ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ
ΣΠΟΥΔΩΝ ΣΤΗ ΔΙΟΙΚΗΣΗ ΕΠΙΧΕΙΡΗΣΕΩΝ
MASTER IN BUSINESS ADMINISTRATION

INTERDEPARTMENTAL MASTER STUDIES PROGRAM ON BUSINESS ADMINISTRATION-MBA

**Thesis**

# USING MACHINE LEARNING IN SOCIAL MEDIA SYSTEMS

of the student

ARTEMIS THEODOSOPOULOU of ATHANASIOS

mba18021

Supervisor: Tarabanis Konstantinos

Submitted as required for the master on Business Administration obtainment

Thessaloniki, September 2019

*Στην οικογένειά μου*

# Acknowledgements

Θα ήθελα να ευχαριστήσω τους ανθρώπους που με την παρουσία και τη συνεργασία τους με βοήθησαν κατά την πραγματοποίηση της εργασίας μου:

Τον καθηγητή του Τμήματος Οργάνωσης και Διοίκησης Επιχειρήσεων, κ. Ταραμπάνη Κωνσταντίνο, για την εμπιστοσύνη του, την καθοδήγησή του, τις συμβουλές του και την ευκαιρία που μου έδωσε να γίνω μέλος μίας ομάδας που αποτελεί παράδειγμα ομαδικότητας και συνεργατικότητας, κατά την εκπόνηση της διπλωματικής μου εργασίας.

Τον κ. Καλαμπόκη Βαγγέλη και την κα. Καραμάνου Αρετή για την πολύτιμη βοήθειά τους και επειδή με ενέπνευσαν να γίνω και να θέλω να γίνομαι καλύτερη.

Την οικογένειά μου και τους φίλους μου που είναι πάντα δίπλα μου και με στηρίζουν σε κάθε βήμα μου…

*«Πώς να πιστέψουν οι άπιστοι τι θάματα μπορεί να γεννήσει η πίστη; Ξεχνούν πως η ψυχή του ανθρώπου γίνεται παντοδύναμη, όταν συνεπαρθεί από μια μεγάλη ιδέα. Τρομάζεις όταν, ύστερα από πικρές δοκιμασίες, καταλαβαίνεις πως μέσα μας υπάρχει μια δύναμη που μπορεί να ξεπεράσει τη δύναμη του ανθρώπου, τρομάζεις. Γιατί δεν μπορείς πια να βρεις δικαιολογίες για τις ασήμαντες ή άνανδρες πράξεις σου, ρίχνοντας το φταίξιμο στους άλλους. Ξέρεις πως εσύ, όχι η μοίρα, όχι η τύχη, μήτε οι άνθρωποι γύρω σου, εσύ μονάχα έχεις, ό,τι και αν κάμεις, ό,τι και αν γίνεις ακέραιη την ευθύνη. Και ντρέπεσαι τότε να γελάς, ντρέπεσαι να περιγελάς αν μια φλεγόμενη ψυχή ζητάει το αδύνατο. Καλά πια καταλαβαίνεις πως αυτή είναι η αξία του ανθρώπου: να ζητάει και να ξέρει πως ζητάει το αδύνατο και να είναι σίγουρος πως θα το φτάσει, γιατί ξέρει πως αν δε λιποψυχήσει, αν δεν ακούσει τι του κανοναρχάει η λογική, μα κρατάει με τα δόντια την ψυχή του κι εξακολουθεί με πίστη, με πείσμα να κυνηγάει το αδύνατο, τότε γίνεται το θάμα, που ποτέ ο αφτέρουγος κοινός νους δε μπορούσε να το μαντέψει: το αδύνατο γίνεται δυνατό.»*

***Απόσπασμα από το βιβλίο του Ν. Καζαντζάκη «Ο Καπετάν Μιχάλης»***

# Abstract

Nowadays, vast use of Internet and especially, social media, as primary sources of information on everything is happening around the world, has unfortunately facilitated the spreading of fake news at the same time. Thus, everyone can alter real news and publish it on a news website or their social media account, or even invent news and promote it as real, misinforming and even disorienting in this way the public. For this reason, it is crucial to find ways to detect fake news as fast as possible, since fake news dissemination can sometimes be proved destructive, mainly as far as political and social issues are concerned, which have the stronger impact on people's lives. Classification algorithms use is one way researchers have found in order to deal with this serious problem. In this thesis, we are going to present such a solution, which deploys Data Science and Machine Learning, in order to build a classifier for fake news detection. More specifically, after studying various articles concerning fake news classification, we are going to implement and evaluate our own classifier in a kernel created in Kaggle platform.

# Table of contents

# Figures list

# Tables list

# Abbreviations

**AI:** Artificial Intelligence

**API:** Application Programming Interface

**AUC:** Area Under Curve

**BOW:** Bag of Words

**CNN:** Convolutional Neural Network

**DT or DTR or J48:** Decision Tree

**FN:** False Negative

**FP:** False Positive

**FR_NB:** Feature-Rank Naïve Bayes

**KNN:** K-Nearest Neighbor

**LIWC:** Linguistic and Word Count

**LR or LOG:** Logistic Regression

**LSTM:** Long Short-Term Memory

**LSVM:** Linear Support Vectors Machine

**MaxEnt:** Maximum Entropy

**ML:** Machine Learning

**MLP:** Multi Layer Perceptron

**NB:** Naïve Bayes

**NLP:** Natural Language Processing

**NLTK:** Natural Language Toolkit

**OSN:** Online Social Network

**PCFG:** Probabilistic Context Free Grammar

**RF or RFO:** Random Forest

**RNN:** Recurrent Neural Network

**ROC:** Receiver Operating Characteristic

**SGD:** Stochastic Gradient Descent

**SVM:** Support Vectors Machine

**TF:** Term Frequency

**TN:** True Negative

**TP:** True Positive

**TFIDF:** Term Frequency-Inverse Document Frequency

**XGB:** Extreme Gradient Boosting

# 1. Introduction

## 1.1 Problem definition

First of all, we have to define what “fake news” means; as [12] states, “fake news” is fabricated information that mimics news media content in form but not in organizational process or intent. Fake-news outlets, in turn, lack the news media’s editorial norms and processes for ensuring the accuracy and credibility of information. Fake news overlaps with other information disorders, such as misinformation (false or misleading information) and disinformation (false information that is purposely spread to deceive people).

Fake news has primarily drawn recent attention in a political context but it has also been documented in information promulgated about topics such as vaccination, nutrition, and stock values. It is particularly pernicious in that it is parasitic on standard news outlets, simultaneously benefiting from and undermining their credibility [12].

While fake news is not a new phenomenon, questions such as why it has emerged as a world topic and why it is attracting increasingly more public attention are particularly relevant at this time [18]. Internet and social media made the access to the news information much easier and comfortable; Internet users can follow the events of their interest in online mode and spread of the mobile devices makes this process even easier. But with great possibilities come great challenges. Mass media have a huge influence on the society and as it often happens, there is someone who wants to take advantage of this fact. Sometimes, to achieve some goals mass media may manipulate the information in different ways. This leads to producing of the news articles that are not completely true or even completely false. There even exist lots of websites that produce fake news almost exclusively. They deliberately publish hoaxes, propaganda and disinformation purporting to be real news - often using social media to drive web traffic and amplify their effect. The main goal of fake news websites is to affect the public opinion on certain matters (mostly political). For example, social media coverage of crisis events may be used by authorities for effective disaster management or by malicious entities to spread rumors and fake news for financial or political benefit. Examples of such websites may be found in Ukraine, United States of America, Germany, China and lots of other countries. Thus, fake news is a global issue as well as a global challenge [8]. Fake news has even been named as 2017's word of the year by Collins dictionary [16].

The rise of fake news highlights the erosion of long-standing institutional bulwarks against misinformation in the Internet age. Concern over the problem is global. However, much remains unknown regarding the vulnerabilities of individuals, institutions, and society to manipulations by malicious actors. Thus, a new system of "safeguards" is needed, in order to deal with this serious and worldwide problem [12].

## 1.2 Research scope and objectives

In this thesis, we are going to implement and evaluate a classifier for fake news detection. After collecting a series of online scientific articles related to fake news detection, we are going to find out which particular methodologies are used, as far as the pre-processing steps followed, the features extraction, the algorithm selection and the evaluation metrics used, are concerned. Then, we are going to select those steps which fit the most to the building of a fake news classifier using Natural Language Processing (NLP), in order to build our own one, selecting a dataset of news articles via Kaggle platform and more specifically, via https://www.kaggle.com/c/fake-news. In this way, our goal is to present how we can deal with the problem of fake news detection, presenting step by step the process we followed, based on the literature sources.

## 1.3 Thesis structure

In chapter 2, we are going to present general information concerning fake news impact, as well as information about fake news dissemination on Online Social Networks (OSNs), in order to finally present how we can deal with fake news detection problem. In chapter 3, we are going to present the methodological approach of this thesis, matching each step of it with the following chapters. In chapter 4, we are going to present the articles studied in this thesis, making a brief description of each of them. In chapter 5, we are going to present the various datasets we studied, and then, we are going to describe the dataset we selected for the building of our own classifier. In chapter 6, we are going to present the pre-processing and the features extraction done in the articles studied, in order to select then and present the steps we followed in our kernel. In chapter 7, we are going to present the various classification algorithms used in the articles studied, as well as the evaluation metrics used, in order to select then and present the algorithms we used in our kernel, as well as the metrics for their performance evaluation. In chapter 8, we are going to present the conclusions we finally drew.

# 2. Fake news problem

## 2.1 Fake news impact

Fake news is now viewed as one of the greatest threats to democracy, journalism, and freedom of expression. It has weakened public trust in governments and its potential impact on the contentious “Brexit” referendum and the equally divisive 2016 U.S. presidential election - which it might have affected - is yet to be realized. The reach of fake news was best highlighted during the critical months of the U.S. presidential election campaign, where the top twenty frequently-discussed false election stories generated 8,711,000 shares, reactions, and comments on Facebook, ironically, larger than the total of 7,367,000 for the top twenty most discussed election stories posted by 19 major news websites. Economies are not immune to the spread of fake news either, with fake news being connected to stock market fluctuations and massive trades. For example, fake news claiming that Barack Obama was injured in an explosion wiped out $130 billion in stock value. These events and losses have motivated fake news research and sparked the discussion around fake news, as observed by skyrocketing usage of terms such as “post-truth” - selected as the international word of the year by Oxford Dictionaries in 2016 [18].

The main cause leading to fake news dissemination is the fact that it can be created and published online faster and cheaper when compared to traditional news media, such as newspapers and television. The rise of social media and its popularity also plays an important role in this surge of interest [18]. About 47% of Americans report getting news from social media, often or sometimes, with Facebook being the dominant source by far. Social media are key conduits for fake news sites. Indeed, Russia successfully manipulated all of the major platforms during the 2016 United States election, according to recent congressional testimony [12]. With the existence of an echo chamber effect on social media, biased information is often amplified and reinforced. Furthermore, as an ideal platform to accelerate fake news dissemination, social media breaks the physical distance barrier among individuals, provides rich platforms to share, forward, vote, and review, and encourages users to participate and discuss online news. This surge of activity around online news can lead to grave repercussions, but also substantial potential political and economic benefits. Such generous benefits encourage malicious entities to create, publish and spread fake news [18].

Internet platforms have become the most important enablers and primarily conduits of fake news, since it is inexpensive to create a website that has the trappings of a professional news organization. It has also been easy to monetize content through online ads and social media dissemination. The Internet not only provides a medium for publishing fake news but offers tools to actively promote dissemination [12].

Social and psychological factors play an important role in fake news gaining public trust and further facilitate the spread of fake news. For instance, humans have been proven to be irrational and vulnerable when differentiating between truth and falsehood while overloaded with deceptive information. Studies in social psychology and communications have demonstrated that human ability to detect deception is only slightly better than chance: typical accuracy rates are in the 55%-58% range, with a mean accuracy of 54% over 1,000 participants in over 100 experiments. The situation is more critical for fake news compared to other types of information, as for news, a representative of authenticity and objectivity, is relatively easier to gain public trust. In addition, individuals tend to trust fake news after repeated exposures (i.e., validity effect), or if it confirms their pre-existing knowledge (i.e., confirmation bias). Peer pressure can also at times “control” their perception and behavior (i.e., bandwagon effect) [18].

## 2.2 Fake news on Online Social Networks (OSNs)

An OSN is an online platform of websites and applications dedicated to facilitate social interactions and personal relationships. Examples of OSNs include Facebook, Twitter, Instagram, Snapchat etc. The users of OSNs can be involved in malicious activities like the spreading of fake news with no reliable sources. The use of them has become a part of people’s daily activities, but it can cause significant harm by propagating misinformation.

Twitter, a social network application, which has grown highly popular in the 21st century, is ranked fourth most popular OSN in the United States of America. Twitter has about 330 million monthly active users with about 500 million of tweets per day. Twitter provides users the freedom to communicate, collaborate, network and share information [1].

Twitter has been widely used during emergencies, such as wildfires and earthquakes. Journalists have hailed the immediacy of the service that allowed reporting breaking news quickly, in many cases, more rapidly than most mainstream media outlets. However, it was also reported that rumors are propagated via tweets in a different way than the actual news. During hurricane Sandy, 86% of Tweets spreading fake images were re-tweets and they were very few original tweets [11].

Information dissemination through these platforms is their most attractive feature, as it is known to be speedy and cost effective. The fact that users are allowed to express themselves with little to no control is also another very attractive aspect of these platforms. As users are afforded the freedom to publish content with no supervision, the problem of information credibility on social networks has also risen in recent years. Crafty users of these platforms can spread information maliciously for reasons that may not be compatible with the good of society. Users are becoming wary that rumors that are spread through online social networks can have detrimental effects. Research on information credibility is thus the best solution to the problem of how to assess the credibility of information and perhaps mitigate the dissemination of misinformation [5].

Currently, researchers have employed various methodologies in studies on information credibility. Some of them consider the problem to be one of classification that should be solved in an automated fashion using machine learning or graph-based algorithms. Others view it as a cognitive problem requiring human-centric verification. Some authors have looked at how various aspects of social media, such as the effect of the name value and user-connectedness, influence users' judgments concerning credibility. Some researchers have gone so far as to create systems to assess credibility automatically in real time. Such systems include TweetCred and Twitter-Trails, created for tweets truthfulness assessment [5].

The main challenge in assessing the credibility of information dissemination on OSNs is the nature of the networks; they are very complex and grow in users and content every day. According to [5], among the many challenges related to studying credibility on social networks and the web are the following:

1. The complexity of social networks and the web creates difficulty in identifying resources for use in studying and assessing credibility.
2. OSNs by their very nature evolve dynamically over time and become very large in size, with various structures that make it difficult to obtain the information needed to discern the credibility of users.
3. The credibility of a user is influenced continuously by various factors, such as changes in the social topography, other users' behavior, preferences and context.
4. Malicious activities can evade existing spam filters through various means. For example, in Twitter malicious users can purchase followers or use tools to automatically generate fake accounts and post tweets with the same meaning but different words.
5. The process of evaluating solutions has also been a problem in terms of resources, given that most researchers are limited in terms of the extent to which they can test their work (Twitter and OSNs limitations).

Thus, it is very difficult to measure the credibility of a user in these networks and to verify his/her posts. As OSNs have become more useful for disseminating information to wider audiences, addressing the above-mentioned challenges to determine the credibility of users in OSNs requires the development of robust techniques for measuring user and content credibility [5].

## 2.3 Fake news detection

As the Internet community and the speed of the spread of information are growing rapidly, automated fact checking of Internet content has gained plenty of interests in the Artificial Intelligence (AI) research community. The goal of automatic fake news detection is to reduce the human time and effort to detect fake news and help to stop spreading them. The task of fake news detection has been studied from various perspectives with the development in subareas of Computer Science, such as Machine Learning (ML), Data Mining and Natural Language Processing (NLP) [14].

