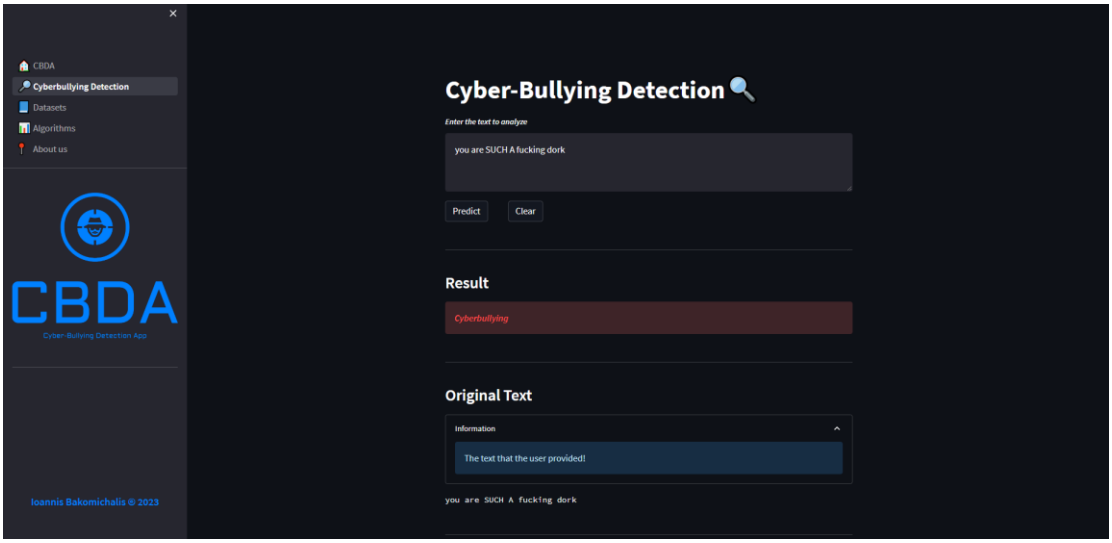


Figure 49 CBDA - Cyberbullying Detection - Cyberbullying Text – Result

Second step is **Original Text**, which is the text that the user gave as input to predict if it is cyberbullying or not.

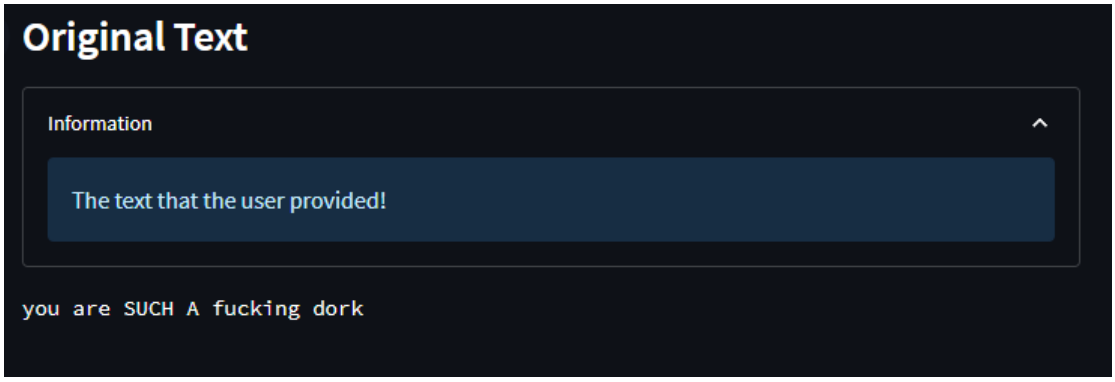


Figure 50 CBDA - Cyberbullying Detection - Cyberbullying Text – Original Text

Third step is the **Cleaned Text**, which is the original text without punctuation, special characters, hashtags, tags and emojis.

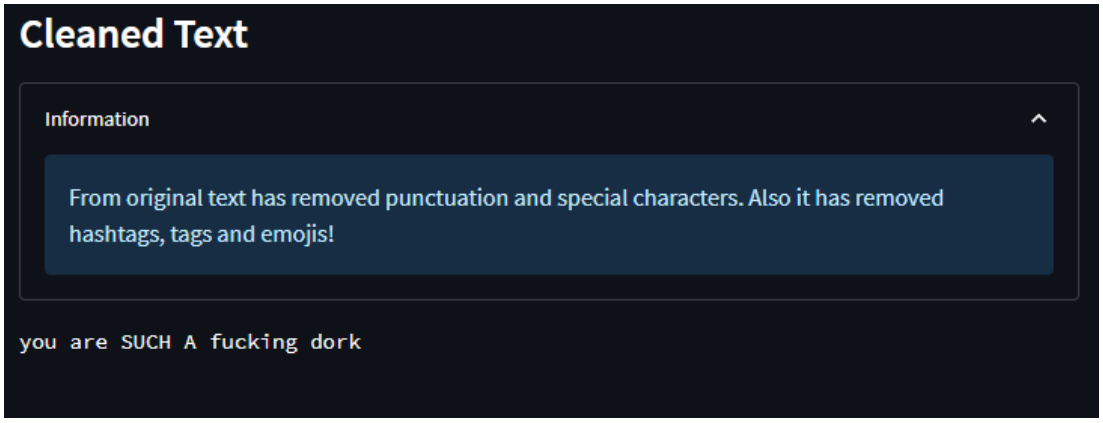


Figure 51 CBDA - Cyberbullying *Detection - Cyberbullying Text – Cleaned Text*

Forth step is the **Transformed Text**, which is the cleaned text in lowercase without stopwords. Also, it has been used Stemming technique.

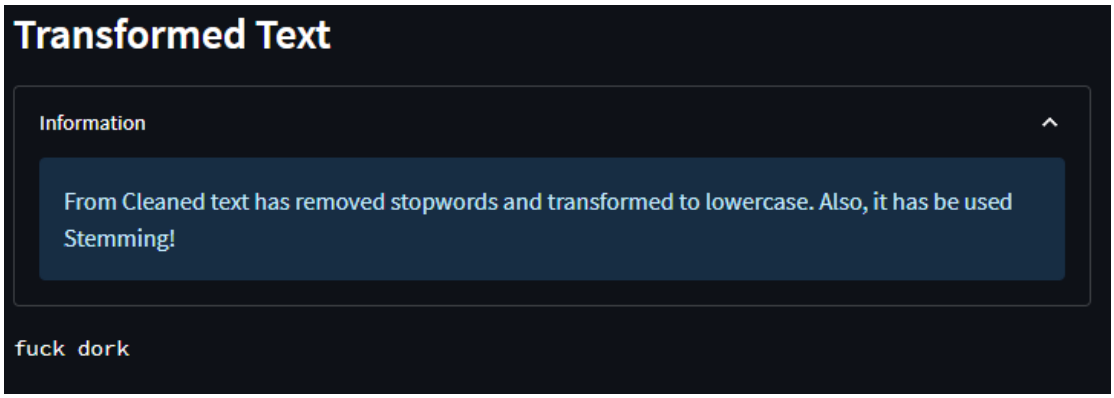


Figure 52 CBDA - Cyberbullying Detection - Cyberbullying Text – Transformed Text

Fifth step is **Binary Prediction**, which is the prediction of the model.

