NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Fake News Detection on news articles using Machine and Deep Learning Techniques

Stavroula G. Iatropoulou

Supervisor: **Alexandros Ntoulas,** Assistant Professor

ATHENS

JULY 2020

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανίχνευση Ψευδών Ειδήσεων σε άρθρα χρησιμοποιώντας τεχνικές Μηχανικής και Βαθιάς Μάθησης

Σταυρούλα Γ. Ιατροπούλου

Επιβλέπων: **Αλέξανδρος Ντούλας,** Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2020

BSc THESIS

Fake News Detection on news articles using Machine and Deep Learning Techniques

Stavroula G. Iatropoulou
S.N.: 1115201500048

SUPERVISOR: **Alexandros Ntoulas,** Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανίχνευση Ψευδών Ειδήσεων σε άρθρα χρησιμοποιώντας τεχνικές Μηχανικής και Βαθιάς Μάθησης

Σταυρούλα Γ. Ιατροπούλου

Α.Μ.: 1115201500048

ΕΠΙΒΛΕΠΩΝ: **Αλέξανδρος Ντούλας,** Επίκουρος Καθηγητής

ABSTRACT

In the recent years a widespread proliferation of fake and fabricated content has been witnessed among various online web platforms, leading to the formation of a distorted and most of the times manipulated public opinion towards political, social or other everyday issues. Therefore, there have been many efforts to develop fake news detection systems to contribute in the unveiling and debunking of deceptive news.

In this thesis we will be working on three different fake news datasets, trying to spot the linguistic differences between fake and truthful articles providing a visualization of the results. The aim of the project is to compare five different machine learning classifiers as well as to develop an ensemble method of different combinations of classification models to investigate which gives best universal results for all three data sources. Afterwards, a simple long short-term memory neural network was developed to examine the results of a deep learning method in comparison to statistical machine learning models.

The methodology followed in this project starts with applying natural language processing and feature extraction techniques which aim to prepare the data to be “fed” into each classification model for training and tuning parameters for each classifier. The results are then presented and compared using barplots, confusion matrices and precision–recall curves.

SUBJECT AREA: Machine Learning, Deep Learning, Data Mining, Data Visualization

KEYWORDS: fake news detection, natural language processing, classification model

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια παρατηρείται εκτεταμένη διάδοση ψευδούς και κατασκευασμένου περιεχομένου μεταξύ διαφόρων διαδικτυακών πλατφορμών διαδικτύου, με αποτέλεσμα τη διαμόρφωση διαστρεβλωμένης και συχνά χειραγωγούμενης κοινής γνώμης σε πολιτικά, κοινωνικά ή άλλα καθημερινά θέματα. Ως εκ τούτου, έχουν καταβληθεί πολλές προσπάθειες για την ανάπτυξη συστημάτων ανίχνευσης ψευδών ειδήσεων για να συμβάλλουν στην αποκάλυψη και στην κατάρριψη παραπλανητικών ειδήσεων.

Σε αυτή την πτυχιακή εργασία θα δουλέψουμε πάνω σε τρία διαφορετικά σύνολα δεδομένων ψευδών ειδήσεων, προσπαθώντας να εντοπίσουμε τις γλωσσικές διαφορές ανάμεσα σε ψευδή και αληθή άρθρα, παρέχοντας μια απεικόνιση των αποτελεσμάτων. Σκοπός της εργασίας είναι η σύγκριση πέντε διαφορετικών ταξινομητών μηχανικής μάθησης, καθώς και η ανάπτυξη μιας μεθόδου συνόλου διαφορετικών συνδυασμών μοντέλων ταξινόμησης για να εξετάσουμε ποια δίνουν τα καλύτερα γενικά αποτελέσματα για τις τρεις πηγές δεδομένων. Στη συνέχεια, αναπτύχθηκε ένα απλό νευρωνικό δίκτυο βραχυπρόθεσμης μνήμης για να εξετάσει τα αποτελέσματα μιας μεθόδου βαθιάς μάθησης σε σύγκριση με τα μοντέλα στατιστικής μηχανικής μάθησης.

Η μεθοδολογία που ακολουθείται σε αυτή την εργασία αρχίζει με την εφαρμογή τεχνικών επεξεργασίας φυσικής γλώσσας και τεχνικών εξαγωγής χαρακτηριστικών που στοχεύουν στην προετοιμασία των δεδομένων τα οποία πρόκειται να “τροφοδοτηθούν” σε κάθε μοντέλο ταξινόμησης για την εκπαίδευση και την ρύθμιση παραμέτρων για κάθε ταξινομητή. Τα αποτελέσματα παρουσιάζονται στη συνέχεια και συγκρίνονται χρησιμοποιώντας ραβδογράμματα, πίνακες σύγχυσης και καμπύλες ακριβείας - ανάκτησης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική Μάθηση, Βαθιά Μάθηση, Εξόρυξη Δεδομένων, Οπτικοποίηση Δεδομένων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: εντοπισμός ψευδών ειδήσεων, επεξεργασία φυσικής γλώσσας, μοντέλο κατηγοριοποίησης

To my family.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Prof. Alexandros Ntoulas, for giving me the opportunity to work on this subject and helping me with his guidance and useful advice on obstacles that came up while implementing this thesis.

CONTENTS

LIST OF FIGURES

LIST OF IMAGES

LIST OF TABLES

PREFACE

The work for this thesis was conducted between October 2019 and June 2020 at the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens and was supervised by Assistant Professor Alexandros Ntoulas. The project was developed on a Linux machine using the Python programming language. For the development of the project, it was of great importance to understand and get familiar with sklearn library and the concepts of machine learning, that it implements, as well as with PyTorch framework used to implement methods connected to deep learning field.

1. INTRODUCTION

The topic of fake news is not something that came up in the recent years along with the technological revolution. The roots of the problem date back to the time when the news started spreading widely after the invention of printing press in 1439 [1]. Any piece of information can be characterized as fake content as long as it can be confirmed that is false and has been intentionally fabricated [1].

Therefore, when building a fake news detection system, not only the linguistic patterns of the information but also the social and political context should be considered. When it comes to fake news, there are three types of fake that should be mentioned: a) serious fabrications (yellow press, tabloids, etc.), b) large-scale hoaxes and c)humorous fake such as satire or news parody [2].

Given various features (title, text, user social engagements, etc.) ***f*** for a news article ***α***, the task of fake news detection is to predict whether the news is a fake news piece or not, such $F : \boldsymbol{f} \rightarrow \{0, 1\}$ that,

$$F(a) = \begin{cases} 1, & \textit{if } a \textit{ is a piece of fake news} \\ 0, & \textit{otherwise}. \end{cases} \qquad (1)$$

where F is the prediction function we want to learn [1]. In this thesis, the input ***f*** from which we extract the important information we need to classify an article ***α*** is the title and text from the ISOT dataset (§ 2.1) and only the text from Gossipcop (§ 2.2.1) and Politifact datasets (§ 2.2.3).

1.1 Importance and challenges of Problem

Social media are becoming more and more popular for news consumption and the amount of people's time spent on social media platforms is increasing day by day due to their fast and easy access. This convenience, though is opposed by the exposure to a large "pool" of fake news that is constantly propagating. Therefore, the process of identifying and debunking fabricated piece of information is crucial not only on an individual level but also has a significant social and political impact.

However, the process of developing a fake news detection system faces several challenges. First, the way that fake news are constructed aims to deceive people into believing false information and, thus, shape public opinion in favor or against specific

directions, depending on the social and political situation that applies. Therefore, except news content, secondary features, such as user engagements in social media, should be taken into consideration to help the identification.

Secondly, one major challenge that needs to be addressed is the scarcity of well-structured and complete datasets. Social media data is multi-modal, mostly user-generated and noisy [3]. Moreover, as happens to other Machine Learning fields of study, it is highly challenging to find a dataset that can be objective and at a certain point representative of the different classes that need to be taken into account. Some of these difficulties came up during the implementation of this thesis and will be described on Section 2.

1.2 Related Work

Different approaches have been emerging recently concerning how to address the problem of the wide spread of fake news and lead to as accurate as possible detection of false information.

Niall J. Conroy et al. [5] propose linguistic cue and network analysis approaches to support the development of fake news detection systems. Linguistic approach consists of finding correlations between language patterns and deception during analysis of the deceptive news content. “Bag of words” or n-grams are simple data representation methods to measure word or term frequencies and reveal patterns of deception. However, these methods may separate words from useful surrounding information leading to misunderstanding of content. In order to tackle the disadvantage of word analysis, deep syntax methods, such as Probability Context Free Grammars (PCFG), are used for deception detection.

Classifiers, such as Support Vector Machines (SVM) and Naive Bayes, trained with the results of the linguistic and sentiment analysis are promising but it may be difficult to generalize because of the real-time nature of news. Network analysis makes use of knowledge networks to build fact-checking methods. It is based on querying publicly available structured networks and analyzing social media behavior. Both the two approaches have set the ground for accurate detection of fake content but combined in a hybrid model may bring better results.

This thesis is based on a similar project described on paper **Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques**, in which n-gram features and machine learning techniques are utilized to perform text analysis aiming to detect fake news [4]. Different methods of features extraction and six different classifiers are proposed and compared during an experimental evaluation conducted on a ISOT dataset (§ 2.1), which is also used in my project. The common ground between this project and my work is the machine learning techniques on data collected from different sources, with the initial ambition to build classification models that lead to universal encouraging results among this data and, also, compare the statistical ML methods with a simple LSTM neural network architecture.

1.3 Objective

The objective of this thesis is to to compare different machine learning classifiers and a deep learning LSTM model as well as to develop ensemble methods of different combinations of classification models to investigate which gives best universal results for all three data sources. My work also aims to collect and construct well-balanced and as representative as possible datasets (§ 2.2.1, § 2.2.3).

The project is implemented on Python 3.6.9 accompanied with various python libraries. For the web crawling part that took place, *BeautifulSoup* (Python library for pulling data out of HTML and XML files) was used. During the preprocessing of the datasets, *re, string* and *gensim* libraries are utilized. As far as algorithms are concerned, *sklearn* (machine learning library for python) is mainly used and *PyTorch* framework for the neural network model. As for the data visualization stage, *wordcloud* (python library for generating wordclouds), *matplotlib* (plotting library for Python), *scikitplot* (visualization library or *"the result of an unartistic data scientist's dreadful realization that visualization is one of the most crucial components in the data science process, not just a mere afterthought."* [6]) and *seaborn* (Python data visualization library based on matplotlib) were used to construct different plots, tables and diagrams.

2. DATASETS

When structuring a Machine Learning project, it is of crucial importance to work with datasets that are well-structured and as representative as possible so as to deliver an objective intuition of the different data classes that they represent. This section summarizes the content of the datasets that were used, the collection process and presents interesting data visualizations to provide a first look on the data. Fake articles are labeled as 1, true articles as 0.

2.1 ISOT Fake News Dataset

2.1.1 Description

The Information Security and Object Technology (ISOT) Fake News Dataset[1] is a compilation of several thousands articles, obtained from different verified news sites and sites indicated to be untrustworthy by Politifact[2] (a fact-checking website) and Wikipedia[3]. The truthful articles were collected by crawling articles from Reuters[4], whereas the fake articles were collected from diverse news sources. The articles refer to different topics, but most of them emphasize on political and World news topics and are divided into 2 sets, train (70%) and test set (30%).

This dataset contains 44898 (true: 48%, fake: 52%) news articles, focused mostly on the period of 2016 to 2017. It is stored in a CSV file and each article is being described by a unique id, an article title, a text, a subject that the news information is about, a date of publication of the article and the label that indicates its veracity denoted as 1 or 0. The data is mostly cleaned from mistakes and non-readable characters, but the punctuation that exist in both texts are kept because it may be utilized for extracting further patterns of the data as a distinguishing feature and, therefore, contribute to a future classification task.

[1] https://www.uvic.ca/engineering/ece/isot/datasets/index.php

[2] https://www.politifact.com/

[3] https://www.wikipedia.org/

[4] https://www.reuters.com/

2.1.2 Data Visualization

- **Word clouds**

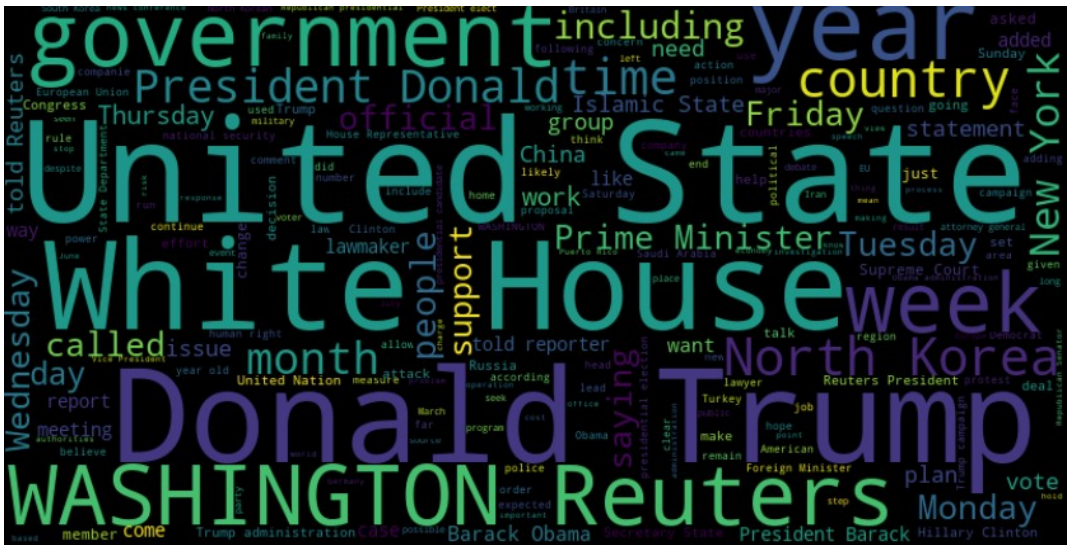

Figure 1: Word cloud representing most used terms in true articles (ISOT).

Word clouds were generated for both true and fake news articles. It is noticeable that in fake articles (Figure 2) the frequency is equally balanced among many of the words (except for terms “Donald Trump”) whereas in true articles (Figure 1) there are terms that stand out from the others (“United State”, “White House”, “Donald Trump”, “WASHINGTON Reuters” ,”government”).

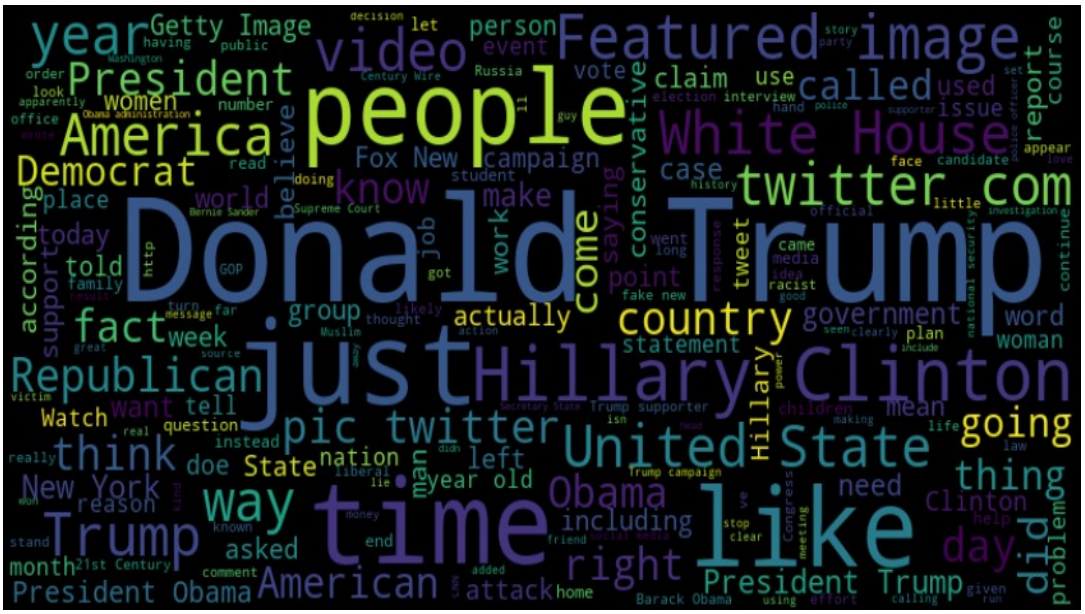

Figure 2: Word cloud representing most used terms in fake articles (ISOT).

- **Histograms**

Histograms were used to point out the linguistic features (text_length, word_count) and sentiment dimension of the article text of ISOT dataset.

