BACHELOR'S DEGREE THESIS

Degree in Statistics

Title: Identification of texts generated by AI (ChatGPT)

Author: Yaiza Bravo de Dios

Advisors: Jordi Cortés Martínez and Daniel Fernández Martínez

Department: Department of Statistics and Operations Research

Academic year: June 2023

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Acknowledgments

First and foremost, I would like to express my gratitude to Jordi and Dani for their dedication and hard work throughout this phase, which has not been easy. They have helped me believe in myself more and have made this project possible.

Secondly, I would like to thank my family, friends, and partner for being by my side on this journey over the past four years. Today, I can confidently say that it has all been worth it.

Lastly, I want to extend my appreciation to the Universities and, most importantly, the professors who have guided me when I felt most lost. Their support has allowed me to learn and grow in the best possible company.

Summary and key words

Abstract: This bachelor's final project aims to address the challenge of identifying texts generated by artificial intelligence (AI) systems. The project implements several classification models, including Classification Trees, Random Forests, Logistic Models, and Support Vector Machines to identify AI-generated texts. These models are trained on a dataset which consists of 160 texts of human and AI-generated texts, with the goal of accurately distinguishing them. The project also includes the implementation of a Shiny application, providing a user-friendly interface for text identification. Among the models evaluated, the Logistic Model achieves the highest accuracy, with 86%.

Keywords: Text identification, artificial intelligence (AI), classification models, Classification Tree, Random Forest, Logistic Model, Support Vector Machines (SVM), Shiny.

Resum i paraules clau

Títol: Identificació de textos generats per IA (ChatGPT)

Resum: Aquest treball final de grau té com a objectiu abordar el repte d'identificar textos generats per sistemes d'intel·ligència artificial (IA). Amb l'auge de l'IA, és cada vegada més important poder distingir entre textos generats per humans i textos generats per màquines. El projecte es centra en el desenvolupament i avaluació de models de classificació, incloent arbres de classificació, boscos aleatoris, models logístics i màquines de vectors de suport. Aquests models s'entrenen amb un conjunt de dades de 160 textos generats per humans i per IA, amb l'objectiu de distingir-los amb precisió. El projecte també ha inclòs la implementació d'una aplicació Shiny que proporciona una interfície d'usuari intuitiva i amigable permetent als usuaris identificar fàcilment si un text és generat per un humà o per un sistema d'IA. Això facilita la utilització dels models desenvolupats i fa accessible la identificació de textos per a un ampli ventall d'usuaris. En resum, aquest treball es concentra en el desafiament d'identificar textos generats per intel·ligència artificial, utilitzant una varietat de models de classificació i obtenint com a millor model el logístic, amb una precisió del 86%. I, a més a més, proporciona una eina Shiny per a una interfície d'usuari comprensible en la identificació de textos generats per IA.

Paraules clau: Identificació de text, Intel·ligència Artificial (IA), models de classificació, Arbre de Classificació, Bosc Aleatori, Model Logístic, Màquines de Vectors de Suport (SVM), Shiny.

American Mathematical Society

62-07 – Data analysis (statistics)

62H30 – Classification and discrimination; cluster analysis

68T45 – Machine vision and scene understanding

68T50 – Natural language processing

Table of contents

List of figures

List of tables

1. INTRODUCTION

1.1 Background

The world has undergone a profound change in the last few decades, with technology playing a central role in shaping the way we live, work, and learn. Education has been no exception, and the COVID-19 pandemic has changed a towards online learning which had been rapidly taking place. This sudden growth has presented many challenges, but it has also opened up new opportunities for innovation and growth.

Higher Education Institutions (HEI) have had to quickly adapt to the challenges posed by the pandemic by shifting to online classes and exams. Despite encountering challenges, it is unlikely that this trend towards online education will reverse soon, as both HEIs and students have come to recognize and appreciate the benefits of remote learning [1]. As mentioned earlier, the COVID-19 pandemic has drastically changed our daily lives and has had a significant impact on various aspects of society, including education. This fact has exposed the inequalities in our educational systems and has highlighted the importance of technology in education.

One of the key factors that has made this transition possible is the rapid development of artificial intelligence (AI) and its applications in various fields, counting education [2]. The ability of AI systems to process and learn from large amounts of data has created new opportunities for personalized learning, automated assessment, and more efficient educational practices. AI has revolutionized the way we approach text generation. From predictive typing to advanced language models, AI algorithms are now capable of producing coherent and meaningful text with remarkable correctness. Even as we look to the future, the role of AI in education will continue to grow, offering new solutions to the challenges faced by educators and students alike. This work aims to explore the field of AI-generated text, analyzing its capabilities to generate human-like texts. As we navigate this unprecedented time, it is crucial to understand the role of technology in shaping the future of education and communication.

ChatGPT is a large language model (LLM) created by OpenAI [3], an AI company in San Francisco. With its ability to generate human-like text, ChatGPT has the potential to revolutionize the way we interact with technology. However, as with any new system, there are also concerns about the ethical implications of AI-generated text, particularly in terms of misinformation and bias. It can produce sophisticated writing on a wide range of topics after being trained on a massive data set of text. It has caused excitement and controversy for its ability to converse with users in English and other languages. This technology is being used for writing essays, talks, summarizing literature, improving papers and more, and is likely to revolutionize research practices and publishing. Nevertheless, the use of ChatGPT and other LLMs may also degrade the quality and transparency of research and spread misinformation. There are two significant challenges facing ChatGPT currently. The first one is that it sometimes makes mistakes. It may provide incorrect answers, but since it does so with a high-level argumentation, it is very difficult to detect its errors if the device does not already know what is entered. The second challenge is that it does not reveal the sources of information it uses to respond, making it difficult to verify its claims or delve deeper into points that are of interest to the user. In the near future, this technology may even advance to the point of being able to design experiments, complete manuscripts, conduct peer review, and support editorial decisions in the publishing process.

The identification of texts generated by AI or human sources has become increasingly challenging as AI-generated outputs of large language models have become indistinguishable from human-generated outputs [4]. This presents a new challenge for education assessment, as it can be difficult to determine which texts have been created by AI and which have not. Recent research in this area has focused on various approaches to identify the source of a text. Nevertheless, research suggests that there are linguistic markers that can separate AI-generated text from human-generated text. For instance, a study by Markowitz, Hancock, and Bailenson (2023) found that AI-generated text tends to be less emotional and less analytic than human-generated text [4]. However, the use of ChatGPT and other similar models will make it increasingly difficult for journal editors to discriminate between human-written and AI-generated texts [5]. Additionally, research suggests that AI-generated texts do not differ from human-written texts in their perceived credibility or trustworthiness where simple and short text types are concerned. It is unclear how AI-written texts beyond simple fact reporting are perceived. Therefore, further research is needed to expand upon the existing literature on automated journalism by investigating the influence of AI authorship on the perception of texts [6]. In reference to creative domains, the credibility of AI-generated content has been questioned. Nevertheless, a study by Gunser et al. (2022) found that readers cannot differentiate between AI-based and human-written texts in terms of subjective credibility and stylistic quality [7]. Despite these advances, there is still much room for improvement in the field of text generation detection: some AI models can generate human-like text that can be challenging to detect. The ethical implications of AI-generated content in the business arena are still being debated [8–10].

In conclusion, although AI-generated text has become highly advanced, there could be still linguistic characteristics that can differentiate it from human-written text. However, more research is required to investigate the impact of artificial intelligence authorship on the perception of text beyond factual reporting because readers can’t distinguish between AI-generated and human-written text in terms of subjective credibility and stylistic quality.

Additionally, statistical methods have shown promise in identifying linguistic patterns and features that distinguish human-generated text from AI-generated text, which could prove to be a useful tool for future research in this field.

1.2 Objectives

General Objective:

This final project proposal aims to accurately distinguish between texts written by a human and those generated by the ChatGPT text generation tool. The process involves collecting a sample of texts from both sources, extracting relevant features, and training classification algorithms using the collected data.

Specific Objectives:

- Estimate several indicators of predictive performance of our algorithm in detecting texts generated by AI, such as accuracy, sensitivity, specificity, positive predictive value, and negative predictive value.
- Identify the key factors that differentiate human and AI-generated texts.
- Compare the predictive performance of our tool with existing state-of-the-art methods for text generation detection by means of the AUC or the accuracy.
- Develop a user-friendly interface to allow easy use of the tool by non-technical users (shiny).

This project is divided into three distinct parts:

1. The first step involves the collection, selection, and processing of text data, which is the most time-consuming and labor-intensive part of the process. Text mining techniques will be used to extract relevant features from each text, which will then be used in the classification model.
2. In second place, comprehensive research of relevant predictive variables/factors to distinguish between human and AI texts will be conducted using reliable sources such as scientific articles, news, interviews, and other trustworthy sources.
3. Finally, once trained, the model will be evaluated to determine the predictive performance measures in identifying the source of the text.

Another crucial aspect of the project was the process of writing. The writing phase was conducted concurrently with the entire work period. This allowed for a continuous flow of progress and enabled us to report on the findings and process as we went along. By doing so, we ensured better organization and structure of the research, which ultimately helped us to achieve our goals more efficiently.

This also allowed us to identify any potential issues or challenges that may have arisen during the research process and address them promptly, resulting in a smoother and more successful outcome.

Regarding the structure of this work, it comprises four chapters:

1. Introduction. It includes the justification and motivation of the topic, the state of the art, and the objectives.

2. Methodology. It includes the set of procedures and techniques used to carry out a study, divided in three parts:

   2.1. Collection, selection, and text mining.

   2.2. Predictive variables and typification of the bibliographical review.

   2.3. Statistical analysis: exploratory data analysis and classification models.

3. Results. The findings derived from the analyses are presented by means of tables and auto-explicative figures.

4. Conclusions. This chapter establishes a close relationship with the topics discussed throughout the study, thus providing answers to the objectives initially set.

2. METHODOLOGY

2.1 Collection, selection, and processing of texts

One of the main goals of this project was to gather, select, and process a set of texts that would make up the sample of the database. To achieve this, a two-part process was implemented.

Initially, a thorough text search was conducted, which was divided into two stages. In the first one, between the 2nd and the 20th of February 2023, a pilot test was conducted using a sample of 20 texts. They were set of pairs of related texts, usually with similar themes or topics, that were used for comparative analysis or evaluation from both human and artificial intelligence sources. These texts were classified as definitions, interviews, and articles. See next section for details on the text typology on statistical and non-statistical topics.

In the second stage, between the 12th and the 19th of March 2023, the search and collection of texts were expanded to obtain a larger sample of the same text classifications as in the pilot study, with the addition of a new one: descriptive book reviews. This resulted in a total of 160 files: 80 human-generated and 80 AI-generated texts. The human-generated texts came from several publication years ranging from 1998 to 2023 (see Figure 2.1). It is important to note that all the information provided by ChatGPT was obtained from the current year in which the study was being conducted (2023). Although the information from ChatGPT has been obtained this year, its sources are only updated until 2021.

Before conducting any statistical analysis, a thorough review was conducted of the texts to ensure that no errors were introduced during the process of copying them to a text (.txt) format. Specific attention was given to checking for errors in spaces, punctuation, and paragraph breaks, as these factors could have a significant impact on the results of the analysis, particularly with respect to variables such as the number of paragraphs in each text.

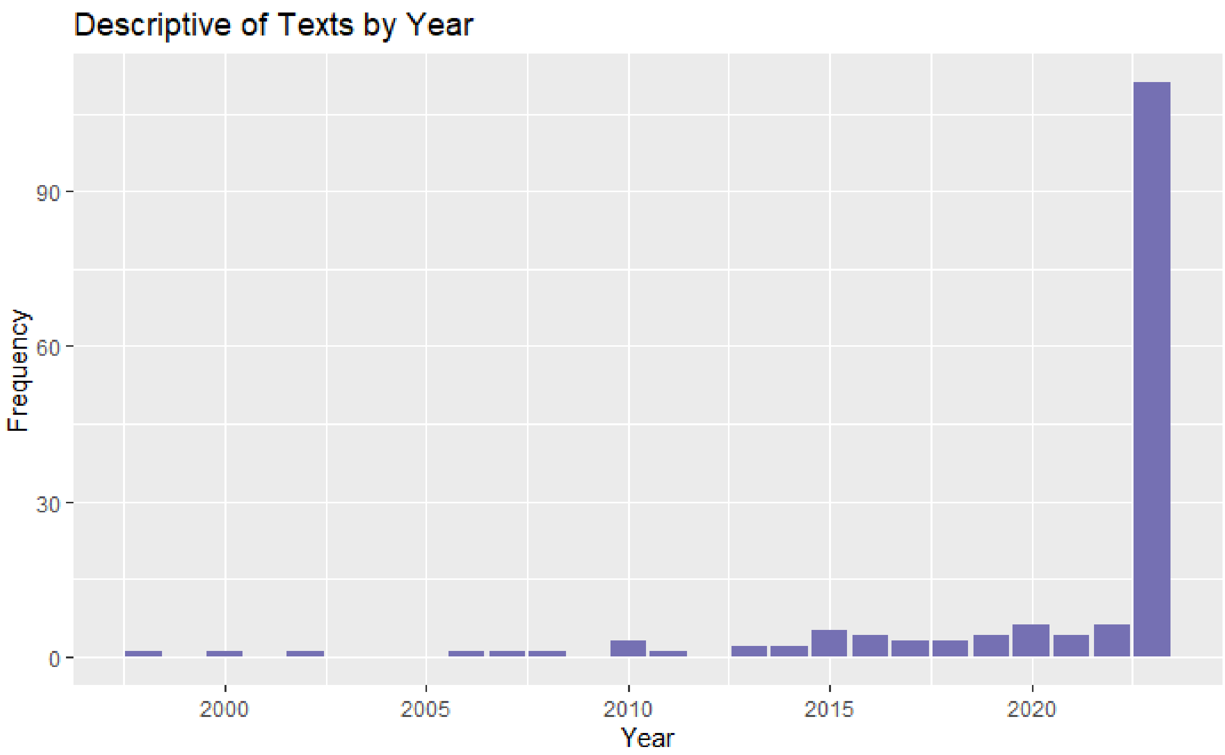


Figure 2.1: Distribution of texts across different years

2.1.1 Text typology

The database samples are paired, consisting of a text generated by a human source and its equivalent generated by an artificial intelligence (ChatGPT), both asked to provide exactly the same information. These texts are divided into different categories, which are described below.

Definition: a statement that explains the meaning of a word or term. It typically provides clarity and understanding by describing the characteristics or essential features of the concept or object. A definition can be a formal statement found in a dictionary or a more informal explanation provided in conversation or writing. The purpose of a definition is to convey the precise meaning of a word or term so that it can be understood correctly and accurately used in communication.

For the collection of human-source definitions, the primary source of information was Wikipedia, as it provides reliable, well-organized, and clear information. On the other hand, to obtain their corresponding artificial intelligence-generated counterparts, we wrote to the ChatGPT: "*Give me the definition of (...)*" followed by the respective topic.

Interview: a conversation between two or more people, typically between an interviewer and an interviewee, where the former asks questions to obtain information from the latter.

Interviews are often used as a research method in social sciences, journalism, and human resources to gather data and insights from individuals with specific knowledge, experience, or perspectives. The interviewer may use various types of questions, such as open-ended questions that allow the interviewee to provide more detailed responses or closed-ended questions that require yes or no answers. Interviews can be conducted in person, over the phone, or online, and they may be recorded, transcribed, or analyzed qualitatively or quantitatively depending on the purpose of the interview.

To find texts of this type, direct transcriptions of interviews in mostly journals were searched. To obtain an equivalent response with ChatGPT, the exact same question asked in the human source interview was posed to it, in order to obtain the corresponding result (e.g. "*Why are businesses using Big Data for competitive advantage?").* Personal questions were excluded.

Article: a written piece of non-fiction prose that is usually published in a newspaper (e.g. The New York Times, *La Vanguardia*), journal (scientific or not) (e.g. Science), or online publication [11, 12]. Articles can be on a wide range of topics, from news and current events to analysis and opinion pieces. They are typically written by journalists or subject matter experts and are intended to inform, entertain, or persuade readers. Articles can vary in length, style, and format, but they typically follow a journalistic structure that includes a headline, byline, lead, body, and conclusion. The style of writing in them can also vary, ranging from objective and informative to subjective and opinionated.

To carry out the search for this type of text, the state of the art of the topic of the articles was mainly extracted, and then we wrote to ChatGPT "*Give me the state of the art of the topic (...)*" and the corresponding topic for its equivalent. Also, in some sections of some articles, a more specific topic was discussed and to find its counterpart with ChatGPT, the title of the section was copied, usually being a question such as "*What causes inflation?*".

Book review (descriptive): a type of review that summarizes and describes the content and style of a book. The review aims to provide readers with an understanding of what the book is about, including the main themes and arguments presented by the author, as well as the writing style and overall structure of the book. The reviewer may also comment on the author's use of language, the effectiveness of the book's organization, and the impact of the book's ideas. The purpose of a descriptive book review is to provide a fair assessment of the book's content, not to express a personal opinion or judgment about the book's quality or value.

The two main human sources of information for finding book reviews were Amazon [13] and Goodreads [14], as their descriptions were quite comprehensive. The selection of books was made without any specific order or preference, and an effort was made to cover different themes as well. To find the equivalent book review generated by artificial intelligence, the prompt

"*Write a book review for (...)*" followed by the book's title and the author's name were used to request it from the ChatGPT.

For this text search, there were two distinct topic types: statistical and non-statistical. Both fields were very broad, and that was also the goal in order to obtain the greatest possible variability of texts for analysis. Some examples of statistical topics include cluster analysis, marketing in statistical studies, the role of statisticians, sampling methods, survival analysis, machine learning, analysis of variance, causal inference, non-parametric statistics, among others. And some examples of non-statistical topics include food innovation, film noir, art deco, neuroscience, the Ukraine war, color psychology, mental disorders, poverty, political corruption, music therapy, among others.

Our sample is fully balanced. Out of a total of 160 texts, the sample is divided as follows: 80 texts generated by artificial intelligence, with 10 texts of each type (statistical definition, non-statistical definition, statistical interview, non-statistical interview, statistical article, non-statistical article, statistical review, and non-statistical review), and the same for the other 80 texts generated by a human source. We can observe its diagram in Figure 2.2.

Sample text classification

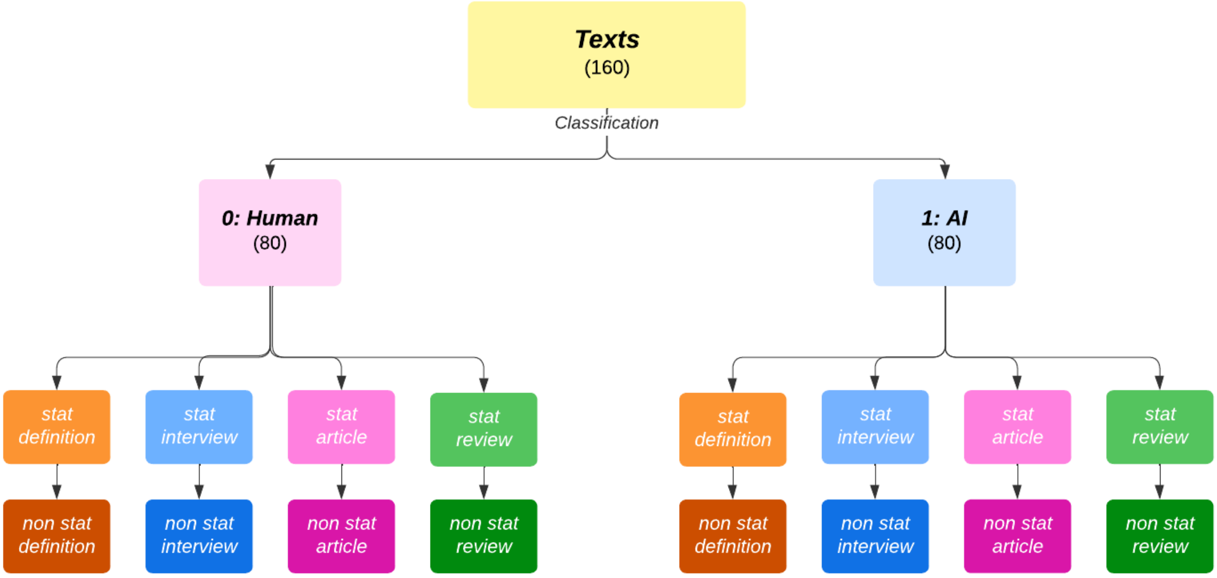


Figure 2.2: Classification of the data base texts

2.2 Predictive variables and typification of the bibliographical review

The objective of this part of the project was to identify a set of variables that would aid us in conducting the proposed study later. These variables needed to contribute to text analysis and be useful in deciphering whether a text was generated by a human source or artificial intelligence.

Firstly, an initial search was conducted on the topic of ChatGPT to gain basic knowledge and an understanding of the state of the art. The keywords used to find related articles were "*ChatGPT*", "*chatbot*", and "*How to identify texts generated by AI*". The main search engine used was Google Scholar and as relevant material was collected, articles were summarized and selected for the study.

Secondly, after initially familiarizing ourselves with the subject matter, our goal was to gather and curate enough bibliographic material, which was then analyzed in greater depth to compile a collection of various factors that could prove useful for the project. As we discovered potential variables for study, they were added to the compilation. In addition, any ideas that arose as a result of the articles were also taken into consideration. It was not an easy task, as it is a new and constantly changing field where a solid hypothesis has not yet been found. From the beginning, it was clear that a quantitative text analysis had to be carried out, as it was fundamental to begin to find differences between the two types of text sources. Furthermore, as the reading of various articles progressed, several ideas could be gathered for the study. Some factors such as the sentiment/emotion of the text, creativity, formal language, style and tone, and coherence were considered and quantified, which also triggered other potential proposals. [15]–[18].

Finally, in order to verify the feasibility of obtaining these possible variables, a search of R packages was conducted that would allow for the measurement, calculation, or analysis of the suggested factors. The list of R packages [19] used in this project can be found in the Appendix 6.2. Then, it was noted that not all the proposed variables were useful, as not all of them could be analyzed as easily as creativity or semantic errors. As a result, a final selection of variables was made and will be presented below.

2.2.1 Variables

In our database, we have 6 categorical nominal variables (see Table 2.1). These ones represent distinct categories or levels without any inherent order or numerical value. Each variable captures qualitative information that allows us to classify observations into specific groups or classes.

Categorical variables

Table 2.1: List of categorical nominal variables with their respective information

Variable	**Definition**	**Type**	**Response**
id	file’s identifier	nominal	HU_n AI_n
outcome	takes on the value of 0 if the text is generated by a human and 1 if it is generated by AI	binary	0 1
typo	category or class to which the text belongs	nominal	definition interview article review
type	indicates whether the text pertains to statistical or non-statistical topics	binary	statistical non statistical
source	URL of the original text if it is written by a human or ChatGPT, otherwise	nominal	ChatGPT URL
year	publication year if the text is written by a human or 2023, otherwise	nominal	1998-2023
pattw	contains the word with the highest frequency in each text	nominal	and the of others

Furthermore, we have 4 numeric discrete variables that play a significant role in our analysis (see Table 2.2). These variables are categorized as discrete because they represent distinct and separate values rather than continuous measurements. Each variable captures specific aspects of the data and contributes valuable information for our analysis.

Discrete variables

Table 2.2: List of discrete variables with their respective definitions

Variable	Definition
lengthl	number of letters of each text
lengthw	number of words of each text
pgf	number of paragraphs of each text
nstc	number of sentences of each text

Finally, in our database we have 18 numeric continuous variables that play a crucial role in our analysis (see Table 2.3). These variables are continuous because they represent measurements on an interval scale.

Continuous variables

Table 2.3: List of continuous variables with their respective definitions.

