UNIVERISTATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ ENGLEZĂ

LUCRARE DE LICENŢĂ

Detectarea stirilor false si evaluarea credibilității stirilor prin intermediul Machine Learning

Conducător științific
Lect. Dr. Bădărînză Ioan

Absolvent
Bîrligă Alexandru-Robert

2022

BABEȘ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE IN ENGLISH

DIPLOMA THESIS

Fake News Detection and News Reliability Evaluation by Means of Machine Learning

Supervisor
Lect. Dr. Bădărînză Ioan

Author
Bîrligă Alexandru-Robert

2022

Abstract

The phenomenon of fake news has been escalating alarmingly in the last few decades, since the outburst of the Internet and social media. The COVID-19 pandemic illustrates perfectly how dangerously and quickly lies, fabricated news, hoaxes and conspiracies can be disseminated at a global level, resulting into mass misinformation, revolt, paranoia and a steep division between people. Given that most people inform themselves about essentially everything through online sources, there is a great need for a collective endeavour to mitigate this issue. In this context, fake news detection has emerged as an increasingly researched field over the last decades.

This thesis examines several approaches for evaluating the reliability and deceitfulness of online news articles. Primarily, the methodology includes supervised machine learning algorithms, such as K-Nearest Neighbors, Decision Trees, Logistic Regression and Support Vector Machines, which are leveraged for classifying news articles by veracity, based upon their contents and headlines. The results were satisfactory, the best models achieving over 90% accuracy scores on the datasets, while also performing promisingly well on real-life inputs. Secondarily, some non-AI methods were also implemented, by means of web-scraping and crowdsourcing All of these news analysis procedures were subsequently brought together under a browser extension, with the final goal of providing users with an accessible tool for evaluating the reliability of any news article.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Contents

1 Introduction **3**

2 Core Concepts of Machine Learning **5**

2.1 Introduction 5

2.2 More on Machine Learning 7

2.2.1 Core Concepts of Supervised Learning 7

2.2.2 K-nearest Neighbors 9

2.2.3 Decision Tree 10

2.2.4 Logistic Regression 13

2.2.5 Support Vector Machine 14

3 Fake News Detection Methods Overview **17**

4 Application Development **19**

4.1 Functionalities 19

4.2 Architecture and Technologies 20

4.2.1 Front-end 20

4.2.2 UI Design 22

4.2.3 Back-end 23

4.2.4 Database 25

4.2.5 Testing 26

4.2.6 Management 26

4.3 Machine Learning Algorithms 27

4.3.1 Datasets 28

4.3.2 Data pre-processing 29

4.3.3 Feature Extraction 30

4.3.4 Training and Results 32

4.4 Source Reliability via Web-scraping 35

4.5 Crowdsourcing / User Feedback 37

5 Conclusion and Future Work **39**

6 References **41**

1 Introduction

Fake news is not simply a product of the last decades. It has been part of humanity for at least centuries. Its first significant outburst dates back to 1400s, when the printing press was invented. Ever since then, fabricated news have been more and more weaponized to manipulate people at increasingly larger scales. In 1835, The New York Sun published 6 articles presenting the discovery of life on the moon. Both World Wars were heavily marked by propaganda, and false and misleading news [1]. And more recently, the US 2016 presidential elections and the COVID-19 pandemic were subjects of large amounts of fake news.

Despite being so widespread, there is no globally agreed definition of the "fake news" term. Nonetheless, many sources provide definitions which have two primary elements in common: authenticity and intent. As for authenticity, fake news contain claims that are factually and verifiably false. As for intent, the major purpose of fake news is to deceive the consumers [2]. That being said, a concise definition of fake news could be formulated in the following manner:

Definition 1 *By fake news we mean any news item that is deliberately and factually untrue and misleading.*

Fake News Detection represents a new area of study. On the one hand, it emerged thanks to the continuous advancements in Artificial Intelligence and, specifically, machine learning supervised learning methods. On the other hand, it appeared as virtually a necessity, given the recent proliferation of the fake news phenomenon, on account of the simultaneous raise of digital media.

Digital and social media are environments in which the dissemination of information is more facile and rapid than at any point in history. Fabricated news have been heavily published and shared as a means of deception, on subjects of high impact, such as the 2016 US presidential elections and the COVID-19 pandemic.

At present, there are multiple fake news identification resources online. They each contribute in various manners to validating the veracity of news and debunking rumours, hoaxes, conspiracies [3]: PolitiFact and The Washington Post Fact Checker analyze the statements and claims of American politicians; Snopes, Full-

Fact, HoaxSlayer and GossipCop debunk fake articles containing fake news, gossips, hoaxes or celebrity rumors.

Although these online fact-checking resources have a notable and admirable contribution to combating fake news, the incredibly high rate at which fabricated information is produced and distributed through unreliable news agencies and suspicious social media accounts, makes the task of manual fake news detection impactful at only a very small scale. That is why, computer science, artificial intelligence and machine learning algorithms could have a more promising large-scale impact and efficiency.

Therefore, the aim of this thesis is to explore and implement a series of algorithms to aid users in protecting themselves against misinformation, by means of machine learning algorithms together with other computer science methods, such as web-scraping and crowdsourcing. As for the structure of the thesis, Chapter 2 provides a rundown of the machine learning notions and algorithms leveraged for the creation of some fake news detection content-based and headline-based classifiers. Chapter 3 reviews some fake news detection related research work, exploring several methodologies in which machine learning and deep learning can be utilized. Chapter 4 gives a comprehensive account of the whole application development process, including the decision-making process, the technologies employed, the architecture of the application, the machine learning algorithms used, how the news sources reliability was determined via web-scraping and how crowdsourcing / user feedback can also be utilized for building a reliability profile for news sources and authors, while also building a new dataset in parallel. Chapter 5 ends the thesis with some final concluding words and what wants to achieve as part of a future effort.

2 Core Concepts of Machine Learning

Machine learning and natural language processing are important components laying at the foundation of this thesis. Therefore, the current chapter provides an overview of some of the relevant fundamental concepts, required for understanding the thesis further on.

2.1 Introduction

Artificial intelligence (AI), originally coined as a term and founded as an academic discipline in the 1950s [4], has become a buzzword not only in the domain of computer science, but also in the popular culture. The growth in popularity and general interest is owed to the accelerated technological advancements and exponential increase in the volumes of data, which fueled the progression of AI from mostly bare theory to a gradual actuality.

There is no singular universally accepted definition for artificial intelligence. This term generated a considerable amount of differences and confusion among specialists[5]. There are numerous proposed definitions, but one that encapsulates to a certain extent a common essence encountered in a fair part of them could be formulated as follows:

Definition 2 *Artificial intelligence is a branch of computer science, concerned with developing computer programs that approach problems emulating the human model of thinking and its typical processes, such as learning, reasoning and self-correction.*

Another term coined in the 1950s is "machine learning". Machine learning is a subfield of artificial intelligence, based on the idea that computers can be programmed to "learn" from data and tackle tasks with minimal human intervention. A definition for this field of study, based on Tom Mitchell's proposal [6], is the following one:

Definition 3 *Machine learning studies the capability of a computer program to learn from experience E with respect to some task T and some performance mea-*

sure P, improving with experience E its performance on the task T, measured by P.

This definition is a more mathematical one, operating with three variables:

1. E (experience) stands for the dataset collected for the learning process, which is leveraged for extracting a general formula from a set of particular examples.

2. T (task) represents the problem which is desired to be solved via a machine learning algorithm.

3. P (performance measure) refers to a set of metrics applied on the machine learning model to mathematically quantify its performance.

Machine learning is a novel field which has been successfully implemented and notably pushed the boundaries in many real applications, including pattern recognition, speech recognition, audio processing, natural language processing (e.g. language translation, search results, smart assistants, text analysis) and computer vision (e.g. object detection, image classification, segmentation).

Out of these, natural language processing has probably the most applications in the area of fake news detection, the subject of this thesis.

Definition 4 *Natural language processing (NLP) is a field of study at the intersection of artificial intelligence and linguistics, which is comprised of a range of computational techniques for analyzing and representing human language [7].*

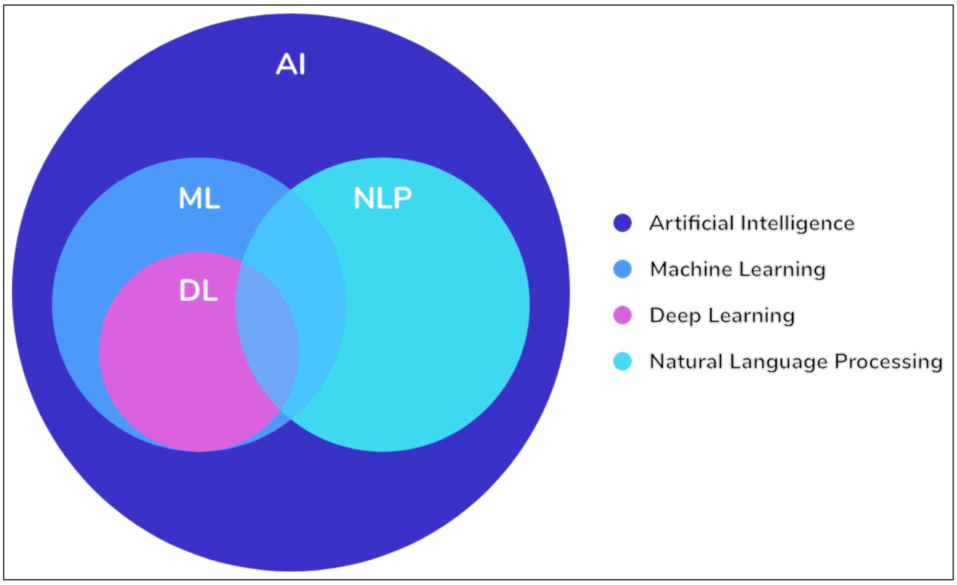


Figure 1: Relation between NLP and Machine Learning.

2.2 More on Machine Learning

2.2.1 Core Concepts of Supervised Learning

Machine learning algorithms can be divided into three categories: **supervised learning**, **unsupervised learning** and **reinforcement learning**. Supervised learning algorithms are trained on a set of input-output pairs, with the aim of creating a mathematical model to map any general input to an output. Conversely, unsupervised learning techniques deal with unlabelled data from which patterns are derived, through various clustering and association algorithms. Lastly, reinforcement learning alg involves algorithms learning by being rewarded for desired behaviors and punished for undesired ones.

