EVALUATION OF MACHINE LEARNING CLASSIFIERS IN ORDINAL MULTICLASS FAKE NEWS DETECTION SCENARIO

Igor Bichara de Azeredo Coutinho

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Carlos Eduardo Pedreira
Geraldo Bonorino Xexéo

Rio de Janeiro
Novembro de 2019

EVALUATION OF MACHINE LEARNING CLASSIFIERS IN ORDINAL MULTICLASS FAKE NEWS DETECTION SCENARIO

Igor Bichara de Azeredo Coutinho

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____
Prof. Carlos Eduardo Pedreira, Ph.D.

_____
Prof. Priscila Machado Vieira Lima, Ph.D.

_____
Prof. Carla Amor Divino Moreira Delgado, D.Sc

RIO DE JANEIRO, RJ – BRASIL
NOVEMBRO DE 2019

Coutinho, Igor Bichara de Azeredo

Evaluation of Machine Learning Classifiers in Ordinal Multiclass Fake News Detection Scenario/Igor Bichara de Azeredo Coutinho – Rio de Janeiro: UFRJ/COPPE, 2019.

XI, 58 p.: il.; 29,7 cm.

Orientadores: Carlos Eduardo Pedreira

Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 51-58.

1. Fake News Detection. 2. Ordinal Classification. 3. Fake News Feature Extraction. I. Pedreira, Carlos Eduardo *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha família e amigos.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

AVALIAÇÃO DE CLASSIFICADORES DE APRENDIZADO DE MÁQUINA EM CENÁRIO MULTICLASSE DE DETECÇÃO DE FAKE NEWS

Igor Bichara de Azeredo Coutinho

Novembro/2019

Orientadores: Carlos Eduardo Pedreira
Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Essa dissertação tem como objetivo avaliar classificadores de aprendizado de máquina e suas técnicas no problema de detecção de fake news. Algoritmos preditivos nesse contexto podem produzir resultados diferentes de acordo com a variância da rotulação de datasets causada pela ambiguidade e subjetividade da semântica textual.

O dataset LIAR foi utilizado nos experimentos desta dissertação. Este dataset foi criado a partir de dados da agência de checagem de fatos PolitiFact que consiste em rótulos com 6 classes ordinais que por sua vez posicionam as declarações políticas no intervalo entre completamente falsa e completamente verdadeira. O experimento original do autor do dataset alcançou 27.4% de acurácia usando redes neurais híbridas com camadas convolucionais CNN e recorrentes LSTM bidirecionais.

A contribuição principal deste trabalho consiste na avaliação de classificadores mais simples usando diferentes técnicas de pré-processamento e seleção de atributos. Além disso, o trabalho explora a natureza ordinal das classes usando um método *ensemble* de classificadores binários já estabelecido na literatura.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

EVALUATION OF MACHINE LEARNING CLASSIFIERS IN ORDINAL MULTICLASS FAKE NEWS DETECTION SCENARIO

Igor Bichara de Azeredo Coutinho

November/2019

Advisors: Carlos Eduardo Pedreira
Geraldo Bonorino Xexéo

Department: Systems and Computing Engineering

This thesis intends to explore machine learning classifiers and techniques to address the problem of fake news detection. Prediction algorithms can generate different results in this problem due to variance in dataset labeling caused by ambiguity and subjectivity of semantic text.

The LIAR Dataset was used in the experiments of this thesis. This dataset derived from PolitiFact fact-checking agency data which is composed of a 6-class ordinal labeling that places political statements in the range between completely false and completely true statements. The original experiment that created the dataset achieved 27.4% class accuracy using hybrid CNN and Bi-Directional LSTM networks.

The main contribution of this work consists of evaluating simpler classifiers focusing on using different preprocessing and feature selection techniques when modeling metadata and text features. Furthermore, this work explores the ordinal characteristics of the class labels and uses simple binary classifiers in an ordinal ensemble method already established in the literature.

Summary

Figure List

Table List

1 Introduction

This work intends to explore machine learning techniques to tackle the fake news detection scenario focusing on its major underlying complications: ambiguity and subjectivity. Machine learning classifiers have achieved different results varying according to dataset subject, dataset labeling methods and, of course, the context in which the statement was captured.

The experiments of this work used the LIAR Benchmark Dataset [1]. This dataset will be described in detail further on and it has six ordinal labels of truth. It is important to notice that evaluating a multilabel ordinal classifier just by its accuracy can be misleading [2]. Appropriate ordinal formulations are expected to perform better in problems with high ambiguity such as misinformation detection. One of such formulations is well established in the literature [3] and will be better explained in section 3.4.2.

The experiments intend to support answers to the following questions:

1. How can simpler machine learning algorithms perform in the fake news detection problem compared to LSTM Networks, which are commonly placed as state-of-art for text classification?
2. How does an ordinal formulation improve classification results in an ordinal multiclass classification problem?

1.1 The Problem

Detection of fake news is now a common research topic that has emerged due to their impact on society. The digital transformation changed the way people consume news articles. Researches in the U.S. [4] showed that consumption of news through social media already surpassed the consumption via print newspaper. From 2016 to 2018, the users who often consumed news in websites raised from 28% to 33%; social media numbers went from 18% to 20%; print newspapers fell from 20% to 18%. The same studies, depicted in Figure 1, show that, for young adults (18-29 years old), social media is the dominating platform.

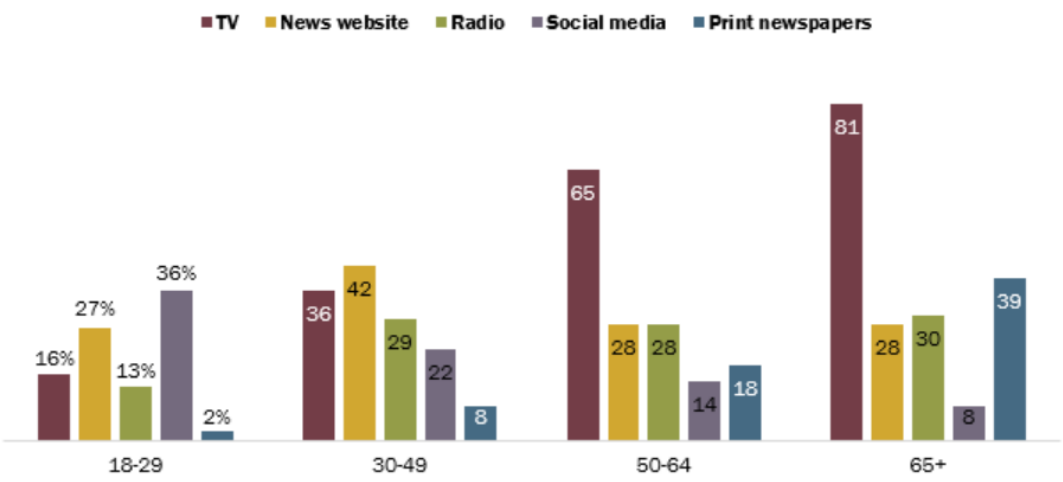

Figure 1 – Percentage of each age group who often get news on each platform: Pew Research Center Survey of U.S. adults. Jul30 - Aug12, 2018

It is common for most social media users to comment daily news online, exposing their personal point of view. Others may only inform themselves in social networks, spreading these comments even further. For most people who are not public figures, shared information has very little compromise with the truth. However, even satires or personally biased comments about news can be a polarizing factor in social media. A particular misinformation can fall into different categories of fake news. Those categories are discussed in in section 1.3.

When discussing politics, or mostly any conflicting subject, it is common for the opinions to diverge and polarize. Studies show [5] that people pay more attention to media that share their beliefs. This effect is inflated further by personalizing algorithms, or "media bubbles". In the context of social media posts, people are more influenced by opinions that they agree with, while the opposite opinions, although present, draw much less attention.

With society polarized into extremes, it becomes easier for ill-intended people to take advantage and feed the social networks with biased news. In that context, the story does not need to be true to get attention.

Some studies support [6] that the results of the 2016 USA`s presidential election were influenced by the amount of misinformation present on social networks. The election was characterized by an abundance of fabricated content, mostly partisan biased towards one way or the other. Similarly, the elections in Brazil [7] and India [8] were also marked by fake stories going viral on social networks and chat apps. These phenomena

have increasingly caught the attention of news agencies, broadcasters, media companies in general, as well as the scientific community.

According to The Reporter`s Lab at Duke University [9], fact-checking has grown consistently from 2014 to 2018, which is depicted in Figure 2. The number of organizations has risen from 44 to 149 in a steady pace. Elections and politics in general are the biggest concern of these organizations.

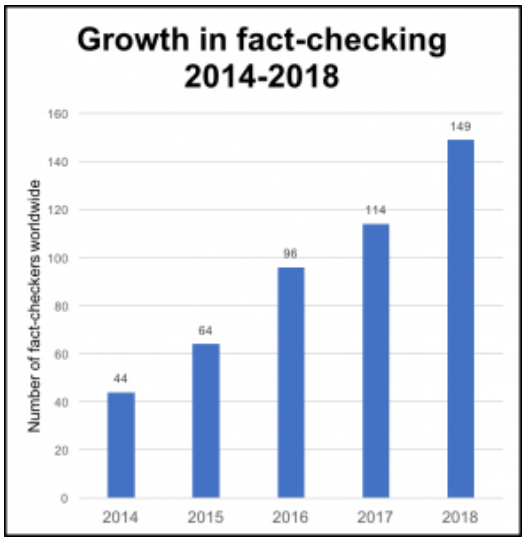

Figure 2 – Number of fact-checking organizations in the last years

Fact-checking agencies have been making partnerships with social network corporations to help addressing the spread of fake news. Facebook, for instance, lets users flag content as false and submits such content to professional fact-checkers [10]. In case false information is confirmed, its ranking in the News Feed is significantly penalized. This feature reduced the natural spread of false content by 80% in the USA [11]. It is also functional in Brazil with the help of *Aos Fatos* and *Agência Lupa* organizations. Recently, similar features were integrated to Instagram app as well. Facebook has also been banning inauthentic accounts that spread political misinformation [12], usually secretly linked to organizations or political parties that benefit from it. Similarly, Twitter has been taking measures against bot accounts [13] that increase the exposure of low-credibility stories by exploiting high influence users, either mentioning them or commenting on their posts.

The scientific community has also risen its eyes to the issue. A simple search for "fake news, false news or inaccurate news" on *IEEE Xplore* returns an increasing number of papers published after 2016. Those numbers, shown in Table 1, also serve as additional evidence that the aftermath of USA Presidential Elections was alarming.

Table 1 – Rising numbers scientific papers addressing fake news

Year	Number of IEEE Explore Papers
2012	1
2013	1
2014	2
2015	1
2016	0
2017	32
2018	71
2019 (Jan-Sep)	64

Great part of those papers comes from the Computer Science field and experiments with text datasets creating machine learning classifiers to predict false content. Some others [14] are less optimistic and place the misinformation problem as a task for both human and machine to solve together. There are also some studies [15] that consider using new technologies like Blockchain to fight the problem in a different way.

1.2 False Information Locations

The information flowing through internet chats and social network is usually composed of not only text but also audio, image and video. A study detailed this effect further [16], calling it a “truthiness” effect. The experiments suggest that people tend to process photos, even nonprobative ones, as pseudo-evidence to support claims.

It is very alarming to see some images edited digitally or just taken completely out of context to spread false information [17]. Some cases, however, have the potential to cause great impact. As an example, in Brazil’s 2018 Presidential Election, the image shown in Figure 3 included the former president Lula, that had big public approval rate, with the voting number of the opposing candidate Bolsonaro.

Figure 3 – False election campaign image in Brazil

In the USA, a photo of the 2016 presidential candidate Hillary Clinton, shown in Figure 4, was taken out of context to suggest her health was failing. In reality, she was just being aided after stumbling as she climbed the steps.

Figure 4 – Photo of Hillary Clinton stumbling was used to suggest health issues

Image editing and fabrication is a common thing these days. Criticism and common sense have been more carefully used by viewers to judge the truthiness of an image. However, with the advances of Deep Learning, fabricating misleading audio and video has become easier.

Manipulating video is nothing new for the cinema industry. However, doing that using traditional methods took time, skill and money. With the increasing advances of techniques such as *Deepfake* [18], anyone could create false audios or videos once the algorithm is trained. That means having any face appear, say or do anything in a video without that person actually doing any of those things. Even though CNN experts argue

that the technology is not yet sophisticated enough [18], they worry the doubt alone is already enough to alter trust in audio and video for good.

Even though this work focuses its efforts in modeling false information detection using text, it is important to remark the convincing power that image, audio and video have when it comes to altering information's trust.

1.3 Fake news characteristics

Humans receive and interpret information in different ways making it difficult to model the problem of misinformation in a systematic way. In a learning problem, the patterns are captured from the data. Results will be poor if the model is fitting biased data [19].

In the polarized environment of the web, it has become increasingly harder to identify false from real news. The term "fake news" has become a tool or weapon for anyone to use against contradictory opinions or stories [20]. Adding to that, an experimental study from the University of Texas [21] has concluded that simply exposing the reader to the term "fake news" in a story`s comments reduces the accuracy of identifying real news.