Variable	Definition
wordl	number of characters per word in each text
sent	vector with the sentiment scores (from -1 (most negative) to +1 (most positive) for each word, with 0 indicating a neutral sentiment) for the input text for each sentence
sentm	vector's mean value of sent
sentsd	vector's standard deviation value of sent
patt	maximum frequency of any word, standardized by dividing it by the total number of words in the text
pron	number of pronouns in each text, standardized by dividing it by the total number of words in the text
art	number of articles (determiners) in each text, standardized by dividing it by the total number of words in the text
voc	number of unique words that appear more than five times in each text, standardized by dividing it by the total number of words in the text

Variable	**Definition**
form	number of informal language forms (e.g., gonna, wanna, kinda, lotta, etc.) in each text, standardized by dividing it by the total number of words in the text
aggr	number of occurrences of words related to aggressive language (e.g., hate, attack, angry, war, violence, aggression) in each text, standardized by dividing it by the total number of words in the text
avw	average number of words per sentence in each text
spgf	average number of sentences per paragraph in each text
wpgf	average number of words per paragraph in each text (it is a linear combination of the two previous variables)
nstcw	ratio of the number of sentences to the total number of words in the text
pgfw	ratio of the number of paragraphs to the total number of words in a text
freqv	standard deviation of the frequencies of each word in each text. It assesses how scattered the word frequencies are in each text
typew	relationship between the number of unique word types and the total number of words in the text, it represents the ratio of unique words with a frequency greater than 5 to the total number of words in each text
sumrep	sum of duplicated words for each text, standardized by dividing it by the total number of words in the text
and	number of *and* words in each text, standardized by dividing it by the total number of words in the text
the	number of *the* words in each text, standardized by dividing it by the total number of words in the text

2.3 Statistical analysis

In this section, classification models that have been applied in the R statistical software [20] to verify the tool's ability to predict the question at hand will be depicted. Before applying the classification models, an Exploratory Data Analysis (EDA) will be conducted to gain insights into the distribution of variables and the relationships between them, described below.

Throughout the project analysis, the RStudio IDE (Integrated Development Environment) has been utilized for implementation [21].

2.3.1 Exploratory data analysis

Once we have obtained the database and variables for the study, our first step was to summarize the main characteristics of our information using both statistical measures and graphical statistical tools.

On one hand, for categorical variables, contingency tables and bar charts are mostly used. On the other hand, for numeric variables, different summary descriptive statistics such as maximum, minimum, mean, standard deviation, and quartiles are used along with boxplots. Furthermore, for bivariate descriptive analysis, the Chi-square test is applied for assessing the independence between categorical variables and the text source (human or AI generated), while the Student's t-test is used for comparing numeric variables between categories of text typology.

The Chi-square test is a statistical test used to assess independence between two categorical variables. However, in some cases, the data may not meet the necessary assumptions for applying the Chi-square test, such as when the sample size is small or when there are very low frequencies in some cells (specifically, the expected counts under independence in each cell should be greater than 5). In these cases, Monte Carlo simulation (implemented in our analysis) can be used to obtain an approximate p-value. The idea behind this simulation is to generate random samples from a probability distribution and calculate the test statistic of interest for each of them. By repeating this process many times, an approximate distribution of the test statistic can be obtained and hence, an approximate p-value.

On the other hand, the t-test is typically used to compare the means of two groups when the variable of interest is numeric. It should also be noted that we have considered that the t-test requires the underlying populations to follow a normal distribution, and that if this assumption is not met, another test such as the Mann-Whitney-Wilcoxon should be used instead. Nevertheless, as the sample size increases, the t-test becomes less sensitive to violations of this assumption. Thus, when we have a large sample, we can use the t-test for all our numerical variables, thanks to the Central Limit Theorem.

2.3.2 Classification models

In order to conduct the analysis for this project, four classification algorithms have been implemented: Classification Trees (CT), Random Forest (RF), Logistic Regression (LR), and Support Vector Machines (SVM).

Among the models mentioned, CT, RF, and SVM are non-parametric models, while LR is a parametric model. These latter make certain assumptions about the underlying data distribution and have a fixed number of parameters. In contrast, non-parametric models do not make strong assumptions about the data distribution and can adapt to more complex patterns without a fixed number of parameters.

Decision Tree Analysis

In statistics, decision trees are classified into two main types:

- **Classification tree analysis** predicts the discrete class to which the data belongs (our case).
- **Regression tree analysis** predicts a real number outcome, such as the price of a house or a patient's length of stay in a hospital.

The term classification and regression tree (CART) analysis encompasses both types of procedures. While trees used for regression and classification share some similarities, there are also variations, having as a fundamental difference the type of response variable.

The **Gini index** [22] is a measure used in decision tree algorithms to evaluate the impurity or heterogeneity of a node in a classification tree. It quantifies the likelihood of misclassifying a randomly chosen element in the dataset if it were randomly labeled according to the distribution of classes in that node. Mathematically, the Gini index is calculated by summing the squared probabilities of each class in the node:

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2, \tag{1}$$

where, p_i is the probability of an object being classified to category i.

The Gini index ranges from 0 to 1, where 0 indicates a pure node (all elements belong to the same class) and 1 indicates maximum impurity (elements are evenly distributed among classes). When constructing a decision tree, the algorithm evaluates different splitting points based on the Gini index to find the one that maximally reduces the impurity in the resulting child nodes. By iteratively splitting the data based on different features and thresholds, the decision tree seeks to create pure nodes and make accurate predictions. In summary, it is used in decision trees to assess the impurity of nodes and guide the splitting process towards creating more homogeneous subsets of data.

Classification Trees

The CT is a machine learning algorithm utilized in the classification of remotely sensed and non-essential data, aiding land cover mapping and analysis. It takes the form of a structural map consisting of binary decisions that ultimately determine the class or interpretation of an object, such as a pixel. The Classification Tree Analysis involves an analytical procedure that constructs a decision tree based on measured attributes like reflectance, utilizing known examples of classes, referred to as training data.

Classification Tree

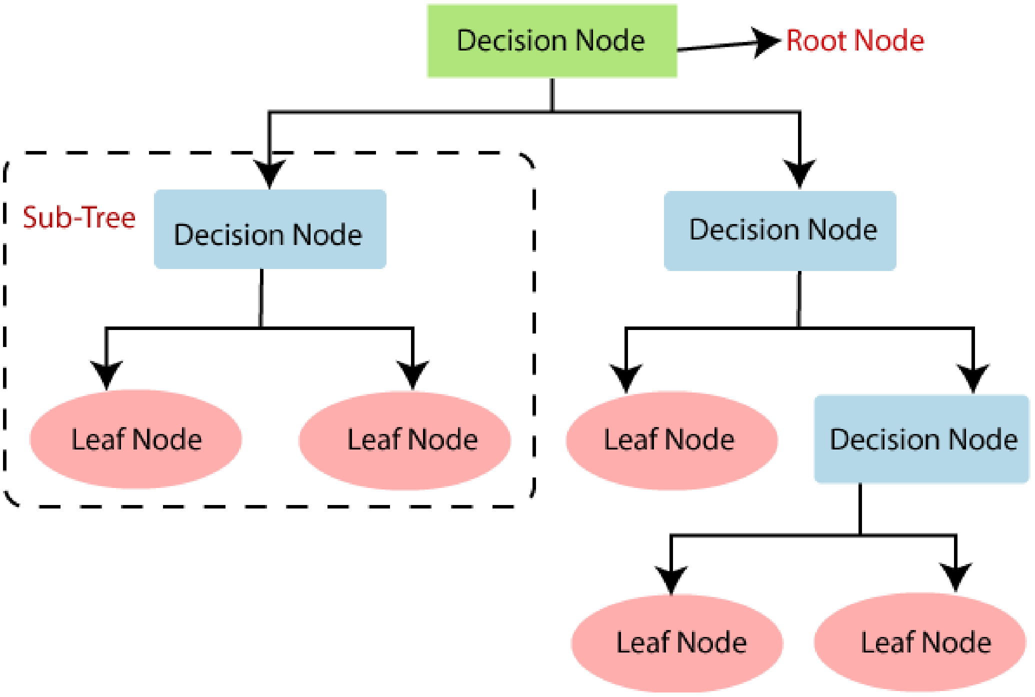


Figure 2.3: Visual representation of the CT algorithm
source: Javatpoint

A CT consists of branches that represent attributes, and leaves that represent decisions. During use, the decision process begins at the trunk and follows the branches until a leaf is reached. The Figure 2.3 shows a simple decision tree based on the red and infrared reflectance of a pixel. It comprises leaf nodes, which represent the final outcomes or classifications, and a root node, which serves as the starting point of the tree. Additionally, there are decision nodes within the tree that contain conditions or rules. These ones play a crucial role in the classification process as they determine how the data is split into different branches. By considering specific features or attributes of the data, the algorithm selects the most appropriate attribute, such as a reflectance band, and a corresponding value that effectively divides the samples into two groups. This division aims to minimize the variability within each subgroup while maximizing the contrast between the groups, facilitating accurate classification.

To build the Classification Tree model, we utilized the *caret* package of R [23]. This R package provides a convenient method to search for optimal tuning parameters for the decision tree model, thereby enhancing its predictive accuracy. For nonparametric methods, one of the requirements is to have a large sample size (n). In our case, the sample size is not sufficiently large, and for this reason, we validate the results using cross-validation. This approach not only enhances robustness but also provides a reliable assessment of the model's performance.

The *cp* parameter, short for complexity parameter, is a tuning parameter used in classification trees. It controls the complexity of the tree by specifying the minimum cost complexity required for a split to occur.

In CT, the goal is to find an optimal balance between tree complexity and accuracy. A smaller value of the *cp* parameter results in a more complex tree with more splits and potentially better accuracy on the training data. However, an overly complex tree may lead to overfitting, where the model fits the training data too closely and performs poorly on unseen data. By adjusting the *cp* parameter, we can control the trade-off between model complexity and performance. A larger value of *cp* promotes simpler trees with fewer splits, reducing the risk of overfitting. It helps to prune the tree by removing branches that do not significantly improve the overall predictive accuracy.

Random Forest

Random Forest [24, 25] is a versatile machine learning algorithm used for various tasks, including classification and regression. It operates as an ensemble learning method by building multiple decision trees during training. In classification tasks, the Random Forest selects the class that is chosen by the majority of trees, while in regression tasks, it returns the average prediction of individual trees. Figure 2.4 schematically illustrates the functioning of the algorithm.

One of the primary motivations behind using RF is to overcome the overfitting problem commonly encountered with decision trees. The algorithm employs a technique called bagging (or bootstrap aggregating), which involves creating multiple samples by randomly drawing data with replacement from the training dataset. Each sample is then used to construct a decision tree. Additionally, feature randomness, also known as feature bagging, is utilized to introduce diversity by selecting a random subset of features for each tree. This helps reduce correlation among the trees and enhances the overall performance of the Random Forest.

The Random Forest model can be applied to effectively solve regression or classification problems (our case). During the training process, a collection of decision trees is created, with each tree constructed from a data sample drawn from the training set with replacement. To assess the performance and validate the model, a technique called out-of-bag (oob) sampling is employed, where one-third of the training samples in average are set aside from each tree as the oob sample. Depending on the specific task at hand, the predicted output of the algorithm can be determined through either averaging the predictions of individual trees or selecting the most frequent category for classification.

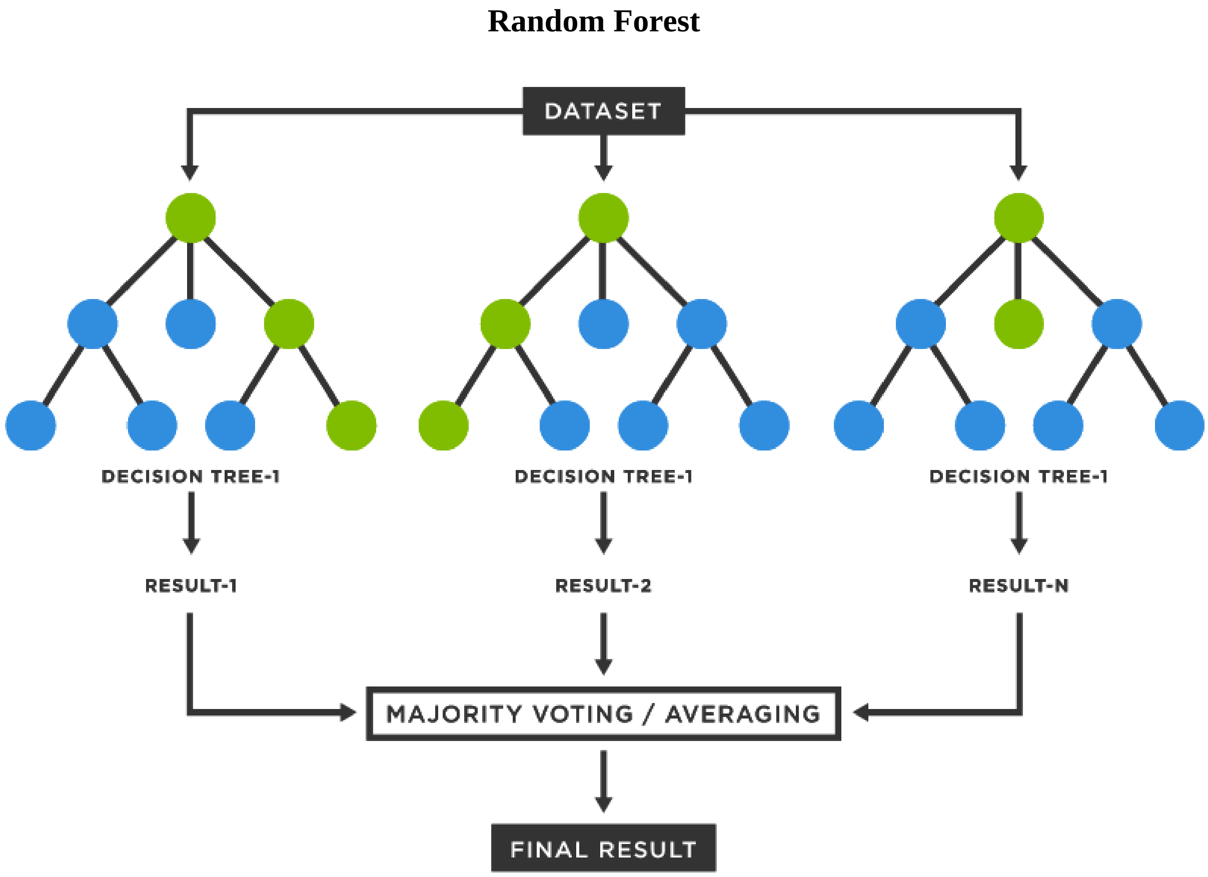


Figure 2.4: Visual diagram illustrating the functioning of the Random Forest algorithm
source: Kaggle

To fit the Random Forest model, we utilized the *caret* and *randomForest* [24] packages in R.

Before training the model, three key hyperparameters need to be set: the node size (the minimum number of data points required to create a new split in a tree which helps control the complexity of the tree and prevents overfitting), the number of trees in the ensemble (the total number of decision trees that are generated in the random forest algorithm), and the number of features sampled (the number of predictors randomly selected at each split when constructing a tree in a random forest). The default value of the node size is typically 1, which means each terminal node can contain only one observation and we specified *tuneLength = 5* to search for optimal tuning parameters, allowing us to fine-tune the model's performance.

Furthermore, the default value in classification Random Forest for the *mtry* hyperparameter is the square root of *p*, where *p* represents the number of variables in the dataset. This means that, automatically, each decision tree in the Random Forest model will consider a randomly selected subset of variables equal to the square root of the total number of variables. The purpose of the *mtry* parameter is to introduce randomness and reduce the correlation among decision trees within the ensemble. By randomly selecting a subset of features at each split, the algorithm ensures that different trees consider different sets of predictors and this helps capture diverse patterns and reduces the risk of overfitting. To find the optimal value for the *mtry* parameter, techniques such as cross-validation or grid search can be employed. These methods evaluate the performance of the model using different values of the hyperparameter and select the ones that yield the best results on unseen data. It is important to note that when using the train function in the context of Random Forest, the formula is converted into a model matrix, which includes the predictor variables. If there are factor variables in the model, they are expanded into dummy variables. As a result, the total number of columns in the model matrix, including these dummy variables, may exceed the number of original predictor variables. Therefore, the *mtry* value may reflect the expanded feature space rather than just the number of original predictors. In summary, the hyperparameter controls the number of predictors considered at each split in Random Forest.

Logistic Regression

Logistic Regression [26, 27] is one of the most popular statistical models used in supervised learning. It is primarily used for predicting binary dependent variables based on a given set of independent variables and unlike linear regression, which is primarily used for solving regression problems, Logistic Regression is specifically designed for solving classification problems. It predicts the probability of a dichotomous outcome belonging to a particular class. To model the relationship between the independent variables and the probability of the outcome, LR fits an "S"-shaped logistic function (see Figure 2.5). This function represents the likelihood of an event occurring, such as determining whether a given text was generated by a language model or not based on certain linguistic features.

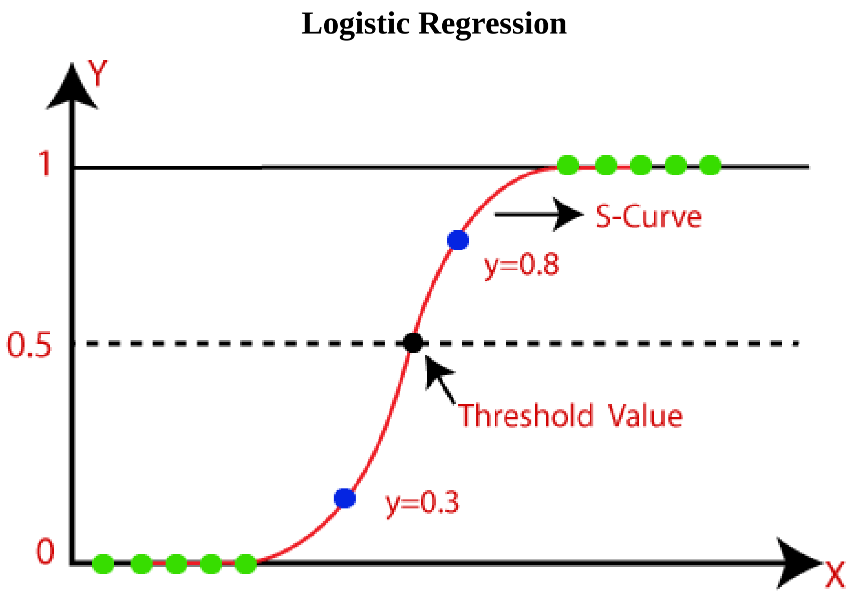


Figure 2.5: Logistic function outline. The value of the threshold does not necessarily need to be 0.5 source: Javatpoint

To transform the outcome of the model to a scale that can take any real value, Logistic Regression applies a logit transformation on the odds, which is the ratio of the probability of success to the probability of failure. The logit transformation, also known as the log odds or natural logarithm of odds, can be expressed as:

$$Logit(p) = \ln\left(\frac{p}{1-p}\right), \tag{2}$$

where p is the probability of success.

The logistic function is then applied to the logit transformation, resulting in the Logistic Regression equation:

$$p(x) = \frac{1}{1+e^{-\frac{x-\mu}{s}}}, \tag{3}$$

where μ is a location parameter (the midpoint of the curve, where $p(\mu) = \frac{1}{2}$) and s is a scale parameter.

Linear Model vs Logistic Model

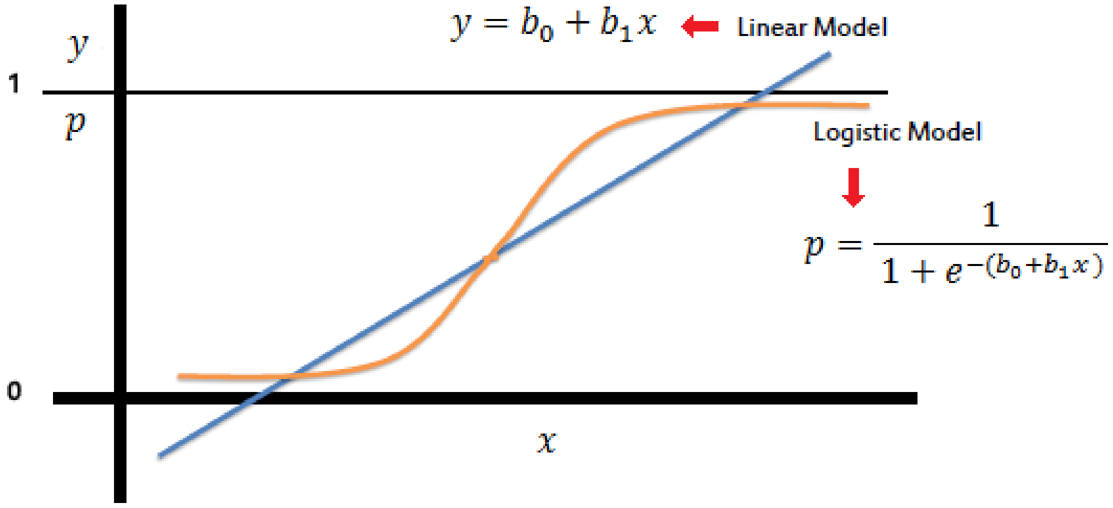


Figure 2.6: Contrast between the linear and the logistic models
source: Medium

In Figure 2.6, a comparison between the Linear Regression model and the Logistic Regression model is presented. The first one is commonly used for predicting continuous outcomes, while the second one is specifically designed for binary classification tasks. The Logistic Regression model demonstrates its capability to estimate the probability of an event occurring based on independent variables. By applying a logit transformation, it maps the outcome to a suitable scale for analysis and facilitates effective classification of new data. This comparison highlights the versatility of Logistic Regression in handling various types of data and its ability to identify influential variables for classification purposes.

In our case, when performing Logistic Regression, since we have a large number of variables and few events relative to the number of variables, we will only select the most relevant ones determined by the CART analysis. Also, for the variables *sumrep* and *and*, they will be scaled to percentages to obtain a better interpretation of the odds ratios.

In addition, in case we encounter convergence issues due to perfect separation problem, we will use the **Penalized Logistic Regression** (PLR) model. This problem occurs when a predictor variable or a combination of them can perfectly discriminate between the outcome classes, resulting in infinite standard error of coefficient estimates and unreliable model predictions. PLR is a regularization technique that addresses the issue of perfect separation by adding a penalty term to the likelihood function and this penalty term helps to stabilize the coefficient estimates. This approach allows for more reliable estimation of the model coefficients and provides a more realistic representation of the relationship between the predictor variables and the outcome. The usual maximum likelihood estimates of the regression coefficients β_r(r = 1, …, k) are obtained by solving the score equations, which are given by $\frac{\partial l}{\partial \beta_r}$ = $U(\beta_r) = 0$, where l represents the log-likelihood function. In PLR, to address the issue of small sample bias, Firth [29] proposed a modification to the score equations given by:

$$U(\beta_r)^* = U(\beta_r) + \frac{1}{2} \cdot trace\ \left[I(\beta)^{-1} \left\{ \frac{\partial l(\beta)}{\partial \beta_r} \right\} \right] \tag{4}$$

Solving $U(\beta_r)^* = 0$ leads to the estimates of Penalized Logistic Regression.

Support Vector Machines

Support Vector Machines [30, 31] is a powerful supervised learning algorithm used for classification and regression tasks. At its core, SVM is a binary classification algorithm that aims to find an optimal hyperplane in a high-dimensional feature space that separates different classes with the maximum margin (margin is the minimum distance from the hyperplane to any point of the sample). The hyperplane is a decision boundary that maximizes the distance between the nearest data points of different classes, known as support vectors (see Figure 2.9). This approach allows the algorithm to be robust to outliers but less robust in the face of mislabeled points in the initial sample and generalize well to unseen data.

The key idea behind Support Vector Machines is to transform the input data into a higher-dimensional space using kernel functions. This transformation enables SVM to capture complex relationships and non-linear decision boundaries that may not be achievable in the original feature space. Common kernel functions used in SVM include linear, polynomial, and radial basis function. In Figure 2.7, we have two feature spaces where we applied a kernel function to the values, while in the second figure (Figure 2.8), the values are first shown in the untransformed scale.