In the context of fake news detection, the majority of research revolves around supervised learning algorithms, hence that is what this thesis will further focus upon.

Supervised learning operates with annotated (labeled) datasets during the training and testing process. Every element from the training dataset is called a feature vector, in which each dimension (i.e. feature) characterizes the input data from a certain aspect. The goal is to learn the correlation between the set of features and the labels for every training example, so that through generalization, the label of a potentially unmet input can be predicted. This process of identifying correlations, anomalies and patterns within a dataset is known as **data mining**.

There are generally no perfect models that can generate the correct output with 100% accuracy for all of the inputs. The discrepancies between the ground truth labels and the actual predictions of the model can be expressed by a **loss function**. Thereby, the purpose of the supervised learning algorithm can also be understood as the minimization of the loss function, which translates into the maximization of the accuracy and other performance metrics of the model.

Supervised learning algorithms can be split into two types: **classifications** and **regressions**. Both algorithms are used for making predictions, but they differ primarily in the type of the output produced. For a given input, classification algorithms predict discrete values, called classes or categories, from a finite set of values, to which the feature vectors are considered to belong. Contrarily, regression algorithms predict continuous values (integer or floating point values), which usually denote either quantities or sizes. For instance, let us consider weather prediction based on certain parameters, such as: humidity, temperature or atmospheric pressure. This task could be approached in two ways: predict if the weather is hot or cold (binary classification) or predict a precise value for the temperature, such as 24 degrees Celsius (regression).

As for fake news detection, in order to determine whether an article is fake or real, a classification algorithm is most suitable. The algorithms employed within this project are probabilistic classifiers, which are not only able to make a categorical prediction (i.e. fake or real), but they can also compute the probability distribution of a news article over the set of classes. Probabilities are more revealing than mere binary values, because they measure the confidence with which the classifier predicts a certain class.

Within this project, the following classification algorithms were used for the problem of fake news detection: **Support Vector Machine**, **Decision Tree**, **Logistic Regression**, **K-nearest Neighbors**.

2.2.2 K-nearest Neighbors

K-nearest Neighbors (KNN) is one of the simplest machine learning supervised algorithms, which is also known to be quite versatile and well-performing. It functions on the principle of proximity relative to the previous training examples. It can be used as a regression algorithm, but it is most encountered in classification problems. KNN is **non-parametric** and **lazy-learning** or **slow-learning**, the two primary properties of KNN.

By non-parametric, it is meant that the classifier does not make any prior strong assumptions about the underlying structure of the data by deciding beforehand the number of parameters of the mapping function which the algorithm models during training. Opposed to parametric algorithms, which can discard the training data after the model is build, KNN stores each training example as a parameter and makes all the upcoming predictions based on the k most similar training patterns performed earlier on. Practically, the sole assumption the model makes is the correlation or similarity between the feature vectors analyzed previously and the new training examples.

By lazy-learning or slow-learning, it is meant that KNN is technically not having an explicit training process and it does not learn from the training data immediately. During the training phase, the KNN algorithm simply stores in memory each of the feature vectors from the training dataset, relying on observable data similarities and distance metrics.

The KNN algorithm can be outlined by the following steps which are iteratively performed for each training or testing example:

1. Locate and store the feature vector on the graph, as a data point.

2. Calculate the distance between the current data point and the rest of the previous training examples. In order to calculate the distance, various formulas can be used, such as the Euclidean distance, the Manhattan Distance, the Minkowski distance or the cosine distance.

3. Consider the k nearest neighbor data point and count the frequency of each label.

4. Assign a class label to the current data point according to the majority vote rule: the label that is in the greatest proportion present around the data point, from the k considered closes ones.

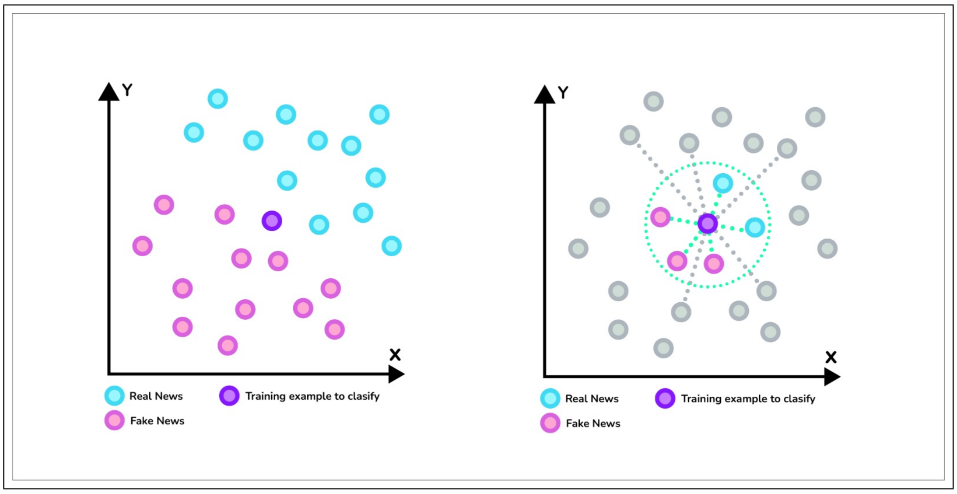


Figure 2: KNN algorithm selecting the K nearest neighbors.

The **advantages** of KNN is that it is relatively simple to implement, can learn non-linear decision boundaries, no training time, new data can be added anytime without a negative effect on the accuracy and there is only one hyperparameter to handle (i.e. k). As of **disadvantages**, does not work very efficiently on large datasets and it is sensitive to noisy data.

2.2.3 Decision Tree

Decision Tree is a supervised learning algorithm used for both classification and regression, having a tree-like structure, where the leaves correspond to output classes and the sub-branches represent series of decision rules, each leading finally to the prediction of an output class. The sub-branches are comprised of internal nodes, each of them containing different features of the dataset. Decision trees can be viewed as graphical representation covering all the solutions to a problem based on a specified set of conditions.

It is called a decision tree because comparable to a tree, it starts from the root node (representing the entire dataset) and expands to an increasing number of sub-branches gradually, learning from the past training examples. Each internal leaf stands for a decision rule. By decision rule, we mean a simple IF-THEN statement, consisting of a condition and a prediction. For instance, IF a person is 65 years old and is obese, THEN they have a high probability of hearth attack of diabetes.

There are two main operation performed on a decision tree, namely splitting (i.e. the operation of ramifying the root node or a decision node into two sub-branches) and pruning (i.e. the operation of compression of the size of the tree, by elimination unimportant or redundant sections of the tree). Also, by pruning one can reduce the complexity and overfitting from the tree.

A decision tree can be decomposed in the following steps:

1. Initialize the tree with only the root node, which contains the whole dataset.

2. Use an attribute selection measure to split the node into two sub-branches according to the best attribute.

3. Create a new decision node containing the best attributed identified at the prior step.

4. Recursively apply the first 3 steps to all the resulting subtrees.

5. Stop the recursivity as soon as the nodes cannot be classified further anymore and regard the final nodes as leaves.

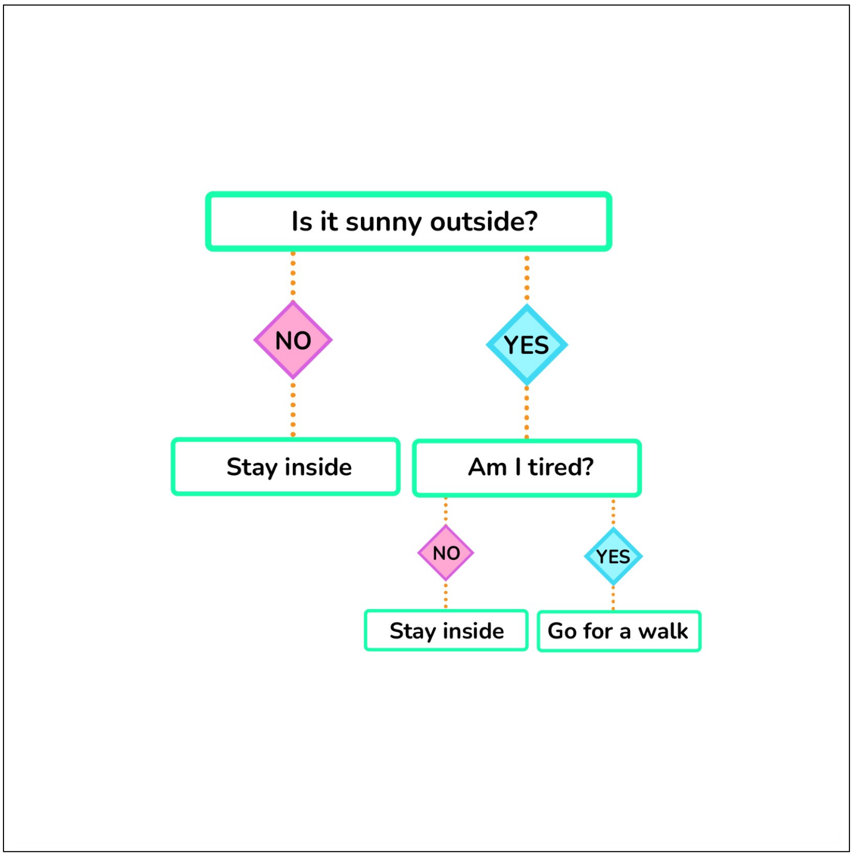


Figure 3: Small example of decision tree.

The pruning process mentioned above can be done in two ways: post-pruning (i.e. execute the pruning after the construction of the decision tree is completed, technique particularly used when the depth of the tree is large and shows signs of overfitting) or pre-prunning (i.e. execute the prunning some time before the completion of the decision tree, technique useful to prevent overfitting).

On the one hand, the upsides of decision tree include: less data preparation is needed (not critical to standardize and normalize the data), data scaling is not required and it has the ability to capture non-linear relationships. On the other hand, the downsides are that decision tree generally require more computational resources and execution time

2.2.4 Logistic Regression

Logistic regression is a supervised machine learning algorithm which is used for classifications and involves drawing a hypothesis function which best fits the categories/classes of a dataset.