With the intention of exploring the actual meaning of the term "fake news", Claire Wardle wrote an article [22] identifying the different types of *misinformation* and *disinformation*. They are described below in Table 2.

Table 2 – Simple fake news breakdown

Misinformation	Inadvertent sharing of false information
Disinformation	Deliberate creation and sharing of information known to be false

The same article also presents the 7 types of misinformation in Table 3 that follows. They are categorized according to their intentions and potential to deceive people. Even though they sit on an increasing scale in terms of false content, the borders between categories can be subjective and ambiguous.

Table 3 – Types of fake news by Claire Wardle

Satire or Parody	No intention to cause harm but has potential to fool
False Connection	When headlines, visuals or captions don't support the content
Misleading Content	Misleading use of information to frame an issue or individual
False Context	When genuine content is shared with false contextual information
Impostor Content	When genuine sources are impersonated
Manipulated Content	When genuine information or imagery is manipulated to deceive
Fabricated Content	New content is 100% false, designed to deceive and do harm

The choice of the LIAR dataset for the experiments of this work was supported by this definition of the problem of fake news. The dataset labels of each statement follow an ordinal scale which could just as well be adapted to the same 7 types described in Table 3.

2 Methodology

Automatic machine learning fake news detection is still a work in progress and there is plenty of different angles the problem can be addressed from. Despite that, some conclusions appear very consistently when trying to detect misinformation. This section will briefly cover these commonly used techniques when approaching text fake news detection with machine learning. It will also introduce the methods from the literature applied to the experiment of section 3 when modeling the fake news detection in an ordinal manner.

Recent studies seem to agree that a rough distinction between true and false is unrealistic and their framework relies in multiple levels of "fakeness" [23]. Another very important concept in any machine learning algorithm but particularly problematic in fake news context is avoiding dataset bias [24]. Most fake news datasets are still under construction or are restricted to a specific time and subject and huge bias can be introduced when using them outside their domain. That is also true when it comes to the LIAR dataset, since its data is restricted to politics and its metadata could not be used outside this context.

Another survey on fake news detection [25] remarked the importance of image content in deceiving readers. The same survey listed some of the available and popular fake news datasets showing that most of them still mainly use text features. For this reason, text feature extraction techniques are still of major importance in this type of algorithms.

2.1 Text Feature Selection Techniques

Numerical inputs are more usual in machine learning classifiers. One iconic and famous dataset is the Boston Housing Prices dataset [26]. In this dataset, for example, the features are a vector of numerical information such as number of rooms, size or neighborhood crime rate. These features can be normalized and fed to a classifier directly. With image features, the MNIST dataset [27], a dataset of handwritten digits, the features fed to the classifier are usually the pixels` luminance value flattened in a vector. Slightly more complex image features can be used, such as color histogram, contrast or edge information.

Text features also need to be converted to numerical vector information and there are numerous well-established techniques to derive valuable information from text data. With

the increasing use of text as features for machine learning classifiers, fields like Information Retrieval, Information Extraction and Natural Language Processing blended together as tools for text feature extraction. Even though their borders can be blurred they can be defined independently [28].

The most trivial way used to represent text in numerical form is by the frequency of words. This classic type of model is called *bag-of-words* [29] and can evolve in terms of complexity. It can be as simple as using binary presence of words as features or more complex such as calculating a sentence vector feature by averaging n-gram [30] features together [31].

A usual way to apply the *bag-of-words* model is to use TF-IDF [32] weighting to score the words or n-grams in each document. This way, frequent terms contribute positively to its relevance in a document while terms that are too ordinary in the collection contribute negatively. Even though the result is a high-dimensional sparse feature matrix, it can still be used for classification after applying proper dimensionality reduction techniques [33]. TF-IDF is frequently used to calculate similarities between texts by comparing its feature tokens scores. One recent example of such applications is detecting relevant video events using closed caption and video synopsis [34]. Examples of python programming libraries implementing TF-IDF are *Scikit-Learn* [35] and *Gensim* [36].

A different, popular and effective way to vectorize text is *word embeddings* [37]. With this approach, the result is a dense low dimensional vector carrying the semantic context information of the term. Along with LDA [38] it became a widely used tool for Topic Modeling [39] applications. The experiment that gave birth to the LIAR Dataset [1] uses word embeddings to represent the text features. Another recent example is the tool called *FakerFact* [40] which uses embeddings and neural networks addressing one small piece of the text at a time detecting characteristics that might suggest false information such as sensationalism, satire and personal opinion. *Gensim* library has word embeddings model implemented, including the extension *Doc2Vec* [41] capable of producing vectors for whole sentences or paragraphs.

Apart from the words, the text also has a few other characteristics to offer as features. It is possible to expose the entities of the text to a classifier using Named Entity Recognition [42]. Additionally, the part-of-speech tagging distribution can provide hints on the text style and therefore be useful as a feature in fake news classification [43]. All these text feature extractions are well established and can be applied by NLP libraries such as *nltk* [44] or *spacy* [45].

2.2 Dimensionality Reduction

The way text is represented as a vector defines its dimensionality and therefore, some of these methods are also a dimensionality reduction technique. LDA for example transforms a large vocabulary vector space into a new space based on its latent topics. *Doc2Vec* can produce a low dimensional vector trained with context words. On the other hand, when text representation is high dimensional, such as in the case of bag-of-words, additional dimensionality reduction is required for classification algorithms to be feasible.

Correlation based feature selection [46] is a commonly used to extract the most correlated features with the output classes while uncorrelated with another feature. Latent Semantic Indexing, a traditional text dimensionality reduction method, applies the concept of Principal Component Analysis [47] to text, using SVD [48] to cluster semantic-related terms together in principal latent components, reducing dimensionality.

Particular domain knowledge of text data can be used along with what's called corpus geometry in [49] to reduce dimensionality of text in order to visualize it in 2D or 3D. T-SNE technique [50] uses random walks to map each datapoint to a two or three-dimensional map and is also widely used to visualize high dimensional data in general. A similar technique is Neighborhood Component Analysis [51], performing an n-dimensional transformation, with n being as low as wanted, such that neighborhood classification performs well.

Here, we opted for reducing the dimensionality of the feature matrix in three different ways and compare its results.

2.3 Text-Oriented Classification Algorithms

Simple classifiers such as Decision Trees, SVMs or KNNs can be applied to predict the classes of texts after they have been modeled into feature vectors appropriately. This will be the approach of this work to evaluate both ordinal class and non-ordinal class formulations.

However, some neural network architectures have proved to be more efficient when applied to text classification. Recurrent Neural Networks have increased in popularity with the advances in deep learning. Their architecture is designed to deal with sequential data, in this case, words or characters representations. The inputs go through the layers of the network along with the memory of inputs fed to the network in the past [52]. This way, the algorithm is fed context data intrinsically. A particular type of RNN called Long Short-Term Memory Networks, or LSTM Networks, can handle a longer

history of memory inputs being used in each layer prediction [53]. LSTM Networks are broadly used in text classification algorithms, including fake news detection [54] [55].

Many of those researches, including the original LIAR work [1], use Convolutional Neural Networks layers together with LSTMs when modeling text classifiers. Traditionally used in image classification, CNNs [56] have the power of producing features that take position into consideration, since they convolute through the data in each convolutional layer. In both CNN and LSTM models, including hybrid networks, *attention mechanism* [57] is used to capture dependencies between each feature pairs reducing ambiguity in subsequent layers. The attention mechanism, particularly *self-attention* used in text, when a sentence is disambiguated by itself, is present in most state-of-the-art NLP algorithms [58].

2.4 Ordinal Classification Formulations

Ordinal Classification is any classification task in which the labels have a natural order between them, usually referring to a unique measure. They usually appear in problems dealing with discrete measures such as HOT, MILD or COLD weather, LIGHT, NORMAL or HEAVY traffic, or in the case of misinformation detection, FALSE, HALF-TRUE, TRUE statement.

With ordinal classes there are dependencies between the predicted labels which are usually not modelled in the classifiers. Although many studies proposed ensemble [59] or novel methods [60] of dealing with ordinal problems, few implementations are available for use in library packages. One example is the *OCAPIS* R package [61] and another is the *ordinalNET* R package [62]. Both studies design algorithms that optimize custom loss functions designed specifically for ordinal regression problems.

A more direct and simpler approach uses a combination of simple binary classifiers and a decision rule based on the binary classes' probability output generating the multiclass prediction [3]. This method was applied in the fake news experiment of section 3 and will be detailed in section 3.4.2.

A handful of studies were also made discussing the evaluation of ordinal classification. Some studies correctly remark [63] that class accuracy does not account for the distance between the predicted and true class, arguing in favor of MSE to achieve "smaller" errors.

A different research study [64] formulated an error measured directly from the confusion matrix. The error for each sample in this case is smoothed along the range of

the possible class labels and increases as the ordinal predicted class get further away from the ordinal true class in the scale. This was the chosen evaluation measure for the ordinal formulation of the experiment in section 3, being detailed under the fake news detection scenario in section 3.4.3

3 The experiments

The experiments will be presented in this section. The main objective is to support answers to the questions introduced in section 1. Simple classifiers are evaluated against the original LIAR work [1] that used a hybrid LSTM and CNN network. Both non-ordinal and ordinal problem formulations are used and compared.

At first hand, this section details the dataset then follows on listing the tools and explaining the methods used.

3.1 LIAR Dataset

The LIAR Dataset [65] was created in 2017 in the work "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection [1]. The dataset is a collection of text and context metadata of over 10 thousand political statements extracted from the fact-checking organization PolitiFact [66]. PolitiFact is owned by the non-profit organization Poynter Institute for Media Studies [67], parent company of the Tampa Bay Times newspaper [68].

The dataset establishes fixed slices for training/validating/test splitting. The speaker affiliations are reasonably balanced, suggesting a good partisan independence. Some of the dataset information is described on Table 4 and Table 5.

Table 4 - Basic Liar Dataset Information

Dataset Statistics	
Training set size	10240
Validation set size	1284
Testing set size	1267
Average statement length (tokens)	17.9
Top-3 Speaker Affiliations	
Democrats	4137
Republicans	5665
None (e.g., Facebook posts)	2181

The output classes are reasonably balanced with a slightly lower presence of the *pants-fire* class, the extremely false statements. The labels are the same used by PolitiFact

when fact-checking a statement. They represent 6 ordinal levels of truthiness and PolitiFact has formal definitions for them to better establish their boundaries.

Table 5 - Dataset Classes Distribution

Class Label	Class Definition	Class Representativity (training + validation sets)
True	The statement is accurate and there is nothing significant missing.	16.01%
mostly-true	The statement is accurate but needs clarification or additional information.	19.21%
half-true	The statement is partially accurate but leaves out important details or takes things out of context.	20.50%
barely-true	The statement contains an element of truth but ignores critical facts that would give a different impression.	16.41%
false	The statement is not accurate.	19.58%
pants-fire	The statement is not accurate and makes a ridiculous claim.	8.29%

Some metadata is available in the dataset along with the statement's text. Each row contains a list of subjects covered in the statement, the speaker, job title, the state where the statement was taken, the speaker`s affiliation and the context of the statement. The history of false class labels for each speaker is also included. Some examples are displayed on Table 6.

Table 6 - Features of example rows

	Example1	**Example2**	**Example3**
Statement	Says the Obama administration plans to reduce …	Romney would turn Medicare into a voucher prog …	Members of Congress passed a pay raise for the…
Subjects	crime, criminal-justice, immigration	debates, medicare	medicare, retirement, social-security
Speaker	michael-mccaul	barack-obama	chain-email
Affiliation	republican	democrat	N/A
Job Title	congressman	President	N/A
State	Texas	Illinois	N/A
Context	U.S. House floor debate	the first presidential debate	a chain e-mail
N Pants-Fire	0	9	105
N False	0	70	43
N Barely True	1	71	11
N Half-True	1	160	8
N Mostly True	4	163	5

As Table 6 shows, most features are either text or categorical. This means that feature selection should play an important role in filtering out irrelevance once they are broken into categories and text vector features. The dimensionality would be too high otherwise. The peculiarity of the history label features will be treated on section 3.3.4 where the strong correlation with the output will cause them to be ruled out as redundant features.

3.2 Programming Ecosystem

This experiment was created, in its majority, using *Python3* [69] inside a *Jupyter Notebook* [70] environment and it is hosted in a *Github* repository [71]. Table 7 lists the most relevant libraries used for data cleaning, feature engineering and machine learning

classification. Some experimentation was also done in *Matlab* regarding the NCA classifier, which will be detailed further.

Table 7 - Experiment Libraries Usage

Library	Functionalities
pandas [72]	Dataset loading, cleaning and exhibiting
numpy [73]	Math operations, correlation checks
nltk [44]	Text tokenization, *POS-Tagging* [74] and *stopwords* filtering [75]
scikit-learn [35]	*TF-IDF* text vectorization, machine learning classifiers, hyperparameter tuning and performance metrics
matplotlib [76]	Data and result plots
fscnca [77]	NCA [51] classifier Matlab [78] library

3.3 Feature Engineering

The first step in feature engineering is splitting them into text and metadata features. Different approaches are used to vectorize them separately. Once they are vectorized they are concatenated horizontally creating the feature matrix that will be used for classification.