Support Vector Machines hyperplanes

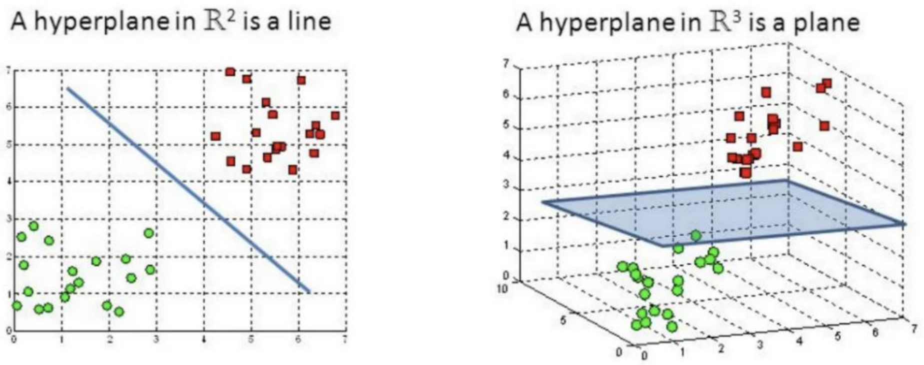


Figure 2.7: R^2 and R^3 SVM hyperplanes
source: Marktechpost

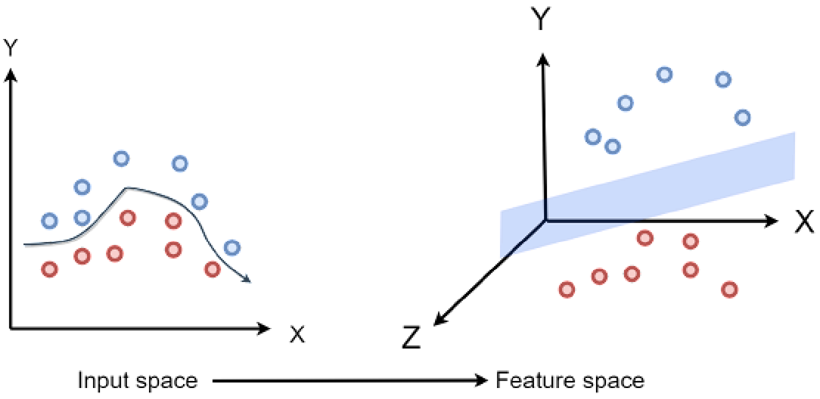


Figure 2.8: SVM feature transformation
source: Pycodemates

During the training phase, SVM learns the optimal hyperplane by solving an optimization problem that involves minimizing the classification error and maximizing the margin. This process involves finding the support vectors and determining the appropriate weights for each data point.

One of the strengths of SVM is its ability to handle datasets with a high number of features, as it focuses on the support vectors instead of all the data points. This makes SVM less susceptible to the curse of dimensionality and more efficient in terms of memory usage and computational time. The execution time of the proposed algorithm exhibits a cubic growth pattern with respect to the number of training examples and a linear growth pattern with respect to the number of variables [32].

Support Vector Machines

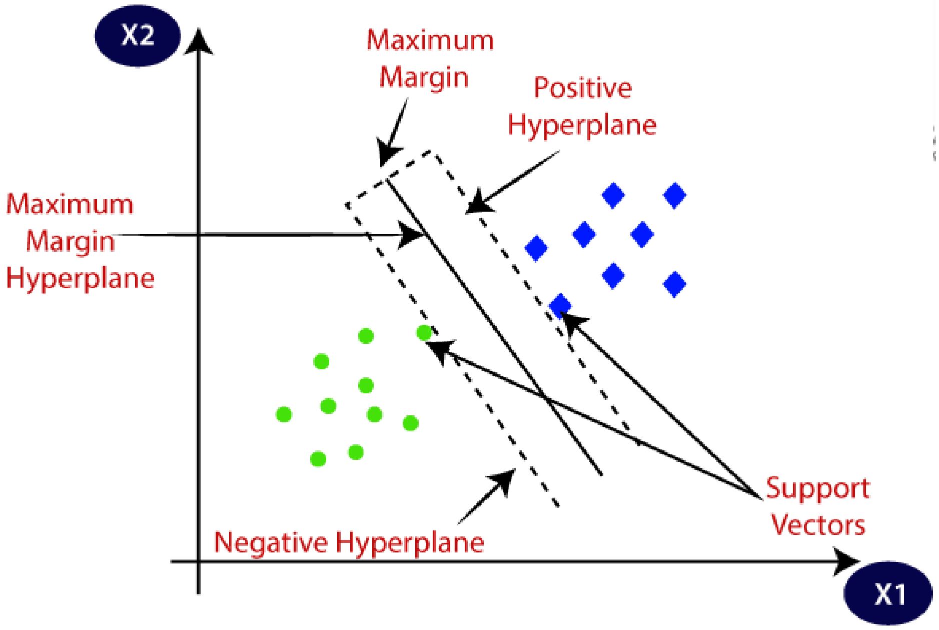


Figure 2.9: *SVM hyperplane visualization*
source: Medium

SVM has several variants depending on how the points where projected in a higher dimension using kernel functions. In our case, the focus will be on the linear and radial SVMs:

- **Linear SVM** aims to find a hyperplane in the feature space that separates the classes with a maximum margin. It assumes a linear decision boundary, is suitable for linearly separable data and it is computationally efficient and works well in high-dimensional spaces.

For a binary classification problem, the linear SVM aims to find an optimal hyperplane in the feature space that separates the two classes. The hyperplane is represented by the equation:

$$W^T X - b = 0, \tag{5}$$

where W is the weight vector perpendicular to the hyperplane, X is the input feature vector, and b is the bias term. The goal is to determine the optimal values of $W$and b.

- The **radial SVM** (or Radial Basis Function, RBF) is commonly used in SVM because of its ability to handle non-linearly separable data by mapping it to a higher-dimensional space. The RBF kernel computes the similarity between two data points based on their distance from a reference point called the center. It assigns higher weights to data points that are closer to the center, capturing the local patterns in the data and this allows the RBF SVM to model complex decision boundaries and handle data with intricate relationships.

In the radial SVM, the kernel function used is the radial basis function, given by:

$$k(X_i, X_j) = \exp\left(-\gamma \left|\left|X_i - X_j\right|\right|^2\right) \text{ for } \gamma > 0\,, \tag{6}$$

where γ is a hyperparameter that controls the width of the RBF kernel and $\left|\left|X_i - X_j\right|\right|^2$ represents the squared Euclidean distance between the support vector X_i and the input sample **X**.

In the context of SVM with an RBF kernel, the **gamma** parameter controls the width of the kernel function (see Figure 2.10). It determines the reach or influence of each training example on the decision boundary. A smaller gamma value results in a broader kernel, where each training example has a wider influence on the decision boundary. This can lead to a smoother decision boundary and potentially capture larger patterns in the data. However, a very small gamma value may cause the model to underfit and fail to capture more intricate patterns. On the other hand, a larger **gamma** value narrows the kernel, making each training example have a more localized influence. This can result in a decision boundary that is more sensitive to individual data points and can capture smaller-scale patterns. However, a very large **gamma** value may cause the model to overfit and perform poorly on unseen data.

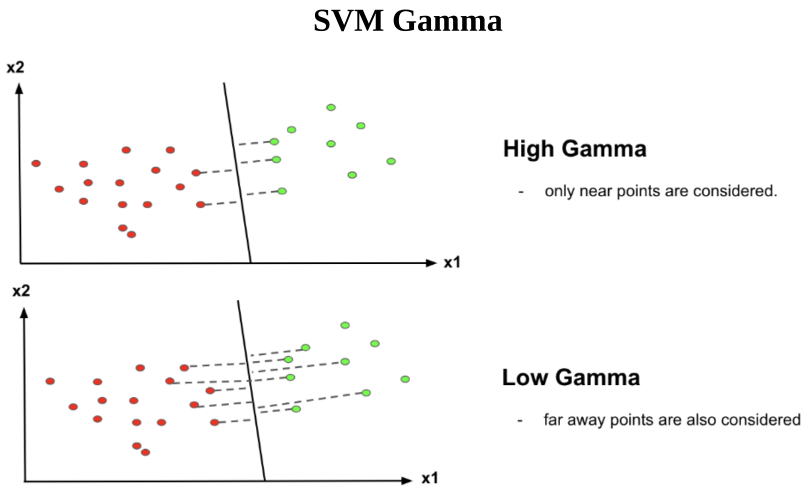

SVM Gamma
x2
x1
High Gamma
- only near points are considered.
x2
x1
Low Gamma
- far away points are also considered

Figure 2.10: SVM high and low gamma comparison
source: Analytics Vidhya

In the context of Support Vector Machines, the **cost** refers to a parameter that controls the balance between accurately fitting the training data and maximizing the margin of separation between classes. It is applied to classification errors made by the model. A higher cost penalizes classification errors more heavily, meaning the model will strive to minimize the number of misclassified instances, even if it results in a smaller margin of separation. On the other hand, a lower cost allows for more classification errors and may result in a larger margin of separation. Adjusting the cost is an important aspect when training an SVM model as it allows for balancing the trade-off between accuracy and generalization of the model. It can be seen as a hyperparameter to control the overfitting. Selecting the appropriate cost depends on the specific problem at hand and may require fine-tuning and experimentation to achieve the best model performance.

In the implementation of Support Vector Machines, we focused on both linear and radial kernels. For the linear SVM, we performed a search using different values of the cost parameter to find the optimal model. We calculated the accuracy and the number of support vectors for each cost value and selected the best-performing model based on the highest ratio of accuracy. In the case of the radial SVM, we conducted a similar grid search by varying the cost and gamma parameters. The accuracy and the number of support vectors were calculated for each combination of cost and gamma values, and we identified the best-performing model by selecting the highest accuracy achieved.

2.3.3 Predictive performance measures

In this section, we will explore some concepts of predictive performance measures, which are essential metrics for evaluating the accuracy and effectiveness of predictive models.

Positive Predictive Value (PPV) and Negative Predictive Value (NPV)

PPV and NPV provide information about the reliability of positive and negative predictions, respectively. The first one, also known as precision, is the probability that a positive prediction or test result is truly positive. It represents the proportion of correctly predicted positive cases out of all the cases predicted as positive and it is calculated as the number of true positives divided by the sum of true positives and false positives. A higher PPV indicates a lower rate of false positives and suggests that a positive prediction is more likely to be accurate.

NPV, on the other hand, is the probability that a negative prediction or test result is truly negative. It represents the proportion of correctly predicted negative cases out of all the cases predicted as negative and it is calculated as the number of true negatives divided by the sum of true negatives and false negatives. A higher NPV indicates a lower rate of false negatives and suggests that a negative prediction is more likely to be accurate.

The positive predictive value (PPV), or precision, is defined as:

$$PPV = \frac{Number\ of\ true\ positives}{\text{Number of true positives} + \text{Number of false positives}} \tag{7}$$

And the negative predictive value (NPV) is defined as:

$$NPV = \frac{Number\ of\ true\ negatives}{\text{Number of true negatives} + \text{Number of false negatives}} \tag{8}$$

Both PPV and NPV are influenced by the prevalence of the condition or event being predicted. When the prevalence is low, even a highly specific test or model may have a lower PPV because the probability of false positives is higher. Conversely, when the prevalence is high, the NPV may be lower because the probability of false negatives is higher (see Figure 2.11).

NPV and PPV

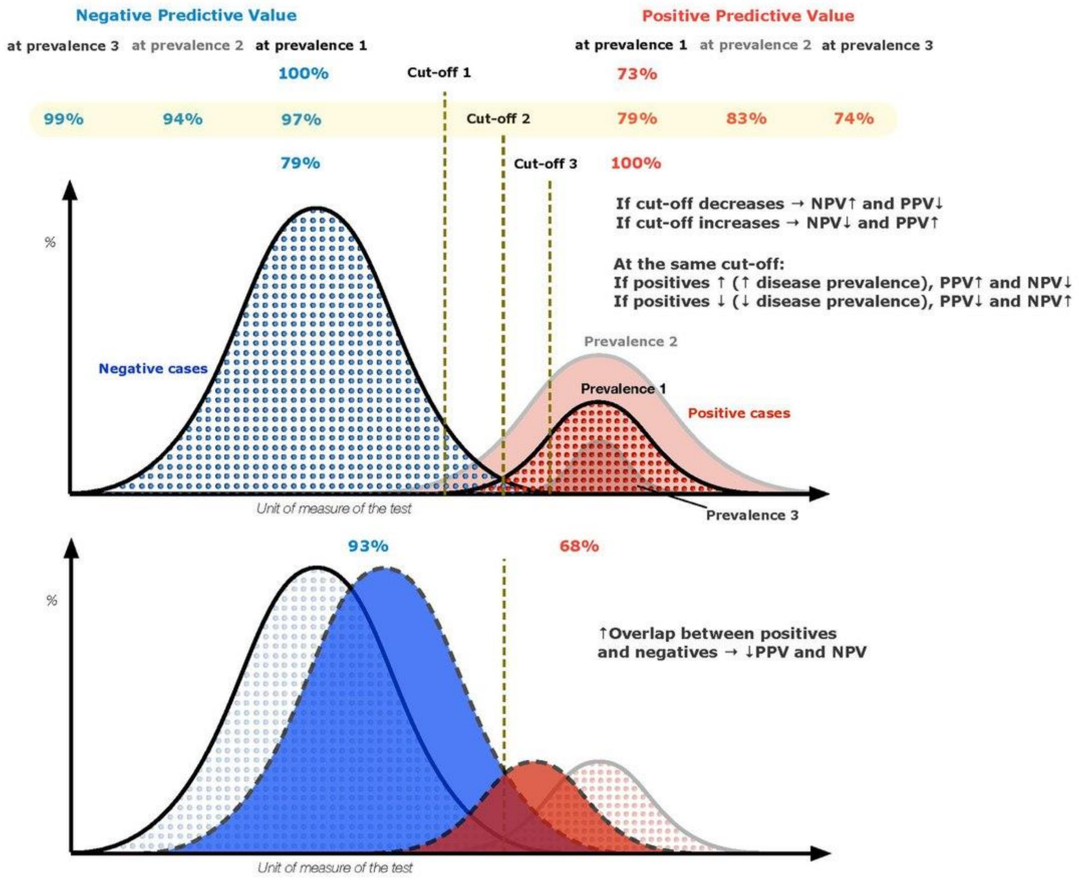


Figure 2.11: Positive and negative predictive values
source: Wikipedia

In our case, PPV is the probability that a text is actually AI-generated if the model has predicted it as AI-generated and NPV is the probability that a text is actually human generated if the model has predicted it as human generated.

Sensitivity and Specificity

Sensitivity, also known as recall, and specificity are two important metrics used to evaluate the performance of a classification model.

Sensitivity measures the proportion of actual positive cases that are correctly identified as positive by a model or a test. It focuses on the ability of the model to correctly detect positive instances. Sensitivity is calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (FN):

$$Sensitivity = \frac{TP}{\mathrm{TP} + \mathrm{FN}} \tag{9}$$

A high sensitivity value indicates that the model is effective in identifying positive cases and has a low rate of false negatives.

Specificity, on the other hand, measures the proportion of correctly identified positive cases out of all cases predicted as positive. It focuses on the accuracy of positive predictions and it is calculated as the ratio of true negatives (TN) to the sum of true negatives and false positives (FP):

$$Specificity = \frac{TN}{\mathrm{TN} + \mathrm{FP}} \tag{10}$$

A high specificity value indicates that the model has a low rate of false positives and it is precise in its negative predictions.

In our case, sensitivity measures the ability of the model to correctly identify texts that are generated by AI (true positives) out of all the texts that are actually generated by AI (true positives + false negatives). It quantifies the proportion of AI-generated texts that the model correctly detects. Furthermore, specificity measures the capability of the model regarding the negative outcomes (human-generated texts). It quantifies the proportion of human-generated texts predicted as human-generated (true positives) out of all the human-generated texts.

The Figure 2.12 provides a graphical visualization to distinguish between precision (PPV) and recall (sensitivity). This visual representation allows us to gain insights into the trade-off between these two performance metrics and it helps to understand how changes in the model's threshold or decision boundary impact the balance between correctly identified positive cases (precision) and the ability to capture all positive cases (recall).

Precision and Recall

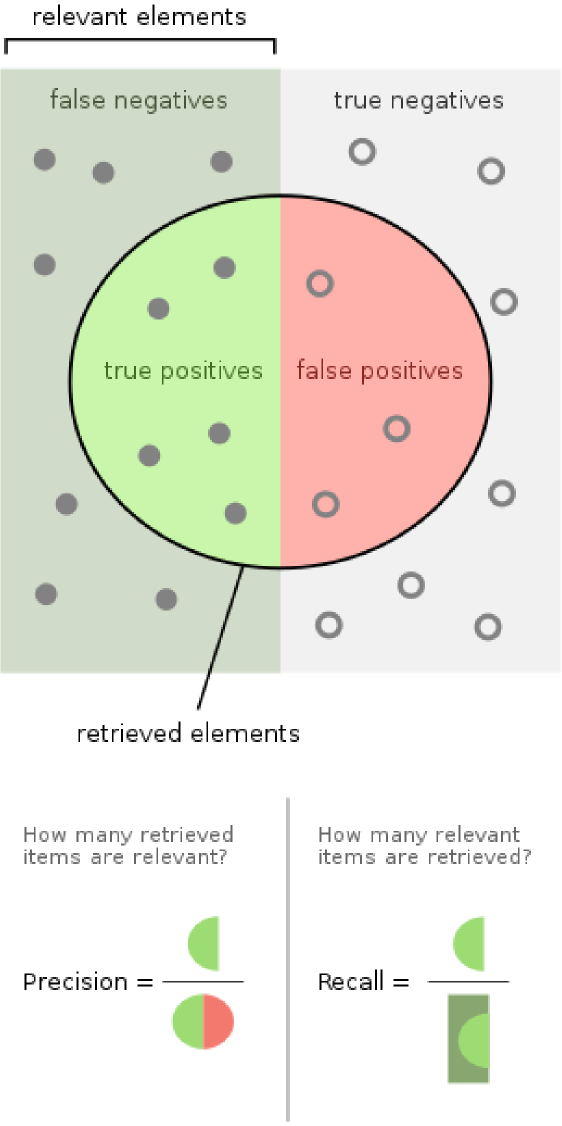


Figure 2.12: Precision and recall diagram
source: Super.ai

Accuracy

In statistics, accuracy refers to how close a measurement or estimate is to the true or expected value of a quantity. That is, it is a measure of how well an accounting system or method can predict or estimate a target variable.

In our context of classification, accuracy refers to the proportion of correct predictions made by a model among all the predictions it has made. In binary classification, it is calculated by dividing the number of correct predictions (true positives and true negatives) by the total number of instances in the dataset.

$$Accuracy \ = \frac{TP + TN}{\mathrm{TP + TN + FP + FN}} \tag{11}$$

Accuracy is a simple and intuitive measure that provides an assessment of the efficiency of a binary classification model. Higher precision indicates that the model made more correct predictions, while lower precision indicates more misclassification. However, accuracy alone may not always provide a complete evaluation of a model's performance, especially when dealing with imbalanced datasets or when the costs of misclassifying different classes vary.

True Negative, False Negative, False Positive, and True Positive

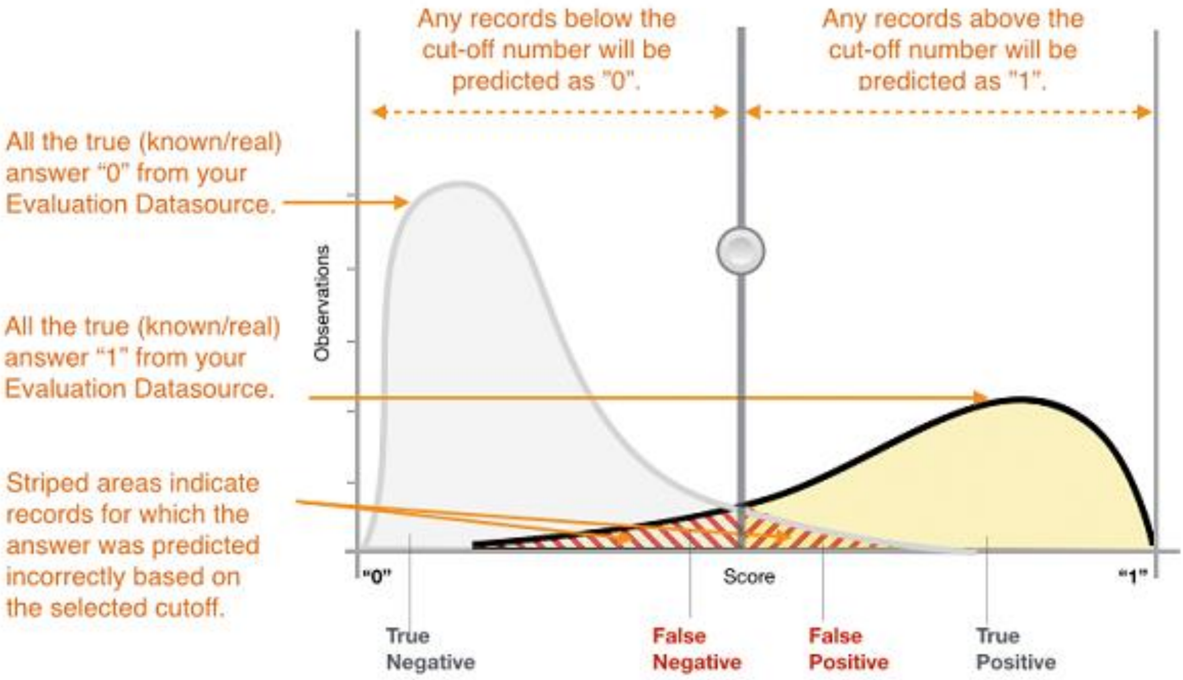

Any records below the cut-off number will be predicted as "0".
Any records above the cut-off number will be predicted as "1".
All the true (known/real) answer "0" from your Evaluation Datasource.
All the true (known/real) answer "1" from your Evaluation Datasource.
Striped areas indicate records for which the answer was predicted incorrectly based on the selected cutoff.
Observations
"0"
Score
"1"
True Negative
False Negative
False Positive
True Positive

Figure 2.13: TN, FN, FP, and TP outline
source: Wentz Wu

In Figure 2.13, we can observe a graphical representation of the components of accuracy, including True Negatives (TN), False Negatives (FN), False Positives (FP), and True Positives (TP). This figure provides a visual depiction of the classification outcomes and helps us understand the performance of our model in terms of these different categories.

ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classification model. It plots the true positive rate (sensitivity) against the false positive rate (1 - specificity) at various classification thresholds (see Figure 2.14). The ROC curve helps to visualize the trade-off between sensitivity and specificity for different decision thresholds.

ROC Curve

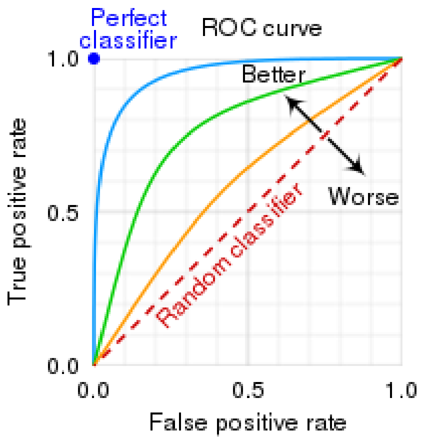


Figure 2.14: ROC curve representation
source: [Wikipedia]

A good predictive model will have a curve with a very close point to the top-left corner. The ROC curve provides valuable insights into the model's ability to distinguish between positive and negative instances and assists in selecting an appropriate threshold for classification based on the desired trade-offs. The Figure 2.15 illustrates the interpretation for different shapes of the ROC curve.

ROC Curve

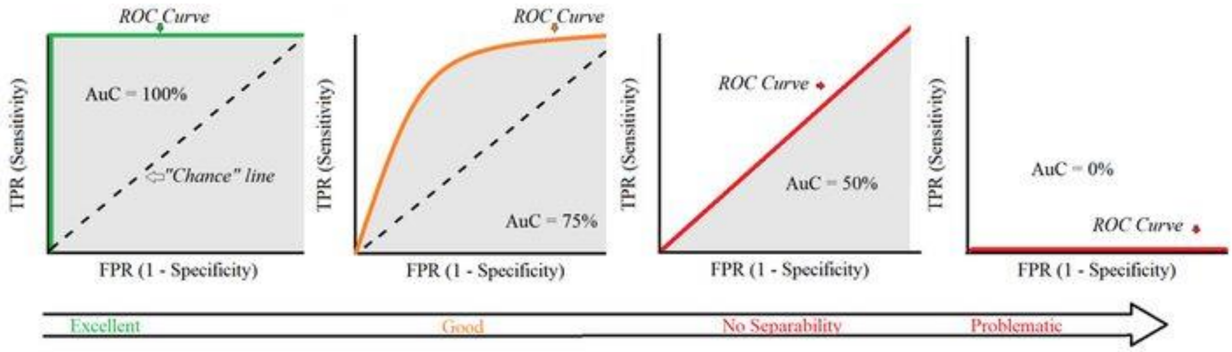


Figure 2.15: ROC curve interpretation diagram
source: [AI Wiki]

Area Under the Curve

The Area Under the ROC Curve (AUC) is a metric commonly used to evaluate the performance of a binary classification model. It represents the overall discriminative power of the model in distinguishing between positive and negative instances across all possible classification thresholds. It ranges from 0 to 1, with a higher value indicating better performance. Then, a higher AUC suggests that the model is more capable of correctly ranking positive instances higher than negative instances, regardless of the threshold used for classification (see Figure 2.16).