Opposed to the linear regression, where the aim is to find a linear function, logistic regression looks for other types of function, such as the commonly utilized sigmoid/logistic function. The sigmoid function is a S-shaped curve, useful for binary classification tasks, given that the domain of the function can be the whole real values set and the output is either greater than and very close to 0 or less than and very close to 1. The sigmoid function "squishes" any value towards the top and bottom margins. If the output of the sigmoid function is less than 0.5, the algorithm will predict that the data point belongs to first class, otherwise that it belongs to second class. The equation of the sigmoid function is the following one:

While in linear regression algorithms, adjusting the linear function is done by minimizing the sum of the distances between each data point to the line (i.e. the least squares method), in logistic regression the maximum likelihood estimation is used instead. The maximum likelihood estimation method attempts to maximize the conditional probability of observing the data X given a specific probability distribution and its parameters theta [8].

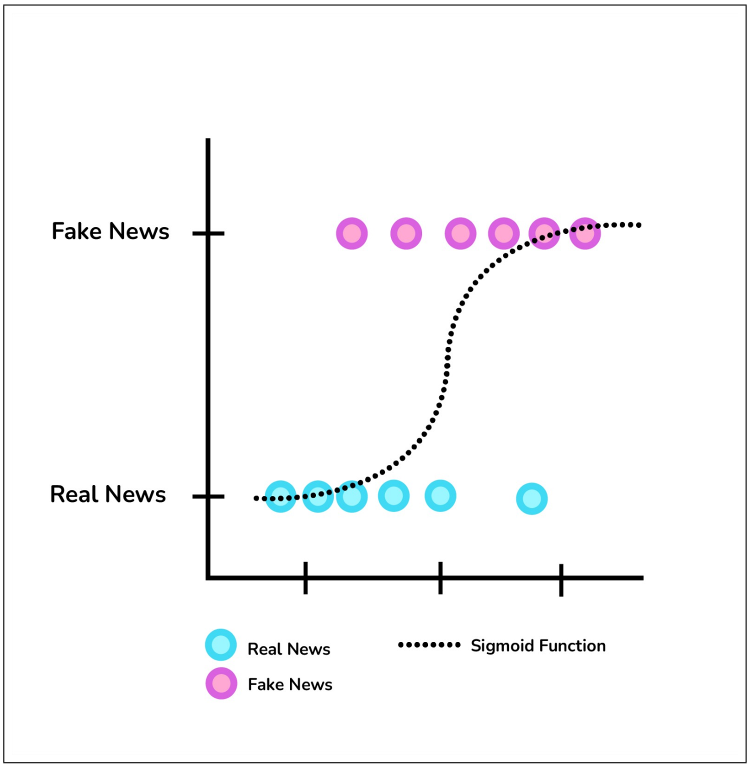


Figure 4: Logistic regression sigmoid function.

The advantages of this algorithm include: efficient training process, easy and intuitive implementation, ability to easily extend to multiple classes. Conversely, there is also one major disadvantage: the constraint of linear boundaries.

2.2.5 Support Vector Machine

Support Vector Machine is another popular supervised learning algorithms, that is often preferred for the high accuracy along with low computation power. The main idea of the algorithm is to search for a hyperplane which separates the data points into the possible output classes.

From all the possible hyperplanes meeting this requirement, the algorithm has to specifically select that which is furthest from the data points of each category. Maximizing the margin between the hyperplane and the data points results in a more solid decision boundary between the classes, making the predictions more confident and reliable. If the hyperplane is discrepantly closer to the data points of a certain class and further from another class, then the observation located in the close proximity of the hyperplane could be classified in the group which is considerably further from.

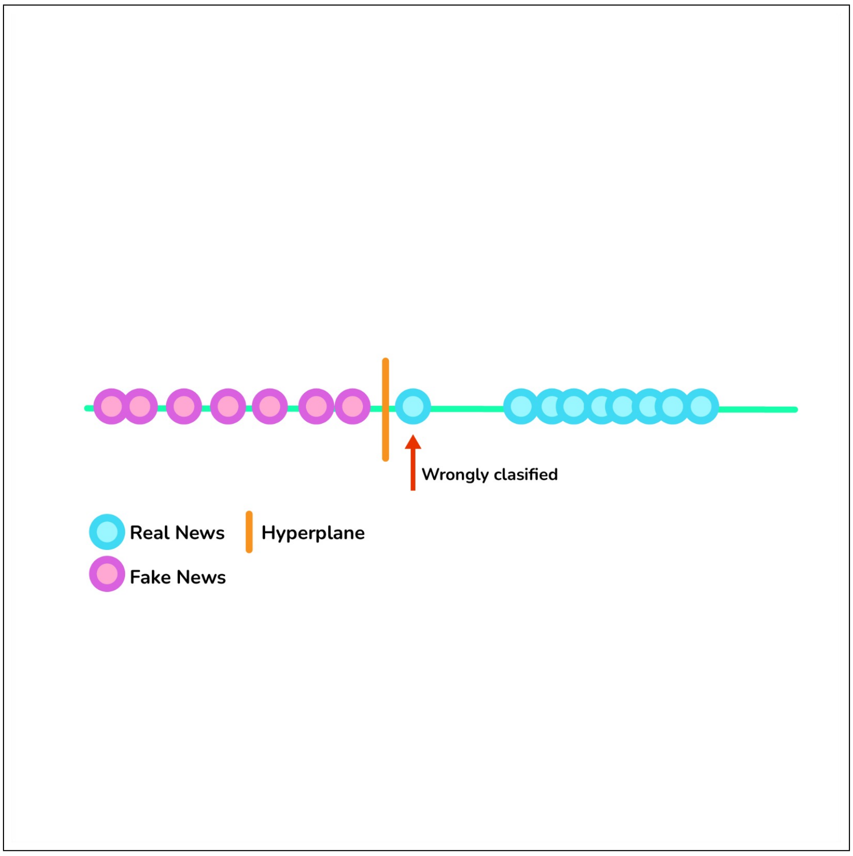


Figure 5: Hyperplane too close to one of the classes.

The name of "Support Vector Machine" reveals the essential elements that the algorithms is based upon: support vectors (i.e. data points that are closest to the hyperplane and which influence the position and orientation of the hyperplane).

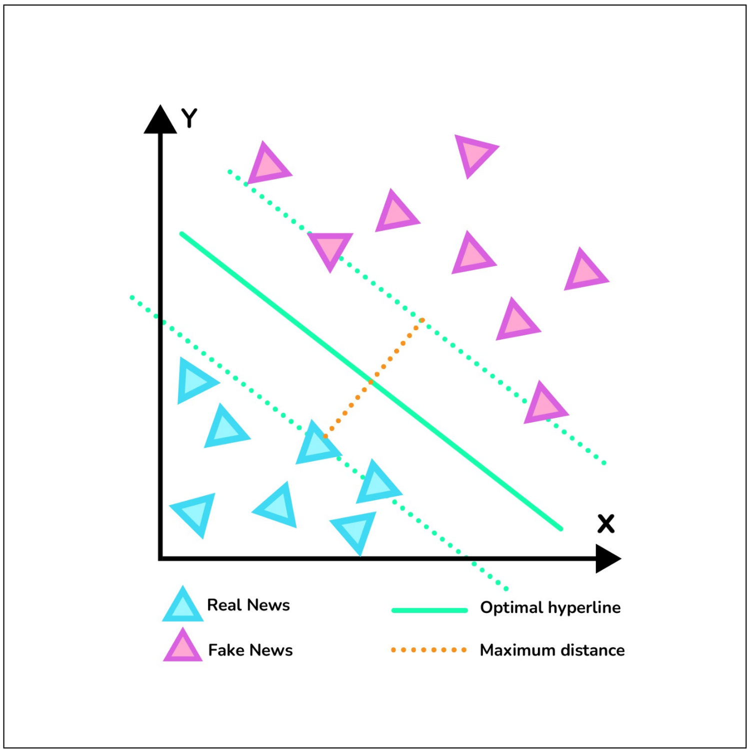


Figure 6: Support Vector Machine optimal hyperplane.

3 Fake News Detection Methods Overview

The last decades marked important advancements in the field of fake news detection. These advancements coincide non only with the exponential rise of fake news owed to the communication and information breakthroughs brought about by the Internet and social media, but also with the remarkable progress recorded in computer science along with artificial intelligence and its subfield of machine learning.

In the relatively novel literature of fake news detection, the solid prospects of machine learning algorithms are very frequently discussed. There are several manners in which machine learning can be leveraged for the identification of fake news, but most of them revolve around two main categories: news content-based and social context-based fake news detection [2]. In what follows in this chapter, these two different types of methods will be further detailed.

On the one hand, news content-based fake news detection incorporates the machine learning approaches which predict the factuality and deceitfulness of news by analyzing the metadata of news article, consisting of the source, headline, body text and images/videos.

B. Horne and S. Adali et. al [9] point out that there is often an underlying presumption that the editorial style in which fake news are written are made to be as similar as possible to real news. Nonetheless, this assumption is most of the times inaccurate and as a matter of fact, the body content of the news can be highly evocative in terms of the reliability of the facts presented in the news. An important element making fake news stand out is the persuasion element, which is most of the times achieved through heuristics rather than strong arguments.

In addition, B. Horne and S. Adali, as well as Y. Chen and J. Conroy et. al [10] highlight that both the body content as well as the headline are very often characterized by clickbait, which refers to the opinionated and inflammatory language used in order to entice reader to read the full article. Therefore, before even reading the content of the article, the lingustic features of tge headline itself can be exploited straight away to predict the misleading intent of the article, often crafted for political or financial gain.

Moreover, extracting other linguistic feature from the body of the content, such as number of quotes, the average length of the quotes, number of external links, number of sources cited, number and length of the graphs can also predict how well fact-checked and credible are the information presented [11].

Besides the headline and the content of the news articles, the visual cues have been proven to be a critical element of propagandistic pieces of news. Images can inspire emotions and legitimacy to the stories, in spite of not bringing little to no arguments or evidence [12].