Just as different classifiers will be tested, different dimensionality reduction methods will also be used to compare the amount of information they retain and therefore determine which method is more suited.

3.3.1 Text Features Representation

The statement text was represented with simple bag of words model and TF-IDF weighting. The choice of word embeddings [37] would result in lower and denser dimensionality capturing semantic context and simplifying the dimensionality reduction phase. However, semantic context is assumed to be represented in the metadata features of the LIAR Dataset, particularly in the subject feature. The choice of bag of words text features is an attempt to capture a particular vocabulary that is more common in false statements.

Special text tokens such as punctuation, numbers and symbols, are sometimes stripped off before applying machine learning. This work addresses these tokens generically as “symbol tokens”. In this approach they were kept in the vocabulary so that they could be checked for relevance in the feature selection process. This choice is explained by the assumption that false and true statements can have different patterns of symbol token usage. Table 8 below shows an example of this symbol tokens’ representation in a sentence before going through the bag of words vectorization.

Table 8 - Example of tokens representation in a text statement

Label	Statement
mostly-true	The most recent Associated Press poll has Nader <DASH> Gonzalez at <NUMERAL_TOKEN> percent <COMMA> without any national coverage <COMMA> against McCain and Obama <DOT>

The symbols were also used to synthetize a new feature which was incorporated into metadata. A feature named *symbol_ratio* was created and it represents the ratio between the count of symbol tokens of the statement and the count of all tokens of the statement.

3.3.2 Statement Text Cleaning

Before applying TF-IDF weighting to the terms, each of the techniques from *Table 9* was applied.

Table 9 - Text cleaning techniques summary

Text Cleaning Technique	Purpose of the Technique
3-Grams Tokenization	Splitting the text into a list of tokens. A count of 3 N-Grams is used to retain some word context information.
No Lowercase	Lowercase is not applied to the text statements in order to keep writing style unchanged.
Stopwords Removal	Ignoring extremely common terms in the English language with no semantic meaning such as determiners or prepositions.
No Stemming	Reducing words to their stem or root would interfere with writing style and this technique was not applied.
POS-Tagging filtering	Reinforcing the stopwords removal step by keeping only the most relevant grammar (nouns, verbs, adjectives and adverbs).

.

3.3.3 TF-IDF Weighting

When vectorizing the text features using bag-of-words, the preprocessing techniques from section 3.3.2 were used and bigrams and trigrams were added to the token vocabulary.

It is common in TF-IDF implementations for the vocabulary to be further reduced by removing additional terms with low specificity terms (high DF) and high specificity terms (low DF) by using simple thresholds. These threshold cuts were performed in one of the feature matrices used in the experiments whereas the others kept the whole matrix and used different dimensionality reduction methods. Details follow in section 0.

The columns of the text feature matrix represent the vocabulary $\mathbf{V_s}$ used as features. The text vectorization process outputs one row vector with $\mathbf{V_s}$ elements for each statement, each element representing that term's relevance to the statement.

3.3.4 Metadata Features Representation

The metadata features are presented in Table 6 with the exception of the statement feature and the addition of the synthetized *symbol_ratio* feature. Although there might seem to be a small number of them, they were categorized and one-hot encoded [79]. That process considerably increased dimensionality.

The historical label features for each speaker, *N-Pants Fire* up until *N-Mostly True,* contain output information for the whole dataset. Therefore, these cannot be used for training a machine learning algorithm. Section 4.1.1 will show that including them in the training caused severe overfitting and misleadingly good results that could not be reproduced using additional data in the future. For that reason, after the initial tests, the historical label features were excluded from the algorithm.

The context feature is also a text feature but it is not diverse enough to be treated as text in the feature representation. To best represent it in categories instead, a POS-Tagging filter was used to keep only the noun tokens before categorizing it. This way, a small number of categories could capture the contexts and represent them as another one-hot encoded feature.

After being able to express the metadata as categories they needed to be encoded to numbers. With the exception of the *symbol_ratio* feature, the metadata features were one-hot encoded, creating one binary feature for each unique value of a feature.

Table *10* shows the number of binary features created when encoding the metadata.

To reduce ambiguity and dimensionality, the states, affiliation and context features were also converged to a list of the most recurring categories, categorizing any value out of that list as ‘unknown’. Table 11 shows those categories in details.

Table 10- Metadata Features after Categorization/Binarization

Source Feature	**Number of Features**
Subject	145
Speaker	3310
State	51
Affiliation	5
Context	9
Symbol Ratio	1

Table 11 - State, Affiliation and Context fixed categories

Feature	List of Categories
State	“Alabama”, “Alaska”, “Arizona”, “Arkansas”, “California”, “Colorado”, “Connecticut”, “Delaware”, “Florida”, “Georgia”, “Hawaii”, “Idaho”, “Illinois”, “Indiana”, “Iowa”, “Kansas”, “Kentucky”, “Louisiana”, “Maine”, “Maryland”, “Massachusetts”, “Michigan”, “Minnesota”, “Mississippi”, “Missouri”, “Montana”, “Nebraska”, “Nevada”, “New Hampshire”, “New Jersey”, “New Mexico”, “New York”, “North Carolina”, “North Dakota”, “Ohio”, “Oklahoma”, “Oregon”, “Pennsylvania”, “Rhode Island”, “South Carolina”, “South Dakota”, “Tennessee”, “Texas”, “Utah”, “Vermont”, “Virginia”, “Washington”, “West Virginia”, “Wisconsin”, “Wyoming”, “unknown”
Affiliation	“republican”, “democrat”, “independent”, “organization”, “unknown”
Context	“interview”, “debate”, “campaign”, “press”, “ad”, “letter”, “article”, “internet”

3.3.5 Feature Matrices Used: Different Dimensionality Reduction Techniques

This subsection will cover the distinct feature matrices used to test the algorithms varying the type of dimensionality reduction in the feature representations. Table 12 below details the methods used in each of them.

Table 12 - Dimensionality Reduction for the three feature matrices used

Feature Matrix	**Type**	**Step1 (Metadata)**	**Step1 (Text)**	**Step2 (Metadata)**	**Step2 (Text)**
FM1	Output Correlation Thresholds + Feature Pair Correlation Thresholds	0.03 minimum output correlation	0.03 minimum output correlation	0.1 maximum feature pair correlation	0.1 maximum feature pair correlation
FM2	PCA / LSI + Text Output Correlation Thresholds	100 components (79.6% variance)	0.02 minimum output correlation	N/A	150 components (67.5% variance)
FM3	Text TFIDF thresholds + PCA/LSI	100 components (79.6% variance)	TF-IDF cuts 0.5 max_df 5 min_df	N/A	150 components (23.5% variance)

With the class labels placed in an ordinal scale of untruthiness, such as in Table 13, it is possible to verify if a feature output correlation is positive (contributes to a false statement) or negative (contributes to a true statement).

Table 13 - Ordinal class labels on a scale

$l = 0$ *true*	$l = 1$ *mostly-true*	$l = 2$ *half-true*	$l = 3$ *barely-true*	$l = 4$ *false*	$l = 5$ *pants-fire*

|----------|

P = 1

From that point of view, selecting the features that have a minimum absolute correlation with the output is a way to exclude the most irrelevant ones. Figure 5 and Figure 6 show the absolute correlations with the output for each feature.

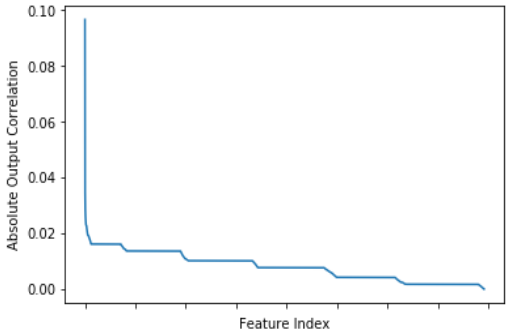

Figure 5 - Text Feature to Output Absolute Correlations

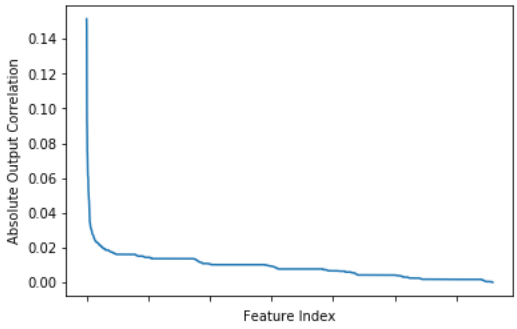

Figure 6 – Metadata Feature to Output Absolute Correlations

From the plots it is possible to verify that only a small number of features contribute to the output. Even the highest correlated/uncorrelated text feature barely reaches 10%. This is something to be expected from text data specially in a subjective output. Some of the high correlated features are displayed in section 4.1.

To reduce the number of features in matrices FM1 and FM2, simple thresholds were established using the "knee" of the curve. The minimum correlation thresholds are detailed in Table 12.

In a similar way, visual graph inspection was used in FM1 to address feature redundancy and further remove less relevant features. Each feature was checked against each other feature for their absolute correlations. Once again, the plots are shown in Figure 7 and Figure 8.

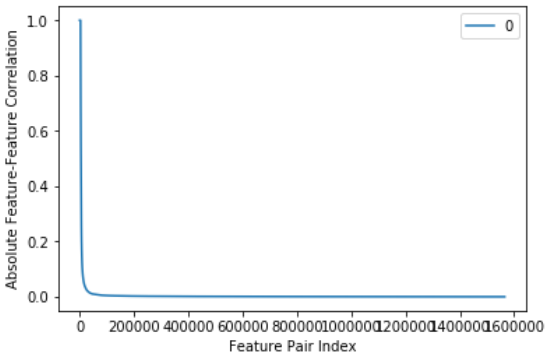

Figure 7 - Text feature to feature correlations

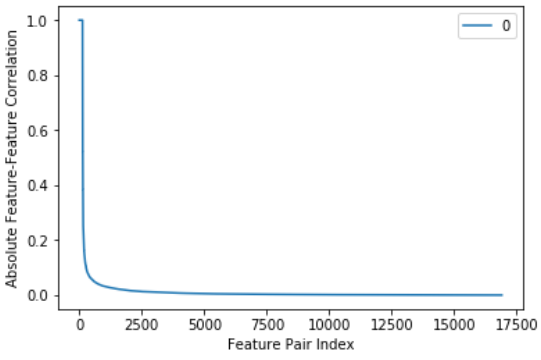

Figure 8 - Metadata feature to feature correlations

Once more, using the "knee" of the curves, threshold cuts from Table 12 excluded one of the features from each high correlated pair.

For FM2 and FM3, the dimensionality reduction was made mostly by extracting the principal components of the data transforming the vector space to a latent lower dimensional space. For metadata, 100 components were used. For text, with initial dimensions over 200k, FM2 used an output correlation threshold to initially reduce dimension before applying LSI with 150 components. For the same reason, FM3 used minimum and maximum document frequency cuts directly from the TF-IDF matrix before using LSI.

3.4 Machine Learning Classification

Starting the classification phase, a scan of different types of classifiers is made in order to find best performing one. With the regular classification results in hand, an ordinal ensemble method is proposed. The process is then repeated using the ordinal formulation expecting an improvement due to the ordinality of the output class labels.

To accurately evaluate the classifiers, a grid search was made checking the mean and standard deviation of the 10-fold cross-validation accuracy score in a total of 10 experiments. The reason for multiple experiments is to dilute the randomness of dataset splitting due to random seed. Each classifier was evaluated with each of the feature matrices from Table 12 and each combination of a grid of hyperparameters.

3.4.1 Hyperparameter Tuning

The process used *scikit-learn* class *GridSearchCV* [80], which is able to automatically run cross-validation tests for a classifier given a parameter grid, obtaining the best performing configuration. Table 14 below shows details of the classifiers and parameter grids used.

The parameters search for the classifiers is focused on testing and finding the best regularization parameter (such as “C” parameter in *SVC* and *LogisticRegression*) and the most suited method of each algorithm (such as “criterion” *entropy* in *RandomForestClassifier* or “algorithm” *kd-tree* in *KNearestNeighbors*).

Table 14 - Algorithm Exploration and Hyperparameter Tuning

Classifier	Parameter Grid	
Naïve Bayes	var_smoothing	[1e-7, 1e-8, 1e-9, 1e-10, 1e-11]
Logistic Regression	Tolerance	[1e-4, 1e-6, 1e-8]
	C	[0.1, 0.3,1, 3,10, 30, 100]
	multi_class	[“ovr”, “multinomial”]
	Solver	[“newton-cg”, “lbfgs”, “sag”, “saga”]
K-Nearest Neighbors	n_neighbors	[3, 5, 10, 20]
	Algorithm	[“ball_tree”, “kd_tree”, “brute”]
	Weights	[“uniform”, “distance”]
	leaf_size	[10, 30, 50]
	P	[1, 2]
Random Forest	n_estimators	[10,30,50,100]
	Criterion	[“gini”, “entropy”]
	max_depth	[5, 10, 30, 50]
	min_samples_leaf	[2, 5, 10]
SVM	C	[3, 10, 30, 100]
	Kernel	[“linear”, “poli”, “rbf”]
	Degree	[3, 5]

Apart from the classifiers on Table 14, the NCA Classifier [51] was also evaluated with the intention to check if its intrinsic feature selection and dimensionality reduction could handle the problem in a better way. In the case of the NCA, only the regularization hyperparameter λ was tuned.