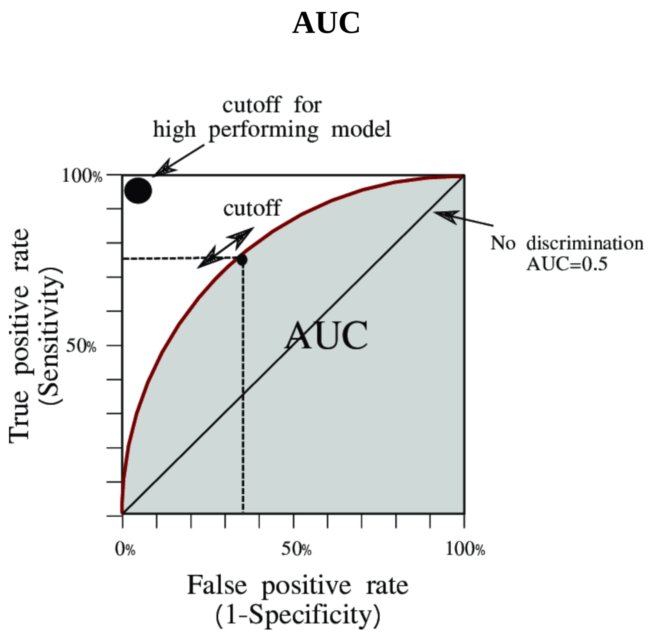


Figure 2.16: AUC outline
source: Github

To calculate the AUC, we integrate the ROC curve from (0,0) to (1,1), which represents the worst and best possible classification scenarios, respectively. The AUC ranges from 0.5 to 1, where an AUC of 1 represents a perfect classifier, and an AUC of 0.5 represents a random classifier.

Cohen’s kappa

Cohen's kappa [33, 34] is a statistical measure used to assess the agreement between two raters or evaluators when dealing with categorical data. It considers both the observed agreement and the agreement that would be expected by chance. The coefficient ranges from -1 to 1. A value of 1 indicates perfect agreement, 0 indicates agreement due to chance, and -1 indicates complete disagreement.

The interpretation of the coefficient is as follows:

- A kappa coefficient greater than 0.8 is considered as excellent agreement.
- A value between 0.6 and 0.8 represents substantial agreement.
- An agreement between 0.4 and 0.6 is considered moderate.
- A value below 0.4 indicates fair or poor agreement.

Cohen's kappa is particularly valuable in situations where balanced datasets are present or when the assessed categories have varying prevalence rates. It offers a robust measure of agreement by accounting for the level of agreement that would be expected by chance alone. This makes Cohen's kappa a reliable and informative metric, especially when evaluating classification performance or inter-rater agreement in such challenging scenarios.

The formula for calculating Cohen's kappa involves comparing the observed agreement (the number of agreements between the raters) and the expected agreement (the agreement that would be expected by chance). The formula is as follows:

$$k = \frac{p_0 - p_e}{1 - p_e}, \tag{12}$$

where p_0 is the observed proportionate agreement between the raters and p_e is the expected proportionate agreement by chance.

2.3.4 Assessing predictive performance

Cross-validation, also known as rotation estimation or out-of-sample testing, is a valuable technique for assessing the generalizability of statistical analysis results to an independent dataset. It involves resampling the data and using different subsets for training and testing the model in multiple iterations. This approach is particularly useful in prediction-focused scenarios, where the objective is to estimate the performance of a predictive model in practical applications. In a prediction problem, the model is typically trained on a known dataset (referred to as the training dataset) and then evaluated on an unknown dataset (referred to as the validation dataset or testing set). The purpose of cross-validation is to assess the model's ability to accurately predict new data that was not used during its estimation process. By doing so, it helps identify potential issues like overfitting or selection bias and provides insights into how well the model will generalize to an independent dataset.

Furthermore, overfitting is a common problem in Classification Trees where the model becomes too complex and overly specific to the training data. It occurs when the model captures noise or irrelevant patterns in the data, leading to poor performance on unseen data and this happens when the tree grows too deep or when the decision rules become too specific, resulting in a high variance and low bias.

One reason for overfitting is a small training dataset that does not provide enough diverse examples to capture the true underlying patterns in the data. In such cases, the tree may learn from random variations or outliers, leading to inaccurate predictions on new data. Another cause of overfitting is the over-selection of features. If the tree is allowed to consider too many variables or interactions, it can fit the noise in the training data and fail to generalize to unseen data. This problem can be mitigated by pruning the tree, limiting the depth, or setting thresholds for variable importance. To address overfitting, techniques like cross-validation can be used to evaluate the model's performance on multiple subsets of the data. Regularization methods, such as reducing the complexity of the tree or applying penalties to overly complex models, can also help prevent overfitting.

To ensure the reliability of our predictions, we employed cross-validation for the Classification Tree, Random Forest, and Logistic Regression models. We used 10 repetitions with 5 folds each in the Random Forest model, and 5 folds in the Logistic Regression model.

Cross-validation is a widely used technique for model evaluation and selection, as it helps assess the performance of a model on unseen data. However, in the case of Support Vector Machines, it is common to use a train/test split instead of cross-validation. There are a few reasons for this.

Firstly, SVMs can be computationally expensive, especially for large datasets or complex models. Performing cross-validation on every fold can significantly increase the training time. In contrast, a train/test split allows for a faster evaluation of the model's performance by training it once on the training set and evaluating it on the separate test set. Additionally, SVMs have certain assumptions and requirements regarding the separation of classes and the selection of hyperparameters. Splitting the data into a training and test set allows for a more realistic assessment of how the model generalizes to unseen data. It provides a clearer indication of how well the SVM performs on new observations and whether it is likely to have good predictive capabilities.

2.3.5 R Shiny

R Shiny [35] is a web application framework that allows to build interactive web-based interfaces for their R code and data analyses. It provides a way to create dynamic and interactive dashboards, visualizations, and data-driven applications without needing to write extensive HTML, CSS, or JavaScript code. At its core, R Shiny leverages the power of R. It enables users to develop web applications that leverage R's rich ecosystem of packages and functions for data manipulation, modeling, and visualization.

The main concept behind R Shiny is the separation of the user interface (UI) and the server logic. The UI defines the appearance and functionality of the web application, while the server handles the data processing and computation. This separation allows for a modular and organized approach to building applications. In R Shiny, the UI is defined using R code and a set of predefined functions and components which include text inputs, buttons, sliders, tables, and plots, among others and developers can customize the appearance and layout of these components using HTML and CSS if desired.

On the other hand, the server is where the data analysis and computation take place. It receives input from the UI, processes it using R functions, and generates the corresponding output, which can be dynamically updated based on user interactions, allowing for real-time data visualization and analysis.

In Figure 2.17, we can observe a schematic representation of the functioning of our Shiny app, which provides a clear and visual understanding of how it operates in our case.

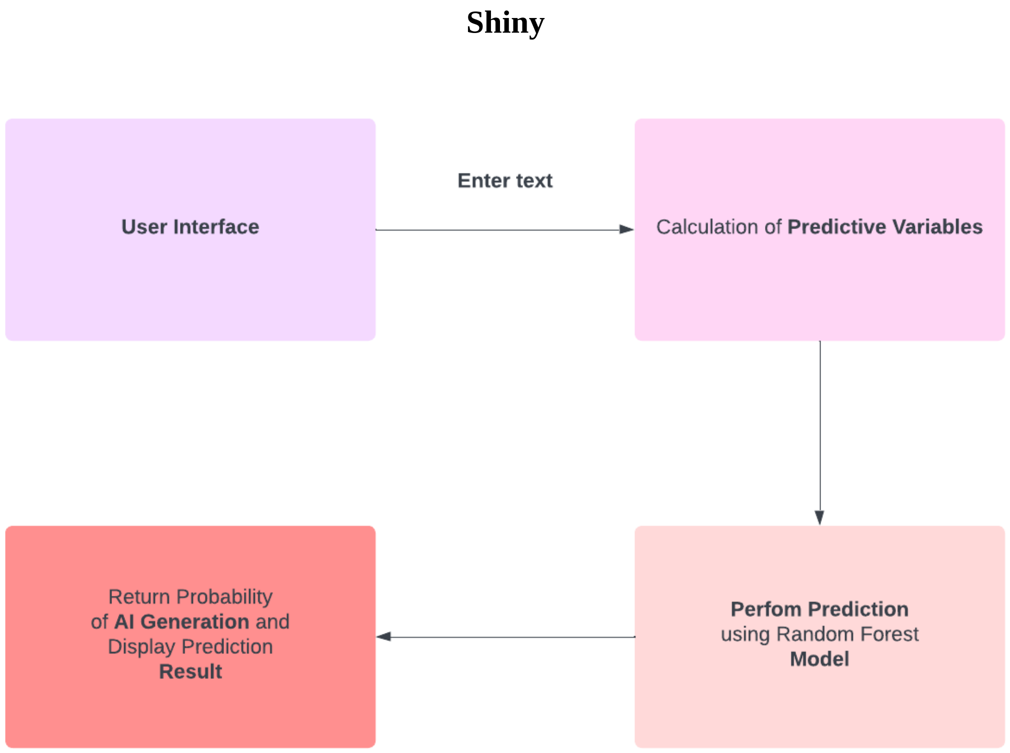

Shiny
User Interface
Enter text
Calculation of Predictive Variables
Perfom Prediction using Random Forest Model
Return Probability of AI Generation and Display Prediction Result

Figure 2.17: Shiny app flowchart

3. RESULTS

3.1 Exploratory Data Analysis (EDA)

As previously stated, the main goal of EDA is to gain insights and knowledge about the data that can inform subsequent data modeling or analysis. Therefore, we will now analyze the variables in the database to achieve this objective and we will present descriptive statistics in graphical form to provide a comprehensive visual representation of the variables' characteristics. For a more detailed analysis and additional information, refer to the Appendix 6.4.

3.1.1 Categorical variables

In this section, we will explore the categorical variables present in our dataset. Categorical variables represent qualitative characteristics or groups, providing valuable insights into different segments or classes within the data. By analyzing these variables, we aim to gain a deeper understanding of the distribution, frequencies, and patterns.

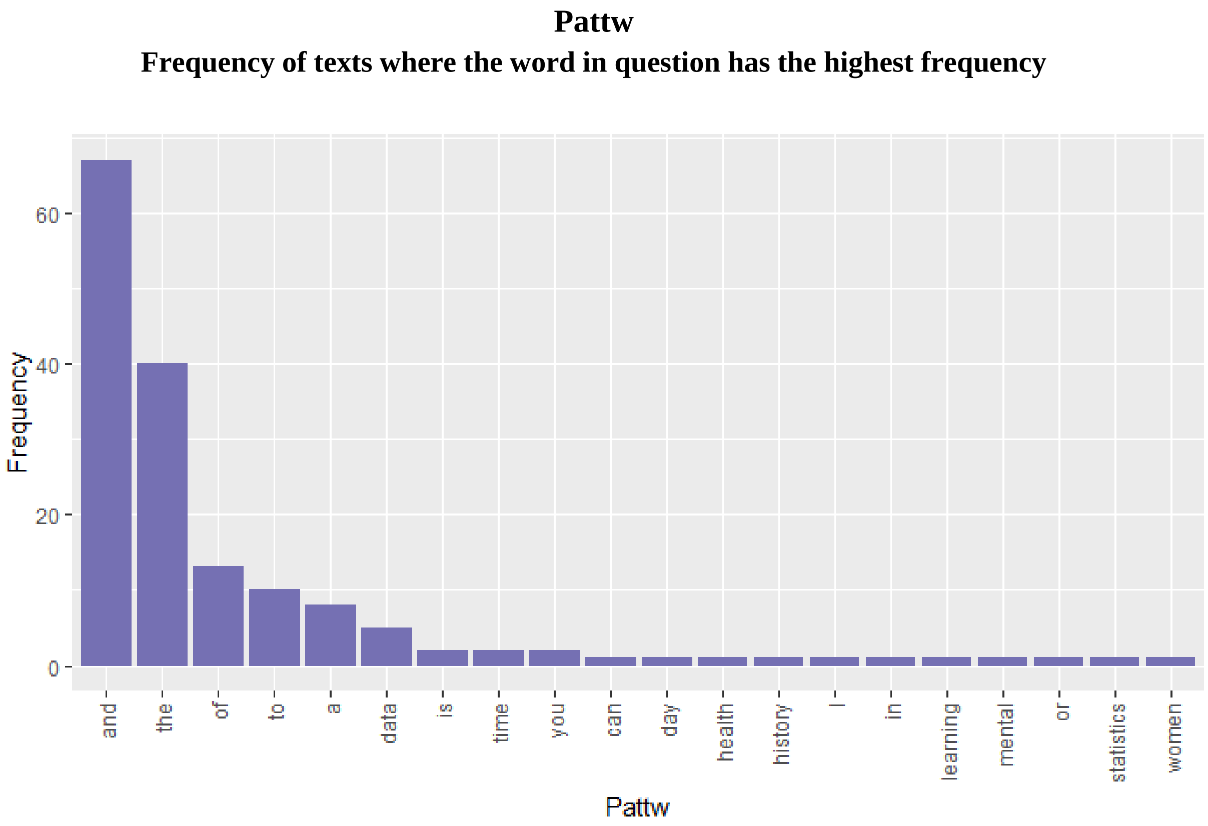


Figure 3.1: Pattw variable representation

As we can observe in Figure 3.1, among the words with the greatest number of texts where it is the most frequent variable, *and* stands out significantly, representing 41.87% of the total number of texts. Following closely behind with a bit of a margin is *the* with a representability of 25%.

3.1.2 Numeric variables

In this part, we will examine the numeric variables in our dataset. These ones represent quantitative measurements or values, providing important numerical information that can be analyzed and interpreted. This analysis will help us understand the patterns, trends, and potential associations between the numeric variables and other features in our dataset.

Table 3.1: Numeric variables summary

Variable	**Min.**	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**SD**
lengthl	266.0	867.2	1443.5	1395.6	1771.2	5089.0	702.6
lengthw	51.0	136.8	219.5	218.4	273.8	765.0	108.1
pgf	1.0	2.0	4.0	3.9	5.0	12.0	2.4
nstc	1.0	6.0	10.0	9.8	13.0	30.0	4.8
wordl	4.660	6.023	6.431	6.379	6.685	7.594	0.494
sentm	-0.453	0.0227	0.138	0.156	0.284	0.743	0.203
sentsd	0.035	0.203	0.265	0.266	0.308	0.698	0.104
patt	0.030	0.049	0.057	0.063	0.071	0.122	0.019
pron	0.000	0.003	0.010	0.018	0.024	0.120	0.023
art	0.019	0.051	0.066	0.070	0.085	0.143	0.026
voc	0.000	0.012	0.02	0.018	0.026	0.035	0.009
form	1.922	2.000	2.000	2.010	2.023	2.118	0.024
aggr	0.000	0.009	0.016	0.024	0.030	0.139	0.025
avw	11.11	19.90	22.33	23.33	25.71	89.00	7.35
spgf	0.857	2.000	2.550	3.143	4.000	10.000	1.73

Variable	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	SD
wpgf	19.60	46.96	59.67	68.65	86.17	199.00	31.17
nstcw	0.011	0.039	0.045	0.046	0.050	0.090	0.011
pgfw	0.005	0.012	0.017	0.018	0.021	0.051	0.008
freqv	0.576	1.157	1.686	1.693	2.074	3.854	0.650
typew	0.000	0.012	0.02	0.018	0.026	0.035	0.009
sumrep	0.274	0.431	0.499	0.495	0.571	0.690	0.098
and	0.000	0.029	0.045	0.047	0.060	0.12	0.024
the	0.000	0.027	0.04	0.042	0.056	0.121	0.024

The Table 3.1 provides a numerical descriptive analysis of the variables. The first four ones were not used in the study as their standardized version were used instead.

As can be observed, precisely these variables along with *avw* and *wpgf* have the greatest standard deviation in their values and have quartile values that are farthest from the mean and median. Afterwards, we see that in general, there isn't much difference between the median and the mean of the variables, with the exception of *lengthl* (number of letters) and *wpgf* (average number of words per paragraph in each text). This may indicate the presence of outliers that are affecting the mean. Furthermore, values that deviate significantly from the 1st or 3rd quartile, in comparison to the interquartile range (IQR), may indicate the presence of outliers, this could be another way to detect them.

3.1.3 Bivariate analysis of two categorical variables

In this section, the bivariate analysis between predictive categorical variables and the outcome will allow us to explore the relationship and associations in our dataset. By examining the joint distribution of these variables, we can gain insights into potential patterns, or dependencies between them.

Pattw by Outcome

By limiting the analysis to the 15 words that appear in more texts as the most frequent, we aim to provide a more focused and informative visualization of the most relevant terms in the dataset. This approach allows us to better understand the overall patterns and trends in the data, and to identify key factors that may influence the outcome variable. Furthermore, it helps to avoid clutter and overcrowding in the visualization, which can obscure the main insights and make the interpretation more challenging.

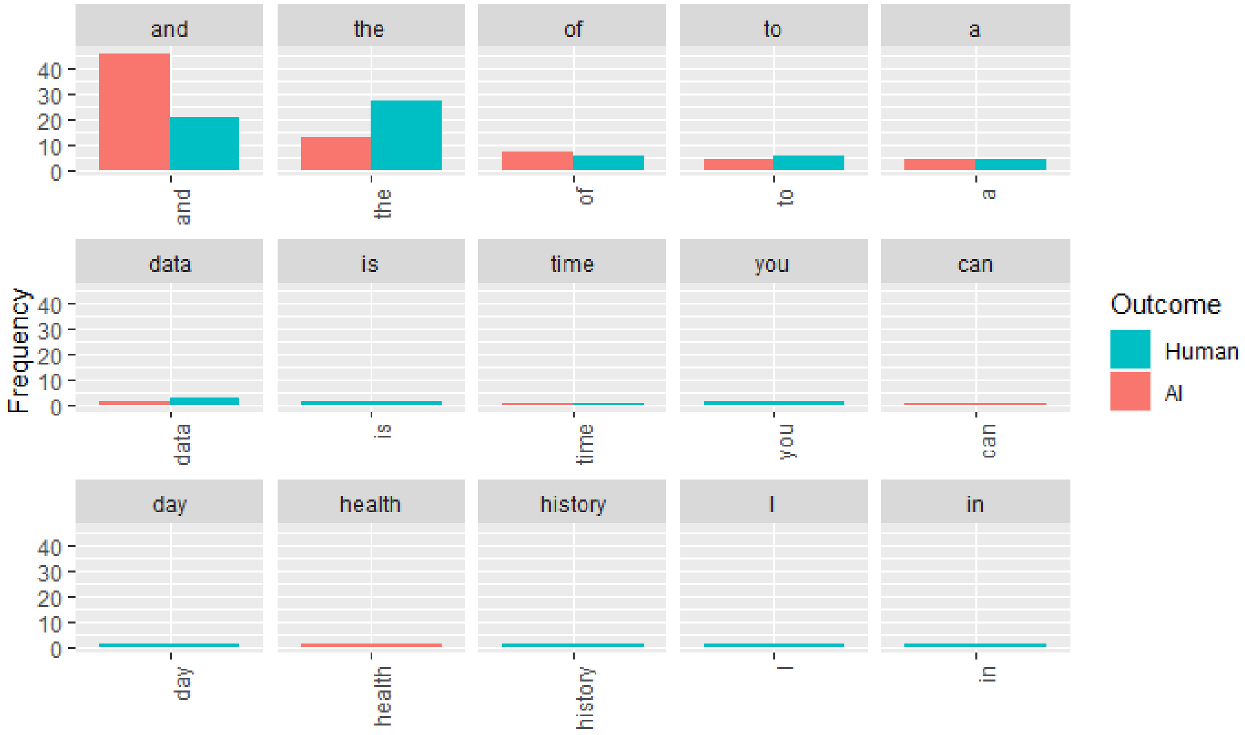


Figure 3.2: Bivariate analysis of Pattw and Outcome variables

This variable measures the frequency of texts where the most frequent word is X (e.g. *time*). It provides an indication of how often the specific word appears as the most frequent word in the texts. In Figure 3.2 we can observe that, on one hand, in texts generated by a human source, among the list of most common words *the* and *and*, represent 16.87% and 13.12% out of all texts, respectively. On the other hand, in texts generated by AI, we see a significant difference with the word *and* (being predominantly the most frequent word in AI-generated texts), representing 28.75%. Then, *the* follows with 8.12%, slightly further away from the first, and *of* with 4.37%.

Table 3.2: *Comparison of frequencies of Human and AI texts*

word	**Human texts proportion**	**AI texts proportion**	**Proportion difference**	**95% CI**	**p-value**
a	0.05	0.05	0	(-0.02, 0.02)	1
and	**0.26**	**0.57**	**0.31**	**(-0.25, -0.06)**	**0.001**
can	0	0.01	0.01	(-0.02, 0.01)	1
data	0.04	0.02	-0.02	(-0.03, 0.04)	1
day	0.01	0	-0.01	(-0.01, 0.02)	1
health	0	0.01	0.01	(-0.02, 0.01)	1
history	0.01	0	-0.01	(-0.01, 0.02)	1
I	0.01	0	-0.01	(-0.01, 0.02)	1
in	0.01	0	-0.01	(-0.01, 0.02)	1
is	0.02	0	-0.02	(-0.01, 0.03)	0.48
of	0.07	0.09	0.02	(-0.05, 0.04)	1
the	**0.34**	**0.16**	**-0.18**	**(0.01, 0.16)**	**0.03**
time	0.01	0.01	0	(-0.02, 0.02)	1
to	0.07	0.05	-0.02	(-0.03, 0.06)	0.75
you	0.02	0	-0.02	(-0.01, 0.03)	0.48

We can’t reject the null hypothesis of independence except in two cases (for *and* and *the*) (see Table 3.2). Then, the variables *and* and *the* were created subsequently as they were found to be potentially significant. We consider the difference in proportions as AI – Human and a 0 value for the categorical variable indicates that the particular word did not appear as the most frequent word in any text, however, it does not imply an absolute absence of the word in all contexts.

3.1.4 Bivariate analysis of binary and continuous variables

In this section, we present the bivariate analysis of binary and continuous variables involves examining the relationship between them. This analysis allows us to explore how the continuous variable varies or differs across the two categories of the binary variable.

Table 3.3: Comparison of continuous variables between Human and AI texts

Variables	Human mean	AI mean	Mean differences	95% CI	p-value
wordl by outcome	**6.23**	6.52	0.29	**(-0.44, -0.14)**	**0.0001**
sentm by outcome	**0.11**	**0.2**	**0.09**	**(-0.14, -0.02)**	**0.01**
sentsd by outcome	0.26	0.27	0.01	(-0.04, 0.02)	0.53
patt by outcome	**0.06**	**0.07**	**0.01**	**(-0.016, -0.004)**	**0.0009**
pron by outcome	**0.02**	**0.01**	**-0.01**	**(0.007, 0.021)**	**0.0001**
art by outcome	0.07	0.07	-0.0003	(-0.007, 0.008)	0.94
voc by outcome	**0.01**	**0.02**	**0.01**	**(-0.009, -0.003)**	**6.96e-05**
form by outcome	2.01	2.01	0.0001	(-0.008, 0.007)	0.97
aggr by outcome	0.02	0.02	0.003	(-0.011, 0.005)	0.47
avw by outcome	23.1	23.57	0.47	(-2.78, 1.84)	0.67
spgf by outcome	**3.94**	**2.34**	**-1.6**	**(1.12, 2.08)**	**1.73e-09**
wpgf by outcome	**83.56**	**53.74**	**-29.82**	**(21.22, 38.40)**	**3.04e-10**
nstcw by outcome	**0.05**	**0.04**	**-0.01**	**(0.0007, 0.0077)**	**0.02**
pgfw by outcome	**0.01**	**0.02**	**0.01**	**(-0.008, -0.004)**	**5.13e-07**
freqv by outcome	**1.48**	**1.91**	**0.43**	**(-0.62, -0.24)**	**1.71e-05**
typew by outcome	**0.01**	**0.02**	**0.01**	**(-0.009, -0.003)**	**6.96e-05**
sumrep by outcome	**0.44**	**0.54**	**0.1**	**(-0.13, -0.07)**	**3.44e-12**
and by outcome	**0.03**	**0.06**	**0.03**	**(-0.03, -0.01)**	**3.823e-10**
the by outcome	0.044	0.041	-0.003	(-0.005, 0.01)	0.54

As is evident from the results in the Table 3.3, there are several statistically significant differences in means (AI mean – Human mean) in numerical variables, as indicated by various p-values smaller than 0.05.

We can divide the variables with significant differences into two groups: those with a higher value in human-generated texts and those with a higher value in texts generated by AI.

