INTERNATIONAL
HELLENIC
UNIVERSITY

Fake News Detection

Tsarapatsanis Vaios

SID: 3308170027

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

DECEMBER 2018

THESSALONIKI – GREECE

Fake News Detection

Tsarapatsanis Vaios

SID: 3308170027

Supervisor:	Prof. Christos Tjortjis
Supervising Committee Members:	Dr. Christos Berberidis
	Dr. Agamemnon Baltagiannis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

DECEMBER 2018

THESSALONIKI – GREECE

Acknowledgment

Regarding the completion of the hereby dissertation thesis, I would like to thank my Supervisor, Professor Christos Tjortjis, for all the suggestions and feedback in order to achieve the determined scopes and goals of my study. His contribution was significant both on academic level but also at individual level.

Abstract

This dissertation was written as a part of the MSc in Data Science at the International Hellenic University. The study is based on fake news detection with machine learning concepts. Literature review on fake news was conducted in order to review the most significant theory concepts and realize the level of advancement regarding this topic by examining related work. A total number of 940 data points were extracted through a daily web scrapping procedure. The research part provides an experimental analysis with 5 well known classifiers and results are evaluated by appropriate metrics. Finally, the last part of the study is referring to the innovation of this study, the Ranking Model approach, which is capable of labeling new inputs as fake or real.

Tsarapatsanis Vaios

7 December 2018

Contents

1 Introduction

Fake news detection refers to the prediction of alterations of a news article, which is intentionally deceptive. Four decades of deception detection research has helped us learn about how well humans are able to detect lies in text. The findings show that we are not so good at it; in fact, we are only 4% better than chance, based on a meta-analysis of more than 200 experiments. Fake news is considered to be a global problem because it rises widely and constantly. Misinformation and disinformation coexist and as a result the public consciousness and opinion of everyone is affected. As a result, the individual is vulnerable, and his free will may be affected. A new system of safeguards is needed, and this study will contribute regarding the accomplishment of that scope. This dissertation is divided in nine chapters.

The first chapter includes the introduction. In the second chapter, all the relevant background information and the literature review is presented, regarding fake news and machine learning concepts. More specifically, there is information about the theory of fake news with respect to appropriate papers. Additionally, the classification algorithms used are explained alongside the necessary evaluation metrics: accuracy, precision, recall, f1-score, confusion matrix and mandatory time to build the model, in order to obtain insights into algorithmic performance.

The third chapter contains the problem statement of the study and the proposed research solution and methodology. Also, the three research questions are stated. The first question is about the discovery of the most suitable model among Bag of Words and Tf-Idf model. The second question is referring to uncovering the best possible algorithm used. In detail, those algorithms are: Multinomial Naive Bayes, Passive-Aggressive, Logistic Regression, Adaboost and MLP. The final question of this study is related with the introduction of the Ranking model, which can classify new input text-data given by the user. This approach constitutes the innovation of the thesis.

The fourth chapter consists of the data extraction and the preparation for the analysis parts. The methodology for data collection was web scrapping, because public data were not suitable in our case. After the completion of that procedure, data were pre-processed according to various techniques. The final part of that section includes the introduction of Bag of words and Tf-Idf model, and their functionality is discussed.

On the fifth chapter, all the experimental results of the classifiers according to the python code are presented and discussed. These experiments test the power of each algorithm with the addition of useful parameters and tuning process. Also, figures with the evaluation metrics are included.

On the sixth chapter, the results with respect to the mentioned evaluation metrics for every algorithm are discussed and the top classifiers are revealed. The seventh chapter includes the introduction of the Ranking model approach. The relevant arguments and its functionality are discussed in order to operate algorithmically. Finally, its strong and weak points are mentioned.

In the eighth chapter, the results of the entire thesis are mentioned and discussed. All the significant points are stated as well as the possible challenges and limitations which were observed during the completion of the study. Moreover, the future work part is presented with all the incoming plans about the continuation of the research study.

The last chapter contains all the source code used for this research and it is split into two parts. The first part includes the python script, which was used in order to extract data from the websites (web scrapping code) and the second part consists of the main code, responsible for all the experiments and applied techniques.

2 Background and literature review

This section is a literature review on fake news. Relative papers were investigated in order to construct this part. Additionally, machine learning concepts are mentioned and an introduction for each selected classifier is enlisted.

2.1 Fake news Theory

Regarding fake news theory, there are a lot of definitions by different academic papers. Additionally, a notable amount of reports reveals the impact of misinformation and the spreading of fake news in our daily life and society in advance.

2.1.1 Definitions of Fake News

It is a fact that the influence of fake news concerning the individual and the society is not a modern phenomenon. Their existence begins right after the development of the printing press in 1439. Regarding the definition of fake news, it is true that there are multiple interpretations and explanations. A widely adopted definition of fake news is about “News that are intentionally and verifiably false and could mislead readers” (Kai Shu, 2017) [1]. Provided the above definition, two significant findings can be observed. Firstly, there is the aspect of news which indeed contains false information and secondly the news that is formulated in order to mislead consumers. According to academic papers, fake news can be divided into several categories, such as satire and parody. However, some papers accept those types as categories of fake news categories, while other papers do not share this idea. For example, several papers treat satire as a type of fake news due to the false oriented content, while on the other hand satire for some is considered a form of entertainment. Despite that, there is a point which is similar among all determinations. Speaking of that, it is common that fake news adopts the format, look, pattern, writing style of articles and real news content in order to achieve a desirable level of credibility. Thus, in simple words it is true that fake news tries to replicate the appearance of trustworthy news. (Edson C. Tandoc Jr., 2018) [2]

2.1.2 Growing Importance

Since the existence of the fake news phenomenon, its pace of growth is undoubtedly notable. To start with, according to media industry the limitations and barriers have been sharply decreased due to two occasions. Specifically, nowadays websites are easily developed and accessible by every user because it is almost effortless to rise the financial earnings through advertising articles on web environment. Fake news can be accessed, shared and quickly spread with ease in social media platforms like Facebook and Twitter. On top of that, it is a fact that the corresponding total number of users of the mentioned platforms has been discernibly increased through the years alongside with the users` growth rate which can be translated to augmentation of fake news circumstances (Gentzkow, 2017) [3].

2.2 Typology of Fake News

Above, it was stated that fake news can be identified in different types. In this section, six possible formations of fake news are analyzed and explained: satire, parody, fabrication, image manipulation, advertising, and propaganda. Additionally, the mentioned categories have been conducted according to academic articles that contain the search term “fake news”.

2.2.1 Satire

Satire is the style of writing that exposes real-world individuals or organizations in a humorous style usually by the treatment of irony (Condren, 2008) [4]. It is a fact that satire constitutes the most widely used format of fake news. There are a lot of programs mainly in television such as the Daily Show in the United States which mimic the viral news by the addition of humor or exaggeration. Those specific individuals refer to themselves as entertainers rather than typical newscasters. Several studies agree on the fact that satirical programs belong to the media ecosystem, while it is true that they have

greater impact and response on the younger audience due to the humorous approach of delivering the news. Besides that, the essential adoption of the humorous style is mostly used in order to grant criticism concerning political and economic oriented news. Other than that, studies have shown that the audience is knowledgeable in the same way as individuals who are informed by other forms of news media. Simultaneously, satirical programs undoubtedly affect the opinions and political trust of their corresponding audience. Lastly, some studies consider the political news satire as a type of fake news due to its format. More specifically, they use the form of newscasts with the addition of humor and ridicule while the content is real and truly based affair (Edson C. Tandoc Jr., 2018) [2].

2.2.2 Parody

A second major format of fake news is parody. There are a lot of similarities between parody and satire. More specifically, in both cases humor is the key factor regarding captivating the audience. On the other hand, there are also dissimilarities between those formats. Speaking of that, parody focuses on the ludicrousness of an affair and highlights them by producing untrue news stories instead of stating comments in a humorous oriented style. There are a lot of parody websites which are mistakenly considered as actual news websites. The reader may intentionally or not label the news as real and as a result, parody is accepted as a fake news format. Finally, it is significant to be mentioned that parody alongside with satire criticize the media in a way. As a result, journalists are careful and mindful about the content and the credibility of the news (Edson C. Tandoc Jr., 2018) [2].

2.2.3 Fabrication

The third type of fake news is fabrication, which differs fundamentally from satire and parody. In detail, the author or producer of an item is often intentionally trying to misinform the interested individuals and there is no clarification about its falseness. Fabricated items are usually published on social media platforms or on websites. More specifically, the deception of the individuals is greatly boosted when organizations are involved regarding the corresponding item`s publication. Similarly to parody news, fabrication items depend on actual affairs but often with political bias. Not surprisingly,

the items gain legitimacy through social media platforms due to the phenomenon of engaging trusted people. Furthermore, the fabricated items achieve legitimacy through the creation of websites which mimic organizations with credibility. Once the reader accepts and trusts the quality of the source then no further investigation on item`s validation will be done by their side. Finally, it is important to be mentioned that there are two dimensions regarding news fabrication. The first dimension refers to the financial motivation of the author. To explain more thoroughly, the increasing number of clicks regarding news results on attracting advertisers. Hence, financial incentives may occur. The second aspect of news fabrication relates to the development of bots that spam the news. As a result, the corresponding news item acquires widespread acceptance. In addition, the content and the format of the fabricated news is similar or even identical to the real (Edson C. Tandoc Jr., 2018) [2].

2.2.4 Image Manipulation

Image Manipulation is another type of fake news, but the effect is visual and not text-oriented. This category refers to the manipulation of an image either on smaller or greater scale. More specifically, a simple violation of the photo could be the color alteration or removing minor parts. On the other hand, more significant adjustments could be the deletion or insertion of an individual into an image. It is known that media take advantage of these techniques in order to attract the audience with visual content. According to the Reuters code of ethics, light effect changes like balancing the color or image`s tone can be accepted as a presentational tool. In contrast, manipulations such as additions and deletions of elements on an image are not allowed because this kind of actions may misinform people. As a result, the level of manipulation of news media`s perspective is controlled. Currently, the same code does not apply on social media. Thus, manipulated images can be shared and confuse people or even worse mislead them (Edson C. Tandoc Jr., 2018) [2].

2.2.5 Advertising and “Clickbait”

Advertising and “Clickbait” is another category of fake news. In this case, the factor that distinguishes this type from the others is the financial gain. False information is formed in order to characterize or promote advertising materials and it is usually

described by presenting the positive features of the product or individual being advertised. This type of information is considered as fake and can be produced by third parties with a genuine approach. On top of that, sometimes those advertising agencies incorporate with television news in order to be delivered to the audience by authentic news reports.

"Clickbait" is a modern phenomenon which spreads more and more around the internet, often aiming at financial gains. It is the process of attracting a user to click on a post which is connected to an irrelevant website page. As a result, the user is moved to another environment which usually constitutes a commercial site or in general unrelated web source. Consequently, it is considered as a type of fake news because it misleads people.

For example, a post on Facebook became viral during 2017 which was "Clickbait". In detail, the post showed a Middle Eastern man speeding in the United Kingdom and getting arrested by the police. Additionally, the headline of the post revealed that the man responded to police that his car was costlier than the annual police-officer`s income. The item lured a lot of people into clicking it while it gained countless negative and hateful comments. However, the post was not connected with any real news affair, but the users were misinformed and moved to a marketing website (Edson C. Tandoc Jr., 2018) [2].

2.2.6 Propaganda

The final type of fake news is Propaganda, which relates to the political scene. The scope of Propaganda is to influence public consciousness and affect the free will of people to the advantage of a government party or public figure or organization (Edson C. Tandoc Jr., 2018) [2]. There are many examples of propaganda being utilized and exploited as an effective tool in order to manipulate and control public consciousness and opinion, such as in Communist parties and Central and Eastern Europe. The outcome was undoubtedly decisive and forceful. In detail, the respective government`s interests and ideas contributed at militarizing and leading societies into a war (Gatov, 2018) [5].

Propaganda and advertising share similarities, but there is a borderline. Speaking of that, a study is important to be mentioned which investigated people who were commenting on social platforms involving a financial exchange by companies. According to that process, companies demanded positive criticism by the paid people and negative criticism for their competitors. The postings were not exclusively

advertising but the motivation behind that is the financial growth regarding the mentioned companies. Propaganda is based on true events, but biased and often grounded on convincing rather than misinforming (Edson C. Tandoc Jr., 2018) [2].

2.3 Classifiers and Evaluation metrics

In this section, the used classifiers and evaluation metrics are presented and discussed. More specifically, those classifiers are Multinomial, Passive Aggressive, Ada-Boost, Logistic Regression and MLP while the evaluation metrics are accuracy, precision, recall, F-score and confusion matrix.

2.3.1 Multinomial Naïve Bayes

Multinomial classifier belongs to the Naïve Bayes family. According to this category, the classification is based on Bayes` rule or Bayes` formula:

$$P(C = c_k|\mathbf{X} = \mathbf{x}) = P(C = c_k) \times \frac{P(\mathbf{X} = \mathbf{x}|C = c_k)}{P(\mathbf{x})}$$

Figure 1: Bayes rule formula

where

$$P(\mathbf{X} = \mathbf{x}) = \sum_{k'=1}^{ec} P(\mathbf{X} = \mathbf{x}|C = c_{k'}) \times P(C = c_{k'})$$

Figure 2: Bayes rule

C is a variable which includes all the possible events. In this study, the variable C contains all the documents of the dataset, is a vector random variable of the feature values $x = (x_1,..,x_j,\ldots,x_d)$. Each document has one vector. P(C is the conditional probability that a document belongs to class c_k. given the $(c_1,..,c_{k`},\ldots,c_{ec})$. X

$= c_k|X= x)$

feature vector x. Hence, conditional probabilities are computed of particular vectors of feature values for documents of each class and the unconditional probability of a document of each class in order to determine $P(C = c_k|X= x)$. As a result, the Bayes` rule can be rewritten as:

$$P(c_k|\mathbf{x}) = P(c_k) \times \frac{P(\mathbf{x}|c_k)}{P(\mathbf{x})}$$

Figure 3

As it was mentioned before $P(c_k |x)$ is the asked calculation. In order to achieve that, we need to compute $P(x |c_k)$ and $P(x)$. Unfortunately, the calculation of $P(x |c_k)$ is complicated, so Bayes suggests the following decomposition of distribution of x conditional on c_k as a technique to fight the mentioned problem:

$$P(c_k|\mathbf{x}) = P(c_k) \times \frac{\prod_{j=1}^{d} P(x_j|c_k)}{P(\mathbf{x})}$$

Figure 4

where

$$P(\mathbf{x}) = \sum_{k'=1}^{e_C} P(c_{k'}) \times \prod_{j'=1}^{d} P(x_{j'}|c_{k'})$$

Figure 5

A classifier which utilizes these equations in order to achieve his goal is known as Naïve Bayes Classifier.

Multinomial classifier is a different approach with regards to the following assumption:

Naïve Bayes: single draw on a vector-valued variable X of length d.

Multinomial: f draws on a d-valued multinomial variable X.

Finally, the advantage of Multinomial classifier in our study is that the document length is resolved very naturally in the model. On the other hand, the disadvantage of this classifier according to the study is that it assumes independence between multiple incidents of the same word (Lewis) [6].

2.3.2 Passive-Aggressive

This classifier belongs to a large-scale learning and there are similarities with Percepton since they do not need a learning rate. A binary classification consists of sequence rounds. On each round, the algorithm investigates an instance and predicts the label to be either +1 or -1. After the completion of the prediction, the error is calculated, and the algorithm adjusts it in order to learn about the weight vector and improve its performance. The weight vector is $sign(w \cdot x)$, where x is the instance. Every time the margin is a positive number then $sign(w_t \cdot x_t) = y_t$ (where y is the label) and the algorithm has produced an accurate prediction The name of the classifier is related to the corresponding update strategy. More specifically, the constrained optimization problem for round t and new weight vector W_{t+1} is presented:

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \quad \text{s.t.} \quad \ell(\mathbf{w};(\mathbf{x}_t,y_t)) = 0.$$

Figure 6: weights for PA algorithm

- **Algorithm is Passive:** Hinger-loss is zero, that is, $w_{t+1} = w$ whenever $\ell t = 0$.
- **Algorithm is Aggressive:** Loss is positive and w_{t+1} is forced to satisfy the constraint $\ell(w_{t+1};(x_t , y_t)) = 0$ regardless of the step-size required.

According to those two behaviours, the algorithm is called Passive-Aggressive. It is significant to be mentioned that due to the aggressive update strategy, the weight vector may be modified dramatically in order to satisfy the constraint. Consequently, this outcome may lead the weight vector into the false direction (Koby Crammer, 2006) [7].

2.3.3 AdaBoost

AdaBoost classifier is related to the boosting concept. Boosting is an ensemble technique. In detail, it is the combination of weak classifiers in order to create a strong classifier with a good performance. A model is constructed by the training data and then a second model is created which corrects the inaccuracies of the first one. This procedure continues until it reaches the maximum number of models. The AdaBoost classifier is characterized by level one decision trees as weak learners. Level one decision trees are very simple because only one decision is involved for the classification. The weights are updated according to the following function:

$$Weight(x_i) = 1/n$$

Where x is the training instance and n is the number of training instances. The decision about the prediction is taken with respect to the weighted average of the weak learners. Each weak classifier calculates the predicted values with +1.0 (if class A is the outcome decision) and -1.0 (if class B is the outcome decision). The predicted values are weighted accordingly to their respective stage values. Finally, AdaBoost algorithm is taking into account all the decisions by calculating the sum of the total weak learners` outputs and classifies:

Class A : In case the sum is positive value.

Class B : In case the sum is negative value.

Finally, it is significant to be stated that AdaBoost is used for binary problems like the one in our study. (Brownlee, 2016) [8]

2.3.4 Logistic Regression

Logistic regression is the appropriate regression analysis for binary classification problems. The logistic regression is used to describe the data and the relationship between one dependent binary variable and one or more independent variables. The goal of logistic regression is to find the best fitting model to describe the relationship between the dependent variable and a set of independent variables.

The specific algorithm creates the coefficients (and its standard errors and significance levels) of a formula. The reason is the capability to predict a logit transformation of the presence probability of the interested characteristic. Rather than choosing parameters that minimize the sum of squared errors, estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values. It can be used in various fields, including machine learning. The assumptions of the algorithm are the following:

- The dependent variable has to be binary.
- No appearance of extreme values in the data.
- No high correlations in between the predictors.

The function of logistic regression classifier (logit) is presented:

$$Ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

Figure 7: Logit function

The function is the natural log of the odds that the dependent variable is equivalent to one of the categories. Finally, it should be stated that logistic Regression is very popular because the logit function is simple regarding the interpretation of the results (Statistics Solutions) [9].

2.3.5 MLP

Artificial neural networks (ANNs) are computing systems with their concept being motivated by biological brains in order to solve difficult problems. The first neural network was a Percepton model which is a single neuron model. Their capability is related to their efficient representation of the training data regarding the interested output`s description. Neural networks can learn any mapping function.

The predictive effectiveness of neural networks is related with the hierarchical structure of the networks. The data structure can select features at different scales and combine them into higher-order features. Neural networks are composed of neurons which have weighted input signals and produce an output signal according to an activation function. Additionally, the mentioned neurons are organised into networks of neurons. More specifically, a raw of neurons is named as layer, and a network may have numerous layers.