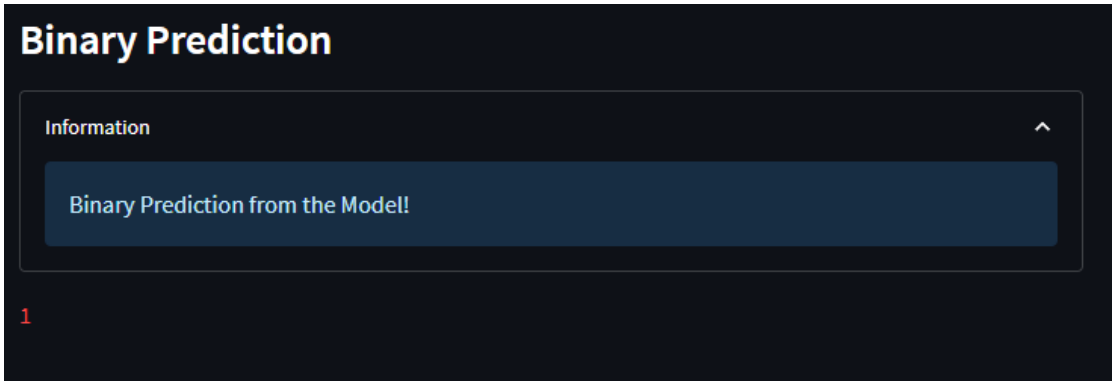


Figure 53 CBDA - Cyberbullying ,Detection - Cyberbullying Text – Binary Prediction

Final step is **Model Accuracy**, which is **91.7%** using Random Forest (RF) Classifier, TF-IDF and Cyber Troll Dataset.

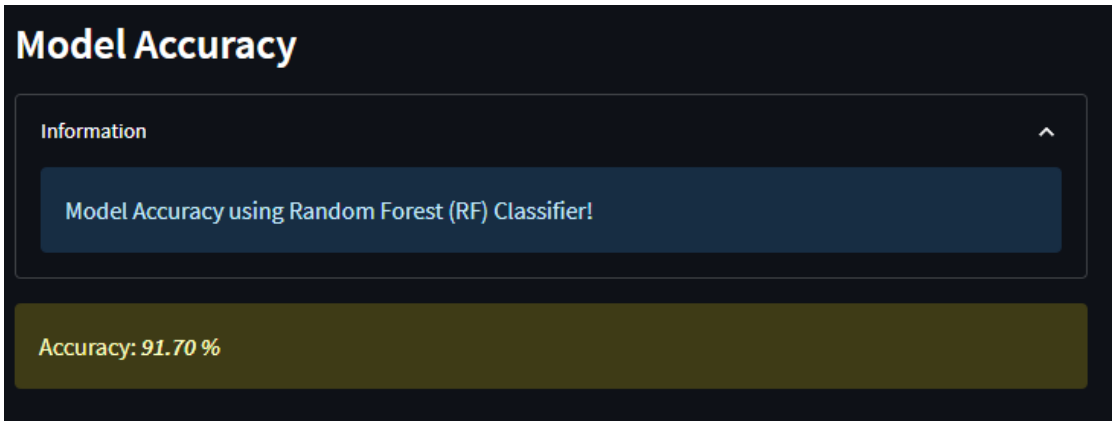


Figure 54 CBDA - Cyberbullying ,Detection - Cyberbullying Text – Model Accuracy

- Not Cyberbullying text

User gives as input "*I hate going out in summer for that reason :(*". Let's see briefly the analysis process.

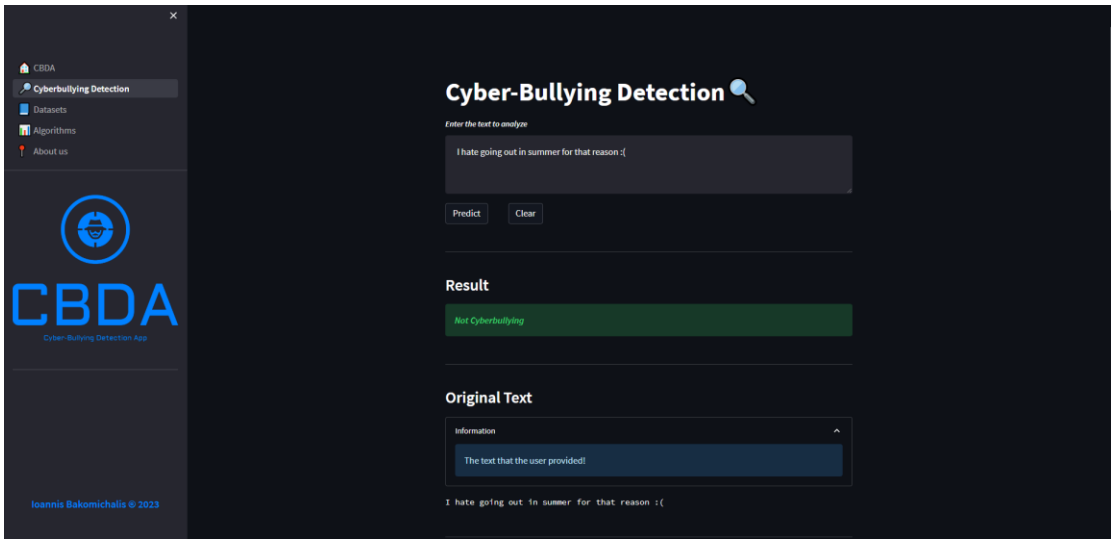


Figure 55 CBDA - Cyberbullying Detection - Not - Cyberbullying Text

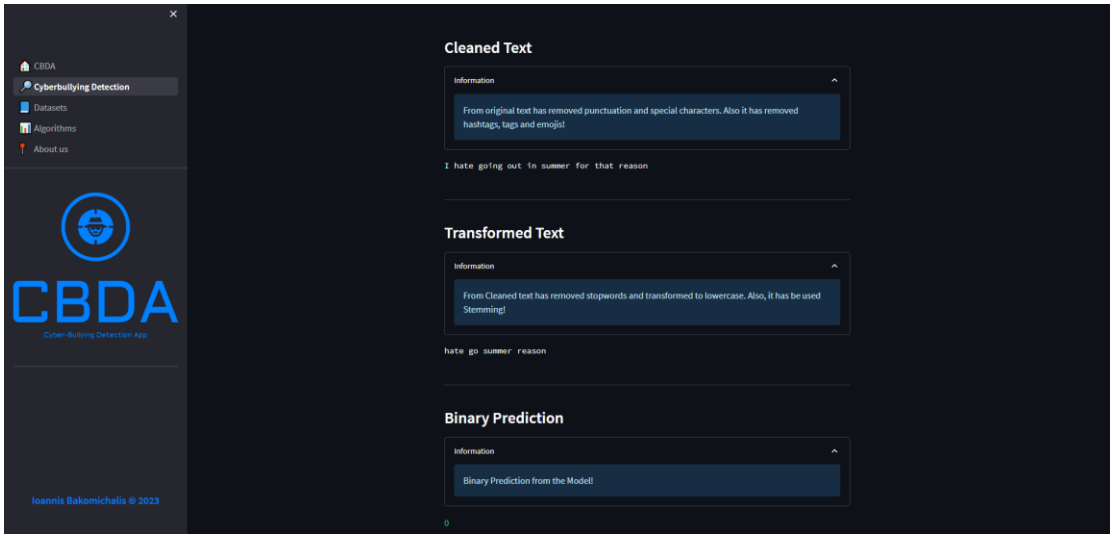


Figure 56 CBDA - Cyberbullying Detection - Not - Cyberbullying Text 2

- **Datasets Page**

In Datasets Page, user should select a dataset between:

- **Cyber Bullying Types Dataset**
- **Cyber Troll Dataset**
- **Classified Tweets Dataset**
- **Cyberbullying Classification Dataset**
- **Cyber Bullying Types Dataset + Cyber Troll Dataset**
- **Cyber Bullying Types Dataset + Cyber Troll Dataset + Classified Tweets Dataset + Cyberbullying Classification Dataset**.