On the other hand, in addition to the content-based algorithms utilizing the metadata of the news articles, there is also some research considering the detection based upon the social context to be even more effective.

There is some literature exploring user-based approaches, working on the observations that most fake news are spread by non-human social media accounts or also called, bots. Individual-level features are exploited to deduce the overall credibility of the user, by inspecting various demographics of the suspicious accounts, such as registration age, number of followers/followees, number of tweets the user has posted / authored and so forth [13]. Apart from individual-level features, some group-level approaches can also capture some important characteristics of the environment in which fake news are spread. Group-level properties are based upon the observation that both fake and real news spreaders have the tendency of creating and adhering to social media communities with the same purpose. Consequently, fake news can be detected by following some characteristics of these groups, such as percentage of verified users, average number of followers or various other statistics that are build by computing the average of individual-level feature of the users comprising the groups.

4 Application Development

This chapter provides a complete rundown on the development process of the fake news detection application build for this thesis, from inception to completion.

4.1 Functionalities

The application developed for this thesis is a web extension built for Google Chrome, which provides users with an accessible tool for evaluating the reliability of online news. The extension performs a multifaceted analysis of any given news article, based upon its URL, headline, content and authors, by means of machine learning algorithms, web scraping and crowdsourcing.

The key functionalities of the application are the following:

1. upon landing on an online news article from a known news source, automatically extract some critical information about the article, based on its HTML source code (i.e. URL, headline, content and authors)

2. after successfully extracting the required information about the article, automatically initiate a multifaceted reliability analysis, which rates the current article from the following standpoints:

   (a) the biased/deceptive character of the language used in the content of the article, predicted by one of the machine learning algorithms implemented for content-based fake news detection

   (b) the deceptive/sensationalist character of the language used in the headline of the article, predicted by one of the machine learning algorithm implemented for title-based clickbait detection

   (c) the reliability of the source based upon the list of reliable / perennial sources web-scraped from Wikipedia [14]

   (d) the reliability of the source based upon previous user feedback

   (e) the credibility of the authors based upon previous user feedback

3. after the completion of the analysis, the users have the ability to provide their own rating / feedback for the article

4.2 Architecture and Technologies

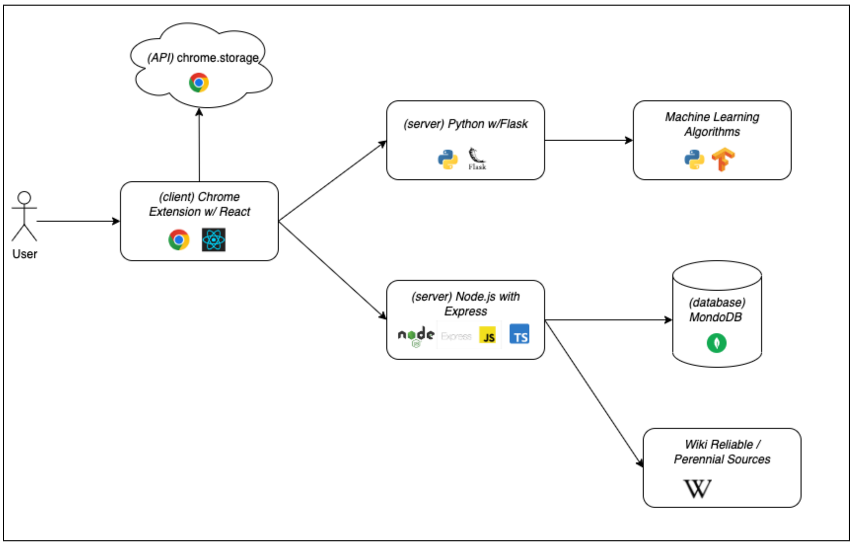

(API) chrome.storage
User
(client) Chrome Extension w/ React
(server) Python w/Flask
Flask
Machine Learning Algorithms
(server) Node.js with Express
node
Express
JS
TS
(database) MondoDB
Wiki Reliable / Perennial Sources
W

Figure 7: Diagram depicting the architecture of the application.

4.2.1 Front-end

The **client-side** (**front-end**) of the application was built as a Chrome Extension, so that all the fake news detection and reliability analysis functionalities are immediately accessible upon opening a given news article on the browser. Compared to a standard web front-end application, a Chrome extension (and browser extensions in general) have a more peculiar architecture.

Chrome extensions have three main components:

1. **Service Worker**, a JavaScript script running in the background of the browser, reacting to events emitted from the browser, such as tabs/windows being open/closed, refreshing the webpage, modifying the URL from a tab etc. Compared to popup and content scripts, the lifecycle of the service worker is not dependent of any webpage. Therefore, the service worker can be running across webpages, terminate only after becoming idle and restart only when needed (i.e. when having to handle an event from the browser). Service workers are also useful if one wants to store a temporary local state

shared across tabs or run an action in the background which should not run only while the popup of the extension is open.

2. **Popup**, the user interface of the chrome extension, which gets toggled when clicking on the icon of the extension located on the taskbar. By means of HTML, CSS and JavaScript, an interface comparable to an ordinary webpage can be written, with certain implicit limitations, such as lack of routing, cannot use standard Redux for state management etc. The lifecycle of the popup lasts only while the popup is open, so if one wants to preserve state between popup sessions, one should either choose to store state in the background script, localStorage or Chrome's Sync or Local Storage.

3. **Content Script**, JavaScript scripts running in the context of webpages. They are useful when it comes to reading or modifying the Document Object Model (DOM) of the webpage. Even though they can perform changes to the HTML and CSS source code of the webpage, they are unable to interact with the JavaScript of the webpage (i.e. access and use functions or variables defined in the context of the web page or extension). Also, they cannot access the Chrome APIs and events. Despite having these limitations, which many have as workaround to communicate via a messages protocol with parent extension, content scripts are useful in many scenarios.

The popup of the news reliability extension was written using the JavaScript library called React, which provides a more efficient development environment and functionalities for writing component-based web interfaces. With the help of components, different logical parts of the view can be separated by concerns.

As for the programming language, I used TypeScript instead of simple JavaScript. TypeScript is a strongly-typed superset of JavaScript, which enables writing code less prone to errors, thanks to the introduction of static typing.

As for the programming paradigm, the code written can be mostly categorized as functional and procedural. Classes were used at minimum, and all the user interface components, service functions, utility functions, React hooks were writing as functions, attempting to respect some of basic functional programming principles: immutable state, no side effects (as much as possible), functions as first-class entities, some use of pure functions and type systems.

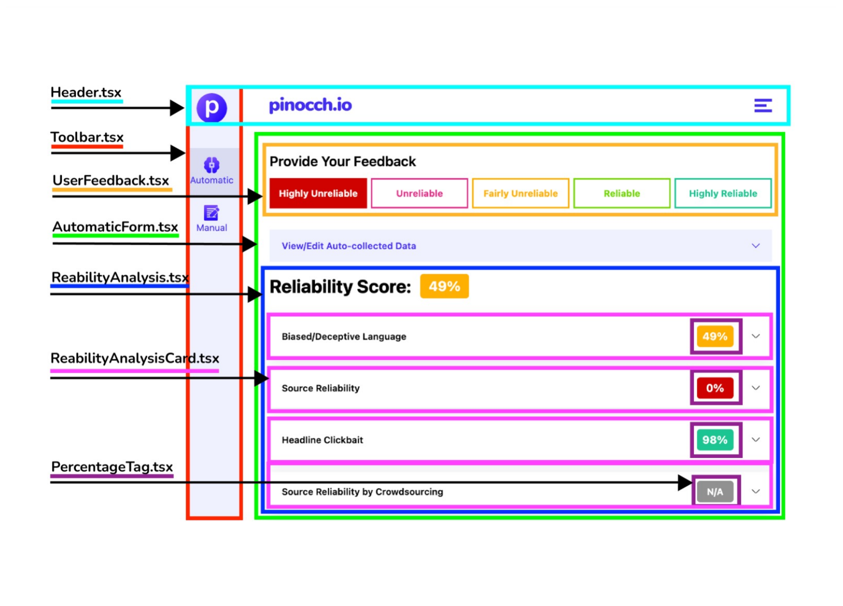


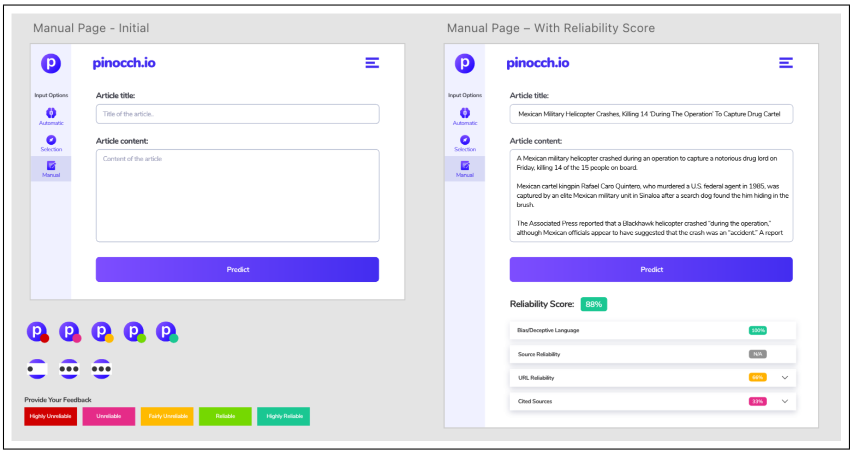
Figure 8: Component-based structure of the user interface.

For styling, instead of plain CSS, TailwindCSS was used instead. TailwindCSS (i.e. similar to Bootstrap, but more customizable), is essentially a large collection of customizable CSS classes, which eliminates the need of having separate CSS files and all the styling can be done by adding the needed classes to the HTML elements.

For HTTP requests to the RESTful APIs build on the back-end, the Axios library was chosen. Axios facilitates the process of creating asynchronous requests.

4.2.2 UI Design

Prior to coding the user interface, I followed the good practice of creating design mockups first. Design mockups are of great utility to pre-plan and visualize how the UI elements should be organized in order to not only have a clean and attractive appearance, but to also project and create the best workflow for the application. For this purpose, I made use of Adobe's UI/UX design software called Adobe XD, which was also useful for creating some graphical elements, such as logos, icons and buttons designs.