A simple network is composed of three layers: input, hidden and output layers. The input layers are not the neurons which were described previously. They just pass the input data to the next layer and it is the visible part of the network. The hidden layer/layers are not exposed to the input. Their network could be simple or deep with a lot of required calculations. Their target is to output the value, but the train procedure differs according to the complexity of the network. Finally, the last hidden layer is called as output layer and it is responsible for the output value. An example of a neural network schema is presented below:

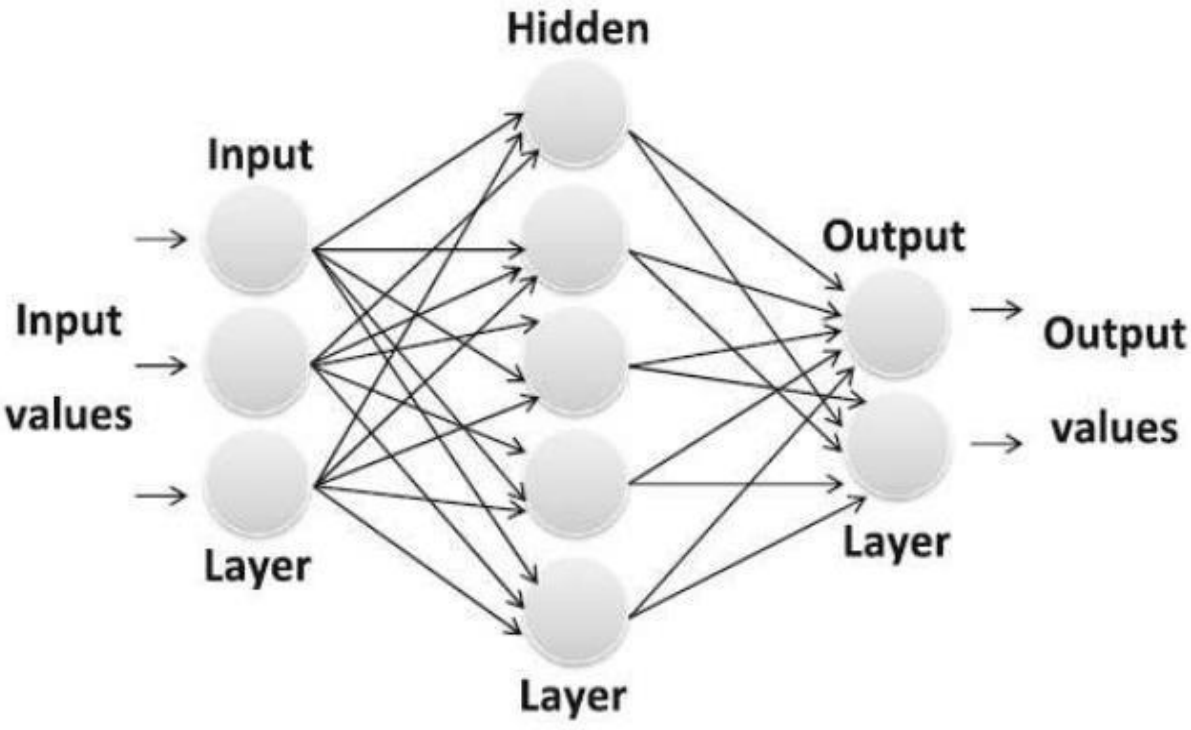


Figure 8: MLP schema

Neural networks can be applied in various problems and gain desirable popularity among other machine learning methods (Brownlee, Crash Course on Multi-Layer Perceptron Neural Networks, 2016) [10].

2.3.6 Evaluation Metrics

A very common, evaluation criteria is the Confusion Matrix. The confusion Matrix depicts includes the True Positive, True Negative, False Positive and False Negative ratio of the algorithm's results. The below figure 9 shows an example of a Confusion Matrix:

		Predicted class	
		Class = Yes	Class = No
Actual Class	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Figure 9: Confusion matrix, an example

The true positives and the true negatives are observations that are correctly predicted, and they are highlighted with green color. The false positives and false negatives are miscalculation of the algorithm and they are highlighted with red color. A good performance for a classifier is achieved through the minimization of those mentioned values, false negative and false positive.

Since those values are defined, it is essential to define the next evaluation criteria that arise from the confusion matrix. In detail, those metrics are accuracy, precision, recall and f1-score. Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. In our study, the dataset is almost symmetric and as a result the usage of this metric is meaningful. Precision is the ratio of correctly predicted

positive observations to the total predicted positive observations. Furthermore, recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. Lastly, f1-score is the weighted average of Precision and Recall. It appears that, this score takes into consideration both false positives and false negatives.

Below, the figure 10 presents their mathematical formulas:

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

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

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

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

Figure 10: mathematical formulas of precision, recall, f1-score and accuracy

All those metrics will contribute in order to discover the best possible classification algorithm. Finally, the required time to build a model constitutes an essential factor and as a result it will be considered for the evaluation of each classifier (Joshi, 2016) [11].

2.3.7 Generalization

Generalization is a crucial machine learning concept for the performance of every model. This terminology is related with the reaction of the machine learning model on unseen data. More specifically, the aim of each model is to generalize well from the training data to any other similar problem. This outcome will ensure a trustworthy model which will be efficient on all case scenarios. Unfortunately, if a learning model does not generalize effectively then two occurrences are responsible for that, known as over-fitting and underfitting. In detail, overfitting and underfitting are the catalysts for machine learning classifiers with poor performance.

An overfitted model is a statistical model that is consisted of more parameters than can be justified by the data. As a result, the model fails to predict as accurate as before with the addition of new data. An example of an over-fitted is following:

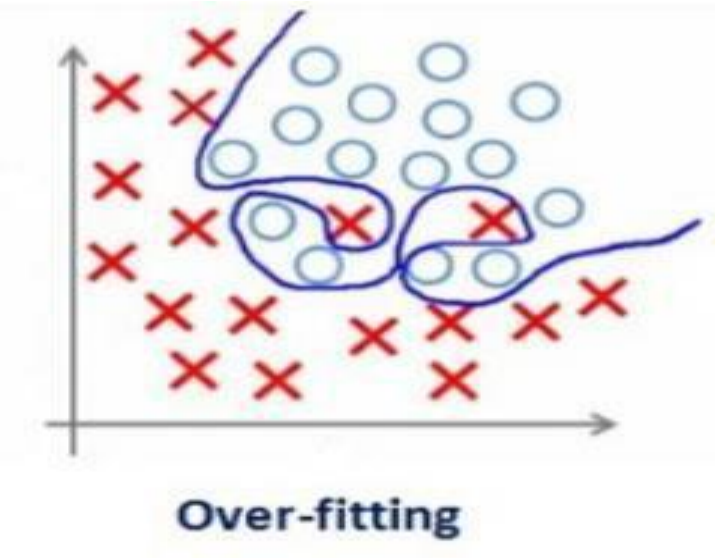


Figure 11: example of overfitted model

As it can observed by the figure, the detail and noise in the training data negatively affect the model`s prediction in fresh data. Random fluctuations in the training data are adopted as concepts and finally learned by the model. When new data are added these concepts do not apply considering the behavior of the recent data might not be the same. Resultantly, the performance of the classifier turns out to be poor. Overfitting is more possible to be caused on nonparametric models which are characterized by flexibility during target function`s learning procedure. According to that, a considerable amount of nonparametric machine learning algorithms includes parameters or techniques to limit and constrain with respect to the level of detail that the model learns without harmful determinations.

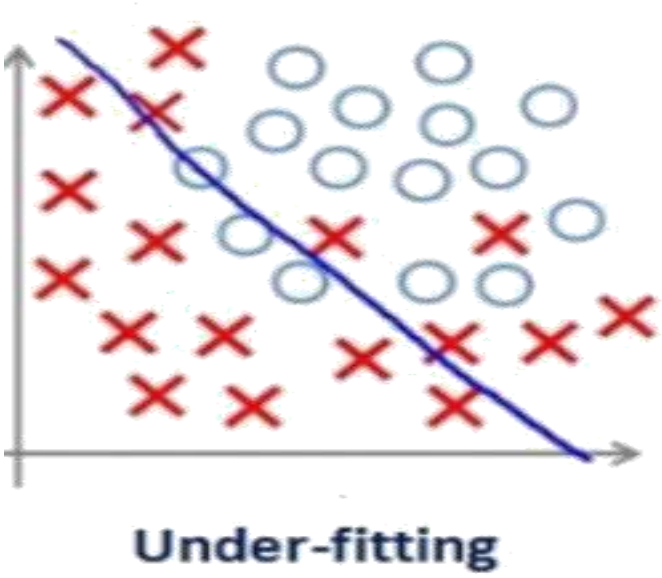


Figure 12: Example of underfitted model

Underfitting is a phenomenon which is related with the inability of modeling the training data or performing well on unseen data. The model is unable understand the relationship among the input values and the target values. As a result, poor performance is the only possible outcome. An example of underfitted model is presented on the figure 12 (Brownlee, Overfitting and Underfitting With Machine Learning Algorithms, 2016) [12].

2.4 Related Work

In the following chapter, recent studies on Fake news detection are presented in order to gain knowledge about techniques and methods which are used in this scientific field. Besides that, the level of activity according to this topic can be observed.

2.4.1 Linguistic and Network-based approaches

This paper utilizes state-of-the-art technologies in order to detect fake news. More specifically, the survey focuses on two approaches. Firstly, the linguistic approaches in which the patterns of the language connected with deception are investigated and analyzed in order to be fully-recognized. For example, most people who lie utilize the language with a specific plan, so they can convince the others.

During this execution, a lot of key features may be observed that reveal them as liars because they are hard to control such as specific verbal frequencies and patterns of pronoun. Technically speaking, the data are represented according to the "bag of words" model, which treats each word with identical significance. As a result, the most frequent words are analyzed, so the deceptive patterns can be uncovered. Simultaneously, tagging of words into corresponding lexical cues for instance, parts of speech is another option of producing frequency sets and therefore explore the linguistic cues regarding to deception. The disadvantage of this representation technique is the isolation of the words and non-exploitation of the united context information.

Also, deeper language structures are involved because word analysis is not sufficient. As a result, Probability Context Free Grammars (PCFG) boosts the accuracy of prediction by enforcing the deep syntax analysis. In detail, a sentence is transformed to a set of its parts of speech which can describe the syntax structure. After that, probabilities are assigned to the corresponding set in order to predict for example if a verb or noun is coming next.

Another improvement of the accuracy score is the semantic analysis addition. Generally, this method is used for describing the content meaning of words with probabilities applied to a large text. This technique can be applied for discovering deception cues with effective results. More specifically, signals of truthfulness are extracted and analysed by a profile which is consisted of personal reviews and opinions. Furthermore, multiple profiles are compared which are derived from a large database with relative data. Hence, the outcome of this process is that a profile with a deceptive writer may stand out or differ from the rest profiles.

Finally, it is worth mentioning that according to Rhetorical Structure Theory (RST) analytic framework the differences of deceptive and truthful messages can be captured. Afterwards, a Vector Space Model (VSM) can be incorporated which is able to determine the position for each message in a multi-dimensional RST environment with regards to the distance of truth and deceptive points.

Speaking about the analysis part, two widely known classifiers were used: Support Vector Machine (SVM) and Naïve Bayes. The choice of the mentioned two models is relates to the fact that they can predict instances with numeric clustering and distances at its core. Meaningful distance functions and correlation indexes are the most valuable factors that influence and finalize the accuracy of the classification process. The under

lying hypothesis is based on the unintentional usage of emotions by the liar`s side and SVM classifiers achieved 86% accuracy which proves that deceivers over-produce negative emotionally words with regards to truthful reviewers.

The linguistic approach is very promising, but the generalization rule is not at a similar desirable level due to the veracity of real-time news. Secondly, the network approaches in which: " network information, such as message metadata or structured knowledge network queries can be harnessed to provide aggregate deception measures". As a result, a hybrid model derives incorporated with machine learning techniques. (Niall J. Conroy) [13]

2.4.2 Bag-of-words and TF-IDF approach

In this approach the text representation of text input is completed through bag-of-words (BOW) and term frequency (TF) and term frequency-inverse document frequency (TF-IDF) techniques. Both models are ideal for language-oriented problems and document classification.

In detail, bag-of-words model can extract features from the text and afterwards use various machine learning classifiers. The approach takes into account the occurrence of a word within the document while the structure and the order of the words are unimportant factors. Subsequently, the lack of studying the word relationships in the context constitutes a limitation. Due to that fact, it is called as "bag" of words model. The complexity of this method derives on the way of designing the vocabulary of known words alongside with the relevant score of its word occurrence and frequency.

The second used model is TF-IDF model. Each word in the collection of documents is assigned with a weighted score which is based on the importance of a word according to how many times it was found in the document.

The UCLMR team calculated TF vector of the headline and body text of a document and afterwards they determined the cosine similarity TF-IDF vectors between those two features. Besides that, tokenisation technique was applied and stop words were removed with regards to pre-processing stage. As a result, a vocabulary was conducted with the 5.000 most frequent words in the training set, while at the same time TF vectors and TF-IDF cosine similarity were connected in a feature vector of total size 10.001. Finally, the combined vector was used into specific classifier algorithms. The schematic diagram of UCLMR`s system is unfolded on Figure 13.

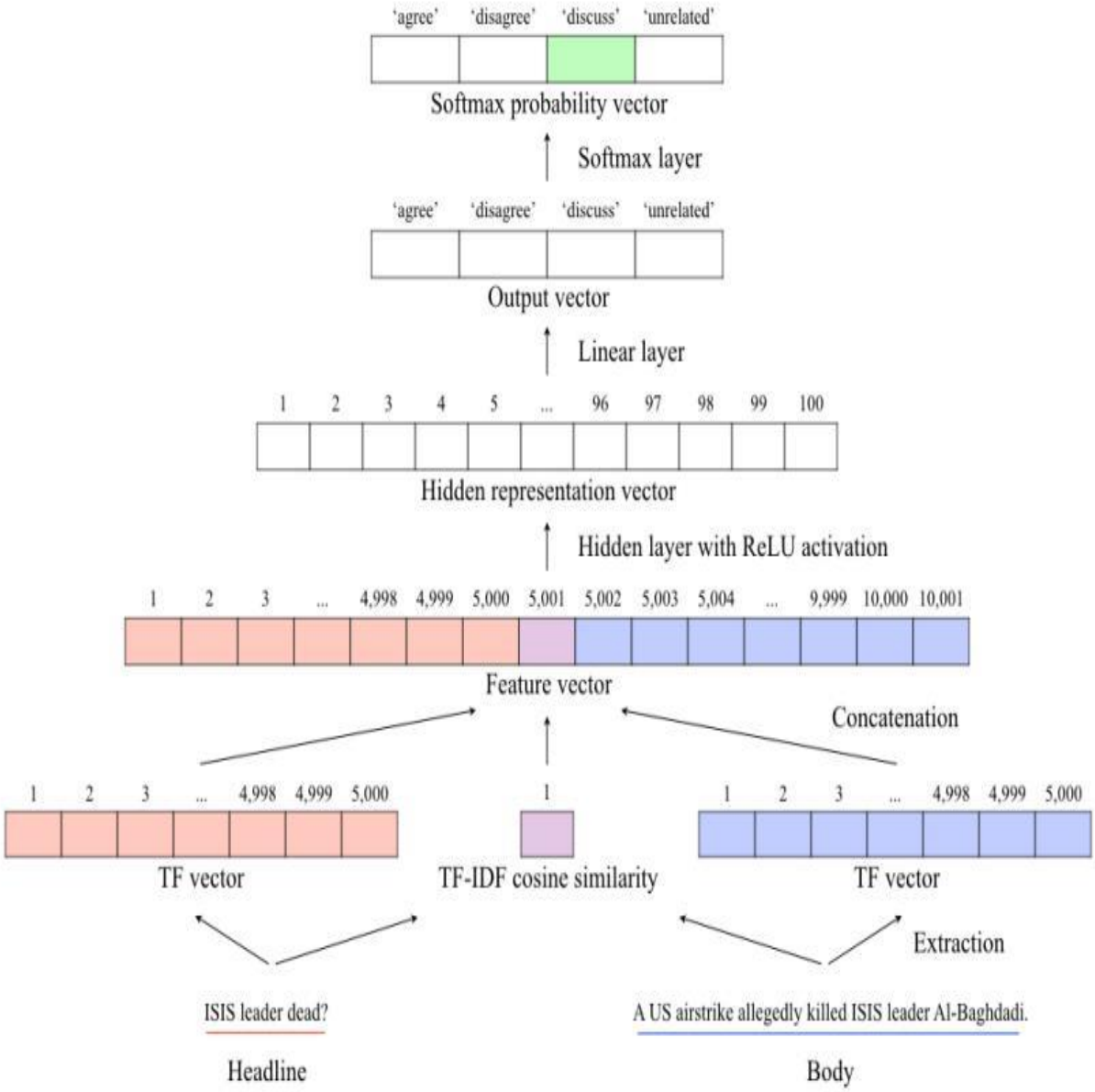


Figure 13: Schematic diagram of UCLMR`s system

The output label includes four possible decisions for every new input data: “agree”, “disagree”, “discuss”, “unrelated”. The chosen classifier for the training procedure was MLP and its performance was tested on 50 random splits of the data. In advance, the

algorithm was evaluated by a confusion matrix. More specifically, the achieved results are presented below:

Pred. / True	'agree'	'disagree'	'discuss'	'unrelated'	Overall	% Accuracy
'agree'	838	12	939	114	1,903	44.04
'disagree'	179	46	356	116	697	6.60
'discuss'	523	46	3,633	262	4,464	81.38
'unrelated'	53	3	330	17,963	18,349	97.90
Overall	1,593	107	5,258	18,455	25,413	88.46

Figure 14: evaluation results of UCLMR`s system

As it can be observed the "agree" label predictions of the classifier achieve an outstanding performance but simultaneously the "disagree" label predictions are poor according to this evaluation method. (Benjamin Riedel, 2018) [14].

3 Problem statement, proposed solution/ methodology

3.1 Topic and Research problem

Fake news mimic news media content with formations which were described in the literature review part. This phenomenon has drawn interest and attention in political text and various other topics such as stock values, vaccination, nutrition etc. Moreover, there are websites whıch publish fake news as satire or humor related to current affairs or new events, while some other websites aim for the profit which is gained by clicks. Internet is becoming an inseparable source of knowledge and entertainment.

Consequently, the internet is an integral part of everyday for every individual. Fake news is concerned to be a global problem because it rises widely and constantly. Undeniably, fake news enlarges other information disorders, such as misinformation (misleading information) and disinformation (false information which is intentionally directed). As a result, the individual is vulnerable and his free will may be affected (British Council) (David M. J. Lazer, 2018) [15, 16]. New methods need to be developed in order to fight this incident.

This study will contribute in addressing this problem and therefore, reinforce the capabilities of every individual. Finally, the major motivation for this topic is the following:

- Web articles and social media are a powerful source of information, but it is a fact that fake news coexist alongside. There are some patterns that can be discovered and utilized in order to combat fake news which are not observable by human. Quantitative methods such as data mining and machine learning can contribute on resolving that problem.

3.2 Research Questions and Methodology

According to this study, three main questions are stated:

1. Which is the most suitable model of this study?

2. Which one is the classifier with the best performance after investigating insightful evaluation metrics?
3. Is there any approach to classify a new input article given by the user as fake or real?