These datasets has been used to select our final model. In this page, it is provided information, overview and plots about the selected dataset.

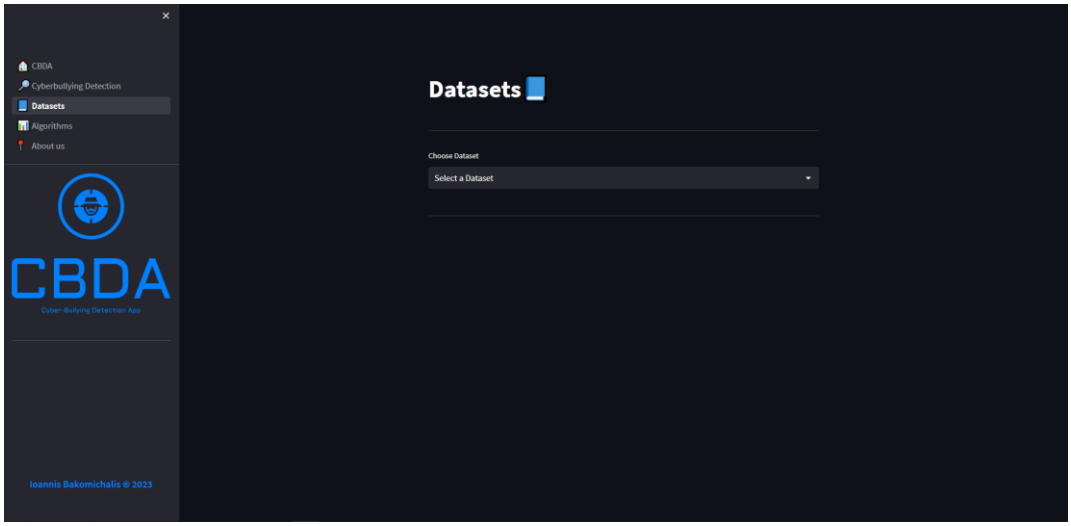


Figure 57 CBDA – Datasets

- Dataset Information

In this subsection, it is provided information on python dataset.info() command, dataset shape, rows shape and columns shape.

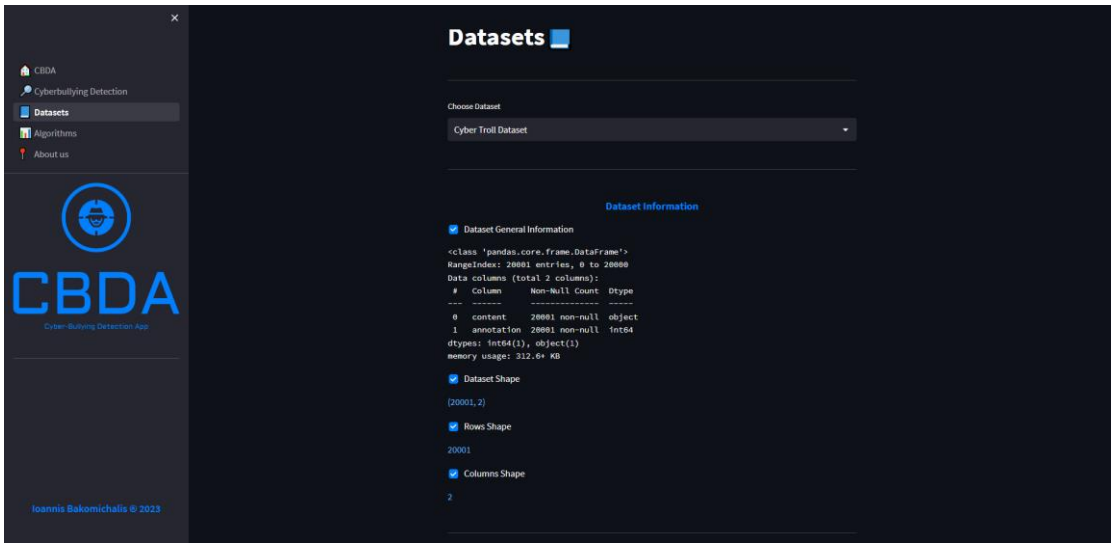


Figure 58 CBDA – Datasets - Dataset Information

- Dataset Overview

In this subsection, it is provided a preview of the dataset, dataset.head() and dataset.tail() python command, dataset columns and dataset.describe() python command.