Article title: Title of the article..

Article content: Content of the article

Article title: Mexican Military Helicopter Crashes, Killing 14 'During The Operation' To Capture Drug Cartel

Article content:

A Mexican military helicopter crashed during an operation to capture a notorious drug lord on Friday, killing 14 of the 15 people on board.

Mexican cartel kingpin Rafael Caro Quintero, who murdered a U.S. federal agent in 1985, was captured by an elite Mexican military unit in Sinaloa after a search dog found the him hiding in the brush.

The Associated Press reported that a Blackhawk helicopter crashed "during the operation," although Mexican officials appear to have suggested that the crash was an "accident." A report

Reliability Score: 88%

Bias/Deceptive Language	100%
Source Reliability	N/A
URL Reliability	66%
Cited Sources	33%

Figure 9: Design mockup of some of the pages and graphical elements.

4.2.3 Back-end

On the server-side (back-end), two servers (RESTful APIs) were built. On the one hand, a server was required to train and expose the machine learning models built for article content-based and headline-based fake news detection. On the other hand, another server was needed to manage some persistent state of the application, such as the set of news sources which are automatically recognized by the web extension or the history of user feedbacks/ratings submitted.

First of all, the server for the machine learning algorithms was built using Python. Python is probably the most popular choice for machine learning, because of its simplicity, flexibility and solid libraries and frameworks.

Scikit-learn is one of the essential libraries used withing this project. It is one of the most popular machine learning algorithms, which provides access to a great variety of machine learning classification and regression algorithms, including all those used in this application: k-nearest neighbor, decision tree, logistic regression and support vector machine. Furthermore, Scikit-learn also gives access to various Natural Language Processing related tools, such as the TF and TF-IDF feature extraction methods which where applied during the training process of the models.

In addition to Skicit-learn, another natural language processing library which came in handy during the development of the classifiers was NLTK (Natural Language Toolkit). This library was primarily used in the pre-processing phase of the each training sessions. NLTK provided tool for each required pre-processing operation, namely tokenizations, lemmatization, stemming, removal of punctuation and removal of stop words.

In machine learning, advanced data manipulation is a fundamental task necessary for by and large every algorithm. In this project, Pandas was leveraged for this purpose. Pandas was particularly useful in loading the datasets stored into CSV file and apply all the data pre-processing and data leakages cleaning.

Partially, the machine learning models that were trained are exposed to the Chrome extension via a RESTful API built using Flask. Flask is a framework for creating python APIs and it often regarded as a microframework, because it does not require any libraries or tools.

A secondary purpose the python server fulfills, besides the machine learning tasks, is an API which receiving as query parameter the URL to a webpage (specifically, to a news article), responds with some attributes of the website later used for the reliability analysis (i.e. content, headline and authors). In order to do this, the Newspaper python tool was used, which via some complex algorithms, several of them involving AI, can extract with a decent performance the aforementioned attributes. Initially, I attempted to extract these attributes manually, with no additional library, by merging all the p HTML tags to retrieve the content, take the heading HTML attribute for the title and look after elements with a class name containing the words "author" or "contributors" to get the authors. Nonetheless, achieving this task proven more difficult than originally anticipated, due to the manifold variables that appear in the HTML structure of each different website.

As for the programming paradigm, opposite to the procedural and functional style adopted at the front-end level, the Python application is based on the Object-Oriented Programming principles, such as abstraction, encapsulation or polymorphism. The classes for the first of the machine learning models were designed and written so that they could be re-used and adapted to other machine learning tools with a minimal number of modifications. This facilitated an accessible iteration through various algorithms, feature extraction methods and datasets, both for content-based and headline-based fake news detection methods.

Second of all, the API built for managing the server state of the application, such as the news sources scraped, previous reliability analysis and user ratings, was developed using Node.js, with the Express framework. Node.js and Express were chosen for the outstanding flexibility and freedom in architectural preferences, the multitudinous libraries and NPM (Node Package Manager) packages. Moreover, using the same language both on the front-end and the back-end (i.e. TypeScript), comes as a benefit when having to couple them together.

Besides managing the data of the application via CRUD operations, the Node.js server was also responsible with the web-scraping part (i.e. scraping the list of articles classified by their reliability from here [14]). For this purpose, the Cheerio NPM package was used, which provides a tool for retrieving and parsing the DOM of any webpage given its URL.

4.2.4 Database

The server is connected to a NoSQL database, created by means of MongoDB, which is JSON-like document oriented cross-platform database management program. If in case of SQL entities are represented as tables, in MongoDB each entity is a document. MongoDB is a popular choice for modern applications thanks to its high performance levels (storing most of the data into the RAM), speed and availability (performs tens of times faster than relational databases), flexibility (does not have a pre-defined schema, offering a dynamic schematic architecture that works with non-structured data) and scalability.

MongoDB also has some disadvantages, but they bore out not to be prominent during the development of the application: no support for JOIN statements (given its non-relational structure), a limited size of 16MB per document and no support for transactions.

To ensure the portability of the application, I decided upon deploying the database to the cloud. I took advantage of the free option provided by MongoDB Atlas. By deploying the server to MongoDB Atlas: a reliable security is ensured for the data, the management of the database is most efficient, there is no need to set up the database locally and the database is practically already prepared for a future production version of the application.

4.2.5 Testing

During the development of the application, both automated and manual testing methods were applied. Automated tests were written predominantly while working on the machine learning algorithms for fake news detection, whereas the front-end of the Chrome extension and the Node.js server managing the state of the application and the web-scraping where mostly tested manually.

Firstly, the reason for writing automated tested for the machine learning algorithms was to ensure that any changes performed to their logic would not lead to underlying errors that would have negatively affect or result in misleading measures of the actual performance of the models. Practically, this was the most sensitive and error-prone part of the project, so automated test were considered a useful addition.

Using the unittest Python library for writing test cases, the dataset cleaning, pre-processing and processing methods were tested. Feeding the correct data optimally processed to the model for training is essential and given that changes were often performed to this part of the code to adapt for each task, test prove to be indeed beneficial in detecting errors. Both the test-last and test-first techniques were employed. As highlighted in various studies like [15].

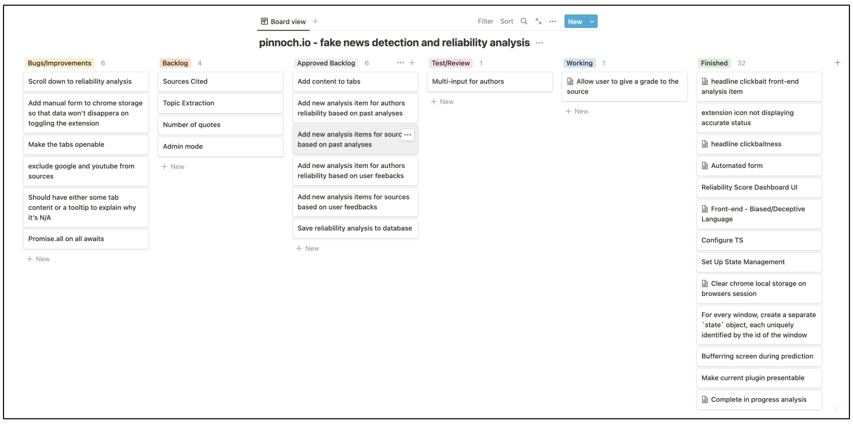
To test the performance of each of the machine learning model, various standard metrics were applied (i.e. accuracy, recall, precision, F1 score). These techniques are further detailed in the following sections of the thesis.

For the manual testing of the APIs, I used an API client application called Insomnia, which enabled verifying and validating every API method. Lastly, the front-end was tested by iterating over various scenarios and edge cases through the user interface.

4.2.6 Management

For source control management, git was used together with Github. The master branch was used for stable production-ready code, while the development branch was periodically merged into master and used as the base branch in which feature branches were merged. Each feature branch was created for different functionalities and logical parts of the application, to avoid modifying directly stable parts of the code. In case there were some severe bugs to be solved immediately, hotfix branches would be used.

For features and bugs management, given that this was an individual project, I decided on employing a simple tool, without too many complex unneeded functionalities. I used Notion, which enables the creation of Kanban boards, frequently used at industrial-level as well. A Kanban board is split into multiple columns, each of them representing different stages of the development of a feature. Feature ideas would be placed into the "Backlog" column, features that are planned to be implemented in the "Approved Backlog" column, features that are currently in progress into the "Working" column, feature which need to be tested and reviewed further in the "Test/Review" column, features that are completed in the "Done" column and various bugs and improvement ideas were placed into the Bugs/Improvements column. In many of the tickets from the board, I added a description and a checklist of tasks that need to be fulfilled to consider the feature implemented and ready to be moved to the "Test/Review" or the "Finished" column.



pinnoch.io - fake news detection and reliability analysis

Bugs/Improvements 6
- Scroll down to reliability analysis
- Add manual form to chrome storage so that data won't disappera on toggling the extension
- Make the tabs openable
- exclude google and youtube from sources
- Should have either some tab content or a tooltip to explain why it's N/A
- Promise.all on all awaits

Backlog 4
- Sources Cited
- Topic Extraction
- Number of quotes
- Admin mode

Approved Backlog 6
- Add content to tabs
- Add new analysis item for authors reliability based on past analyses
- Add new analysis items for sourc based on past analyses
- Add new analysis item for authors reliability based on user feebacks
- Add new analysis items for sources based on user feedbacks
- Save reliablility analysis to database

Test/Review 1
- Multi-input for authors

Working 1
- Allow user to give a grade to the source

Finished 32
- headline clickbait front-end analysis item
- extension icon not displaying accurate status
- headline clickbaitness
- Automated form
- Reliability Score Dashboard UI
- Front-end - Biased/Deceptive Language
- Configure TS
- Set Up State Management
- Clear chrome local storage on browsers session
- For every window, create a separate `state` object, each uniquely identified by the id of the window
- Bufferring screen during prediction
- Make current plugin presentable
- Complete in progress analysis

Figure 10: Notion Kanban board for features and bugs management.

4.3 Machine Learning Algorithms

In this section, an outline of the content-based and headline-based fake news detection algorithms is presented, talking about the dataset selection, data processing, feature extraction, training and results.