The methodology of this study is explained in six steps.

- First, a uniformly sampled large dataset of web articles is extracted and collected. The type of these data may belong to fabrication, advertising, “Clickbait”, Parody or Propaganda category.
- Afterwards, pre-processing is applied to the data in order to be clean and structured for the analysis part.
- Convert the text of data into integers so the models of this study can extract insightful rules and patterns.
- Apply various machine learning algorithms on both models and discover the one with the best performance.
- According to the optimal model and classifier, calculate the weights for each word in the dataset.
- Finally, compute the Ranking index and label a new unseen input record which is given by the user.

4 Design and Implementation

In this section, three parts are presented. The first part is about the construction of the dataset, while the second part is referring to the necessary pre-processing stages for the data in order to be clean and structured. Finally, the last part is about the model contraction of this study: Bag of words and Tf-Idf model.

4.1 Dataset

The first step in order to find the ideal dataset regarding our analysis part was an online survey. Generally, there are multiple ways to retrieve data such as: fast checking, industry detectors, online platforms (Kaggle, github), and expert journalists. In our case, we searched on the mentioned platforms for the most widely used datasets according to our topic. The results were not encouraging due to certain reasons. In detail, in some cases the existence of real news was not possible. Other than that, there were scenarios without accurate labeling. Finally, another challenge for getting the dataset is time oriented. Verification of real time events and affairs is not an easy task.

As a result, the methodology for extracting data was web scrapping combined with online dataset. More specifically, our final dataset includes 2 labels, **real news:** news that are provided by popular news articles websites and **fake news:** news that are confirmed to be fake.

A python script was created, which can retrieve data from 3 websites: New York times, Reuters world news, Washington Post. All the information was labeled as real news while the total number of records is 534. The script was able to retrieve 45 real news data every day from all the 3 aforementioned websites. This relates to the fact that the front page of those websites is filled with renewed data approximately once per day. The entire process for real news data extraction lasted 12 days. The daily extraction of data is presented in figure 15:

Websites	Daily data extracted	% of total Real news
New York Times	20	45%
Reuters	15	33%

Washington Post	10	22%

The same methodology could not apply for the fake news part. There are websites, such as hoax-slayer which provide confirmed fake news, but the automation of the downloading process regarding the data could not be executed due to the page`s format. Only manual contribution is possible for extracting those desirable records.

Subsequently, the fake news data were downloaded by 2 available datasets on Kaggle, alongside with hoax-slayer`s aforementioned manual procedure. Afterwards the fake news data were combined with the other part (Real news). The total number of fake records is 406. That leads to a dataset with a total number of 940 records with three attributes: Article Id, Title and Summary.

- **Article Id:** The Id number of each article in the entire dataset.
- **Title:** large heading displayed above the article's content and the basis for the article's page name and URL.
- **Summary:** the text which briefly describes the entire article`s text with a specific format.

Figure 15: Data examples

Two online articles (New York Times & Washington Post) are observed on figure 2 as an example of the dataset`s attribute construction. More specifically, the Title consists of the bold words while the other part composes the Summary of the document.

4.2 Pre-processing stages

Data pre-processing enables the transformation of raw data into an understandable format with structured substance. The extracted data for the study could be incomplete, inconsistent, unstructured or contain missing data. As a result, the classifier might not be able to operate or correctly execute its procedure. Data pre-processing is a method which efficiently resolves such situations (Technopedia) [17].

According to our case, missing data were filled in order to avoid algorithmic failures in the analysis part. At the same time, punctuations and elements were removed from each document`s text so the noise in the dataset is minimized at a desirable level. Additionally, stop words were removed because their existence provides no insights or any useful information.

Lastly, stemming and lemmatization techniques were applied to the dataset. Stemming is a technique which removes the end of a word or sometimes the beginning of a word. Stemming method is very useful in this study. For example, this process helps the model to handle two words with the same meaning (same information) but one in singular and one in plural, as one. Otherwise, the model will not be able to grasp that non-meaningful difference and as a result, it will accept them as distinct words.

Lemmatization is a technique of converting the words to its original dictionary form. There are several dictionaries where the algorithm is allowed to investigate the morphological form of each word. Unquestionably, lemmatization is beneficial for our study because it achieves reduction of the inflectional forms of a word to a common base formation. (Risueno, 2018) [18].

4.3 Model Construction

In this section, two models are discussed that were used in the analysis part. In detail, it is Bag of Words model and Tf-Idf Model.

4.3.1 Bag of Words (BOW) Model

The first model which was used for the analysis part is the bag of words (BOW) model. More specifically, count vectorizer can convert a collection of text documents to a matrix of token counts. Those tokens are unique, and they form a dictionary. As a result, the total size of the matrix is: *Documents X unique tokens.* An example of the mentioned matrix is following on Figure 3. (Analytics Vidhya, 2017) [19]

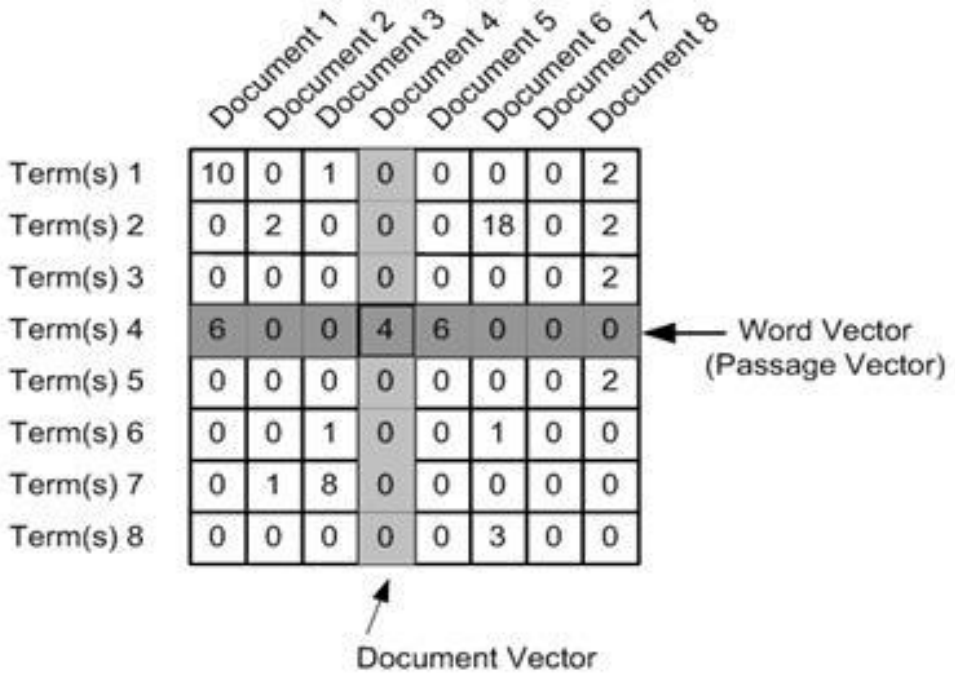


Figure 16: Matrix of Bag of words model

According to that matrix, it can be observed that the first term of the dictionary exists 10 times in Document 1, 0 times in Document 2, 1 time in Document 3 etc. At the same time, the word vector and document vector can be seen on the marked areas for further details of each word or document, according to the term occurrences. Consequently, terms that are mostly used in fake or real news data can be spotted and afterwards give the necessary information to each algorithm about classifying an article. Finally, unusual words with low occurrences can be revealed which may lead to the fact that they are genuine for identifying real news in the collection of documents.

4.3.2 Tf–Idf Model

The second model of the study is Tf-Idf Model. This model is different from the bag of words model since it considers the occurrence of a word in the entire corpus and not in a single document. More specifically, it calculates the relative frequency of all the words in a document and compared with the inversion proportion of the specific word over the

whole corpus-dataset. The mathematical formula for calculating TF-IDF is following: for a term t in a document d, the weight Wt,d of term t in document d is given by:

$$Wt, d = TFt,d \log(N/DFt) \text{ , where:}$$

- TFt,d is the number of occurences of t in document d
- Dft is the number of documents containing the term t.
- N is the total number of documents in the collection of documents.

Besides that, it is significant to be stated that common words have higher TF-IDF values. According to our study, words with low participation will be scored with small Tf-idf values and this could give an insight. In detail, the fact that these words are rare could signify its authenticity and probably exist on real content. Usually, fake contents duplicate the format and some words that are in real news, so they will be clustered together with higher Tf-Idf values. In this study, the tf-idf vectorizer was used in order to convert the collection of raw documents into a matrix of Tf-Idf features and finally build the model (Ramos) [20].

5 Experimental Results

In this section the results of 5 different classifiers: **Multinomial, Passive-Aggressive, MLP, AdaBoost, Logistic Regression** will be presented with respect to both models. The metrics for the evaluation part are: ~~10 fold Cross-Validation on Accuracy and also~~ accuracy, precision, recall, F1-score as well as the confusion matrix of a specific split. Finally, it is significant to mention that the test size is 33% of the total dataset. The major factor for deciding regarding the parameter tuning will be the average accuracy after 10 folds of out-of-sample Cross-Validation. Finally, the code was seeded with random estate = 10 for all the experiments.

5.1 Multinomial Classifier

The first algorithm is Naive Bayes classifier for multinomial models. The motivation for picking this classifier is because of its simple design which makes them very attractive. Moreover, they have been demonstrated to be fast, reliable and accurate in a number of applications. The specific classifier is ideal for discrete features, such as word counts for text classification and this is the beneficial reason for including it in the study. It is a fact, that alpha parameter affects the results of the classification and it needs further explaining. In detail, for each class *y*, the distribution is parameterized by vectors $\theta y. = (\theta y_1, \ldots, \theta y_2)$ where n is the number of features and θy_i is the probability $P(x_i \mid y)$ of feature *i* appearing in a sample belonging to class *y*

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Where,

- N_{yi} is the number of times feature i appears in a sample of class *y* in the training set T.
- N_y is the total count of all features for class *y*.

Alpha parameter may have values where $a \geq 0$. This holds for features which are not present in the learning samples and avoids zero probabilities in further computations (Andrew McCallum, 1998) [21]. Different values of the alpha parameter where tried out in order to find out the best possible performance of the classifier. More specifically, a table of the most significant alpha values is following for Bag of words model:

Alpha (a)	Accuracy	Precision	Recall	F1-score
0.1	0.817	0.836	0.817	0.818
0.2	**0.823**	**0.839**	**0.823**	**0.824**
0.3	0.817	0.834	0.817	0.818
0.4	0.810	0.828	0.810	0.812
0.5	0.804	0.826	0.804	0.806
0.6	0.803	0.825	0.803	0.806
0.7	0.800	0.821	0.800	0.802
0.8	0.804	0.825	0.804	0.806
0.9	0.800	0.823	0.800	0.802
1	0.794	0.820	0.794	0.796

After all the experiments it can be revealed that the optimal value for the alpha parameter in Bag of Words model is 0.2. Also, it is a fact that as alpha parameter increases the performance of the Multinomial classifier slightly decreases. Finally, the confusion matrix is the following:

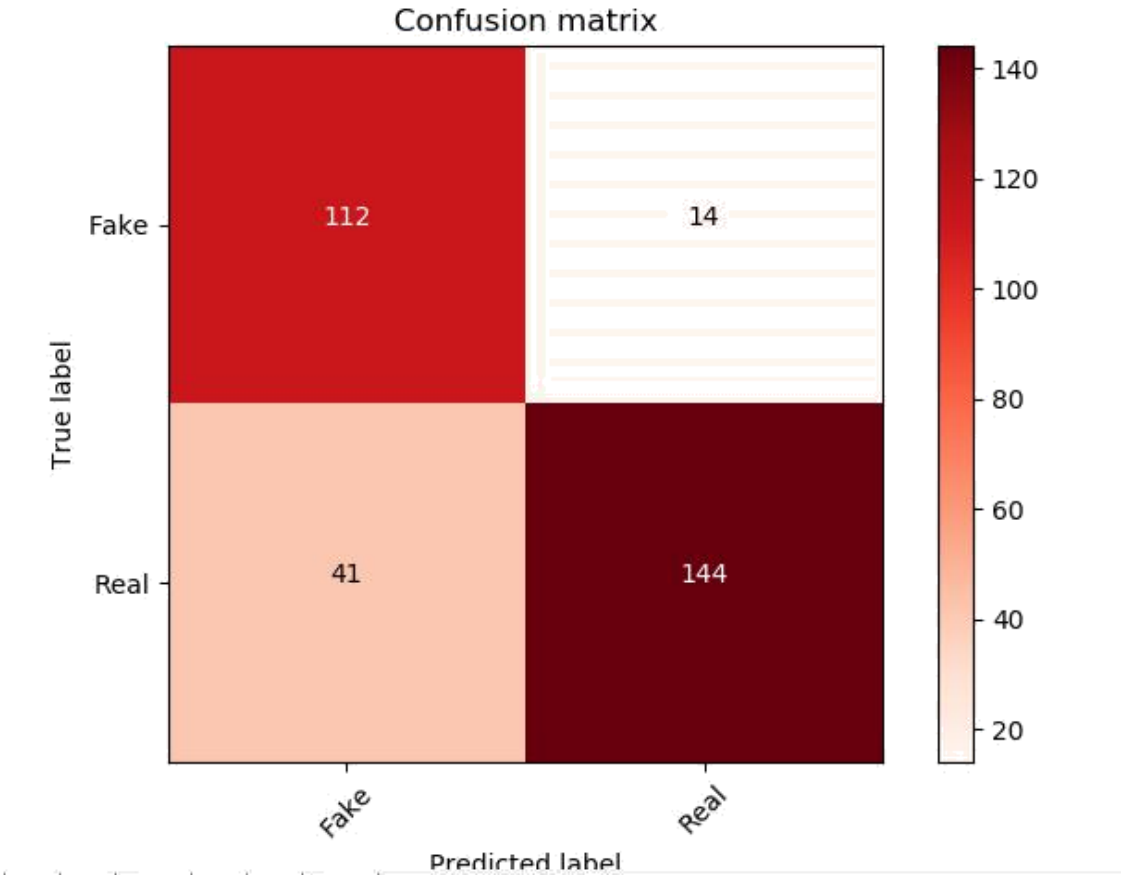


Figure 15

The outcome of the Figure 18 shows that 112 out of the total 126 fake records were correctly predicted as fake, while there were 14 errors. On the other hand, 144 out of the total 185 real records were correctly predicted as real, while there were 41 errors.

The same experiments were executed for the second model of the study (Tf-Idf Model). In specific, the table regarding the ideal alpha parameter is the following:

Alpha (a)	Accuracy	Precision	Recall	F1-score
0.1	**0.826**	**0.847**	**0.826**	**0.828**
0.2	0.817	0.838	0.817	0.818
0.3	0.817	0.838	0.817	0.818
0.4	0.814	0.834	0.814	0.816
0.5	0.820	0.834	0.820	0.822
0.6	0.817	0.834	0.817	0.818
0.7	0.817	0.832	0.817	0.818

0.8	0.817	0.830	0.817	0.818
0.9	0.823	0.835	0.823	0.825
1	0.823	0.835	0.823	0.825

In this case, a = 0.1 constitutes the best option for Multinomial classifier. As a result, the accuracy of the algorithm after 10-fold cross validation reached the point of 83.618%. The corresponding confusion matrix is below:

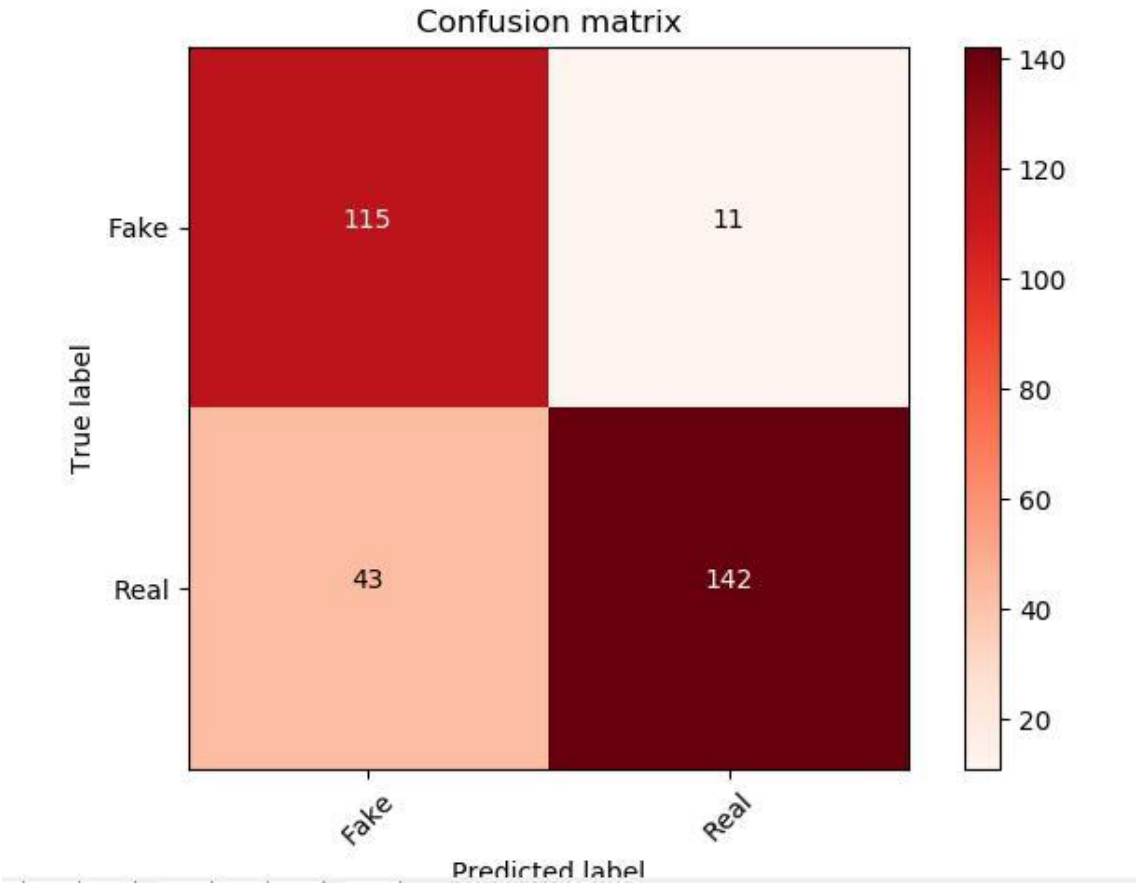


Figure 16

Figure 19 clarifies that 115 out of the total 126 fake records were correctly predicted as fake, while there were 11 misclassifications. Afterwards, 142 out of the total 185 real records were correctly predicted as real, while there were 43 misclassifications.

Undoubtedly, the used evaluation metrics revealed that the Multinomial classifier is marginally more efficient in Tf-idf model in comparison to Bag of words model.

Model	Alpha	Accuracy	Precision	Recall	F1-score	10 fold Cross-Validation
BOW	0.2	0.823	0.839	0.823	0.824	82.40%
TF-IDF	0.1	0.826	0.847	0.826	0.828	82.80%

5.2 Passive-Aggressive Classifier

The second chosen algorithm for the analysis part is Passive-Aggressive classifier. Two parameters were tested in order to accomplish the best tuning of the algorithm. In detail, those are the parameter n_iter and C.