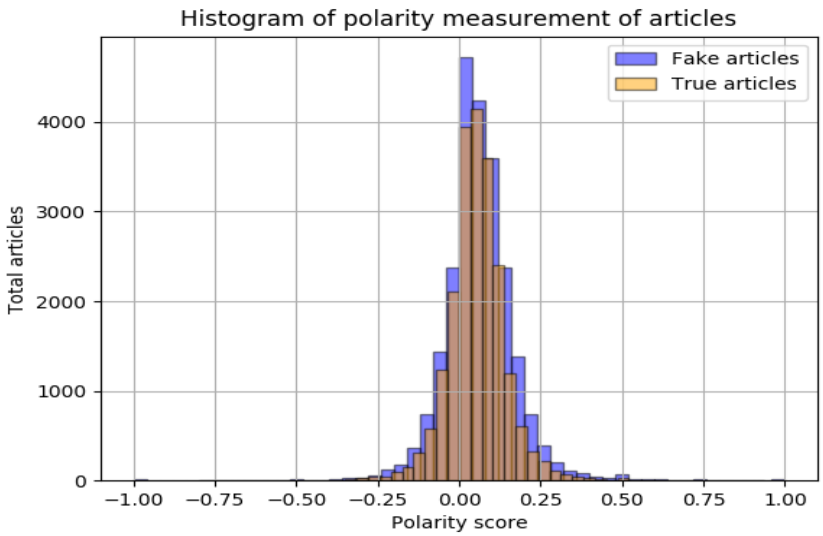

Figure 3: Sentiment (polarity score) in both fake and true articles (ISOT).

The sentiment histogram (Figure 3) shows that both true and fake articles are equally distributed around 0 which means that the corpus of the specific dataset consists of neutral sentiment texts.

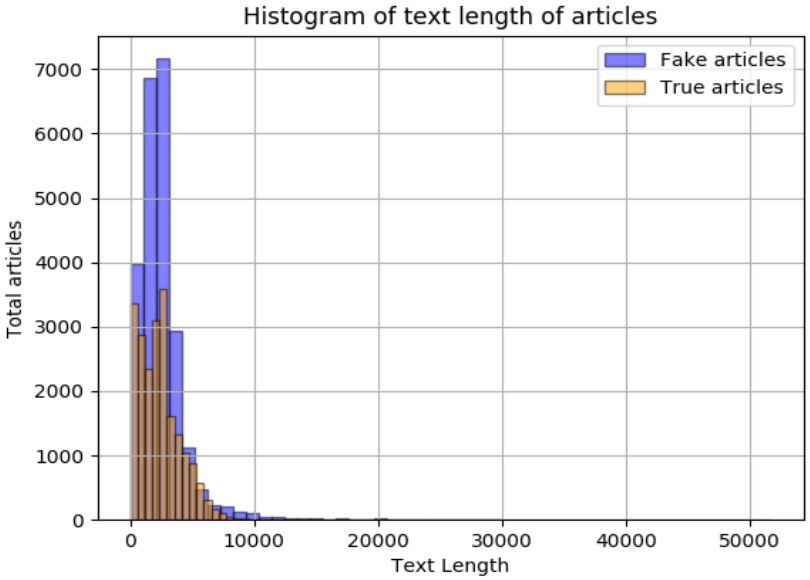

Figure 4: Text length in both fake and true articles (ISOT).

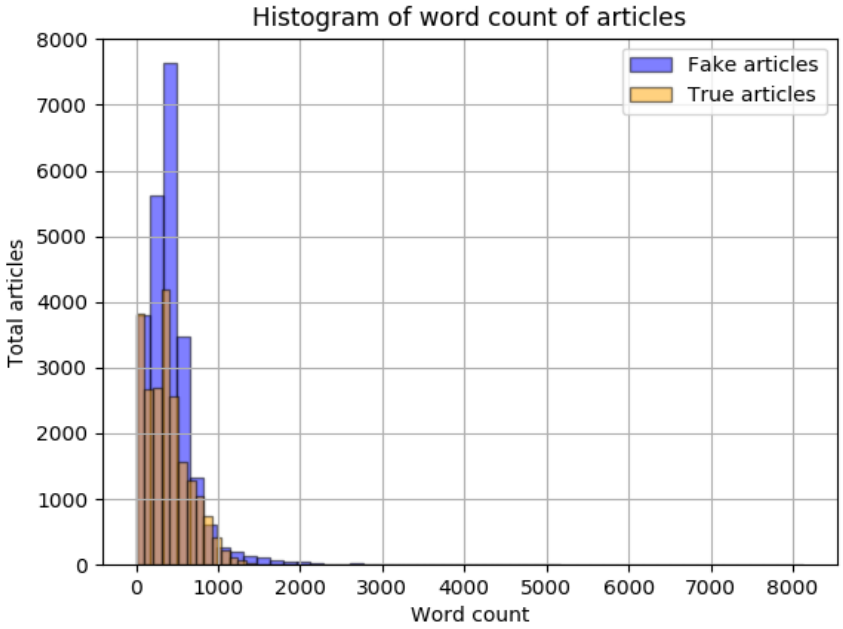

Figure 5: Word count in both fake and true articles (ISOT).

Both texts of true and fake articles range between ~10-1000 total text length (Figure 4). As far as the amount of words used (Figure 5), true articles use up to 1000 words, whereas fake texts consists mostly of around 500-600 reaching max of 2000 words in some articles.

- **Barplots**

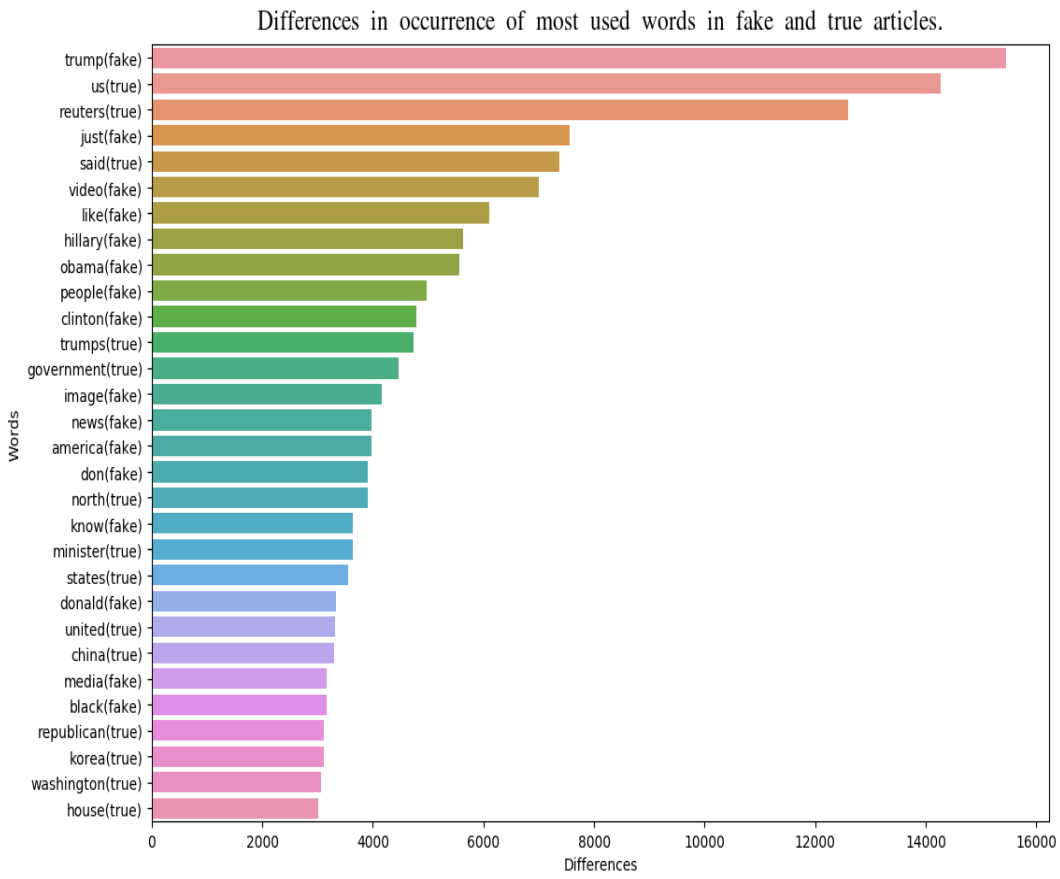

Figure 6: Occurrence difference of most used words in fake and true articles (ISOT)

By summing up the frequency of the common terms in both types of articles and computing the differences, it seems to be helpful to understand which common words could make a difference in classifying an article as truthful or fake. For example, the word “trump” is dominant in both categories, as shown in Figure 1 and Figure 2, but its occurrence in fake articles is more than 15000 more frequent than in true articles.

2.2 Fake News Net Dataset

Fake News Net[5] provides python code for web crawling urls of several political and gossip websites. Because of the privacy and cookies policy of the most websites, the python script fails to obtain the text of the article and gets stuck on the “Accept cookies…” message. I attempted to write a new python script (scraper.py[6]), using the urls provided from the repository, that surpasses this obstacle in many cases but manual handling and cleaning of the dataset was a time-consuming task but crucial for both Gossipcop and Politifact dataset.

2.2.1 Gossipcop Dataset

Regarding Gossipcop dataset, #articles: 18044, true articles: 77%, fake articles: 23%, divided also into 2 sets, train and test set.

2.2.2 Gossipcop Data Visualization

- **Word Clouds**

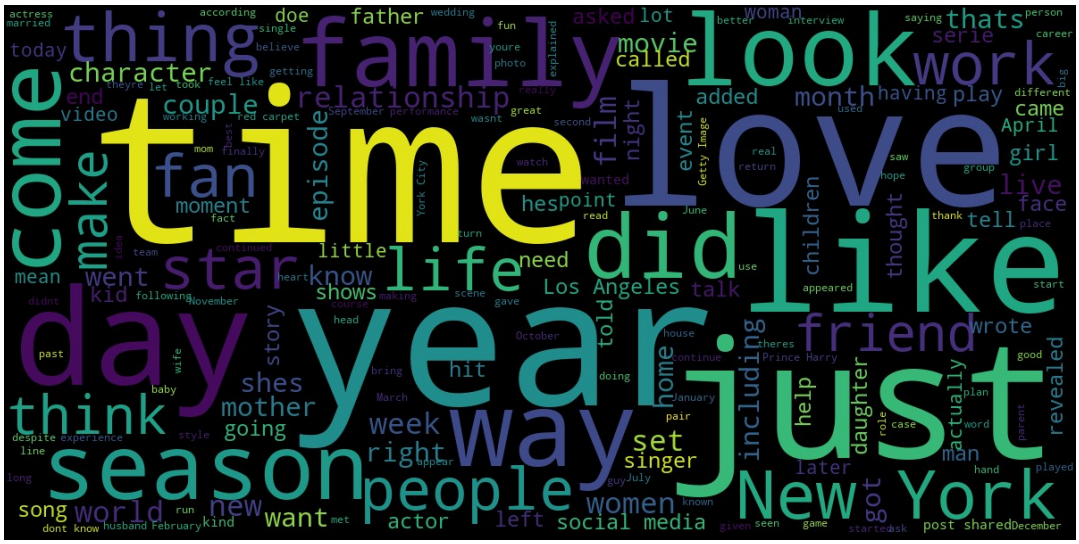

Figure 7: Word cloud representing most used terms in true articles (Gossipcop).

[5] https://github.com/KaiDMML/FakeNewsNet

[6] https://github.com/stavIatrop/Fake-News-Detection/blob/master/FakeNewsNetDataset/scraper.py

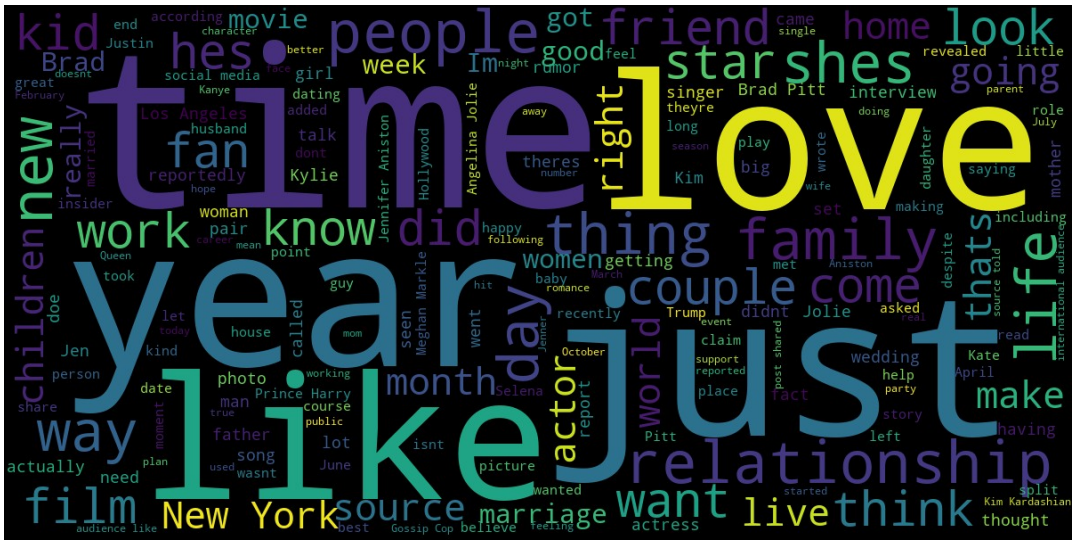

Figure 8: Word cloud representing most used terms in fake articles (Gossipcop).

The word clouds indicate that true and fake texts use almost the same words more frequently, but there are differences in less frequent terms.

- **Histograms**

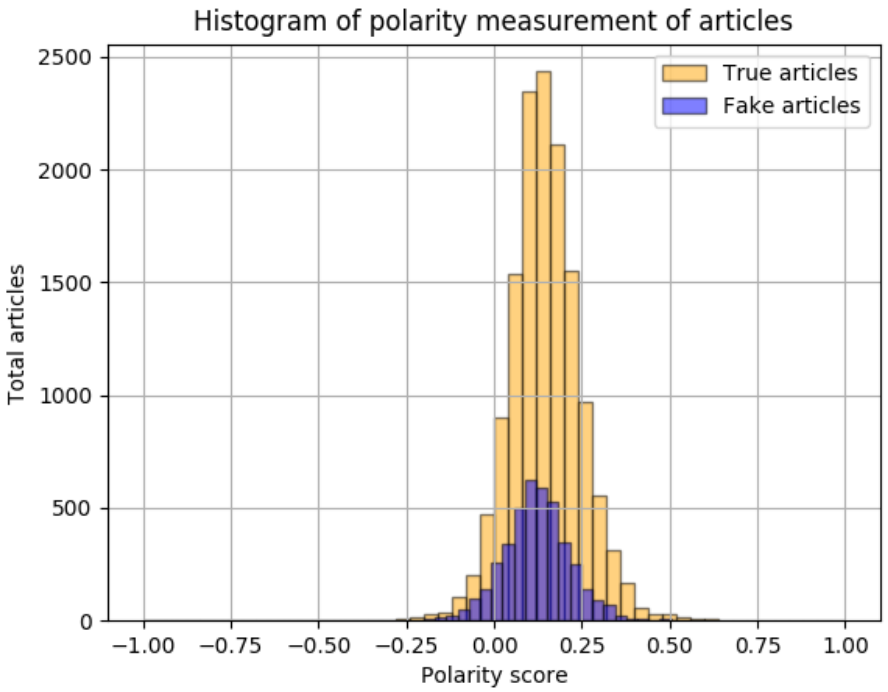

Figure 9: Sentiment (polarity score) in both fake and true articles (Gossipcop).

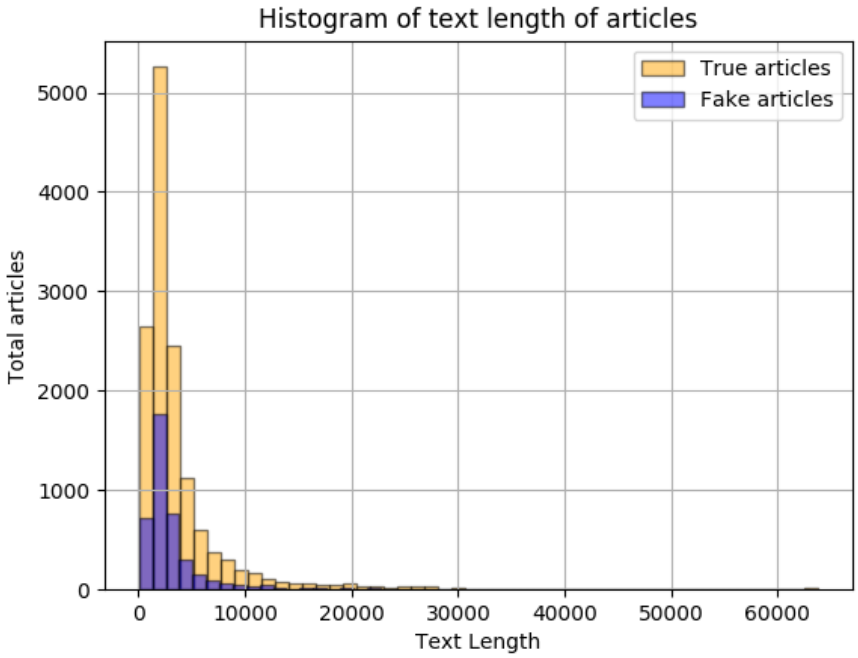

Figure 10: Text length in both fake and true articles (Gossipcop).