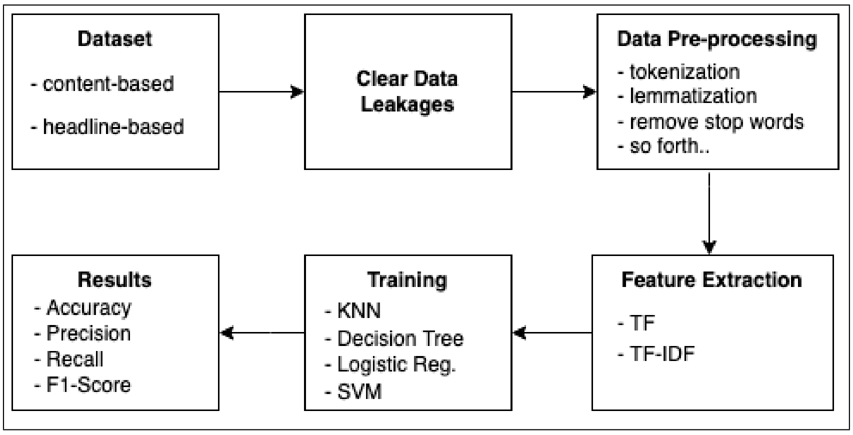


Figure 11: The process of building a classifier.

4.3.1 Datasets

For the content-based fake news classification, a dataset [16] with over 35.000 news articles was chosen, with roughly half of them classified as fake and half of them real. On the one hand, the real news were collected from Reuters articles mostly [17], which along with the Associated-Press and Agence France-Presse is one of largest and most generally reliable news agencies in the world. Reuters is known for the objective manner in which the articles are written, hence it can be regarded as a good source to form a dataset of real news. On the other hand, the fake news data is a selection of news classified by PolitiFact and Facebook as fake. PolitiFact is a fact-checking nonprofit project, where journalists, editors and reporters rate articles and claim from politicians.

However, the dataset is not perfect in its initial form, due to the data leakages it contains [18]. In the context of machine learning, data or information leakages refer to additional or abnormally distributed information in the training dataset, which can cause misleading model predictions in production. This kind of information is too peculiar to the dataset and does not generally behave the same way for real-life inputs, leading to overfitting. The result is that the machine learning models yield misleadingly high accuracy scores. In order to resolve this issue, the detected data leakages had to be cleared, by: removing the "Reuters" substring which appeared as a prefix for most of the real news (texts that contained this prefix could be predicted with 100% accuracy just based on that prefix), removing the subject and the date columns (because they were unequally distributed between fake and real news, they would mislead the models) and removing duplicate news (if they end up both in the training and the testing tests, the ground truth would be leaked to the training set).

For the headline-based fake news classification, the selected dataset [19] contains over 32.000 entries, namely headlines that were classified as either clickbait or non-clickbait. Clickbait stand for bombastic and often deceptive headlines or texts, whose score is to convince users to access their website. These types of headlines are generally unacceptable for objective and trustworthy news articles. The data was collected from a series of news sites. The clickbait headlines were collected from 'BuzzFeed', 'Upworthy', 'ViralNova', 'Thatscoop', 'Scoopwhoop' and 'ViralStories', while the non-clickbait headlines are from 'WikiNews', 'New York Times', 'The Guardian', and 'The Hindu'.

4.3.2 Data pre-processing

Data pre-processing marks a key step in building a natural language processing algorithm, in which certain parts of the dataset irrelevant or deceptive are either modified or dropped, in order to enhance the results of the model.

The content-based dataset as well as the headline-based dataset were subjected to the same pre-processing techniques, extensively applied in many other text processing algorithms.

1. Converting letters to lower case, as the case of the texts are not considered to convey an important message in case of my algorithms.

2. Removing hyperlinks and HTML elements, which most of the time represent parts that were not meant to be scraped (a flaw of the scraping algorithm) and which have no relevance in whether an article is fake or real.

3. Removing stop words, since they carry little or no meaning whatsoever. They are the most commonly used words, indispensable in fake news in the same manner as in real news.

4. Dropping punctuation marks, for the same reason as for which stop words are removed.

5. Word tokenization, which splits texts into smaller units, namely words, from which features are extracted later.

6. Lemmatization, a superior version of stemming. Contrarily to stemming, lemmatization takes into account the word's part of speech, producing accurate root forms that are real dictionary words every time. For instance, lemmatization would convert the word "caring" to "care", while stemming would erroneously plainly drop the "ing" suffix and obtain the word "car".

4.3.3 Feature Extraction

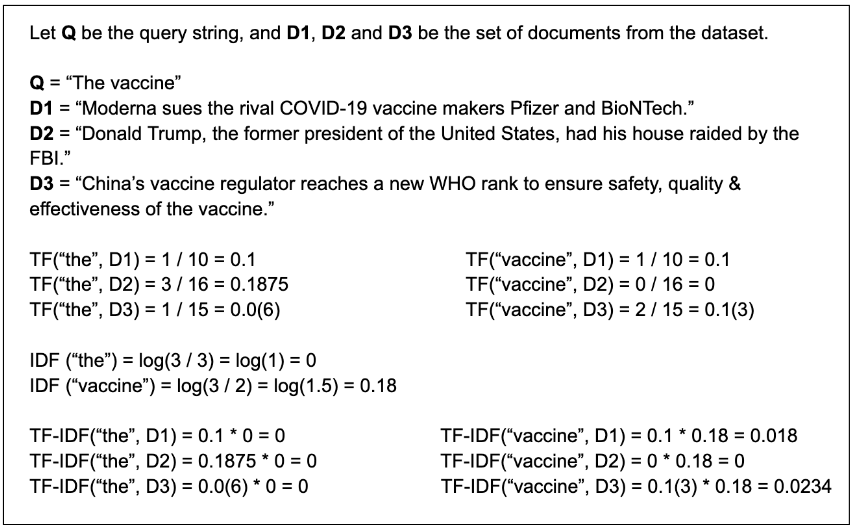
Once the dataset was cleaned from data leakages and was subjected to some preprocessing operations, it reaches the final stage before being fed for training to the machine learning model: feature extraction. This step involves applying a function or a series of function on the dataset in order to extract some characteristics of each data entry that the model will learn during the training process and generalize into a function. This resulting function will be later used by the model to make predictions on unseen data.

There are a variety of feature extraction methods, but the one used within this thesis for the training of the classifiers is TF-IDF (i.e. term frequency–inverse document frequency). Before understanding TF-IDF, one should beforehand understand another feature extraction method that the formed is founded upon, called TF (term frequency).

As the name suggests, TF (term frequency) refers to the frequency of a word in a document, or in other words, the frequency of a word in one entry from the dataset. Suppose we have the following dataset entry "Donald Trump, the former president of the United States, had his house raided by the FBI.". In this case, the term frequency of the word "the", is equal to ratio between the number of occurrences of the word "the" in the document and the total number of words in the document. Thus, TF("the") = 3 / 16 = 0.1875. The larger the output of the TF function is, the more relevant and common is the world in the context of that document. As a result, the words that are more relevant and are more frequently met in certain output classes than in others, will help the model to make a prediction in the favour of one of those classes in which the word is more frequent.

IDF represents the inverse document frequency of the word across the entire set of documents. IDF involves computing the logarithm of ration between the total number of documents and the number of documents containing a certain word. Suppose we have a total number of 15000 documents and 1750 of the documents contain the word "vaccine". Hence, IDF("vaccine") = log(15000 / 1750) = log(8.57) = 0.93. Contrarily to TF, rare words will have high scores (approaching 1), whereas common words will have low scores (approaching 0).

Now that both TF and IDF were separately discussed previously, we can finally compute the TF-IDF function. The original purpose of TF-IDF was related to document search and information retrieval. Suppose the query string is "The vaccine". The TF-IDF function will yield a greater score for each document, proportionally to the frequency of each word from the query string found in the documents, assigning higher scores to more rare words like "vaccine" compared to more common words like "the". TF-IDF for a specified word is the product between the TF and the IDF of that word. Below, an example of calculation for TF-IDF is provided.



Let **Q** be the query string, and **D1**, **D2** and **D3** be the set of documents from the dataset.

Q = “The vaccine”
D1 = “Moderna sues the rival COVID-19 vaccine makers Pfizer and BioNTech.”
D2 = “Donald Trump, the former president of the United States, had his house raided by the FBI.”
D3 = “China’s vaccine regulator reaches a new WHO rank to ensure safety, quality & effectiveness of the vaccine.”

TF(“the”, D1) = 1 / 10 = 0.1
TF(“the”, D2) = 3 / 16 = 0.1875
TF(“the”, D3) = 1 / 15 = 0.0(6)

TF(“vaccine”, D1) = 1 / 10 = 0.1
TF(“vaccine”, D2) = 0 / 16 = 0
TF(“vaccine”, D3) = 2 / 15 = 0.1(3)

IDF (“the”) = log(3 / 3) = log(1) = 0
IDF (“vaccine”) = log(3 / 2) = log(1.5) = 0.18

TF-IDF(“the”, D1) = 0.1 * 0 = 0
TF-IDF(“the”, D2) = 0.1875 * 0 = 0
TF-IDF(“the”, D3) = 0.0(6) * 0 = 0

TF-IDF(“vaccine”, D1) = 0.1 * 0.18 = 0.018
TF-IDF(“vaccine”, D2) = 0 * 0.18 = 0
TF-IDF(“vaccine”, D3) = 0.1(3) * 0.18 = 0.0234

Figure 12: Example of calculation of TF-IDF.

4.3.4 Training and Results

Before starting the training and subsequently the testing process, the dataset should be apportioned into two according subsets. When performing this split, one needs to ensure on the one hand that the training set is large enough to produce meaningful and significant statistical measures. On the other hand, it is essential that the underlying structure of both the training and the testing dataset are both respectively representative of the entire dataset.

In order to ensure a good training-test dataset split, the method used resembles the 5-fold cross-validation technique was used and was achieved through a sklearn library, which follows the following general procedure:

1. Shuffle the dataset randomly.
2. Divide the shuffled dataset into 5 subgroups.
3. Retrieve one group for testing.
4. Retrieve the rest of the groups for training.