- n_iter is the number of passes over the training data which is known as epochs.
- C is a float number which refers to the maximum step size (regularization).

To start with, 7 different values of n_iter were tested in order to see how the classifier reacts on those inputs. Hence, these are the results of the experiments according to BOW model:

n_iter	Precision	Recall	F1-score	Cross-Validation
50	0.813	0.810	0.811	80.76%
100	**0.815**	**0.814**	**0.814**	**81.71%**
150	0.804	0.804	0.804	81.08%
200	0.805	0.804	0.804	80.92%
500	0.817	0.817	0.817	81.56%
1000	0.804	0.804	0.804	81.08%
5000	0.805	0.804	0.804	81.40%

The best possible value for n_iter parameter is 100. However, it is important to mention that the alternations on the evaluation metrics are almost steady. Afterwards, the next parameter is C. After trying various inputs, the results revealed that C=0.01 is the best option for Bag of Words model. Some indicative values are presented:

Model	C	Accuracy
BOW	**0.01**	**0.804**
BOW	0.50	0.798
BOW	1	0.800
BOW	1.90	0.803

Nevertheless, it decreases the overall accuracy of the classifier, so the parameter C was removed. Finally, the confusion matrix is following with tuned classifier:

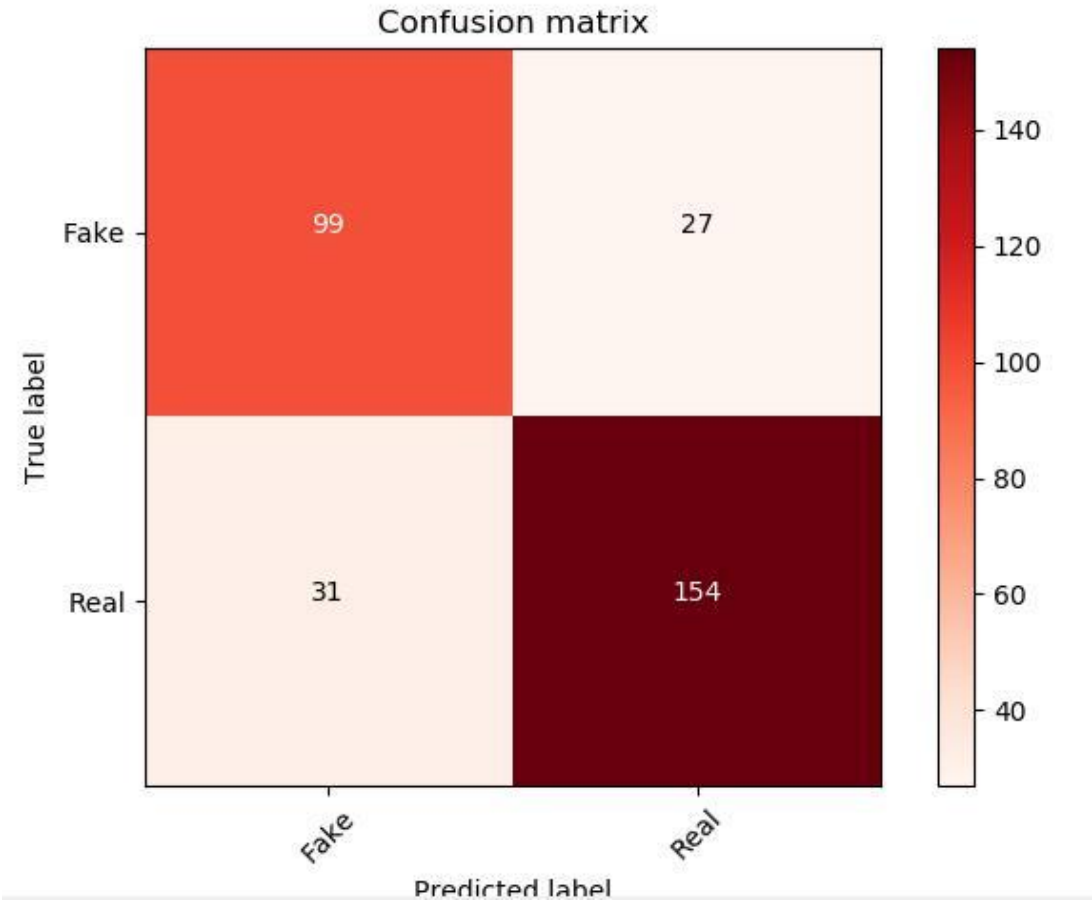


Figure 17

The results of the Figure 17 show that 99 out of the total 126 fake records were correctly predicted as fake, while there were 27 errors. Also, 154 out of the total 185 real records were correctly predicted as real, while there were 31 errors.

The same experiments were tested out for the second model, Tf-Idf. Firstly, these are the results of n_iter trial and error procedure:

n_iter	Precision	Recall	F1-score	Cross Validation
50	0.860	0.855	0.856	86.17%
100	0.860	0.855	0.856	86.33%
150	0.850	0.846	0.847	86.32%
200	0.861	0.859	0.860	85.34%
500	0.861	0.859	0.860	85.39%
1000	**0.871**	**0.868**	**0.869**	**85.53%**
5000	0.865	0.862	0.862	85.42%

It is revealed that the best value for n_iter parameter is 1000 but all the outcomes are slightly different. Thereupon, the following table shows the C parameter reactions:

Model	C	Accuracy
TF-IDF	0.01	0.859
TF-IDF	**0.50**	**0.865**
TF-IDF	1	0.859
TF-IDF	2.00	0.859

Out of all cases, the most promising value for C parameter is 0.50 but as it seems the insertion of that parameter decreases the overall accuracy. As a result, the final decision was the deletion of the mentioned parameter. Finally, the confusion matrix is the last step for the evaluation part for the second model:

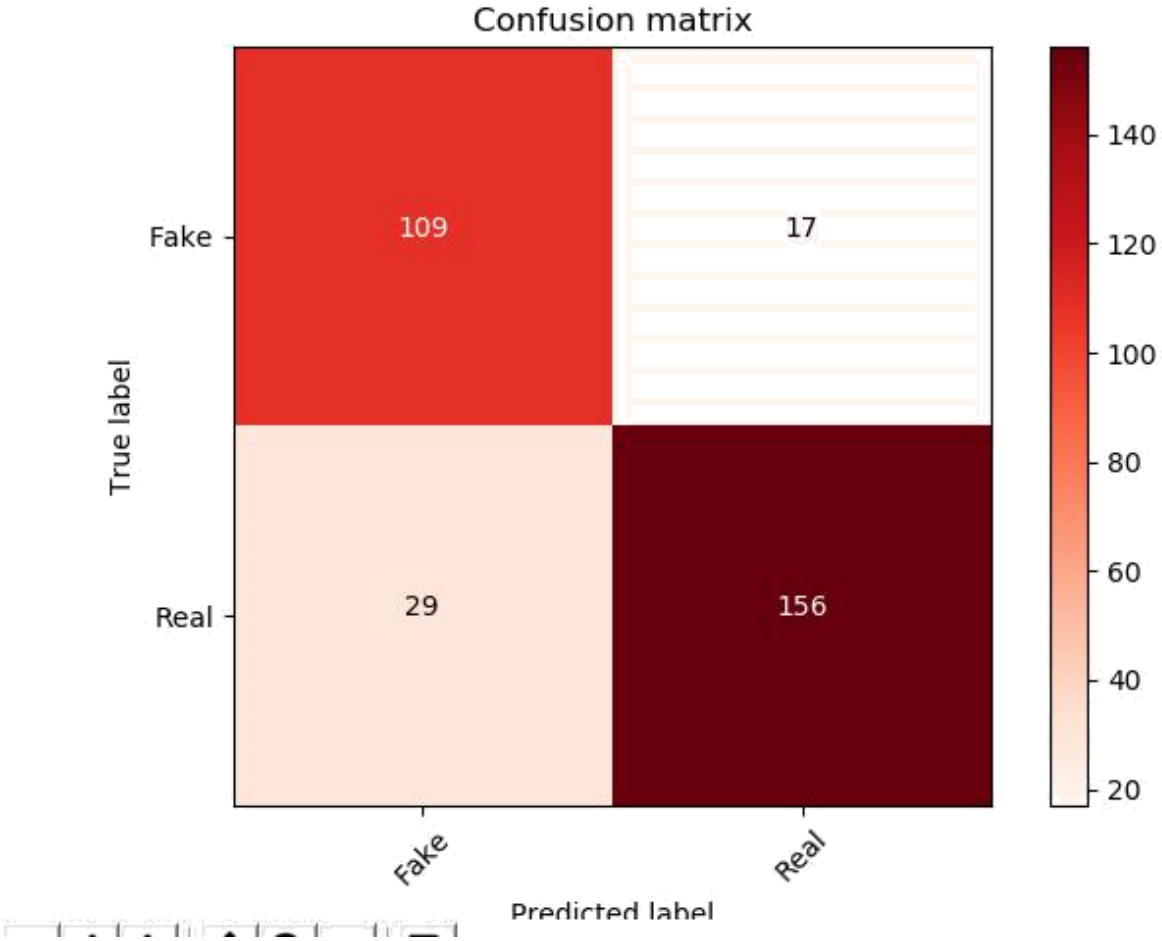


Figure 18

Figure 18 reveals that 109 out of the total 126 fake records were correctly predicted as fake, while there were 17 errors. On top of that, 156 out of the total 185 real records were correctly predicted as real, while there were 29 errors.

Observing the performance of Passive-Aggressive algorithm for both models, we conclude that the classifier`s utilization on Tf-Idf model outperforms thr BOW`s scenario.

Model	n_iter	C	Cross-validation Accuracy	Precision	Recall	F1-score
BOW	100	-	81.71%	0.815	0.814	0.814
TF-IDF	1000	-	85.53%	0.871	0.868	0.869

5.3 Logistic Regression

The third classifier used is Logistic Regression which is a liner classifier. The performance of this algorithm is widely tested in text-oriented problem and as a result, that was the main motivation for selecting it in the study. There are two parts for evaluating Logistic Regression. First, the classifier was tested without any parameters. The second part is referring to the addition of C parameter as a catalyst for tuning it in both models. It is significant to mention that C parameter is a float variable which signifies the regularization strength. In detail, smaller values specify stronger regularization.

1st PART: Bag of Words model

Model	Accuracy	Precision	Recall	F1-score	10-fold Cross val-idation
BOW	0.823	0.824	0.820	0.823	81.71%

Additionally to those metrics, the confusion matrix gave those results:

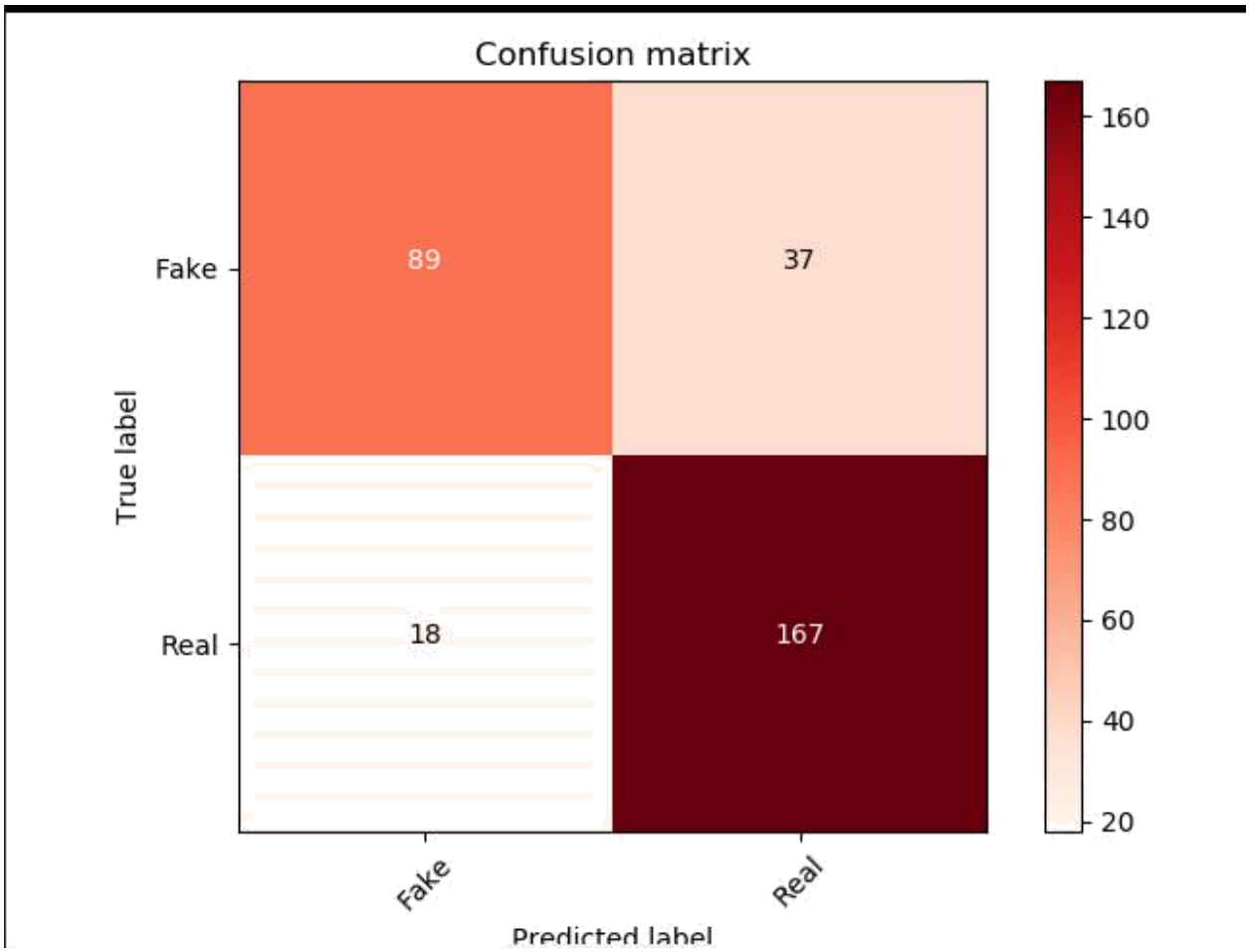


Figure 19

Figure 19 reveals that 89 out of the total 126 fake records were correctly predicted as fake, while there were 37 errors. At the same time, 167 out of the total 185 real records were correctly predicted as real, while there were 18 errors. So far, Logistic regression managed to predict more efficiently the Real records while on the other hand, it performed worse regarding Fake records compared to the other two classifiers. There is a comparative advantage in the second part of the confusion matrix.

2nd PART: Bag of Words model

C	Accuracy	Precision	Recall	F1-score	Cross-Validation
0.01	0.846	0.847	0.846	0.843	80.92%
0.5	0.820	0.817	0.820	0.817	81.88%
1	0.823	0.823	0.823	0.820	81.71%
1.5	0.823	0.823	0.823	0.820	82.03%

2.0	**0.826**	**0.826**	**0.826**	**0.824**	**82.03%**

The best value for C is 2 which is the point with the stronger regularization. Bigger inputs of C (C > 2.0) give us the same results or worse repeatedly. Finally, the confusion matrix with the correct tuning of the classifier is following:

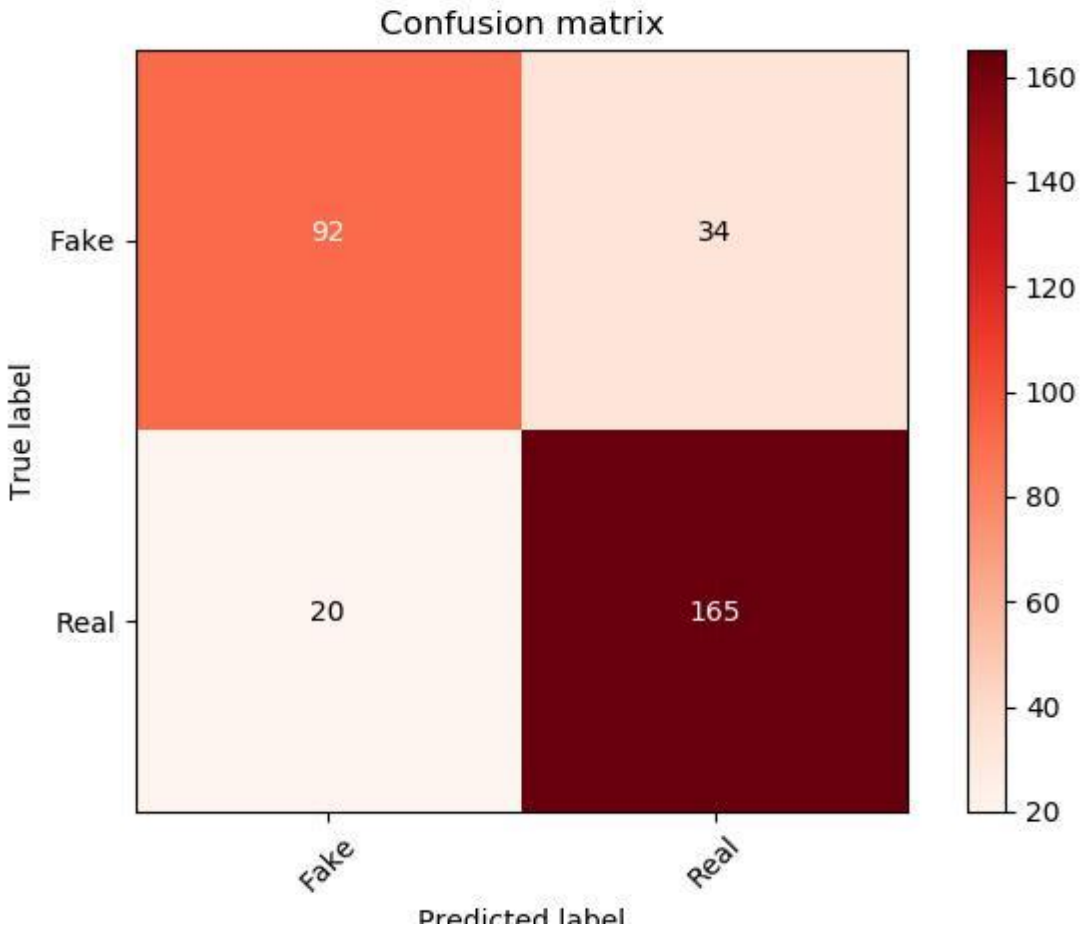


Figure 20

After tuning, it can be observed that there is an improvisation on the first part of the matrix (Fake label predictions) but at the same time, the predictions on the second part of the matrix (Real label predictions) are worse. Overall, the error was slightly decreased.

Afterwards, the same process with regards to TF-IDF model is following.