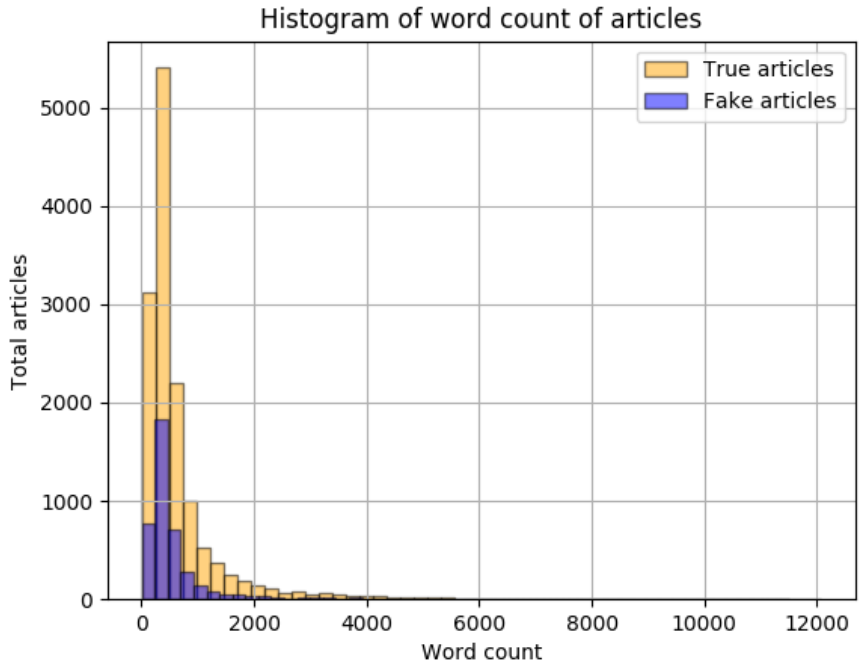

Figure 11: Word count in both fake and true articles (Gossipcop).

The general sentiment that texts emit is neutral to positive in both classes (Figure 9). As for the linguistic features, fake articles are around 400-600 length and 500-1000 words on average. True content fluctuates between the same numbers, but

there also articles that reach 20000 text length and others that use more than 2000 words to elaborate.

- **Barplots**

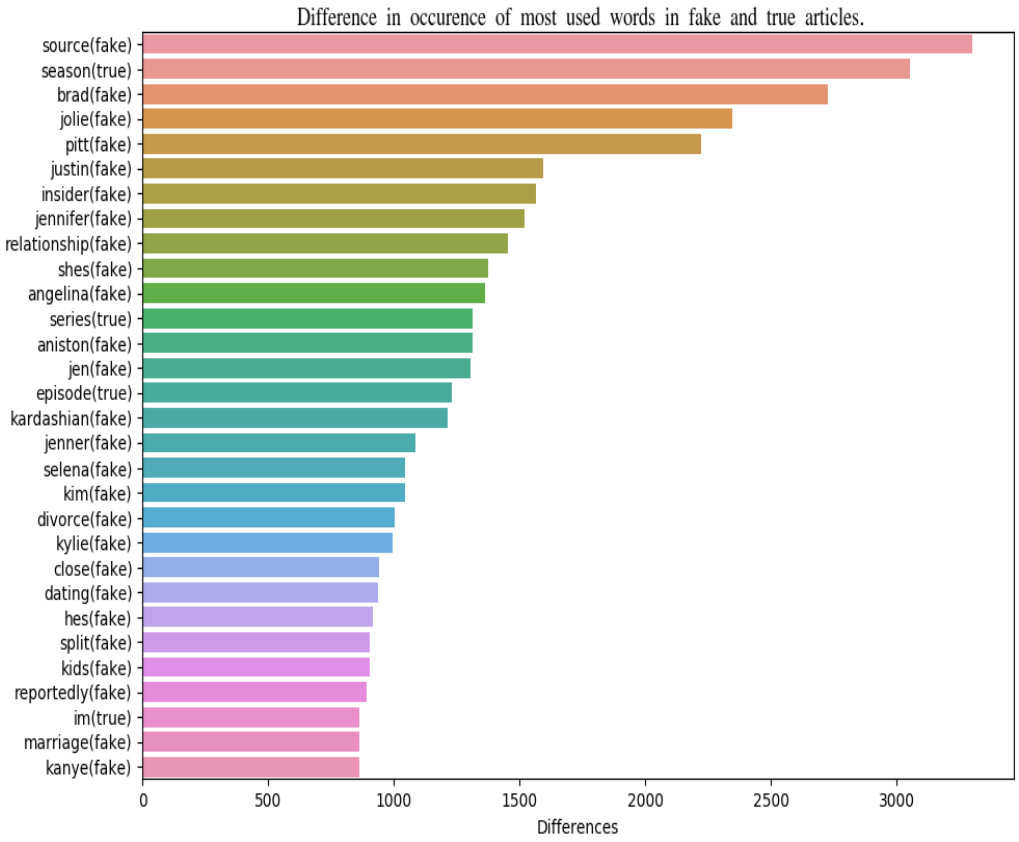

Figure 12: Occurrence difference of most used words in fake and true articles (Gossipcop)

It seems to be more helpful to understand from the occurrence difference barplot (Figure 12) than from word clouds (Figure 7, 8) which common words could make a difference in classifying an article as truthful or fake, as it eliminates the words that prevail in both texts.

2.2.3 Politifact Dataset

After the cleanup, regarding Politifact dataset, #articles: 699 , true artilces: 53% , fake articles: 47%, divided into 2 sets, train and test set.

2.2.4 Politifact Data Visualization

- **Wordclouds**

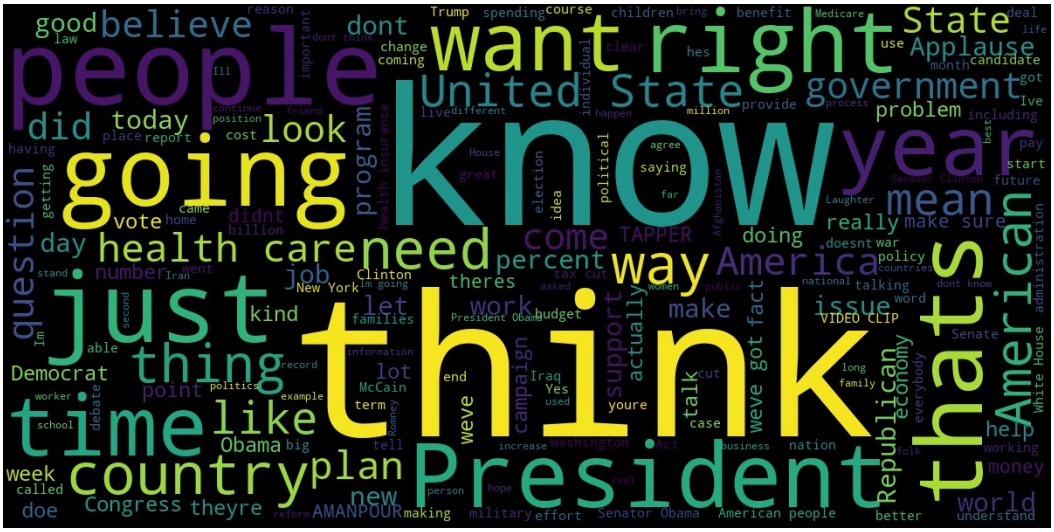

Figure 13: Word cloud representing most used terms in true articles (Politifact).

True terms word cloud (Figure 13) consist of neutral and abstract words mainly, whereas fake terms word cloud (Figure 14) comprises names and terms that refer to political topics.

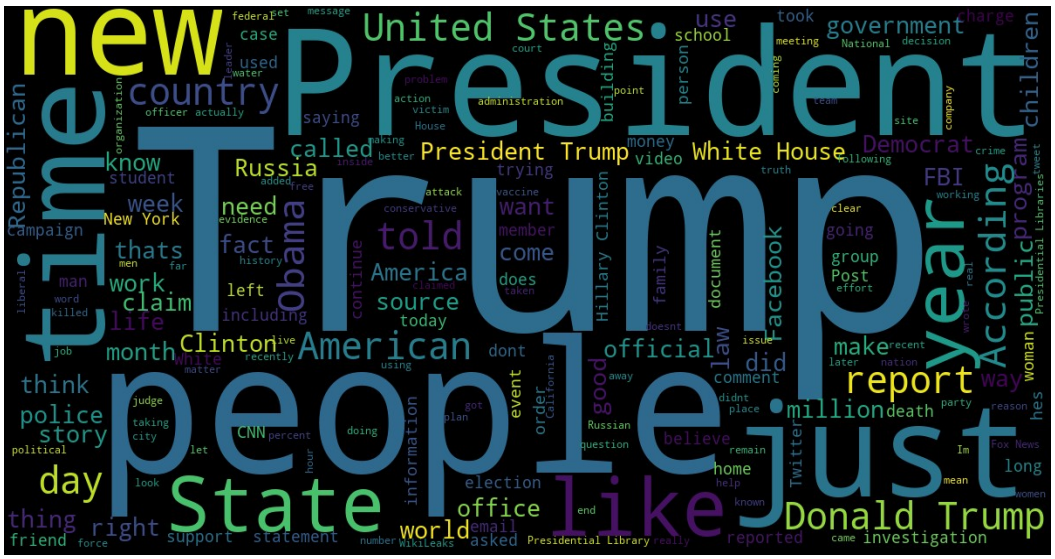

Figure 14: Word cloud representing most used terms in fake articles (Politifact).

- **Histograms**

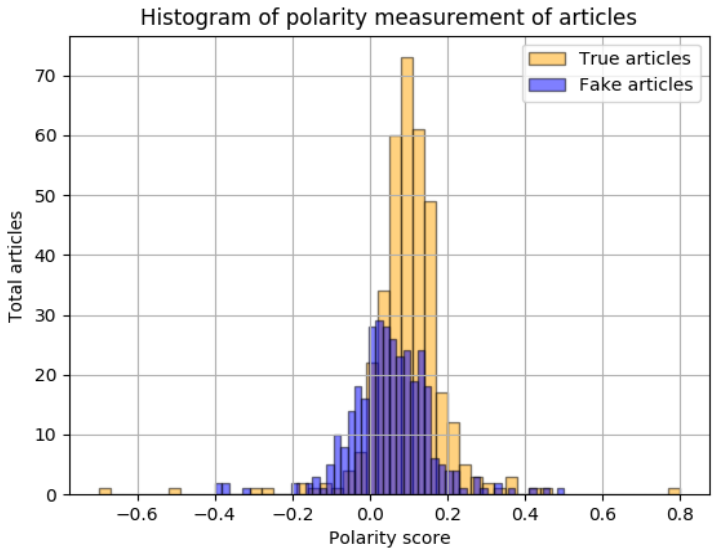

Figure 15: Sentiment (polarity score) in both fake and true articles (Politifact).

The words used fake articles emit a neutral general sentiment, whereas true content tend to a more positive angle, but with not a difference that stands out.

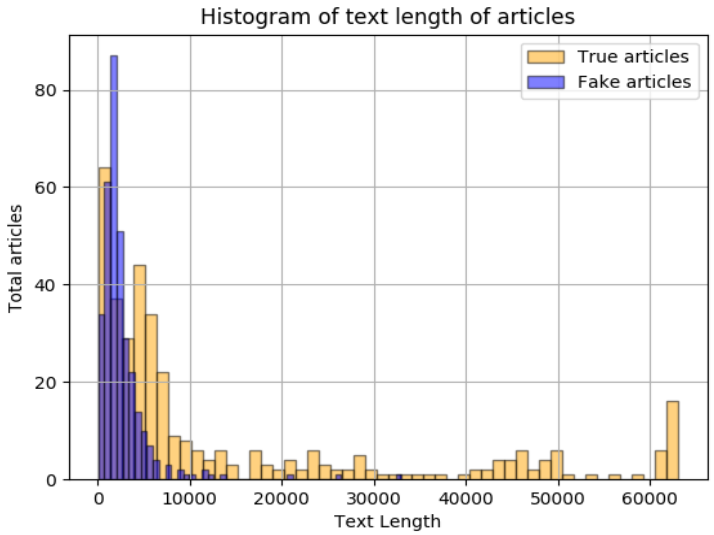

Figure 16: Text length in both fake and true articles (Politifact).

The corpus of fake texts is, compared to truthful content, shorter and fewer terms are used. True articles consists of various word count and text length content ranging from 1000-60000 characters (Figure 16) and from 100-11500 words (Figure 17) .

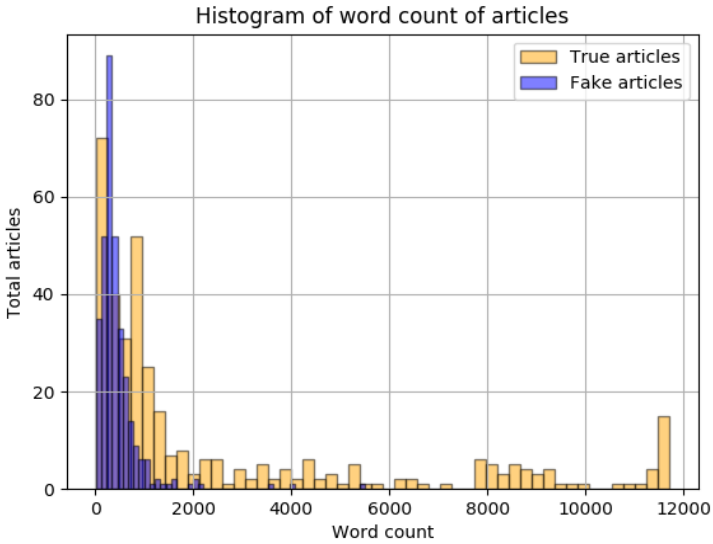

Figure 17: Word count in both fake and true articles (Politifact).

- **Barplots**

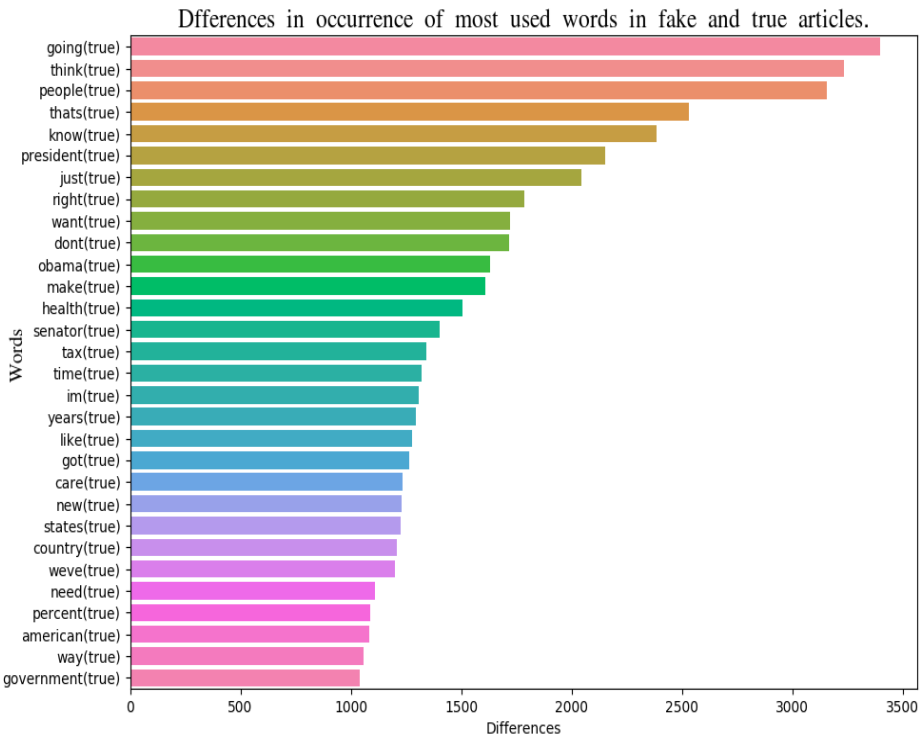

Figure 18: Occurrence difference of most used words in fake and true articles (Politifact).

The fact that was pointed out before, that the truthful articles are larger and richer in words, is also depicted in the occurrence difference barplot (Figure 18), as it comprises common words that are found more frequent in true content.

2.3 Data Preprocessing

In Machine Learning field, after the collection of the data, it is of utmost importance that the data will be prepared and brought to a form that is easily interpreted and analyzed by machines.

2.3.1 Tokenization and cleaning

Tokenization is the process of demarcating and possibly classifying sections of a string of input characters [7]. It is a necessary step, as the process of converting a document into a vector, so it can be valid input for machine learning model, goes through extracting tokens from sequential strings. Equally important is the cleaning of these tokens. It is highly possible that the data, except of useful information, also, consist of a lot of "noise". In the aforementioned datasets, symbols like emojis were manually cleaned and symbol punctuation as well as non-alphanumeric characters were eliminated using a Python script by utilizing *re*[7]*, string*[8] libraries.

2.3.2 Stemming

In linguistic morphology and information retrieval, stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form [8]. For example, the words "detection", "detected", "detects" and "detector" will reduce to "detect". Therefore, the stemming process is important step in NLP as it reduces the volume of text data reserving the essence of the original terms.

In this project, the stemming algorithm was implemented via Porter Stemmer*, gensim* library[9] for topic modelling, which is found to be more efficient and with smaller error ratio than other stemming algorithms.