3.4.2 Modeling as an Ordinal Classification

Rather than using more complex classifiers, this experiment attempts to take advantage of the ordinality of the classes, representing it in an ensemble model of simple classifiers such as the ones listed in Table 14.

As stated in section 2.4, the work [3] provided a simple formulation of an ensemble method that divides the $k\ classes$ classification problem into $k-1$ binary classifiers that predicts $y > l$. The method represents each class label l as an integer from 0 to $k-1$, an ordered scale, creating a binary classifier for each class label interval boundary. The convergence of the binary classifiers is done by combining the output probabilities of each binary classifier $P(y > l)$.

The following Table 15 summarizes the binary classifiers exposing their negative and positive classes. Finally, Table 16 details how to go from 5 binary probability predictions to the original 6 class output probability for a sample.

Table 15 - 5 binaries classifiers for a 6-class classification problem

Binary Classifier	**Negative Class**	**Positive Class**
C_0	$y \leq 0$	$y > 0$
C_1	$y \leq 1$	$y > 1$
C_2	$y \leq 2$	$y > 2$
C_3	$y \leq 3$	$y > 3$
C_4	$y \leq 4$	$y > 4$

Table 16 - Convergence of Binary Classifiers Probabilities

Class Probability	**Combined Binary Classifier Probabilities**
$P("true")$	$1 - P(y > 0 \mid X)$
$P("mostly - true")$	$P(y > 0 \mid X) - P(y > 1 \mid X)$
$P("half - true")$	$P(y > 1 \mid X) - P(y > 2 \mid X)$
$P("barely - true")$	$P(y > 2 \mid X) - P(y > 3 \mid X)$
$P("false")$	$P(y > 3 \mid X) - P(y > 4 \mid X)$
$P("pants - fire")$	$P(y > 4 \mid X)$

When predicting the output of a sample, each of the binary probability outputs is evaluated before computing the final class probabilities. After that, the final probabilities should be normalized between 0 and 1. The sample receives the label with the highest probability.

3.4.3 Evaluation Metric for Ordinal Classification

The hypothesis to be verified is that modeling the problem as an ordinal classification should improve not only the base accuracy of the model but also make prediction errors less impactful. This means that label prediction errors should be adjacent or at least closer to the true labels in the ordinal scale. A misclassification of a "true" statement as a "mostly-true" statement is much less harmful than a "true" statement being classified as "false".

This kind of evaluation of the ordinal classifier calls for an error metric of class dispersion, a metric that captures this distance between the ordinal labels. An error metric for ordinal classification based on the confusion matrix dispersion is proposed in [64] and defines a sample error by:

$$OC_{\beta} = \min\left(1; 1 - \frac{1}{1 + \ |r - c\ |} + \ \beta|r - c|\ \right)$$

The values of r and c correspond to the row and column indexes of the confusion matrix. The error goes to 0 when $r = c$ in the diagonal of the confusion matrix, meaning the class is correctly classified. The parameter β is responsible for smoothing the error between 0 and 1 depending on the distance $|r - c|$, just as Figure 9 illustrates.

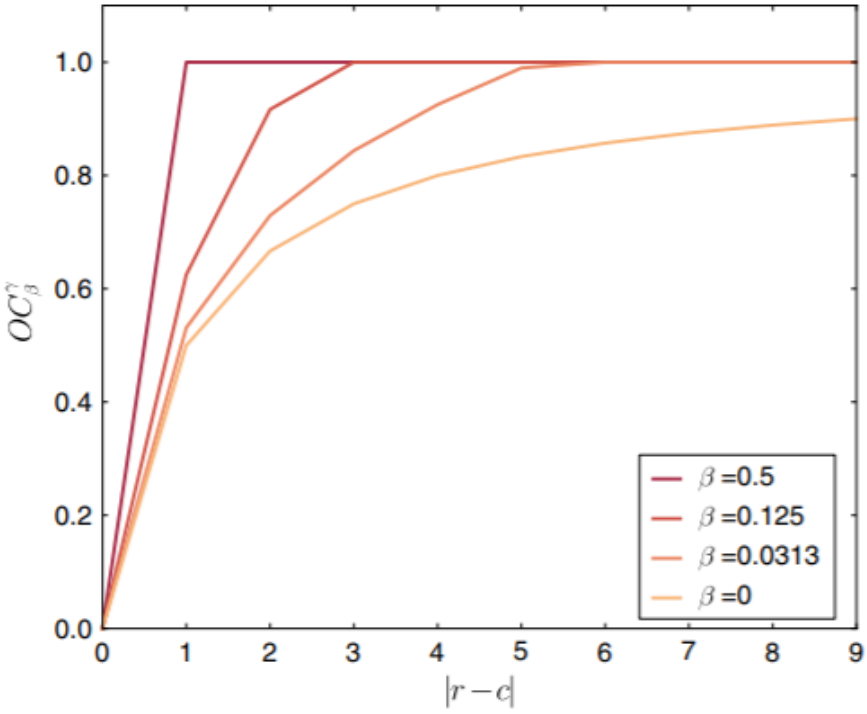

Figure 9 – Ordinal sample error smoothing based on confusion matrix

Each curve in Figure 9 shows the sample error OC_β assuming intermediate values between 0 and 1. The error increases as the error distance between the predicted and true ordinal classes increased. With $\beta = 0.5$ the error metric converges to a hard error of 1 for the sample, not taking into consideration the distance between the predicted and true ordinal classes in this case.

For the LIAR dataset problem, the maximum $|r - c|$ distance is 5, so in the experiment, $\beta = 0.0313$ is used to smooth the 6-class error into the range of 0 and 1. OC_β is checked in the end of the ordinal classification to check for reduced error compared to the simple classifications from section 3.4.1.

4 Results

This section will cover the results of the experiment. Starting the section, the initial results regarding the history features *N-Pants Fire* up until *N-Mostly True* will be presented, justifying the exclusion of them. Following, the feature matrices' visualizations are detailed, along with some data insights derived from feature selection. Moving to the classification, the results of the grid search from section 3.4.1 is presented. After that, the results of the same hyperparameter tuning is presented, this time for the ordinal formulation using the binary ensemble, detailing each binary classifier result and evaluating the ensemble using the OC_β measure.

4.1 Data Insights

4.1.1 False Information History Features

The LIAR Dataset carries 5 metadata features that correspond to the number of false labels according to the following Table 17.

Table 17 - Label History Features

n_barely-true	Number of *barely-true* statements for the speaker on whole dataset
n_false	Number of *false* statements for the speaker on whole dataset
n_half-true	Number of *half-true* statements for the speaker on whole dataset
n_mostly-true	Number of *mostly-true* statements for the speaker on whole dataset
n_pants-fire	Number of *pants-fire* statements for the speaker on whole dataset

A number of reasons led to the exclusion of these features for classification. First of all, they represent a direct class label output proportion for each author, something that skews the prediction of speakers in the direction of their past statements. Second, the history features fail to provide information on a speaker with a small amount of previous statements while being of great importance for speakers that have many statements in the

dataset. Also linked to that, it makes the dataset hard to scale in size in the future, since new speakers will have zero information in those features. Lastly, besides the extremely high output correlation, they also contribute to overfitting the classifiers since they contain information of the whole dataset, including training, validation and test splits. They would need to be recalculated for each of those splits separately.

Following in Table 18, the data visualization for initial tests including the history features is shown side by side with the visualization of the data composed of the history features alone.

Table 18 – T-SNE 2D Visualizations for History Features

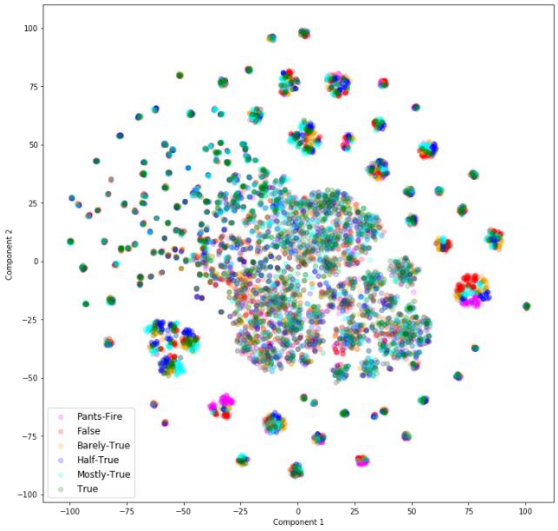

Figure 10 - All Features including History Features

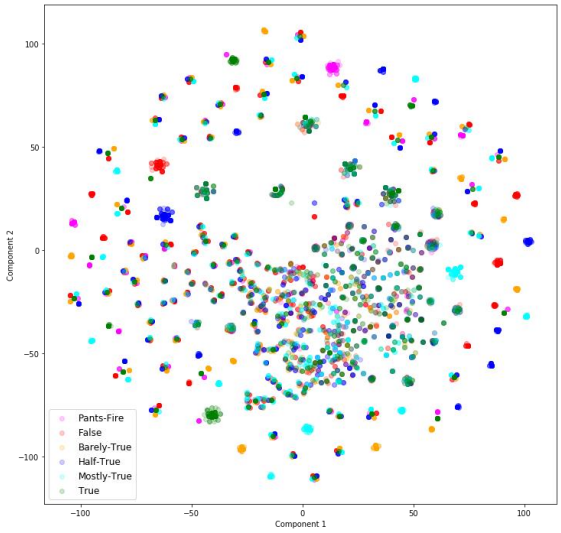

Figure 11 - Only History Features

When comparing the data 2D projections with the other matrices used it will be easy to inspect that the false history information does a very good job at making the class labels less intertwined. It is important to note that PCA could capture the information of the history feature in its principal components, therefore achieving better performance in representing the data with history features when reducing dimensionality.

Some classifiers, such as K-Nearest Neighbors and Random Forest, captured the history output information better than others providing misleading overfitted results. Table 19 shows these results, using only history as features.

Table 19 - Single Classifier Classification Results for History Features Only

	Naive Bayes	Logistic Regression	RandomForest	K-Nearest Neighbors	SVM
Mean 10-fold Accuracy	0.189	0.223	**0.630**	**0.642**	0.278
Std 10-fold Accuracy	0.0012	0.0007	0.0015	0.0017	0.0025
10-fold Average Training Time (ms)	98	8082	2827	4041	66705
Best Parameters	Var_smoothing = 1e-11	Tolerance = 1e-6 C = 100 multi_class = "ovr" solver = "sag"	n_estimators = 100 criterion = "gini" max_depth = 50 min_samples_leaf = 2	n_neighbors = 20 algorithm = "ball_tree" leaf_size = 10 weights = "distance	C = 100 Kernel = "poly" Degree = 5

4.1.2 Feature Matrix 1: Data Insights

As Table 12 details, FM1 is composed of all features excluding history with its dimensionality reduced by feature output correlation and feature redundancy. Following Figure 12 shows a 2D visualization using T-SNE to project the data.

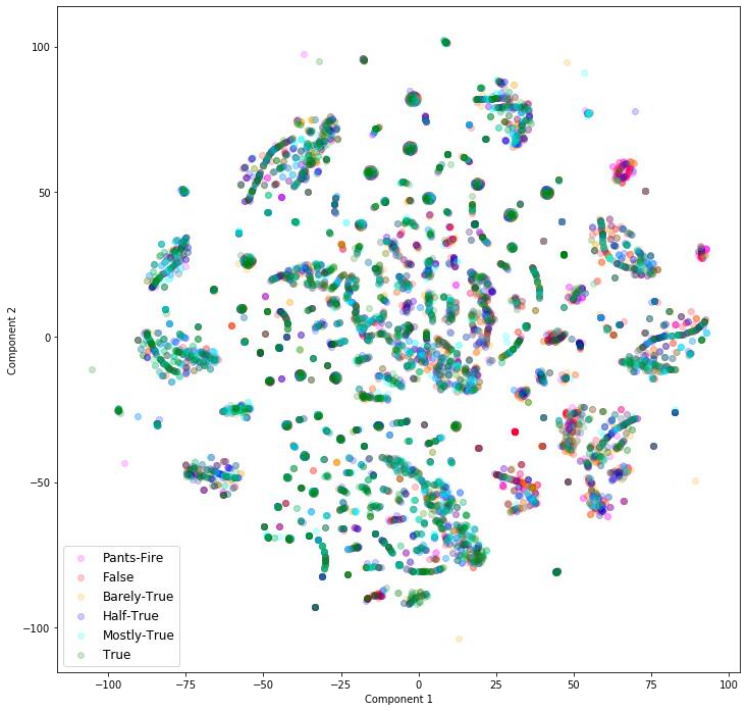

Figure 12 - T-SNE 2D Visualizations for FM1

The following Table 20 will summarize the dimensionality of the experiment using FM1.