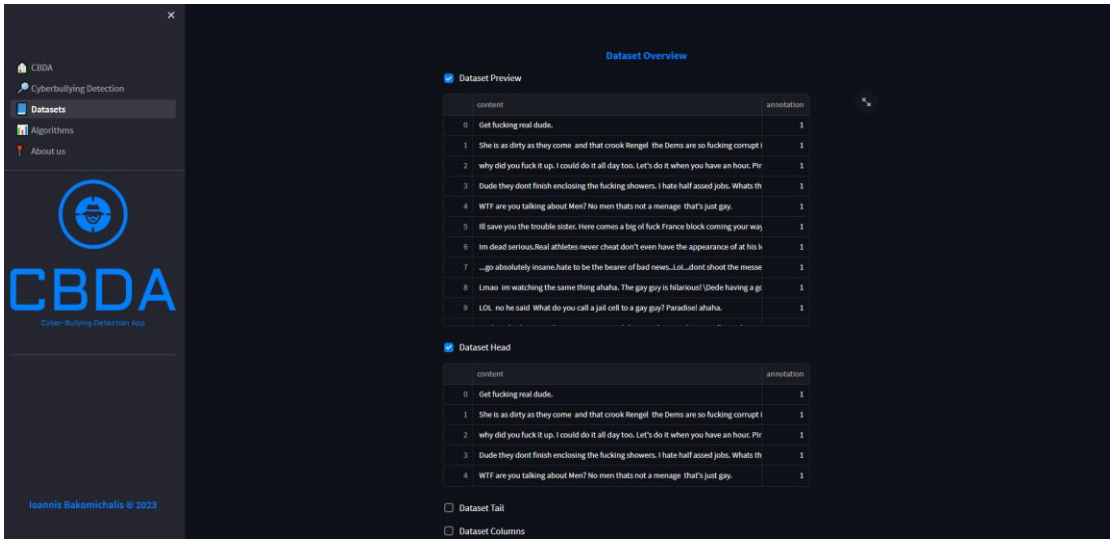


Figure 59 CBDA - Datasets - Dataset Overview

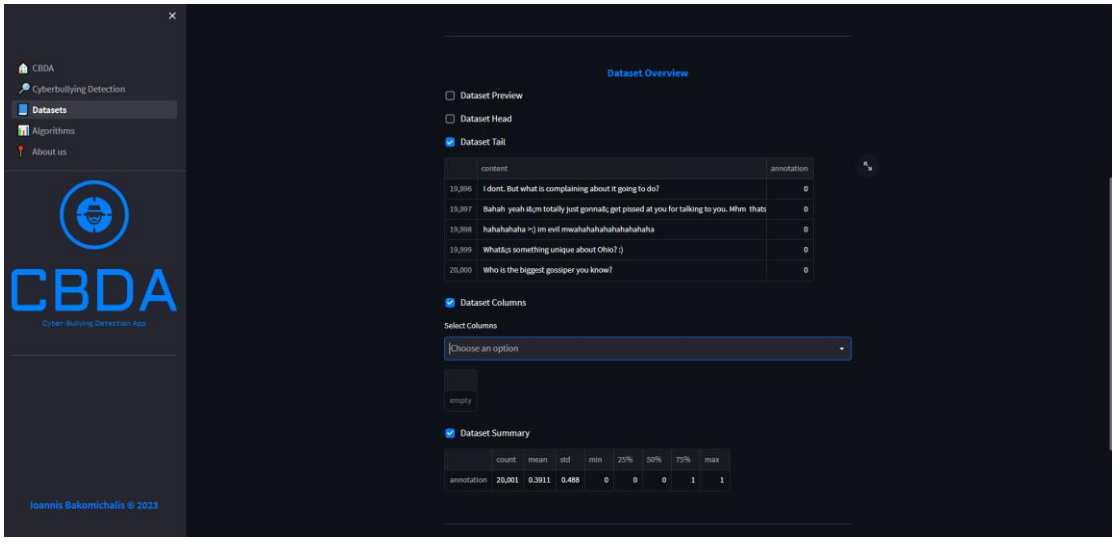


Figure 60 CBDA - Datasets - Dataset Overview 2

- Dataset Plots

In this subsection, it is presented a bar chart and a pie chart for the selected dataset, for the number of values in target column for cyberbullying or not.

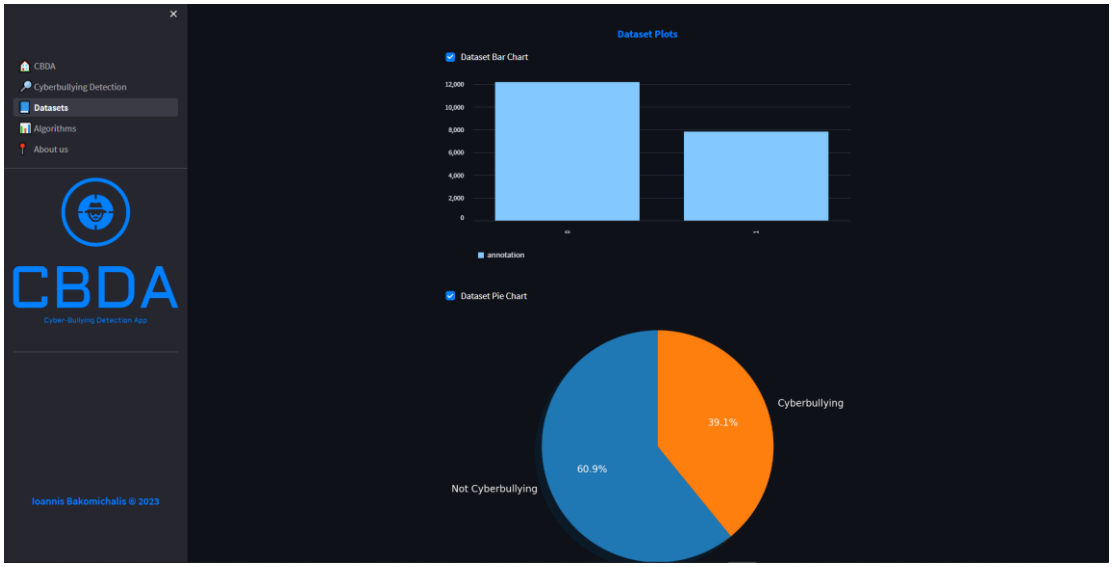


Figure 61 CBDA - Datasets - Dataset Plots

- **Algorithms Page**

In Algorithms Page, user should select Dataset, Vectorizer Technique and Machine Learning algorithm to discover the accuracy of the model. As it mentioned, it is like a playground, where user can find the accuracy metric of our researching approach for the best model.

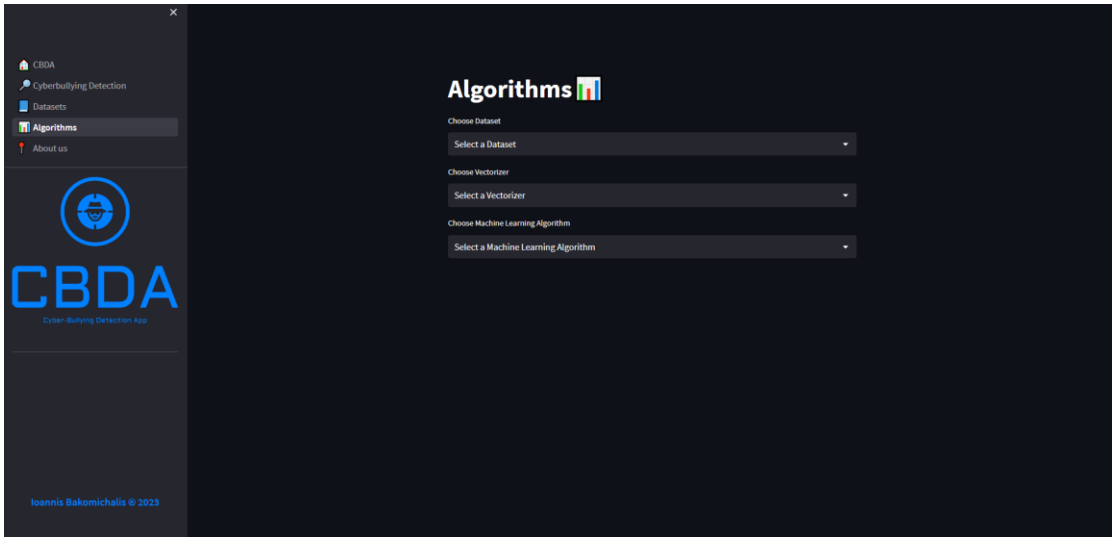


Figure 62 CBDA – Algorithms

User should select between below datasets:

- **Cyber Bullying Types Dataset**
- **Cyber Troll Dataset**
- **Classified Tweets Dataset**
- **Cyberbullying Classification Dataset**
- **Cyber Bullying Types Dataset + Cyber Troll Dataset**
- **Cyber Bullying Types Dataset + Cyber Troll Dataset + Classified Tweets Dataset + Cyberbullying Classification Dataset**.