1st PART: TF-IDF model

Model	Accuracy	Precision	Recall	F1-score	10-fold Cross val-idation

TF-IDF	0.859	0.862	0.859	0.856	79.32%

Confusion matrix results are presented below:

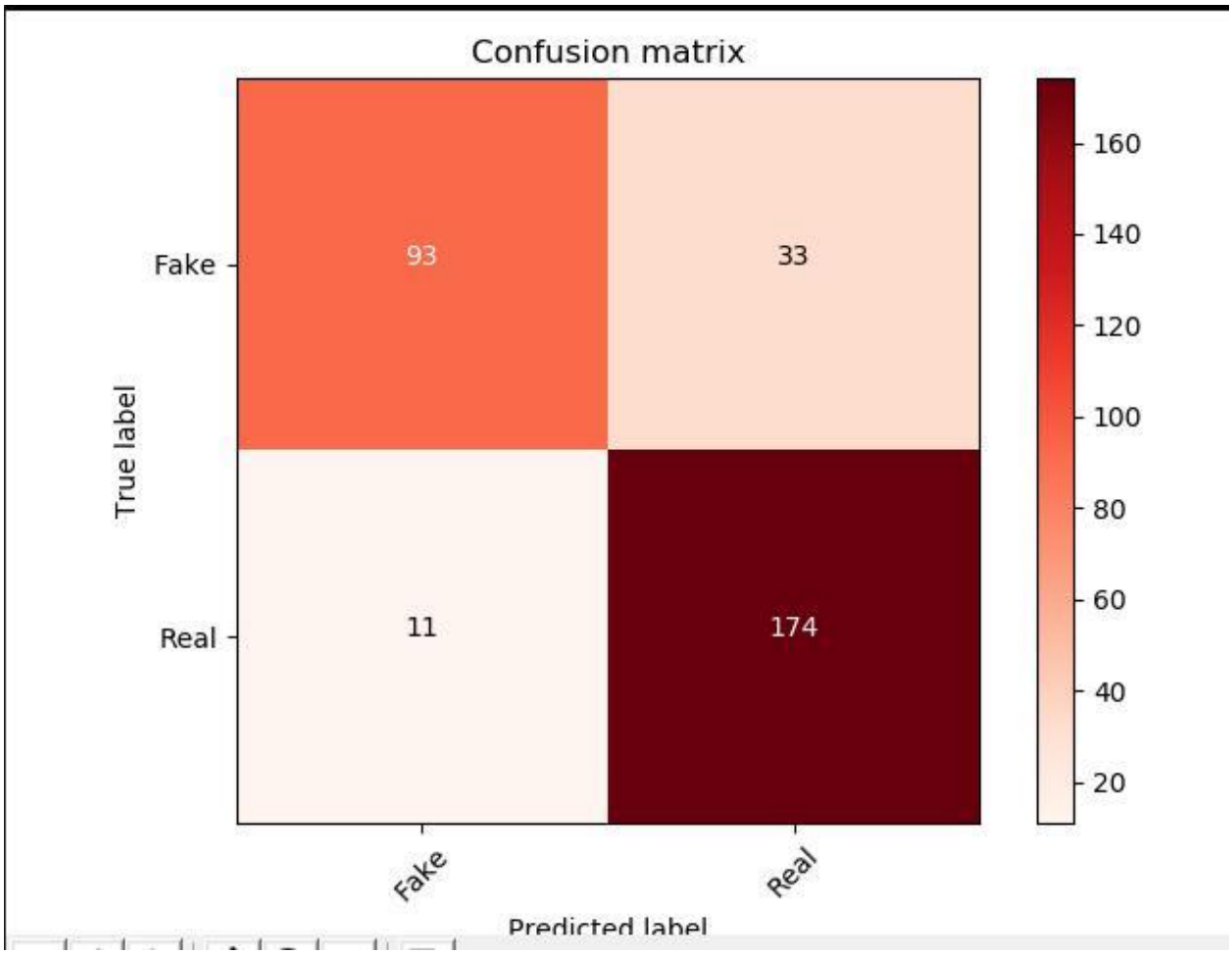


Figure 21

The Figure 21 reveals that 93 out of the total 126 fake records were correctly predicted as fake, while there were 33 errors. Also, 174 out of the total 185 real records were correctly predicted as real, while there were 11 errors. Without the tuning process, it seems that TF-IDF model beats Bag of Words model with Logistic Regression classifier.

2nd PART: TF-IDF model

C	Accuracy	Precision	Recall	F1-score	Cross-Validation
0.01	0.596	0.354	0.595	0.444	55.48%

0.5	0.823	0.836	0.823	0.816	75.20%
1	0.859	0.862	0.859	0.856	79.32%
1.5	0.868	0.870	0.868	0.867	80.76%
2.0	0.871	0.872	0.871	0.870	81.72%
2.5	0.873	0.876	0.873	0.870	82.67%
3	**0.875**	**0.879**	**0.875**	**0.870**	**83.30%**

In comparison with Bag of Words model, the results here are a slightly different. More specifically, the value of 3 for C parameter led the model into chaotic predictions. It is significant to mention that the 10-fold cross validation of the 1st part for the applied classifier on TF-IDF model is lower than the individual evaluation metrics. That outcome verifies the above suspicions. Consequently, the model failed under the pressure of the value with greatest regularization strength. Moreover, it is noticeable that as the value of C increases it greatly improves the performance of the classifier through all the stages. As a result, the best value of C parameter for this case is 3. The relevant confusion matrix is following:

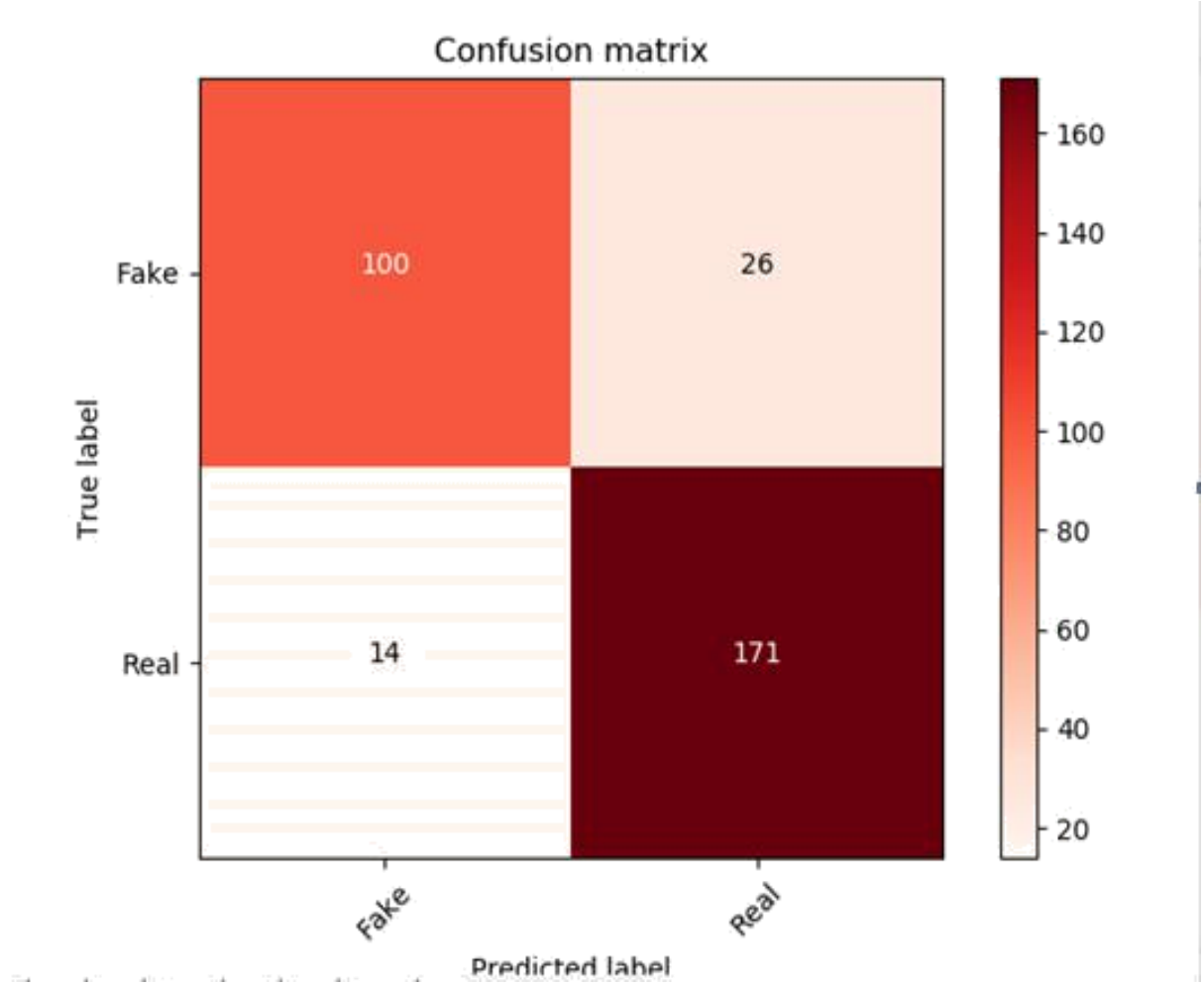


Figure 22

The predictions regarding the first part of the confusion were greatly enhanced and the error was decreased. But, the inaccuracy on the second part of matrix was slightly increased. Generally, the miscalculations were diminished.

After all the analysis, it appears that TF-IDF model surpassed Bag of Words model regarding Logistic Regression`s case.

Model	C	Cross Validation Accuracy	Precision	Recall	F1-score
BOW	2.0	82.03%	0.826	0.826	0.824
TF-IDF	3.0	83.30%	0.879	0.875	0.870

5.4 AdaBoost Classifier

The fourth algorithm in the study is the AdaBoost Classifier. The classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. Hence, it is quite alluring to check the results and how the classifier reacts regarding the distribution of the weights. The interested parameter is called n_estimators according to sklearn library which was imported in Python. In specific, this parameter refers to the maximum number of estimators at which boosting is terminated. When the perfect fit is accomplished, the learning procedure is interrupted early. Thence, the experiments are divided into two parts. The first part is concerning the results of the chosen algorithm`s performance without any parameters included. The second part involves the examination of the outcomes regarding the addition of n_estimators parameter into the classifier`s execution. It is significant to comment that the AdaBoost algorithm was tested in both models (BOW & TF-IDF).

1st PART: Bag of Words model

Model	Accuracy	Precision	Recall	F1-score	10-fold Cross val-idation
BOW	0.852	0.851	0.852	0.851	82.34%

In addition to those evaluation metrics, confusion matrix showed this outcome:

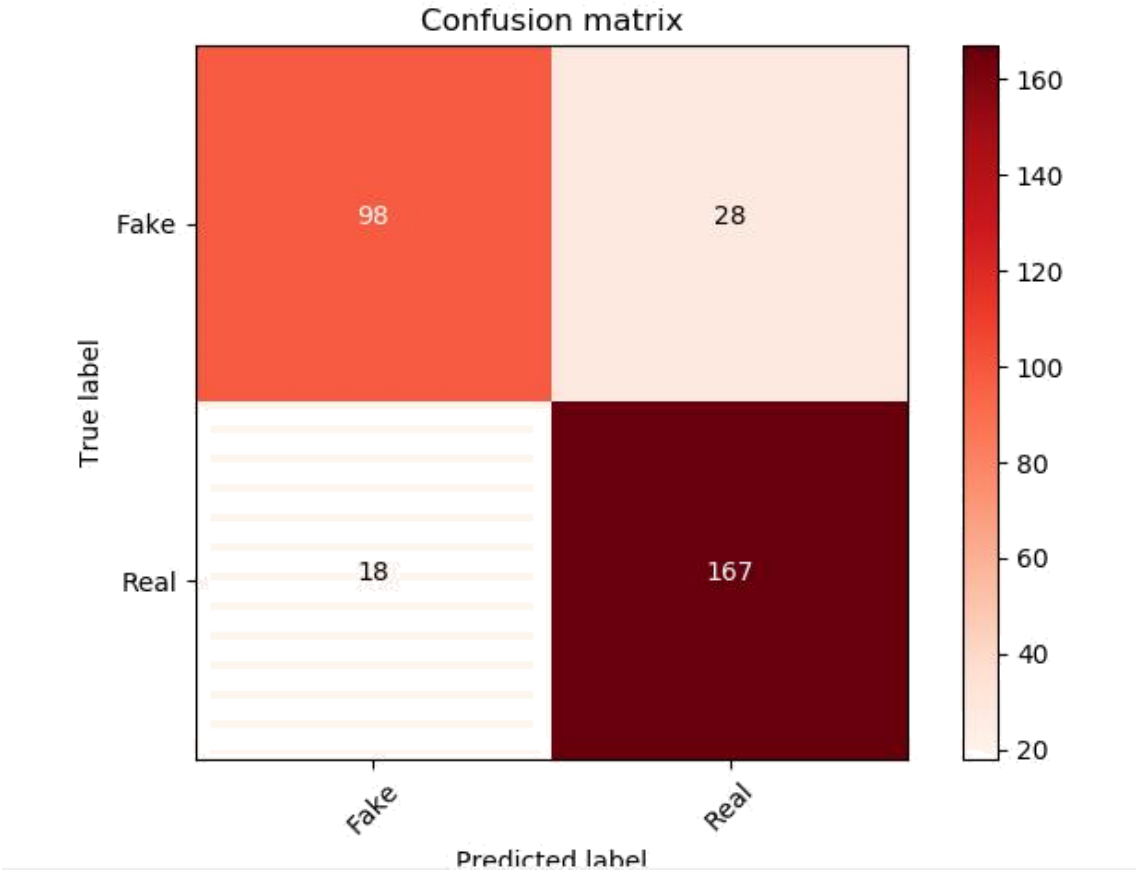


Figure 23

The Figure 23 reveals that 98 out of the total 126 fake records were accurately predicted as fake, while there were 28 errors. Concurrently, 167 out of the total 185 real records were also properly predicted as real, while there were 18 errors.

2nd PART: Bag of Words model

Four values of the parameter n_estimators were test and the code proceeded to this output:

n_estimators	Accuracy	Precision	Recall	F1-score	10-fold Cross validation
100	0.842	0.842	0.842	0.841	81.39%
200	**0.856**	**0.856**	**0.856**	**0.856**	**83.95%**
300	0.836	0.838	0.836	0.836	82.52%
400	0.797	0.811	0.797	0.799	77.11%

As it can be observed, when n_estimators takes the value of 200 the classifier hits the maximum accuracy which is marginally better than the case without any parameters (1st Part). All the other possible input values have a negative impact to the accuracy with respect to the previous state.

Similar procedure for the TF-IDF model will be presented next.

1st PART: TF-IDF model

Model	Accuracy	Precision	Recall	F1-score	10-fold Cross val-idation
TF-IDF	0.823	0.822	0.823	0.822	82.34%

. Confusion matrix of this experiment is following:

z

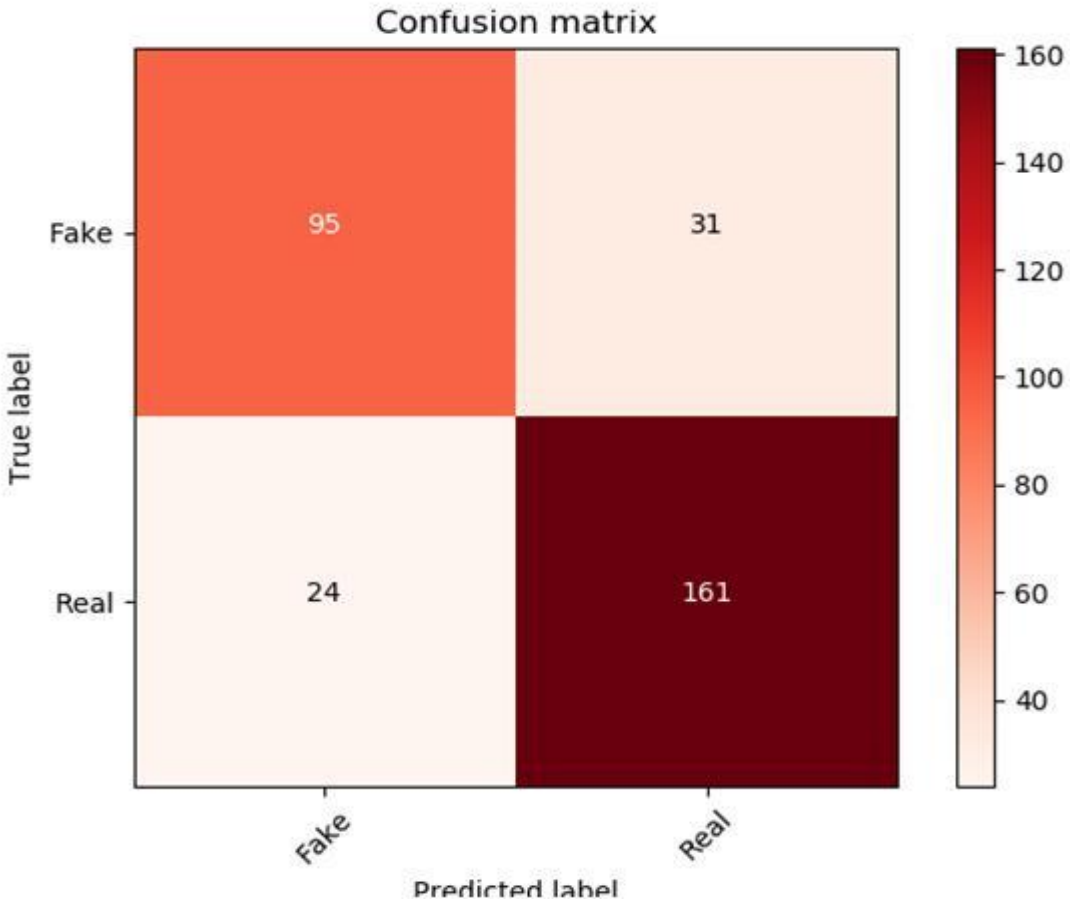


Figure 24

The Figure 24 shows that 95 out of the total 126 fake records were correctly predicted as fake, while there were 31 errors. Simultaneously, 161 out of the total 185 real records were also properly predicted as real, while there were 24 miscalculations.

2nd PART: TF-IDF model

n_estimators	Accuracy	Precision	Recall	F1-score	10-fold Cross validation
100	**0.814**	**0.813**	**0.813**	**0.813**	**85.05%**
200	0.791	0.798	0.791	0.792	77.58%
300	0.765	0.781	0.765	0.767	78.54%
400	0.772	0.787	0.772	0.774	76.79%

Undoubtedly, the experiment results are optimal when n_estimators = 100 for this case. Overall, after that point it is a fact that when n_estimators increases the performance of the classifier is decreasing.