In this way, we got to employ 20% of the dataset for testing and 80% for training, making sure that there underlying form is comparable and they both embody the overall structure of the whole dataset.

To quantify the performance of the models, the confusion matrix should be build. A Confusion matrix is an N dimensional square matrix used for evaluating the performance of a classification model, where N represents the number of output classes. The matrix performs a comparison of the predicted values and the actual true values. This provides a holistic view of how well a machine learning model performs and what kind of errors were made.

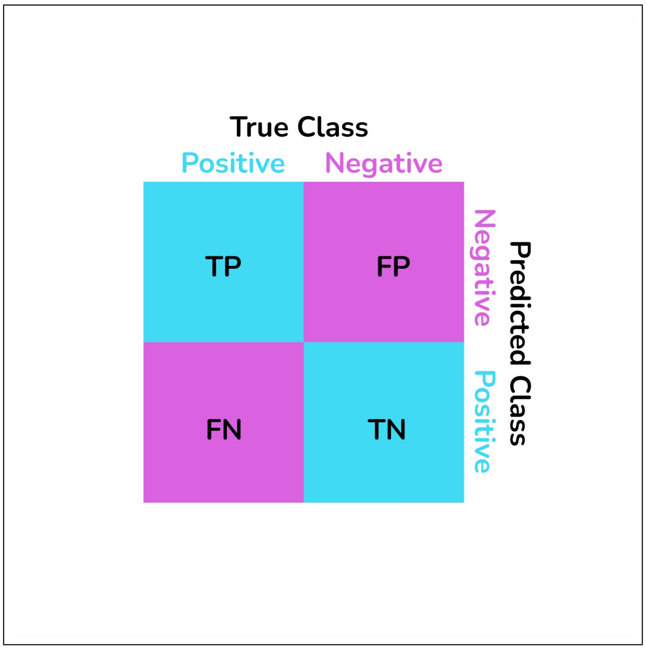


Figure 13: The elements of a confusion matrix.

In the case of fake news detection, where the classifier answers with yes or no at whether an article is fake or not, the positives refer to the fake news and negative to real news.

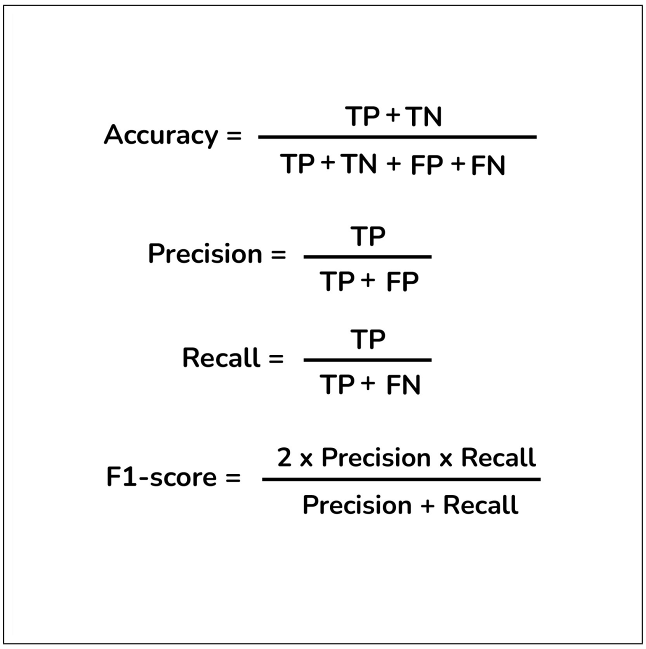
Four different standard machine learning models were used, all based on the parameters generated in the confusion matrix.

1. **Accuracy**: expresses the overall percentage of correctly predicted training examples (i.e. articles that were accurately predicted as fake news and articles that were correctly classified as real news).

2. **Precision**: describes the ratio between the correctly classified positive observations (i.e. correctly classified fake news) and the total number of correctly classified positive and negative observations. Practically, it reveals how well the model performs on accurately detecting fake news.

3. **Recall**: going hand-in-hand with the precision score, the recall denotes out of all the fake news, how many were actually correctly labeled as fake

4. **F1-Score**: leveraging the previously computed recall and precision metrics, computes the harmonic mean of them. The F1 score is less intuitive than accuracy, but many times is of greater utility in determining the performance of the classifier.



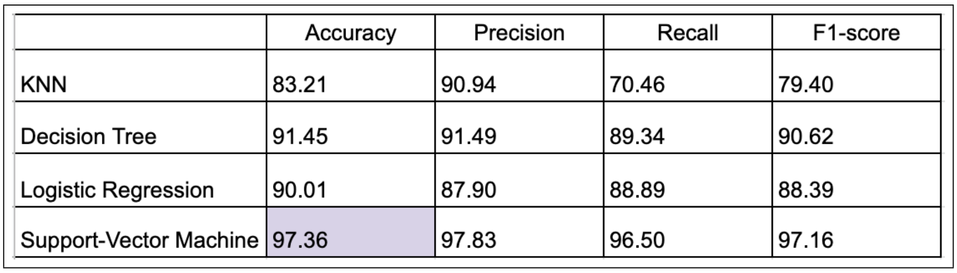
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

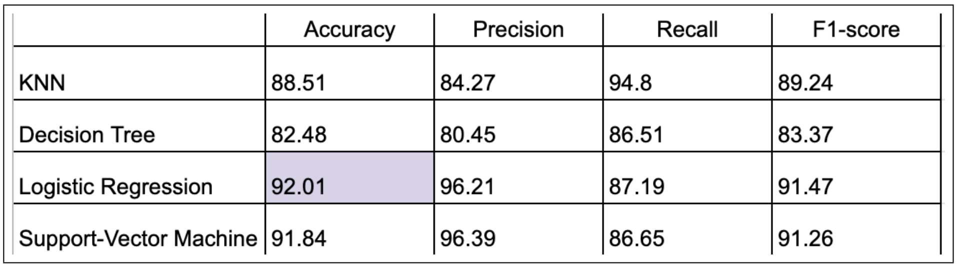
$$\text{F1-score} = \frac{2 \text{ x Precision x Recall}}{\text{Precision + Recall}}$$

Figure 14: Formulas for accuracy, precision, recall and F1-score



	Accuracy	Precision	Recall	F1-score
KNN	83.21	90.94	70.46	79.40
Decision Tree	91.45	91.49	89.34	90.62
Logistic Regression	90.01	87.90	88.89	88.39
Support-Vector Machine	97.36	97.83	96.50	97.16

Figure 15: Content-based classification metrics.



	Accuracy	Precision	Recall	F1-score
KNN	88.51	84.27	94.8	89.24
Decision Tree	82.48	80.45	86.51	83.37
Logistic Regression	92.01	96.21	87.19	91.47
Support-Vector Machine	91.84	96.39	86.65	91.26

Figure 16: Headline-based classification metrics.

4.4 Source Reliability via Web-scraping

Through a rather ample undertaking, the Wikipedia community compiled a sizeable list of news sources and classified them by their overall reliability [14]. The put forward a "request for comment" on their noticeboard. A request for comment (abbreviated as RfC) is a process for requesting outside input concerning disputes, policies, guidelines or article content. Via this procedure, Wikipedia gathered evidence, reports, questions and in-depth conversations from the community and compiled the general consensus into a table containing hundreds of news sources rated by their reliability.

As for the legend of the rating system, the following ratings or statuses were attributed to the news articles:

1. "Generally reliable", a rating for news sources about which the editors have a general consensus of being reliable in their respective area of expertise. These kind of sources have a solid reputation for fact-checking, accuracy and error correction. Sources are considered to belong to this category as long as they fulfill certain criteria: their articles are not rejected by more authoritative news sources, they do not make exceptional, conspiratorial or sensationalist claims and so forth.

2. "No consensus", describing news sources which editors were unable to reach a unanimous agreement about and which did not stand out negatively by any dubious or false claims. These sources are regarded as moderately reliable and the information presented by them should be ready with cautiosness.

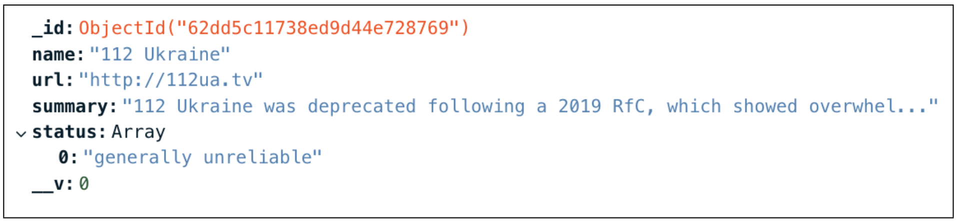
3. "Generally unreliable", label news sources whose news are by and large considered at least questionable. These sources might lack a serious editorial team, might perform poor on fact-checking, rely on user-generated content and so on.

4. "Deprecated" and "Blacklisted" news sources that are deprecated or not available for public use anymore or blacklisted due to persisting abuse. Either way, they are recommended not to be used.

There was a proposed project called Sourceror [20], which planned to build an API to enable developers to use this data in a more accessible manner, without the need to parse this data themselves. Nevertheless, as far as I researched, no further progress was done in this direction. Therefore, in order to provide this data into my application for users to effortlessly access the reliability of the source, the API had to be build by myself.

The web-scraping is performed at the level of the Node.js server, by means of the cheerio node package. At a high-level, what the algorithms does is to parse the table from the HTML code of the reliable sources article from Wikipedia [14] and extract for each news source the name, URL, status and summary, the latter explaining briefly the reasoning behind the reliability status assigned to each source. The part that proved slightly more difficult was to extract the ULR of each webpage, because this information was not included in the table. The only URLs included in the table were the URLs from the Wikipedia page of each source. Finally, in order to obtian the URL for each news source the application accessed each wikipedia page of each news source and using some HTML class filters was able to successfully extract the URL from most news articles.

All the scrape functionality is exposed via an API endpoint and when making a request to that endpoint, the news are scraped and saved into the MondoDB. database of the application.



```
_id: ObjectId("62dd5c11738ed9d44e728769")
name: "112 Ukraine"
url: "http://112ua.tv"
summary: "112 Ukraine was deprecated following a 2019 RfC, which showed overwhel..."
⌄status: Array
   0: "generally unreliable"
__v: 0
```

Figure 17: A document from the database storing one of the news sources scraped.