Many scientists believe that the fake news issue may be addressed by means of ML and AI, because AI algorithms have recently started to work much better on lots of classification problems, such as image recognition, voice detection and so on, since hardware is cheaper and bigger datasets are available [8].

First, dealing with a fake news detection problem includes a dataset selection as the input of the classifier. This input can be text ranging from short statements to entire articles. Additional information such as speakers' identity can be appended. Then, according to [14], the problem is formulated as a classification, regression or clustering problem, but classification is more frequently used:

- **Classification:** The most common way is to formulate the fake news detection as a binary classification problem. However, categorize all the news into two classes (fake or real) is difficult because there are cases where the news is partly real and partly fake. To address this problem, adding additional classes is a common practice. There are mainly two ways of adding additional classes. One is to set a category for the news which is neither completely real nor completely fake. The other one is to set more than two degrees of truthfulness. The latter method reflects human judgments more delicately.
- **Regression:** Fake news detection can also be formulated as a regression task, where the output is a numeric score of truthfulness. Formulating the task in this way can make it less straightforward to do the evaluation. Usually, evaluation is done by calculating the difference between the predicted scores and the ground truth scores, or using Pearson/Spearman Correlations. However, since the available datasets have discrete ground truth scores, the challenge here is how to convert the discrete tables to numeric scores.
- **Clustering:** One of the conditions for fake news classifiers to achieve good performances is to have sufficient labeled data. However, to obtain reliable labels requires a lot of time and labor. Therefore, semi-supervised and unsupervised methods are proposed. The task is then formulated as a clustering problem instead of a classification one.

# 3. Methodology

## 3.1 Introduction

In this chapter, the methodology approach, followed in this thesis, is presented. This approach is divided into 5 steps, which are described below.

## 3.2 Methodology steps

The methodology consists of the following steps:

- We collect online scientific articles via Google Scholar website, which are related to fake news detection. We focus our research on articles which have been published recently, and more specifically after 2016, and derive either from conferences or journals. The keywords we use to find these articles are: <FAKE NEWS>, <FAKE NEWS DETECTION>, <FAKE NEWS CLASSIFIERS>, <FAKE NEWS> AND <MACHINE LEARNING>. Then, we briefly present each article found.
- We divide the datasets found in the literature sources in two categories, news articles and tweets, and we describe each dataset used. Then, after reviewing two datasets of news articles, the one found in the Datacamp course "Natural Language Processing Fundamentals in Python" (https://www.datacamp.com/courses/natural-language-processing-fundamentals-in-python) and the other in the Kaggle InClass Prediction Competition "Fake News: Build a system to identify unreliable news articles" (https://www.kaggle.com/c/fake-news), we select the dataset we are going to use for the building of our own classifier. Our dataset selection is based on the dataset size, as well as its data balance, as far as the number of news entries labeled as fake and the number of news entries labeled as real is concerned.
- We present the pre-processing and the features extraction steps followed in each research of the two aforementioned categories. Then, we describe each step separately. Finally, we present the steps we are going to follow for the building of our own classifier, presenting the corresponding part of our code and commenting on it.
- We present the algorithms, along with their evaluation metrics, used in each category, news articles and tweets. Then, we select the algorithms we are going to use, as well as the metrics for their performance evaluation, presenting the corresponding parts of our code

and commenting on them. The algorithms and the evaluation metrics selection is based on their frequency of use, as well as on which algorithms have the best performance compared to others.

- We present the conclusions we draw via the whole process followed and we make suggestions for future research.

Each of the steps presented above is going to be a discrete chapter of this thesis.

# 4. Articles Collection

## 4.1 Introduction

We began our methodology by looking for online scientific articles. The articles, which were finally collected, are the following:

**Table 4.1:** Articles collected

| Article | Year | Conference/Journal | Reference |
|---|---|---|---|
| **Classification for Authorship of Tweets by Comparing Logistic Regression and Naïve Bayes Classifiers** | 2018 | Conference | [1] |
| **Using Machine Learning for News Verification** | 2018 | Conference | [2] |
| **Detection of Online Fake News Using N-gram Analysis and Machine Learning Techniques** | 2017 | Conference | [3] |
| **Fake News Identification on Twitter with Hybrid CNN and RNN models** | 2018 | Conference | [4] |
| **A Credibility Analysis System for Assessing Information on Twitter** | 2016 | Journal | [5] |
| **Automatically Identifying Fake News in Popular Twitter Threads** | 2017 | Conference | [6] |
| **Evaluating Machine Learning Algorithms for Fake News Detection** | 2017 | Conference | [7] |

| | | | |
|---|---|---|---|
| **Fake News Detection Using Naïve Bayes Classifier** | 2017 | Conference | [8] |
| **Fake News Detection** | 2018 | Conference | [9] |
| **Automatic Detection of Fake News on Social Media Platforms** | 2017 | Conference | [10] |
| **Identifying Tweets with Fake News** | 2018 | Conference | [11] |
| **The Science of Fake News** | 2018 | Journal | [12] |
| **Development of Classification Models for Fake News Detection** | 2017 | Individual Research Project Report | [13] |
| **A Survey on Natural Language Processing for Fake News Detection** | 2018 | - | [14] |
| **Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking** | 2017 | Conference | [15] |
| **A Deep Ensemble Framework for Fake News Detection and Classification** | 2018 | Conference | [16] |
| **Fake News Detection on Social Media: A Data Mining Perspective** | 2017 | Journal | [17] |
| **Fake News: A Survey of Research, Detection Methods, and Opportunities** | 2018 | Journal | [18] |

## 4.2 Brief description of each article

In research [1], the importance of authorship attribution is pointed out, since anonymous information distribution increases with the rapid increase of internet usage around the world. A dataset consisting of known and unknown authors' tweets is used and the aim is to classify authors of tweets using Naïve Bayes (NB) and Logistic Regression (LR) classifiers.

In research [2], after selecting a public dataset located in a github repository, which contains news articles collected in total between the years 2015 and 2017, written in English, NB classifier is used for news classification.

In research [3], the dataset selected includes news articles that revolve around the 2016 U.S. elections and the articles which discuss topics around it. The aim of this research is to present a fake news detection model, investigating six different machine learning techniques, namely K-Nearest Neighbor (KNN), Support Vectors Machine (SVM), Linear Support Vectors Machine (LSVM), (Decision Tree) DT, (Stochastic Gradient Descent) SGD and LR.

In research [4], a framework is proposed that detects and classifies fake news messages from Twitter posts using hybrid of Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models.

In research [5], the aim is to analyze and assess the credibility of tweets and users, using a dataset composed of real-world data from Twitter regarding the Saudi-led campaign against Houthi rebels in Yemen. DT, NB, Feature-Rank Naïve Bayes (FR_NB) and Random Forest (RF) are the classifiers used.

In research [6], a method for automating fake news detection on Twitter is developed by learning to predict accuracy assessments in two credibility-focused Twitter datasets, CREDBANK and PHEME. This method is then applied to Twitter content sourced from BuzzFeed's fake news dataset. DT and RF are the classifiers used.

In research [7], a dataset published by Signal Media is selected, containing about 1 million articles from a variety of news sources from September 2015. This dataset is tested on multiple classification algorithms, namely SVM, DT, RF, SGD and (Extreme Gradient Boosting) XGB.

In research [8], dataset selected by BuzzFeed News is used, containing information about Facebook posts, each of which represents a news article. Then, NB algorithm is used for news classification.

In research [9], using a dataset deriving from Github, which contains Facebook posts, the aim is to predict whether a post is real or fake, using NB classifier.

In research [10], after selecting a dataset from BuzzFeed News, containing Facebook news posts during the U.S. presidential election 2016, SVM, RF, XGB, DT and LR classifiers are used, in order to automatically identify fake news.

In research [11], the aim is to identify tweets with fake news content, using a dataset which consists of tweets concerning Hurricane Sandy and miscellaneous events. SVM and DT are the classifiers used.

In research [13], the dataset derives from Kaggle, as far as fake news articles are concerned, and from reliable news sources, according to Forbes, as far as real news articles are concerned. LSVM, LR, NB and RF are the classifiers used for fake news detection.

In research [15], a dataset selected from politifact.com is used. NB, LR and LSTM are the classifiers used in order to determine the truthfulness of text.

In research [16], LIAR dataset is selected, in order to detect and classify fake news, using CNN and LSTM models.

Researches not mentioned above, were deployed in order to receive general information concerning the problem of fake news detection and its impact on various fields, mainly the political and social ones.

# 5. Datasets

## 5.1 Introduction

After collecting our articles, we divided them into two categories. The first category consists of those whose datasets contain news articles and the second category consists of those whose datasets contain tweets. We present the two categories in the following table:

**Table 5.1:** Datasets categories

| | **News articles** | **Tweets** |
|---|---|---|
| **[1]** | | 🟥 |
| **[2]** | 🟦 | |
| **[3]** | 🟦 | |
| **[4]** | | 🟥 |
| **[5]** | | 🟥 |
| **[6]** | | 🟥 |
| **[7]** | 🟦 | |
| **[8]** | 🟦 | |
| **[9]** | 🟦 | |
| **[10]** | 🟦 | |
| **[11]** | | 🟥 |
| **[13]** | | 🟥 |
| **[15]** | 🟦 | |
| **[16]** | 🟦 | |

In this chapter, we are going to describe the datasets used in both categories, referring to their size, their subject and the sources from which they derive. Then, we are going to present the sources where we looked for datasets consisting of real and fake news, in order to finally select the dataset we used for the building of our classifier.

## 5.2 Description of news articles datasets

Below, a table, which includes all datasets used, is presented:

**Table 5.2:** News articles datasets

| | [2] | [3] | [7] | [8] | [9] | [10] | [15] | [16] |
|---|---|---|---|---|---|---|---|---|
| **Reuters.com** | | ■ | | | | | | |
| **Kaggle** | | ■ | | | | | | |
| **BuzzFeed News** | | | | ■ | | ■ | | |
| **LIAR** | | | | | | | | ■ |
| **Politifact.com** | | | | | | | ■ | |
| **Reliable news sources according to Forbes** | | | ■ | | | | | |
| **Signal Media** | | | | | | | | |
| **Github** | ■ | | | | ■ | | | |

- **Github:**

The first dataset deriving from Github is a public dataset located in a github repository (https://github.com/GeorgeMcIntire/fake_real_news_dataset), compiled in equal parts for ten thousand five hundred (10,558) news items collected in total between the years 2015 and 2017 written in English with their title, full text and false or true label, which were taken from different media, making scrapping processes in news web portals for half of real news and news from a published dataset in Kaggle conformed only by false news [2].

The second dataset produced by GitHub, contains 11,000 news articles tagged as real or fake. It has 6335 rows and 4 columns. The 4 columns consist of index, title, text and label. News categories included in this dataset include business, science and technology, entertainment, and health. The authenticity of this dataset lies in the fact that it was checked by journalists and then labeled as “Real” or “Fake”. Title involves the minimal information required to understand the news article similar to the heading of the newspaper, which describes the content within. Text entails a detailed description of the news article embedded with peculiarities like location, details, people involved and their background etc. Label is basically a tag which tells whether the news articles are “Fake” or “Real” [9].

- **Reuters.com and Kaggle:**

This dataset was entirely collected from real world sources. News was collected from Reuters.com, news website for real news articles. As for the fake news, it was collected from a fake news dataset on Kaggle.com. The collector of the dataset collected fake news items from unreliable websites that Politifact has been working with Facebook to stamp out. Politifact is a site led by Tampa Bay Times journalists who actively fact-check suspicious statements. One unique quality of Politifact is that each quote is evaluated on a 6-point scale of truthfulness ranging from "True" (factual) to "Pants-on-Fire False" (absurdly false). This scale allows for distinction between categories like mostly true (the facts are correct but presented in an incomplete manner) or mostly false (the facts are not correct but are connected to a small kernel of truth). 12,600 fake news articles were used from Kaggle.com and 12,600 truthful articles. The focus was only on political news article because they are currently the main target of spammers. The news article from both fake and truthful categories happened in the same timeline, specifically in 2016. Each of the articles length is bigger than 200 characters. For every article, the following information is available:

- Article Text
- Article Type
- Article label (fake or truthful)
- Article Title
- Article Date

The focus was only on news articles that revolve around the 2016 United States elections and the articles that discuss topics around it. In total, 2,000 articles were picked from real and fake articles previously collected, 1,000 fake articles and 1,000 real articles. The 2,000 articles represent a subset of the dataset described in the previous section that focuses only on politics [3].

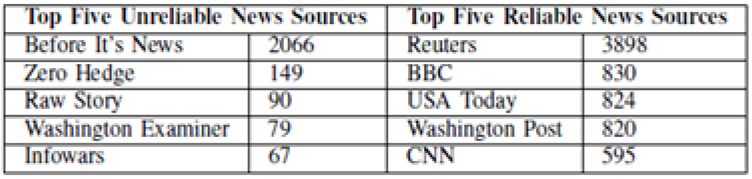
- **Signal Media:**

This dataset is in conjunction with the Recent Trends in News Information Retrieval 2016 conference, to facilitate conducting research on news articles. The dataset contains about 1 million articles from a variety of news sources from September 2015. Sources include major news outlets like Reuters as well as local news sources and blogs. From this dataset, articles from verified reliable sources (labeled as 0) and verified unreliable sources (labeled as 1) are included.

The cleaned dataset contains 11051 articles. 3217 (29%) are labeled as fake. The reliable articles come from 14 unique sources. The unreliable articles come from 61 unique sources. In particular, for fake news the examples are heavily drawn from one source: Before It's News [7].

In the following Table I, top unreliable and reliable sources are compared by article frequency:

**Table 5.3:** Comparison of top unreliable and reliable sources by article frequency



| Top Five Unreliable News Sources | | Top Five Reliable News Sources | |
|---|---|---|---|
| Before It's News | 2066 | Reuters | 3898 |
| Zero Hedge | 149 | BBC | 830 |
| Raw Story | 90 | USA Today | 824 |
| Washington Examiner | 79 | Washington Post | 820 |
| Infowars | 67 | CNN | 595 |

[7]

- **<u>BuzzFeed News:</u>**

The two studies – [8] and [10] – are relied on ground-truth labeling of fake and non-fake news postings on Facebook from BuzzFeed, which are augmented by retrieving additional data via the Facebook developer Application Programming Interface (API). BuzzFeed selected a total of nine self-proclaimed news pages, which are active on Facebook and have earned the coveted verified blue check mark from Facebook, which gives them an additional layer of credibility on the platform, three left-wing associated pages (The other 98%, 3.24M fans; Addicting Info, 1.22M fans; Occupy Democrats, 4.14M fans), three right-wing associated pages (Eagle Rising, 0.62M fans; Right Wing News, 3.38M fans; Freedom Daily, 1.36M fans) and three mainstream associated outlets (Politico, 1.18M fans; CNN Politics, 1.9M fans; ABC News Politics, 0.46M fans).

BuzzFeed news employees logged and fact-checked each of the posts from these nine pages, that was published on them over a period of seven weekdays (Sept. 19-23 and Sept. 26-27, 2016). They labeled each of the posts as "mostly true", "mostly false", "mixture of true and false" and "no factual content". They also gathered additional data:

Facebook engagement numbers (shares, comments and reactions) for each post were added from the Facebook API. They also noted whether the post was a link, photo, video or text. Raters were asked to provide notes and sources to explain their rulings of “mixture of true and false” or “mostly false”. They could also indicate whether they were unsure of a given rating, which would trigger a second review of the same post in order to ensure consistency. Any discrepancies between the two ratings were resolved by a third person. The same person conducted a final review of all posts that were rated mostly false to ensure they warranted that rating. As a sanity check, posts in the final sample that were assigned the label “mostly false” were fact-checked again.