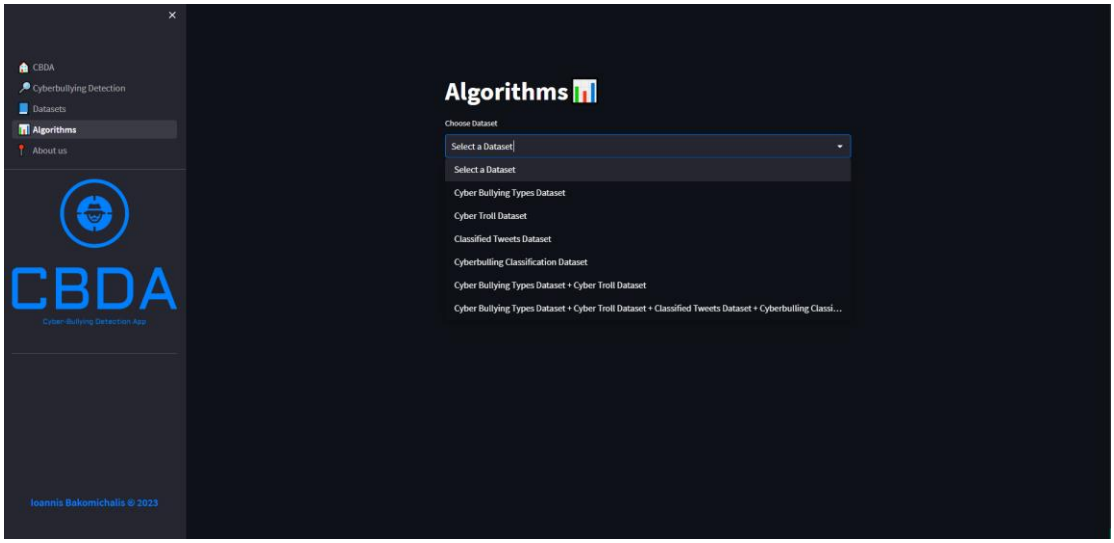


Figure 63 CBDA - Algorithms - Select Dataset

User should select between below Vectorization techniques:

- **TF-IDF**
- **CountVectorizer**

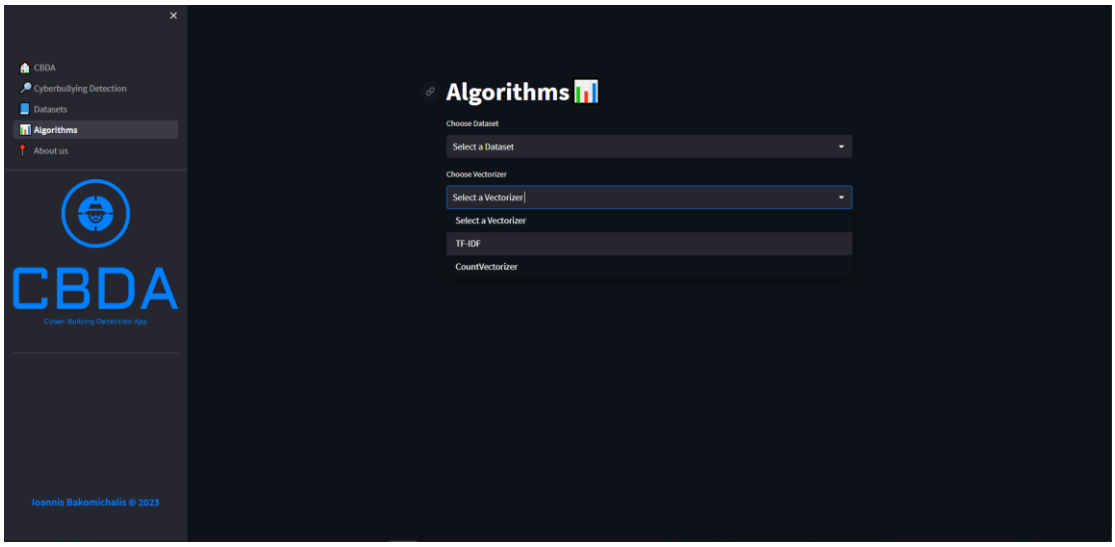


Figure 64 CBDA – Algorithms - Select Vectorizer

User should select between below Machine Learning Algorithms:

- **Logistic Regression (LR)**
- **Decision Tree (DT)**
- **Random Forest (RF)**
- **XGBoost (XGB)**
- **Multinomial Naïve Bayes (NB)**
- **Support Vector Machine (SVM)**
- **Bagging Decision Tree (BaggingDT)**
- **Boosting Decision Tree (BoostingDT)**

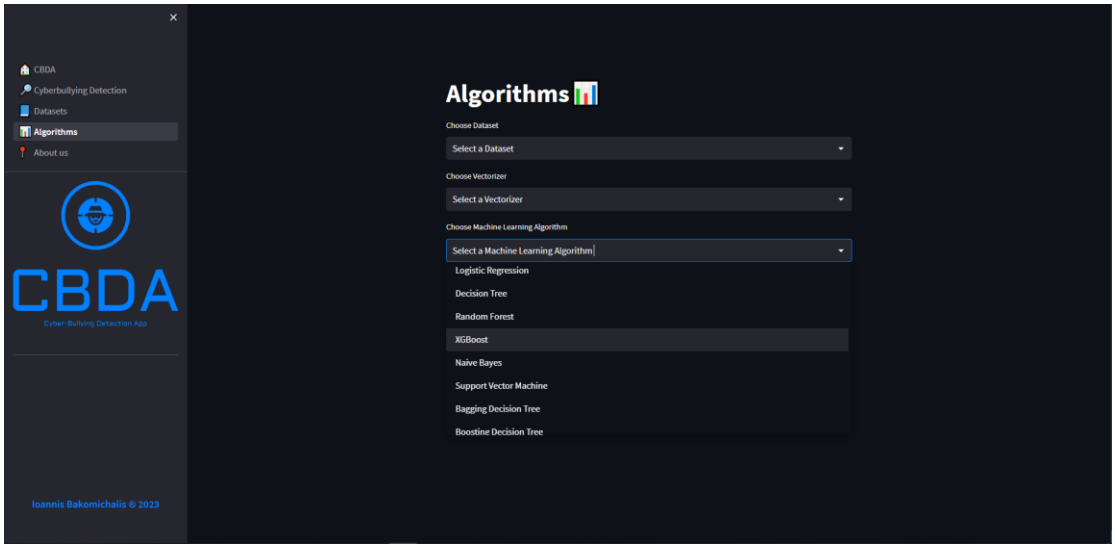


Figure 65 CBDA - Algorithms - Select Algorithm

As an example, user select **Cyber Troll Dataset**, **TF-IDF** and **Random Forest** classifier, which is our best model, and has accuracy **91.70%** (92%)

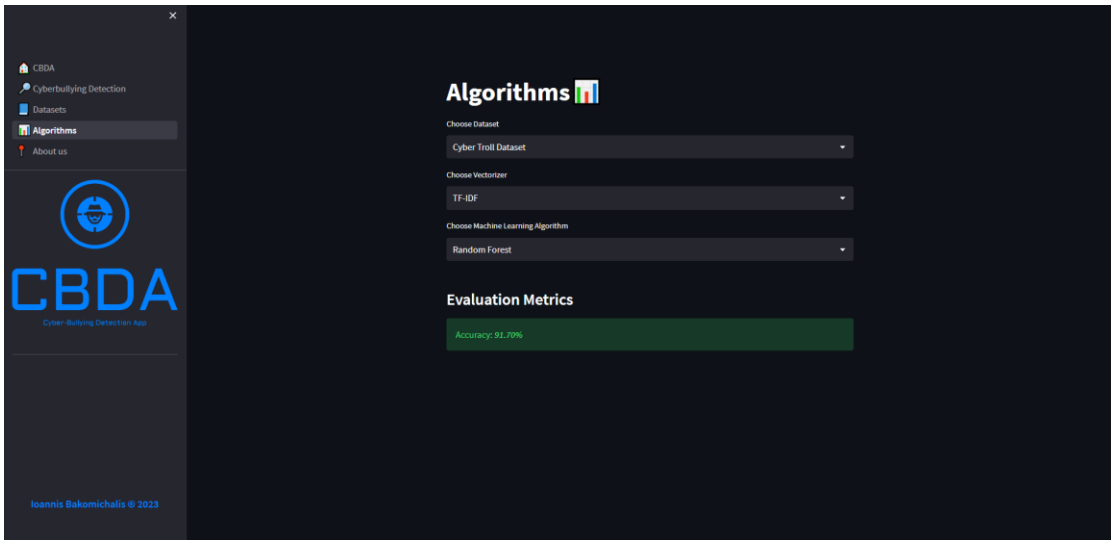


Figure 66 CBDA - Algorithms – Example

- **About us Page**

In About us Page, user can collect some information about the project and communication details.