The ones that belong to the first group are (higher value in human texts): *pron* (number of pronouns), *spgf* (average number of sentences per paragraph), *wpgf* (average number of words per paragraph) and *nstcw* (ratio of the number of sentences to the total number of words).
The ones belonging to the second group are (higher value in AI texts): *wordl* (number of characters per word in each text), *sentm* (sentiment score mean), *patt* (maximum frequency of any word that appears multiple times), *voc* (number of unique words that appear more than five times), *pgfw* (ratio of the number of paragraphs to the total number of words), *freqv* (standard deviation of the frequencies of each word), *typew* (relationship between the number of unique word types and the total number of words), *sumrep* (sum of word repeats for each text, standardized by dividing it by the total number of words), and *and* (number of *and* words).

Finally, a large difference in means indicates that there is a substantial difference between the groups, but it should be borne in mind the scale of the variable considered to evaluate that difference. Its 95% confidence interval is an estimated range of values in which the true difference of means is expected to lie. If the interval does not include the zero, the difference of means is considered statistically significant. In this case, the variables with a confidence interval that does not include zero are the same as mentioned above.

3.2 Classification Methods

In this section, we present the results of our classification methods, including Classification Trees, Random Forest, Logistic Regression, and Support Vector Machines. These models were applied to our dataset with the objective of predicting and classifying the target variable. We will provide an overview of the performance metrics and evaluation measures used to assess the effectiveness of each method.

3.2.1 Classification Tree

In the following part, we will delve into the results obtained from the CT method. This algorithm is an effective approach for classification tasks, utilizing a decision tree structure to partition the dataset based on predictor variables.

In our case, the optimal cp value was determined to be 0.075 (see Figure 3.3) and we can observe low discrimination. The accuracy of the model for the optimal value of cp is 0.79, which means that the model is able to correctly predict 79% of the cases. The model has been implemented using the *caret* package in R [36]. In the obtained result, the value of kappa for the optimal value of cp is 0.65, suggesting moderate agreement between the observed classifications and those predicted by the model.

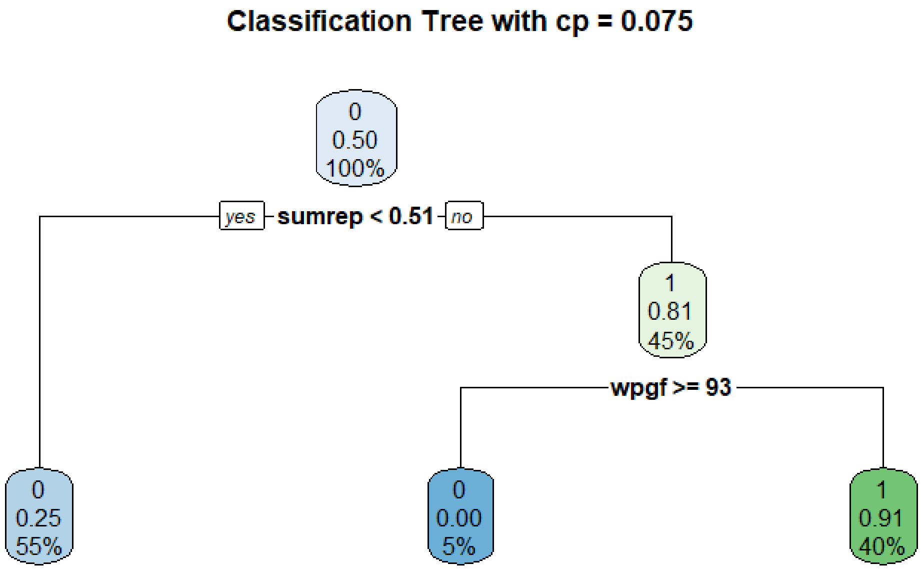


Figure 3.3: Classification Tree analysis with the optimal cp

At the initial node, the average value is 0.5 (equally distributed binary classification) between texts generated by humans and by artificial intelligence. We can observe that if the sum of repetitions is less than 0.51, it is more likely that the text is generated by humans (0). The average value of the response variable (prediction error) that falls in the node is 0.25 (i.e., out of every 100 cases that end up in the node, 75 will be human-generated texts and 25 will be AI-generated) and 55% of the initial sample observations end here.

Alternatively, if the sum of repetitions is greater than 0.51, it is more likely that the text is generated by AI, where the 81% of the 45% of the total texts are classified as 1 (AI). In this case (when the sum of occurrences is greater than 0.51), if the average number of words per paragraph is equal to or greater than 93, they will be texts generated by humans with 0% error, and 5% of the initial observations end at this final node. However, if the average number of words per paragraph is less than 93, it is more likely to be a text generated by AI with an average error value of 0.91, which means that out of every 100 observations, 91 will be AI-generated texts and 9 will be human-generated texts.

We can observe the ranking of importance according to this algorithm measured by the reduction in impurity of the variable (Gini index) in the Table 3.4:

Table 3.4: Seven most relevant variables for CT

Variable importance						
sumrep	**pgfw**	**wpgf**	**freqv**	**typew**	**voc**	**spgf**
20	17	17	14	9	9	9

And the Table 3.5 displayed below is the average of the confusion matrices from each of the folds created during cross-validation. For the confusion matrix, we consider 1 (AI) as the positive class since the objective of the study is to identify texts generated by artificial intelligence. For this model, we observe that the highest value is in the true negative category (the model correctly identifies text generated by human sources). Additionally, the predictions are well balanced (50% and 50%), indicating that the model is not biased towards any specific class and the sensitivity and specificity of the model would be 0.9 and 0.69, respectively.

Table 3.5: CT Confusion Matrix

	Reference	
Prediction	**0**	**1**
0	45.0	15.6
1	5.0	34.4

To determine which variables discriminate the most between 0 and 1 as in the first tree, we only obtain one important variable, it was decided to generate another classification tree with a smaller cp value, even though it may not be optimal, in order to gain more insights on which variables are relevant in the study.

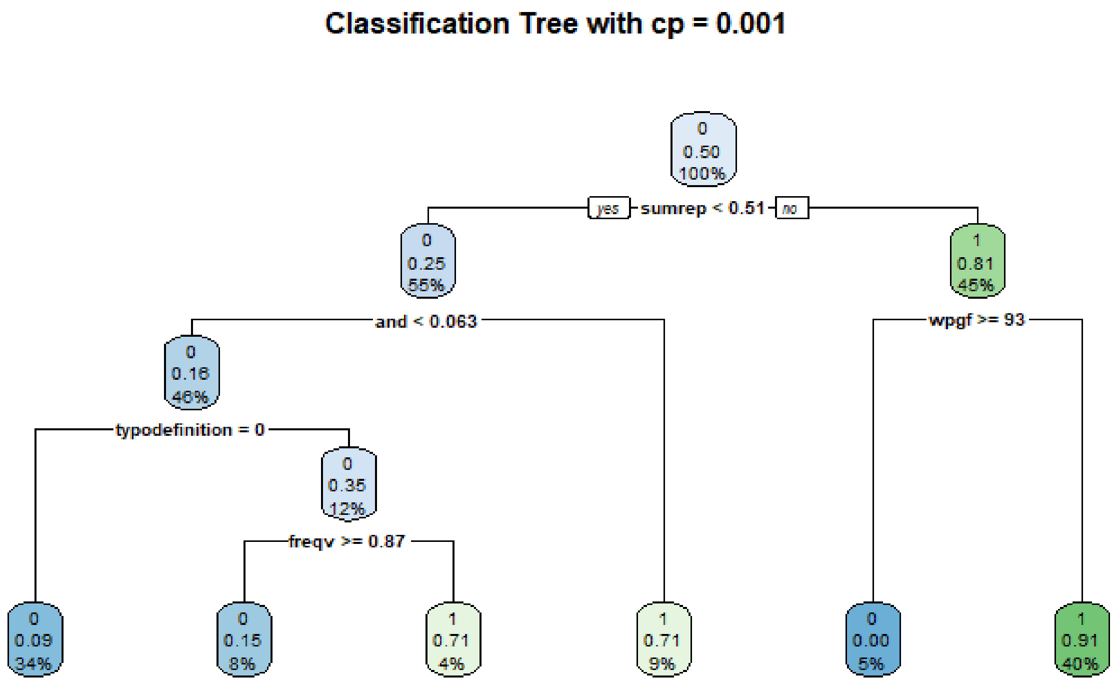


Figure 3.4: Classification Tree analysis with determined cp

After analyzing the results, we can see in Figure 3.4 that the tree maintains the same initial nodes with their corresponding prediction errors and percentages, but it has extended significantly, especially towards the left side of the tree.

In cases where the sum of repetitions is less than 0.51, and the frequency of occurrences of the word *and* is greater than 0.063, it is more likely to be generated by a AI source where the 71% of the 9% of the total texts are classified as AI. Otherwise, if the frequency of occurrences of the word and is less than 0.063 and it is not a definition type text, there will be a higher likelihood that it is a text generated by a human source. The average value of the response variable (prediction error) that falls in the node is 0.09 (i.e., out of every 100 cases that end up in the node, 91 will be human-generated texts and 9 will be AI-generated) and 34% of the initial sample observations end here. Stepping back, in the case of a definition type text, if the standard deviation of the frequencies of each word is greater or equal than 0.87, it will be more likely to be a human text with a prediction error rate of 0.15. Finally, remaining in the same case but with a standard deviation of the frequencies of each word less than 0.87, the text will have a higher probability of being AI, where the 71% of the 4% of the total texts are classified as AI.

3.2.2 Random Forest

In this section, we present the results and analysis of the Random Forest algorithm which is a powerful machine learning algorithm that combines multiple decision trees to make accurate predictions and classifications.

The accuracy of the model is 0.83, which is provided by the confusion matrix and means that out of all the predictions the model made, 83% were correct. The Table 3.6 shows the variable importance measures (using the Gini index) for the Random Forest model. The *Overall* column displays the average importance across all variables.

Table 3.6: Variables ordered by relevance for RF

Rank	Variable	Overall	Rank	Variable	Overall
1	sumrep	100.00	**11**	avw	19.48
2	pgfw	56.24	**12**	typodefinition	16.84
3	wpgf	55.19	**13**	patt	15.49
4	and	52.11	**14**	nstcw	13.77
5	spgf	47.20	**15**	form	13.76
6	freqv	35.39	**16**	pattwand	11.58
7	wordl	28.98	**17**	art	10.76
8	voc	26.30	**18**	aggr	9.48
9	typew	24.26	**19**	typoreview	8.37
10	pron	19.96	**20**	pattwis	7.89

Furthermore, in Figure 3.5, we can compare the top 23 most relevant variables in a more visual way. The plot provides a graphical representation of the variable importance measures in the model. It visualizes the variability or spread of the importance values across different trees in the forest and shows the mean decrease accuracy (*MeanDecreaseAccuracy*) as the main metric to quantify the importance of each variable.

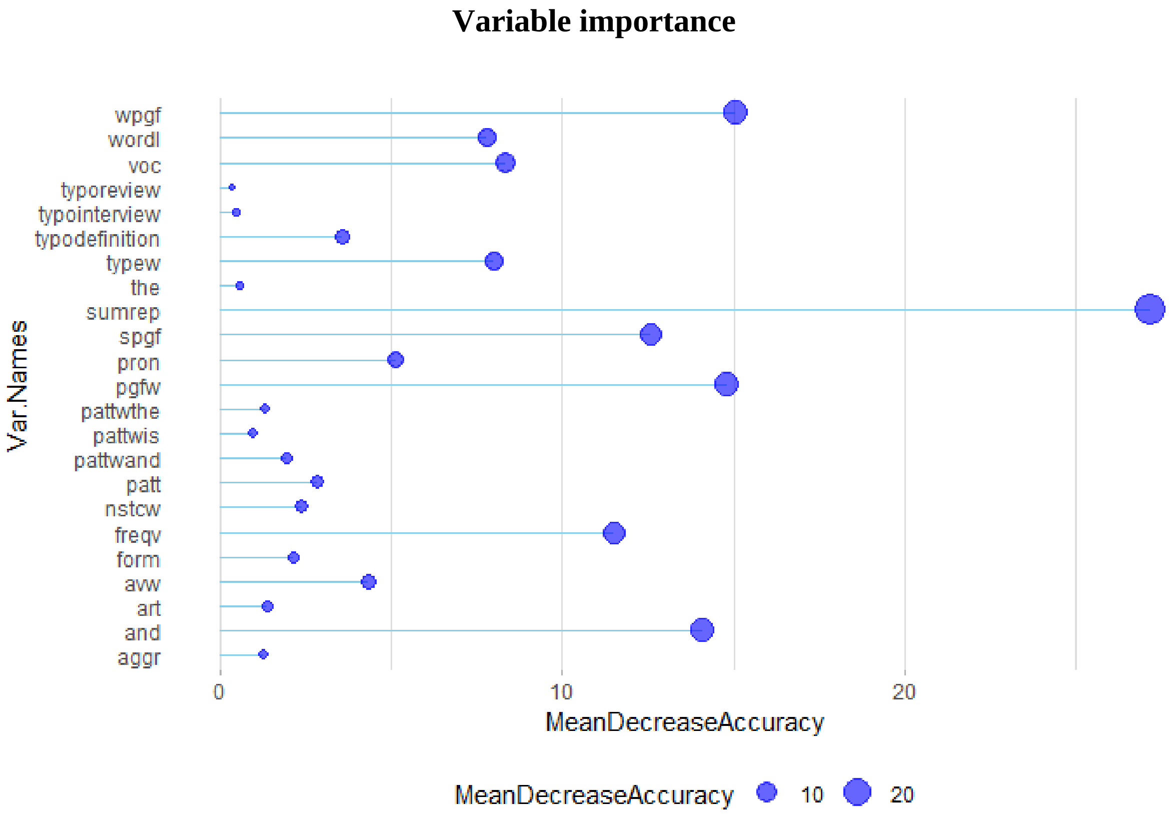


Figure 3.5: 23 most important variables in RF model

The following Table 3.7 provides a comprehensive overview of the accuracy and kappa scores for each mtry setting. This information allows us to assess the performance of the model under different variable selection scenarios, aiding us in determining the optimal mtry value for our classification task.

Table 3.7: RF model performance table

mtry	Accuracy	Kappa
2	0.806	0.612
12	**0.827**	**0.654**
22	0.825	0.650
32	0.817	0.635
42	0.817	0.635

The confusion matrix (Table 3.8) presented is the average of the confusion matrices obtained from 5-fold cross-validation, which was repeated 10 times. Once again, the predictions are well balanced, and we observe that the majority of the predictions fall into the true negative category (the model correctly identifies text generated by human sources). The accuracy rate of the model would be 0.89 and 0.77 for each category (Human, AI).

Table 3.8: RF Confusion Matrix

	Reference	
Prediction	**0**	**1**
0	44.4	11.6
1	5.6	38.4

3.2.3 Logistic Regression

In this section, we will delve into Logistic Regression, a powerful statistical modeling technique for binary classification problems. It is particularly suited for scenarios where we want to predict the probability of an event occurring based on a set of predictor variables.

The accuracy of the logistic model is 0.86, which means that it correctly predicts the outcome variable in 86% of cases. The Table 3.9 presents the results of the model analysis, including the independent variables and their respective odds ratio (OR), 95% CI and p-values. These p-values indicate the significance of each variable or each category of a nominal variable in relation to the outcome, providing valuable information for understanding the factors that contribute to (or predict) the occurrence of the outcome of interest.

Table 3.9: LR model summary results

Variable	**OR**	**95% CI**	**p-value**
sumrep_100	**1.32**	**(1.19, 1.48)**	**5.06e-07**
wpgf	**0.96**	**(0.94, 0.98)**	**0.00099**
and_100	**2.28**	**(1.63, 3.19)**	**1.43e-06**
typodefinition	**9.95**	**(2.05, 48.28)**	**0.00436**
typointerview	1.85	(0.34, 9.88)	0.47
typoreview	2.10	(0.46, 9.67)	0.34
freqv	**0.21**	**(0.06, 0.75)**	**0.02**

As we can see in the results, based on the p-value of the model's variables, we can say that the variables *sumrep_100* (sum of word repeats, in percentage), *wpgf* (the average number of words per paragraph), *and_100* (frequency of occurrences of the word *and*, in percentage), *typodefinition* (definition type texts), and *freqv* (standard deviation of the frequencies of each word) are statistically significant in the model, as their p-value is less than 0.05.

Furthermore, when it comes to the odds ratio (OR), values greater than 1 suggest a positive association between the predictor variable and the response variable. Conversely, values less than 1 indicate a negative association.

Regarding the variable *typo*, it is justified that it can be included in various models for possible interactions (e.g., in CART or Random Forest) or as a confounding variable in Logistic Regression. However, we find the obtained odds ratio value surprisingly high.

For the ROC curve of the model (see Figure 3.6), achieving a strong result indicates that the model possesses a high discriminative capability and can effectively differentiate between the positive and negative classes.

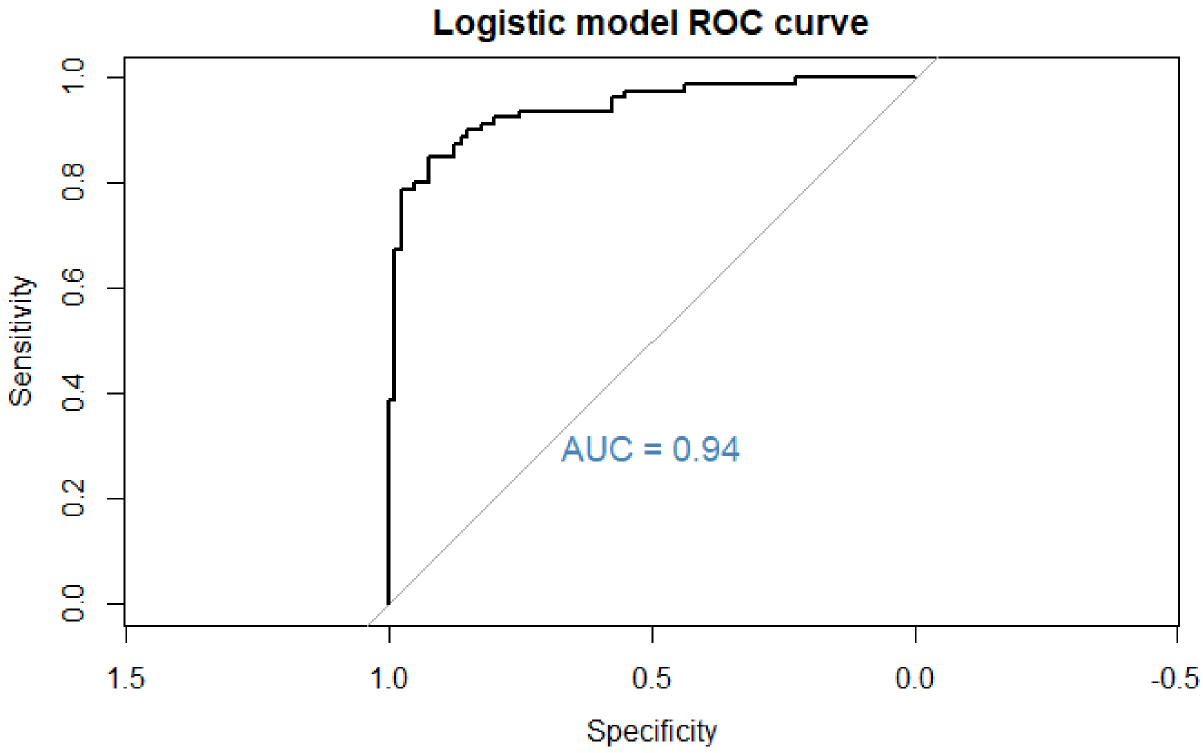


Figure 3.6: Logistic Model ROC curve and AUC

As we can observe in Table 3.10, the predictions remain perfectly balanced, with the majority still falling into the true negative category (indicating text generated by human sources, which is indeed the case). The proportion of true positive predictions (indicating text generated by AI) is similar, showing that the model performs reasonably well in identifying AI-generated text. The accuracy rate of the model would be 0.85 and 0.86 for each category (Human, AI).

Table 3.10: LR Confusion Matrix

	Reference	
Prediction	**0**	**1**
0	42.5	6.9
1	7.5	43.1

3.2.4 Support Vector Machines

In this study, we employed Support Vector Machine as a robust classification algorithm to predict the outcome of our dataset. We explored two variations of SVM to capture different patterns and relationships within the data: linear and radial.

Lineal

The linear SVM model demonstrated promising predictive performance. It achieved an accuracy of 82% on the test set with an optimal cost of 41, indicating that it correctly classified 82% of the instances. As mentioned in the methods section, a train/test approach was used for the model evaluation. Furthermore, upon examining the confusion matrix of the model (Table 3.11), we can observe that the classes are evenly distributed and well balanced. As expected, the true positive (the model predicts that it is a text generated by AI, and it is indeed generated by AI) and the true negative (the model predicts that it is a text generated by a human source, and it is indeed generated by a human source) have significantly higher values compared to other categories. This indicates that the model is performing quite well in accurately classifying the instances into their respective categories. The accuracy rate of the model would be 0.83 and 0.9 for each category (Human, AI).

Table 3.11: Linear SVM Confusion Matrix

	Reference	
Prediction	**0**	**1**
0	20	2
1	4	18

Radial

The radial SVM model achieved an accuracy of 45% on the test set with optimal cost and gamma values of 1 each. Nevertheless, this result is not very favorable compared to the performance of other models because all instances are predicted as AI-generated texts (Table 3.12). That model becomes useless for making predictions.

Table 3.12: Radial SVM Confusion Matrix

	Reference	
Prediction	**0**	**1**
0	0	0
1	24	20

Finally, as we have observed for SVM, the linear model achieves better results compared to the radial model. In Figure 3.7, we can visually compare their ROC curves.

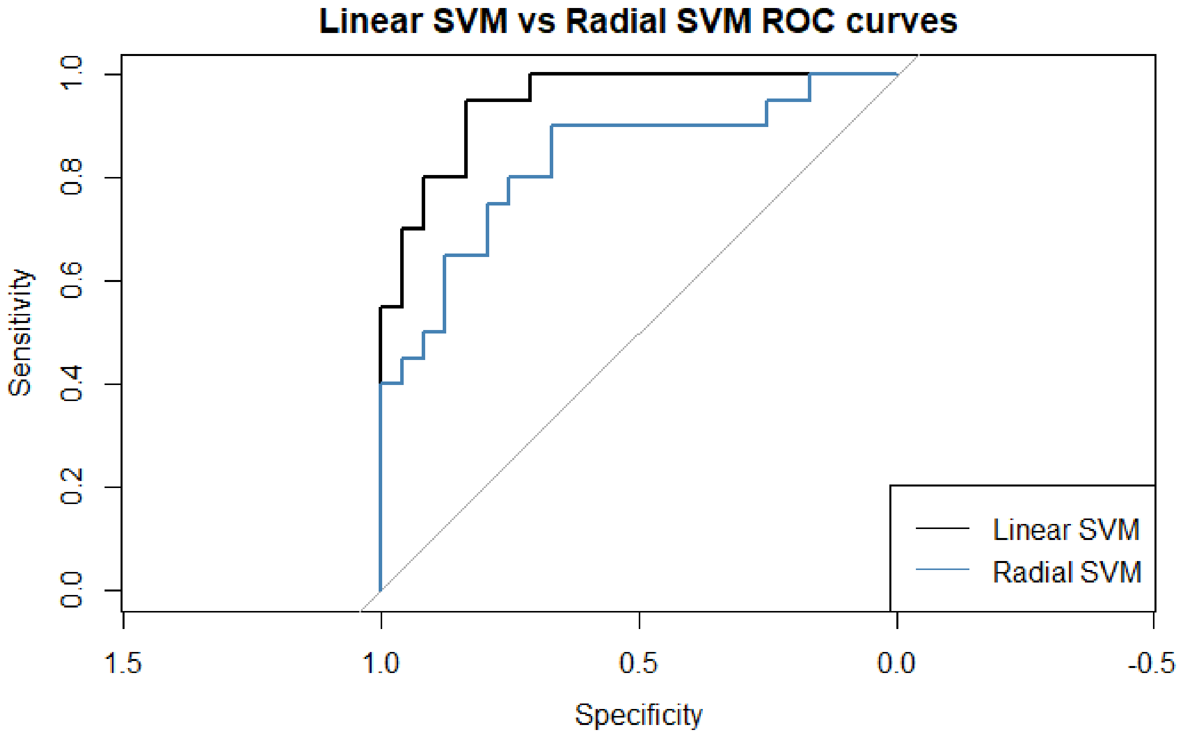


Figure 3.7: Linear SVM and Radial SVM ROC curves comparison

3.2.5 Comparison of the models

In this section, we present a comparative analysis of the five different implemented methods: Classification Tree, Random Forest, Logistic Regression, and both Support Vector Machine techniques. The aim of this analysis is to perform a comparison of their predictive capability as well as to identify the key variables that contribute to the predictive power of each model.