[7] https://docs.python.org/3/library/re.html

[8] https://docs.python.org/3/library/string.html

[9] https://radimrehurek.com/gensim/parsing/porter.html

2.4 Features Extraction

Features extraction techniques need to be implemented when dealing with large amount of data, as most of it can be redundant and irrelevant causing computations to be time-consuming and bring misleading classification results.

2.4.1 TF-IDF

TF-IDF is a numerical statistic that represents the relevance of a term in a document among others in a set of documents. It combines two metrics, Term Frequency and Inverse Document Frequency. The first metric takes into account the times that a term appears in a document and gives higher rank. However, this may lead to emphasizing trivial words, such as “the”, “and”, etc., which are highly common. Hence, the IDF measure counterbalances this problem by weighting the term concerning its occurrence in the document corpus, giving more value to those that appear rarely.

In this work, the vectorization was performed using the TfidfVectorizer module of *sklearn* library[10]. The vectorizer is also responsible for filtering out useless words, also known as stop words in NLP, such as “a”, “an”, “in”, which have low information value.

2.4.2 Singular Value Decomposition

Data is usually described with high dimensional matrices which consist of columns, a number of which have no predictive value and their processing, by extension, will lead to waste of computational power and poor classification performance. To tackle this problem, a way to perform dimensionality reduction is using *Singular Value Decomposition* (SVD), a linear algebra method that allows factorization of a high-dimensional matrix by eliminating less important parts of that representation to produce an approximate representation with any desired number of dimensions [9] utilizing the properties of the eigenvectors and eigenvalues.

For the dimensions of data to be reduced in order to be input for the machine learning classifier, TruncatedSVD module of *sklearn* library[11] was used. TruncatedSVD takes as input the output matrix of TfidfVectorizer, a process known as *Latent Semantic Analysis* (LSA).

[10] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[11] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

2.4.3 Word Embeddings

Word embeddings are a common way to represent the vocabulary of a document in the Deep Learning field. Each word is assigned a word vector of real numbers (weights) that aim to capture the context of the word as well as the relation with the rest of the words in the data corpus [16]. Usually, neural network methods are used to learn these vector representation based on the usage of the words. The embedding dimension depends on the task and how complex the model is going to be.

There are two options while using word embeddings; either learn an embedding from words in a large text corpus or use pre-trained embeddings as are or fine-tune with new data. In this project, GloVe[12], or Global Vectors, provided by Stanford, were used as a pre-trained word representation to utilize the already computed weights of 6 billion words with embedding dimension of 50 for the embedding matrix to be fed in the NN model.

The data used to learn the pre-trained GloVe embeddings is highly possible that consists of a different vocabulary from the one used in each one of the datasets of this thesis leading to the fact that each vocabulary may not be entirely covered by the embeddings, so standard preprocessing steps, such as stop-word removal, stemming, etc., might not have the expected results. Therefore, in Jupyter Notebook (text_preprocessing_embeddings.ipynb[13]), I attempted to check and improve the coverage of the vocabulary by the GloVe embeddings as much as possible. The conclusion was that the preprocessing should include non-ASCII characters removal, lower case transformation, keeping punctuation and replacing numbers with a number of '#' characters depending on their digits.

[12] https://nlp.stanford.edu/projects/glove/

[13] https://github.com/stavIatrop/Fake-News-Detection/blob/master/text_preprocessing_embeddings.ipynb

3. ALGORITHMS

This section refers to the algorithms that were used during the classification process.

3.1 Support Vector Machine

Support Vector Machine (SVM) is an algorithm used in supervised learning, either for regression or classification tasks. The main goal of SVM, given a set of labeled data points from two different classes, is to find, among others, the hyperplane that best separates these data points. To choose among all the possible planes found, maximizing the margin between both classes should be taken into consideration, as the maximum distance reinforces the confidence for classifying future data points [10].

Depending on which side of the decision boundary of a hyperplane, a data point falls, it gets assigned to the corresponding class. The higher the number of the dimensions of features are, the more sophisticated the hyperplane is. Data points that are closer to the hyperplane are called support vectors and, hence, regulate its position and orientation.

The SVM classification algorithm was implemented in this project using the SVC module of *sklearn* libary[14].

3.2 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is an algorithm used for classification and regression. The training process of this algorithm is based on the idea that similar data points are close to each other. This proximity is calculated by different metrics measuring distance, such as Euclidean, Minkowski, Manhattan etc. When an unclassified data point comes up, the classifier performs a majority voting task by taking into account the *k* most similar data points, regarding the proximity already calculated, and decides the class label that is to be assigned in that point.

The metrics mentioned before, as well as the *k* value are of major concern when it comes to decide their value as they are important for achieving better accuracy [11]. The process that is usually followed to determine these values is called parameter tuning and it takes an amount of experiments to come to a conclusion.

[14] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

The KNN classification algorithm was implemented using the KneighborsClassifier module of *sklearn.neighbors* library[15].

3.3 Logistic Regression

Logistic Regression (LR) is a predictive analysis algorithm used for classification tasks. Its main objective is to assign new observations to a discrete set of class labels, based on the idea of probability [12]. This is achieved by mapping the output values of LR to probabilities using a specific prediction function called *Sigmoid* function, also known as *Logistic* function. The decision of assigning a specific class label to a probability is regulated by a parameter called threshold which forms the decision boundary. The value of the threshold defaults at 0.5 but this varies depending on several reasons, concerning the predicted probabilities or class distribution, the metrics used in training, etc. and needs tuning.

The LR in this project is implemented using the linear model LogisticRegression module of *sklearn* library[16].

3.4 Decision Trees

Decision Trees (DT) are widely used in decision analysis and machine learning fields. It is a decision-making tool that uses a tree-like graph of decisions and their possible outcomes, including chance event outcomes, resource costs and utility [13]. A Decision Tree is represented by *internal nodes* which indicate a condition concerning an attribute. Each internal node splits into *branches* depending on the outcome of the condition until it reaches the end where it does not split anymore and ends up to *leaf nodes* representing the decision taken concerning the class label that is going to be assigned.

The DT algorithm was implemented using the DecisionTreeClassifier module of *sklearn* library[17].

[15] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[16] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[17] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

3.5 Random Forest

Random Forest (RF) is an ensemble classification method that combines multiple Decision Trees. RF is based on the idea that each decision tree predicts an output label for the given input and the label that is most predicted is assigned, a concept known as the wisdom of crowds. Therefore, predictions from different decision-trees should not be related and be better than random guessing for the ensemble classifier have better results. The uncorrelated tree-based models are more accurate combined all together as an ensemble model than each model individual due to the fact that the trees try to balance each others' different errors [14].

The RF algorithm was implemented using the RandomForestClassifier module of *sklearn ensemble* library[18].

3.6 Ensemble Classifier

Along with the custom classifiers, an ensemble method was implemented combining the five aforementioned custom classifiers. The objective is to build a model that acts as a voting classifier and computes the weights to be assigned to each classifier's prediction. First, the probabilities produced by the classifiers are kept in a matrix for every training instance, resulting to each training example to be correlated with a probability vector. This matrix of vectors is then passed on a Logistic Regression model which computes the weights and produces the result label true (0) or fake (1). Moreover, a voting classifier performing a simple majority voting among the models' predictions was implemented in comparison to the aforementioned ensemble model.

3.7 Long short-term memory Neural Network

Long short-term memory (LSTM) units are a subset of Neural Networks and especially Recurrent Neural Networks (RNN) which are used to recognize output patterns from corresponding input patterns. LSTMs take as input single data points or a sequence of data and, contrary to feed-forward networks, they have the ability to keep track of previous long-term dependencies in the input sequences and decide for the output [17]. The LSTM used for this project performs sequence classification which means that after it reads all the sequence of data, produces the output.

[18] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

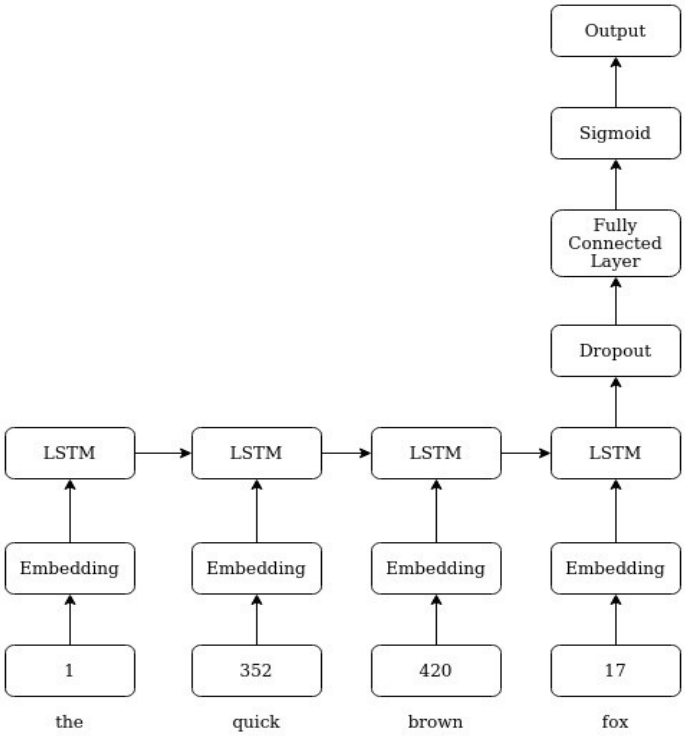

Figure 19: LSTM architecture

The LSTM architecture used for this project is shown in Figure 19 and consists of the embedding layer which is initialized with the embedding matrix created by the weights of the pre-trained GloVe embeddings, a 2-layer LSTM unit with dropout, a separate dropout for regularization as it will result to a smaller network that is less prone to over-fitting, a linear fully-connected layer and a sigmoid activation function.

The LSTM model was implemented using *PyTorch* framework[19].

[19] https://pytorch.org/tutorials/

4. IMPLEMENTATION AND RESULTS

This section refers to the process that has been followed throughout this project and draws comparisons between the results presented.

4.1 Training and Evaluation Process

The training process consists of three stages, tuning each classifier, choosing the parameters that bring the most accurate results using K-Fold Cross-Validation and evaluating the model on the test set, regarding the selected evaluation metrics in both of the two last stages. As far as the LSTM model is concerned, the hyperparameter (learning rate, epochs, dropout rate, etc.) tuning was done by trying different values and checking the training and validation loss.

4.1.1 Tuning Parameters

The training data used to train the classifiers below was preprocessed with TfIdfVectorizer, with different values of *max_df* (ignore terms that have a document frequency strictly higher than the given threshold) for each classifier and dataset, as the data are diverse and from different distributions, and *sublinear_tf=true* (sublinear tf scaling is applied in data), and dimensionality reduction is performed using TruncatedSVD (algorithm set to “arpack”, which is more efficient regarding memory issues that came up because of the large vocabulary).

The parameter selection is performed using GridSearchCV, sklearn module[20] that uses exhaustive search over a specified set of parameter values. To narrow down the different values of the parameters used in each classifier, Validation Curves, module of *sklearn* library[21], which plot the train and test score in order to maximize the test score and reduce as much as possible over-fitting. For each of five custom machine learning classifiers, a set of parameters were chosen to be tuned, as they seem to affect more the results than others.

[20] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[21] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.validation_curve.html

SVM

The parameters that are tuned for each dataset is kernel, regularization parameter C and gamma (if best choice for kernel is not linear). Kernel is a function that is used when the data are of higher dimensions and need transformation in order to compute the similarity or closeness between each other. Kernel's option are 'linear' and 'rbf' during grid search. C value has to do with the margin of the hyperplane. The higher the C value is, the smaller the margin is and, thus, the classification error of training points will be lower. However, this is risk-bearing concerning the generalization of the classifier to data not seen before and can lead to overfitting. Lastly, the gamma parameter is used for non-linear hyperplanes and the higher the gamma value is, the more important will the classifier consider data points closer to the hyperplane.

KNN

The set of parameters that are of importance are number of neighbors(n_neighbors), weights and metric. Choosing the value of n_neighbors is mainly done by trial and error, keeping in mind that lower values may lead to overfitting whereas higher values to underfitting. Weights parameter chooses between two values, 'uniform', all data points have the same influence in the "neighborhood", and 'distance', the influence of a data point increases the closer it is to the query point. Last parameter is a distance metric that uses a distance function between 'Euclidean', 'Manhattan' and 'Minkowski' with the last one accompanied with parameter 'p', which indicates the p-norm, for values higher than 2 (Minkowski is equivalent to Manhattan when p = 1 and to Euclidean when p = 2).

LR

The parameters that were tuned in LR classifier are 'penalty', 'C' parameter and 'solver' parameter. Penalty values, 'l1' or 'l2' refer to the regularization technique that is implemented to tackle overfitting when the number of features is large. C value is a regularization parameter, just like in SVM, that is correlated with the lambda regularization factor with the formula $C = \frac{1}{\lambda}$. The last parameter determines the solver

that is going to be used for the optimization problem of the objective function.

DT

The set of parameters that are to be optimized are ‘max_depth’, ‘min_samples_split’ and ‘criterion’. Starting from the last mentioned, the criterion parameter defines how tree nodes are split according to an impurity measure, implemented in this case with Gini Impurity or Information Gain/Entropy measure. Information Gain is used to decide which feature provides most information about a class, whereas Gini Impurity denotes the probability of a point to be misclassified when randomly selected.

The other two parameters are used to tackle the issue of ovefitting. Especially, ‘max_depth’ should not be set to a higher value as this will lead the model to learn relations very specific to a particular sample. The minimum number of samples required to perform a split in an internal node should, for the same reason as above, not be extremely high, but also, not extremely low as this is going to lead to higher bias.

RF

As Rf is an ensemble method which combines several decision trees, the parameters that were tuned are the same as the DT Classifier and, also, the number of the trees that are going to be aggregated (‘n_estimators’).

4.1.2 K-Fold Cross-Validation

During the process of comparing and selecting a model for a given predictive analysis problem, a validation technique is performed to give insight on how well a model generalizes to unseen data, not used during the training process, called Cross-Validation (CV). CV is a way of preventing under- or over-fitting when there is limited amount of input data. The specific method used in this project was K-Fold Cross-Validation (implemented using KFold, *sklearn* module[22]). The approach of this method can be divide in 5 steps:

1. Randomly shuffle and split the dataset into to k folds.
2. Training the model using as training set the k – 1 folds.
3. Use of the remaining k^{th} fold as test set and validation of the model retaining the scores/errors.
4. Repetition of steps 2, 3 until every K-fold is used as test set.

[22] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

5. Then, all retained errors from each iteration are averaged and this gives a less biased estimation of the model performance.

A diagram illustrating the aforementioned description of the method is depicted in Image 1 [15].

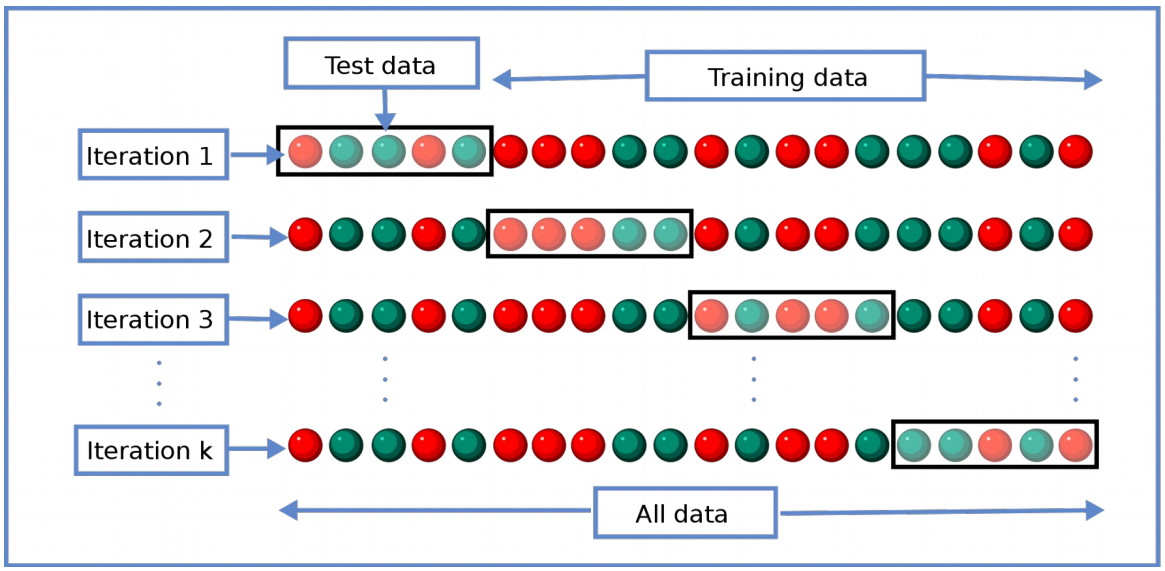

Image 1: Process of K-Fold Cross-Validation. [15]

An important aspect when estimating the results of a machine learning model is to determine the metrics that are going to be used to measure its performance. Before describing the metrics in this project, four definitions about the possible outcomes should be given [1]:

- True Positive (**TP**): when predicted fake news article is actually labeled as fake news;
- True Negative (**TN**): when predicted true news article is actually labeled as true news;
- False Negative (**FN**): when predicted true news article is actually labeled as fake news;
- False Positive (**FP**): when predicted fake news article is actually labeled as true news.