In the end, BuzzFeed team rated and gathered data on 2,282 posts. There were 1,145 posts from mainstream pages, 666 from right-wing pages and 471 from left-wing pages. The difference in the number of posts for each group is a result of them publishing with different frequencies.

As far as the second research is concerned, ground-truth data of human fact-checked fake and non-fake articles posted during the United States presidential elections 2016 are utilized. Specifically, a balanced sample of 460 Facebook postings is drawn of nine left-wing, right-wing and mainstream media outlets as well as 125,725 associated user comments.

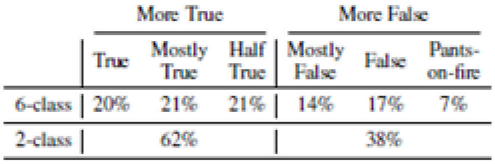
More specifically, from the BuzzFeed news sample of 2,282 labeled Facebook news posts, posts from “no factual content” category are removed, posts where raters were unsure about their rating, incomplete observations and posts without any comments or reactions. The categories “mixture of true and false” and “mostly false” are combined to the category “fake” and the remaining posts belong to the category “non-fake”. The “non-fake” observation category is then randomly downsampled to yield a balanced sample of 460 posts in the final sample. Using these posts, all metrics (e.g. number of shares) are updated and additional data (e.g. images used in the posts) are downloaded as well as all 125,725 associated comments on January 18th, 2017.

- **Politifact.com:**

10,483 labeled statements in total were collected from Politifact and its spin-off sites and a subset of 4,366 statements was analyzed, which are direct quotes by the original speaker.

Quotes are split into training/development/test set of {2575, 712 and 1074} statements, respectively, so that all of each speaker's quotes are in a single set. The experiment was run in two settings, one considering all 6 classes and the other considering only 2 (treating the top three truthful ratings as true and the lower three as false), as shown in the Table 4 below [15]:

**Table 5.4:** PolitiFact label distribution



| | More True | | | More False | | |
|---|---|---|---|---|---|---|
| | True | Mostly True | Half True | Mostly False | False | Pants-on-fire |
| 6-class | 20% | 21% | 21% | 14% | 17% | 7% |
| 2-class | | 62% | | | 38% | |

[15]

- **<u>LIAR:</u>**

This dataset is annotated with six fine-grained classes and comprises of about 12,800 annotated short statements along with various information about the speaker. The statements, which were mostly reported during the time interval 2007 to 2016, are considered for labeling by the editors of Politifact.com. Each row of the data contains a short statement, a label of the statement and the 11 other columns correspond to various information about the speaker of the statement. The dataset consists of three sets, namely a training set of 10,269 statements, a validation set of 1,284 statements and a test set of 1,266 statements [16].

## 5.3 Description of tweets datasets

Below, a table, which includes all datasets used, is presented:

**Table 5.5:** Tweets datasets

| | [1] | [4] | [5] | [6] | [11] | [13] |
|---|---|---|---|---|---|---|
| **Kaggle** | | | | | | 🟥 |
| **Tweets from known/unknown authors** | 🟥 | | | | | |
| **CREDBANK** | | | | 🟥 | | |
| **PHEME** | | | | 🟥 | | |
| **Tweets centered on 5 rumor stories** | | 🟥 | | | | |
| **Hurricane Sandy and miscellaneous events** | | | | | 🟥 | |
| **Reliable news sources according to Forbes** | | | | | | 🟥 |
| **Tweets about Saudi-led campaign against Houthi rebels in Yemen** | | | 🟥 | | | |

- **Tweets from known/unknown authors:**

This dataset was streamed from Twitter, with maximum of three thousand tweets per author. The dataset consists of two features, the author (user name) and the tweets. The author is the response variable, which is made up of two classes, known and unknown authors. The known authors consist of twelve celebrities and prominent users on Twitter, while the unknown authors also consist of twelve regular Twitter users, but not as popular as the known authors. The labels were labeled known and unknown based on the author's class as a celebrity or not.

The RESTful Twitter API for fetching of tweets was used, since the focus is on the users' tweets. The collected tweets were transformed into an array that populated the csv document.

The dataset consists of the author and the tweets only and all re-tweets which are messages retransmitted from other users were removed, since the goal of the research is on authorship attribution. The Twitter dataset used for this research is made up of 46,895 tweets with 22,333 tweets in the known class and 24,566 in the unknown class of authors [1].

- **Tweets centered on 5 rumor stories:**

This dataset consists of approximately 5,800 tweets centered on five rumor stories. The dataset consists of original tweets and they are labeled as rumors or non-rumors. The events were widely reported in online, print and conventional electronic media such as radio and television at the time of occurrence:

- CharlieHebdo
- SydneySiege
- Ottawa Shooting
- Germanwings Crash
- Ferguson Shooting [4]

- **Tweets about Saudi-led campaign against Houthi rebels in Yemen:**

The proposed system was applied to a set of real-world data from Twitter regarding the Saudi-led campaign against Houthi rebels in Yemen. The full dataset was divided into three separate datasets. The crawled data consisted of 1,416,443 tweets by 489,330 unique users. Datasets A and T were generated from tweets related to the Yemen civil war. Dataset T was generated from tweets in mid-December 2015 using keywords “Taiz” and dataset A, which contains tweets related to the condition of the city of Aden after pro-government fighters recaptures the city, was compiled using “Aden” as the keyword. For the purposes of this research, the collected tweets were divided into two groups: experimental data and prediction data. These datasets have no tweets in common. After removing the intersection of A and T, 23,000 tweets were randomly sampled concerning topic T from the experimental dataset and 25,000 tweets concerning topic A. Then, 4,000 tweets were randomly sampled concerning topic T from the prediction dataset and 7,000 tweets concerning topic A. In addition, from each dataset, a distinct list of users was examined.

Tweets are collected using two different Twitter APIs: a streaming API and an API for searching for tweets regarding different events. The streaming API is used to collect datasets on given events. The search API is used to collect users’ tweets histories simultaneously [5].

- **CREDBANK and PHEME:**

A method for automating fake news detection on Twitter is developed in [6], by learning to predict accuracy assessments in these two credibility-focused Twitter datasets. CREDBANK is a crowdsourced dataset of accuracy assessments for events in Twitter and PHEME is a dataset of potential rumors in Twitter and journalistic assessments of their accuracies. This method used is applied to Twitter content sourced from BuzzFeed's fake news dataset. All three datasets are publicly available.

During each rumor selected in the PHEME dataset, journalists selected popular (e.g. highly retweeted) tweets extracted from Twitter's search API and labeled these tweets as rumor or non-rumor. This construction resulted in a set of 330 labeled rumorous source tweets across 140 stories. Of the 330 conversation trees in PHEME, 159 were labeled as true, 68 false and 103 verified [6].

CREDBANK is a large scale crowdsourced dataset which contains 60 million Tweets covering 96 days, starting from October 2015, grouped into 1,049 events with a 30-dimensional vector of truthfulness labels. Each event was rated on a 5-point Likert scale of truthfulness by 30 human annotators from Amazon Mechanical Turk [17]. They simply concatenate 30 ratings as a vector because they find it difficult to reduce it to a one-dimensional score [14].

- **Hurricane Sandy and miscellaneous events:**

The dataset comprises of tweets on hurricane Sandy event and miscellaneous events including MH370, Boston marathon, Paris attack and Russia air strike on Syria etc. It has 5,349 English tweets with images or URLs that are accessible while others are either deleted or marked as private by their owners. Among them, 3,812 are hurricane Sandy tweets and 1,537 miscellaneous events tweets [11].

- **Kaggle and reliable news sources according to Forbes:**

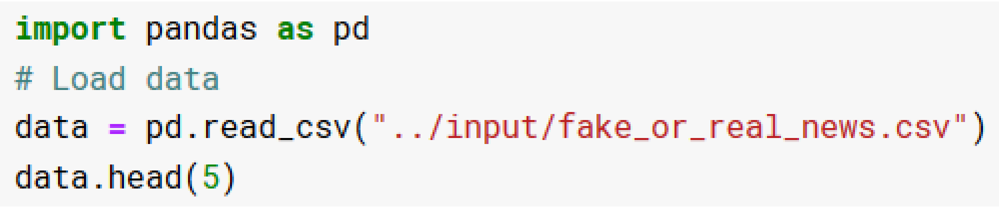
The data for the training and testing of the models were obtained from different sources. Fake articles were taken from the Kaggle dataset that provided a total of 12,999 news posts classified as "Fake". To compensate for the fact that real news make up a larger percentage of total articles, the number of real posts used for training the models was significantly higher. Political articles were downloaded from the most reliable news sources according to Forbes, which includes sites like cnn.com and nytimes.com. In total, about 28,000 real articles were used in training. In addition, some articles URLs from infamous websites that tend to post fake news were scraped.

The final dataset contained 40,000 news articles. However, the data had been saved across different formats, so python functions were created to merge it all into a common csv file with columns “text” and “type” [13].

## 5.4 Selection and description of our dataset

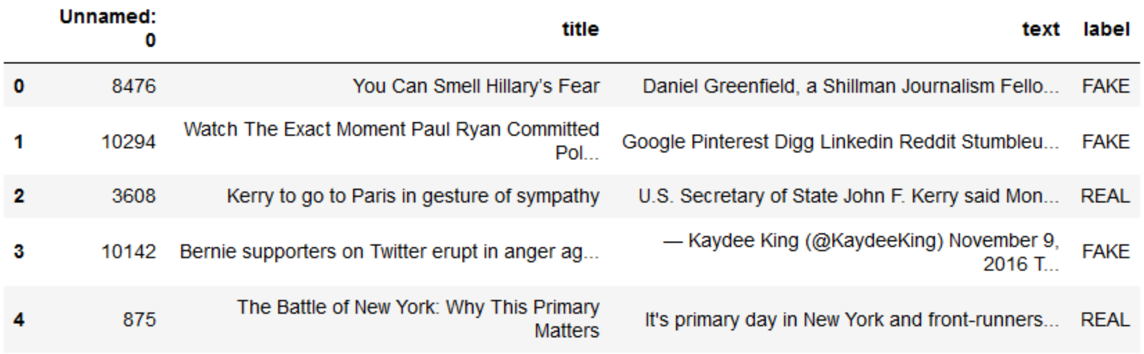
Before beginning building our own classifier, we had to select a dataset, which would contain both real and fake news articles. First, we found such a dataset in the interactive course of Datacamp “Natural Language Processing Fundamentals in Python”. This dataset is named “Fake_or_Real_news” and it includes a csv file named “fake_or_real_news”. After loading the data from the csv file into a Pandas DataFrame, using the “read_csv” function in Pandas, we found that it consists of 6,335 entries (rows) and 4 columns, namely “Unnamed: 0”, “title”, “text” and “label”. The column “label” contains the label given to each news entry, which is either “FAKE” or “REAL”. This dataset includes 3,164 entries labeled as “FAKE” and 3,171 entries labeled as “REAL”, which means that our dataset is nearly balanced as far as the number of “FAKE” and “REAL” news included is concerned. Below, Figure 5.1 presents the part of our code in the kernel we created in Kaggle, using Python as a programming language, considering the loading of our data, and Table 5.3 presents the first 5 entries of our dataset, in order to acquire an overview of it:

**Figure 5.1:** Data Loading of “Fake_or_Real_news”



```
import pandas as pd
# Load data
data = pd.read_csv("../input/fake_or_real_news.csv")
data.head(5)
```

**Table 5.6:** "Fake_or_Real_news" dataset's 5 first entries



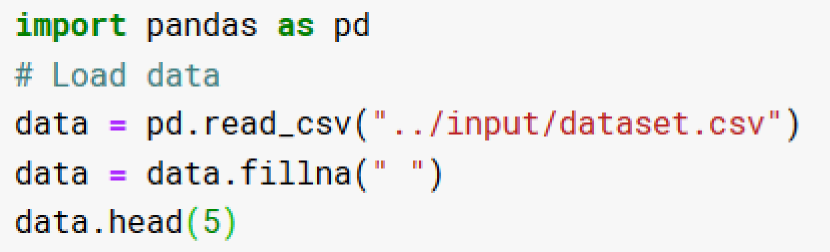
| | Unnamed: 0 | title | text | label |
|---|---|---|---|---|
| 0 | 8476 | You Can Smell Hillary's Fear | Daniel Greenfield, a Shillman Journalism Fello... | FAKE |
| 1 | 10294 | Watch The Exact Moment Paul Ryan Committed Pol... | Google Pinterest Digg Linkedin Reddit Stumbleu... | FAKE |
| 2 | 3608 | Kerry to go to Paris in gesture of sympathy | U.S. Secretary of State John F. Kerry said Mon... | REAL |
| 3 | 10142 | Bernie supporters on Twitter erupt in anger ag... | — Kaydee King (@KaydeeKing) November 9, 2016 T... | FAKE |
| 4 | 875 | The Battle of New York: Why This Primary Matters | It's primary day in New York and front-runners... | REAL |

We also looked for a dataset consisting of real and fake news in Kaggle platform. There, we found a dataset consisting of two sub-datasets. The dataset is named "Fake News" and the two sub-datasets are included in the csv files "train.csv" and "test.csv". After loading the data from the csv files into Pandas DataFrames, using the "read_csv" function in Pandas, we found that the train DataFrame consists of 20,800 entries (rows) and 5 columns, named "id", "title", "author", "text" and "label", which is either 0 for real news or 1 for fake news, whereas test DataFrame consists of 5,200 entries (rows) and 4 columns, which are the same ones with train DataFrame's columns, without "label" column. The reason why test DataFrame does not contain the "label" column is the fact that the dataset "Fake News" is used in order to predict for the test data whether they are real or fake, without knowing the true label of them. Thus, we could select only the train data of this dataset and then split them into train and test, in order to train our classifier and then, evaluate its performance. Having a look at the train dataset, we discovered that there are some missing data, which we fill using the "fillna" function. Then, we counted the number of fake and real news, and we found out that there are 10,387 real news entries and 10,413 fake ones, so this dataset is nearly balanced, too.

Finally, we decided to use the train dataset of the "Fake News" dataset as the dataset of our research, since it is nearly balanced as far as the fake and real news entries included are concerned. This is very important for the training phase of our classifier, because if the data is skewed in favor of one of the two classes, 0 and 1, the majority class can dominate the minority class and the classifier may only learn one concept instead of two distinct concepts. Also, this dataset is larger than the first dataset, which is another advantage contributing to its selection, since the larger the dataset, the more we can extract insights that we trust from that dataset.