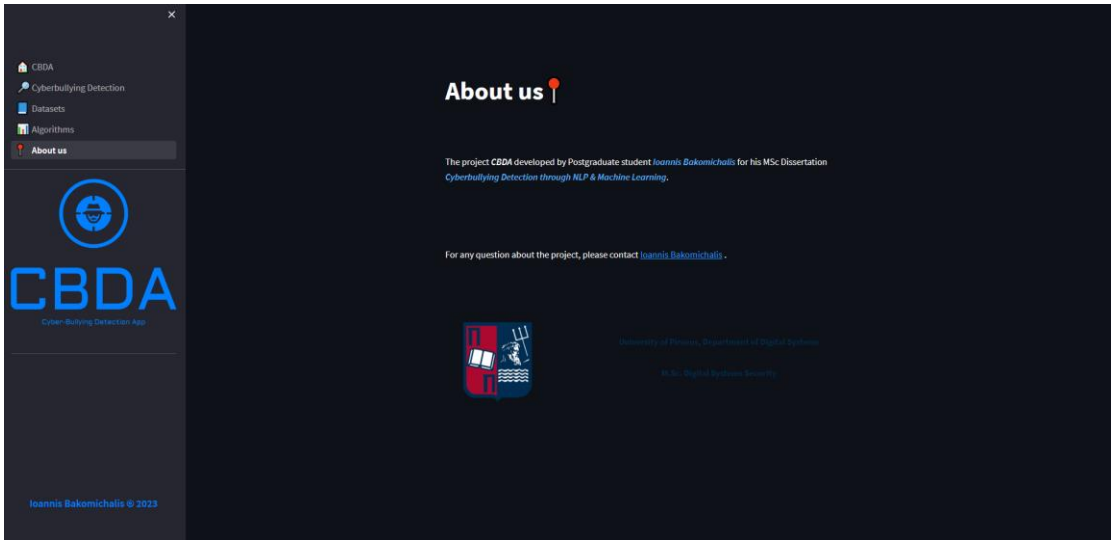


Figure 67 CBDA - About Us

6.Conclusion

In this section of the conclusion, we provided an overview of our intentions behind this thesis work and the results that we obtained. We also discussed possible enhancements that could help in this area.

6.1 Conclusion

The impact of cyberbullying is dramatically increasing due to ease of access to the Internet. This results in psychological and physical harm to victims. Our first task was to understand cyberbullying thoroughly, frequent targets of cyberbullying, and its consequences. Only then, we could identify and detect it successfully. The primary goal of this thesis is to introduce a new model as well as a web application for cyberbullying detection.

Since most of the authors collect data and classify the contents without actually making it publicly available; therefore, a common problem in any machine learning projects is the unavailability of data. Data is crucial in any research as it makes the research successful by comparing results and approaches. For this purpose, we collected 4 datasets for cyberbullying detection from IEEE, Zenodo and Kaggle. We examined any dataset and their combination using eight classification algorithms and different Natural Language Processing and Machine Learning techniques, with purpose to find the best model for our machine learning problem.

For cyberbullying detection, we have experimented with Logistic Regression, Decision Tree, Random Forest, XGBoost, Multinomial Naïve Bayes, Support Vector Machine, Bagging Decision Tree and Boosting Decision Tree. Moreover, we use TF-IDF and CountVectorizer techniques for feature extraction. Finally, we use stemming and lemmatizing.

In Cyber Bullying Types Dataset, LR has the best performance with a detection accuracy of 91% (90.88%) and precision of 99%, using TF-DF. For Cyber Troll Dataset, RF has accuracy 92% (91.70%) with TF-IDF. Furthermore, Classified Tweets Dataset using RF has accuracy 91% (91.45%) with TF-IDF. In Cyberbullying

Classification Dataset, SVM with TF-IDF/CV has detection accuracy 86% (86.27%). The best classifier performance for Combination of Datasets using TF-IDF/ CountVectorizer technique is SVM with 85% (84.72). In Cyber Bullying Types Dataset & Cyber Troll Dataset RF with TF-IDF technique has accuracy of 90% (90.29%). Finally, our best model is RF using TF-IDF from Cyber Troll Dataset with detection accuracy of 92% (91.70%).

For the development of our web application CBDA, we used the best model of our examination of datasets, techniques and classifiers. In CBDA, user can detect cyberbullying automatically, discover our datasets and examine our approach with a variety of algorithms, datasets and NLP/ML techniques.

6.2 Future Experiments & Work

Cyberbullying detection is a research field where plenty of advancements can be made. We believe that the first step should be to standardize the definition of cyberbullying globally, because only a model can induce something that we, as humans, are not fully aware of. Stricter rules and more powerful guidelines could be significant steps toward homogenizing the concept, which is still unclear and varies by country.

From Feature Engineering approach, can be use N-grams in TF-IDF feature extraction technique. Moreover, it can be used Number of Sentences, Number of Words and Number of Characters, that can give us better results in this classification problem.

Moreover, it can be used bigger and more clear datasets or combination of them, which will have been developed from researchers specific for cyberbullying detection. As we know in data science and Machine Learning the most important aspect is the data.

Finally, it can be used Deep Learning for examining the evaluation metrics in the selected datasets, using different types of neurons and Deep Learning techniques.

References

[1] A Systematic Review of Machine Learning Algorithms in Cyberbullying Detection: Future Directions and Challenges, Muhammad Arif, 2021.

[2] Alotaibi, M.; Alotaibi, B.; Razaque, A. A Multichannel Deep Learning Framework for Cyberbullying Detection on Social Media. Electronics 2021, 10, 2664. https://doi.org/10.3390/electronics 10212664.

[3] A Study of Cyberbullying Detection Using Machine Learning Techniques, S.M. Kargutkar, V. Chitre, 2020.

[4] A Comparative Analysis of Machine Learning Techniques for Cyberbullying Detection on Twitter, A. Muneer, S. M. Fati, 2020.

[5] Cyber Bullying Detection on Social Media using Machine Learning, 2021.

[6] Raj, C.; Agarwal, A.; Bharathy, G.; Narayan, B.; Prasad, M. Cyberbullying Detection: Hybrid Models Based on Machine Learning and Natural Language Processing Techniques. Electronics 2021, 10, 2810. https://doi.org/10.3390/electronics10222810 .

[7] Cyberbullying Detection on Social Networks Using Machine Learning Approaches, 2020.

[8] Cyberbullying Detection using Natural Language Processing, IJRASET Publication 2022, International Journal for Research in Applied Science & Engineering Technology (IJRASET).

[9] Cyber-Bullying Detection via Text Mining and Machine Learning, 2021.

[10] Detecting Cyberbullying in Social Commentary Using Supervised Machine Learning, 2020.

[11] Detection of Cyberbullying on Social Media using Machine Learning, 2022.

[12]Social Media Cyberbullying Detection using Machine Learning, 2019.

[13] Jahan M. (2020) Cyber Bullying Identification and Tackling Using Natural Language Processing Techniques. University of Oulu, Degree Programme in Computer Science and Engineering.