The metrics used in this project can be initially described by the following formulas:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (4)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

Precision metric (2) indicates the ability of the classifier not to label as fake a sample that is true. However, if a dataset is biased, high precision can be easily accomplished with fewer positive predictions [1]. Therefore, recall (3) is used to measure the ability of the classifier to find all the positive samples. During the tuning process, a combination of precision and recall is used, the F1 metric (4), is used to determine the model or the model parameters that bring the most accurate results.

The reason why F1 score is preferred to accuracy measure is that the latter can be considerably modified by large amount of True Negatives which are of lower importance than False Negatives and False Positives. Thus, F1 score contributes to improve the results in tuning process as a metric that seeks a balance between precision and recall, even when the dataset is skewed.

4.1.3 Evaluating on Test Set

When the best combination of parameters for each classifier is chosen, the 20% of the dataset, serving as test set, is used to evaluate model's performance on unseen data. In this phase of the project, each of the above metrics are computed and taken into account to get an overall impression of classifier's reliability. In addition to the metrics mentioned above, Confusion Matrix and Precision-Recall curve are used to better estimate results. Precision-Recall curves are especially used when there is a moderate to large class imbalance in the dataset, as it happens with the case of Gossipcop dataset (§ 2.2.1).

4.2 Results for each Dataset – ML Classifiers

This section summarizes the results that came up after the training process of classifiers. (The number of components used for every classifier, before tuning SVD and performing dimensionality reduction, is 500.)

4.2.1 Results on ISOT Fake News Dataset

Results before tuning

Table 1: Results on 5-fold CV on the training set (ISOT).

Classifier	Accuracy	F1-score
Support Vector Machines (SVM)	0.98	0.99
K-Nearest Neighbors (KNN)	0.79	0.83
Logistic Regression (LR)	0.97	0.97
Decision Trees (DT)	0.9	0.91
Random Forests (RF)	0.95	0.95

Table 2: Results on predictions made on the test set (ISOT).

Classifier	Accuracy	F1-score	Precision /class0	Recall/ class0	Precision/ class1	Recall/ class1
Support Vector Machines (SVM)	0.98	0.98	0.98	0.99	0.99	0.99
K-Nearest Neighbors (KNN)	0.79	0.83	0.99	0.59	0.72	0.99
Logistic Regression (LR)	0.97	0.97	0.97	0.98	0.98	0.97
Decision Trees (DT)	0.92	0.92	0.92	0.91	0.92	0.93
Random Forests (RF)	0.95	0.95	0.93	0.97	0.97	0.94

The results before tuning the ISOT Fake News Dataset reach absolute accuracy in almost every classifier except KNN in both training data and unseen data. After tuning the number of features by performing dimensionality reduction and resulting to 200 components the results were:

Table 3: Results on 5-fold CV on the training set, with 200 components used in TruncatedSVD (ISOT).

Classifier	Accuracy	F1-score
Support Vector Machines (SVM)	0.99	0.99
K-Nearest Neighbors (KNN)	0.93	0.93
Logistic Regression (LR)	0.98	0.98
Decision Trees (DT)	0.94	0.94
Random Forests (RF)	0.97	0.97

There is already an improvement in KNN Classifier before the parameter tuning process has begun.

Results after tuning parameters

Table 4: Results on 5-fold CV on the training set, with 200 components used in TruncatedSVD, before and after tuning (ISOT).

Classifier	Accuracy (before)	Accuracy (after)	F1-score (before)	F1-score (after)
Support Vector Machines (SVM)	0.98	0.99	0.99	0.99
K-Nearest Neighbors (KNN)	0.79	0.95	0.83	0.95
Logistic Regression (LR)	0.97	0.99	0.97	0.99
Decision Trees (DT)	0.9	0.94	0.91	0.94
Random Forests (RF)	0.95	0.97	0.95	0.97

There is notable improvement in KNN Classifier, but overall the results are impressively accurate. The reason behind these impressive results may be found to the fact that the true articles stem only from Reuters news outlet, whereas untruthful ones came from various legitimate news sites. Therefore, it is as if the classifiers try to distinguish

'Reuters' or 'not Reuters' rather than 'fake' or 'true' label among the articles, probably because of the similarity that may find in the writing style of the Reuters' journalists.

Results on Test Set

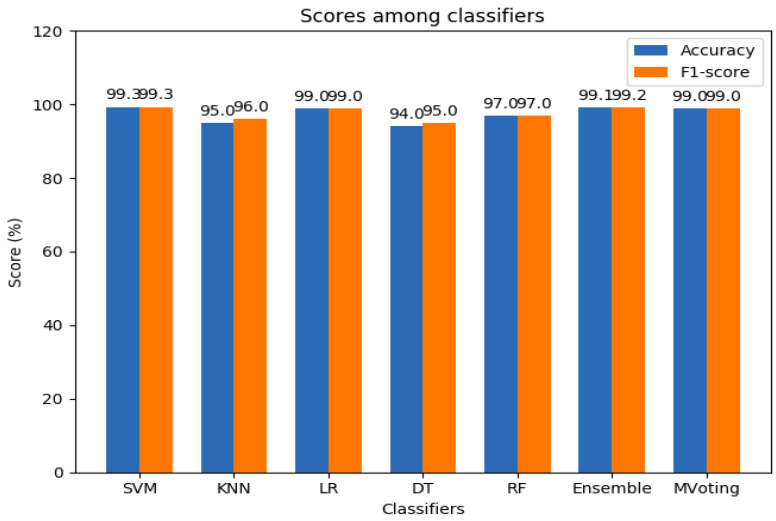

Figure 20: Accuracy and F1-score among classifiers (ISOT)

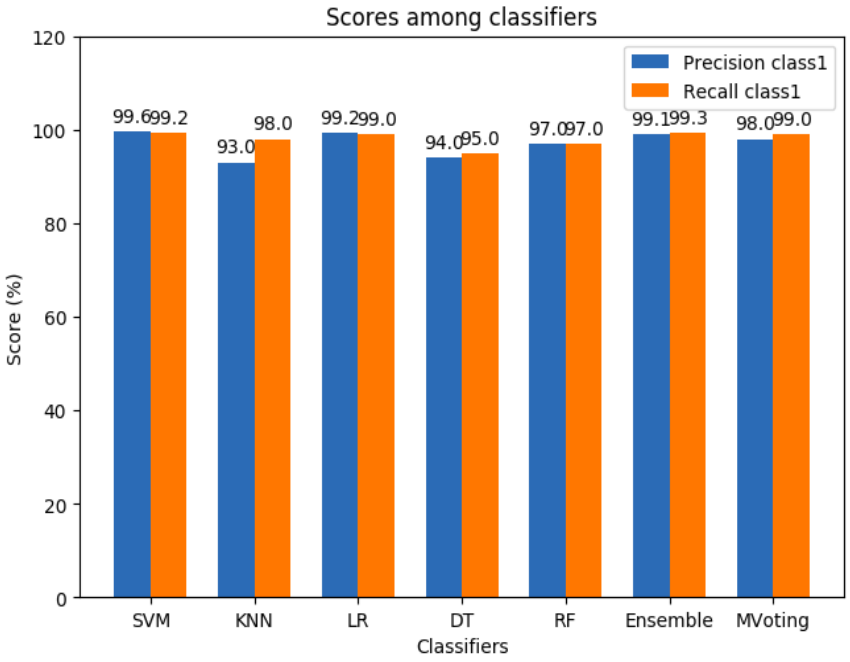

Figure 21: Precision and Recall for class1 among classifiers (ISOT).

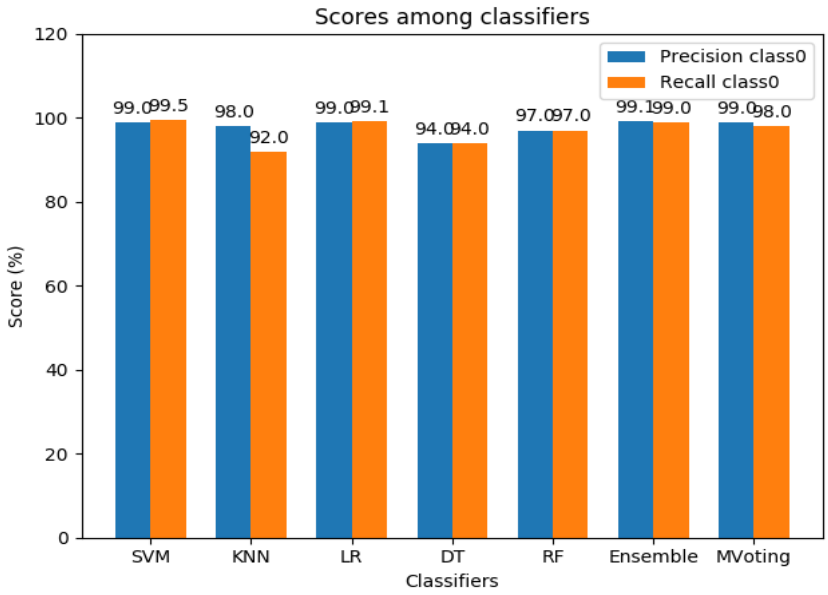

Figure 22: Precision and Recall for class0 among classifiers (ISOT).

Almost every classifier, included ensemble classifier, has the ability to separate the data points between the two classes, except for DT classifier that seems to be a little less accurate, but still with good results overall. Just like the evaluation on training set, the evaluation on test set comes up with pretty accurate results, as it was expected, probably because of the reason mentioned before and, thus, this dataset, although it consists of a large corpus, can be characterized as an “easy to learn” dataset.

4.2.2 Results on Gossipcop Dataset

Results before tuning

Table 5: Results on 5-fold CV on the training set (Gossipcop).

Classifier	Accuracy	F1-score
Support Vector Machines (SVM)	0.88	0.68
K-Nearest Neighbors (KNN)	0.83	0.46
Logistic Regression (LR)	0.87	0.62
Decision Trees (DT)	0.78	0.52
Random Forests (RF)	0.84	0.53

Table 6: Results on predictions made on the test set (Gossipcop).

Classifier	Accuracy	F1-score	Precision/ class0	Recall/ class0	Precision/ class1	Recall/ class1
Support Vector Machines (SVM)	0.88	0.68	0.9	0.95	0.79	0.65
K-Nearest Neighbors (KNN)	0.83	0.46	0.86	0.97	0.81	0.46
Logistic Regression (LR)	0.87	0.63	0.88	0.95	0.79	0.58
Decision Trees (DT)	0.81	0.55	0.85	0.94	0.68	0.42
Random Forests (RF)	0.84	0.51	0.85	0.98	0.84	0.42

The results before tuning the Gossipcop Dataset are marking a high accuracy overall with SVM and LR Classifiers leading in both training and testing data, with DT Classifier having better result on test set. That is happening probably because the initial separation of dataset has led to an ‘easy’ test split concerning the learning process of this classifier. However, the F1-score is kept in low levels, with some classifiers to be as good as random guessing. This probably stems from the fact that the dataset is imbalanced, consisting of less fake (class1) articles (~70% true articles), leading to a biased model towards truthful documents, something, also, indicated by the low recall results for class 1.

After tuning the number of features by performing dimensionality reduction and resulting to 150 components the results were:

Table 7: Results on 5-fold CV on the training set, with 150 components used in TruncatedSVD (Gossipcop).

Classifier	Accuracy	F1-score
Support Vector Machines (SVM)	0.88	0.68
K-Nearest Neighbors (KNN)	0.85	0.61
Logistic Regression (LR)	0.86	0.61
Decision Trees (DT)	0.79	0.55
Random Forests (RF)	0.86	0.63

Dimensionality reduction seem to mark an improvement of ~14% on both KNN and RF Classifiers F1-score.

Results after tuning parameters

Table 8: Results on 5-fold CV on the training set, with 150 components used in TruncatedSVD, before and after tuning (Gossipcop).

Classifier	Accuracy (before)	Accuracy (after)	F1-score (before)	F1-score (after)
Support Vector Machines (SVM)	0.88	0.88	0.68	0.69
K-Nearest Neighbors (KNN)	0.83	0.86	0.46	0.62
Logistic Regression (LR)	0.87	0.87	0.62	0.67
Decision Trees (DT)	0.78	0.83	0.52	0.56
Random Forests (RF)	0.84	0.87	0.53	0.63

There has been an improvement in most classifiers, but the imbalance of the dataset is still notable in F1-score.

Results on Test Set

Because of the dataset imbalance, Precision-Recall curves are used to better visualize each classifier’s ability to separate the two classes.

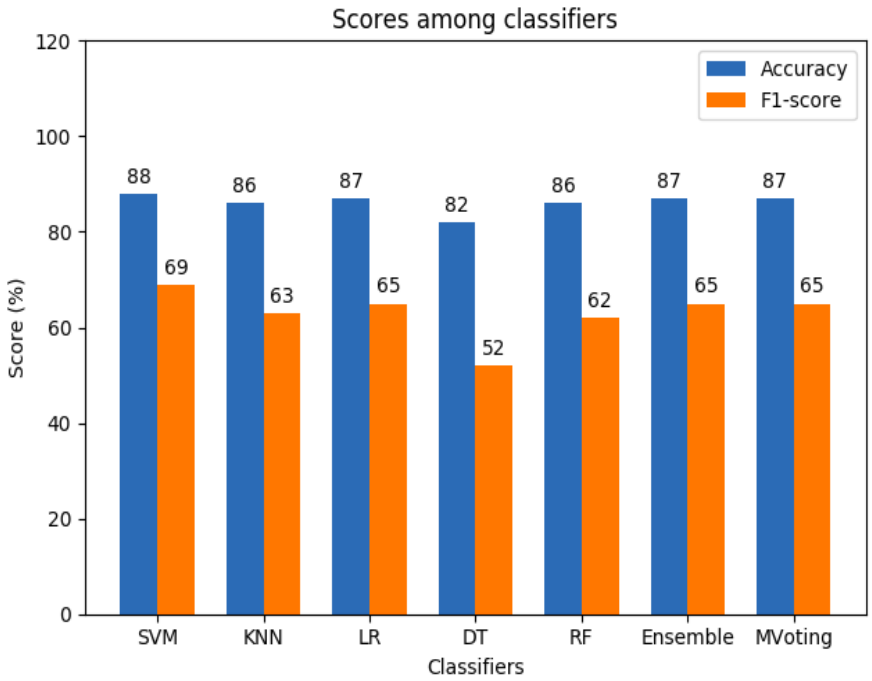

Figure 23: Accuracy and F1-score among classifiers (Gossipcop)

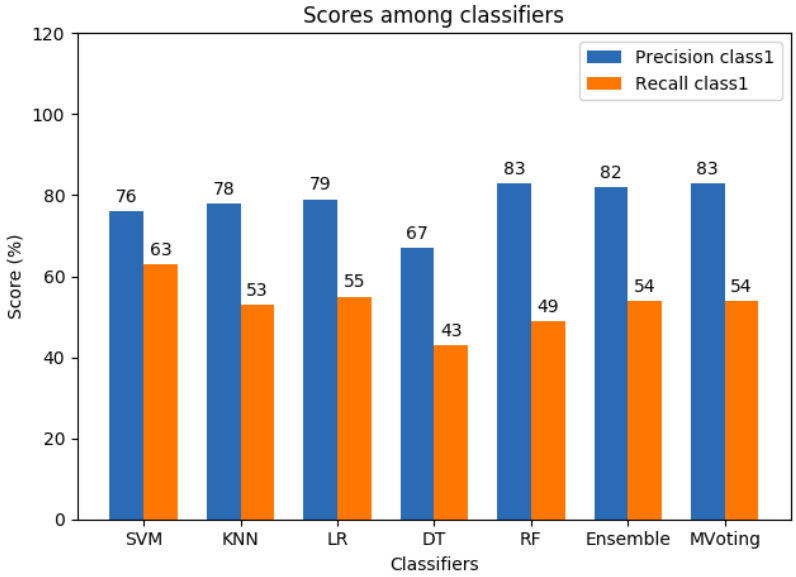

Figure 24: Precision and Recall for class1 among classifiers (Gossipcop)

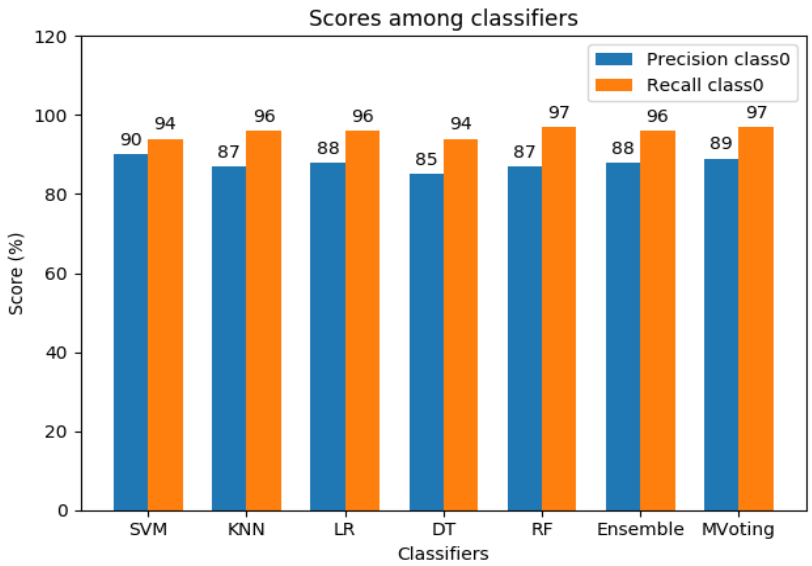

Figure 25: Precision and Recall for class0 among classifiers (Gossipcop)

Ensemble classifier does not seem to make any improvement from the highest already recorded results, probably because the percentage of classifiers' agreement in their predictions is already high, roughly 81%, but performs rather well regarding accuracy and precision measurements.