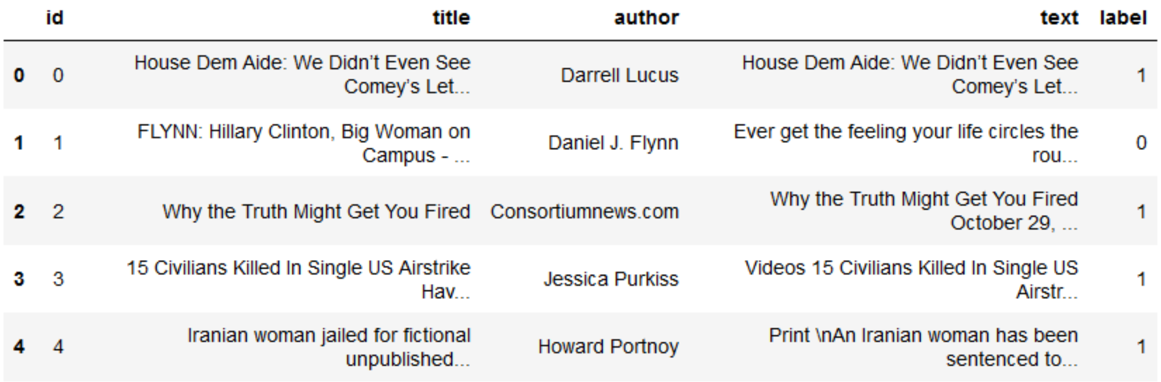
After selecting this dataset, we renamed it to “Data for Fake News Classifier”. Below, Figure 5.2 presents the part of our code in the kernel we created in Kaggle, named “Fake News Classifier”, using Python as a programming language, considering the loading of our data, and Table 5.4 presents the first 5 entries of our dataset, in order to acquire an overview of it:

**Figure 5.2:** Data Loading of “Data for Fake News Classifier”



```python
import pandas as pd
# Load data
data = pd.read_csv("../input/dataset.csv")
data = data.fillna(" ")
data.head(5)
```

**Table 5.7:** “Data for Fake News Classifier” dataset’s 5 first entries



| | id | title | author | text | label |
|---|---|---|---|---|---|
| **0** | 0 | House Dem Aide: We Didn't Even See Comey's Let... | Darrell Lucus | House Dem Aide: We Didn't Even See Comey's Let... | 1 |
| **1** | 1 | FLYNN: Hillary Clinton, Big Woman on Campus - ... | Daniel J. Flynn | Ever get the feeling your life circles the rou... | 0 |
| **2** | 2 | Why the Truth Might Get You Fired | Consortiumnews.com | Why the Truth Might Get You Fired October 29, ... | 1 |
| **3** | 3 | 15 Civilians Killed In Single US Airstrike Hav... | Jessica Purkiss | Videos 15 Civilians Killed In Single US Airstr... | 1 |
| **4** | 4 | Iranian woman jailed for fictional unpublished... | Howard Portnoy | Print \nAn Iranian woman has been sentenced to... | 1 |

# 6. Pre-processing and features extraction steps

## 6.1 Introduction

In this chapter, we are going to present and describe the pre-processing and the features extraction steps followed in each category, news articles and tweets. Then, we are going to select those steps we are going to follow as far as the data preparation is concerned, in our own kernel. The whole process will be presented and explained in this chapter.

## 6.2 Data preparation steps of news articles datasets

Below, a table, which includes all steps followed for the data preparation is presented:

**Table 6.1:** Data preparation steps of news articles datasets

| | | [2] | [3] | [7] | [8] | [9] | [10] | [15] |
|---|---|---|---|---|---|---|---|---|
| **Pre-processing steps** | **Stopwords removal** | | ■ | | | | | |
| | **Punctuation removal** | | ■ | | | | | |
| | **Lowercasing** | | ■ | | | | | |
| | **Sentences segmentation** | | ■ | | | | | |
| | **Stemming** | | ■ | | | | | |
| | **Non-letter characters removal** | | ■ | | | | | |
| | **No-text or “null” as text ignorance** | | | | ■ | | | |
| | **Named entity recognition** | | | ■ | | | | |
| **Features extraction steps** | **LIWC application** | | | | | | | ■ |
| | **PCFG use** | | | ■ | | | | |
| | **Word count** | | | | | | ■ | |
| | **Polarity calculation** | | | | | | ■ | |
| | **Loudness calculation** | | | | | | ■ | |
| | **Readability calculation** | | | | | | ■ | |
| | **Check for citation or question** | | | | | | ■ | |
| | **BOW model use** | ■ | ■ | | ■ | ■ | | |
| | **TFIDF model use** | ■ | ■ | ■ | | | | ■ |
| | **N-grams use** | | ■ | ■ | | ■ | | |

**Pre-processing steps:**

- **Stopwords removal:**

Stop words are insignificant words in a language that will create noise when used as features in text classification. These are words commonly used a lot in sentences to help connect thought or to assist in the sentence structure. Articles, prepositions and conjunctions and some pronouns are considered stop words. Common words such as a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or, that, the, these, this, too, was, what, when, where, who, will, etc, are removed [3].

- **Stemming:**

After tokenizing the data, the next step is to transform the tokens into a standard form. Stemming simply is changing the words into their original form and decreasing the number of word types or classes in the data. For example, the words “Running”, “Run” and “Runner” will be reduced to the word “run”. Stemming is used in order to make classification faster and efficient. Furthermore, Porter stemmer is used, which is the most commonly used stemming algorithm due to its accuracy [3].

- **Named entity recognition:**

If a corpus is of political nature, the model’s knowledge of the people and institutions mentioned in the article text has to be limited. Otherwise, the model faces the risk of simply learning patterns such as “Clinton corrupt” which describe the topic and viewpoint of the text, rather than the outcome of interest (is this source reliable or not). Additionally, these patterns will be highly sensitive to the particular news cycle. To address this concern, a step is introduced during tokenization to use Spacy’s named entity recognition to replace all mentions of named entities with a placeholder, e.g. <-NAME-> or <-ORG-> [7].

- **No-text or “null” as text ignorance:**

Some of the articles in a dataset may be broken – they do not contain any text at all or they contain “null” as a text. These articles are ignored [8].

- **Article’s source name, twitter handles and e-mail addresses removal:**

It is important that the articles are scrubbed of any mention of the name of the source. Because the reliable/unreliable classification is determined at the source level, this step is necessary to ensure the model does not just learn the mappings from known sources to labels. Twitter handles and e-mail addresses (which often show up in journalist biographies) are also stripped for the same reason [7].

**Features extraction steps:**

- **Normalized syntactical dependency frequency (Probabilistic Context Free Grammar or PCFG) use:**

Spacy can be used to tokenize and parse syntactical dependencies of each document. Spacy's algorithm is a transition-based, greedy, dynamic oracle using Brown clusters that is comparable in accuracy to Stanford's PCFG, but dramatically faster and more lightweight.

Each token is tagged with one of 46 possible syntactic dependency relations, such as "noun subject" or "preposition". The frequency of occurrences of each dependency tag is counted and normalized by the total number of dependencies in the document. Again, Sklearn is used to convert these frequencies into sparse matrices suitable for training models [7].

- **Polarity calculation:**

The polarity is calculated via a dictionary-based approach as implemented in the R library "gdap". The word list is used in related studies and entails 2,003 positive and 4,776 negative opinions as well as 23 negation words [10].

- **Loudness calculation:**

The loudness of the texts is calculated by dividing the number of capitalized characters and the number of empathies characters [10].

- **Readability calculation:**

The readability is estimated by calculating the Flesch-Kincaid grade level [10].

- **LIWC application:**

To characterize differences between news types, various lexical resources can be applied. Such a lexicon is Linguistic and Word Count (LIWC) [15]. The way that the LIWC program works is fairly simple. Basically, it reads a given text and counts the percentage of words that reflect different emotions, thinking styles, social concerns, and even parts of speech. Because LIWC was developed by researchers with interests in social, clinical, health, and cognitive psychology, the language categories were created to capture people’s social and psychological states [19].

- **BOW and TFIDF models use:**

One of the challenges of text categorization is learning from high dimensional data. There is a large number of terms, words and phrases in documents that lead to a high computational burden for the learning process. Furthermore, irrelevant and redundant features can hurt the accuracy and performance of the classifiers. Thus, it is best to perform feature reduction to reduce the text feature size and avoid large feature space dimension. Generally, two different features selection methods are used, Bag of Words (BOW) and Term Frequency-Inverse Document Frequency (TFIDF). These methods are described in the following:

BOW is an approach that utilizes the counts of words appearing in the documents to figure out the similarity between documents. Each document is represented by an equal length vector that contains the words counts. Next, each vector is normalized in a way that the sum of its elements will add to one. Each word count is then converted into the probability of such word existing in the documents. For example, if a word is in a certain document it will be represented as one, and if it not in the document it will be set to zero. Thus, each document is represented by groups of words.

TFIDF is a weighing metric often used in information retrieval and Natural Language Processing (NLP). It is a statistical metric used to measure how important a term is to a document in a dataset. A term important increases with the number of times a word appears in the document, however, this is counteracted by the frequency of the word in the corpus. One of the main characteristics of TFIDF is the fact that it weighs the term frequency while scaling up the rare ones [3].

- **N-grams use:**

N-gram modeling is a popular feature identification and analysis approach used in language modeling and NLP fields. N-gram is a contiguous sequence of items with length n. It could be a sequence of words, bytes, syllables or characters. The most used n-gram models in text categorization are word-based and character-based n-grams [3].

## 6.3 Data preparation steps of tweets datasets

Below, a table, which includes all steps followed for the data preparation is presented:

**Table 6.2:** Data preparation steps of tweets datasets

| | | [1] | [4] | [5] | [6] | [11] | [13] |
|---|---|---|---|---|---|---|---|
| **Pre-processing steps** | **Greeting & redundant tweets' removal** | | | 🟥 | | | |
| | **Stopwords removal** | 🟥 | | | | | 🟥 |
| | **Tweets' padding, adding 0** | | 🟥 | | | | |
| | **Non-letter characters removal** | | | | | | 🟥 |
| | **Text within speech marks removal** | | | | | | 🟥 |
| **Features extraction steps** | **User features extraction** | | | 🟥 | 🟥 | 🟥 | |
| | **Content features (text & sentiment) extraction** | | | 🟥 | 🟥 | 🟥 | |
| | **Structural features extraction** | | | | 🟥 | | |
| | **Temporal features extraction** | | | | 🟥 | | |
| | **Hybrid features extraction** | | | 🟥 | | | |
| | **User reputation measurement** | | | 🟥 | | | |
| | **BOW model use** | 🟥 | | | | | |
| | **TFIDF model use** | 🟥 | | | | | 🟥 |

**Features extraction steps:**

- **Content features extraction:**

- **Text features:** They include some characteristics related to the content of the tweet such as the length of a message, the number of replies and/or the number of re-tweets, which may reflect the importance of the tweet, as well as whereas the tweet contains #tags and “@mentions”, as well as URLs and number of static and animated emoticons.

- **Sentiment features:** Content features measure textual aspects of tweets, like polarity (the average positive or negative feelings expressed in the tweet), subjectivity (a score of whether a tweet is objective or subjective) and disagreement, as measured by the amount of tweets expressing disagreement in the conversation. They also include the calculating of the number of positive and negative words, based on a predefined sentiment words list [5], [6].

The thirteen content features extracted by [11] are the following:

- *tweetLength* (number of characters in the tweet)
- *wordCount* (number of word)
- *noOfQuestionMark* (number of question marks)
- *noOfExclamationMark* (number of exclamation marks)
- *containsQuestionMark* (whether question mark is presented)
- *containsExclamationMark* (whether exclamation mark is presented)
- *noOfUpperCaseLetter* (number of upper case letters)
- *noOfhasgTags* (number of hashtags)
- *noOfUrls* (number of URLs)
- *noOfRetweets* (number of retweets)
- *isUrlCredible* (whether the URLs in the tweet are valid)
- *isImageCredible* (whether the images in the tweet are credible)
- *sentimentScore* (tweet's sentiment score)

Tweet's sentimentScore:

Sentiment analysis is the process of computationally determining whether a piece of writing is positive, negative or neutral by assigning a sentiment score between -1.0 (most negative) to 1.0 (most positive). It is analyzed by using TextBlob and Natural Language Toolkit (NLTK) corpora. NLTK is a leading platform to work with human language data. NLTK provides library to compute sentiment score but is very primitive and hard to use. TextBlob is a wrapper on top of NLTK library.

To compute sentimentScore feature values, firstly the tweet is cleaned to remove all hyperlinks, special characters etc., using simple Regex. Secondly, a TextBlob object from the tweet text is created using the TextBlob library as follows:

- The text in the tweet is tokenized into tokens (words).
- The stop words are removed from the list of tokens.
- Significant features/tokens like adjectives, adverbs etc., are tagged and selected to pass into sentiment classifier (Naïve Bayes Classifier (NB) in Text Blob) to be classified as positive, negative or neutral by assigning a polarity between -1.0 to 1.0. A tweet is considered positive in nature if its polarity is greater than 0, negative if it is less than 0 and neutral if it 0.

- **User features:**

They capture properties of tweet authors, such as age, gender, education, political orientation, Twitter verified status, account age, any user preferences, the number of followers, the number of friends and the number of re-tweeted tweets, as well as the replies of user's tweets [5], [6].

The seven user features extracted by [11] are the following:

- *noOfFriends* refers to the number of friends for the Twitter user.
- *noOfFollowers* is user's number of followers.
- *friendFollowRatio* is the ratio between user's *noOfFriends* and *noOfFollowers*.
- *noOfTimesListed* is the number of times the user has listed.
- *isUserHasURL* is whether the user has URL.
- *isVerifiedUser* shows whether the user is a verified user.
- *noOfTweets* lists number of Tweets the user has posted.

- **Hybrid features:**

Hybrid-level feature extraction in the credibility assessment process is considered for two reasons. First, at the tweet level, the 140-character length limit of Twitter messages makes them to a certain degree inappropriate for analysis with high-impact topic models. From another perspective, individual tweets do not provide sufficient information about the combination of precise latent topics within them. Second, users can easily obtain thousands of followers in a minute from so-called Twitter follower markets [5].

- **Structural features:**

Structural features capture Twitter-specific properties of the tweet stream, including tweet volume and activity distributions, e.g. number of tweets, average tweet length, thread lifetime, proportions of re-tweets, mentions or media shares [6].

- **Temporal features:**

  They capture trends in the content features over time, e.g. the slopes of the number of tweets or average author age over time [6].

- Removal of short tweets that do not give useful information about the author [1].
- Normalizing of hyper links, re-tweeted tweets and hashtags with a new text URL, RT and TAG respectively [1].
- **User reputation measurement:**

  Measuring user reputation is an important aspect of the problem to be solved because the phenomenon of inspiration is widespread, especially on social networks. To measure user expertise and reputation, some different measures are used, which are considered to have a huge impact on Twitter. This can be accomplished by measuring reputation through how popular a user is and how sentimental he/she is:

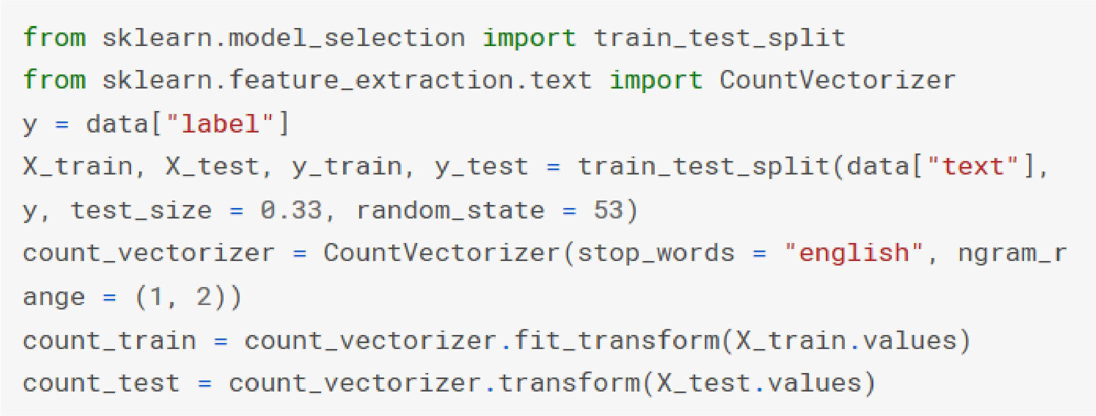
  - **User Sentiment History:** The sentimentality of a user influences his or her judgments of tweet credibility with respect to an event or topic, especially when the user is inclined favorably or unfavorably toward some sects or groups. Some users have reasons for disseminating information that may be considered misleading and can contribute to chaos, as in the case of the Arab Spring in 2011. Sentiment defines the factors that affect social relationships, psychological states of users and their orientation. Sentiment also involves an analysis of why a user trusts a trustee or not. In a study on calculation of the number of positive and negative words in a message, based on a predefined "sentiment words" list, researchers found that the least credible messages are associated with negative social events and contain strong negative sentiment words and opinions.
  - **User Popularity Score:** It is a measure of user popularity and it can be explained in the form of an algorithm that describes criteria suitable for ranking each user on the network regarding his reputation. Re-tweets, favorites and mentions are considered to be the best indicators from a quantitative perspective. This implies that a tweet that has been retweeted many times is considered to be attractive to the reader. Nevertheless, the

most critical indicators are qualitative. One good example is the relationship between the reader and the tweeter of the tweet [5].