Table 3.13: Model performance metrics table

Model	Accuracy	Kappa	AUC	PPV	Recall
CT	0.79	0.65	0.84	0.87	0.69
RF	0.83	0.65	0.85	0.87	0.77
LR	0.86	0.75	0.94	0.86	0.86
SVM linear	0.82	0.72	0.94	0.81	0.90
SVM radial	0.45	0	0.86	0.45	1

We can observe in Table 3.13 that the accuracy values obtained for the Classification Tree, Random Forest, and Logistic Regression models are 0.79, 0.83, and 0.86, respectively.

Moreover, in addition to accuracy, the other metrics provide us with relevant information about the models. We observe that Logistic Regression has the highest accuracy, a better kappa (which accounts for the agreement between the model's predictions and the expected predictions by chance) and AUC (which evaluates the performance of a model's classification predictions). However, the Classification Tree and Random Forest achieve the highest PPV (the proportion of true positive predictions out of the total positive predictions made by the model), but the Logistic Regression model obtains the best recall (which measures the proportion of actual positive instances that are correctly identified by the model).

Additionally to the Classification Tree, Random Forest, and Logistic Regression models, we also evaluated the performance of Support Vector Machines with both linear and radial kernels. Upon analyzing the results of both the linear and radial SVM models, we observe notable differences in their performance. The linear SVM model has an accuracy of 82% and a high AUC of 0.94. Furthermore, the precision and recall values of 0.81 and 0.90, respectively. In contrast, the radial SVM model has lower predictive performance.

Table 3.14: Variables importance table

Classification Tree	Random Forest	Logistic Regression
sumrep	sumrep	sumrep
pgfw	pgfw	wpgf
wpgf	wpgf	and
freqv	and	typodefinition
typew	spgf	freqv

Upon examining the results, we can compare the most relevant variables for each model and identify similarities and differences (see Table 3.14). To the best of our knowledge, we do not have methods for SVM to obtain the relevant variables of the model. Therefore, they have not been added to the table.

Firstly, it is evident that the variable *sumrep* (sum of word repeats) holds the highest importance across all the models. Additionally, *pgfw* (ratio of the number of paragraphs to the total number of words) appears in Classification Tree and Random Forest, and *wpgf* (average number of words per paragraph) in all the three models. Moreover, the Random Forest and Logistic Regression align in recognizing the relevance of *and* (number of *and* words) and the variable *freqv* (the standard deviation of the frequencies of each word) has shown statistical significance in the first and third model. Lastly, each model has identified distinct variables as important, including *typodefinition* (definition type text), *spgf* (average number of sentences per paragraph), and *typew* (relationship between the number of unique word types and the total number of words).

3.3 Shiny

In this concluding section of the results analysis, we will delve into a practical example showcasing the functionality of the tool we have developed.

The AI-Detector PLUS (https://shiny-eio.upc.edu/pubs/ai-detector-plus/) is a Shiny application that allows users to enter a text and obtain the probability that the text was generated by an AI system. The interface consists of two main components: the input sidebar panel and the main panel that displays the prediction output (see Figure 3.8).

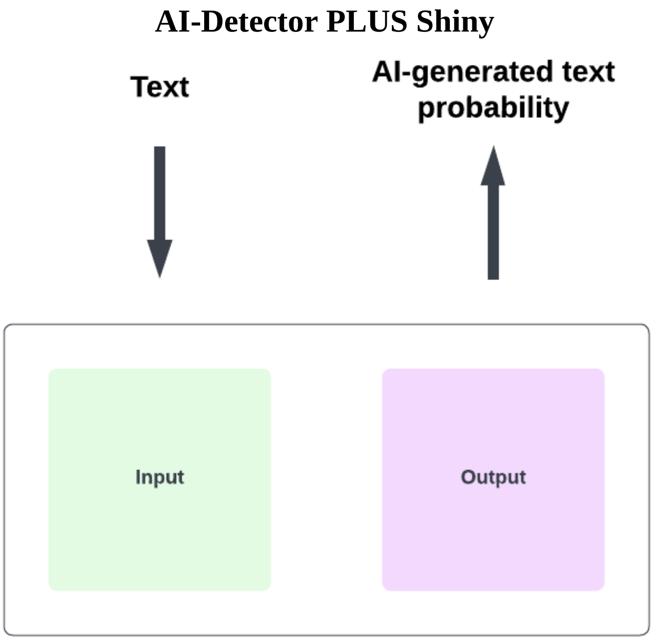


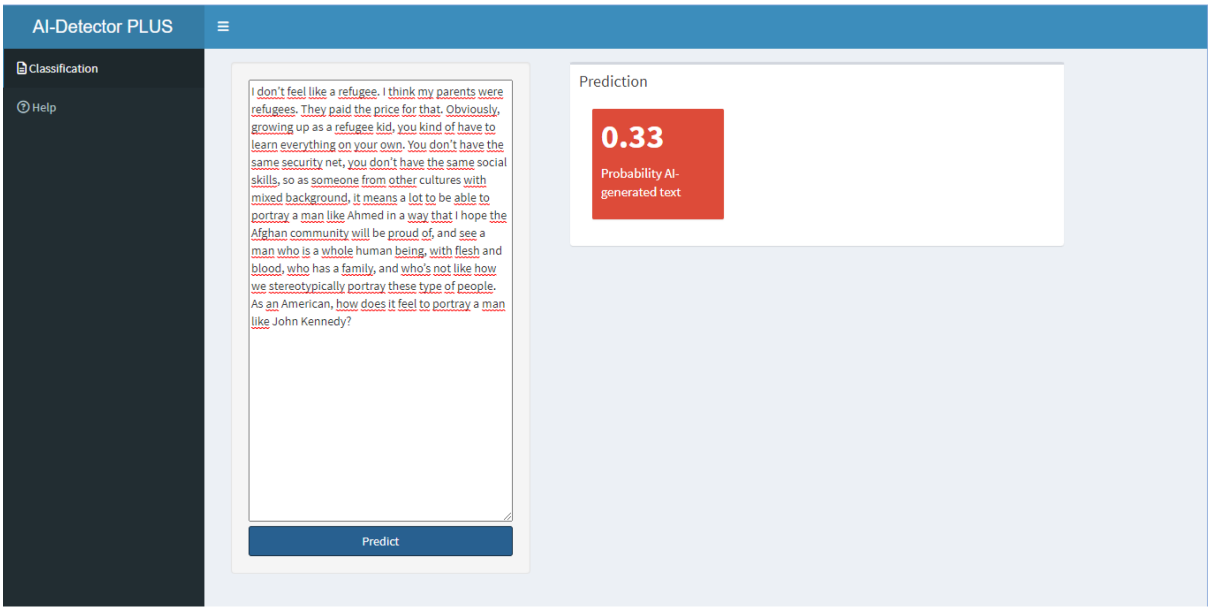
Figure 3.8: AI-Detector PLUS Shiny interface

The input sidebar panel contains a text input field where users can enter their text. They can write any text they want, and the application will analyze it using a pre-trained Random Forest model. The AI-Detector PLUS follows a client-server architecture, where the client represents the user interface (UI), and the server handles the logic and data processing. The server component is responsible for loading the pre-trained Random Forest model and performing the prediction based on the user's input. When the user clicks the *Predict* button, an event is triggered in the server, which then retrieves the text entered by the user from the input field. It processes the text by extracting various features, such as the length of the text, the number of words, repetition patterns, personal pronouns, articles, quantity vocabulary, number of paragraphs, number of sentences, and more.

Next, the server creates a new data frame with the extracted features. This data frame is then passed to the pre-trained Random Forest model, which was loaded earlier. The model predicts the probability that the text was generated by an AI system using the predict function. Specifically, it extracts the probability of the positive class (AI-generated) from the prediction result. Finally, the server returns the predicted probability to the UI, where it is displayed in the verbatim text output component. The user can view the probability and interpret it as the likelihood that the input text was generated by an AI system (shown in green if the probability is equal to or greater than 0.5, and in red otherwise).

As an additional feature, a small help section has been created using HTML to assist users in using the tool correctly in case of any doubts. Users simply need to click on the *Help* button, which will directly lead them to the instructions.

Shiny example I



I don't feel like a refugee. I think my parents were refugees. They paid the price for that. Obviously, growing up as a refugee kid, you kind of have to learn everything on your own. You don't have the same security net, you don't have the same social skills, so as someone from other cultures with mixed background, it means a lot to be able to portray a man like Ahmed in a way that I hope the Afghan community will be proud of, and see a man who is a whole human being, with flesh and blood, who has a family, and who's not like how we stereotypically portray these type of people. As an American, how does it feel to portray a man like John Kennedy?

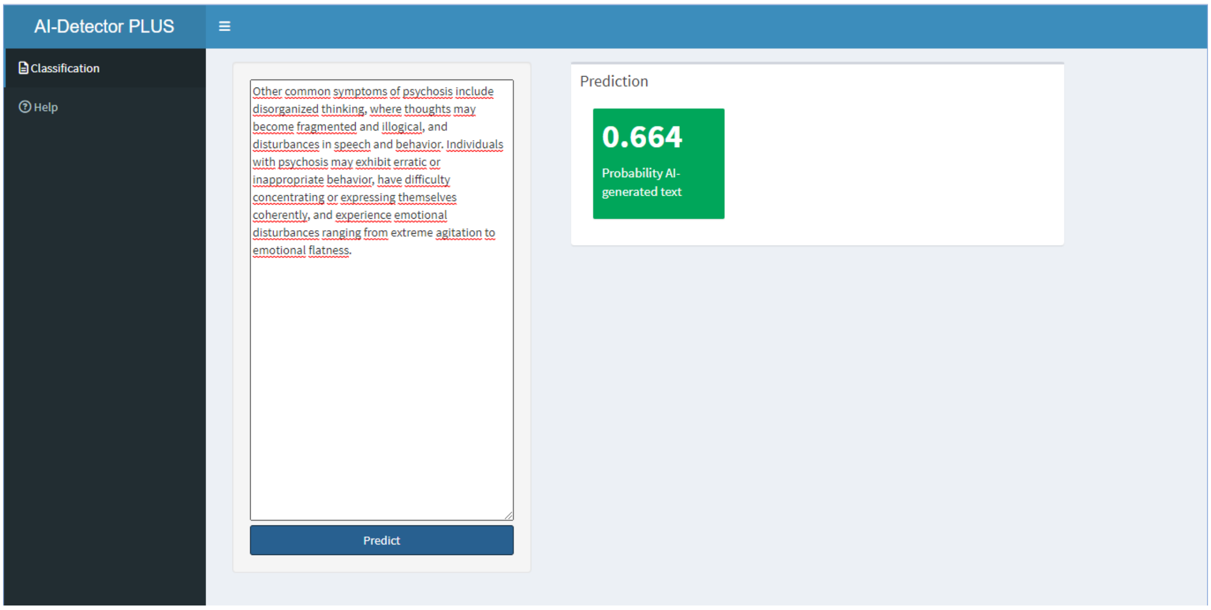
0.33

Probability AI-generated text

Figure 3.9: Human text AI-Detector PLUS example

In this scenario (Figure 3.9), the text has been generated by a human source, and indeed, the prediction indicating it originated from an AI source is low, with only a 33% probability.

Shiny example II



Other common symptoms of psychosis include disorganized thinking, where thoughts may become fragmented and illogical, and disturbances in speech and behavior. Individuals with psychosis may exhibit erratic or inappropriate behavior, have difficulty concentrating or expressing themselves coherently, and experience emotional disturbances ranging from extreme agitation to emotional flatness.

Figure 3.10: AI text AI-Detector PLUS example

In this other case (Figure 3.10), an AI-generated text has been analyzed, and the probability of it being an AI-generated text is over 66%.

4. CONCLUSIONS

In this project, we have developed and evaluated a text classification tool based on the classification models' algorithms to detect texts generated by AI. Our analysis focused on assessing the predictive performance of the model and identifying key factors that differentiate human-generated from AI-generated texts. The results of this research project might have practical applications in areas such as detecting fake news and identifying unauthorized use of text generation tools in academic environments.

Among all the models tested, the Random Forest model has exhibited promising performance in detecting texts generated by AI. Its high accuracy, substantial agreement level, strong discrimination ability (AUC), and reliable identification of AI-generated texts (PPV and Recall) collectively indicate its effectiveness in distinguishing between AI-generated and non-AI-generated texts. These results highlight the model's robustness and potential utility in real-world applications related to identifying AI-generated content. Although the Logistic Regression model also performed very well, it was decided to implement in the application the Random Forest method because one of the main variables that emerged in the Logistic Regression model was the text type, which cannot be calculated within the text introduced by the user into the application.

The differences in accuracy among the different models can be attributed to their underlying algorithms and characteristics. The Classification Tree model utilizes a hierarchical structure of decision rules to classify observations. However, it may suffer from overfitting, especially when the tree becomes complex and captures noise in the data. This can lead to a lower accuracy compared to other models. Random Forest, on the other hand, is an ensemble method that combines multiple decision trees. It utilizes bootstrapping and random feature selection to build a robust and diverse set of trees. This ensemble approach helps to reduce overfitting and improve generalization, resulting in a higher accuracy. Finally, Logistic Regression is a linear parametric model that estimates the probability of a binary outcome based on a set of predictor variables. It assumes a linear relationship between the predictors and the log-odds of the outcome and it can handle both continuous and categorical predictors, making it versatile. However, its performance heavily relies on the linearity assumption and may be limited when dealing with complex non-linear relationships in the data. However, it is important to note that the performance of these models can vary depending on the specific dataset and problem at hand.

Additionally to the Classification Tree, Random Forest, and Logistic Regression models, we also evaluated the performance of Support Vector Machines model with both linear and radial kernels. Upon analyzing the results of both the linear and radial SVM models, we observe notable differences in their performance. The linear SVM model demonstrates strong predictive capabilities and a high AUC. It exhibits balanced proportions in the confusion matrix, indicating its ability to capture patterns and nuances in the data. Furthermore, the precision and recall values highlight its effectiveness in correctly identifying texts generated by artificial intelligence. In contrast, the radial SVM model shows lower overall performance. It achieves smaller accuracy and Kappa values, suggesting poor agreement beyond what would be expected by chance, and a lower precision which implies a higher rate of false positives. However, the high AUC indicates a reasonable discriminatory power, and the excellent recall value indicates that the radial SVM model can accurately identify all texts generated by artificial intelligence.

In this research study, we aimed to identify key factors that differentiate human-generated texts from those generated by AI. Through our analysis, we have identified several noteworthy factors that play a crucial role in distinguishing between the two types of texts, encompassing linguistic characteristics, style patterns, vocabulary usage, and grammatical structure. Understanding these factors is essential for the effective detection of AI-generated texts.

First and foremost, the factor that consistently stood out across all models was the sum of repetitions, which was consistently higher in AI-generated texts. This factor exhibited remarkable significance in differentiating between human and AI-generated texts across the board. Its prominent role suggests that the level of repetition within a text serves as a valuable indicator for detecting AI-generated content. Other influential variables were the ratio of the number of paragraphs to the total number of words and the average number of words per paragraph in each text, which demonstrated a strong association with the identification of AI-generated texts. Additionally, the number of articles and *and* words emerged as key factors in our study. They revealed a significant relationship with text generation, further strengthening its relevance in distinguishing between human and AI-generated texts. Lastly, we observed that the standard deviation of the frequencies of each word provided valuable insights into differentiating the two text categories. Although not as prominent, it still contributed to the overall discriminative power of the models, suggesting that the number of unique words appearing more than five times in each text, which is closely linked to the sum of repetitions, the average number of sentences per paragraph, the number of *the* words, and the definition type text can provide additional insights when assessing the nature of the content.

As observed, the focus of the project has shifted towards a quantitative analysis of text rather than a more creative approach, as initially proposed with potential factors such as sentiment/emotion, creativity, formal language, style and tone, and coherence. These factors were challenging to analyze given our resources and were inherently abstract, considering that the project topic is relatively new and constantly evolving.

This research has yielded interesting and promising results that can contribute to the advancement of AI-generated text identification and this field of knowledge. However, it is worth noting that there are other experiments that have achieved higher accuracy levels (79%-90%). Nonetheless, our results are not significantly divergent from them [37]–[39].

The development of a user-friendly interface using Shiny deserves special mention in our study. The accessibility and ease of use of the interface play a crucial role in promoting the adoption and utility of the tool among a wider user base and the significance of accessibility cannot be overstated. The intuitive layout and straightforward functionalities empower users to input their text, execute predictions, and obtain results without any confusion or difficulty. Moreover, emphasizing ease of use has a direct impact on the overall user experience and satisfaction.

In the context of this project on AI-generated text detection, it is important to address the ethical considerations surrounding the use of AI. The responsible and ethical use of AI technologies plays a crucial role in ensuring the integrity and reliability of information. Detecting AI-generated texts serves as a valuable tool in mitigating potential risks associated with the dissemination of misleading or fabricated content. By identifying such texts, we contribute to maintaining transparency and trust in the information ecosystem. This not only benefits individuals seeking accurate and reliable information but also helps in safeguarding the reputation of organizations and promoting a healthy and informed society.

It is worth noting that the application of AI in our project has extended beyond the realm of text detection. It has aided in language translation and content creation, aiding in the development of effective communication across different languages. This underscores the potential for AI technologies to enhance productivity, cross-cultural understanding, and global collaboration.

By acknowledging the ethical considerations and harnessing the positive potential of AI, we can strive for a balanced approach that maximizes the benefits while mitigating the associated risks. Through responsible use and ongoing ethical evaluation, we can ensure that AI technologies, such as AI-generated text detection, contribute to a more reliable and trustworthy information landscape while promoting the well-being of individuals and societies.

Finally, although this has been an extensive project with interesting results, there have been some limitations in conducting the work, primarily due to time constraints, as well as limited conditions and possibilities. However, these limitations pave the way for future research directions:

- Dataset size and diversity: it is important to consider that the performance of the classification model can be affected by the size and diversity of the dataset used. In future research, it would be interesting to explore larger and more diverse datasets including different text sources such as songs. Furthermore, considering the calibration of the classification models and the validation of the Logistic Model may provide additional insights.

- Generalization to other languages: the study focused on recognizing AI-generated text in a particular language (English). It would be useful to investigate the applicability of the models in other languages, which may pose additional challenges due to linguistic and cultural differences. Examining the use of paradigms in different languages provides a broader understanding of their effectiveness and usefulness in different contexts.

- Acquisition of more recent data: models and algorithms are constantly improving in AI. It would be worth considering having the most recent and up-to-date information to display the latest types and structures of AI-generated text. This allows us to assess the performance of the models in a rapidly changing environment and maintain the validity and relevance of the findings.

- Other sampling techniques evaluation: in addition to the models used in this study, there are other classification techniques and schemes that could be explored. Investigating the application of models such as neural networks, gradient boosting, or deep learning methods could expand the options and allow for more comprehensive comparisons among different classification approaches in the identification of AI-generated texts.

- Expansion of the Shiny application: although favorable results have been obtained, the AI-detector PLUS can always be improved. By enhancing the techniques of the models and optimizing performance, the application can be expanded to include other algorithms such as Logistic Regression, after observing its good results. Additionally, the aesthetic aspects can be improved to achieve a more visually appealing design.

By addressing these limitations and exploring future research avenues, we can further advance our understanding of AI-generated text and improve the effectiveness of classification models and applications in this field.

5. BIBLIOGRAPHY

[1] A. Bashir, S. Bashir, K. Rana, P. Lambert, and A. Vernallis, “Post-COVID-19 Adaptations; the Shifts Towards Online Learning, Hybrid Course Delivery and the Implications for Biosciences Courses in the Higher Education Setting,” *Front Educ (Lausanne)*, vol. 6, Aug. 2021, doi: 10.3389/feduc.2021.711619.

[2] J. Huang, S. Saleh, and Y. Liu, “A review on artificial intelligence in education,” *Academic Journal of Interdisciplinary Studies*, vol. 10, no. 3, pp. 206–217, May 2021, doi: 10.36941/AJIS-2021-0077.

[3] “OpenAI.” https://openai.com/ (accessed Apr. 14, 2023).

[4] D. M. Markowitz, J. T. Hancock, and J. N. Bailenson, “Linguistic Markers of AI-Generated Text Versus Human-Generated Text: Evidence from Hotel Reviews and News Headlines.”

[5] N. Curtis, “To ChatGPT or not to ChatGPT? The Impact of Artificial Intelligence on Academic Publishing,” *Pediatric Infectious Disease Journal*, vol. 42, no. 4. 2023. doi: 10.1097/INF.0000000000003852.

[6] A. Lermann Henestrosa, H. Greving, and J. Kimmerle, “Automated journalism: The effects of AI authorship and evaluative information on the perception of a science journalism article,” *Comput Human Behav*, vol. 138, 2023, doi: 10.1016/j.chb.2022.107445.

[7] V. E. Gunser, S. Gottschling, B. Brucker, S. Richter, D. C. Çakir, and P. Gerjets, “The Pure Poet: How Good is the Subjective Credibility and Stylistic Quality of Literary Short Texts Written with an Artificial Intelligence Tool as Compared to Texts Written by Human Authors?,” in *Proceedings of the 44th Annual Meeting of the Cognitive Science Society: Cognitive Diversity, CogSci 2022*, 2022. doi: 10.18653/v1/2022.in2writing-1.8.

[8] L. Illia, E. Colleoni, and S. Zyglidopoulos, “Ethical implications of text generation in the age of artificial intelligence,” *Business Ethics, Environment and Responsibility*, vol. 32, no. 1, 2023, doi: 10.1111/beer.12479.

[9] “TV3.” Accessed: Apr. 14, 2023. [Online]. Available: https://www.ccma.cat/324/espanya-investiga-chatgpt-per-una-possible-violacio-de-la-proteccio-de-dades-dels-usuaris/noticia/3223411/

[10] “TV3”, Accessed: Apr. 14, 2023. [Online]. Available: https://www.ccma.cat/324/com-sha-de-regular-la-intelligencia-artificial-com-chatgpt-els-escenaris-a-europa/noticia/3222649/

[11] A. T. Jebb, L. Tay, W. Wang, and Q. Huang, “Time series analysis for psychological research: Examining and forecasting change,” *Front Psychol*, vol. 6, no. JUN, 2015, doi: 10.3389/fpsyg.2015.00727.

[12] "Article", Accessed: Jun. 16, 2023. [Online]. Available: https://www.instride.com/insights/gender-diversity-in-the-workplace/

[13] "Amazon." https://www.amazon.com/ (accessed Jun. 15, 2023).

[14] "Goodreads".

[15] Ö. Aydin and E. Karaarslan, "OpenAI ChatGPT Generated Literature Review: Digital Twin in Healthcare." [Online]. Available: https://ssrn.com/abstract=4308687

[16] T. Susnjak, "ChatGPT: The End of Online Exam Integrity?," Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.09292

[17] Q. Zhou, B. Li, L. Han, and M. Jou, "Talking to a bot or a wall? How chatbots vs. human agents affect anticipated communication quality," *Comput Human Behav*, vol. 143, Jun. 2023, doi: 10.1016/j.chb.2023.107674.

[18] S. Mitrović, D. Andreoletti, and O. Ayoub, "ChatGPT or Human? Detect and Explain. Explaining Decisions of Machine Learning Model for Detecting Short ChatGPT-generated Text," Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.13852

[19] "RPackage." https://cran.r-project.org/web/packages/available_packages_by_name.html (accessed Jun. 16, 2023).

[20] "R." www.R-project.org/ (accessed Apr. 14, 2023).

[21] "Rstudio", Accessed: Jun. 16, 2023. [Online]. Available: https://posit.co/downloads/

[22] T. Daniya, M. Geetha, and K. S. Kumar, "Classification and regression trees with gini index," *Advances in Mathematics: Scientific Journal*, vol. 9, no. 10, pp. 8237–8247, 2020, doi: 10.37418/amsj.9.10.53.

[23] "Caret Package", Accessed: Jun. 16, 2023. [Online]. Available: https://www.jstatsoft.org/article/view/v028i05

[24] L. Breiman, "Random Forests," 2001.

[25] "Random Forest", Accessed: Jun. 16, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[26] M. P. LaValley, "Logistic regression," *Circulation*, vol. 117, no. 18. pp. 2395–2399, May 2008. doi: 10.1161/CIRCULATIONAHA.106.682658.