After experimenting with the combination of classifiers to be used in the ensemble model, the results that came up when removing the three classifiers that were a little weak in discovering fake articles (KNN, DT, RF), were better in respect to the ability of the ensemble classifier to find all the truthful samples (recall, F1-score) but less precise.

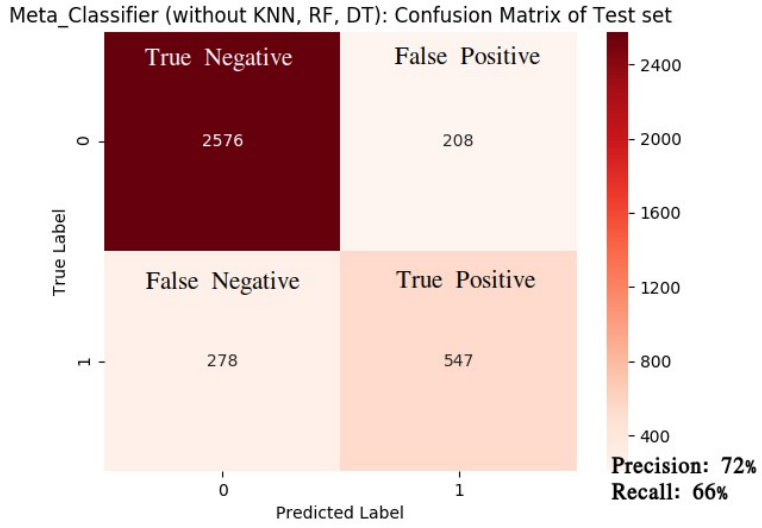

Figure 26: Confusion matrix regarding results of ensemble combining SVM/LR

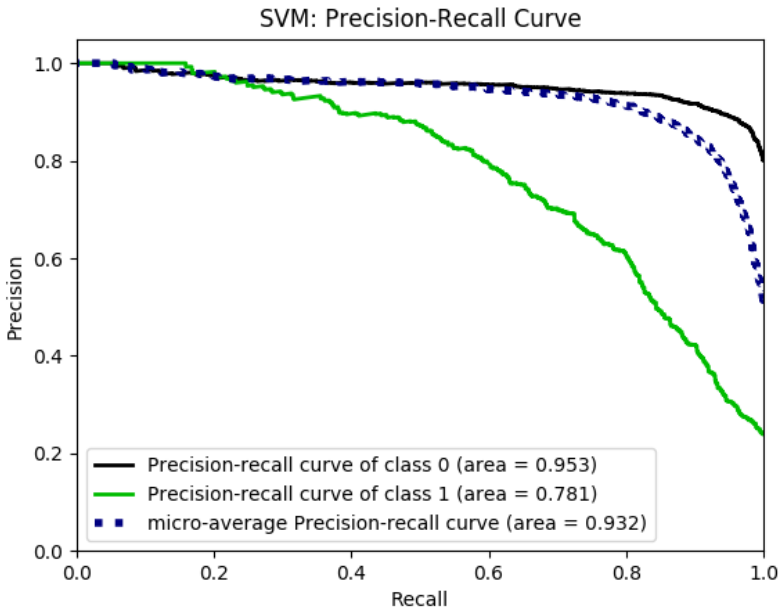

Figure 27: SVM Classifier - Precision-Recall Curve on test set (Gossipcop).

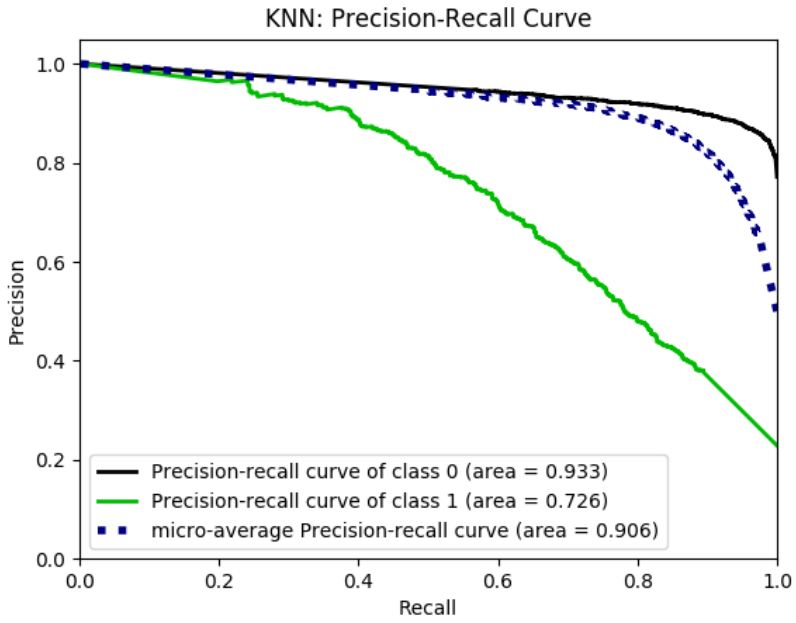

Figure 28: KNN Classifier - Precision-Recall Curve on test set (Gossipcop)

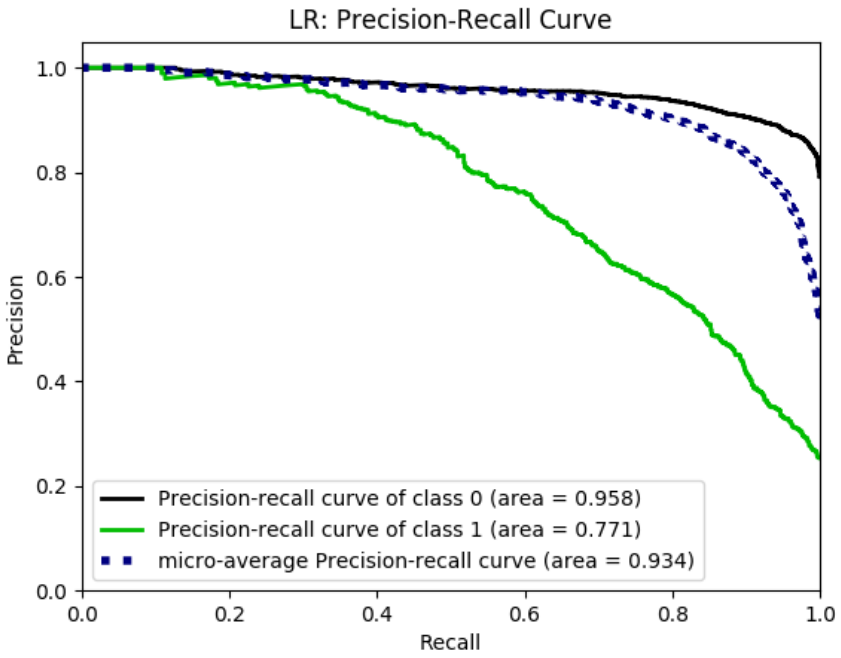

Figure 29: LR Classifier - Precision-Recall Curve on test set (Gossipcop).

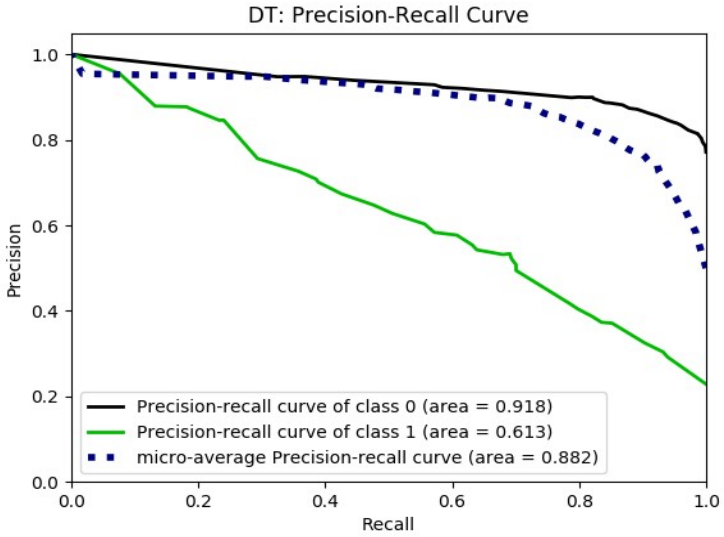

Figure 30: DT Classifier - Precision-recall Curve on test set (Gossipcop).

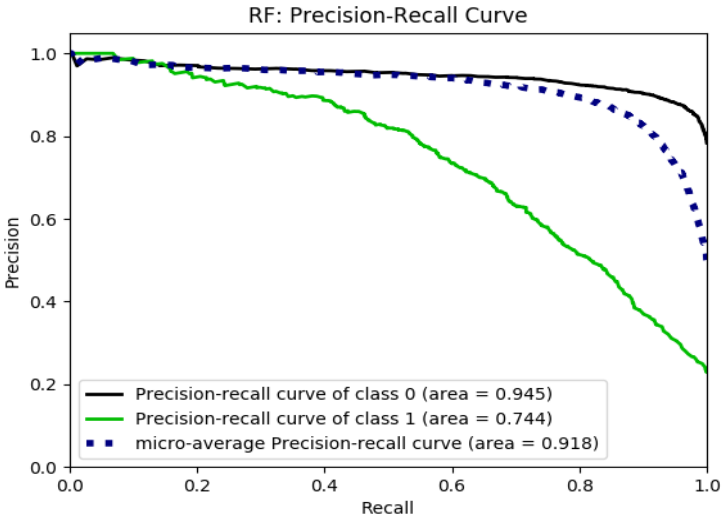

Figure 31: RF Classifier - Precision-recall Curve on test set (Gossipcop).

As shown in the above Precision-recall Curves, all classifiers have a hard time distinguishing fake articles, while they excel in discovering the truthful ones.

In general, the tuning process contributed to the overall improvement of all classifiers', except DT which seems that with default tuning generalizes better or it is because of the particular initial split of the test set, ability to separate the two classes, especially KNN and RF, although there is still room for further refinement.

Results on Balanced Version of Gossipcop

In order to check the aforementioned assumption, that lack of distinguishing ability for class 1 (fake) is due to the imbalance of the dataset, a new balanced Gossipcop dataset was created by keeping all the fake articles (4126) and adding only the same portion of true articles (4126 of 13918). So, the new balanced dataset was divided in train set (80%) and test set (20%), as before but now with a 50/50 balance of true/fake articles of the same distribution as in the previous experiments.

Results before tuning

Table 9: Results on 5-fold CV on the training set (Gossipcop_balanced).

Classifier	Accuracy	F1-score
Support Vector Machines (SVM)	0.81	0.8
K-Nearest Neighbors (KNN)	0.73	0.69
Logistic Regression (LR)	0.8	0.79
Decision Trees (DT)	0.69	0.69
Random Forests (RF)	0.78	0.77

The results on the new balanced training set at first glance seem to be improved, even before tuning, as far as F1-score is concerned. The trade-off with this improvement, though, is that overall classifiers' accuracy has dropped.

Table 10: Results on predictions made on the test set (Gossipcop_balanced).

Classifier	Accuracy	F1-score	Precision/ class0	Recall/ class0	Precision/ class1	Recall/ class1
Support Vector Machines (SVM)	0.8	0.78	0.78	0.87	0.86	0.75
K-Nearest Neighbors (KNN)	0.72	0.67	0.73	0.87	0.84	0.68
Logistic Regression (LR)	0.8	0.79	0.77	0.85	0.84	0.75
Decision Trees (DT)	0.73	0.72	0.72	0.74	0.74	0.71
Random Forests (RF)	0.77	0.77	0.76	0.81	0.79	0.74

Results after tuning parameters

Table 11: Results on 5-fold CV on the training set, with 200 components used in TruncatedSVD, before and after tuning (Gossipcop_balanced).

Classifier	Accuracy (before)	Accuracy (after)	F1-score (before)	F1-score (after)
Support Vector Machines (SVM)	0.81	0.82	0.8	0.81
K-Nearest Neighbors (KNN)	0.73	0.77	0.69	0.75
Logistic Regression (LR)	0.8	0.8	0.79	0.79
Decision Trees (DT)	0.69	0.73	0.69	0.72
Random Forests (RF)	0.78	0.8	0.77	0.79

Results on Test Set

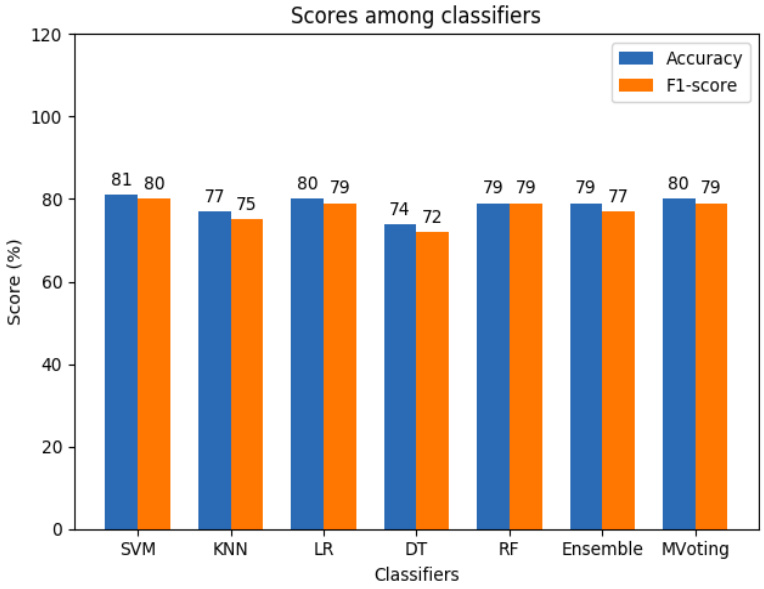

Figure 32: Accuracy and F1-score among classifiers (Gossipcop_balanced)

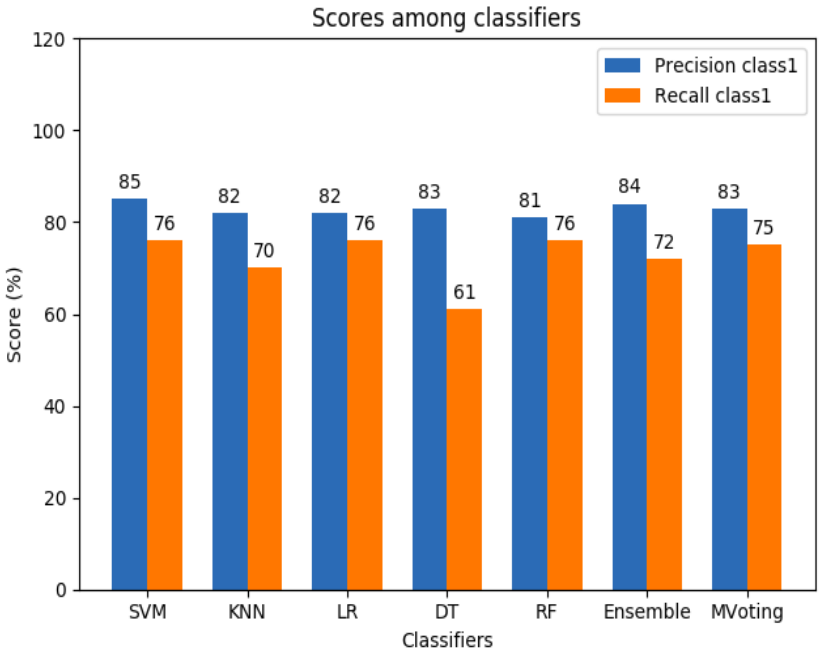

Figure 33: Precision and Recall for class1 among classifiers (Gossipcop_balanced)

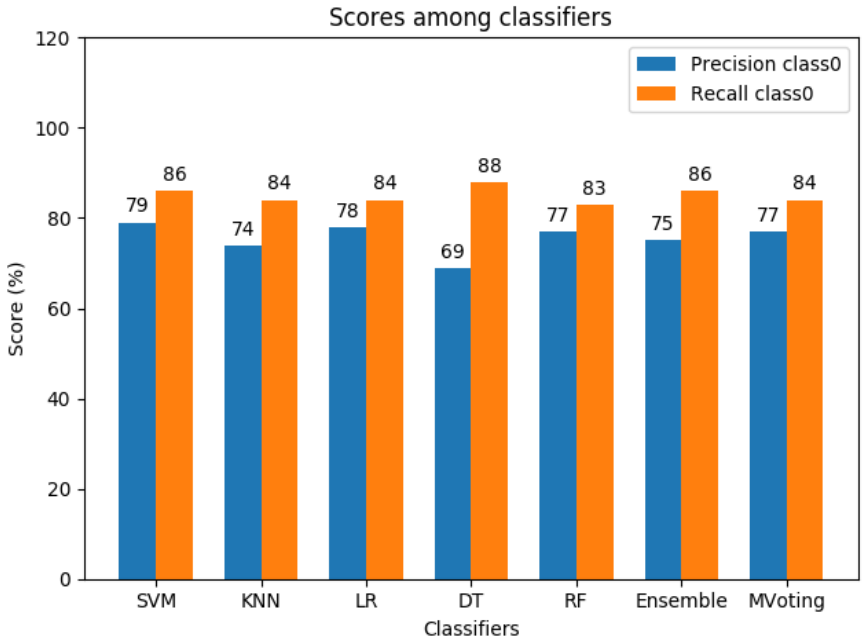

Figure 34: Precision and Recall for class1 among classifiers (Gossipcop balanced)

There is a notable difference between this balanced dataset and the original one, as far as ensemble and all conventional classifiers' ability to predict accurately the fake data, although there is still need for improvement (Figure 29). Therefore, precision, recall and F1-score metrics are improved, compared to the results of the unbalanced data, and accuracy is decreased. Former higher results in accuracy gave a disorienting impression, because of the biased behavior of every classifier in favor of the class that the majority of the data belong to.