## 6.4 Selection and description of our data preparation steps

After considering the various methodologies described above, we selected those that we would use in our kernel. We decided to use as pre-processing steps the no-text fields' ignorance, mentioned in the previous chapter, as well as the stopwords removal. Also, we selected the BOW model with n-grams' size ranging from 1 to 2, for features extraction. The reason why we decided to follow the above methodology is the fact that these steps are mainly followed for the building of those classifiers of the literature sources using the algorithms we selected in our kernel, and which we are going to present, describing their evaluation metrics, in the next chapter.

**Figure 6.1:** Pre-processing and features extraction code



```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
y = data["label"]
X_train, X_test, y_train, y_test = train_test_split(data["text"],
y, test_size = 0.33, random_state = 53)
count_vectorizer = CountVectorizer(stop_words = "english", ngram_r
ange = (1, 2))
count_train = count_vectorizer.fit_transform(X_train.values)
count_test = count_vectorizer.transform(X_test.values)
```

First, we import the necessary modules:

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
```

CountVectorizer is used to convert a collection of text documents to a matrix of token counts.

Then, we create a series named y, to store the labels of the news entries, which is also the outcome our model will learn:

```
y = data[“label”]
```

Next step is the creation of the training and test sets. Taking into consideration the Datacamp course “Natural Language Processing Fundamentals in Python” and the process followed in the respective section “Building a “fake news classifier”, the function train_test_split will take the 33% of rows to mark as test data, removing them from the training data. We have also stated random_state, so that we can have a repeatable result.

```
X_train, X_test, y_train, y_test = train_test_split(data[“text”], y, test_size = 0.33, random_state = 53)
```

Then, we initialize a CountVectorizer object, named count_vectorizer, providing as parameters the stop_words and the ngram_range. In this way, out text is turned into a bag-of-words vector. Stop_words parameter is given the value “english” and ngram_range parameter is given as the lower boundary the value 1 and as the upper boundary the value 2. For ngram_range, all values of n such that min_n ≤ n ≤ max_n will be used [21].

```
count_vectorizer = CountVectorizer(stop_words = “english”, ngram_range = (1, 2))
```

Finally, we transform the training and the test data using only the “text” column values.