[27] *Logistic Regression*. Accessed: Jun. 16, 2023. [Online]. Available: https://books.google.es/books?hl=ca&lr=&id=64JYAwAAQBAJ&oi=fnd&pg=PR13&dq=logistic+regression&ots=DtjR1W5qoM&sig=QrLdU8-FH5M1gIP81iNiG9gpris#v=onepage&q=logistic%20regression&f=false

[28] Y. Yan *et al.*, "Comparison of standard and penalized logistic regression in risk model development," *JTCVS Open*, vol. 9, pp. 303–316, Mar. 2022, doi: 10.1016/j.xjon.2022.01.016.

[29] G. Heinze and M. Schemper, "A solution to the problem of separation in logistic regression," *Stat Med*, vol. 21, no. 16, pp. 2409–2419, Aug. 2002, doi: 10.1002/sim.1047.

[30] A. Karatzoglou, D. Meyer, W. Wien, and K. Hornik, "Journal of Statistical Software Support Vector Machines in R," 2006. [Online]. Available: http://www.jstatsoft.org/

[31] “Support Vector Machines”, Accessed: Jun. 16, 2023. [Online]. Available: https://books.google.es/books?hl=ca&lr=&id=HUnqnrpYt4IC&oi=fnd&pg=PA1&dq=support+vector+machine&ots=gakJzu-nQi&sig=7OpB6ptSJk7AHQJVJ87yTXp72u4#v=onepage&q=support%20vector%20machine&f=false

[32] Y. Guo, Z. Zhang, and F. Tang, “Feature selection with kernelized multi-class support vector machine,” *Pattern Recognit*, vol. 117, Sep. 2021, doi: 10.1016/j.patcog.2021.107988.

[33] M. J. Warrens, “Five Ways to Look at CohenÃ¢Â€Â™s Kappa,” *J Psychol Psychother*, vol. 05, no. 04, 2015, doi: 10.4172/2161-0487.1000197.

[34] A. Ben-David, “About the relationship between ROC curves and Cohen’s kappa,” *Eng Appl Artif Intell*, vol. 21, no. 6, pp. 874–882, Sep. 2008, doi: 10.1016/j.engappai.2007.09.009.

[35] “Package ‘shiny’ Type Package Title Web Application Framework for R,” 2022.

[36] “Caret Package.” https://topepo.github.io/caret/ (accessed Jun. 17, 2023).

[37] OpenAI, “AI Classifier .” https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text (accessed Jun. 08, 2023).

[38] S. Mitrović, D. Andreoletti, and O. Ayoub, “ChatGPT or Human? Detect and Explain. Explaining Decisions of Machine Learning Model for Detecting Short ChatGPT-generated Text,” Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.13852

[39] J. Guerrero, “Detecting AI Generated Text Using Neural Networks,” 2023. [Online]. Available: https://digitalcommons.tamusa.edu/srs_2023

[40] “CRAN.” https://cran.r-project.org/ (accessed May 26, 2023).

[41] “R documentation.” https://rdocumentation.org/ (accessed May 26, 2023).

6. APPENDIX

The purpose of this appendix is to provide additional information and details about the research process and methodology used in the study.

6.1 Acronyms

This glossary serves as a reference guide, ensuring clarity and understanding of the terminology used in the context of this subject matter.

Term	**Meaning**
EDA	Exploratory Data Analysis
Min	Minimum
1st Qu.	First quartile
3rd Qu.	Third quartile
Max	Maximum
CI	Confidence interval
SD	Standard Deviation
CT	Classification Tree
CART	Classification and Regression tree
RF	Random Forest
LR	Logistic Regression
SVM	Support Vector Machines
PLR	Penalized Logistic Regression
RBF	Radial Basis Function
PPV	Positive Predictive Value
NPV	Negative Predictive Value
TP	True Positives
TN	True Negative
FN	False Negatives
FP	False Positives
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve

6.2 Packages

In this section, we present an overview of the R packages utilized in our study obtained from the CRAN repository [40]. These ones provide specialized tools and functions and the information regarding them were obtained from [40] the RDocumentation [41]. Next, we will explain the mentioned packages.

Package	Description
caret	Comprehensive toolkit for machine learning and predictive modeling. It provides a unified interface and a wide range of functionalities for data preprocessing, feature selection, model training, tuning, and evaluation. The caret package offers support for numerous machine learning algorithms, including decision trees, random forests, support vector machines, gradient boosting, and many more. It provides a consistent syntax for training and evaluating models across different algorithms, simplifying the modeling process.
DescTools	Comprehensive toolkit for descriptive statistics and exploratory data analysis. It provides a wide range of functions to summarize, visualize, and manipulate data. The package offers various descriptive statistics functions, including measures of central tendency (mean, median, mode), dispersion (variance, standard deviation, range), quantiles, and percentiles. It also includes functions for calculating skewness, kurtosis, and other distributional properties.
dplyr	Provides a set of functions for data manipulation, organized around the central concept of a data frame. It offers a consistent set of verbs that help to solve the most common data manipulation challenges including filtering, selecting, arranging, mutating, and summarizing data. The package is designed to work seamlessly with other packages in the tidyverse, making it a popular tool for data wrangling tasks.
e1071	Comprehensive toolbox for statistical modeling and machine learning. It provides a wide range of functions and algorithms for various tasks, including classification, regression, clustering, and support vector machines.
ggplot2	Provides a flexible and intuitive approach to creating visually appealing and informative plots. With ggplot2, users can construct a wide range of visualizations, including scatter plots, line plots, bar charts, histograms, and more. It emphasizes the concept of aesthetics, where data attributes like color, shape, size, and transparency can be mapped to variables, facilitating the exploration of multidimensional

	data. It also supports the use of facets, which allows for the creation of small multiples, enabling the visualization of different subsets of data simultaneously.
knitr	Powerful tool for dynamic report generation and reproducible research. It allows users to combine code, text, and output into a single document, making it easier to create reports, presentations, and manuscripts that seamlessly integrate data analysis and visualizations.
lintr	Tool that helps identify and report coding style and potential programming errors in R code. It provides a set of static code analysis rules that check for common issues, such as unused variables, missing whitespace, inconsistent indentation, and more. By analyzing R code, the lintr package promotes best coding practices, improves code readability, and helps identify potential bugs or inefficiencies.
openNLP	Toolkit that provides a wide range of natural language processing (NLP) functionalities. It allows users to perform tasks such as tokenization, part-of-speech tagging, named entity recognition, chunking, and parsing on textual data. The package offers pre-trained models for multiple languages and supports the training of custom models.
pROC	Tool for evaluating and analyzing binary classification models, with a focus on ROC analysis. It provides functions and methods for calculating a variety of performance measures, generating ROC curves, and comparing multiple models.
quanteda	Package for managing and analyzing textual data. It provides a suite of text processing and analysis functions that allow users to perform tasks such as corpus creation, document feature extraction, frequency analysis, and text classification. The package is designed to be flexible and scalable, making it suitable for a wide range of applications in fields such as linguistics, political science, and media studies.
RandomForest	Popular and versatile tool for building random forest models, an ensemble learning method that combines multiple decision trees to make predictions. It offers a robust implementation of the random forest algorithm, which is known for its ability to handle complex datasets and capture non-linear relationships between variables.
rpart	Widely used package for building decision trees and conducting recursive partitioning analysis. It provides an implementation of the Classification and Regression Trees algorithm. The rpart package allows users to build decision trees by recursively partitioning the data based on selected variables and splitting criteria. It supports both

	classification and regression tasks, making it suitable for a wide range of predictive modeling problems.
rpart.plot	Visualization tool specifically designed for decision trees built using the rpart package. It provides enhanced plotting capabilities that allow for clear and aesthetically appealing visualizations of decision trees. Rpart.plot offers several advantages over the default plot method in rpart and more flexibility in customizing the appearance of the decision tree, allowing users to adjust colors, fonts, and line styles. It also supports additional visual elements, such as node labels, variable labels, and branch labels, which can enhance the interpretability of the tree.
RColorBrewer	Useful tool for generating visually appealing color palettes for data visualizations. It provides a collection of color palettes with a range of color schemes that are optimized for presenting categorical and sequential data.
ROCR	Powerful tool for evaluating and visualizing the performance of binary classification models. It provides a wide range of functions and methods for assessing the accuracy, discrimination, and calibration of classification models.
SentimentAnalysis	Provides a simple interface to various sentiment analysis algorithms and pre-built sentiment lexicons. It allows users to analyze the sentiment of text data at different levels of granularity, including document-level, sentence-level, and aspect-level analysis. The package provides functionalities for sentiment polarity classification, emotion classification, and sentiment strength scoring. It can be useful in various applications such as social media monitoring, customer feedback analysis, and opinion mining.
sentimentr	Powerful tool for sentiment analysis and emotion detection in text data. It provides a range of functions that analyze the sentiment of individual words, sentences, or entire documents. The package leverages a pre-trained sentiment lexicon and employs linguistic rules to calculate sentiment scores, indicating the positivity or negativity of the text.
shiny	Powerful web application framework that allows users to create interactive and dynamic web applications directly from R code. It enables the creation of web-based dashboards, data visualization tools, and interactive data analysis interfaces without the need for HTML, CSS, or JavaScript knowledge.
shinydashboard	Provides a framework for creating interactive dashboards with a visually appealing and user-friendly layout. It is built on top of the

	Shiny web application framework, allowing you to create professional-looking dashboards for data visualization and analysis. With this package, you can design your dashboard using a combination of predefined layout components, including header, sidebar, and body sections. The header typically contains the title and navigation elements, while the sidebar offers options for user interaction and selection. The body section is where the main content and visualizations are displayed.
spellcheckr	Corrects misspelled words in English using an algorithm based on Peter Norvig's spell correction approach, with the ability to handle up to three edits. The algorithm calculates the probability of the intended correction for a given word by comparing all possible candidate corrections against the original word and selects the correction with the highest probability.
stepPlr	Tool for L2 penalized logistic regression for both continuous and discrete predictors, with forward stagewise/forward stepwise variable selection procedure.
stringr	Provides a cohesive set of functions for working with strings of text. It offers easy-to-use tools for string manipulation, including pattern matching, string replacement, and splitting strings into substrings based on specific delimiters.
stargazer	Tool for creating well-formatted and customizable summary tables from various model objects. It simplifies the process of presenting regression models, time series models, and other statistical models in a tabular format for inclusion in reports, articles, or presentations.
syuzhet	Used for sentiment analysis and emotion detection in text. It provides a set of functions that extract different types of sentiment or emotion signals from text data. The package uses a dictionary-based approach to identify sentiment and emotion words and then aggregates them to calculate the overall sentiment or emotion score for a text. The package includes several pre-trained sentiment and emotion dictionaries, but users can also create their own custom dictionaries or modify existing ones.
tidyr	Allows for easy data reshaping and tidying. It provides a set of functions that help to organize messy datasets into a consistent and clear format. The package focuses on two main types of data reshaping: gathering and spreading. The gathering function takes multiple columns and gathers them into key-value pairs, while the spreading function takes key-value pairs and spreads them into multiple columns.

tidytext	Package for text mining and natural language processing tasks. It provides a framework for tidy data principles and functions to perform tasks such as tokenization, stemming, and stopword removal. The package also includes datasets and functions for sentiment analysis and other common text analysis tasks.
tidyverse	Collection of powerful and interconnected R packages designed for data manipulation, visualization, and analysis. It follows a unified philosophy known as the "tidy data" principles, which emphasizes consistency, simplicity, and ease of use.
tm	Provides a set of functions for Text Mining. It enables users to preprocess, manage, and analyze textual data. The package includes tools for text cleaning, transformation, tokenization, stopword removal, stemming, and creation of document-term matrices. It also offers functionalities for text visualization, feature selection, topic modeling, sentiment analysis, and classification.
udpipe	Provides tokenization, parts-of-speech tagging, and dependency parsing for raw text in many languages. It is based on the UDPipe project, which combines a neural network architecture with a pipeline of finite-state automata to achieve fast and accurate results on natural language processing tasks.

6.3 Functions

We will present a compilation of essential R functions that were utilized in the context of our project. They have been carefully selected to facilitate efficient and accurate data processing, enabling us to extract meaningful insights and draw informed conclusions from our dataset.

Function	**Description**
actionButton	Creates a button input control in a Shiny application and allows users to trigger specific actions or events when the button is clicked.
arrange	Part of the dplyr package in R and used to reorder rows in a data frame based on one or more variables. It allows you to sort the rows in ascending or descending order, facilitating data exploration and analysis.
as.character	Used to convert objects of various types into character (text) format. It coerces the input object into a character vector, allowing you to work with and manipulate the data as text.
auc	Part of various packages in R, including pROC, ROCR, and pROC, and used to calculate the Area Under the Receiver Operating Characteristic (ROC) Curve (AUC). The AUC is a widely used metric to assess the overall performance of a binary classification model.
boxplot	Used to create box-and-whisker plots in R. It is a graphical tool for visualizing the distribution of a continuous variable or numerical data across different categories or groups.
chisq.test	Used to perform a chi-square test of independence or goodness-of-fit. It is part of the stats package and is primarily used for categorical data analysis.
confusionMatrix	Part of the caret package in R and used to create a confusion matrix, which is a table that summarizes the performance of a classification model. It helps evaluate the accuracy and effectiveness of a model by comparing predicted class labels with actual class labels.
Corpus	Part of the tm (text mining) package in R. It creates a corpus, which is a collection of text documents, used for text mining and natural language processing tasks. It creates a corpus object from a collection of plain text documents or from a folder containing plain text files.
dashboardBody	Used to create the body section of a dashboard in a Shiny application. The body section is where the main content and visualizations of the dashboard are displayed.
dashboardPage	Used to create a complete dashboard layout for a Shiny application. It provides a structured framework that includes a header, sidebar, and

	body sections, allowing you to organize and present information in a visually appealing manner.
dashboardSidebar	Used to create the sidebar section of a dashboard in a Shiny application. The sidebar provides a space for interactive controls and navigation options, allowing users to customize the displayed content or perform specific actions.
expand.grid	Part of the caret package in R and used to compute variable importance measures for a trained model. Variable importance measures help identify the relative importance or contribution of different predictor variables in predicting the target variable.
filter	Part of the dplyr package in R and is used to extract rows from a data frame that meet specified conditions. It allows you to filter and subset the data based on one or more criteria, facilitating data exploration and analysis.
ggplot	The primary function in the ggplot2 package, which is a popular data visualization library in R. It is used to create a wide range of high-quality and customizable plots, such as scatter plots, line plots, bar plots, histograms, and more.
max	Used to find the maximum value in a given set of numbers or a vector. The max function takes one or more numeric arguments and returns the maximum value in the input set. If the input set contains any NA values, the function returns NA unless the na.rm argument is set to TRUE.
names	Used to get or set the names of an object, such as a vector, list, or data frame. If no argument is passed to names, it returns the names of the object. If an argument is passed, it sets the names of the object to the specified values.
nchar	Used to count the number of characters in a string. The function returns the number of characters in each element of a character vector or a single string. The function takes one main argument, which is the character vector or string to be counted. It can also take an optional type of argument to specify the type of character encoding used in the string. By default, type is set to "unknown", which means that the function will try to automatically detect the character encoding.
nsentence	Assuming that it is a custom function that is defined to count the number of sentences in a given text, it would take a string of text as input and return the number of sentences in that text. The function could use regular expressions or other techniques to identify the sentence boundaries and count the number of matches to determine the sentence count.

observeEvent	Used to define reactive behavior based on specific events or triggers in a Shiny application. It allows you to specify an event and the corresponding actions to be executed when that event occurs.
predict	A generic function that is used to make predictions based on a trained model. It allows you to apply a trained model to new data and obtain predictions or estimated values for the target variable.
pull	part of the dplyr package in R and used to extract a single column from a data frame as a vector. It allows you to easily extract and work with a specific variable from a larger dataset.
read_excel	From the readxl package in R that allows you to read Excel files into R as data frames. The read_excel function can read both .xls and .xlsx files. It is a flexible function that can handle many variations in data formatting and allows for the selection of specific sheets and cells.
renderPrint	Used to display printed output within a Shiny application. It takes an R expression as its argument and renders the printed result on the user interface.
roc	Part of the pROC package in R and used to calculate and plot Receiver Operating Characteristic (ROC) curves. ROC curves are commonly used in binary classification problems to assess the performance of a classification model by evaluating the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) at different classification thresholds.
rpart.plot	Part of the rpart.plot package in R and used to visualize decision trees generated using the rpart package. It provides an enhanced and more customizable way to plot decision trees, making it easier to interpret and analyze the tree structure.
sapply	Part of the apply family of functions in R, which is used to apply a function to a vector or a list of values and return a simplified result. Specifically, it returns a vector or matrix, depending on the input, after applying a function to each element of a vector or list. The sapply function takes two main arguments: the input vector or list to apply the function to and the function to be applied.
sd	Used to calculate the standard deviation of a numeric vector or a column in a data frame. It measures the dispersion or variability of the values in the dataset.
select	Part of the dplyr package in R and used to select specific columns (variables) from a data frame. It allows you to subset and extract only the columns you need for analysis, visualization, or further data manipulation.

sentiment	Used to analyze text data and classify it according to the sentiment expressed in the text, such as positive, negative, or neutral. This function assigns a sentiment score to each sentence in a text based on the presence of positive and negative words in the sentence. The sentiment score ranges from -1 (most negative) to +1 (most positive), with 0 indicating a neutral sentiment. It takes one main argument, which is a character vector of text data to be analyzed. It also provides various options to customize the analysis, such as custom sentiment lexicons and handling of negation and punctuation.
seq	Used to generate a sequence of values with a specified start, end, and increment. It is commonly used to create a vector of equally spaced values or to define a sequence of numbers for looping or indexing purposes.
shinyApp	The main function used to create a Shiny application in R. It allows you to combine UI (user interface) and server components to build interactive web applications.
slice_head	Part of the dplyr package in R and used to select the first n rows of a data frame. It is helpful when you want to examine a subset of the data or extract a specific number of top rows from a larger dataset.
sort	Used to sort elements in a vector or a data frame in ascending order. It rearranges the elements of the input object and returns a new object with the sorted values.
str_count	Part of the stringr package in R. It is used to count the number of matches of a pattern in a string or vector of strings and returns the number of occurrences of a specified pattern in each element of the character vector or string. It takes two main arguments: the string or vector of strings to search, and the regular expression pattern to count.
str_extract_all	Part of the stringr package in R and used to extract all occurrences of a pattern in a character vector or string. This function takes two main arguments: the character vector or string to search, and the regular expression pattern to extract. The function returns a list of character vectors, with each element of the list corresponding to one element of the input vector or string. Each element of the output list contains all the non-overlapping matches of the pattern in the corresponding input string.
strsplit	Used to split a character string into substrings based on a specified delimiter or pattern. It takes a character vector as input and returns a list of character vectors, where each element of the list corresponds to the split substrings.

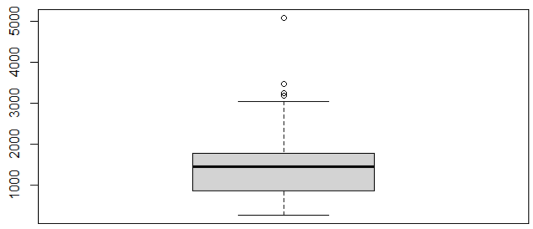
summary	A generic function that provides a concise summary of the statistical properties and characteristics of an object. Its behavior varies depending on the type of object being summarized.
svm	A powerful and widely used machine learning algorithm for classification and regression tasks.
table	Used to create a tabulation of the counts of each unique value or combination of values in a vector or set of vectors. The function takes one or more vector arguments and returns a table object containing the counts of each unique value or combination of values in the input vectors.
t.test	Used to perform a t-test, which is a statistical test to compare the means of two groups and assess whether they are significantly different from each other. Commonly used when dealing with numerical data and wanting to determine if there is evidence of a significant difference between the means of two populations or samples.
tm_map	In the tm package in R used for text mining and natural language processing tasks. It is used to apply a specific text transformation function to a corpus or document. It takes a corpus or document as input and applies a series of text transformations such as stemming, stopword removal, case conversion, tokenization, or regular expression replacement to the text. The output is a transformed corpus or document that can be used for further analysis.
train	Is part of the caret package in R and is used for training machine learning models. It provides a unified interface for model training, making it easier to compare and evaluate different algorithms.
trainControl	Part of the caret package in R and used to define the control parameters for model training and evaluation. It provides a flexible way to specify various settings related to cross-validation, resampling, and performance metrics.
unlist	Used to simplify a list by removing all levels of nesting and returning a vector that combines all the elements from the original list. It takes one main argument, which is the list to be simplified and can also take additional arguments to specify how the list should be simplified.
varImp	Used to define the control parameters for model training and evaluation. It provides a flexible way to specify various settings related to cross-validation, resampling, and performance metrics.
which.max	Used to identify the index or position of the maximum value within a numeric vector or array. It returns the index of the first occurrence of the maximum value.

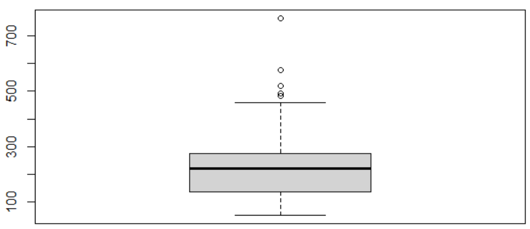
6.4 Exploratory data analysis

In addition to the previously presented descriptive statistics, we also generated boxplots to visually explore the distribution of our numerical variables. Boxplots provide a quick and effective way to identify potential outliers, visualize the variability and skewness of the data, and detect potential differences in central tendency between groups. For brevity, we present the boxplots in the appendix of this paper.

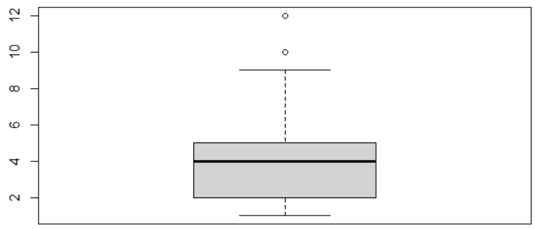
Boxplots numeric variables

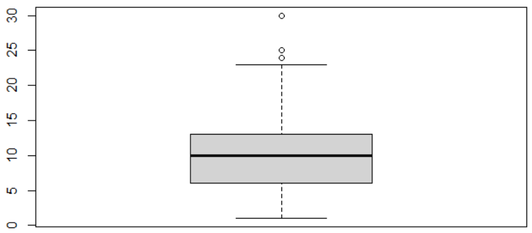
The number of letters (lengthl) and the number of words (lengthw).



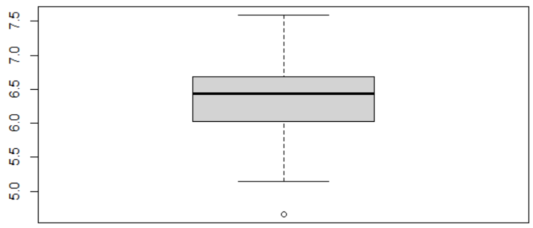


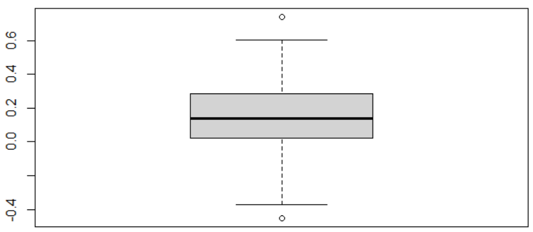
The number of paragraphs (pgf) and the number of sentences (nstc).