According to analysis part, it is a fact that overall AdaBoost classifier has a greater performance in Tf-Idf model in comparison with Bag of word`s case. The ideal performances for both models according to this algorithm are presented below:

Model	n_estimators	Accuracy	Precision	Recall	F1-score	10-fold Cross validation
BOW	200	0.856	0.856	0.856	0.856	83.95%
TF-IDF	100	0.814	0.813	0.813	0.813	85.05%

In conclusion, it is interesting to notice an output which is based on the learning_rate parameter. This parameter shrinks the contribution of each classifier. There is a trade-off between learning_rate and n_estimators. By inserting the value of 3 to the learning_rate parameter which is an extreme case, the confusion matrixes for BOW & TF-IDF are the following:

- Bag of Words

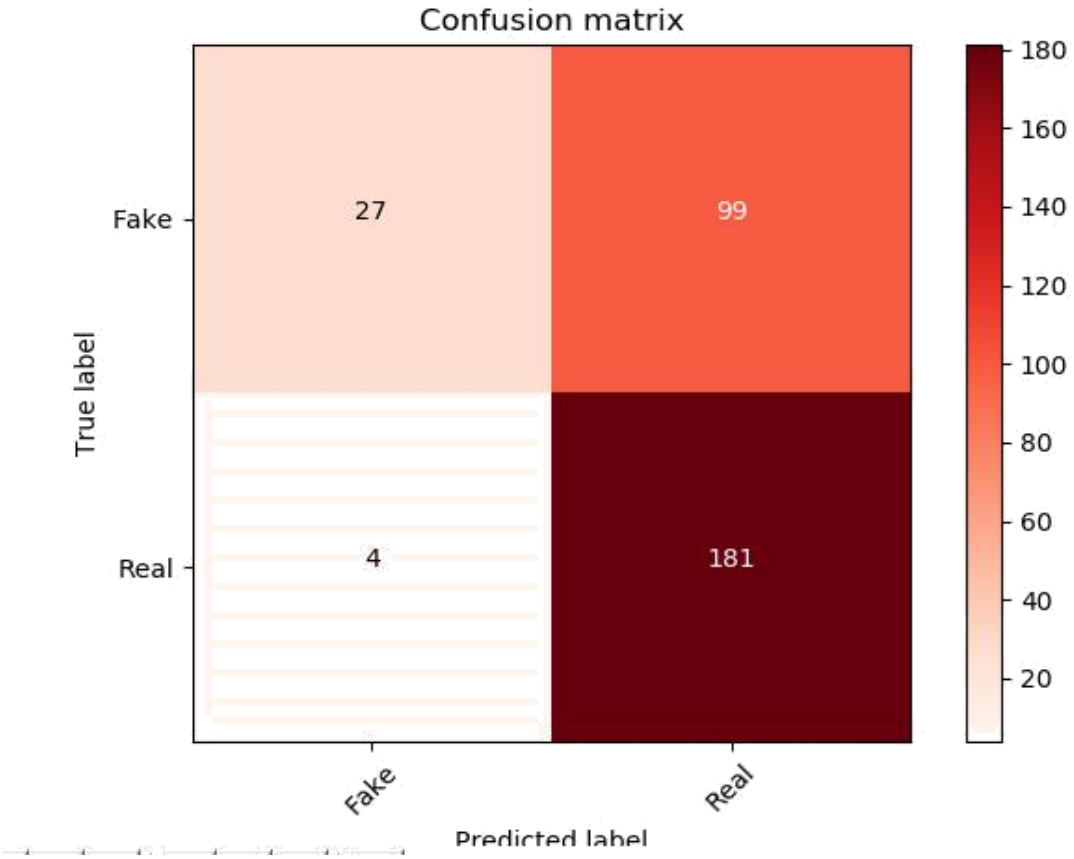


Figure 25

- TF-IDF

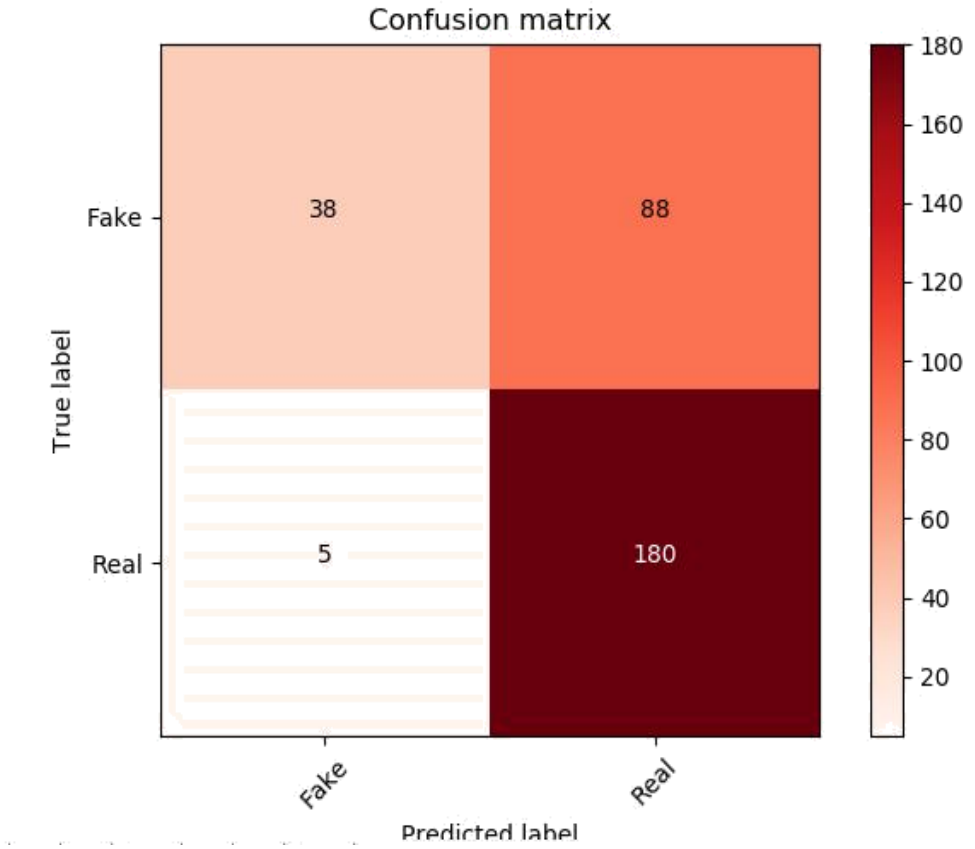


Figure 26

In both cases, the predictions regarding the first part of the matrix are ruined but surprisingly, the second of part of the matrix is significantly improved. This behavior can be translated that as follows: as the number of estimators increases, the predictions approach the perfection with respect to the second part of the confusion matrix. Nevertheless, the overall accuracy of the classifier was dropped.

5.5 MLP Classifier

The last tested classifier for this study is Multi-Layer Percepton which is a supervised learning algorithm. MLP can learn a non-linear function approximator and it is different from logistic regression since it is between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Essentially, that was the motivation for including it to the list of the tested algorithms regarding the study. MLP classifier implements various parameters according to the sklearn library. The first sig-

nificant parameter is called alpha, which helps in avoiding overfitting by penalizing weights. As a result, the parameter is related to the regularization phenomenon. An example of different input values of alpha is following alongside with three model`s reactions which were generated in sklearn`s documentation.

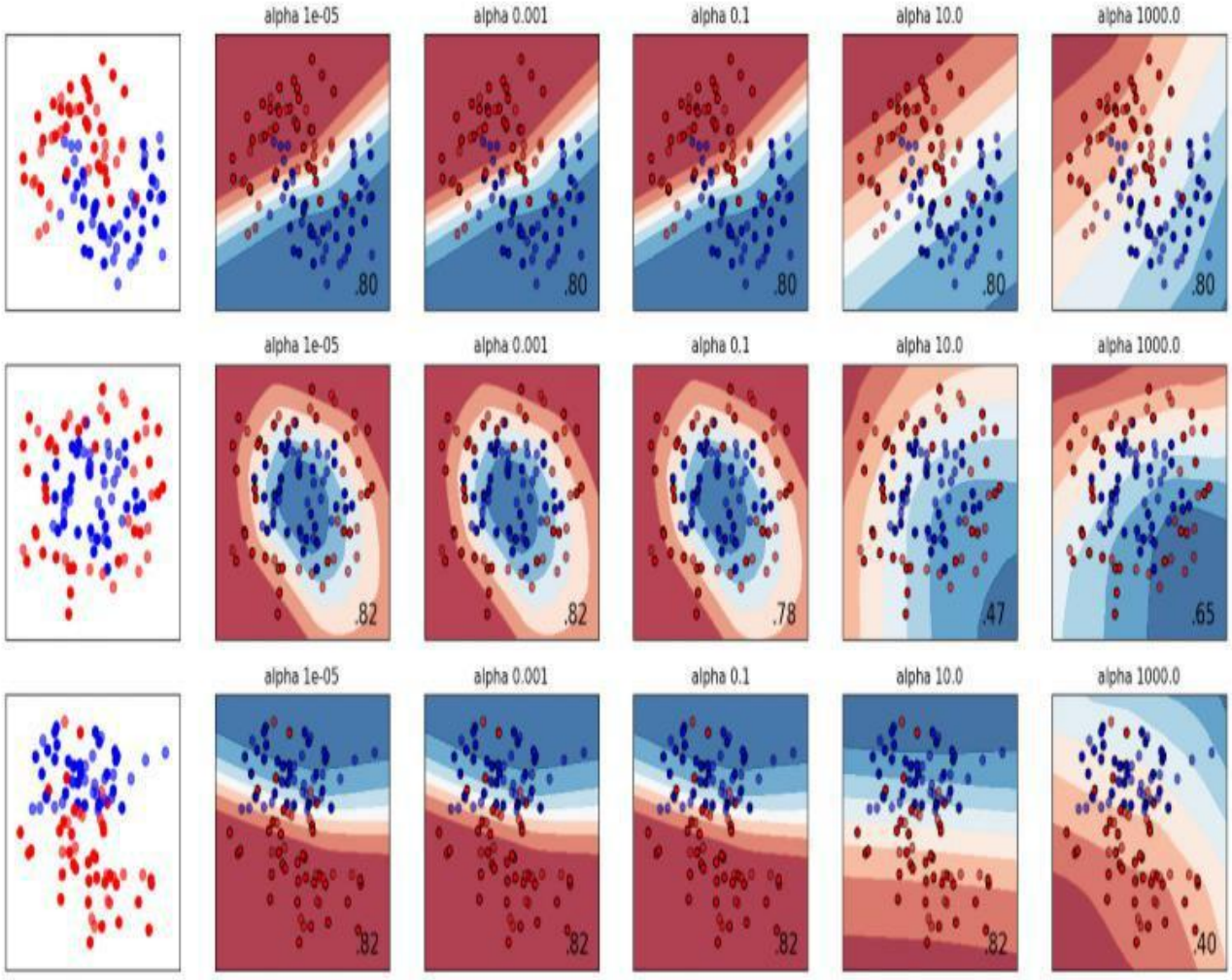


Figure 27: different values of alpha parameter

Unquestionably, as alpha parameter increases, the decision function of the classifier is adjusted to the model`s behavior. Consequently, cross validation method thrives of importance for the experiments due to possible overfitting situation. Some of these values will be tried out and examined in this study. Likewise, in previous algorithm cases, the experiments are split into two parts. The first section is about results with respect to non-parametric execution of the specific classifier and the second section is described

by experiments with parameters added while the major evaluation metric will be the 10-fold cross validation.

1st PART: Bag of words model

Model	Accuracy	Precision	Recall	F1-score
BOW	0.804	0.808	0.804	0.799

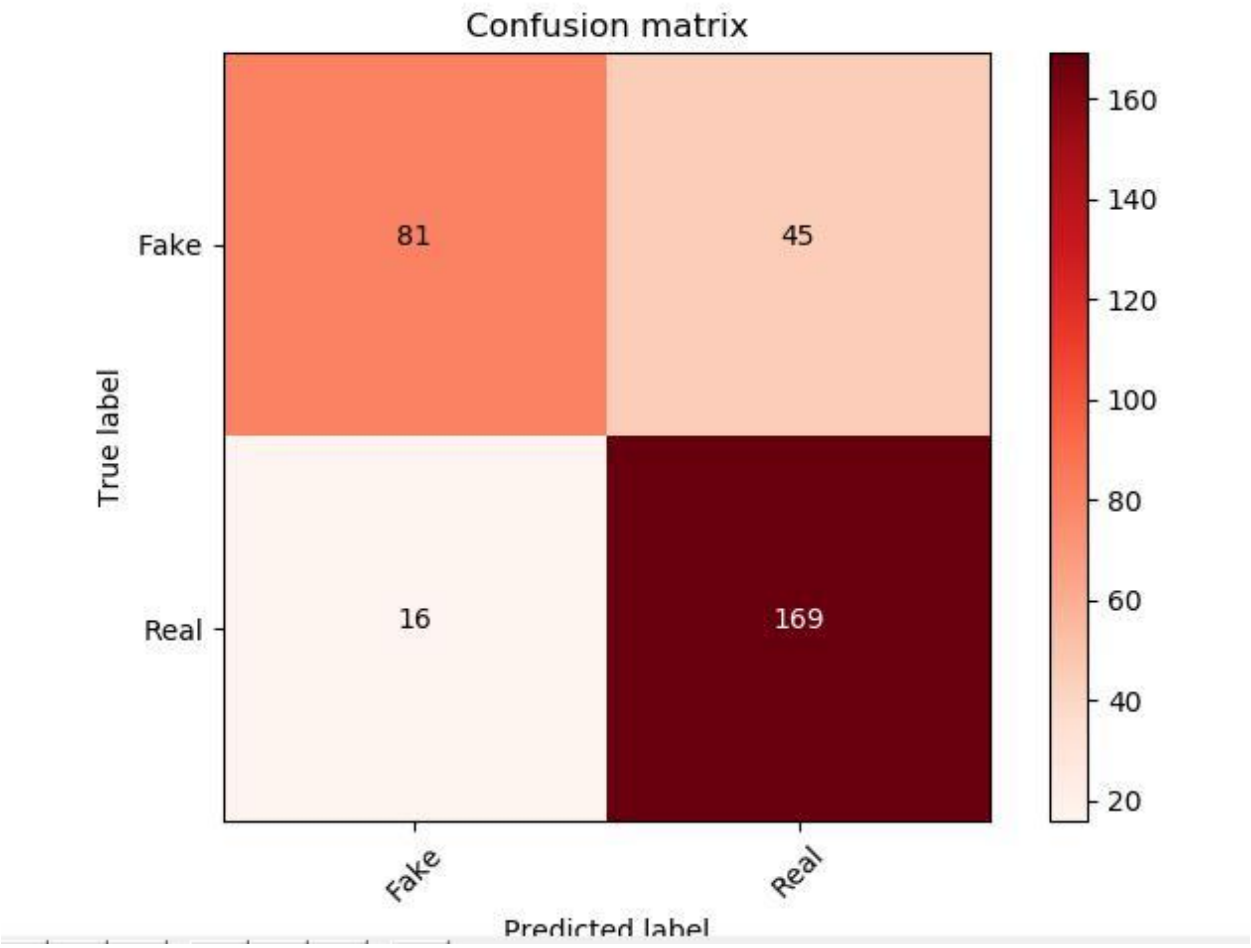


Figure 28

The Figure 28 shows that 81 out of the total 126 fake records were accurately predicted as fake, while there were 45 errors. Also, 169 out of the total 185 real records were also properly predicted as real, while there were 16 wrong predictions. The performance of the classifier on the second part of the matrix is exceeding with regards to the results of the first part.

2nd PART: Bag of words model

Alpha	Accuracy	Precision	Recall	F1-score	10 fold Cross-validation
1e-05	0.826	0.830	0.826	0.823	81.55%
0.001	0.833	0.834	0.833	0.830	80.28%
0.1	0.826	0.826	0.826	0.825	82.04%
0.2	**0.814**	**0.813**	**0.814**	**0.811**	**82.35%**
0.3	0.807	0.806	0.807	0.805	82.20%
10.0	0.814	0.819	0.814	0.808	81.24%

As it was mentioned before, the crucial evaluation metric for this case is the outcome of Cross Validation`s process. This goal is accomplished when a parameter takes the value of 0.2 but overall the results are not that different. The fluctuation according to the specific evaluation metric is approaching to be smooth as the alpha parameter increases.

1st PART: TF-IDF model

Model	Accuracy	Precision	Recall	F1-score
TF-IDF	0.846	0.857	0.846	0.847

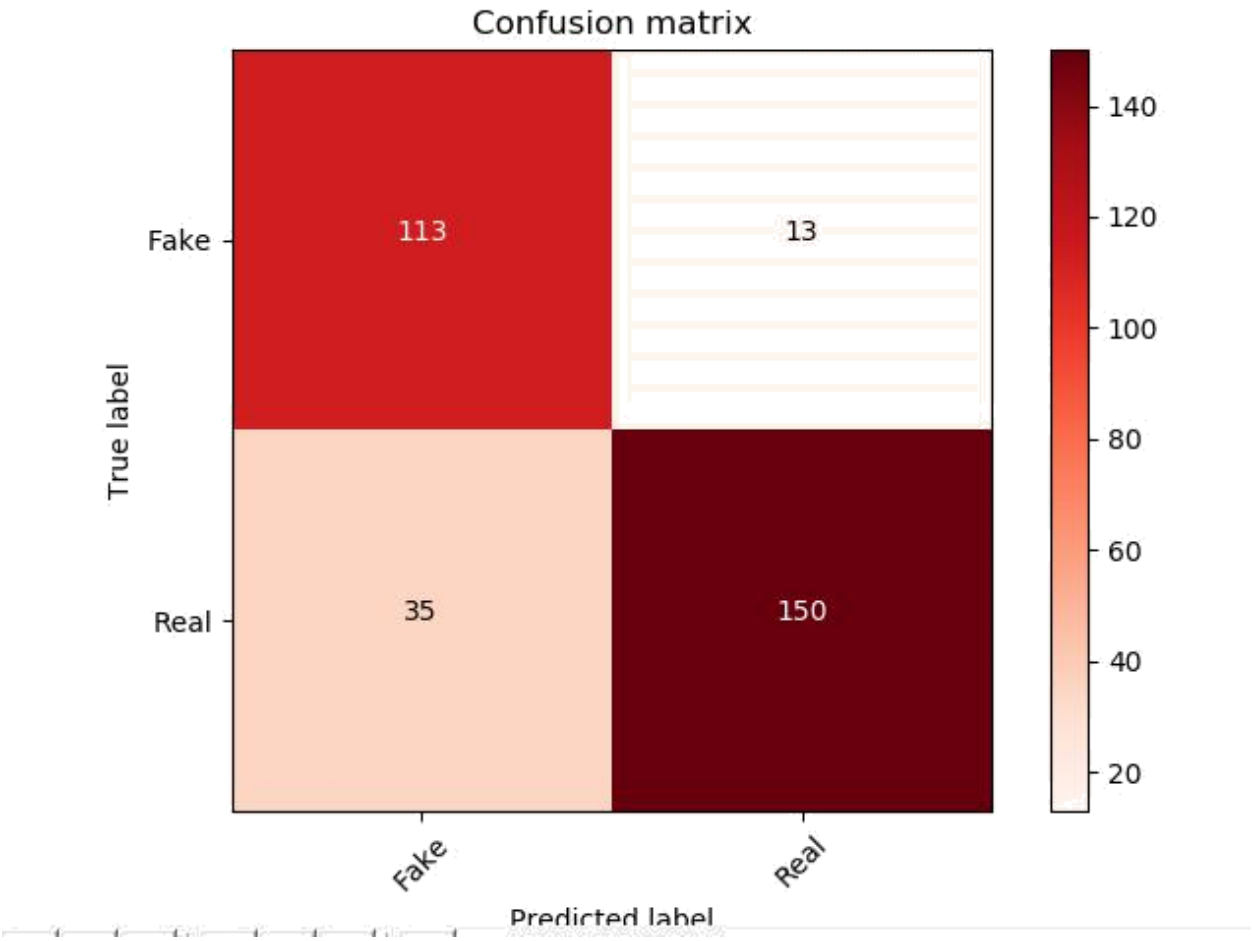


Figure 29

The presented confusion matrix displays that 113 out of the total 126 fake records were accurately predicted as fake, while there were 13 inaccurate predictions. Besides that, 150 out of the total 185 real records were also properly predicted as real, while there were 35 missteps. Interestingly, the performance of the classifier on the first part of the matrix is superior with respect to the results of the second`s part which is the opposite outcome of BOW model`s scenario.