[14] "Cyberbullying: What Is It and How to Stop It | UNICEF." *UNICEF*, https://www.unicef.org/end-violence/how-to-stop-cyberbullying .

[15] "CYBERBULLYING | English Meaning - Cambridge Dictionary." *Cambridge Dictionary | English Dictionary, Translations & Thesaurus*, https://dictionary.cambridge.org/dictionary/english/cyberbullying . Accessed 27 Feb. 2023.

[16] "What Is Cyberbullying | StopBullying.Gov." *StopBullying.Gov*, 24 Sept. 2019, https://www.stopbullying.gov/cyberbullying/what-is-it .

[17] "HISTORY – CYBER BULLYING." *CYBER BULLYING*, 13 Oct. 2018, https://cyberbullying4less.wordpress.com/history/ .

[18] The Bark Team. "The History of Cyberbullying: A Helpful, Informative Guide - Bark." *Bark*, 22 Mar. 2017, https://www.bark.us/blog/the-history-of-cyberbullying/ .

[19] "Cyberbullying Facts - Cyberbullying Research Center." *Cyberbullying Research Center*, https://cyberbullying.org/facts .

[20] Pandey, Eshaan. "Natural Language Processing (NLP) in Data Science: 10 Techniques." *KnowledgeHut: Professional Bootcamps and Certification Courses Provider for Your Future*, Knowledgehut, 15 Feb. 2023, https://www.knowledgehut.com/blog/data-science/nlp-data-science .

[21] Diego Lopez. "*Your Guide to Natural Language Processing (NLP)*. "2019, p. https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1.

[22] "What Is Natural Language Processing? | IBM." *IBM - United States*, https://www.ibm.com/topics/natural-language-processing .

[23] Contributors to Wikimedia projects. "Natural Language Processing - Wikipedia." *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., 22 Sept. 2001, https://en.wikipedia.org/wiki/Natural_language_processing .

[24] Louis, Antoine. "A Brief History of Natural Language Processing — Part 1 | by Antoine Louis | Medium." *Medium*, Medium, 7 July 2020, https://medium.com/@antoine.louis/a-brief-history-of-natural-language-processing-part-1-ffbcb937ebce .

[25]Louis, Antoine. “A Brief History of Natural Language Processing — Part 1 | by Antoine Louis | Medium.” *Medium*, Medium, 7 July 2020, https://medium.com/@antoine.louis/a-brief-history-of-natural-language-processing-part-1-ffbcb937ebce .

[26] Vijay, Ronak. “A Gentle Introduction to Natural Language Processing.” *TowardsDataScience*, 29 June 2019, https://towardsdatascience.com/a-gentle-introduction-to-natural-language-processing-e716ed3c0863 .

[27] “Top 10 NLP Tools & Services in 2022.” *MonkeyLearn Blog*, 11 Mar. 2020, https://monkeylearn.com/blog/natural-language-processing-tools/ .

[28] Kozaczko, Dominik. “8 Best Python NLP Libraries | Sunscrapers.” *Sunscrapers*, 26 June 2018, https://sunscrapers.com/blog/8-best-python-natural-language-processing-nlp/ .

[29] “NLTK :: Natural Language Toolkit.” *NLTK :: Natural Language Toolkit*, https://www.nltk.org/ .

[30] “TextBlob: Simplified Text Processing — TextBlob 0.16.0 Documentation.” https://textblob.readthedocs.io/en/dev/ .

[31] “Overview - CoreNLP.” CoreNLP, https://stanfordnlp.github.io/CoreNLP/ .

[32] RaRe-Technologies. “GitHub - RaRe-Technologies/Gensim: Topic Modelling for Humans.” *GitHub*, https://github.com/RaRe-Technologies/gensim .

[33] “SpaCy · Industrial-Strength Natural Language Processing in Python.” *SpaCy · Industrial-Strength Natural Language Processing in Python*, https://spacy.io/ .

[34] “Scikit-Learn: Machine Learning in Python — Scikit-Learn 0.16.1 Documentation.” *Scikit-Learn: Machine Learning in Python — Scikit-Learn 0.16.1 Documentation*, https://scikit-learn.org/ .

[35] “Welcome to Polyglot’s Documentation! — Polyglot 16.07.04 Documentation.” *Welcome to Polyglot’s Documentation! — Polyglot 16.07.04 Documentation*, https://polyglot.readthedocs.io/en/latest/index.html .

[36] “An Introduction to Machine Learning.” MonkeyLearn, https://monkeylearn.com/machine-learning/ .

[37] “An Introduction to Machine Learning - GeeksforGeeks.” *GeeksforGeeks*, GeeksforGeeks, 24 Aug. 2017, https://www.geeksforgeeks.org/introduction-machine-learning/

[38] “What Is Machine Learning? | IBM.” *IBM - United States*, https://www.ibm.com/topics/machine-learning .

[39] Burns. “What Is Machine Learning and Why Is It Important?” *Enterprise AI*, TechTarget,https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML .

[40] “What Is Machine Learning? Everything You Need to Know - Appventurez.” *Appventurez*, 2 June 2020, https://www.appventurez.com/blog/beginners-guide-to-machine-learning .

[41] Jacinto, Andrea. “Machine Learning History | StarTechUP.”, 9 Sept. 2022, https://www.startechup.com/blog/machine-learning-history /.

[42] “Machine Learning Tasks - ML.NET | Microsoft Learn.” *Microsoft Learn: Build Skills That Open Doors in Your Career*, https://learn.microsoft.com/en-us/dotnet/machine-learning/resources/tasks .

[43] “Machine Learning Tasks - Python.” *Learn Python Programming - Python*, https://pythonprogramminglanguage.com/machine-learning-tasks/ .

[44] Kumar, Ajitesh. “Most Common Machine Learning Tasks - Data Analytics.” *Data Analytics*, 4 Dec. 2022, https://vitalflux.com/7-common-machine-learning-tasks-related-methods/ .

[45] Contributors to Wikimedia projects. “Binary Classification - Wikipedia.” https://en.wikipedia.org/wiki/Binary_classification .

[46] ProjectPro. “15 Machine Learning Use Cases and Applications in 2023.” *ProjectPro*, , 5 July 2022, https://www.projectpro.io/article/machine-learning-use-cases/476 .

[47] Sharma, Rounak. “What Are Machine Learning Applications? Top 10 Use Cases.” *Emeritus Online Courses*, 19 Dec. 2022, https://emeritus.org/blog/machine-learning-what-are-machine-learning-applications /.

[48] “Data Science vs. Machine Learning: What’s the Difference? | Coursera.” *Coursera*, https://www.coursera.org/articles/data-science-vs-machine-learning .

[49] “Top Machine Learning Use-Cases and Algorithms.” *DataCamp*, https://www.datacamp.com/blog/top-machine-learning-use-cases-and-algorithms .

[50] “Data Science & Machine Learning: Role of ML in Data Science.” *Zuci Systems*, 21 Dec. 2021, https://www.zucisystems.com/blog/what-is-the-role-of-machine-learning-in-data-science/ .

[51] Kanade, Vijay. “Top 10 Machine Learning Algorithms.” *Spiceworks*, 5 May 2022, https://www.spiceworks.com/tech/artificial-intelligence/articles/top-ml-algorithms/ .

[52] Tavasoli, Simon. “Top 10 Machine Learning Algorithms You Need to Know in 2023 | Simplilearn.” *Simplilearn.Com*, Simplilearn, 9 Nov. 2016, https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article .