Table 20 - Dimensionality Reduction of FM1 during preprocessing and feature selection

Features	Initial Dimension	Dimension after Vectorization/Binarization	Dimension after Feature Selection by Output Correlation	Dimension after Feature Selection by Feature Redundancy
Text	9634	197908	1251	253
Metadata	12	3532	130	92

When performing Feature Selection by Output Correlation using absolute values, it is possible to find some hints on the most relevant features. Some with positive correlation, meaning they make a statement label grow in the misinformation ordinal scale (Table 13) and some having negative correlation and making a statement stay low in the ordinal scale, closer to the "truth" label. Table 21 shows these hints on metadata features and Table 22 on text features.

Table 21 - Most correlated metadata features. Negative indicates truth bias while positive indicates false bias (Table 13)

Metadata Feature	Origin Metadata	Discrete Value	Output Correlation
speaker_chain-email	Speaker	chain-email	+ 15.14%
affiliation_democrat	Affiliation	democrat	- 14.65%
speaker_donald-trump	Speaker	donald-trump	+ 10.52%
speaker_blog-posting	Speaker	blog-posting	+ 8.96%
subject_health-care	Subject	health-care	+ 7.48%
subject_religion	Subject	Religion	+ 6.72%
symbol_ratio	N/A	N/A	- 6.67%
speaker_viral-image	Speaker	viral-image	+ 5.28%
subject_economy	Subject	Economy	- 4.78%
speaker-michele-bachmann	Speaker	michele-bachmann	+ 4.73%

Table 22 - Most correlated text features. Negative indicates truth bias while positive indicates false bias (Table 13)

Text Feature	Output Correlation
<NUMERAL_TOKEN>	- 9.67%
Obama	+ 7.77%
Obamacare	+ 5.89%
Care	+ 5.47%
Walker	+ 5.40%
Georgia	- 5.29%
Average	- 5.08%

4.1.3 Feature Matrix 2: Data Insights

As Table 12 details, FM2 is composed of all features excluding history with its dimensionality reduced by SVD Principal Component Analysis. Following Figure 13 shows a 2D projection of the data.

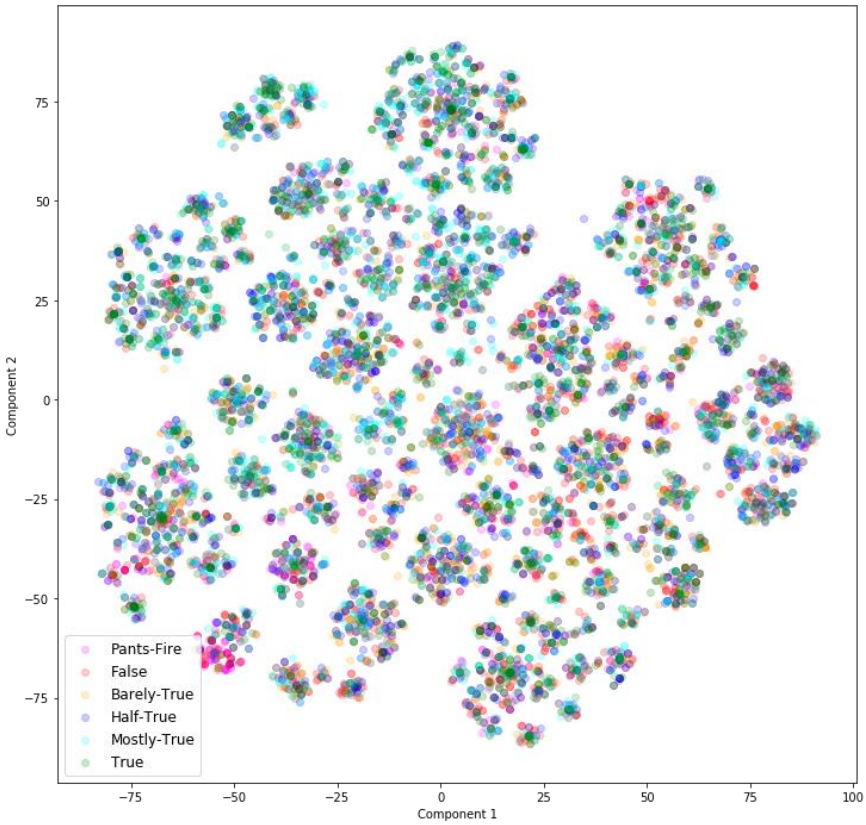

Figure 13 - T-SNE 2D Visualizations for FM2

FM2 matrix kept 100 metadata components and 150 text components. In a similar way to the previous FM1 matrix, we can analyze the SVD principal components to check which features contributes to them. The following Table 23 will summarize the dimensionality of the experiment using FM2.

Table 23 - Dimensionality Reduction of FM2 during preprocessing and feature selection

Features	Initial Dimension	Dimension after Vectorization/Binarization	Dimension after Feature Selection by Output Correlation	Dimension after PCA	Dimension after LSI
Text	9634	197908	1251	N/A	150
Metadata	12	3532	N/A	100	N/A

Table 24 details how much variance from the original data the remaining components were able to represent in FM2.

Table 24 - SVD Explained Variance for FM2

SVD Process	Components	Original Variance Explained
Metadata Features SVD	100	79.65%
Text Features SVD	150	67.5%

Reverse engineering the total feature contribution weight to the remaining components we can list the top original features selected by the SVD process. Table 25 shows the top 5 metadata and Table 26 shows the top 5 text features.

Table 25 - Most Represented Metadata Features with SVD on FM2

Metadata Feature	Origin Metadata	Discrete Value	Feature Weight (principal contribution)
subject_children	Subject	children	6.796
subject_labor	Subject	labor	6.725
subject_workers	Subject	workers	6.442
subject_poverty	Subject	poverty	6.254
subject_public-health	Subject	public-health	6.232

Table 26 - Most Represented Text Features with SVD on FM2

Text Feature	Feature Weight (principal contribution)
highest	8.096
<NUMERAL_TOKEN> <DOT>	7.970
time	7.537
average	7.356
<NUMERAL_TOKEN> <NUMERAL_TOKEN>	7.320

4.1.4 Feature Matrix 3: Data Insights

The third feature matrix is also composed of all features excluding history with its dimensionality reduced by both TF-IDF threshold cuts on text followed by SVD Principal Component Analysis, applied to both text and metadata. Following Figure 14 shows a 2D projection.

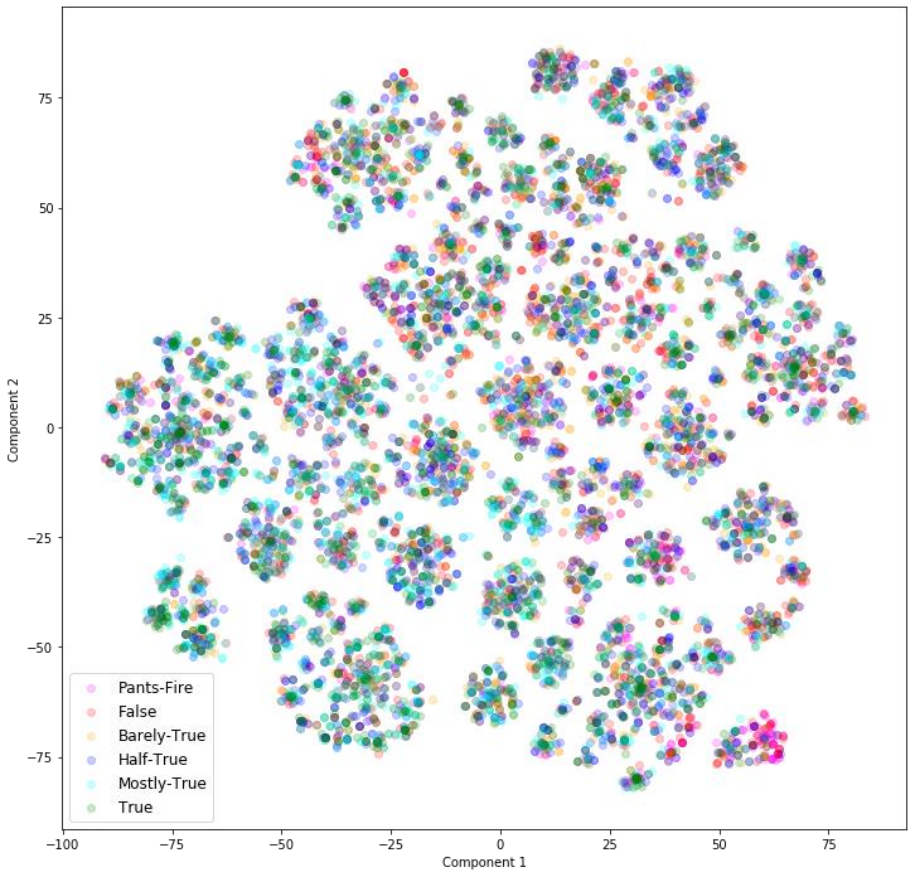

Figure 14 - T-SNE 2D Visualizations for FM3

Table 27 details dimensionality reduction steps on FM3 while Table 28 lists the variance explained by the SVD process.

Table 27 - Dimensionality Reduction of FM3 during preprocessing and feature selection

Features	**Initial Dimension**	**Dimension after Vectorization/Binarization**	**Dimension after Feature Selection TF-IDF Thresholds**	**Dimension after PCA**	**Dimension after LSI**
Text	9634	197908	7708	N/A	150
Metadata	12	3532	N/A	100	N/A

Table 28 - SVD Explained Variance for FM3

SVD Process	Components	Original Variance Explained
Metadata Features SVD	100	79.65%
Text Features SVD	150	23.49%

The most represented metadata features are the same as FM2, listed in Table 25, since there was no change in the metadata processing. Table 29 details most represented text features in the principal components.

Table 29 - Most Represented Text Features with SVD on FM3

Text Feature	Feature Weight (principal contribution)
voted	8.373
<SUSPENSION_POINTS>	7.345
Wisconsin	7.194
<POSSESSIVE_CONTRACTEDIS>	7.141
government	7.138

4.2 Single Classifiers

This section will present the single classifier best results after hyperparameter tuning with grid search. Table 30-32 show results for each feature matrix. The NCA classifier results are displayed in Table 33-35.

4.2.1 Grid Search Results

Table 30 - Single Classifier Classification Results for FM1

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
Accuracy (μ)	0.1982	**0.2634**	0.2599	0.2410	0.2318
Accuracy (σ)	0.0008	0.0014	0.0017	0.0033	0.0039
Precision (μ)	0.2972	0.2828	0.2739	0.2481	0.2481
Precision (σ)	0.0186	0.0016	0.0039	0.0034	0.0041
Recall (μ)	0.1982	0.2634	0.2599	0.2410	0.2318
Recall (σ)	0.0008	0.0014	00017	0.0034	0.0039
F1-Score (μ)	0.1202	0.2374	0.2223	0.2371	0.2154
F1-Score (σ)	0.0016	0.0013	0.0014	0.0033	0.0052
OC_β (μ)	0.5942	**0.4925**	0.4959	0.5203	0.5314
OC_β (σ)	0.0024	0.0009	0.0007	0.0022	0.0028
Average Training Time (ms)	243	21178	3320	9288	20768
Best Parameters	var_smoothing = 1e-07	tolerance = 1e-8 C = 3 multi_class = "multinomial" solver = "saga"	n_estimators = 100 criterion = "entropy" max_depth = 50 min_samples_leaf = 10	n_neighbors = 20 algorithm = "ball_tree" leaf_size = 30 weights = "uniform"	C = 100 Kernel = "linear" Degree = 5

Table 31 - Single Classifier Classification Results for FM2

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
Accuracy (μ)	0.2159	**0.2685**	0.2598	0.2336	0.2378
Accuracy (σ)	0.0014	0.0030	0.0025	0.0021	0.0030
Precision (μ)	0.2323	0.2737	0.2978	0.2376	0.2507
Precision (σ)	0.0015	0.0035	0.0072	0.0021	0.0035
Recall (μ)	0.2159	0.2685	0.2598	0.2336	0.2378
Recall (σ)	0.0014	0.0030	0.0025	0.0021	0.0030
F1-Score (μ)	0.2074	0.2643	0.2248	0.2307	0.2319
F1-Score (σ)	0.0014	0.0032	0.0020	0.0020	0.0030
OC_β (μ)	0.5291	**0.4909**	0.4911	0.5282	0.5275
OC_β (σ)	0.0010	0.0018	0.0014	0.0019	0.0025
10-fold Average Training Time (ms)	899	120574	21641	15270	41141
Best Parameters	var_smoothing = 1e-08	tolerance = 1e-6 C = 10 multi_class = "ovr" solver = "lbfgs"	n_estimators = 100 criterion = "gini" max_depth = 10 min_samples_leaf = 10	n_neighbors = 20 algorithm = "brute" leaf_size = 30 weights = "distance"	C = 100 Kernel = "linear" Degree = 3