2nd PART: TF-IDF model

Alpha	Accuracy	Precision	Recall	F1-score	10-fold Cross val-idation
1e-05	0.842	0.853	0.842	0.844	83.30%
0.001	0.842	0.853	0.842	0.844	83.62%

0.1	**0.859**	**0.859**	**0.859**	**0.859**	**85.84%**
0.2	0.855	0.855	0.855	0.855	84.74%
0.3	0.855	0.856	0.855	0.855	85.55%
10.0	0.595	0.354	0.595	0.444	55.48%

In general, the 10-fold cross validation scores for TF-IDF model are preferable over BOW`s model outputs with MLP classifier. The best score is achieved with a=0.1 while it is significant to be mentioned that values of $a \geq 10$ lead the classifier into a very poor performance. This value signified that from this point the bad classifications begin and as a result, greater values than 10 were not tested.

To sum up, the optimal choice according the alpha parameter value selection for the MLP classifier for both models is:

Model	alpha	Accuracy	Precision	Recall	F1-score	10-fold Cross val-idation
BOW	0.2	0.814	0.813	0.814	0.811	82.35%
TF-IDF	0.1	0.859	0.859	0.859	0.859	85.84%

6 Evaluation and Discussion

After discovering the optimal values for each parameter on all classifiers, cross-validation technique will be applied to the rest evaluation metrics in order to compare all the possible models. The following table contains all the mentioned outcomes with 10 folds cross-validation:

Model	Classifier	Parame-ter	Precision	Recall	F-score	Accuracy	Time to build the model
BOW	Multinomi-al	a = 0.2	83.90%	82.30%	82.40%	82.40%	FAST
TF-IDF		a = 0.1	84.70%	82.60%	82.80%	82.80%	FAST
BOW	Passive-Aggressive	n_iter = 100	81.68%	81.08%	80.84%	81.71%	FAST
TF-IDF		n_iter = 1000	85.58%	85.53%	85.47%	85.53%	FAST
BOW	AdaBoost	n_estimat ors = 200	84.71%	83.95%	83.67%	83.95%	MODER-ATE
TF-IDF		n_estimat ors = 100	85.46%	85.06%	84.93%	85.05%	MODE-TRATE
BOW	Logistic Regression	C = 2.0	82.96%	82.03%	81.65%	82.03%	FAST
TF-IDF		C = 3.0	84.63%	83.31%	82.89%	83.30%	FAST
BOW	MLP	a = 0.2	83.41%	82.52%	81.99%	82.35%	SLOW
TF-IDF		a = 0.1	85.88%	85.57%	85.69%	85.84%	SLOW

The table shows the entire classifier`s results with the corresponding evaluation metrics. To start with, it is significant to be mentioned that the major technique for evaluating our models is the accuracy of 10-fold cross-validation. According to that procedure, the classifiers predicted the test set with different training sets (out-of-sample testing). More specifically, the number of altered training sets is 10, so it calculates the average accuracy of all the 10 folds process. As a result, with regards to the mentioned machine learning technique, insights will arise about how well the models perform on unseen data. Empirical knowledge about the generalization rule of each model is obtained. The best model with respect to the accuracy metric is MLP with a score of 85.84% which is followed by Passive-Aggressive classifier with 85.53% score. Both classifiers have high performance, while they achieved those similar evaluation scores on Tf-Idf model. Additionally, the precision, recall and f1-scores results are very similar for both models. Their differences are almost negligible. Therefore, the mandatory time for each classifier to build the model was considered. The needed time for Passive-Aggressive classifier to execute the procedure was very low in contrast to MLP. This constitutes a disadvantage of MLP and it is significant to be stated. Most importantly, in general, the necessary time for the MLP classification`s finalization was the most immense out of all required times for the rest algorithms. This aftermath was expected because it belongs to the Neural Network family with a lot of computations involved.

Classifiers	BOW	TF-IDF
Multinomial		✓
Passive-Aggressive		✓
Logistic Regression		✓
AdaBoost		✓
MLP		✓

Overall, it is clear that all classifiers produced better results in TF-IDF model than Bag of words model. This outcome is related to the fact that TF-IDF model takes into account the occurrence of a word in the entire corpus and not in a single document. As a result, the same words in all documents are connected and affect the decision and rules of each classifier differently, while in Bag of words the words are treated individually in each document.

7 Ranking Model Approach

The innovation of this study is about a new model which can label a new input data which is given by the user as Fake or Real. This model is called Ranking Model Approach. In detail, this approach is related with an index, which is able to inspect each unique observed word with respect to its contribution for being fake or real. Some words are more likely to be seen in fake records or in real records. This indicator is called Ranking index. Positive values of this index signal the words to be labeled as real contributors, while negative values mark the words as fake contributors. A new text-input by the user will be rated according to the sum of each word`s ranking index value. More specifically, if the input is composed of five words $w_1,w_2,w_3,w_4,w_5$, the score $s$ will be the aggregation of the numerical values of each word`s Ranking index r_1,r_2,r_3,r_4,r_5 as it follows below:

$$S = W_1 (r_1) + W_2 (r_2) + W_3 (r_3) + W_4 (r_4) + W_5 (r_5)$$

- $S > 0$ means that the text will be labeled as Real
- $S < 0$ means that the text will be labeled as Fake
- $S = 0$ means that the text has the same possibilities to be Fake or Real. Nevertheless, this is an extreme scenario.

Additionally, it is significant to be mentioned that higher numerical values of the S score are guaranteeing greater insurance about the final taken decision by the Ranking Model.

The catalyst factor for the classification of a new unseen data is the ranking index. This indicator is calculated according to two steps. Firstly, a vectorizer is needed which could arise from Bag of Word`s model or Tf-Idf`s model. The vectorizer is able to convert the collection of documents into a matric of token counts. At evaluation and discussion part, Tf-Idf model accomplished better performance and came up to be more efficient than Bag of Words model. Consequently, the Tf-idf vectorizer will be used as the first argument for the creation of the mentioned indicator. Secondly, the other argument is achieved through the ideal algorithm of our classification analysis part. Hence, it is

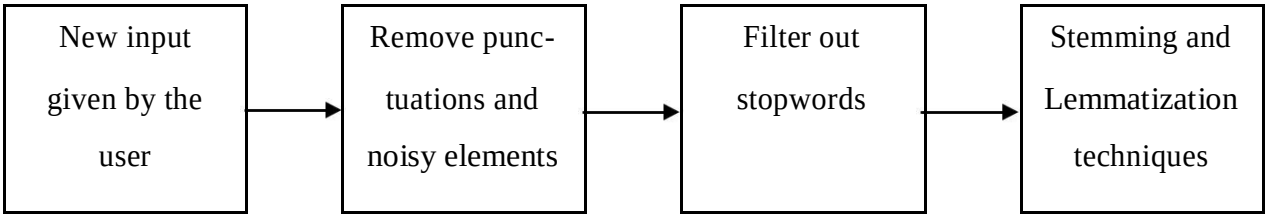
proved that Passive-Aggressive algorithm with n_iter = 1000 as its tuned parameter, is the best choice. Each classifier produces different results with regards to the calculation of the ranking index, but the optimal outcome for the Ranking Model is reached with the best classifier`s performance. In our study, the following picture inspects the words with the top ten ranking index`s values for real and fake class:

Fake -2.085400730691558 clinton	Real 3.220986314874112 mr
Fake -2.015230024004718 hillary	Real 2.674248937472122 said
Fake -1.8575357095261096 world	Real 1.801357713640927 ms
Fake -1.8486963184292011 october	Real 1.627124845408249 breitbart
Fake -1.750386002870344 election	Real 1.610861971345862 tuesday
Fake -1.4228930832904927 america	Real 1.5196468003341932 president
Fake -1.4018904876218052 source	Real 1.2395859217472767 prime
Fake -1.3316530063569614 home	Real 1.201496752030939 saturday
Fake -1.3309332298599108 david	Real 1.087589198383544 despite
Fake -1.3247382011562001 water	Real 1.0643473897184095 european

Figure 30: Ranking index for real and fake words

According to the above picture, words with political content seem to achieve the greatest contribution regarding the fake class. At the same time, words like mr, ms, president and prime contribute more with respect to the real class. Undoubtedly, the mentioned words which are characterized by a formally formatted style are mostly observed to be in real data records.

The text-input given by the user will be pre-processed similarly with the rest dataset. More specifically, the final form of the new data will be achieved through these pre-processing stages:



The flow chart of the Ranking model according to its decision steps is presented below:

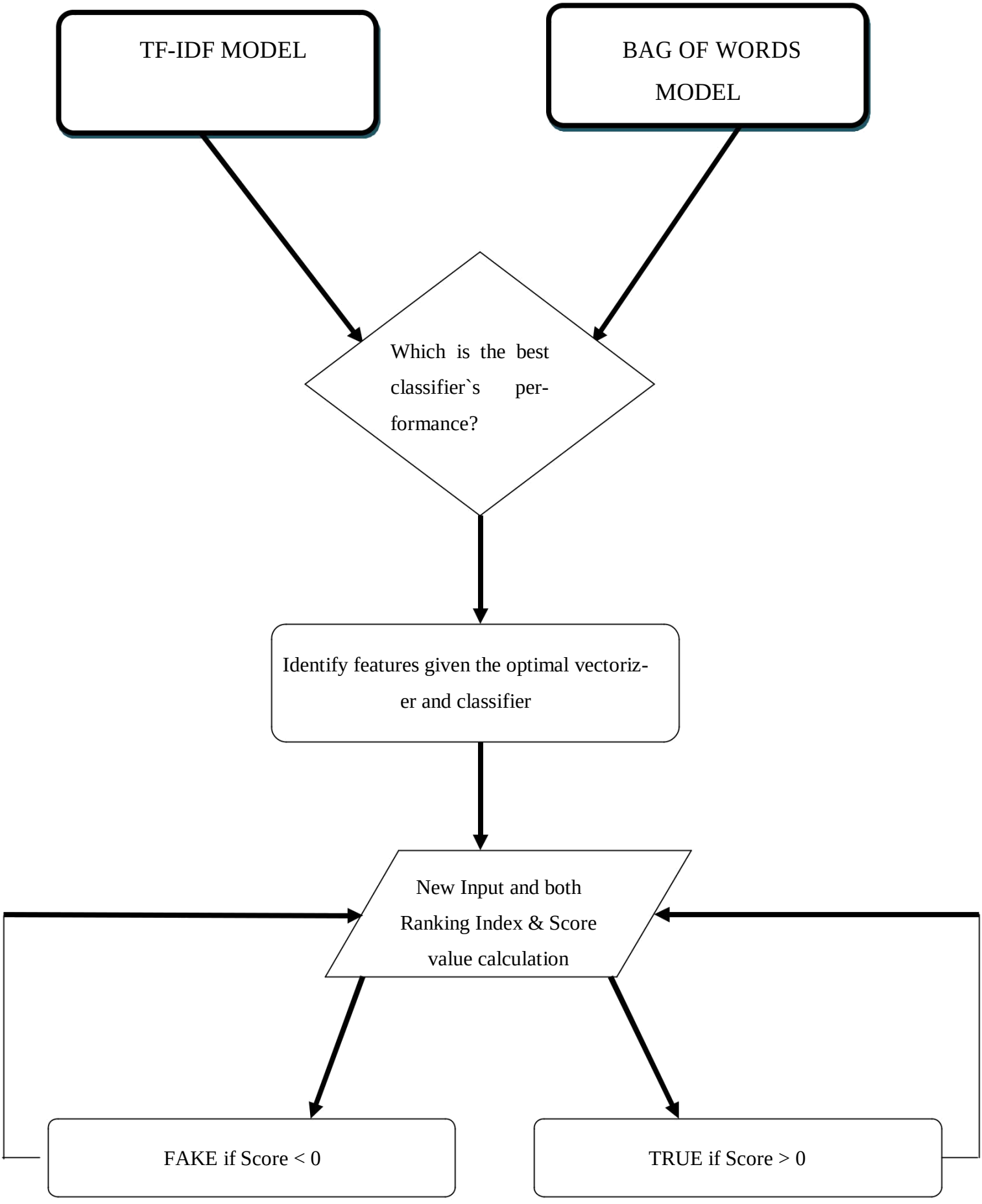
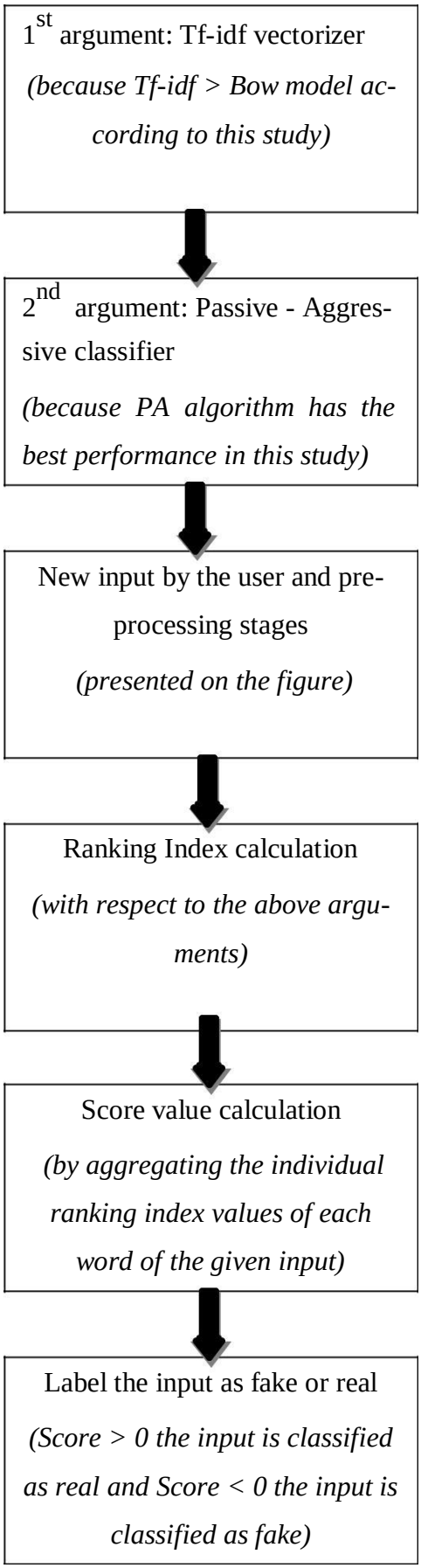


Figure 31: Flow chart of Ranking Model

In our study, by taking into account the steps presented on the figure 31, the Ranking model will behave as it is shown to the following stages:



Finally, it is significant to be mentioned that the addition of new data to dataset will affect all the results. As the dataset increases, the dictionary which contains each unique observed word will be enlarged and consequently the outcomes of the analysis will alter. The ranking model is dynamic regarding to possible changes in the dataset in order to increase the dictionary of the words.

8 Conclusions and future work

The aim of this dissertation was to discover an efficient way to label an article as fake or real with respect to machine learning concepts and decisions from classification algorithms. In detail, this outcome was achieved through four analysis parts.

The first part is referring to the data extraction, preparation and model construction. The data extraction was very challenging because the web choices are limited, not that qualitative and sometimes unstructured. As a result, web scrapping on article websites was a method to surpass and overcome this problem. Afterwards, the finalized dataset was pre-processed according to well-known and useful text-oriented techniques, such as stemming and lemmatization. Finally, the text-data needed to be converted into meaningful numerical values, so the classifiers can fully-operate and execute their algorithmic parts. Two models were selected for that goal, Bag of words model and Tf-Idf model. The crucial part to be mentioned is that those models have a major difference. The Tf-idf model considers the occurrence of a word in the entire corpus, while the bag of words model in a single document. It is notable to investigate the performance and reaction from well-known classification algorithms.

The second section is related to the experimental results. In detail, five algorithms were tested and their behavior was investigated. The selected algorithms are Multinomial Naïve Bayes, Passive-Aggressive, Logistic Regression, AdaBoost and MLP. Different parameter values for each classifier were used in order to find the best possible outcomes.

The third part of the thesis is about the evaluation and discussion of the performance for each classifier. The used evaluation metrics were precision, recall, f1-score, confusion matrix and the necessary time to build the model, while cross-validation technique with 10 folds was applied in order to check the regularization rule for every model. It ended that the classifiers with the most effective performance are Passive-Aggressive

and MLP. However, the Passive-Aggressive algorithm was ideal with respect to the mandatory operational time.

The final part is related to the innovation of this study. More specifically, the Ranking Model approach is introduced. According to that model, a new input data given by the user can be labeled regarding our two classes. In order to achieve that, a ranking index had to be calculated, which needs two arguments. The first argument refers to the ideal vectorizer for the text-conversion into numerical values. It was revealed that all classifiers performed better in Tf-idf model than in the Bag of Words model. As a result, tf-idf vectorizer was selected. The second argument is about the optimal classifier of the study. As shown in the previous section, passive-aggressive achieved the best performance, so it was preferred amongst the rest algorithms. Hence, the calculation of the ranking index was computed, and the model was capable to classify new unseen data efficiently. By doing that, the scope of the study was accomplished.

8.1 Future work

Several different adaptations, tests, and experiments have been left for future work due to scope limitations. Future work concerns deeper analysis of particular mechanisms, new proposals to try different methods, etc. There are some ideas that I would have liked to try during the entire procedure.

1. The enlargement of the dataset with regards to the web scrapping method. It would be interesting to see how the Ranking Model reacts on a bigger dictionary of words, because all the results would be different.
2. Additional classifiers in order to discover more experimental results and observe the performance of the new classifiers. It may be that the top classifier of this study, passive - aggressive, is outperformed by the new algorithms. As a result, the Ranking model would be more efficient, due to the alteration of one of its arguments.
3. The ranking model could contain more than two classes. For example, there could be four classes: fake: high probability, fake: lower probability, real: high probability, real: lower probability. More specifically, the score indicator is able to provide numerical values, which can be interpreted with confidence boundaries such us the above. The interesting part is to discover the point (nu-

merical values of Score variable), where the classes are separated efficiently. The outcome of this procedure will be the production of an accurate decision function for the Ranking model.

Bibliography

[1] Kai Shu, A. S. (2017). Fake News Detection on Social Media: A Data Mining Perspective. Arizona, USA.

[2] Edson C. Tandoc Jr., Z. W. (2018). Defining "Fake news". *Digital Journalism*, 137-153.

[3] Gentzkow, H. A. (2017). Social Media and Fake News in the 2016. *Journal of Economic Perspectives*, 211–236.

[4] Condren, C. D. (2008). Defining parody and satire: Australian copyright law and its new exception: Part 2 - Advancing ordinary definitions. 401-421.

[5] Gatov, V. (2018, January 18). Propaganda in a Fake News World. Russia. http://intersectionproject.eu/article/russia-world/propaganda-fake-news-world

[6] Lewis, D. D. (n.d.). Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval.

[7] Koby Crammer, O. D.-S. (2006). Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 551-585.

[8] Brownlee, J. (2016, April 25). Boosting and AdaBoost for Machine Learning.

[9] *Statistics Solutions*. (n.d.). Statistics Solutions: https://www.statisticssolutions.com/what-is-logistic-regression/

[10] Brownlee, J. (2016, May 17). Crash Course on Multi-Layer Perceptron Neural Networks.