[53] “What Is a Decision Tree | IBM.” *IBM - United States*, https://www.ibm.com/topics/decision-trees .

[54] “Top 9 Python Libraries for Machine Learning in 2023 | UpGrad Blog.” *UpGrad Blog*, 3 Oct. 2022, https://www.upgrad.com/blog/top-python-libraries-for-machine-learning/ .

[55]“The 6 Python Machine Learning Tools Every Data Scientist Should Know About - KDnuggets.”, https://www.kdnuggets.com/2022/05/6-python-machine-learning-tools-every-data-scientist-know.html .

[56] Nederkoorn, Cordny. “Top 10 Python Packages for Machine Learning - ActiveState.” *ActiveState*, 27 Feb. 2020, https://www.activestate.com/blog/top-10-python-machine-learning-packages /.

[57] “Best Python Libraries for Machine Learning - GeeksforGeeks.” *GeeksforGeeks*, GeeksforGeeks, 18 Jan. 2019, https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/ .

[58] Quinn Radich. “What Is a Machine Learning Model? | Microsoft Learn.” *Microsoft Learn: Build Skills That Open Doors in Your Career*, https://learn.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model .

[59] “What Are Machine Learning Models?” *Databricks*, https://www.databricks.com/glossary/machine-learning-models .

[60] “Machine Learning Process - Overview” | Medium, *Medium*, https://medium.com/analytics-vidhya/machine-learning-process-overview-1daa05c30150 .

[61] Dr. N. Ananthi, August 29, 2021, "Cyber Bullying Types Datasets", IEEE Dataport, doi: https://dx.doi.org/10.21227/bsdy-zw62.

[62] “Cyber Troll Dataset | Zenodo.” *Zenodo*, 12 Feb. 2020, https://zenodo.org/record/3665663#.ZAHx2HZBxaQ .

[63] “Classified Tweets | Kaggle.” *Kaggle: Your Machine Learning and Data Science Community*, https://www.kaggle.com/datasets/munkialbright/classified-tweets .

[64] “Cyberbullying Classification | Kaggle.” *Kaggle: Your Machine Learning and Data Science Community*, https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification .

[65] J. Wang, K. Fu, C.T. Lu, “SOSNet: A Graph Convolutional Network Approach to Fine-Grained Cyberbullying Detection,” Proceedings of the 2020 IEEE International Conference on Big Data (IEEE BigData 2020), December 10-13, 2020.

[66] “Python (Programming Language) - Wikipedia.”,Wikimedia Foundation, Inc, https://en.wikipedia.org/wiki/Python_(programming_language) .

[67] “Pandas Introduction.” *W3Schools Online Web Tutorials*, https://www.w3schools.com/python/pandas/pandas_intro.asp .

[68] “Scikit-Learn - Wikipedia.” *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., https://en.wikipedia.org/wiki/Scikit-learn .

[69] “Pandas (Software) - Wikipedia.” *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., https://en.wikipedia.org/wiki/Pandas_(software) .

[70] “Matplotlib - Wikipedia.” *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., https://en.wikipedia.org/wiki/Matplotlib .

[71] “Python Seaborn Tutorial - GeeksforGeeks.” *GeeksforGeeks*, GeeksforGeeks, 8 Feb. 2021, https://www.geeksforgeeks.org/python-seaborn-tutorial/ .

[72] “Natural Language Toolkit - Wikipedia.” *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., https://en.wikipedia.org/wiki/Natural_Language_Toolkit.

[73] Meir, Dor. “The Good, the Bad and the DataSpell.” *Towards Data Science*, 16 Sept. 2022, https://towardsdatascience.com/the-good-the-bad-and-the-dataspell-a86fec8fd6e1 .

[74] Pryke, Benjamin. “How to Use Jupyter Notebook in 2020: A Beginner’s Tutorial.” *Dataquest*, 24 Aug. 2020, https://www.dataquest.io/blog/jupyter-notebook-tutorial/ .

[75] “GitHub - Wikipedia.” *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., 22 July 2008, https://en.wikipedia.org/wiki/GitHub .

[76] Narkhede, Sarang. “Understanding Logistic Regression.” *Towards Data Science*, https://towardsdatascience.com/understanding-logistic-regression-9b02c2aec102 .

[77] Navlani, Avinash. “Decision Tree Classification in Python Tutorial.” *DataCamp*, https://www.datacamp.com/tutorial/decision-tree-classification-python .

[78] Koehrsen, Will. “An Implementation and Explanation of the Random Forest in Python.” *Medium*, https://medium.com/towards-data-science/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76 .

[79] “XGBoost - GeeksforGeeks.” *GeeksforGeeks*, GeeksforGeeks, 18 Sept. 2021, https://www.geeksforgeeks.org/xgboost/?ref=lbp .

[80] "Naive Bayes Classifier in Machine Learning - Javatpoint." , https://www.javatpoint.com/machine-learning-naive-bayes-classifier .

[81] Gandhi, Rohith. "Support Vector Machine — Introduction to Machine Learning Algorithms." *Towards Data Science*, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 .

[82] Patel, Ashish. "Bagging — Ensemble Meta Algorithm for Reducing Variance | by Ashish Patel | ML Research Lab | Medium." *Medium*, ML Research Lab, 15 Aug. 2019, https://medium.com/ml-research-lab/bagging-ensemble-meta-algorithm-for-reducing-variance-c98fffa5489f .

[83] "Boosting- Ensemble Meta Algorithm for Reducing Bias | by Ashish Patel | ML Research Lab | Medium." *Medium*, ML Research Lab, 16 Mar. 2020, https://medium.com/ml-research-lab/boosting-ensemble-meta-algorithm-for-reducing-bias-5b8bfdce281 .

[84] "Pickle — Python Object Serialization — Python 3.11.2 Documentation." *Python Documentation*, https://docs.python.org/3/library/pickle.html .

[85] "Streamlit • The Fastest Way to Build and Share Data Apps." *Streamlit* , https://streamlit.io/ .

[86] "HTML5 - Wikipedia." *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., https://en.wikipedia.org/wiki/HTML5 .

[87] "CSS - Wikipedia." *Wikipedia, the Free Encyclopedia*, Wikimedia Foundation, Inc., https://en.wikipedia.org/wiki/CSS .

[88] "Streamlit Community Cloud - Streamlit Docs." *Streamlit Documentation*, https://docs.streamlit.io/streamlit-community-cloud .

[89] Khanna, Chetna. "Text Pre-Processing: Stop Words Removal Using Different Libraries." *Towards Data Science*, https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a .

[90] “Stemming vs Lemmatization in NLP: Must-Know Differences.” *Analytics Vidhya*, 28 June 2022, https://www.analyticsvidhya.com/blog/2022/06/stemming-vs-lemmatization-in-nlp-must-know-differences/ .

[91] Gillis, Alexander S. , “What Is Data Splitting and Why Is It Important?”, *Enterprise* https://www.techtarget.com/searchenterpriseai/definition/data-splitting .

[92] “Understanding TF-ID: A Simple Introduction.” *MonkeyLearn Blog*, 10 May 2019, https://monkeylearn.com/blog/what-is-tf-idf/ .

[93] “Using CountVectorizer to Extracting Features from Text - GeeksforGeeks.” *GeeksforGeeks*, 15 July 2020, https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/ .

[94] Zadane, Mayura. “Model Evaluation Metrics for Machine Learning Algorithms - Knoldus Blogs, https://blog.knoldus.com/model-evaluation-metrics-for-machine-learning-algorithms/#classification-metrics .