```
count_train = count_vectorizer.fit_transform(X_train.values)
count_test = count_vectorizer.transform(X_test.values)
```

Fit_transform creates the bag-of-words dictionary and vectors for each document using the training data. After calling fit_transform on the training data, we call transform on the test data to create a bag-of-words vector using the same dictionary. The training and test vectors need to use a consistent set of words so the trained model can understand the test input [21].

# 7. Algorithms and evaluation metrics selection

## 7.1 Introduction

In this chapter, we are going to present the algorithms and the evaluation metrics applied on news articles datasets, as well as the algorithms and the evaluation metrics applied on tweets datasets. Those of the second category are going to be briefly presented, because we are mainly interested in those of the first category, since our dataset contains exclusively news articles. Then, we are going to select the algorithms and the evaluation metrics we will use in our kernel, taking into consideration two criteria: 1) which algorithms, as well as which evaluation metrics are more frequently used, and 2) which algorithms have the best performance. The whole process will be presented and explained. Below, a table is given which includes the most frequently used evaluation metrics in the binary classification process:

**Table 7.1:** Evaluation Metrics

| | |
|---|---|
| **Accuracy** | $\frac{\mathrm{TP+TN}}{\mathrm{P+N}} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$ |
| **Precision or Positive Predictive Value (PPV)** | $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$ |
| **Recall or Sensitivity or TPR** | $\frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ |
| **Specificity or TNR** | $\frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}$ |
| **$F_1$-score** | $\frac{\mathrm{2TP}}{\mathrm{2TP+FP+FN}}$ |
| **FNR or miss rate** | $\frac{\mathrm{FN}}{\mathrm{P}} = \frac{\mathrm{FN}}{\mathrm{TP+FN}}$ |
| **FPR or fall-out** | $\frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$ |

## 7.2 News articles algorithms and evaluation metrics selection

Below, a table, which includes the algorithms used, is presented:

**Table 7.2:** News articles algorithms

| | [2] | [3] | [7] | [8] | [9] | [10] | [15] | [16] |
|---|---|---|---|---|---|---|---|---|
| **KNN** | | ■ | | | | | | |
| **SVM** | | ■ | ■ | | | ■ | | |
| **LSVM** | | ■ | | | | | | |
| **DT** | | ■ | ■ | | | ■ | | |
| **SGD** | | ■ | ■ | | | | | |
| **LR** | | ■ | | | | ■ | ■ | |
| **NB** | ■ | | | ■ | ■ | | ■ | |
| **RF** | | | ■ | | | ■ | | |
| **XGB** | | | ■ | | | ■ | | |
| **CNN** | | | | | | | | ■ |
| **LSTM** | | | | | | | ■ | ■ |

Below, a table, which includes the evaluation metrics used, is presented:

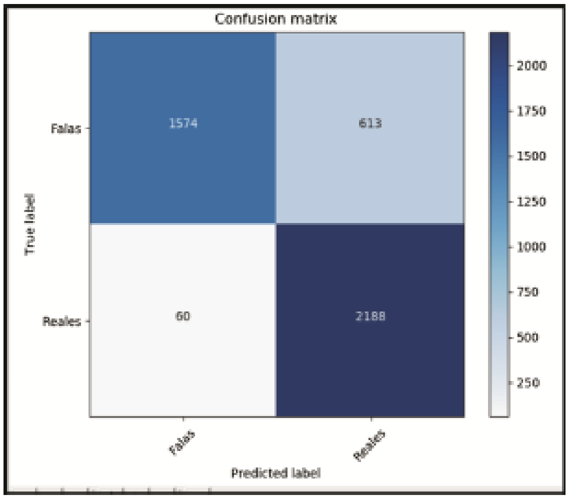
**Table 7.3:** News articles evaluation metrics

| | [2] | [3] | [7] | [8] | [9] | [10] | [15] | [16] |
|---|---|---|---|---|---|---|---|---|
| **Accuracy** | | ■ | ■ | ■ | | ■ | ■ | ■ |
| **Precision** | | ■ | ■ | ■ | | ■ | | ■ |
| **Recall** | | ■ | ■ | ■ | | ■ | | ■ |
| **$F_1$-score** | | | | | | ■ | | ■ |
| **Error rate** | | | | | | ■ | | |
| **Specificity** | | | | | | ■ | | |
| **AUC** | | | ■ | | ■ | | | |
| **Confusion Matrix** | ■ | | | | | | | |

- **Naïve Bayes (NB):**

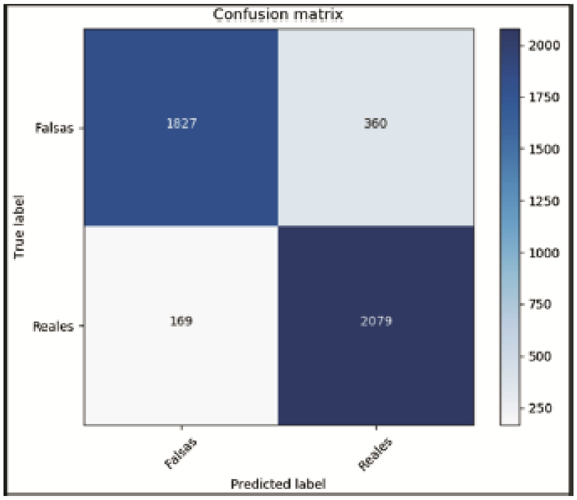
In the first research [2] using NB algorithm, where both Bag of Words (BOW) and Term Frequency-Inverse Document Frequency (TFIDF) models are used, it is proved that NB algorithm is more effective when using the BOW model, since it successfully classified 89.3% of the news. More specifically, it correctly classified as false 1827 news and as true 2079 news. In the following figures, the confusion matrices with the results of both classifications are presented:

**Figure 7.1 :** Confusion matrix : TFIDF



[2]

**Figure 7.2 :** Confusion matrix : BOW



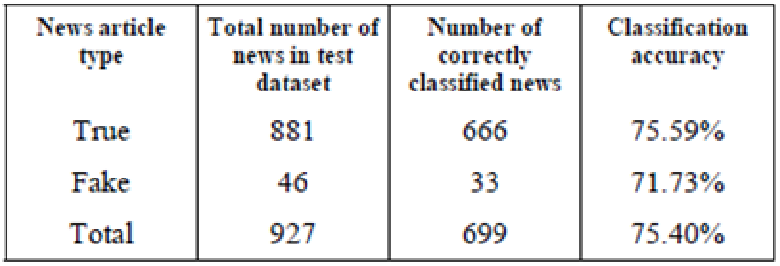
[2]

In the second research [8] using NB algorithm, the classification accuracy was 75.4%. More specifically, the classification accuracy for true news articles and false news articles was roughly the same, although classification accuracy for fake news was slightly worse, which may be caused by the skewness of the dataset, since only 4.9% of it is fake news.

The precision of the given classifier equals to 0.71; recall on the other hand equals to 0.13. Such a low value of the recall once again is caused by the skewness of the data in the test dataset [8].

Below, Table 7.4 presents the received results of the NB algorithm in this research:

**Table 7.4:** Received results of the NB algorithm



| News article type | Total number of news in test dataset | Number of correctly classified news | Classification accuracy |
|---|---|---|---|
| True | 881 | 666 | 75.59% |
| Fake | 46 | 33 | 71.73% |
| Total | 927 | 699 | 75.40% |

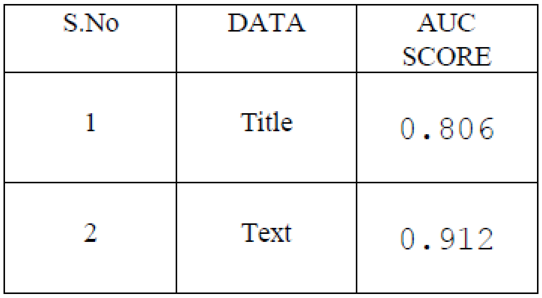
[8]

In the next research [9] using NB algorithm, it can be seen from the results obtained from its implementation that the Area Under Curve (AUC) scores increase, when the amount of data existing under a particular tag increases, as seen in the case of Title and Text, as Title consists a smaller version of news articles and Text is a descriptive version of the same.

Now on comparing the results of different methodologies, where the concept of n-grams is introduced in the second model, we can see that AUC scores improve with n-grams as visible, because the number of vectors in the second model increase, hence providing better judgment capacity to the second model.

In the following tables, AUC scores with and without n-grams are presented:

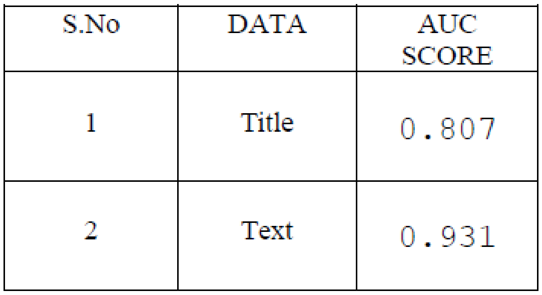
**Table 7.5:** AUC scores without n-grams



| S.No | DATA | AUC SCORE |
|---|---|---|
| 1 | Title | 0.806 |
| 2 | Text | 0.912 |

[10]

**Table 7.6:** AUC scores with n-grams



| S.No | DATA | AUC SCORE |
|---|---|---|
| 1 | Title | 0.807 |
| 2 | Text | 0.931 |

[10]

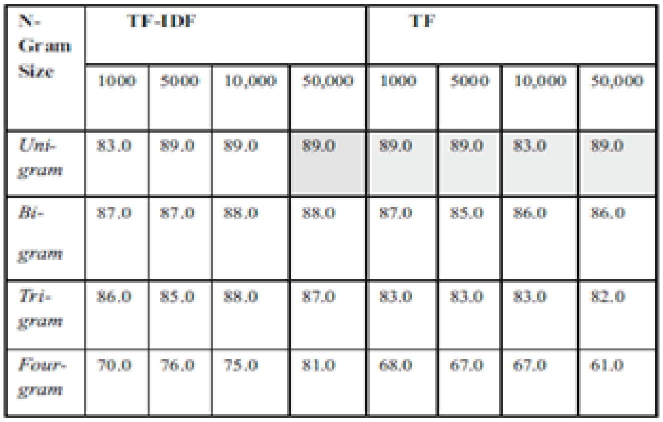
In research [15], NB achieves an accuracy of 56% when using the LIWC features.

- **<u>Logistic Regression (LR):</u>**

In research [3], LR was used, studying the impact of the size (n) of n-grams on its performance. First, unigram (n=1) was used, then bigram (n=2), then steadily n increased by one, until reaching the value n=4. Furthermore, each n value was tested combined with a different number of features.

Below, Table 7.7 presents the results of the classification algorithm used, for the two different features extraction methods BOW and TFIDF, the different size of n-grams ranging from 1 to 4, and the different number of top features selected, ranging from 1,000 to 50,000:

**Table 7.7:** LR Accuracy Results



| N-Gram Size | TF-IDF | | | | TF | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 5000 | 10,000 | 50,000 | 1000 | 5000 | 10,000 | 50,000 |
| *Uni-gram* | 83.0 | 89.0 | 89.0 | 89.0 | 89.0 | 89.0 | 83.0 | 89.0 |
| *Bi-gram* | 87.0 | 87.0 | 88.0 | 88.0 | 87.0 | 85.0 | 86.0 | 86.0 |
| *Tri-gram* | 86.0 | 85.0 | 88.0 | 87.0 | 83.0 | 83.0 | 83.0 | 82.0 |
| *Four-gram* | 70.0 | 76.0 | 75.0 | 81.0 | 68.0 | 67.0 | 67.0 | 61.0 |

[3]

We observe from the above table that the highest accuracy of 89% is achieved using uni-gram, with either TF or TFIDF model. Also, the accuracy of the LR algorithm decreases as the size of n-grams increases.

Next research using LR algorithm [10], achieved an accuracy of 76.74%, an error rate of 23.26%, specificity of 71.74%, sensitivity of 81.74%, precision of 74.69% and $F_1$-score of 77.82%.

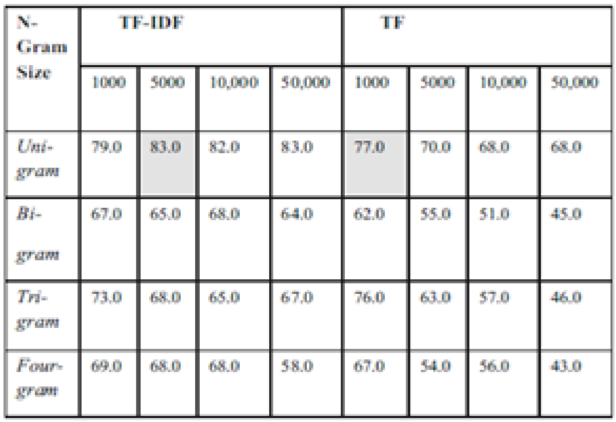
Finally, research [15], when using the LR algorithm with the LIWC features, achieved an accuracy of 55%.

- **<u>K-Nearest Neighbor (KNN):</u>**

In research [3], KNN was used, studying the impact of the size (n) of n-grams on its performance. First, unigram (n=1) was used, then bigram (n=2), then steadily n increased by one, until reaching the value n=4. Furthermore, each n value was tested combined with a different number of features.

Below, Table 7.8 presents the results of the classification algorithm used, for the two different features extraction methods BOW and TFIDF, the different size of n-grams ranging from 1 to 4, and the different number of top features selected, ranging from 1,000 to 50,000:

**Table 7.8:** KNN Accuracy Results



| N-Gram Size | TF-IDF | | | | TF | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 5000 | 10,000 | 50,000 | 1000 | 5000 | 10,000 | 50,000 |
| *Uni-gram* | 79.0 | 83.0 | 82.0 | 83.0 | 77.0 | 70.0 | 68.0 | 68.0 |
| *Bi-gram* | 67.0 | 65.0 | 68.0 | 64.0 | 62.0 | 55.0 | 51.0 | 45.0 |
| *Tri-gram* | 73.0 | 68.0 | 65.0 | 67.0 | 76.0 | 63.0 | 57.0 | 46.0 |
| *Four-gram* | 69.0 | 68.0 | 68.0 | 58.0 | 67.0 | 54.0 | 56.0 | 43.0 |

[3]

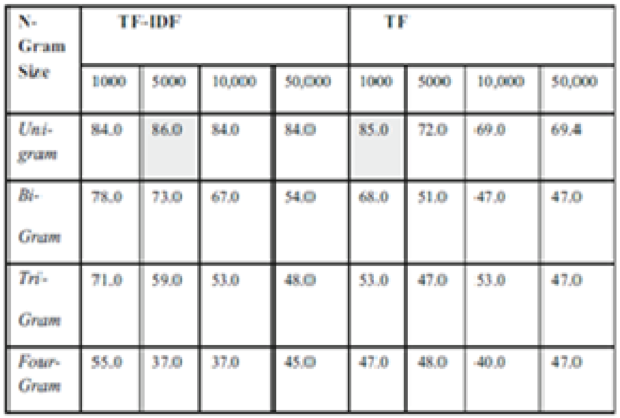
We observe from the above table that the highest accuracies of 83% and 77% are achieved using uni-gram with TFIDF and TF models respectively. Also, the accuracy of the KNN classifier decreases as the size of n-grams increases.

- **Support Vectors Machine (SVM):**

In research [3], SVM was used, with n-grams ranging from 1 to 4 as described before, testing each n value combined with a different number of features.

Below, Table 7.9 presents the results of the classification algorithm used, for the two different features extraction methods BOW and TFIDF, the different size of n-grams ranging from 1 to 4, and the different number of top features selected, ranging from 1,000 to 50,000:

**Table 7.9:** SVM Accuracy Results



| N-Gram Size | TF-IDF | | | | TF | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 5000 | 10,000 | 50,000 | 1000 | 5000 | 10,000 | 50,000 |
| *Uni-gram* | 84.0 | 86.0 | 84.0 | 84.0 | 85.0 | 72.0 | 69.0 | 69.4 |
| *Bi-Gram* | 78.0 | 73.0 | 67.0 | 54.0 | 68.0 | 51.0 | 47.0 | 47.0 |
| *Tri-Gram* | 71.0 | 59.0 | 53.0 | 48.0 | 53.0 | 47.0 | 53.0 | 47.0 |
| *Four-Gram* | 55.0 | 37.0 | 37.0 | 45.0 | 47.0 | 48.0 | 40.0 | 47.0 |

[3]

We observe from the above table that the highest accuracies of 86% and 85% are achieved using uni-gram with TFIDF and TF models respectively. Also, the accuracy of the SVM algorithm decreases again as the size of n-grams increases.

Next research [7], when using the SVM algorithm, along with TFIDF bi-gram features, achieves an accuracy of 76.2%, a precision of 81.3%, a recall of 48.1% and an AUC score of 85.6%. All the above metrics are higher when using only TFIDF bi-gram features, removing PCFG features, indicating that they add little predictive value to the models.

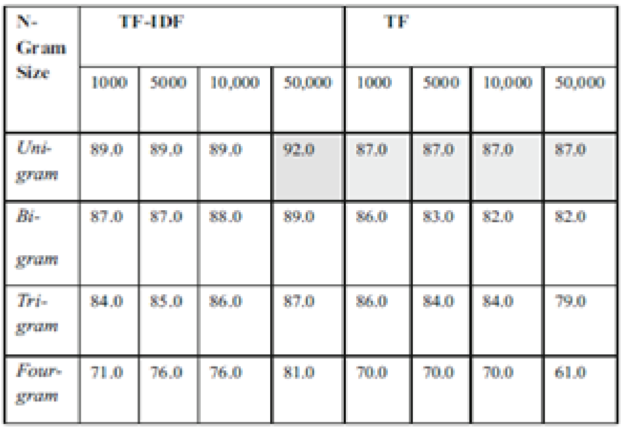
Finally, research [10], when using the SVM algorithm, achieves its highest accuracy of 80.87%. Also, an error rate of 19.13%, specificity of 73.48%, sensitivity of 88.26%, precision of 77.12% and $F_1$-score of 82.18% are achieved.

- **Linear Support Vectors Machine (LSVM):**

In research [3], LSVM was used, with n-grams ranging from 1 to 4 as described before, testing each n value combined with a different number of features.

Below, Table 7.10 presents the results of the classification algorithm used, for the two different features extraction methods BOW and TFIDF, the different size of n-grams ranging from 1 to 4, and the different number of top features selected, ranging from 1,000 to 50,000:

**Table 7.10:** LSVM Accuracy Results



| N-Gram Size | TF-IDF | | | | TF | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 5000 | 10,000 | 50,000 | 1000 | 5000 | 10,000 | 50,000 |
| *Uni-gram* | 89.0 | 89.0 | 89.0 | 92.0 | 87.0 | 87.0 | 87.0 | 87.0 |
| *Bi-gram* | 87.0 | 87.0 | 88.0 | 89.0 | 86.0 | 83.0 | 82.0 | 82.0 |
| *Tri-gram* | 84.0 | 85.0 | 86.0 | 87.0 | 86.0 | 84.0 | 84.0 | 79.0 |
| *Four-gram* | 71.0 | 76.0 | 76.0 | 81.0 | 70.0 | 70.0 | 70.0 | 61.0 |

[3]

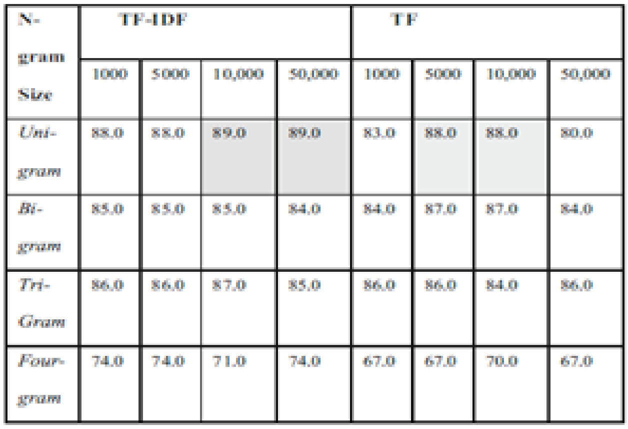
We observe from the above table that the highest accuracies of 92% and 87% are achieved using uni-gram with TFIDF and TF models respectively. Also, the accuracy of the LSVM algorithm decreases again as the size of n-grams increases.

- **Decision Tree (DT):**

In research [3], DT was used, with n-grams ranging from 1 to 4 as described before, testing each n value combined with a different number of features.

Below, Table 7.11 presents the results of the classification algorithm used, for the two different features extraction methods BOW and TFIDF, the different size of n-grams ranging from 1 to 4, and the different number of top features selected, ranging from 1,000 to 50,000:

**Table 7.11:** DT Accuracy Results



| N-gram Size | TF-IDF | | | | TF | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 5000 | 10,000 | 50,000 | 1000 | 5000 | 10,000 | 50,000 |
| *Uni-gram* | 88.0 | 88.0 | 89.0 | 89.0 | 83.0 | 88.0 | 88.0 | 80.0 |
| *Bi-gram* | 85.0 | 85.0 | 85.0 | 84.0 | 84.0 | 87.0 | 87.0 | 84.0 |
| *Tri-Gram* | 86.0 | 86.0 | 87.0 | 85.0 | 86.0 | 86.0 | 84.0 | 86.0 |
| *Four-gram* | 74.0 | 74.0 | 71.0 | 74.0 | 67.0 | 67.0 | 70.0 | 67.0 |

[3]

We observe from the above table that the highest accuracies of 89% and 88% are achieved using uni-gram with TFIDF and TF models respectively. Also, the accuracy of the LSVM classifier decreases again as the size of n-grams increases.

In the next research [7] using Bounded DT algorithm, the highest values of AUC, precision, recall and accuracy metrics are achieved by combining Probabilistic Context Free Grammar (PCFG) and TFIDF bi-gram features. More specifically, they are equal to 65.9%, 66.9%, 37.9% and 67.6% respectively.

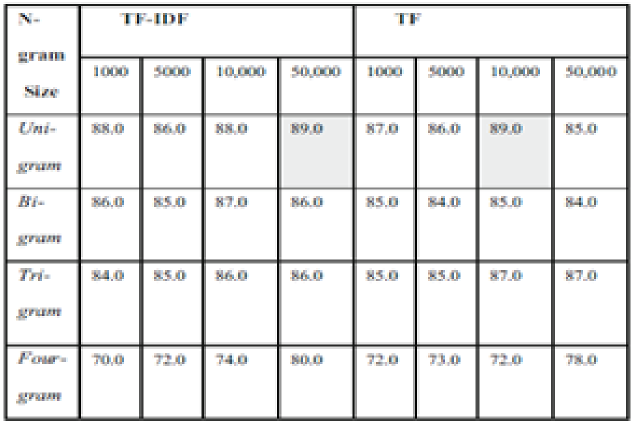
Finally, research [10], when using the DT algorithm, achieves an accuracy of 78.26%, an error rate of 21.74%, specificity of 72.17%, sensitivity of 84.35%, precision of 75.69% and $F_1$-score of 79.57%.

- **<u>Stochastic Gradient Descent (SGD):</u>**

In research [3], SGD was used, with n-grams ranging from 1 to 4 as described before, testing each n value combined with a different number of features.

Below, Table 7.12 presents the results of the classification algorithm used, for the two different features extraction methods BOW and TFIDF, the different size of n-grams ranging from 1 to 4, and the different number of top features selected, ranging from 1,000 to 50,000:

**Table 7.12:** SGD Accuracy Results



| N-gram Size | TF-IDF | | | | TF | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 5000 | 10,000 | 50,000 | 1000 | 5000 | 10,000 | 50,000 |
| *Uni-gram* | 88.0 | 86.0 | 88.0 | 89.0 | 87.0 | 86.0 | 89.0 | 85.0 |
| *Bi-gram* | 86.0 | 85.0 | 87.0 | 86.0 | 85.0 | 84.0 | 85.0 | 84.0 |
| *Tri-gram* | 84.0 | 85.0 | 86.0 | 86.0 | 85.0 | 85.0 | 87.0 | 87.0 |
| *Four-gram* | 70.0 | 72.0 | 74.0 | 80.0 | 72.0 | 73.0 | 72.0 | 78.0 |

[3]

We observe from the above table that the highest accuracy of 89% is achieved using uni-gram with either TFIDF or TF model. Also, the accuracy of the SGD algorithm decreases again as the size of n-grams increases.

Next research [7] using the SGD algorithm, achieves its highest AUC score, precision and accuracy metrics when using only TFIDF bi-gram features. More specifically, they are equal to 88.3%, 88.8% and 77.2% respectively. Only the recall which is equal to 45.3% is much lower than the value of 71.7% achieved with the combination of TFIDF bi-gram features with PCFG.

- **Random Forest (RF):**

In research [7], as far as RF algorithm is concerned, the highest evaluation metrics of AUC, precision, recall and accuracy are overall achieved when using only TFIDF bi-gram features, and they are equal to 78.8%, 82.9%, 25.3% and 67.6%.

In research [10], evaluation metrics of accuracy, error rate, specificity, sensitivity, precision and $F_1$-score used, are equal to 80.87%, 19.13%, 75.22%, 86.52%, 77.97% and 81.93% respectively.

- **Extreme Gradient Boosting (XGB):**

In research [7], as far as XGB algorithm is concerned, the highest evaluation metrics of AUC, precision, recall and accuracy are overall achieved when using only TFIDF bi-gram features, and they are equal to 79.4%, 41%, 22.3% and 68.7%.

In research [10], evaluation metrics of accuracy, error rate, specificity, sensitivity, precision and $F_1$-score used, are equal to 78.70%, 21.30%, 73.91%, 83.48%, 76.51% and 79.64% respectively.

- **Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM):**

In research [16], the highest accuracy of 44.87% is achieved when combining the CNN and the Bi-directional LSTM models, while that of Bi-LSTM model is 42.65% and that of CNN model is 42.89%. The evaluation also shows that on the precision measure the combined model performs best with an average precision of 55%, while that of Bi-LSTM model is 53% and that of CNN model is 48%. The combined model even performs better with respect to recall and $F_1$-score measures. The combined model yields the average recall of 45% and average $F_1$-score of 43%, while that of Bi-LSTM model is 43% and 41% respectively and of the CNN model is 43% and 42% respectively.

In research [15], LSTM performs better when only using text as input, without the Linguistic and Word Count (LIWC) features, since an accuracy of 56% is achieved in this case, whereas an accuracy of 52% is achieved in achieved in the other case. Thus, the LIWC features do not improve the neural model much, indicating that some of this lexical information is perhaps redundant to what the model was already learning from the text.

## 7.3 Tweets algorithms and evaluation metrics selection

Below, a table, which includes the algorithms used, is presented:

**Table 7.13:** Tweets algorithms

| | [1] | [4] | [5] | [6] | [11] | [13] |
|---|---|---|---|---|---|---|
| **SVM** | | | | | 🟥 | |
| **LSVM** | | | | | | 🟥 |
| **DT** | | | 🟥 | 🟥 | 🟥 | |
| **LR** | 🟥 | | | | | 🟥 |
| **NB** | 🟥 | | 🟥 | | | 🟥 |
| **FR_NB** | | | 🟥 | | | |
| **CNN** | | 🟥 | | | | |
| **LSTM** | | 🟥 | | | | |
| **RF** | | | 🟥 | 🟥 | | 🟥 |

Below, a table, which includes the evaluation metrics used, is presented:

**Table 7.14:** Tweets evaluation metrics

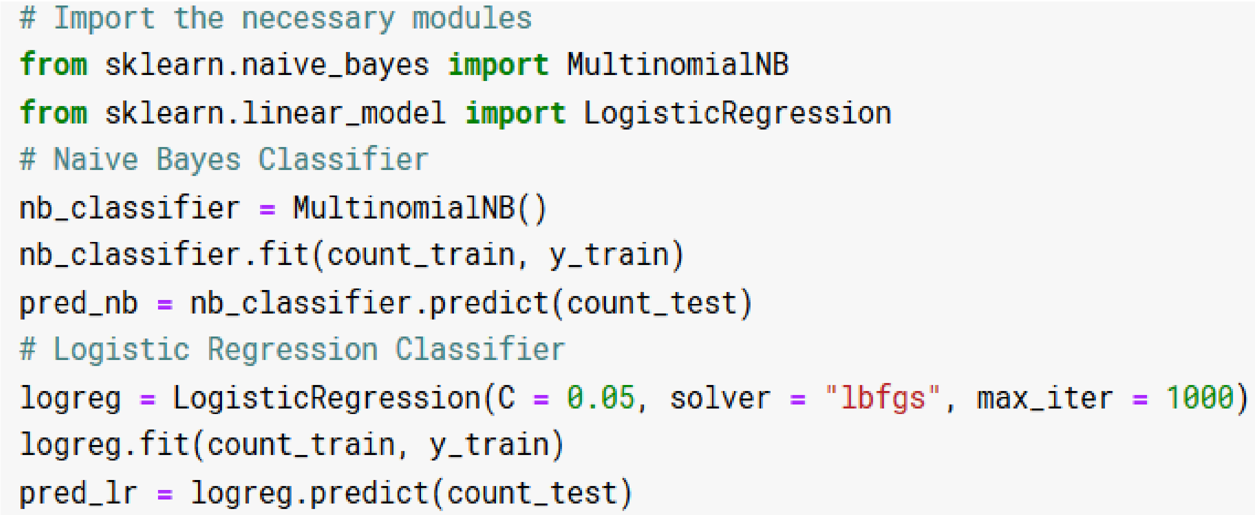
| | [1] | [4] | [5] | [6] | [11] | [13] |
|---|---|---|---|---|---|---|
| **Accuracy** | 🟥 | 🟥 | 🟥 | 🟥 | | 🟥 |
| **Precision** | | 🟥 | 🟥 | | 🟥 | |
| **Recall** | | 🟥 | 🟥 | | 🟥 | |
| **$F_1$-score** | | 🟥 | 🟥 | | 🟥 | |
| **Error rate** | | | 🟥 | | | |
| **ROC** | | | 🟥 | 🟥 | | |
| **AUC** | | | | 🟥 | | |
| **Confusion Matrix** | 🟥 | | | | | 🟥 |
| **Kappa statistics** | | | 🟥 | | | |
| **FP rate & TP rate** | | | 🟥 | | 🟥 | |

## 7.4 Selection and description of our algorithms and evaluation metrics

After considering the algorithms used for news articles classification along with their evaluation metrics, we decided to use NB and LR algorithms, since not only they are the most frequently used, but they also perform well. We also tried using LSVM because it performs well, too, and even better than the other two algorithms, but it takes very much time to run, so we decided not to include it in our research.

Below, Figure 7.1 shows the part of our code concerning the use of the two aforementioned algorithms:

**Figure 7.3:** Use of NB and LR algorithms