Table 32 - Single Classifier Classification Results for FM3

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
Accuracy (μ)	0.2159	**0.2631**	0.2584	0.2388	0.2381
Accuracy (σ)	0.0020	0.0023	0.0022	0.0025	0.0035
Precision (μ)	0.2312	0.2728	0.2806	0.2427	0.2450
Precision (σ)	0.0019	0.0027	0.0036	0.0026	0.0044
Recall (μ)	0.2159	0.2631	0.2584	0.2388	0.2381
Recall (σ)	0.0020	0.0023	0.0022	0.0025	0.0035
F1-Score (μ)	0.2120	0.2558	0.2376	0.2360	0.2320
F1-Score (σ)	0.0020	0.0023	0.0028	0.0025	0.0037
OC_β (μ)	0.5535	0.4959	**0.4957**	0.5267	0.5302
OC_β (σ)	0.0013	0.0017	0.0013	0.0018	0.0028
Average Training Time (ms)	1136	116131	15503	17733	39639
Best Parameters	var_smoothing = 1e-09	tolerance = 1e-6 C = 0.3 multi_class = “ovr” solver = “lbfgs”	n_estimators = 100 criterion = “gini” max_depth = 50 min_samples_leaf = 10	n_neighbors = 20 algorithm = “ball_tree” leaf_size = 50 weights = “distance”	C = 3 Kernel = “linear” Degree = 3

4.2.2 NCA Classifier: Results

Table 33 - NCA Classification Results for FM1

Lambda	Mean Acc	Std Acc	Mean OCB	Std OCB
8.677e-07	0.2394	0.0034	0.5168	0.0027
8.677e-06	0.2401	0.0035	0.5166	0.0030
8.677e-05	**0.2462**	0.0032	**0.5082**	0.0022
8.677e-04	0.2383	0.0035	0.5211	0.0048
8.677e-03	0.2050	0	0.5188	0
8.677e-02	0.2050	0	0.5188	0

Table 34 - NCA Classification Results for FM2

Lambda	Mean Acc	Std Acc	Mean OCB	Std OCB
8.677e-07	0.2144	0.0040	0.5482	0.0036
8.677e-06	0.2131	0.0039	0.5491	0.0033
8.677e-05	0.2154	0.0036	0.5466	0.0031
8.677e-04	**0.2291**	0.0039	**0.5303**	0.0034
8.677e-03	0.2050	0	0.5188	0
8.677e-02	0.2050	0	0.5188	0

Table 35 - NCA Classification Results for FM3

Lambda	Mean Acc	Std Acc	Mean OCB	Std OCB
8.677e-07	0.2115	0.0062	0.5527	0.0048
8.677e-06	0.2120	0.0034	0.5527	0.0031
8.677e-05	0.2136	0.0061	0.5515	0.0051
8.677e-04	**0.2278**	0.0035	**0.5342**	0.0033
8.677e-03	0.2050	0	0.5188	0
8.677e-02	0.2050	0	0.5188	0

4.3 Ordinal Binary Ensemble

This section will cover the results of the same classifiers, this time modeled in an ordinal ensemble as described in section 3.4.2. Hyperparameter tuning is made to maximize accuracy of each binary classifier before combining them.

4.3.1 Grid Search Results

Table 36 follows and shows the hyperparameter tuning results for each of the binary classifiers for FM1.

Table 36 - Ordinal hyperparameter tuning for each binary classifier using FM1

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
C_0 Accuracy (μ)	0.2597	0.8401	0.8399	0.8392	0.8387
C_0 Accuracy (σ)	0.0007	0.0002	0.0001	0.0003	0.0006
C_0 Best Parameters	var_smoothing: 1e-07	tolerance: 1e-03 solver: "saga" C: 30 multi_class: "ovr"	n_estimators: 10 criterion: "entropy" max_depth: 10 min_samples_leaf: 5	n_neighbors: 20 algorithm: "brute" leaf_size: 30 weights: uniform	C: 30 kernel: "rbf" degree: 5
C_1 Accuracy (μ)	0.4328	0.6633	0.6591	0.6363	0.6494
C_1 Accuracy (σ)	0.0009	0.0008	0.0014	0.0016	0.0011
C_1 Best Parameters	var_smoothing: 1e-07	tolerance: 1e-06 solver:"newton-cg" C: 100 multi_class: "multinomial"	n_estimators: 100 criterion: "entropy" max_depth: 30 min_samples_leaf: 2	n_neighbors: 20 algorithm: "kd-tree" leaf_size: 10 weights: uniform	C: 30 kernel: "rbf" degree: 5
C_2 Accuracy (μ)	0.6125	0.6308	0.6261	0.6105	0.6258
C_2 Accuracy (σ)	0.0006	0.0013	0.0015	0.0022	0.0017
C_2 Best Parameters	var_smoothing: 1e-07	tolerance: 1e-03 solver: "saga" C: 10 multi_class: "multinomial"	n_estimators: 100 criterion: "gini" max_depth: 30 min_samples_leaf: 5	n_neighbors: 20 algorithm: "ball-tree" leaf_size: 10 weights: uniform	C: 30 kernel: "rbf" degree: 3
C_3 Accuracy (μ)	0.7351	0.7430	0.7441	0.7378	0.7402
C_3 Accuracy (σ)	0.0011	0.0003	0.0011	0.0008	0.0012
C_3 Best Parameters	var_smoothing: 1e-10	tolerance: 1e-03 solver: "newton-cg" C: 100 multi_class: "multinomial"	n_estimators: 100 criterion: "gini" max_depth: 30 min_samples_leaf: 2	n_neighbors: 20 algorithm: "brute" leaf_size: 30 weights: uniform	C: 30 kernel: "rbf" degree: 5
C_4 Accuracy (μ)	0.2605	0.9206	0.9204	0.9198	0.9193
C_4 Accuracy (σ)	0.0008	0.0003	0.0004	0.0003	0.0004
C_4 Best Parameters	var_smoothing: 1e-07	tolerance: 1e-03 solver: "newton-cg" C: 100 multi_class: "multinomial"	n_estimators: 100 criterion: "entropy" max_depth: 50 min_samples_leaf: 5	n_neighbors: 20 algorithm: "ball-tree" leaf_size: 50 weights: uniform	C: 30 kernel: "rbf" degree: 3

Table 37 below summarizes the results for FM1 using the ordinal ensemble method.

Table 37 - Ordinal Ensemble Classification Results for FM1

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
Accuracy (μ)	0.1874	**0.2632**	0.2582	0.2392	0.2506
Accuracy (σ)	0.0003	0.0011	0.0026	0.0026	0.0016
Precision (μ)	0.210	0.2819	0.2697	0.2472	0.2314
Precision (σ)	0.0263	0.0016	0.0035	0.0029	0.0096
Recall (μ)	0.1874	0.2632	0.2582	0.2392	0.2506
Recall (σ)	0.0003	0.0011	0.0026	0.0026	0.0016
F1-Score (μ)	0.0774	0.2392	0.2302	0.2360	0.1988
F1-Score (σ)	0.0004	0.0011	0.0026	0.0025	0.0016
OC_β(μ)	0.6144	**0.4922**	0.4931	0.5171	0.5056
OC_β(σ)	0.0003	0.0006	0.0016	0.0017	0.0010

Following Table 38 shows the hyperparameter tuning results for each of the binary classifiers for FM2.

Table 38- Ordinal hyperparameter tuning for each binary classifier using FM2

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
C_0 Accuracy (μ)	0.7158	0.8402	0.8401	0.8377	0.8398
C_0 Accuracy (σ)	0.0006	0.0003	0.0002	0.0004	0.0004
C_0 Best Parameters	var_smoothing: 1e-10	tolerance: 1e-03 solver: "saga" C: 3 multi_class: "multinomial"	n_estimators: 100 criterion: "gini" max_depth: 30 min_samples_leaf: 2	n_neighbors: 20 algorithm: "ball-tree" leaf_size: 50 weights: uniform	C: 30 kernel: "linear" degree: 3
C_1 Accuracy (μ)	0.6303	0.6676	0.6613	0.6407	0.6509
C_1 Accuracy (σ)	0.0013	0.0018	0.0012	0.0015	0.0011
C_1 Best Parameters	var_smoothing: 1e-10	tolerance: 1e-06 solver: "saga" C: 3 multi_class: "multinomial"	n_estimators: 100 criterion: "entropy" max_depth: 50 min_samples_leaf: 5	n_neighbors: 20 algorithm: "kd-tree" leaf_size: 10 weights: distance	C: 30 kernel: "linear" degree: 5
C_2 Accuracy (μ)	0.6049	0.6433	0.6334	0.5924	0.6213
C_2 Accuracy (σ)	0.0020	0.0015	0.0016	0.0014	0.0015
C_2 Best Parameters	var_smoothing: 1e-08	tolerance: 1e-06 solver: "newton-cg" C: 10 multi_class: "multinomial"	n_estimators: 100 criterion: "entropy" max_depth: 30 min_samples_leaf: 10	n_neighbors: 20 algorithm: "ball-tree" leaf_size: 50 weights: uniform	C: 30 kernel: "linear" degree: 5
C_3 Accuracy (μ)	0.6699	0.7403	0.7407	0.7299	0.7378
C_3 Accuracy (σ)	0.0011	0.0004	0.0007	0.0011	0.0009
C_3 Best Parameters	var_smoothing: 1e-09	tolerance: 1e-06 solver: "saga" C: 10 multi_class: "multinomial"	n_estimators: 100 criterion: "gini" max_depth: 50 min_samples_leaf: 2	n_neighbors: 20 algorithm: "ball-tree" leaf_size: 50 weights: uniform	C: 100 kernel: "linear" degree: 3
C_4 Accuracy (μ)	0.8097	0.920713	0.9200	0.9188	0.9176
C_4 Accuracy (σ)	0.0005	0.0004	0.0004	0.0004	0.008
C_4 Best Parameters	var_smoothing: 1e-10	tolerance: 1e-08 solver: "newton-cg" C: 10 multi_class: "ovr"	n_estimators: 50 criterion: "gini" max_depth: 50 min_samples_leaf: 2	n_neighbors: 20 algorithm: "ball-tree" leaf_size: 50 weights: distance	C: 30 kernel: "linear" degree: 3

Following Table 39 summarizes the results for FM2 using the ordinal ensemble method.

Table 39 - Ordinal Ensemble Classification Results for FM2

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
10-fold Accuracy (μ)	0.2206	**0.2687**	0.2504	0.2287	0.2632
10-fold Accuracy (σ)	0.0014	0.0023	0.0030	0.0013	0.0026
Precision (μ)	0.2283	0.2758	0.2523	0.2328	0.2696
Precision (σ)	0.0018	0.0022	0.0033	0.0017	0.0031
Recall (μ)	0.2206	0.2687	0.2504	0.2287	0.2632
Recall (σ)	0.0014	0.0024	0.0030	0.0013	0.0026
F1-Score (μ)	0.2136	0.2639	0.2487	0.2252	0.2499
F1-Score (σ)	0.0015	0.0023	0.0031	0.0012	0.0024
OC_{β}(μ)	0.5352	**0.4875**	0.5074	0.5321	0.4892
OC_{β}(σ)	0.0007	0.0014	0.0019	0.0009	0.0018

Following Table 40 shows the hyperparameter tuning results for each of the binary classifiers for FM3.

Table 40- Ordinal hyperparameter tuning for each binary classifier using FM3

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
C_0 Accuracy (μ)	0.6517	0.8399	0.8400	0.8387	0.8379
C_0 Accuracy (σ)	0.0016	0	0.0001	0.0005	0.0006
C_0 Best Parameters	var_smoothing: 1e-11	tolerance: 1e-03 solver: "newton-cg" C: 0.1 multi_class: "ovr"	n_estimators: 100 criterion: "gini" max_depth: 50 min_samples_leaf: 2	n_neighbors: 20 algorithm: "ball-tree" leaf_size: 50 weights: distance	C: 3 kernel: "rbf" degree: 5
C_1 Accuracy (μ)	0.5835	0.6602	0.6564	0.6430	0.6538
C_1 Accuracy (σ)	0.0014	0.0011	0.0019	0.0027	0.0022
C_1 Best Parameters	var_smoothing: 1e-08	tolerance: 1e-03 solver: "sag" C: 0.3 multi_class: "ovr"	n_estimators: 100 criterion: "entropy" max_depth: 50 min_samples_leaf: 2	n_neighbors: 20 algorithm: "ball-tree" leaf_size: 50 weights: distance	C: 10 kernel: "rbf" degree: 3
C_2 Accuracy (μ)	0.5895	0.6333	0.6266	0.5950	0.6242
C_2 Accuracy (σ)	0.0014	0.0016	0.0022	0.0016	0.0014
C_2 Best Parameters	var_smoothing: 1e-11	tolerance: 1e-06 solver: "lbfgs" C: 1 multi_class: "ovr"	n_estimators: 100 criterion: "entropy" max_depth: 50 min_samples_leaf: 10	n_neighbors: 20 algorithm: "kd-tree" leaf_size: 50 weights: uniform	C: 3 kernel: "rbf" degree: 5
C_3 Accuracy (μ)	0.6449	0.7362	0.7367	0.7296	0.7308
C_3 Accuracy (σ)	0.0012	0.0006	0.0008	0.0011	0.0009
C_3 Best Parameters	var_smoothing: 1e-11	tolerance: 1e-08 solver: "saga" C: 0.1 multi_class: "multinomial"	n_estimators: 100 criterion: "gini" max_depth: 30 min_samples_leaf: 2	n_neighbors: 20 algorithm: "brute" leaf_size: 30 weights: uniform	C: 3 kernel: "rbf" degree: 5
C_4 Accuracy (μ)	0.7301	0.9198	0.9188	0.9193	0.9195
C_4 Accuracy (σ)	0.0012	0.0003	0.0004	0.0003	0.0004
C_4 Best Parameters	var_smoothing: 1e-11	tolerance: 1e-03 solver: "saga" C: 10 multi_class: "ovr"	n_estimators: 30 criterion: "gini" max_depth: 30 min_samples_leaf: 2	n_neighbors: 10 algorithm: "brute" leaf_size: 10 weights: distance	C: 3 kernel: "rbf" degree: 5

Following Table 41 summarizes the results for FM3 using the ordinal ensemble method.