4.5 Crowdsourcing / User Feedback

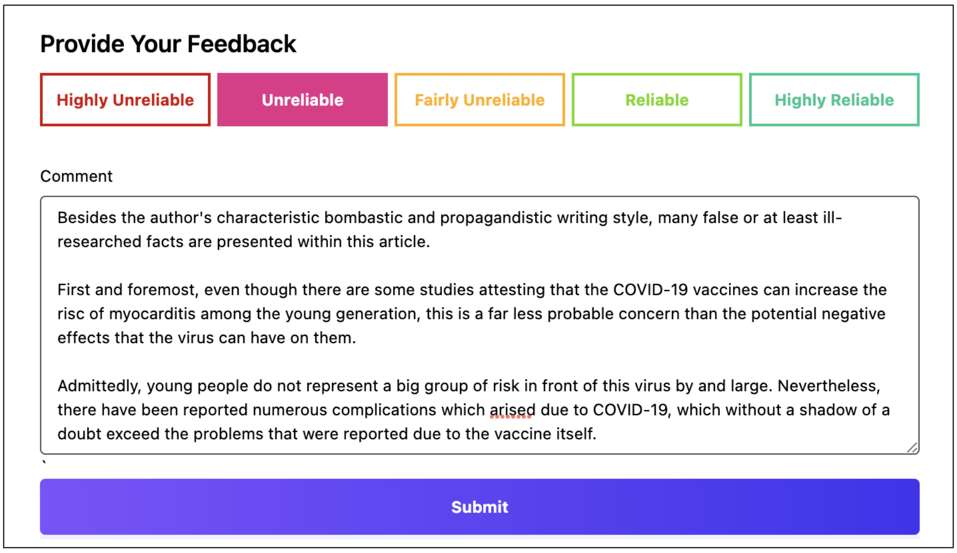
Crowdsourcing represents the practice of collecting users input in order to solve a certain task. In the fake news detection browser extension build for this thesis, the user input is collected through feedback forms, in which users can provide a personal review with regards to their opinion on the reliability of the article they have just used the application for. The extension provides a review system which enables the user to select from 5 different possible ratings: "Highly Unreliable", "Unreliable", "Fairly Unreliable", "Reliable" and "Highly Reliable." Optionally, the users can attach a short explanatory comment to justify their rating.

Besides evaluating websites from the perspective of machine learning algorithms, it can be also highly useful to know and rate the reliability of the news articles, news sources and authors from the perspective of the users using the application. Collecting input from users is beneficial for cultivating a community, which reunites around the common aim of combating fake news, having conversations and promoting reliable news websites.

The information obtained through crowdsourcing was leveraged to provide two additional reliability analysis standpoints in the application: the reliability of the article from the perspective of the news source and the authors. When performing a reliability analysis for any given news article, in the background this action triggers a query to the database which selects all the past user feedbacks that were submitted for articles having the same authors and / or the same news source. Based on the responses of the queries, the arithmetic mean of all the past feedbacks is computed, resulting into two final scores for the reliability of the source and the reliability of the article. As mentioned previously and as seen in the figure below, each user feedback should contain a rating. Each rating has a corresponding score from

0 to 100 (i.e. "Highly Unreliable" - 0, "Unreliable" - 25, "Fairly Unreliable" - 50, "Reliable" - 75 and "Highly Reliable" - 100). These corresponding numerical scores are used for the computation of arithmetic mean of all past feedbacks.

In addition to providing two additional analysis points, crowdsourcing was also used with the motivation of collecting a dataset that can be further used to train machine learning models upon. One of the largest issues in the area of fake news detection is the small number of up-to-date datasets of fake and real news. This user feedbacks submitted through this application could potentially result in a multi-labelled dataset that is updated with data on a regular basis. However, there are is one great issue that need to be overcome in the future. Users could provide intentionally misleading reviews, so perhaps moderators would be needed to curate the list of reviews and ban suspicions user accounts or bots that could use the application. Alternatively, maybe a much better solution would be to implement a machine learning model to detect fake review. There is quite a lot of research on fake reviews detection, such as [21][22], that could perhaps be useful in determining the veracity of the users' reviews.



Besides the author's characteristic bombastic and propagandistic writing style, many false or at least ill-researched facts are presented within this article.

First and foremost, even though there are some studies attesting that the COVID-19 vaccines can increase the risc of myocarditis among the young generation, this is a far less probable concern than the potential negative effects that the virus can have on them.

Admittedly, young people do not represent a big group of risk in front of this virus by and large. Nevertheless, there have been reported numerous complications which arised due to COVID-19, which without a shadow of a doubt exceed the problems that were reported due to the vaccine itself.

Figure 18: User feedback form screenshot from the application.

5 Conclusion and Future Work

This thesis explored the proliferating phenomenon of fake news, delved into the novel area of research dedicated to combating and detecting fake news, and provided an overview on the application development process, from inception to completion, concentrating primarily on machine learning approaches and secondarily on other possible approaches, such as web-scraping and crowdsourcing.

The machine learning algorithms used for classifying article news as fake or real based upon their contents and headlines yielded good results, the best ones achieving over 90% accuracy scores, while also performing promisingly well on real input data.

The goal of this thesis to provide a straightforward, well-structured and easy-to-use tool whereby users can perform multi-faceted analyzes on the reliability of online news was met to satisfactory degree, even though I consider this project to be at its first version, with a great amount of additional functionalities that can be implemented and improvements that can be performed.

On the one hand, there is definitely place for lots of improvements to the machine learning algorithms. Even though they performed well from the standpoint of machine learning metrics, the datasets in the field of fake news detection are not yet at an entirely satisfactory stage. Most datasets have flaws, the most important including the limited amount of articles as well as the often disputed, subjective and questionable process through which news are labeled as fake or real. Given the disputable nature of the datasets, unsupervised learning methods could represent a promising alternative that should be explored. Moreover, deep learning algorithms such as LSTM (long short-term memory) could in all likelihood generate promising and perhaps better results as well.

On the other hand, there are many functionalities ideas that did not get to be implemented in this first version of the application. First and foremost, given that social media networks, such as Twitter and Facebook, are the environments in which fake news are most rapidly disseminated, it would be a critical step to integrate this web extension to analyze the reliability of the social media posts. Secondly, automatically performing reliability analyzes and fake news detection on the Google Search results, would categorically enhance the experience of the user and would help them to select more easily the source from which to inform themselves. Lastly, many improvements and additions could be made to the crowdsourcing or user feedback side of the application, in order to curate the list of received feedbacks and collect a reliable and constantly updated dataset of fake and real news collected from past user feedbacks and machine learning predictions, that could be further on used for training new machine learning models.

6 References

[1] Posetti J. and Matthews A. A short guide to the history of 'fake news' and disinformation. 2018.

[2] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. 2017.

[3] Xinyi Zhou and Reza Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Comput. Surv.*, 53(5), sep 2020.

[4] Matthew Helm, Andrew Swiergosz, Heather Haeberle, Jaret Karnuta, Jonathan Schaffer, Viktor Krebs, Andrew Spitzer, and Prem Ramkumar. Machine learning and artificial intelligence: Definitions, applications, and future directions. *Current Reviews in Musculoskeletal Medicine*, 13, 02 2020.

[5] Joost N Kok, Egbert J Boers, Walter A Kosters, Peter Van der Putten, and Mannes Poel. Artificial intelligence: definition, trends, techniques, and cases. *Artificial intelligence*, 1:270–299, 2009.

[6] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[7] Elizabeth D. Liddy. Natural language processing. 2001.

[8] Jason Brownlee. A gentle introduction to logistic regression with maximum likelihood estimation. `https://machinelearningmastery.com/logistic-regression-with-maximum-likelihood-estimation/`. Accessed: 2022-08-20.

[9] Benjamin D. Horne and Sibel Adali. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *ArXiv*, abs/1703.09398, 2017.

[10] Yimin Chen, Nadia Conroy, and Victoria Rubin. Misleading online content: Recognizing clickbait as “false news”. 11 2015.

[11] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 231–240, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[12] How photos fuel the spread of fake news. `https://www.wired.com/2016/12/photos-fuel-spread-fake-news/`. Accessed: 2022-08-20.

[13] Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108, 2013.

[14] Wikipedia:reliable sources/perennial sources. `https://en.wikipedia.org/wiki/Wikipedia:Reliable_sources/Perennial_sources`. Accessed: 2022-08-20.

[15] Davide Fucci, Hakan Erdogmus, Burak Turhan, Markku Oivo, and Natalia Juristo. A dissection of the test-driven development process: Does it really matter to test-first or to test-last? *IEEE Transactions on Software Engineering*, 43(7):597–614, 2017.

[16] Fake and real news dataset. `https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset`. Accessed: 2022-08-20.

[17] Ahmed H, Traore I, and Saad S. Detecting opinion spams and fake news using text classification. *Security and Privacy*, 2017.

[18] 5+ data leaks [100 `https://www.kaggle.com/code/mosewintner/5-data-leaks-100-acc-1-word-99-6-acc#Other-features-and-dead-giveaways`. Accessed: 2022-08-20.

[19] Clickbait dataset. `https://www.kaggle.com/datasets/amananandrai/clickbait-dataset`. Accessed: 2022-08-20.

[20] 2019/grants/sourceror: The wikipedia community's platform against disinformation. `https://wikiconference.org/wiki/2019/Grants/Sourceror:_The_Wikipedia_community%27s_platform_against_disinformation`. Accessed: 2022-08-20.

[21] Rami Mohawesh, Shuxiang Xu, Son N. Tran, Robert Ollington, Matthew Springer, Yaser Jararweh, and Sumbal Maqsood. Fake reviews detection: A survey. *IEEE Access*, 9:65771–65802, 2021.

[22] Shaohua Jia, Xianguo Zhang, Xinyue Wang, and Yang Liu. Fake reviews detection based on lda. In *2018 4th International Conference on Information Management (ICIM)*, pages 280–283, 2018.

[23] Choudhary A. and Arora A. Linguistic feature based learning model for fake news detection and classification. *Expert Systems with Applications*, 2020.