```
# Import the necessary modules
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
# Naive Bayes Classifier
nb_classifier = MultinomialNB()
nb_classifier.fit(count_train, y_train)
pred_nb = nb_classifier.predict(count_test)
# Logistic Regression Classifier
logreg = LogisticRegression(C = 0.05, solver = "lbfgs", max_iter = 1000)
logreg.fit(count_train, y_train)
pred_lr = logreg.predict(count_test)
```

First, we import the necessary modules:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
```

Then, for NB classifier, after initializing our class, we call fit method with the training data, count_train and y_train. After fitting our NB classifier, we call predict method, in order to perform classification on the array of test vectors. More specifically, predict will use the trained model to predict the label based on the test data vectors. The predicted labels are saved in the variable pred_nb, in order to test the accuracy in the next step of the performance evaluation [20].

```
nb_classifier = MultinomialNB()
nb_classifier.fit(count_train, y_train)
pred_nb = nb_classifier.predict(count_test)
```

Then, for LR classifier, first step we initialize our class using the following parameters:

```
logreg = LogisticRegression(C = 0.05, solver = “lbfgs”, max_iter = 1000)
```

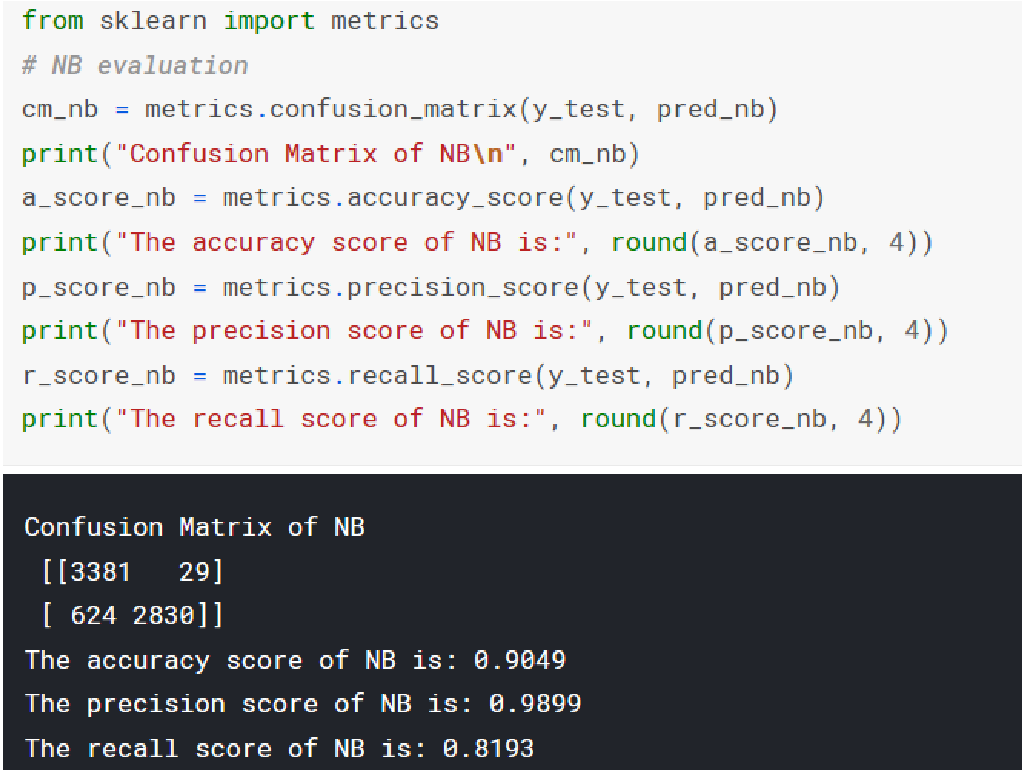
- **solver = “lbfgs”:** This solver is used by default for its robustness.
- **C = 0.05:** We selected a small value of C parameter, so that we would achieve a stronger regularization.
- **max_iter = 1000:** Max_iter is the maximum number of iterations taken for the solver to converge. After running our code with several values of max_iter, we selected the value 1000 (default value = 100), since in this case it did not appear a warning message.

Then, we call fit method with the training data, count_train and y_train. After fitting our LR classifier, we call predict method, in order to perform classification on the array of test vectors. More specifically, predict will use the trained model to predict the label based on the test data vectors. The predicted labels are saved in the variable pred_lr, in order to test the accuracy in the next step of the performance evaluation.

```
logreg.fit(count_train, y_train)
pred_lr = logreg.predict(count_test)
```

As a first step of evaluating our classifiers, we calculated the evaluation metrics of accuracy, precision and recall, as well as the confusion matrix for each of them. The corresponding parts of the code, along with their results are presented below:

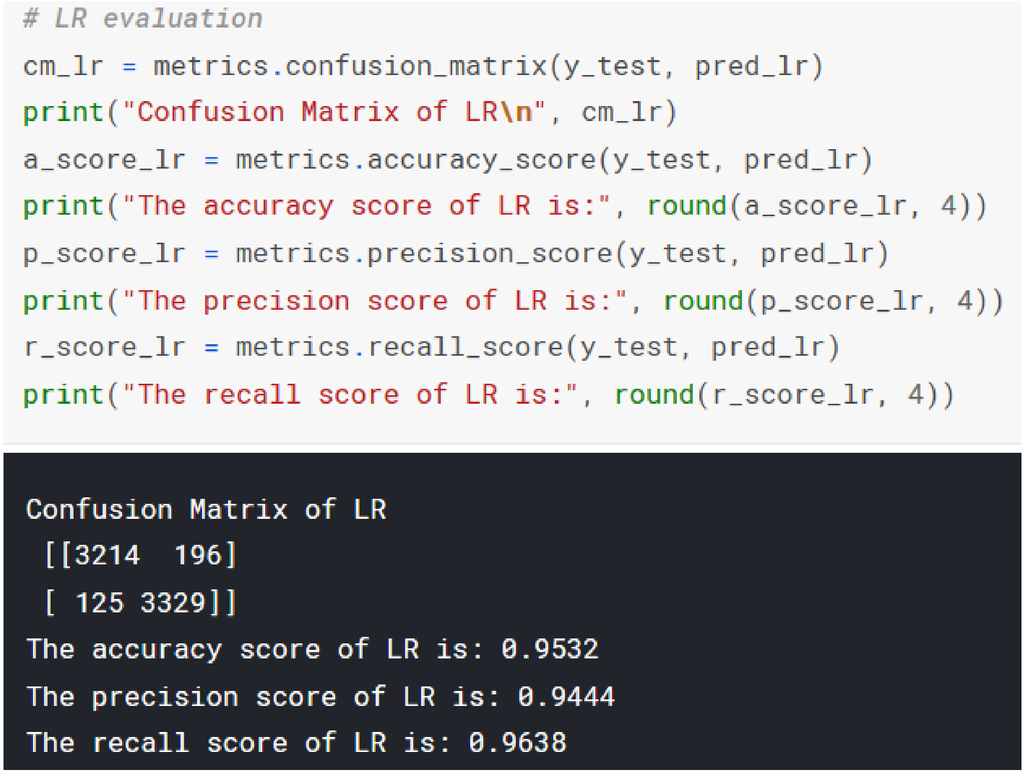
**Figure 7.4: Evaluation metrics of NB algorithm**



```
from sklearn import metrics
# NB evaluation
cm_nb = metrics.confusion_matrix(y_test, pred_nb)
print("Confusion Matrix of NB\n", cm_nb)
a_score_nb = metrics.accuracy_score(y_test, pred_nb)
print("The accuracy score of NB is:", round(a_score_nb, 4))
p_score_nb = metrics.precision_score(y_test, pred_nb)
print("The precision score of NB is:", round(p_score_nb, 4))
r_score_nb = metrics.recall_score(y_test, pred_nb)
print("The recall score of NB is:", round(r_score_nb, 4))
```

```
Confusion Matrix of NB
 [[3381   29]
 [ 624 2830]]
The accuracy score of NB is: 0.9049
The precision score of NB is: 0.9899
The recall score of NB is: 0.8193
```

**Figure 7.5: Evaluation metrics of LR algorithm**



```
# LR evaluation
cm_lr = metrics.confusion_matrix(y_test, pred_lr)
print("Confusion Matrix of LR\n", cm_lr)
a_score_lr = metrics.accuracy_score(y_test, pred_lr)
print("The accuracy score of LR is:", round(a_score_lr, 4))
p_score_lr = metrics.precision_score(y_test, pred_lr)
print("The precision score of LR is:", round(p_score_lr, 4))
r_score_lr = metrics.recall_score(y_test, pred_lr)
print("The recall score of LR is:", round(r_score_lr, 4))
```

```
Confusion Matrix of LR
 [[3214  196]
 [ 125 3329]]
The accuracy score of LR is: 0.9532
The precision score of LR is: 0.9444
The recall score of LR is: 0.9638
```

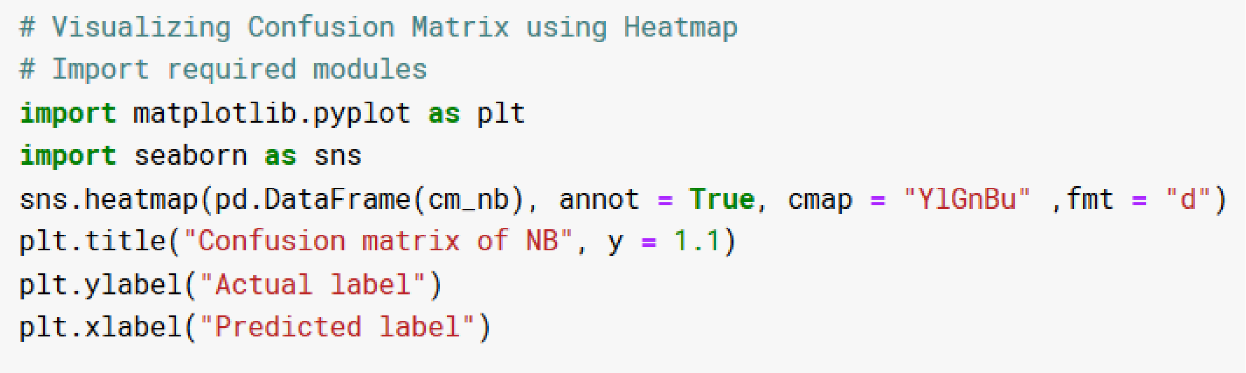
First, we import the necessary module:

from sklearn import metrics

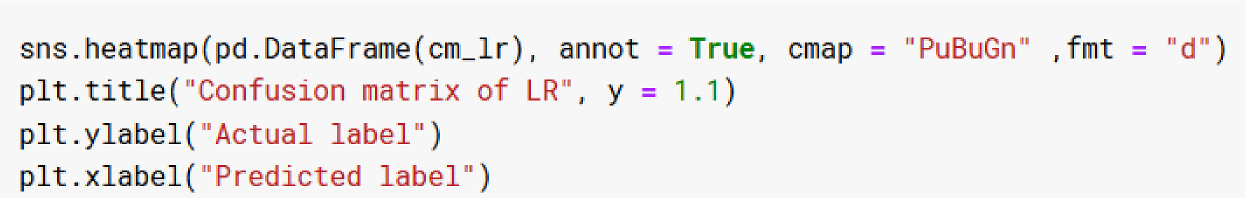
Then, we use the function confusion_matrix in order to calculate the confusion matrix of each classifier. This function is given as parameters the true labels of the news entries (y_test) and the predicted ones by each classifier (pred_nb and pred_lr). Next step, accuracy, precision and recall are calculated, by using the functions accuracy_score, precision_score and recall_score functions respectively, which are given as parameters the same with those of confusion_matrix function. Finally, we print the results as shown in the figures above, keeping for accuracy, precision and recall 4 decimal points.

We also visualized the confusion matrices of the two classifiers. The corresponding parts of the code are presented below:

**Figure 7.6:** Visualization of confusion matrices of NB and LR algorithms



```
# Visualizing Confusion Matrix using Heatmap
# Import required modules
import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(pd.DataFrame(cm_nb), annot = True, cmap = "YlGnBu" ,fmt = "d")
plt.title("Confusion matrix of NB", y = 1.1)
plt.ylabel("Actual label")
plt.xlabel("Predicted label")
```



```
sns.heatmap(pd.DataFrame(cm_lr), annot = True, cmap = "PuBuGn" ,fmt = "d")
plt.title("Confusion matrix of LR", y = 1.1)
plt.ylabel("Actual label")
plt.xlabel("Predicted label")
```

First, we import all the necessary modules.

from sklearn import metrics

import matplot.lib.pyplot as plt

Then, we draw a heatmap using the heatmap function, first for NB and then for LR classifiers. In the heatmap function, we give the following parameters:

- **The Pandas DataFrame**
- **annot = True:** If annot is True, then the data value is written in each cell of the heatmap.
- **cmap = "YlGnBu"** and **cmap = "PuBuGn":** Sequential Colormaps
- **fmt = "d":** Each cell is annotated with the numeric value using integer formatting.

As far as the title of the heatmap is concerned, we define parameter y = 1.1, in order to place the title further higher than the highest y position in the plot, which corresponds to y value equal to 1.