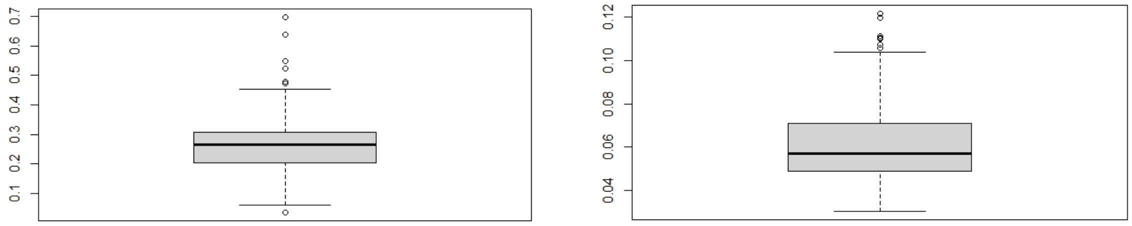


The number of characters per word (wordl) and the sentiment score mean (sentm)

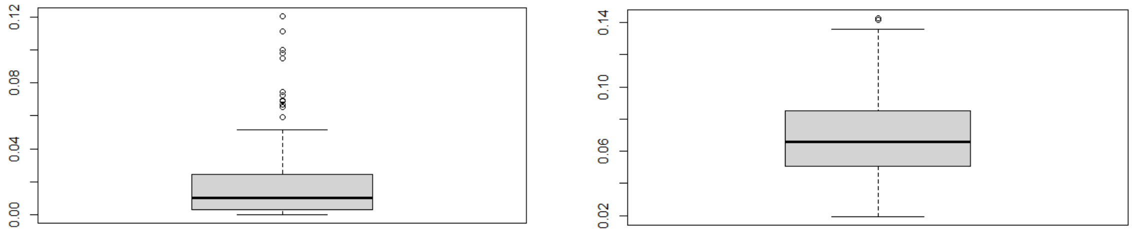




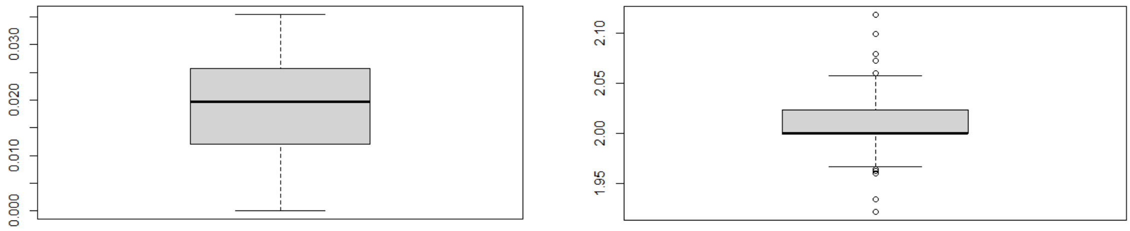
The sentiment score standard deviation (sentsd) and the maximum frequency of any word that appears multiple times (patt)



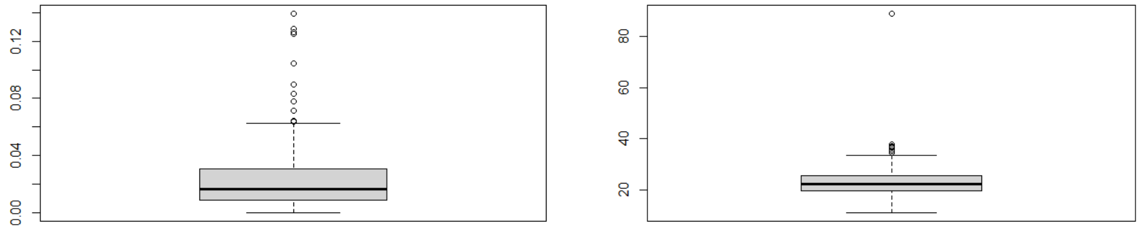
The number of pronouns (pron) and the number of articles (art)



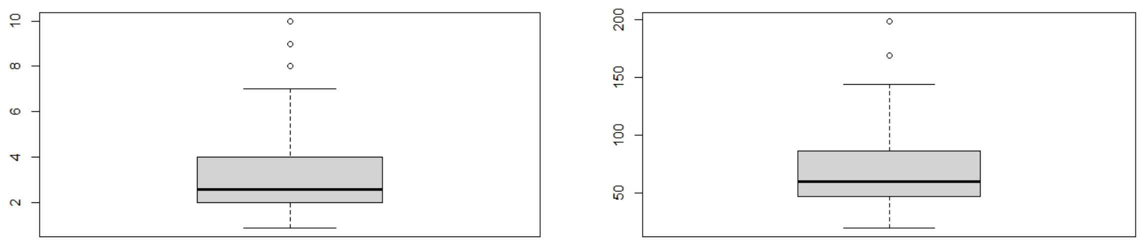
The number of unique words that appear more than five times (voc) and the number of informal language forms (form)



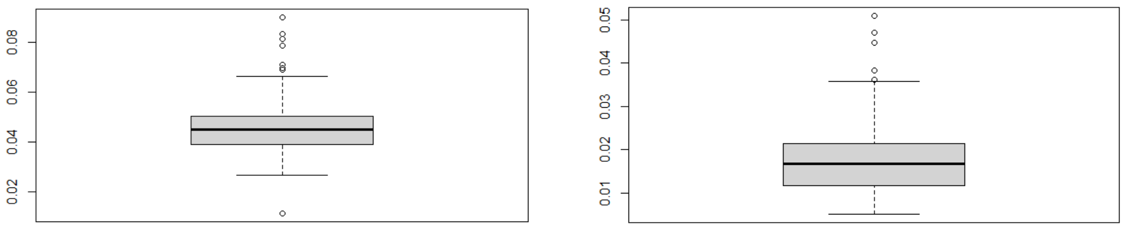
The number of occurrences of words related to aggressive language (aggr) and the average number of words per sentence (avw)



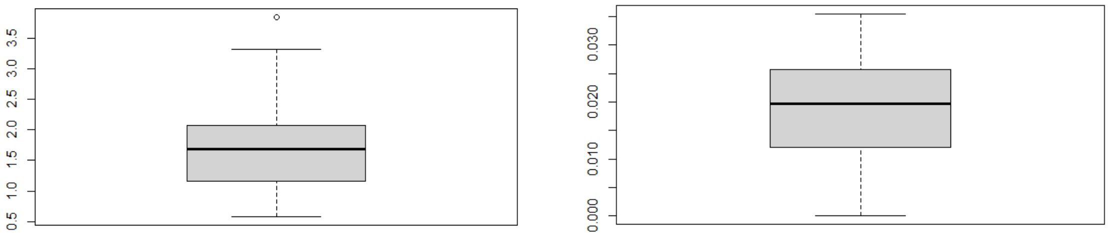
The average number of sentences per paragraph (spgf) and the average number of words per paragraph (wpgf)



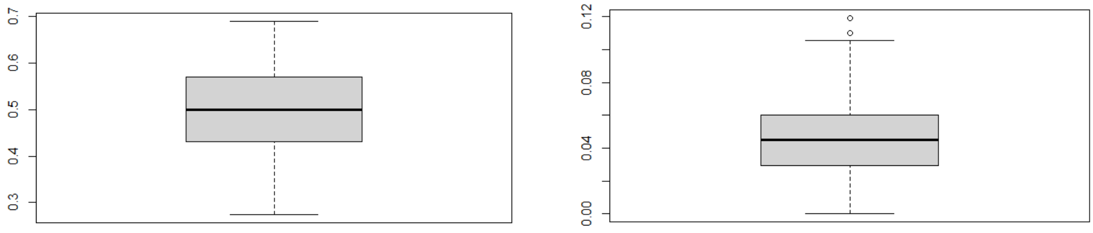
The ratio of the number of sentences to the total number of words (nstcw) and the ratio of the number of paragraphs to the total number of words (pgfw)



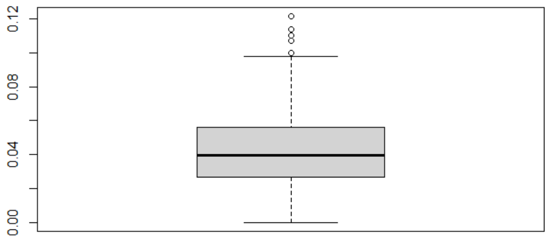
The standard deviation of the frequencies of each word (freqv) and the relationship between the number of unique word types and the total number of words (*typew*)



The sum of word repeats for each text, standardized by dividing it by the total number of words (*sumrep*) and the number of *and* words (*and*)

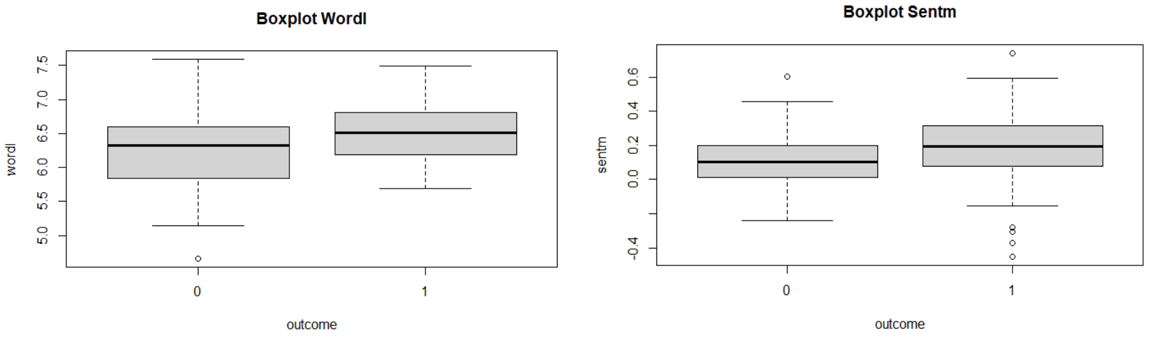


The number of *the* words (*the*)

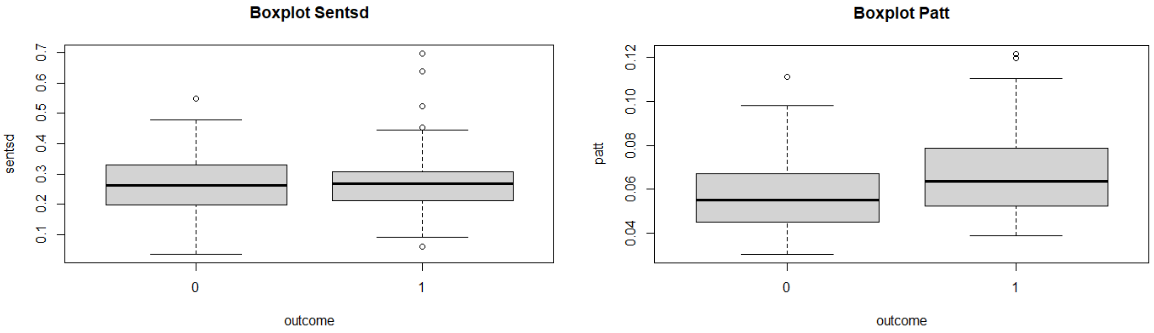


Boxplots outcome with numeric variable

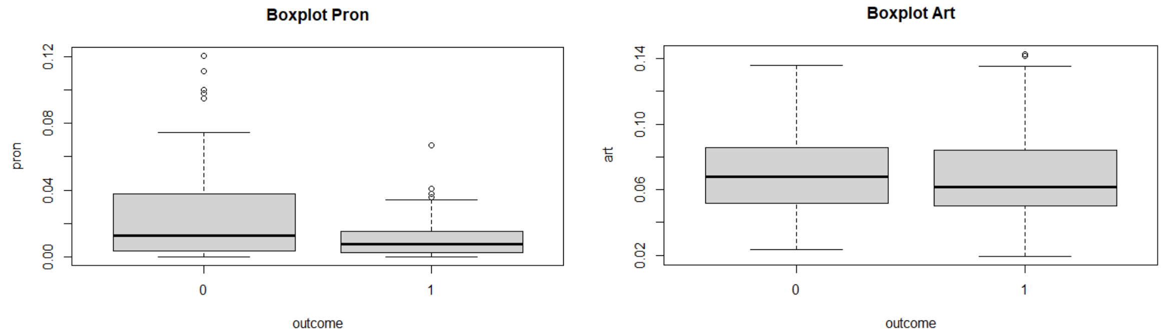
Distributions of the variables *wordl* and *sentm* across the two levels of the variable *outcome*



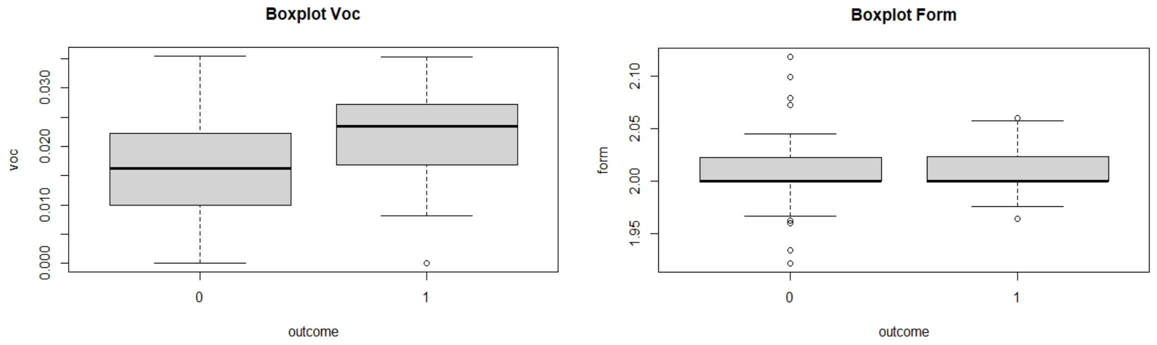
Distributions of the variables *sentsd* and *patt* across the two levels of the variable *outcome*



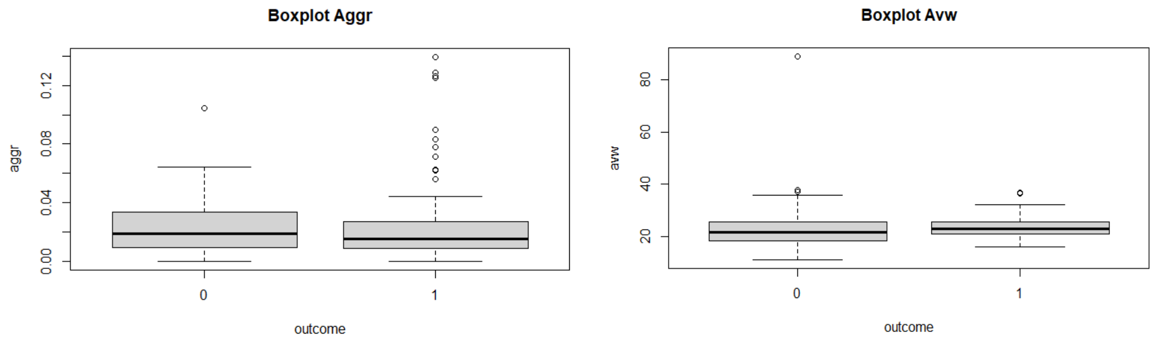
Distributions of the variables *pron* and *art* across the two levels of the variable *outcome*



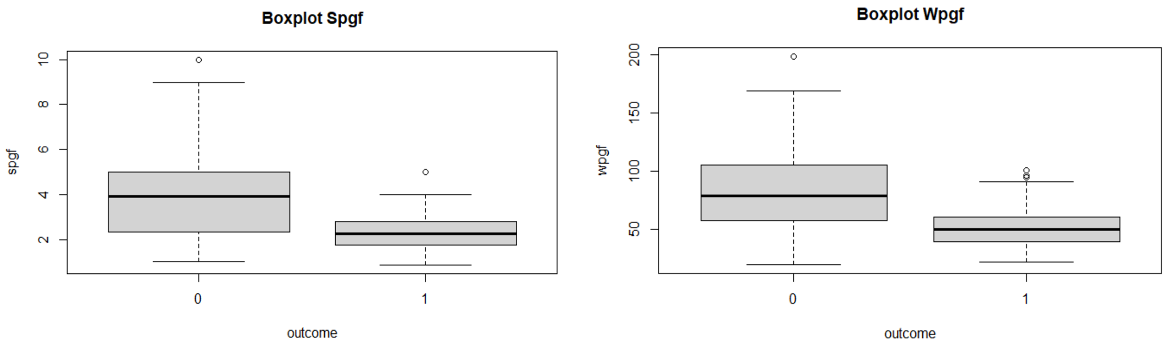
Distributions of the variables *voc* and *form* across the two levels of the variable *outcome*



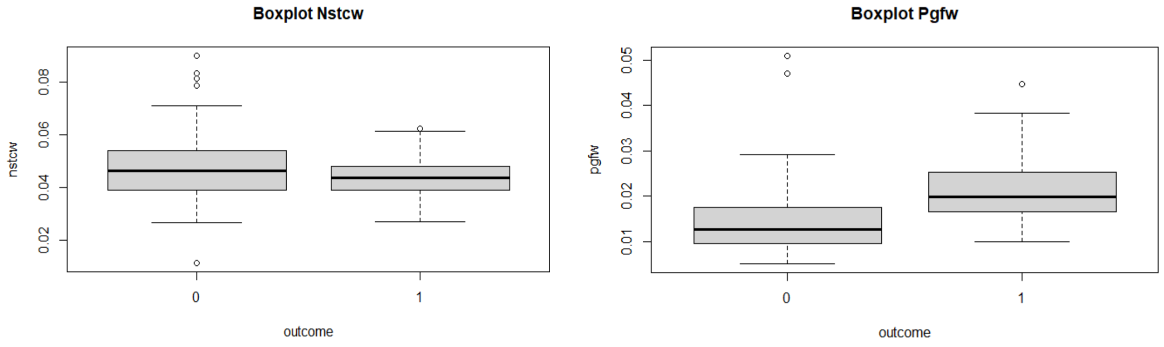
Distributions of the variables *aggr* and *avw* across the two levels of the variable *outcome*



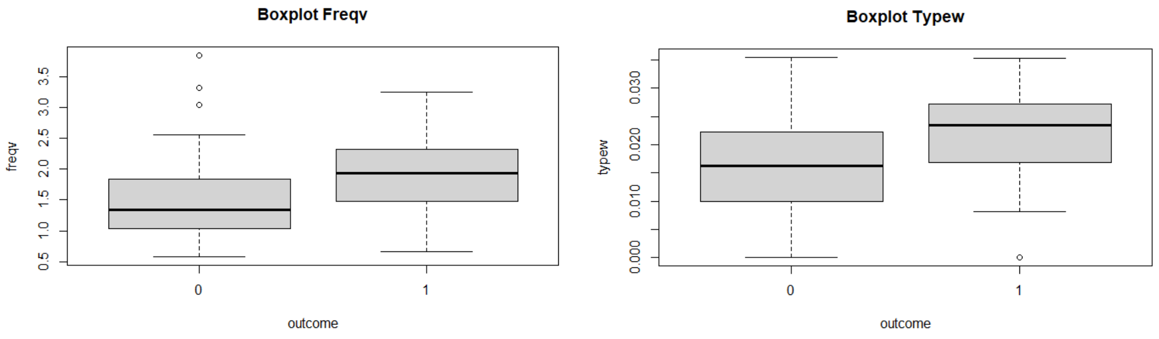
Distributions of the variables *spgf* and *wpgf* across the two levels of the variable *outcome*



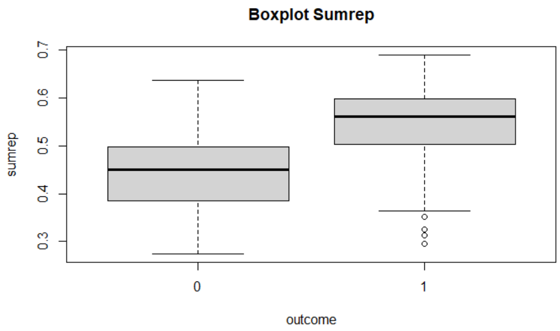
Distributions of the variables *nstcw* and *pgfw* across the two levels of the variable *outcome*

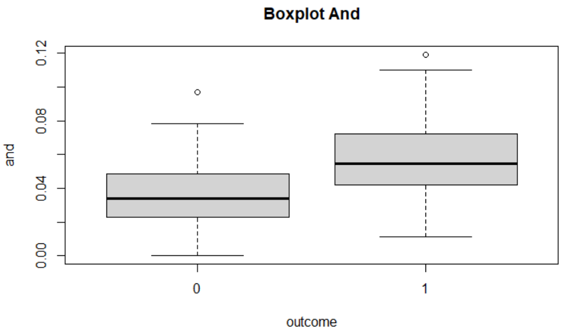


Distributions of the variables *freqv* and *typew* across the two levels of the variable *outcome*

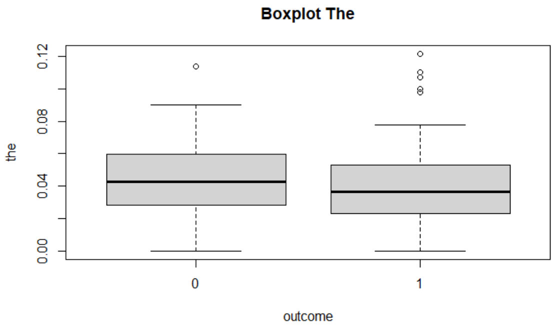


Distributions of the variables *sumrep* and *and* across the two levels of the variable *outcome*



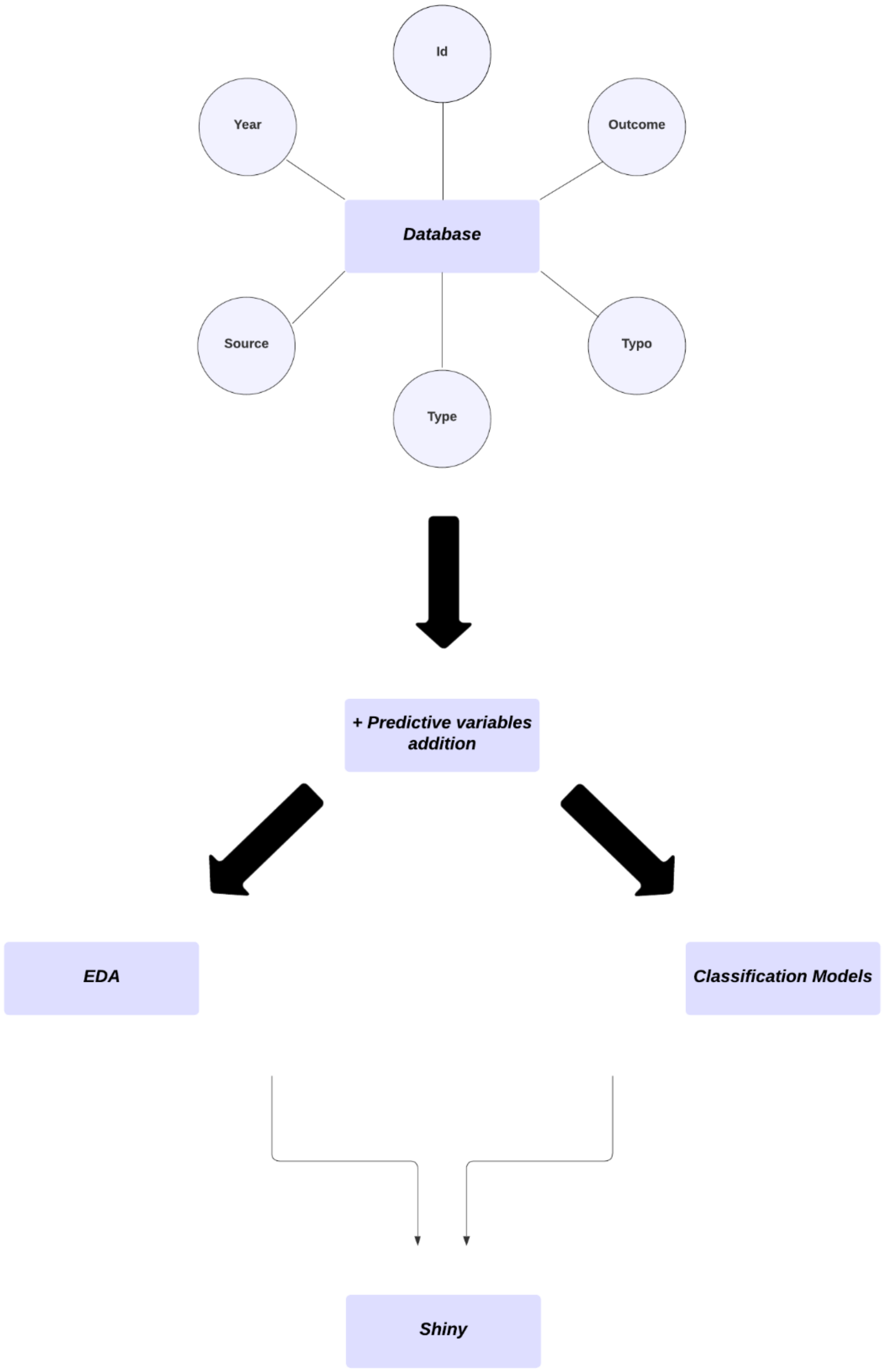


Distribution of the variable *the* across the two levels of the variable *outcome*



6.5 R code

Finally, presented below is a flowchart of the R code used in the project for a better understanding and visualization of the analysis process. The corresponding code is available on GitHub for reference and accessibility: https://github.com/yaizabravo/TFG.


Id
Year
Outcome
Database
Source
Typo
Type
+ Predictive variables addition
EDA
Classification Models
Shiny