Ensemble classifier does not seem to give better results than SVM which has the best performance among all others, but still marks one of the higher results as far as how precise it is.

4.2.3 Results on Politifact Dataset

Results before tuning

Table 12: Results on 5-fold CV on the training set (Politifact).

Classifier	Accuracy	F1-score
Support Vector Machines (SVM)	0.87	0.88
K-Nearest Neighbors (KNN)	0.53	0.0069
Logistic Regression (LR)	0.88	0.89
Decision Trees (DT)	0.78	0.77
Random Forests (RF)	0.81	0.81

Table 13: Results on predictions made on the test set (Politifact).

Classifier	Accuracy	F1-score	Precision/ class0	Recall/ class0	Precision/ class1	Recall/ class1
Support Vector Machines (SVM)	0.8	0.75	0.94	0.84	0.94	0.84
K-Nearest Neighbors (KNN)	0.53	0.03	0.52	0.99	0.5	0.01
Logistic Regression (LR)	0.86	0.87	0.94	0.85	0.94	0.85
Decision Trees (DT)	0.81	0.78	0.8	0.78	0.77	0.79
Random Forests (RF)	0.69	0.57	0.66	0.97	0.94	0.45

The poorly results in KNN Classifier is probably due to the initial number of components used in dimensionality reduction, which is set to 500. After tuning the number of features in SVD and resulting to 50 components the results were:

Table 14: Results on 5-fold CV on the training set, with 50 components used in TruncatedSVD (Politifact).

Classifier	Accuracy	F1-score
Support Vector Machines (SVM)	0.91	0.9
K-Nearest Neighbors (KNN)	0.85	0.85
Logistic Regression (LR)	0.9	0.9
Decision Trees (DT)	0.78	0.76
Random Forests (RF)	0.9	0.89

Results on Test Set

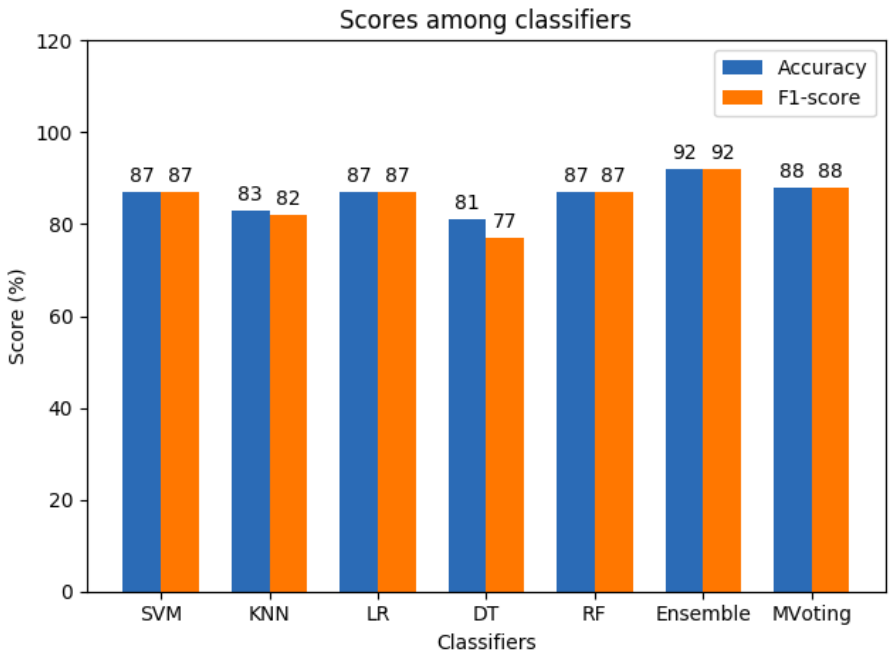

Figure 35: Accuracy and F1-score among classifiers (Politifact)

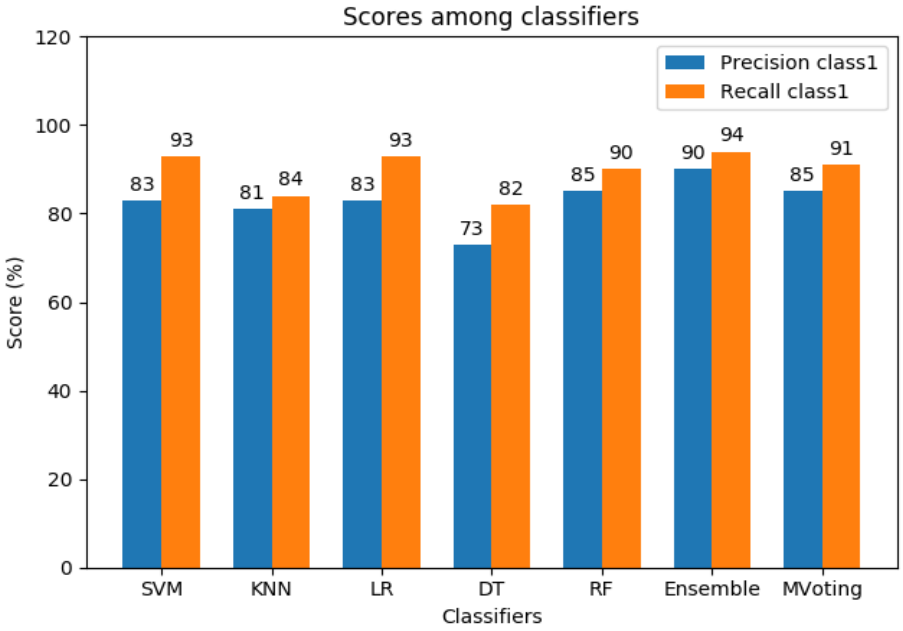

Figure 36: Precision and Recall for class 1 among classifiers (Politifact)

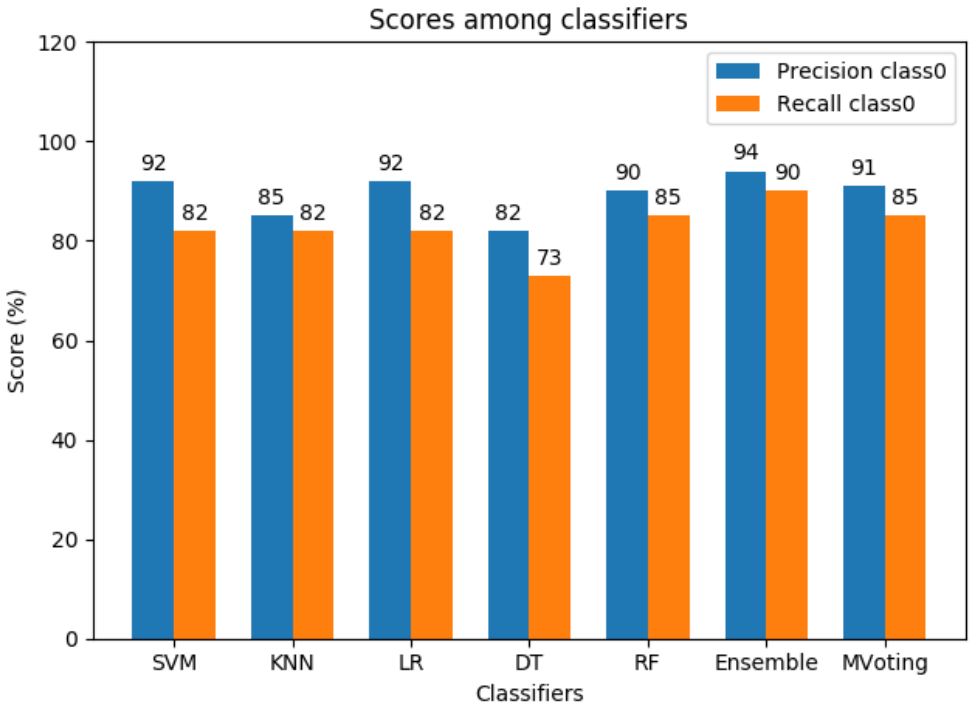

Figure 37: Precision and Recall for class 0 among classifiers (Politifact)

It seems that ensemble classifier improves the overall performance and gives best results compared to the rest classifiers. This could be happening mainly because the predictions of the classifiers agree at roughly 69%, leaving room for the meta-classifier

to combine the different probabilities that the models produce and end up to the most accurate one.

4.3 Mixing Classifiers and Datasets

This section summarizes the experiment with the Fake News Net Dataset (Gossipcop/Politifact) to use models that were particularly tuned for a dataset to the other one in order to observe the generalization ability of a model on completely different test data.

4.3.1 Evaluating Gossipcop Dataset using Politifact Models

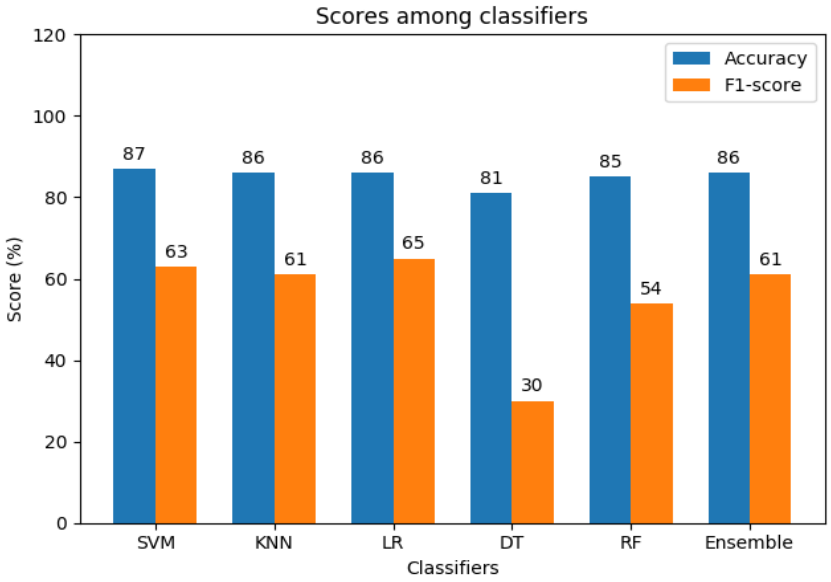

Figure 38: Politifact models performance (acc/f1) on Gossipcop data.

Overall, accuracy and F1-score (Figure 33) are retained on almost the same levels in every classifier, including ensemble, except DT and RF Classifiers' ability to predict the fake samples accurately (low Recall and F1-score). Something that needs to be noted, is that precision improved in every classifier that maybe indicates politifact models are managing to distinguish truthful articles without confusing them with fake ones in unseen data.

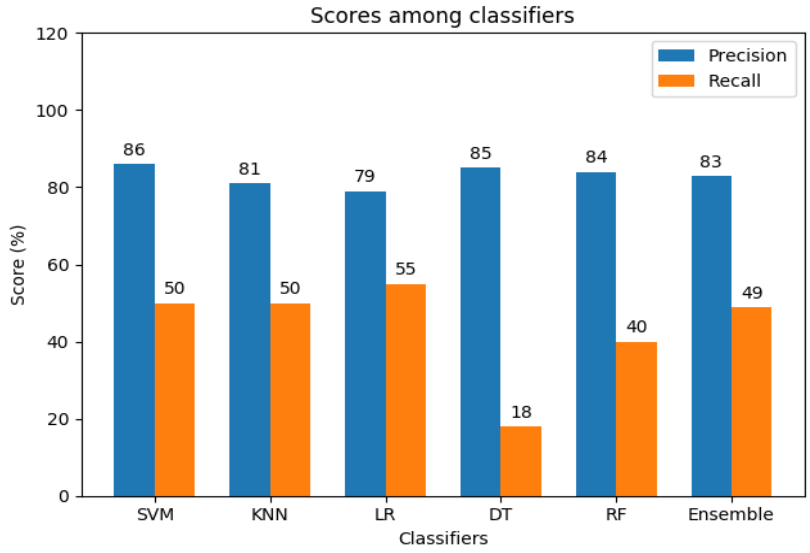

Figure 39: Politifact models performance (pre/rec) on Gossipcop data.

4.3.2 Evaluating Politifact Dataset using Gossipcop Models

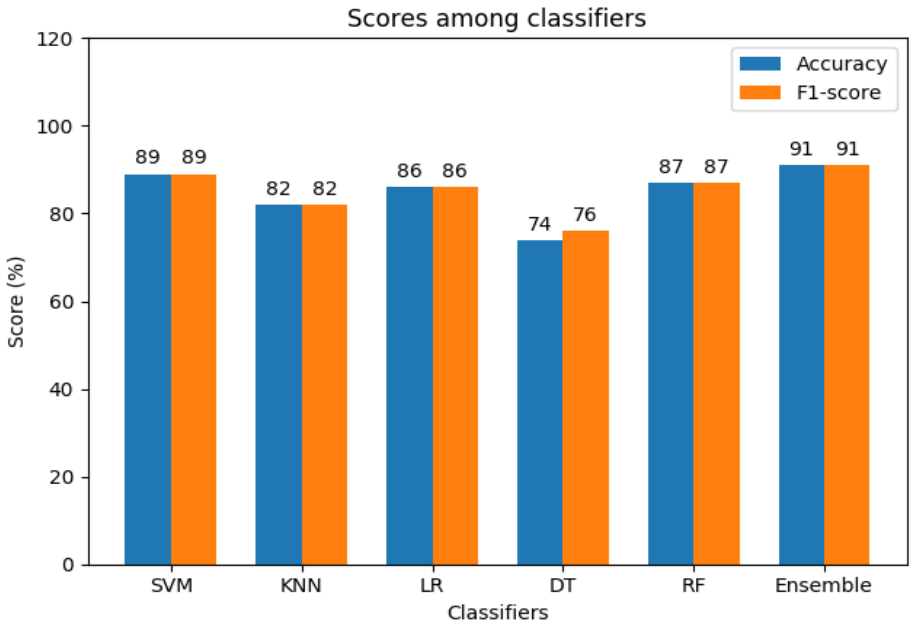

Figure 40: Gossipcop models performance (acc/f1) on Politifact data.

All models seem to perform well on unseen data and especially SVM have improved the results of the politifact models seeming to generalize better than the others.

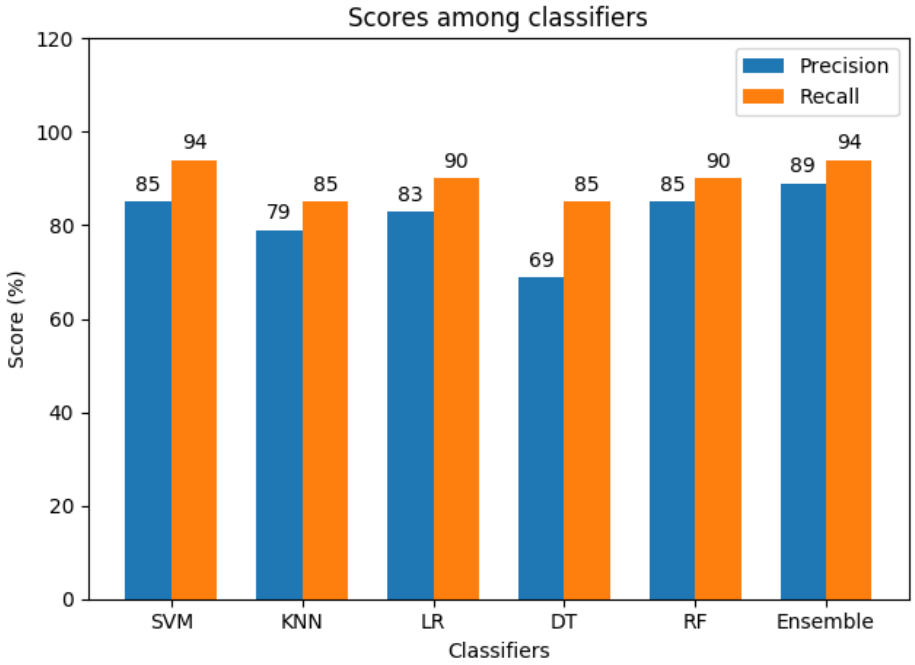

Figure 41: Gossipcop models performance (pre/rec) on Politifact data.