Table 41 - Ordinal Ensemble Classification Results for FM3

	Naive Bayes	Logistic Regression	Random Forest	K-Nearest Neighbors	SVM
10-fold Accuracy (μ)	0.2096	**0.2573**	0.2430	0.2351	0.2477
10-fold Accuracy (σ)	0.0018	0.0016	0.0045	0.0018	0.0027
Precision (μ)	0.2299	0.2596	0.2453	0.2343	0.2528
Precision (σ)	0.0021	0.0017	0.0051	0.0017	0.0034
Recall (μ)	0.2096	0.2573	0.2430	0.2351	0.2477
Recall (σ)	0.0018	0.0016	0.0045	0.0018	0.0027
F1-Score (μ)	0.1949	0.2478	0.2398	0.2319	0.237
F1-Score (σ)	0.0016	0.0017	0.0046	0.0017	0.0029
OC_β(μ)	0.5684	**0.4952**	0.5152	0.5314	0.5013
OC_β(σ)	0.0014	0.0012	0.0032	0.0013	0.0019

4.3.2 NCA Ordinal Classifier: Results

The ordinal formulation premises did not hold for the NCA classifier and it performed poorly giving random guesses results and high OC_β errors.

Table 42 - Poor Results for Ordinal Ensemble with NCA Classifier

	Accuracy C_0	Accuracy C_1	Accuracy C_2	Accuracy C_3	Accuracy C_4	Accuracy Ensemble	OC_β error
FM1	0.2600	0.2067	0.1957	0.2339	0.2901	0.1600	0.6525
FM2	0.2374	0.1773	0.1787	0.2018	0.2718	0.1562	0.6564
FM3	0.2369	0.1889	0.1787	0.1998	0.2693	0.1558	0.6562

5 Discussion

5.1 Dataset Considerations

While the LIAR Dataset is indeed a valuable asset to address the fake news detection problem, some caution remarks are in order and there is some space for improvement in the data.

The false label history features from Table 17 stand out as a weak feature for predicting unseen data. Any statement from a new speaker, one not previously part of the dataset, will receive no prediction value from those features. The same can be said for the speaker feature itself.

The history features contain extremely correlated output value for each speaker. This can lead to both good and bad realizations. On the bright side, it brings huge attention to the speaker as a red flag for fake news. A speaker that has a reputation to be untruthful seems to continue on this path, according to the results in section 4.1.1. That is a useful conclusion to reach. On the other hand, it introduces a big overfitting bias in the dataset, meaning a lot more data on a huge range of political figures would be needed in order to reduce the chance for unseen data to produce poor results.

In section 4.1.1, the SVD was able to capture that information reducing the dimensionality to the history features themselves, generating extremely good results in the case of Nearest Neighbors Classification. Figure 11 and Table 19 show how simpler the data becomes and how both Random Forest classifier and KNN classifier achieve over 60% accuracy on a 6-class classification problem. Other classifiers did not have such improvements including the original work's [1] hybrid CNN/Bi-Directional LSTM achieving similar results in the mid 20-30% accuracy range.

Since the dataset relies on data extracted from PolitiFact [66] the class labels are designed to be interpreted by humans. It is still to be verified if the ordinal labels created by PolitiFact can generalize well in a machine learning scenario and be just as effective as it has been on fighting fake news via manual fact checking.

A very positive remark to be made towards the dataset is its size. The LIAR Dataset remains as one of the largest fake-news oriented dataset created so far. With a lot more fact-checking agencies starting their work around the globe the scientific machine learning community might benefit from more large-scale datasets soon. Hopefully a

standardized and machine learning oriented approach is taken in building those datasets in order to bring better prediction performance.

5.2 Feature Selection Insights

Before the actual classification results' compilation, there are plenty of insights that result from analyzing the features that remain after generating FM1, FM2 and FM3.

As a result of using output correlation for feature selecting metadata in FM1, Table 21 unearths valuable knowledge regarding misinformation. The data shows that when the speaker is not an actual person, such as in the case of a *chain e-mail*, *blog-posting* or *viral-image*, the statement has a higher chance of being false. The same happens with subjects such as *health-care* or *religion*. On the other hand, the subject *economy* and a higher *symbol_ratio* gives a higher chance of truth to the statement. Reinforcing the *symbol_ratio* truth inclination, the *NUMERAL_TOKEN* text variable also has truth correlation according to Table 22.

Feature matrices FM2 and FM3 presents similar truth correlation regarding *NUMERAL_TOKEN* and their metadata is processed with SVD which selected the subject features as most relevant to its components. This is an interesting result and could mean that the subject features have a higher generalization power even though they have less direct output correlation. Subjects in general can also be easily obtained through other text feature selection methods like word embedding or topic modeling and it's good to verify they are actually very relevant metadata features for detecting fake content.

Even though those realizations are important in the context of the fake news problem, they must be verified further. The LIAR Dataset has a decent size but is constrained to the USA's 2016 election campaign context while the fake news context is somewhat infinite. The more large-scale datasets are studied more accurate the insights and predictions will be.

5.3 Classifiers' Performance and Ordinal Formulation

Starting from the three feature matrices comparison, while FM1 provided a great number of insights described in previous section, FM2 performed best in both ordinal and non-ordinal classifications.

The best performing classifier was a simple Logistic Regression. After being hyperparameter tuned, it achieved an average 10-fold cross-validation accuracy of 26.85% and an OC_β error of 49.09% as stated in Table 31. The average is taken running

10-fold cross-validation 10 times to dilute randomness of dataset splitting, since fixing the random seed varied the result both ways in approximately 1%.

Before starting ordinal formulation, this result is quite satisfactory. A much simpler classifier such as a Logistic Regression performs on par with a much more complex neural network architecture, which achieved 27.4% in a single cross-validation experiment. Additionally, a simpler classifier is more explainable and provides useful insights.

The ordinal formulation of the problem is used as an attempt to take advantage of the ordinality of the output classes and increase overall accuracy and diminish error impact (OC_β). Even though the OC_β error is not used intrinsically in the loss function, it is expected to decrease when using ordinal classification framework if the classes are indeed ordinal in nature.

While indeed achieving lower OC_β error and higher accuracy, the improvements were minimal and inside the standard deviation variation could be considered equal. This does not invalidate the ordinality of the classes or the framework proposed by [3] otherwise the accuracy was supposed to be lower than the regular classification. The most performing classifier was also the Logistic Regression in the FM2 matrix achieving an average 26.87% accuracy and 48.75% OC_β error (Table 39).

Most classifiers had slight improvements or remained at the same range of performance when using the ordinal formulation except for the NCA classifier that performed poorly in ordinal manner, needing additional research (Table 42) to find the reason for it. It is important to remark that other classifiers simpler than a neural network such as Random Forest or SVM also performed similarly to Logistic Regression achieving results pretty close to the 26% mark. This might be due to the fact that neural networks do not perform at their best when using limited size datasets due to their complexity. Neural Networks' power relies on huge amounts of data, which is not usually available for fake news detection just yet.

Additional metrics such as precision, recall and F1-score are also available for reference in the results section in both regular and ordinal classifications.

Lastly, it is interesting to analyze the results of the individual binary classifiers of the ordinal formulation such as in Table 38. As the classifiers range from C_0 to C_4, the threshold between true and false statement moves from one extreme to the other. The accuracy of the binary classifiers follows that pattern being much higher when the

threshold is closer to one of the extremes (C_0 and C_4) and being lower as the threshold travels in the middle (C_1, C_2, C_3). This is explained by misinformation being indeed a continuous and ambiguous concept and therefore it is easier to detect an extreme lie than a partial lie.

Following, as an illustration, is a closer look at the performance of the 5 binary classifiers for the best performing Logistic Regression using FM2.

Table 43 - Ordinal binary classifiers accuracy varying with true/false threshold

	C_0	C_1	C_2	C_3	C_4
Accuracy (μ)	0.8402	0.6676	0.6433	0.7403	0.9207

5.4 Fuzzy Interpretation

The difficulty of labeling an ambiguous dataset with 6 ordinal classes could make Fuzzy Logic [81] useful to this problem [82]. With Fuzzy Logic, a statement can be a member of multiple classes to different degrees, which is mostly why a statement becomes *"mostly-true"* or *"barely-true"* to begin with.

Similarly, even with no fuzzy output variables, the probability output of classifiers could be used for that purpose. Therefore, since the classes measure a single concept, the untruthiness of a statement, a single fuzzy output might make more sense in aiding decision making after prediction.

6 Conclusions

It became clear with the results of the experiment that using artificial intelligence to find untruthiness in data is not an easy task. However, this work has achieved reasonable results with simple classifiers.

All of the preprocessing tasks have done good jobs exposing relevant features to the classifiers even without the use of word embeddings. FM1 and FM2 were particularly more important than FM3 when bringing useful insights. The metadata seemed to be enough to represent semantics and context information. Results shown in Table 21 and Table 22 can give a grasp of what the patterns are in this dataset, however, a much bigger dataset analysis would be required to check consistency, especially with potential polemic results.

The subjectivity of the untruthiness levels becomes obvious when looking at the results of the binary classifiers in Table 38.. This means that, for any fake news detection system, the results will vary according to these untruthiness thresholds or, in other words, the amount of subjectivity the system tries to capture. This is also valid for the task of dataset labeling. Many levels of untruthiness to choose from leads to eventual inconsistency in the labeling process affecting the patterns to be detected using machine learning algorithms.

This experiment, however, managed to achieve an accuracy in the order of 26.87% accuracy in a 6-class classification problem. Random guesses would result in 16.67% and majority guesses would result in 20.5%. The ordinal formulation did not reduce the OC_{β} significantly as expected, but the assumptions from section 3.4.2 still held. One possibility is that labeling subjectivity could be introducing noise to the actual ordering of the classes, preventing better results with ordinal classification.

Fake news detection in machine learning is still a work in progress. It depends on a highly precise human labeling process when constructing large datasets. Even then, each dataset will, most likely, be constrained to its domain.

What appears to be extremely useful is the insights obtained from datasets like this. For that reason, this work is expected to contribute to verifying techniques that can be used in text and metadata not only for classification itself but also to mine valuable information regarding true or false statements. Additionally, the experiments showed that similar results can be achieved with simpler classifiers using appropriate feature selection, especially when datasets are limited to be used with deep neural networks. Lastly, this work contributes to reinforcing the need of a standardized way to capture the different levels of untruthiness in statements that proved hard to deal with even when modeling the classification in an ordinal manner.

7 References

[1] W. Y. Wang, ""Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection," em *Association for Computational Linguistics - https://arxiv.org/abs/1705.00648,* Vancouver, 2017

[2] Cardoso, Jaime S.; Sousa, Ricardo; INESC Porto, "Measuring the Performance of Ordinal Classification," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 25, nº 8, pp. 1173-1195, 2011

[3] E. &. H. M. Frank, "A Simple Approach to Ordinal Classification," *Lecture Notes in Computer Science,* vol. 2167, pp. 145-156, 2001

[4] E. Shearer, "Fact Tank," Pew Research, 10 December 2018. [Online]. Available: https://www.pewresearch.org/fact-tank/2018/12/10/social-media-outpaces-print-newspapers-in-the-u-s-as-a-news-source/

[5] B. N. Anand, "The U.S. Media's Problems Are Much Bigger than Fake News and Filter Bubbles," Harvard Business Review, 5 January 2017. [Online]. Available: https://hbr.org/2017/01/the-u-s-medias-problems-are-much-bigger-than-fake-news-and-filter-bubbles

[6] R. Gunther, P. Beck and E. Nisbet, "Fake News May Have Contributed to Trump's 2016 Victory," Ohio State University, 8 March 2018. [Online]. Available: https://assets.documentcloud.org/documents/4429952/Fake-News-May-Have-Contributed-to-Trump-s-2016.pdf.