[11] Joshi, R. (2016, September 9). Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures.

[12] Brownlee, J. (2016, March 21). Overfitting and Underfitting With Machine Learning Algorithms.

[13] Niall J. Conroy, V. L. (n.d.). Automatic Deception Detection: Methods for Finding Fake News.

[14] Benjamin Riedel, I. A. (2018, May 21). A simple but tough-to-beat baseline for the Fake News Challenge stance detection task.

[15] *British Council*. (n.d.). http://learnenglishteens.britishcouncil.org/skills/reading/advanced-c1-reading/rise-fake-news

[16] David M. J. Lazer, M. A. (2018, March 9). The science of fake news.

[17] *Technopedia*. (n.d.). https://www.techopedia.com/definition/14650/data-preprocessing

[18] Risueno, T. (2018, February 28). *Bitext*. retrieved from https://blog.bitext.com/what-is-the-difference-between-stemming-and-lemmatization/

[19] *Analytics Vidhya*. (2017, June 4). retrieved from https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

[20] Ramos, J. (n.d.). Using TF-IDF to Determine Word Relevance in Document Queries.

[21] Andrew McCallum, K. N. (1998). A comparison of event models for Naive Bayes text classification.

9 Appendix

9.1 Web scrapping code

```
from urllib.request import urlopen as uReq
from bs4 import BeautifulSoup as soup

########################################## NYtimes

my_url = 'https://www.nytimes.com/section/world'

#opening connections and grabbing the
page uClient = uReq(my_url) page_html =
uClient.read()
uClient.close()

#html parsing
page_soup = soup(page_html,"html.parser")

#grabs each article
articles = page_soup.find_all("div",{"class": "story-meta"})

filename = "RealnewsNY10.csv"
f = open(filename,"w")

Label = 'Real'
headers = "Title, Summary, Label\n"

f.write(headers)

print("NYtimes worlds news ---------------------------- ")

for article in articles:


    title_article = article.find_all("h2",{"class":"headline"})
    title = title_article[0].text.replace(","," ").strip()

    summary_article = article.find_all("p",{"class":"summary"})
    summary = summary_article[0].text.replace(","," ")

    f.write(title + "," + summary + "," + Label + "\n")

f.close()

########################################## Reuters

my_url2 = 'https://www.reuters.com/news/world'

#opening connections and grabbing the
page uClient = uReq(my_url2) page_html =
uClient.read()
uClient.close()
```

```
#html parsing
page_soup = soup(page_html,"html.parser")

#grabs each article
articles = page_soup.find_all("div",{"class":
"ImageStoryTem-plate_image-story-container"})

filename = "RealnewsReuters10.csv"
f = open(filename,"w")

Label = 'Real'
headers = "Title, Summary, Label\n"

f.write(headers)

print("Reuters worlds news ----------------------------- ")

for article in articles:


    title_article = arti-
cle.find_all("h2",{"class":"FeedItemHeadline_headline"})
    title = title_article[0].text.replace(","," ").strip()

    summary_article = arti-
cle.find_all("p",{"class":"FeedItemLede_lede"})
    summary = summary_article[0].text.replace(","," ")

    f.write(title + "," + summary + "," + Label + "\n")

f.close()

########################################## The Washington Post

my_url3 =
'https://www.washingtonpost.com/news/worldviews/?utm_term=.f120edf532
6 e'

#opening connections and grabbing the page
uClient = uReq(my_url3) page_html =
uClient.read()
uClient.close()

#html parsing
page_soup = soup(page_html,"html.parser")

#grabs each article
articles = page_soup.find_all("div",{"class": "story-body col-xs-
8 col-md-8"})

filename = "RealnewsWashingtonP10.csv"
f = open(filename,"w")

Label = 'Real'
headers = "Title, Summary, Label\n"

f.write(headers)

print("Washington worlds news ----------------------------")

for article in articles:
```

```
    title_article = article.find("div",{"class":"story-headline"})
    title_article = title_article.h3.a.get_text()
    title = title_article.replace(","," ").strip()

    summary_article =
article.find("div",{"class":"story-description"})
    summary_article = summary_article.p.get_text()
    summary = summary_article.replace(","," ")

    f.write(title + "," + summary + "," + Label + "\n")

f.close()
```

9.2 Main code

```
import pandas as pd
import numpy as np
import sklearn
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer from
stemming.porter2 import stem
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score,
recall_score, make_scorer, f1_score, accuracy_score
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifi-er
from sklearn.linear_model import
LogisticRegression from sklearn.neural_network
import MLPClassifier from sklearn.metrics import
average_precision_score from sklearn import svm
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.feature_extraction.text import
HashingVectorizer from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn import preprocessing
from sklearn.feature_extraction.text import
CountVectorizer from sklearn.feature_extraction.text
import TfidfVectorizer from sklearn.exceptions import
NotFittedError import itertools
import string
from matplotlib import pyplot as plt
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split


#Importing my dataset
dataset = pd.read_csv(r"C:\Users\Vakis\Desktop\DatasetFR2.csv",
encod-ing="ISO-8859-1")

#Preprocessing stages of our dataset
# The English stop words are used to remove specific words and
very commonly used words
```

```
stpwds = set(nltk.corpus.stopwords.words("english"))

#Fit NaN values with spaces
dataset = dataset.fillna(' ')

#preprocessing of our data
for i in range(0,len(dataset['summary'])):
   #remove noisy elements
   dataset['summary'][i] = re.sub(r'[^a-zA-Z]', ' ',
str(dataset['summary'][i]))

#Stemming and Lemmatizing
porter_stemmer = PorterStemmer()
wordnet_lemmatizer = WordNetLemmatizer()
for i in range(0,len(dataset['summary'])):
 dataset['summary'][i] = por-
ter_stemmer.stem(str(dataset['summary'][i]))
 dataset['summary'][i] = word-
net_lemmatizer.lemmatize(str(dataset['summary'][i]))

# Creating y variable y =
dataset.label
dataset.drop("label",axis=1)

#Train and Test data split
X_train, X_test, y_train, y_test =
train_test_split(dataset['summary'], y, test_size=0.33,
ran-dom_state=10)

#Initialize CountVectorizer
count_vectorizer = CountVectorizer(stop_words='english')

#Fit and transform count vectorizer on summary for train and test
data count_train = count_vectorizer.fit_transform(X_train) count_test
= count_vectorizer.transform(X_test)


#Initialize CountVectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.9)

#Fit and transform tfidf on summary for train and test
data tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)

count_df = pd.DataFrame(count_train.A, col-
umns=count_vectorizer.get_feature_names())
tfidf_df = pd.DataFrame(tfidf_train.A, col-
umns=tfidf_vectorizer.get_feature_names())

difference = set(count_df.columns) - set(tfidf_df.columns)

print(count_df.equals(tfidf_df))
print(count_df.head())
print(tfidf_df.head())

#Construction of confusion matrix as an evaluation
option def plot_confusion_matrix(cm, classes,
                         normalize=False,
                         title='Confusion matrix',
                         cmap=plt.cm.Reds):
    """
```

```
    See full source and example:
    http://scikit-
learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.
h tml

    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:,
        np.newaxis] print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

#Inspect the most informative words for fake and real news
respective-ly
def most_informative_feature_for_binary_classification(vectorizer,
classifier, n=100):
    """
    See: https://stackoverflow.com/a/26980472

    Identify most important features if given a vectorizer and
binary classifier. Set n to the number
    of weighted features you would like to show. (Note: current
imple-mentation merely prints and does not
    return top classes.)
    """

    class_labels = classifier.classes_
    feature_names = vectorizer.get_feature_names()
    topn_class1 = sorted(zip(classifier.coef_[0], feature_names))[:n]
    topn_class2 = sorted(zip(classifier.coef_[0], feature_names))[-n:]

    for coef, feat in topn_class1:
        print(class_labels[0], coef, feat)

    print()

    for coef, feat in reversed(topn_class2):
        print(class_labels[1], coef, feat)
```

```
############################### TF IDF
MODEL ###############################
print("-----------------------------TF-IIDF MODEL CLASSIFICATION-----
-----------------------")

################Importing MultinomialNB
classifier :Classifier1##################
classifier1 = MultinomialNB(alpha=0.1)

#fit the classifier X_train
classifier1.fit(tfidf_train, y_train)

#Perfoming Prediction X_test
predictions1 = classifier1.predict(tfidf_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions1)
print("accuracy of Multinomial classifier: %0.3f" %accuracy)

#Confusion matrix performance for classifier1
cm = metrics.confusion_matrix(y_test, predictions1,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for Multinomial classifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier1, tfidf_train, y_train, cv=10)
precision = cross_val_score(classifier1,tfidf_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier1,tfidf_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier1,tfidf_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for Multinomial
classifi-er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for Multinomial
classifi-er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for Multinomial
classifi-er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for Multinomial
classifi-er:",(str(np.mean(f1_score)*100) + ' %'))


##################Importing PassiveAggresive
classifier :Classifier2##################
classifier2 = PassiveAggressiveClassifier(max_iter=1000,
ran-dom_state=10)

#fit the classifier X_train
classifier2.fit(tfidf_train, y_train)

#Perfoming Prediction X_test
predictions2 = classifier2.predict(tfidf_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions2)
print("accuracy of PassiveAggresive Classifier: %0.3f"

%accuracy) #Confusion matrix performance for classifier2
```

```
cm = metrics.confusion_matrix(y_test, predictions2,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for PassiveAggresive classifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier2, tfidf_train, y_train, cv=10)
precision = cross_val_score(classifier2,tfidf_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier2,tfidf_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier2,tfidf_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for PA classifi-
er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for PA classifi-
er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for PA classifi-
er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for PA classifi-
er:",(str(np.mean(f1_score)*100) + ' %'))


###################Importing MLP: Classifier3#################
classifier3 = MLPClassifier(alpha=0.1, random_state=10)
#fit the classifier X_train
classifier3.fit(tfidf_train, y_train)

#Perfoming Prediction X_test
predictions3 = classifier3.predict(tfidf_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions3)
print("accuracy of MLP classifier: %0.3f" %accuracy)

#Confusion matrix performance for classifier3
cm = metrics.confusion_matrix(y_test, predictions3,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for MLP classifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier3, tfidf_train, y_train, cv=10)
precision = cross_val_score(classifier3,tfidf_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier3,tfidf_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier3,tfidf_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for MLP classifi-
er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for MLP classifi-
er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for MLP classifi-
er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for MLP classifi-
er:",(str(np.mean(f1_score)*100) + ' %'))
```

```
###################Importing Logistic Regression:
Classifi-er4###################
classifier4 = LogisticRegression(C=3.0, random_state=10)

#fit the classifier X_train
classifier4.fit(tfidf_train, y_train)

#Perfoming Prediction X_test
predictions4 = classifier4.predict(tfidf_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions4)
print("accuracy of Logistic Regression classifier: %0.3f" %accuracy)

#Confusion matrix performance for classifier4
cm = metrics.confusion_matrix(y_test, predictions4,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for Logistic Regressionclassifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier4, tfidf_train, y_train, cv=10)
precision = cross_val_score(classifier4,tfidf_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier4,tfidf_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier4,tfidf_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for Logistic Regression
classifi-er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for Logistic Regression
classifi-er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for Logistic Regression
classifi-er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for Logistic Regression
classifi-er:",(str(np.mean(f1_score)*100) + ' %'))


###################Importing AdaBoost: Classifier5##################
classifier5 = AdaBoostClassifier(n_estimators=100, random_state=10)

#fit the classifier X_train
classifier5.fit(tfidf_train, y_train)

#Perfoming Prediction X_test
predictions5 = classifier5.predict(tfidf_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions5)
print("accuracy of AdaBoost classifier: %0.3f" %accuracy)

#Confusion matrix performance for classifier5
cm = metrics.confusion_matrix(y_test, predictions5,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for AdaBoost classifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier5, tfidf_train, y_train, cv=10)
```

```
precision = cross_val_score(classifier5,tfidf_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier5,tfidf_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier5,tfidf_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for AdaBoost
classifi-er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for AdaBoost
classifi-er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for AdaBoost
classifi-er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for AdaBoost
classifi-er:",(str(np.mean(f1_score)*100) + ' %'))


############################## BAG-OF-WORDS
MODEL ########################
print("-----------------------------BAG-OF-WORDS MODEL
CLASSIFICA-TION-------------------------")

################Importing MultinomialNB
classifier :Classifier1##################
classifier1 = MultinomialNB(alpha=0.2)

#fit data
classifier1.fit(count_train, y_train)

#Perfoming Prediction X_test
predictions1 = classifier1.predict(count_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions1)
print("accuracy of Multinomial Classifier: %0.3f" %accuracy)

#Confusion matrix performance with classifier1
cm = metrics.confusion_matrix(y_test, predictions1,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for Multinomial classifier\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier1, count_train, y_train, cv=10)
precision = cross_val_score(classifier1,count_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier1,count_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier1,count_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for Multinomial
classifi-er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for Multinomial
classifi-er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for Multinomial
classifi-er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for Multinomial
classifi-er:",(str(np.mean(f1_score)*100) + ' %'))
```

```
###################Importing PassiveAggresive
classifier :Classifier2##################
classifier2 = PassiveAggressiveClassifier(max_iter=100,
ran-dom_state=10)

#fit the data
classifier2.fit(count_train, y_train)

#Perfoming Prediction X_test
predictions2 = classifier2.predict(count_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions2)
print("accuracy of PassiveAggresive Classifier: %0.3f" %accuracy)

#Confusion matrix performance with classifier2
cm = metrics.confusion_matrix(y_test, predictions2,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for PassiveAggresive classifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier2, count_train, y_train, cv=10)
precision = cross_val_score(classifier2,count_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier2,count_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier2,count_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for PA classifi-
er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for PA classifi-
er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for PA classifi-
er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for PA classifi-
er:",(str(np.mean(f1_score)*100) + ' %'))

###################Importing MLP: Classifier3##################
classifier3 = MLPClassifier(alpha=0.2, random_state=10)

#fit the classifier X_train
classifier3.fit(count_train, y_train)

#Perfoming Prediction X_test
predictions3 = classifier3.predict(count_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions3)
print("accuracy of MLP Classifier: %0.3f" %accuracy)

#Confusion matrix performance for classifier3
cm = metrics.confusion_matrix(y_test, predictions3,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for MLP classifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier3, count_train, y_train, cv=10)
```

```
precision = cross_val_score(classifier3,count_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier3,count_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier3,count_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for MLP classifi-
er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for MLP classifi-
er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for MLP classifi-
er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for MLP classifi-
er:",(str(np.mean(f1_score)*100) + ' %'))

###################Importing Logistic Regression:
Classifi-er4###################
classifier4 = LogisticRegression(C=0.1, random_state=10)

#fit the classifier X_train
classifier4.fit(count_train, y_train)

#Perfoming Prediction X_test
predictions4 = classifier4.predict(count_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions4)
print("accuracy of Logistic Regression Classifier: %0.3f" %accuracy)

#Confusion matrix performance for classifier4
cm = metrics.confusion_matrix(y_test, predictions4,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for Logistic Regression classifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier4, count_train, y_train, cv=10)
precision = cross_val_score(classifier4,count_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier4,count_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier4,count_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for Logistic Regression
classifi-er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for Logistic Regression
classifi-er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for Logistic Regression
classifi-er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for Logistic Regression
classifi-er:",(str(np.mean(f1_score)*100) + ' %'))

###################Importing AdaBoost: Classifier5###################
classifier5 = AdaBoostClassifier(n_estimators=200, random_state=10)

#fit the classifier X_train
classifier5.fit(count_train, y_train)

#Perfoming Prediction X_test
```

```
predictions5 = classifier5.predict(count_test)

#Evaluation of the Results
accuracy = metrics.accuracy_score(y_test, predictions5)
print("accuracy of AdaBoost Classifier: %0.3f" %accuracy)

#Confusion matrix performance for classifier5
cm = metrics.confusion_matrix(y_test, predictions5,
labels=['Fake', 'Real'])
plot_confusion_matrix(cm, classes=['Fake', 'Real'])
print("Confusion matrix for AdaBoost classifier:\n",cm)

#Cross validation with 10 folds
scores = cross_val_score(classifier5, count_train, y_train, cv=10)
precision = cross_val_score(classifier5,count_train, y_train,
scor-ing='precision_weighted', cv=10)
recall = cross_val_score(classifier5,count_train, y_train,
scor-ing='recall_weighted', cv=10)
f1_score = cross_val_score(classifier5,count_train, y_train,
scor-ing='f1_weighted', cv=10)

print("Cross validation on Accuracy for AdaBoost
classifi-er:",(str(np.mean(scores)*100) + ' %'))
print("Cross validation on Precision for AdaBoost
classifi-er:",(str(np.mean(precision)*100) + ' %'))
print("Cross validation on Recall for AdaBoost
classifi-er:",(str(np.mean(recall)*100) + ' %'))
print("Cross validation on F1-score for AdaBoost
classifi-er:",(str(np.mean(f1_score)*100) + ' %'))

#################################################################
Weighted Ranking Model ##############################################

def most_informative_feature_for_binary_classification(vectorizer,
classifier, n=100):
    """
    See: https://stackoverflow.com/a/26980472

    Identify most important features if given a vectorizer and
binary classifier. Set n to the number
    of weighted features you would like to show. (Note: current
imple-mentation merely prints and does not
    return top classes.)
    """

    class_labels = classifier.classes_
    feature_names = vectorizer.get_feature_names()
    topn_class1 = sorted(zip(classifier.coef_[0], feature_names))[:n]
    topn_class2 = sorted(zip(classifier.coef_[0], feature_names))[-n:]

    for coef, feat in topn_class1:
        print(class_labels[0], coef, feat)

    print()

    for coef, feat in reversed(topn_class2):
        print(class_labels[1], coef, feat)


most_informative_feature_for_binary_classification(tfidf_vectorizer,
classifier2, n=30)
```

```
feature_names = tfidf_vectorizer.get_feature_names()

#number of the unique words in the dataset
number_of_words = len(feature_names)

#weighted score for each and every word
Weight = sorted(zip(feature_names,
classifi-er2.coef_[0]))[:number_of_words]

#This is the index in order to decide if an unkown input is Fake
or Real
Ranking_Index = []

#Inserting a new input by the user
Input_Text = input('Enter the text:')
#Preprocessing stage for the unkown input
Input_Text = re.sub(r'[^a-zA-Z]', ' ', str(Input_Text))
#Keep only lower letters
Input_Text = Input_Text.lower()
#Filter out stopwords
Input_Text= ' '.join([word for word in Input_Text.split() if word
not in (stopwords.words('english'))])
print("Preprocessed text: ", Input_Text)

for word, score in Weight:
    if word in Input_Text.split():
        Ranking_Index.append(score)

#Ranking Index includes the score of each and every word that
contains print(Ranking_Index)
#Summing the elements of Ranking_Index in order to get the final
Rank-ing Score
Ranking_Index_Sum = sum(Ranking_Index)

if Ranking_Index_Sum > 0:
    print("The Article is Real with a score:", Ranking_Index_Sum)
elif Ranking_Index_Sum <0:
    print("The Article is Fake with a score:", Ranking_Index_Sum)
elif Ranking_Index_Sum == 0:
    print("The Article has the same probability to be Fake or Real")
else:
    print("There was an error. Please try again!")
```