4.4 Results for LSTM

The LSTM model was implemented only for Fake News Net Datase because ISOT Dataset could be characterized as skewed, as it is pointed out already, and demands a lot of computational power as it is an extremely large dataset. For both datasets batch size was set to 50, sequence length to 200 and the criterion used for calculating the loss is chosen to be Binary Cross Entropy Loss[23] (BCELoss), suitable for binary classification problems, and Adam optimizer for optimization.

The process of generating batches is done by random sampling the documents. This means that in each batch every training example is a random slice of the initial document of size 200 (sequence length) which in every epoch changes resulting to "feeding" the whole document eventually. If initial document is shorter than the sequence length, training example is padded with zeros at the end.

4.4.1 Balanced Gossipcop

After trying different combinations of hyperparameters, training for 50 epochs with learning rate 0.0001, dropout rate 0.3 in LSTM layers as well as dropout rate 0.6 after

[23] https://pytorch.org/docs/master/generated/torch.nn.BCELoss.html

LSTM layers, validation and training loss keep reducing, with minor fluctuations, until ~40th epoch where validation loss started raising (Figure 38).

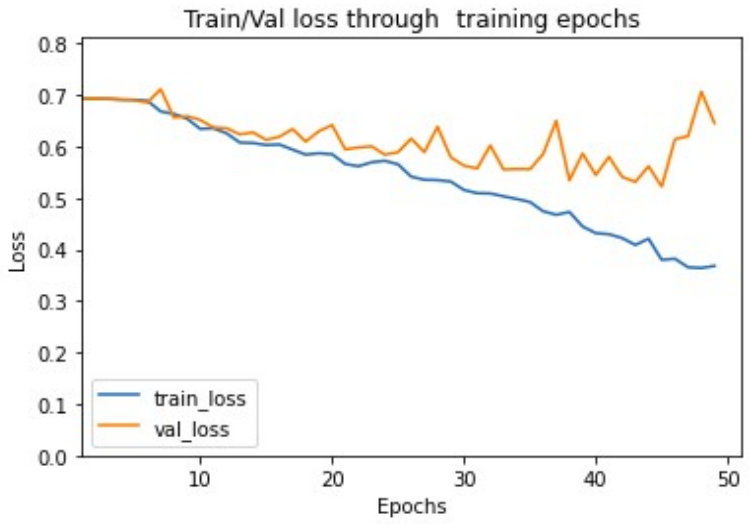

Figure 42: Loss through the training epochs (Gossipcop).

The loss that was marked during evaluation of test set was ~0.62.

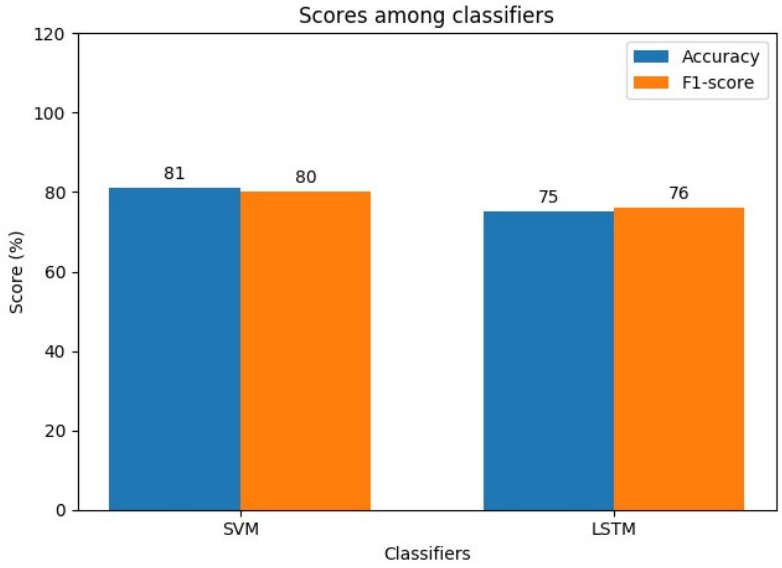

Figure 43: Comparison between LSTM and SVM on Gossipcop (acc, f1).

The results of the LSTM model indicate that the neural network performs well on the dataset but does not reach the ability to distinguish the patterns between the words as accurately as SVM Classifier does, which gives best results among the rest statistical ML classifiers.

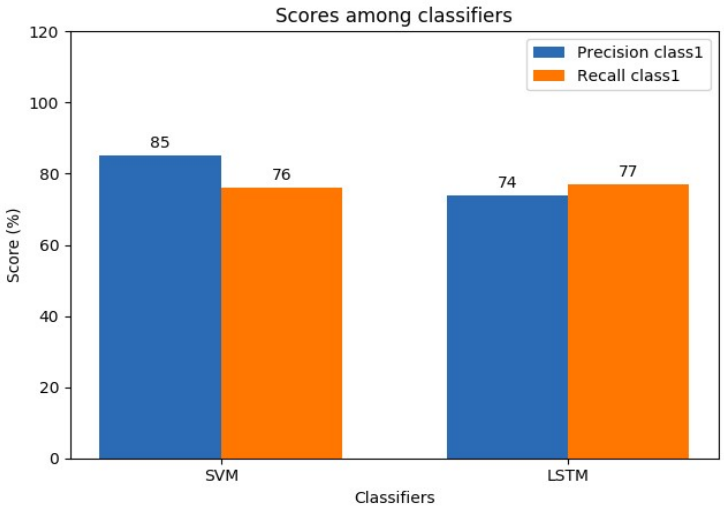

Figure 44: Comparison between LSTM and SVM on Gossipcop (pre, rec for class1).

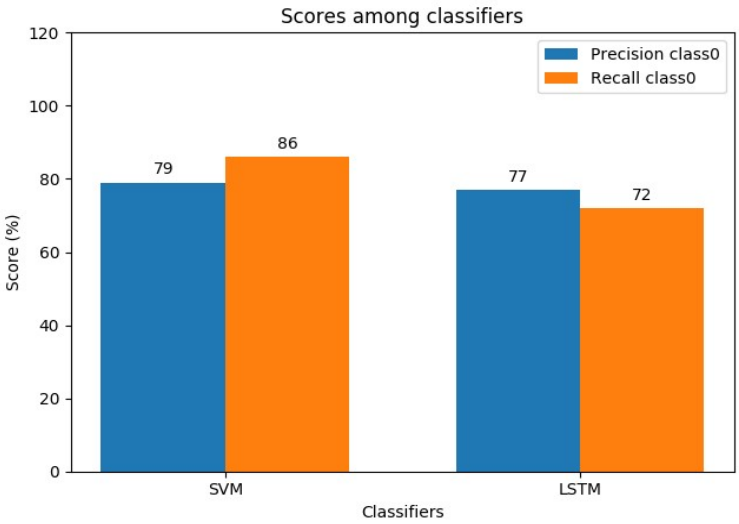

Figure 45: Comparison between LSTM and SVM on Gossipcop (pre, rec for class0).

4.4.2 Politifact

The training of LSTM model for the Politifact dataset concluded in learning rate of 0.0001, dropout rate 0.3 in LSTM and dropout rate 0.6 after the LSTM layer.

Figure 46: Loss through the training epochs (Politifact).

At first 20 epochs of training, both losses remain steady at 0.7 and then start to reduce with some fluctuations meanwhile.

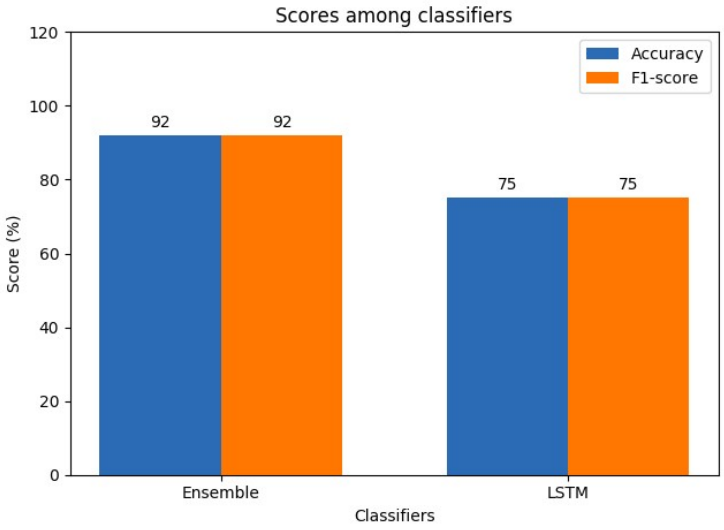

Figure 47: Comparison between LSTM and Ensemble classifier on Politifact (acc, f1).

The test loss during the test evaluation was ~0.62 and the general performance of the neural network is satisfying but does not reach the ability of the ensemble classifier to predict in such accurate way.

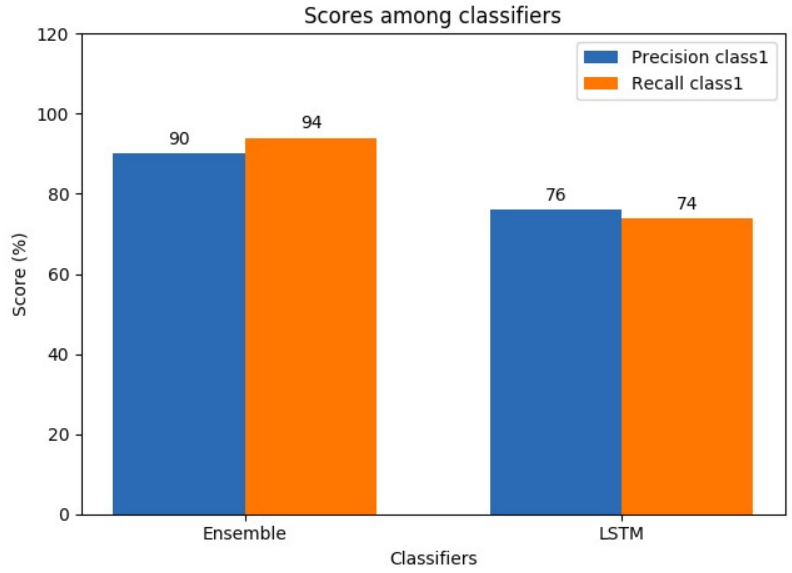

Figure 48: Comparison between LSTM and Ensemble classifier on Politifact (pre, rec for class1).

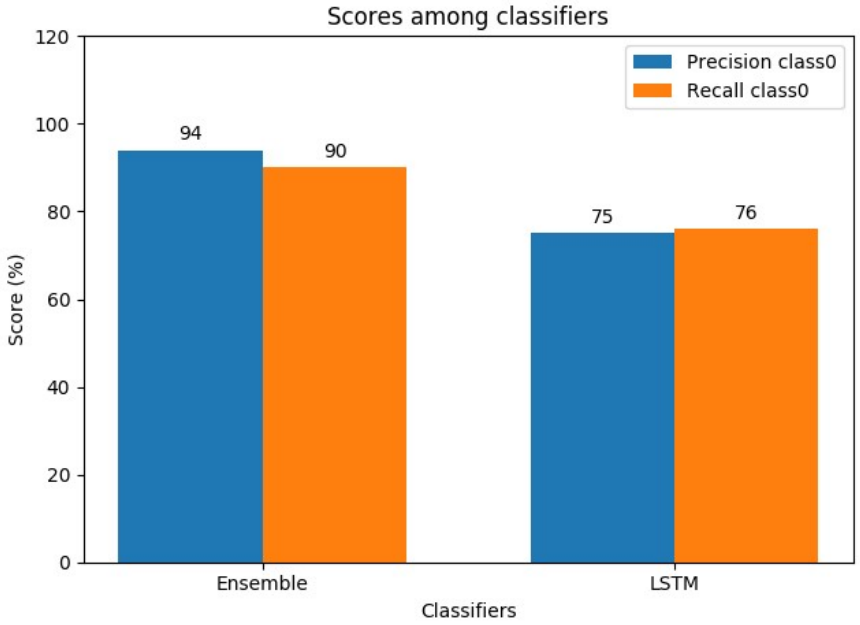

Figure 49: Comparison between LSTM and Ensemble classifier on Politifact (pre, rec for class0).

5. CONCLUSION

5.1 Discussion on Results

The objective of this thesis was to gather, construct and assert fake news datasets concerning how well-structured and representative are, as well as to test the behavior of various machine learning classifiers, as well as a simple LSTM neural network architecture, towards distinguishing truthful and fake articles among these datasets.

The results in the previous section has shown that one of the most significant tasks in machine learning is picking the right dataset, as the results on ISOT dataset were unexpectedly impressive even before tuning the classifiers, indicating a very targeted and easy-to-learn dataset which cannot be safely trusted to evaluate the models. As for the Gossipcop dataset, it is shown that the balance of the two classes is a key factor for better results in the ability of the classifier to separate them.

The experiment of the ensemble classifier seems to give overall good results but there is only improvement if the classifiers that are combined do not have a really high percentage of agreement in their predictions, as it happens with the Politifact dataset in which ensemble model surpasses the rest classifiers.

The attempt to try the models that were trained on a dataset to another gave some interesting results. The Politifact models seem to have a generalization ability to find the truthful articles among the Gossipcop dataset as Precision metric is increased, whereas the Gossipcop models seem to have overall good results but only SVM classifier improves the overall performance of the Politifact equivalent model.

Applying deep learning techniques in the two most reliable datasets in my disposal, showed that a simple neural network architecture, which takes advantage of the sequence of the words provided having the ability to “remember” for a certain “time window”, gives promising results. However, it did not surpass the the ML classification methods used in this project, which is maybe the result of datasets being structured in a way that the sequence of data may not play such important role as the selection and particular linguistic features of a word itself.

5.2 Future Work

Machine Learning in such aspects of social life such as detection of fake news pieces plays an important role and leaves a lot of room for more improvements in the future as the data collected are increasing exponentially day by day leading to the training of more accurate models. Therefore, the construction of well-balanced and representative datasets that can capture the essence of a wide spectrum of text is a matter that needs to be taken into consideration.

This project focuses on the linguistics characteristics of the news piece and the selection of words in each text to train the classifiers. An expansion in this approach is to combine these features with the exploitation the user social engagements on text (up-votes, number of shares, etc.) on the training process. Deep learning techniques applied in this work aim also to take into account the correlation between words concerning their position in text and preserve the punctuation, contrary to ML techniques. An interesting approach would also be to combine statistical machine learning techniques with deep neural networks as an ensemble model.

Overall, natural language processing and data science fields in service of fake news detection can reveal patterns to better understand the data available and utilize them in best way to promising results.

ABBREVIATIONS - ACRONYMS

CSV	Comma-separated values
DT	Decision Trees
ISOT	Information Security and Object Technology
KNN	K-Nearest Neighbors
LR	Logistic Regression
LSTM	Long short-term memory
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network
PCFG	Probability Context Free Grammars
RF	Random Forest
SVM	Support Vector Machine
SVD	Singular Value Decomposition
TF-IDF	Term Frequency-Inverted Document Frequency

REFERENCES

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

[2] Victoria L. Rubin, Chen, Y. & Conroy N. J. Deception detection for news: three types of fakes. Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community, 2015. American Society for Information Science, 83.

[3] Shu, K., Bernard, H.R., & Liu, H. (2019). Studying Fake News via Network Analysis: Detection and Mitigation. ArXiv, abs/1804.10233.

[4] Ahmed, Hadeer & Traore, Issa & Saad, Sherif. (2017). Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. 127-138. 10.1007/978-3-319-69155-8_9.

[5] Conroy, N.J., Rubin, V.L., Chen, Y.: Automatic deception detection: methods for finding fake news. In: Proceedings of the Association for Information Science and Technology (2015)

[6] “Scikit-plot Documentation.” [Online]. Available: https://scikit-plot.readthedocs.io/en/stable/.

[7] “Tokenization” [Online]. Available: https://en.wikipedia.org/wiki/Lexical_analysis#Tokenization.

[8] “Stemming” [Online]. Available: https://en.wikipedia.org/wiki/Stemming.

[9] “Dimensionality Reduction – Stanford InfoLab” [Online]. Available: http://infolab.stanford.edu/~ullman/mmds/ch11.pdf

[10] Rohith Gandhi, “Support Vector Machine — Introduction to Machine Learning Algorithms”, towardsdatascience, 2018, [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[11] Dhilip Subramanian, “A Simple Introduction to K-Nearest Neighbors Algorithm”, towardsdatascience, 2019, [Online]. Available: https://towardsdatascience.com/a-simple-introduction-to-k-nearest-neighbors-algorithm-b3519ed98e

[12] Ayush Pant, “Introduction to Logistic Regression”, towardsdatascience, 2019, [Online]. Available: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

[13] Rajesh S. Brid, “Decision Trees—A simple way to visualize a decision”, medium, 2018 [Online]. Available: https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

[14] Tony Yiu, “Understanding Random Forest”, towardsdatascience, 2019, [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[15] Sanjay.M, “Why and how to Cross Validate a Model?”, towardsdatascience, 2018, [Online]. Available: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f

[16] Dhruvil Karani, “Introduction to Word Embedding and Word2Vec”, towardsdatascience, 2018, [Online]. Available: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

[17] “Long short-term memory” [Online]. Available: https://en.wikipedia.org/wiki/Long_short-term_memory