[7] D. Phillips, "Brazil battles fake news 'tsunami' amid polarized presidential election," The Guardian, 10 October 2018. [Online]. Available: https://www.theguardian.com/world/2018/oct/10/brazil-fake-news-presidential-election-whatsapp-facebook

[8] K. Ponniah, "WhatsApp: The 'black hole' of fake news in India's election," BBC News, 6 April 2019. [Online]. Available: https://www.bbc.com/news/world-asia-india-47797151

[9] M. Stencel e R. Griffin, "Fact-checking triples over four years," Duke Reporter`s Lab, 22 2 2018. [Online]. Available: https://reporterslab.org/fact-checking-triples-over-four-years/

[10] J. Porter, “Facebook is turning its fact-checking partners loose on Instagram,” The Verge, 7 5 2019. [Online]. Available: https://www.theverge.com/2019/5/7/18535116/instagram-fact-checking-facebook-dashboard-misinformation

[11] T. Henriksson, “Facebook: labelling of false stories reduces the spread of fake news by 80%,” World News Publishing Focus, 12 October 2017. [Online]. Available: https://blog.wan-ifra.org/2017/10/12/facebook-labelling-of-false-stories-reduces-the-spread-of-fake-news-by-80

[12] Facebook, “Removing Coordinated Inauthentic Behavior From Israel,” Facebook Newsroom, 16 May 2019. [Online]. Available: https://newsroom.fb.com/news/2019/05/removing-coordinated-inauthentic-behavior-from-israel/

[13] M. Temming, “How Twitter bots get people to spread fake news,” Science News, 20 November 2018. [Online]. Available: https://www.sciencenews.org/article/twitter-bots-fake-news-2016-election

[14] E. Strickland, “AI-human partnerships tackle "fake news": Machine learning can get you only so far-then human judgment is required,” *IEEE Spectrum,* vol. 55, nº 9, pp. 12-13, 2018

[15] W. Shang, M. Liu, W. Lin e M. Jia, “Tracing the Source of News Based on Blockchain,” em *IEEE/ACIS,* Singapore, 2018

[16] E. J. A. Newman, M. Garry, C. Unkelbach, D. M. Bernstein, D. S. Lindsay e R. A. Nash, “Truthiness and falsiness of trivia claims depend on judgmental contexts.,” *Journal of Experimental Psychology: Learning, Memory, and Cognition,* vol. 45, nº 5, pp. 1337-1348, 2015

[17] Rugile, “30 Fake Viral Photos People Believed Were Real,” Bored Panda, [Online]. Available: https://www.boredpanda.com/fake-news-photos-viral-photoshop/?utm_source=google&utm_medium=organic&utm_campaign=organic

[18] Stanford University/Michael Zollhoefer; The Max Planck Institute for Informatics; University of Washington; Carnegie Mellon; University Colorado Denver, “https://edition.cnn.com/interactive/2019/01/business/pentagons-race-against-deepfakes/,” CNN Business, [Online]. Available:

https://edition.cnn.com/interactive/2019/01/business/pentagons-race-against-deepfakes/

[19] D. Shapiro, “Artificial Intelligence and Bad Data,” Towards Data Science, 6 November 2017. [Online]. Available: https://towardsdatascience.com/artificial-intelligence-and-bad-data-fbf2564c541a

[20] D. Funke, “Reporters: Stop calling everything ‘fake news’,” Poynter, 29 August 2018. [Online]. Available: https://www.poynter.org/fact-checking/2018/reporters-stop-calling-everything-fake-news/

[21] E. Van Duyn e J. Collier, “Priming and Fake News: The Effects of Elite Discourse on Evaluations of News Media,” *Mass Communication and Society,* vol. 22, nº 1, pp. 29-48, 2019

[22] C. Wardle, “Fake news. It’s complicated.,” First Draft, 16 February 2017. [Online]. Available: https://firstdraftnews.org/fake-news-complicated/

[23] P. R. S. S.-S. J. T. Hamid Karimi, “Multi-Source Multi-Class Fake News Detection,” em *Association for Computational Linguistics*, Santa Fe, 2018

[24] P. Mike Tamir, “Detecting Fake News using Machine Learning,” Analytics Vidhya, 11 April 2019. [Online]. Available: https://medium.com/analytics-vidhya/detecting-fake-news-using-machine-learning-with-mike-tamir-ph-d-dd97277742ff

[25] S. &. K. A. P. Parikh, “Media-Rich Fake News Detection: A Survey.,” em *MIPR 2018*, Miami, 2018

[26] D. Harrison e D. Rubinfeld, “Housing Values in Suburbs of Boston,” [Online]. Available: https://www.kaggle.com/c/boston-housing/overview

[27] Y. LeCun, C. Cortes e C. J. Burges, “THE MNIST DATABASE,” [Online]. Available: http://yann.lecun.com/exdb/mnist/

[28] S. P. M. A. S. S. E. D. T. J. B. G. K. K. Mehdi Allahyari, “A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques,” University of Georgia, 28 July 2017. [Online]. Available: https://arxiv.org/abs/1707.02919

[29] J. Brownlee, “A Gentle Introduction to the Bag-of-Words Model,” Machine Learning Mastery, 9 October 2017. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[30] D. Jurafsky, “Language Modeling - Introduction to N-grams,” Stanford University, [Online]. Available: https://web.stanford.edu/class/cs124/lec/languagemodeling2017.pdf

[31] E. G. P. B. T. M. Armand Joulin, “Bag of Tricks for Efficient Text Classification,” Facebook AI Research, 6 July 2016. [Online]. Available: https://arxiv.org/abs/1607.01759

[32] P. R. Christopher D. Manning e Hinrich Schütze, “Scoring, term weighting & the vector space model,” em *Introduction to Information Retrieval*, Cambridge University Press. 2008, 2008, pp. 109-133

[33] B. A. Asaad e M. Erascu, “A Tool for Fake News Detection,” em *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Timisoara, 2018

[34] Á. A. I. C. Edmundo Hoyle, “GLOBOPLAY THUMBNAILS: USING AI TO EXTRACT THE BEST FRAME TO REPRESENT DRAMA SERIES,” em *IBC 2019*, Amsterdam, 2019

[35] P. e. al., “Scikit-learn: Machine Learning in Python,” *JMLR 12*, pp. 2825-2830, 2011

[36] R. R. a. P. Sojka, “Software Framework for Topic Modelling with Large Corpora,” em *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, Valletta, Malta, ELRA, 2010, pp. 45-50

[37] Tomas Mikolov; Ilya Sutskever; Kai Chen; Greg Corrado; Jeffrey Dean; Google Inc., “Distributed representations of words and phrases and their compositionality,” *NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems*, vol. 2, pp. 3111-3119, 2013

[38] D. M. Blei, A. Y. Ng e M. I. Jordan, “Latent Dirichlet Allocation,” *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 2003

[39] S. Li, “Topic Modeling and Latent Dirichlet Allocation (LDA) in Python,” Towards Data Science, 31 May 2018. [Online]. Available: https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24

[40] “FakerFact,” 2018. [Online]. Available: https://www.fakerfact.org/about

[41] T. M. Quoc V. Le, “Distributed Representations of Sentences and Documents,” Google Inc, 22 May 2014. [Online]. Available: https://arxiv.org/abs/1405.4053v2

[42] W. C. W. Herley Shaori Al-Ash, “Fake News Identification Characteristics Using Named Entity Recognition and Phrase Detection,” em *10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Kuta, Indonesia, 2018

[43] S. A. Benjamin D. Horne, “This Just In: Fake News Packs a Lot in Title, Uses Simpler, Repetitive Content in Text Body, More Similar to Satire than Real News,” ArXiv, 2017. [Online]. Available: https://arxiv.org/abs/1703.09398

[44] “Natural Language Toolkit,” NLTK Project, 2005. [Online]. Available: https://www.nltk.org

[45] M. a. M. I. Honnibal, “Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,” Spacy, 2017. [Online]. Available: https://spacy.io/

[46] Mark A. Hall; The University of Waikato, *Correlation-based Feature Selection for Machine Learning*, Hamilton, New Zealand, 1999

[47] S. &. S. U. &. T. S. &. D. S. &. S. D. &. S. R. &. P. S. &. L. M. Mishra, “Principal Component Analysis,” *International Journal of Livestock Research,* 2017

[48] MIT, “Singular Value Decomposition (SVD) tutorial,” [Online]. Available: http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm

[49] K. B. G. L. Yi Mao, “Dimensionality Reduction for Text using Domain Knowledge,” *ACL Anthology,* vol. Coling 2010: Posters, pp. 801-809, 2010

[50] L. v. d. Maaten e G. Hinton, “Visualizing Data using t-SNE,” *Journal of Machine Learning Research ,* vol. 9, pp. 2579-2605, 2008.

[51] Jacob Goldberger, Sam Roweis, Geoff Hinton, Ruslan Salakhutdinov; Department of Computer Science, University of Toronto, “Neighbourhood Components Analysis,” [Online]. Available: https://www.cs.toronto.edu/~hinton/absps/nca.pdf

[52] A. Karpathy, “The Unreasonable Effectiveness of Recurrent Neural Networks,” Andrej Karpathy blog, 21 May 2015. [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[53] C. Olah, “Colah's blog,” 27 August 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[54] Oluwaseun Ajao; Deepayan Bhowmik; Shahrzad Zargari; C3Ri Research Institute, "Fake News Identification on Twitter with Hybrid CNN and RNN Models," Arxiv.org, 29 Jun 2018. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1806/1806.11316.pdf

[55] F. Y. A. M. Sohan De Sarkar, "Attending Sentences to detect Satirical Fake News," em *COLING*, 2018.

[56] A. Karpathy, "CS231n Convolutional Neural Networks for Visual Recognition," Stanford.EDU, 2019. [Online]. Available: http://cs231n.github.io/convolutional-networks/

[57] C. Nicholson, "A Beginner's Guide to Important Topics in AI, Machine Learning, and Deep Learning," Skymind, 2019. [Online]. Available: https://skymind.com/wiki/attention-mechanism-memory-network

[58] N. S. N. P. J. U. L. J. A. N. G. L. K. I. P. Ashish Vaswani, "Attention Is All You Need," Arxiv.org, December 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[59] K. Dembczynski, W. Kotlowski e S. Roman, "Ensemble of Decision Rules for Ordinal Classification with Monotonicity Constraints.," em *Rough Sets and Knowledge Technology, Third International Conference*, Chengdu, China, 2008.

[60] Jaime S. Cardoso; Joaquim F. Pinto da Costa; Universidade do Porto, "Learning to Classify Ordinal Data: The Data Replication Method," *Journal of Machine Learning Research,* vol. 8, pp. 1393-1429, 2007

[61] M. C. Heredia-Gomez, S. Garcıa, P. A. Gutierrez e F. Herrera, "OCAPIS: R package for Ordinal Classification And Preprocessing In Scala," Arxiv.org, 17 March 2019. [Online]. Available: https://arxiv.org/abs/1810.09733

[62] P. J. R. B. M. H. Michael J. Wurm, "Regularized Ordinal Regression and the ordinalNet R Package," Arxiv.org, 15 Jun 2017. [Online]. Available: https://arxiv.org/abs/1706.05003

[63] J. N. Gaudette L., "Evaluation Methods for Ordinal Classification," em *Canadian Conference on Artificial Intelligence*, 2009

[64] Jaime S. Cardoso, Ricardo Gamelas Sousa, University of Porto, "Measuring the Performance of Ordinal Classification," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 25, pp. 1173-1195, 2011

[65] W. Yang Wang, “Index of /~william/data,” University of California, Santa Barbara, 23 April 2017. [Online]. Available: https://sites.cs.ucsb.edu/~william/data/

[66] Poynter Institute, “PolitiFact,” 1 January 2007. [Online]. Available: https://www.politifact.com/

[67] Poynter Institute for Media Studies, “Poynter,” 1 January 1975. [Online]. Available: https://www.poynter.org/

[68] Tampa Bay Newspaper, “Tampa Bay Newspaper,” 1 January 1884. [Online]. Available: https://www.tampabay.com/

[69] Python Software Foundation, “python,” 2001. [Online]. Available: https://www.python.org/

[70] Jupyter Project and Community, “jupyter,” 2014. [Online]. Available: https://jupyter.org

[71] Igor Bichara; Carlos Eduardo Pedreira; Geraldo Xexéo, “fakenews-LIAR,” COPPE - PESC, 2019. [Online]. Available: https://github.com/ibichara/fakenews-LIAR

[72] Pandas Community, “Pandas Data Analysis Library,” NumFOCUS, 2014. [Online]. Available: https://pandas.pydata.org/

[73] NumPy developers, “NumPy,” NumFOCUS, 2001. [Online]. Available: https://www.numpy.org/

[74] N. Project, “Categorizing and Tagging Words,” 1 April 2019. [Online]. Available: https://www.nltk.org/book/ch05.html

[75] N. Project, “Accessing Text Corpora and Lexical Resources,” 1 April 2019. [Online]. Available: https://www.nltk.org/book/ch02.html

[76] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Computing in Science and Engineering*, vol. 9, nº 3, pp. 90-95, 2007.

[77] MathWorks, “Feature selection using neighborhood component analysis for classification,” [Online]. Available: https://www.mathworks.com/help/stats/fscnca.html

[78] The MathWorks, Inc., “MathWorks,” 1994. [Online]. Available: https://www.mathworks.com/products/matlab.html

[79] J. Brownlee, “Machine Learning Mastery,” 28 July 2017. [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

[80] S. Learn, “GridSearchCV,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[81] T. J. Ross, Fuzzy Logic with Engineering Applications, John Wiley & Sons, 2009

[82] Hassall, John; University of Wolverhampton, “Methods of Analysing Ordinal/Interval Questionnaire Data using Fuzzy Mathematical Principle,” Wolverhampton, 1999