The results are shown in the two diagrams below:

**Figure 7.7:** NB Confusion matrix

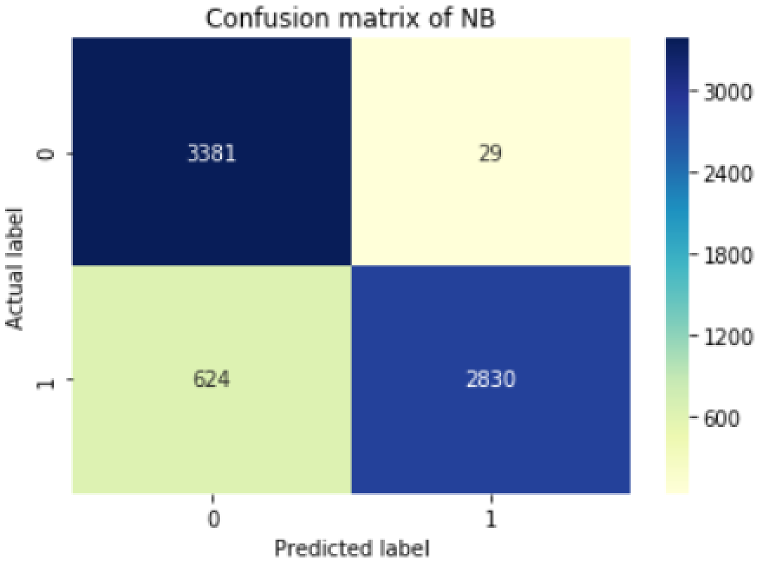
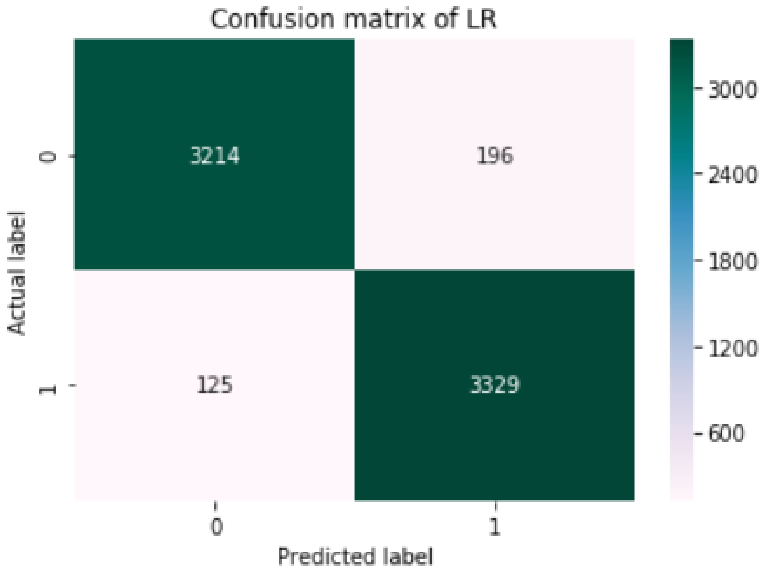
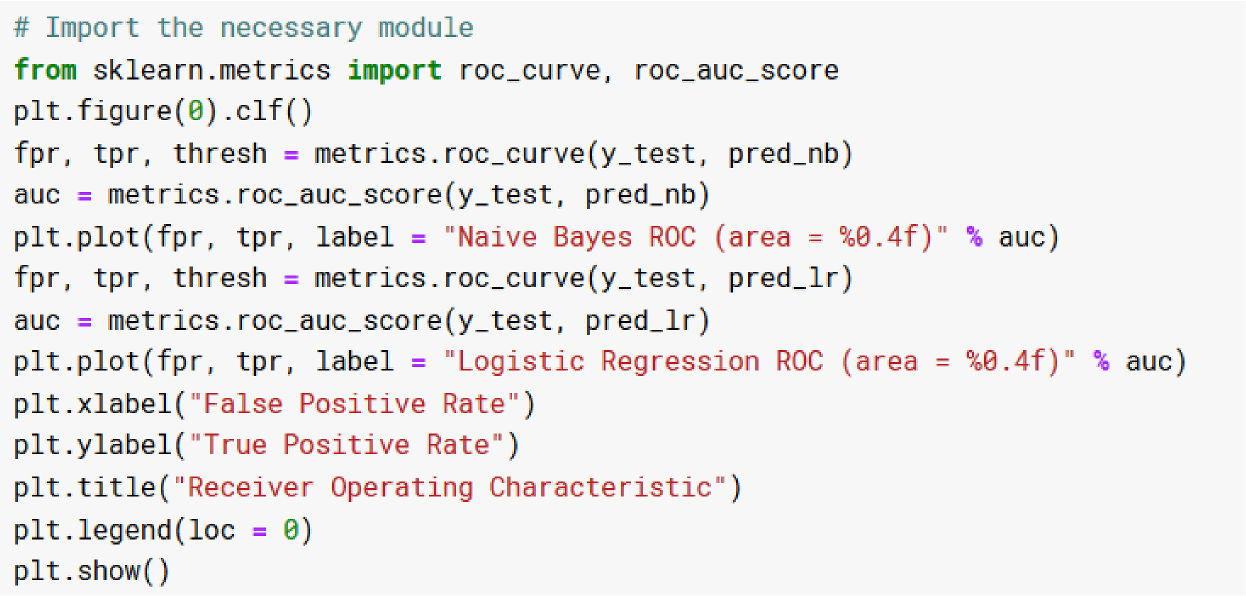


**Figure 7.8:** LR Confusion matrix



Finally, we plotted the combined diagram of their ROC curves, including also the AUC scores for each of them. The part of the code for plotting this diagram is presented below:

**Figure 7.9:** ROC curves and AUC scores



```
# Import the necessary module
from sklearn.metrics import roc_curve, roc_auc_score
plt.figure(0).clf()
fpr, tpr, thresh = metrics.roc_curve(y_test, pred_nb)
auc = metrics.roc_auc_score(y_test, pred_nb)
plt.plot(fpr, tpr, label = "Naive Bayes ROC (area = %0.4f)" % auc)
fpr, tpr, thresh = metrics.roc_curve(y_test, pred_lr)
auc = metrics.roc_auc_score(y_test, pred_lr)
plt.plot(fpr, tpr, label = "Logistic Regression ROC (area = %0.4f)" % auc)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic")
plt.legend(loc = 0)
plt.show()
```

First, we import the necessary modules:

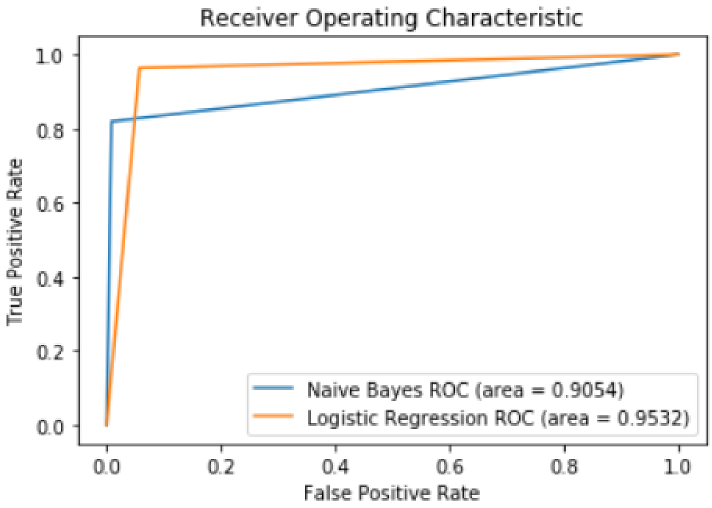
```
from sklearn.metrics import roc_curve, roc_auc_score
```

The roc_curve function is used to compute the Receiver Operating Characteristic (ROC) curve of both classifiers, given as parameters the true labels of the news entries (y_test) and the predicted ones by each classifier (pred_nb and pred_lr). This function returns fpr, tpr and thresh, which correspond to the false positive rate, the true positive rate and the threshold, since ROC curve is created by plotting the tpr against the fpr, at various threshold settings. The roc_auc_score function is used to compute the AUC scores of both classifiers, given again as parameters the true labels of the news entries and the predicted ones by each classifier.

```
plt.figure(0).clf()
fpr, tpr, thresh = metrics.roc_curve(y_test, pred_nb)
auc = metrics.roc_auc_score(y_test, pred_nb)
plt.plot(fpr, tpr, label = “Naïve Bayes ROC (area = %0.4f)” % auc)
fpr, tpr, thresh = metrics.roc_curve(y_test, pred_lr)
auc = metrics.roc_auc_score(y_test, pred_lr)
plt.plot(fpr, tpr, label = “Logistic Regression ROC (area = %0.4f)” % auc)
plt.legend(loc = 0)
```

The result is shown in the diagram below:

**Figure 7.10:** ROC curves and AUC scores



## 8. Conclusions

One way to deal with the problem of fake news detection is via classification. After studying various researches concerning fake news detection, using multiple types of algorithms, as well as different datasets and data preparation methodologies, we implemented our classifier in a kernel created in Kaggle platform, in order to understand the whole process followed and draw proper conclusions, which agree with the results of the literature sources.

It became clear from Chapter 2 that scientists look for ways of dealing with fake news dissemination, focusing on Machine Learning (ML) and Artificial Intelligence (AI), since these fields are gaining more and more ground in our days, as far as solving various types of classification problems is concerned. Fake news detection can be faced as a Natural Language Processing (NLP) problem, since the various datasets used in the researches we studied make use of multiple NLP basics, like the words identification and separation, or the sentences segmentation.

After selecting our dataset, taking into consideration its size and data balance, we did the data preparation, selecting those steps which were adopted the most by Naïve Bayes (NB) and Logistic Regression (LR) algorithms we used, too, in our research. These particular algorithms were selected, based on certain criteria. Those were their frequency of use in the articles studied, as well as their performance, compared to those of the other algorithms used. Thus, it was very crucial for our research to reach proper results.

In Chapter 7, all evaluation metrics of our classifiers were presented. First of all, we confirmed that LR achieves higher accuracy than NB, since the former achieved accuracy of 95.32%, whereas the latter achieved accuracy of 90.49%. Considering as positive class the “fake” class, which receives the value “1”, and as negative class the “real” class, which receives the value “0”, True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) for the two classifiers are shown in the following table:

**Table 7.15:** TP, FP, TN and FN of NB and LR algorithms

| | TP | FP | TN | FN |
|---|---|---|---|---|
| **NB** | 2830 | 29 | 3381 | 624 |
| **LR** | 3329 | 196 | 3214 | 125 |

The precision score of 98.99% of NB algorithm is higher than the precision score of 94.44% of LR algorithm, whereas the recall score of 81.93% of the former is far lower than the recall score of 96.38% of the latter. We can also calculate the $F_1$-score of each algorithm, which is the harmonic mean of precision and recall taking both metrics into account. Using the equation for $F_1$-score given in Table 7.1, we find that NB algorithm has $F_1$-score equal to 89.66%, whereas LR algorithm has $F_1$-score equal to 95.4%, which means that LR classification model is more balanced than the NB one, since it has a higher $F_1$-score.

Another visualization technique we used for showing the performance of our classifiers, except from the confusion matrix, was the Receiver Operating Characteristic (ROC) curve, along with the Area Under Curve (AUC) score of each of them. Considering Figure 7.10 presented in Chapter 7, we confirm again that the LR algorithm overall performs better than the NB algorithm, since the former has an AUC score of 95.32%, whereas the latter has an AUC score of 90.54%.

After studying the literature sources and conducting our own research based on them, we would propose a further research on the importance given on FN, which stands for this news which is fake, but is predicted as real, since we consider worse for a news website or a Twitter account to publish fake news, presenting it as real. On the contrary, although it is certainly not right to hide real news considering it as fake, the impact of this case is lower than that of the previous one. Besides, it is very different deliberately spreading fake news, since the aim of such an action hides devious motives, whereas the aim of hiding real news considering it as fake is mainly done in order to protect the public and further examine the truthfulness of the news before publishing it. Thus, we consider that, when building and evaluating a fake news classifier, it is important to consider having a small number of FN, which also leads to a lower recall score.

# References

**1.** Aborisade, O.M., Anwar, M. (2018) ‘Classification for Authorship of Tweets by Comparing Logistic Regression and Naïve Bayes Classifiers’, *2018 IEEE International Conference on Information Reuse and Integration for Data Science*

**2.** Agudelo, G.E.R., Parra, O.J.S. and Medina, J. (2018) ‘Using Machine Learning for News Verification’, *Lecture Notes in Computer Science*. C.Springer

**3.** Ahmed, H., Traore, I. and Saad, S. (2017) ‘Detection of Online Fake News Using N-gram Analysis and Machine Learning Techniques’, *Lecture Notes in Computer Science*. C.Springer

**4.** Ajao, O., Bhowmik, D. and Zargari, S. (2018) ‘Fake News Identification on Twitter with Hybrid CNN and RNN models’, *Proceedings of the 9th International Conference on Social Media and Society,* p.226-230

**5.** Alrubaian, M., Al-Qurishi, M., Hassan, M.M. and Alamri, A. (2016) ‘A Credibility Analysis System for Assessing Information on Twitter’, *IEEE Transactions on Dependable and Secure Computing*

**6.** Buntain, C., Golbeck, J. (2017) ‘Automatically Identifying Fake News in Popular Twitter Threads’, *2017 IEEE International Conference on Smart Cloud*

**7.** Gilda, S. (2017) ‘Evaluating Machine Learning Algorithms for Fake News Detection’, *2017 IEEE 15th Student Conference on Research and Development*

**8.** Granik, M., Mesyura, V. (2017) ‘Fake News Detection Using Naïve Bayes Classifier’, *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering*

**9.** Jain, A., Kasbe, A. (2018) ‘Fake News Detection’, *2018 IEEE International Students’ Conference on Electrical, Electronics and Computer Sciences*

**10.** Janze, C., Risius, M. (2017) ‘Automatic Detection of Fake News on Social Media Platforms’, *Pacific Asia Conference on Information Systems 2017 Proceedings*

**11.** Krishnan, S., Chen, M. (2018) ‘Identifying Tweets with Fake News’, *2018 IEEE International Conference on Information Reuse and Integration for Data Science*

**12.** Lazer, D.M.J., Baum, M.A., Benkler, Y., Berinsky, A.J., Greenhill, K.M., Menczer, F., Metzger, M.J., Nyhan, B., Pennycook, G., Rothschild, D., Schudson, M., Sloman, S.A., Sunstein, C.R., Thorson, E.A., Watts, D.J. and Zittrain, J.L. (2018) 'The Science of Fake News'

**13.** Mesa, J.P. 'Development of Classification Models for Fake News Detection'

**14.** Oshikawa, R., Qian, J. and Wang, W.Y. (2018) 'A Survey on Natural Language Processing for Fake News Detection'

**15.** Rashkin, H., Choi, E., Jang, J.Y., Volkova, S. and Choi, Y. (2017) 'Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking', *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p.2931-2937

**16.** Roy, A., Basak, K., Ekbal, A. and Bhattacharyya, P. (2018) 'A Deep Ensemble Framework for Fake News Detection and Classification', *2018 International Conference on Web Intelligence*

**17.** Shu, K., Sliva, A., Wang, S., Tang, J. and Liu, H. (2017) 'Fake News Detection on Social Media: A Data Mining Perspective', *ACM SIGKDD Explorations Newsletter*, 19(1), p.22-36

**18.** Zhou, X., Zafarani, R. (2018) 'Fake News: A Survey of Research, Detection Methods, and Opportunities', *ACM Computing Surveys*, 1(1)

## Webpages

**19.** http://liwc.wpengine.com/

**20.** https://www.datacamp.com/courses/natural-language-processing-fundamentals-in-python

**21.** https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html