Instituto Politécnico Nacional

Centro de Investigación en Computación

T E S I S

Fake News Spreaders Profiling and Fake News Detection in Social Media

para obtener el grado de:

Maestría en Ciencias de la Computación

presenta:

Ing. Sergio Arturo Damian Sandoval

Directores de tesis:

Dr. Francisco Hiram Calvo Castro
Dr. Alexander Gelbukh

Ciudad de México

Junio 2021

SIP-14
REP 2017

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 12:00 horas del día 17 del mes de mayo del 2021 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Posgrado del: **Centro de Investigación en Computación** para examinar la tesis titulada:

"Fake News Spreaders Profiling and Fake News Detection in Social Media"

del (la) alumno (a):

Apellido Paterno:	DAMIAN	Apellido Materno:	SANDOVAL	Nombre (s):	SERGIO ARTURO

Número de registro: B 1 9 0 3 9 4

Aspirante del Programa Académico de Posgrado: **Maestría en Ciencias de la Computación**

Una vez que se realizó un análisis de similitud de texto, utilizando el software antiplagio, se encontró que el trabajo de tesis tiene 7 % de similitud. **Se adjunta reporte de software utilizado**.

Después que esta Comisión revisó exhaustivamente el contenido, estructura, intención y ubicación de los textos de la tesis identificados como coincidentes con otros documentos, concluyó que en el presente trabajo **SI** [ ] **NO** [X] **SE CONSTITUYE UN POSIBLE PLAGIO.**

JUSTIFICACIÓN DE LA CONCLUSIÓN: *(Por ejemplo, el % de similitud se localiza en metodologías adecuadamente referidas a fuente original)*

Las coincidencias son mínimas en relación con el resto del texto. Éstas corresponden a menciones específicas de materiales y métodos en los que esta tesis se basa.

****Es responsabilidad del alumno como autor de la tesis la verificación antiplagio, y del Director o Directores de tesis el análisis del % de similitud para establecer el riesgo o la existencia de un posible plagio.**

Finalmente, y posterior a la lectura, revisión individual, así como el análisis e intercambio de opiniones, los miembros de la Comisión manifestaron **APROBAR** [X] **NO APROBAR** [ ] la tesis, en virtud de los motivos siguientes:

Muestra un método novedoso para seleccionar características, y demuestra su efectividad en una tarea de identificación de perfiles que propagan noticias falsas. También hace un análisis de cuáles son las características de dichos perfiles.

COMISIÓN REVISORA DE TESIS

Dr. Francisco Hiram Calvo Castro
Director de tesis

Dr. Grigori Sidorov

Dr. Carlos Alberto Duchanoy Martínez

Dr. Alexander Gelbukh
2º. Director de Tesis

Dr. Ildar Batyrshin

Dra. Gina Gallegos García

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION

Dr. Marco Antonio Moreno Ibarra
PRESIDENTE DEL COLEGIO DE PROFESORES

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, siendo el día **07** del mes de **Junio** del año **2021**, el que suscribe **Sergio Arturo Damian Sandoval** alumno del Programa de **Maestría en Ciencias de la Computación**, con número de registro **B190394**, adscrito al **Centro de Investigación en Computación**, manifiesta que es el autor intelectual del presente trabajo de Tesis bajo la dirección del **Dr. Francisco Hiram Calvo Castro** y del **Dr. Alexander Gelbukh** y cede los derechos del trabajo intitulado "**Fake News Spreaders Profiling and Fake News Detection in Social Media**", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o directores del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección **sergio.damian181091@gmail.com**. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Sergio Arturo Damian Sandoval

Resumen

Actualmente, Internet se ha convertido en la fuente principal de information debido a su velocidad de divulgar noticias. Pero el incremento de la divulgación de desinformación en los medios sociales también se ha incrementado durante estos años, teniendo un alto impacto en la sociedad. Uno de los principales tipos de desinformación son las noticias falsas, las cuales son consideradas como textos que contienen información incorrecta acerca de algún hecho, pretendiendo cambiar la opinión de los usuarios acerca del tema.

El presente trabajo busca desarrollar una solución a este problema, abordando dos tareas relacionadas entre sí: Detectar noticias falsas en artículos y detectar posibles cuentas de usuario que divulgan noticias falsas, de forma que se tengan diferentes alternativas para evitar el consumo de este tipo de información. Este trabajo propone analizar clasificadores de aprendizaje máquina, los cuales a través de N-gramas y características estadísticas descriptivas, se pueda seleccionar el mejor clasificador que cumpla el objetivo. Además se considera usar herramientas de Inteligencia Artificial Explicable (XIA), para entender el tipo de características que son relevantes para el problema, en diferentes corpus. Este trabajo se enfoca en los lenguajes de inglés y español, con la intención de prevenir el consumo de noticias falsas en dos de los lenguajes más hablados del mundo.

La solution propuesta supera los resultados encontrados en el estado del arte (aquellos que usan el mismo corpus), con una exactitud media del 0.7825 para la detección de divulgadores de noticias falsas para ambos lenguajes y además es una alternativa para la detección de noticias falsas, concluyendo esto último cuando se comparan los resultados con el estado del arte que aborda esa perspectiva del problema.

Abstract

Nowadays, Internet has become the main information source due to its fast way of spreading news. But, the rise of misinformation spreading in social media has also been increased during recent years, having a huge impact in society. One of the main type of misinformation is fake news. Fake news are considered to have incorrect information about a fact, pretending to change the user's opinion about the topic.

The present work attempts to develop a solution to this problem, addressing to tasks: to detect fake news from articles, and to detect possible fake news spreaders accounts, in order to have different alternatives to avoid the consume of this kind of information. This work proposes to analyze machine learning classifiers and through word n-grams and statistical features, select the best classifier that can accomplish the objective. In addition, the use of Explainable Artificial Intelligence methods (XIA) are considered, to understand what kind of features are relevant to the problem in different corpus. This work focuses in English and Spanish languages, with the intention to prevent the consume of fake news in two of the most spoken languages in the world.

The solution surpasses the State of the Art results (using the same corpora), with a mean accuracy of 0.7825 for fake news spreaders detection in both languages and it is a good alternative for fake news detection, concluding this when the work is compared with the State of the Art that manages this perspective of the problem.

Acknowledgments

"Live as if you were to die tomorrow. Learn as if you were to live forever."
– Mahatma Gandhi

I would like to express my apreciation and sincere thanks to the *Centro de Investigación en Computación* and *Instituto Politécnico Nacional* for giving me the opportunity to be part of the community and for all the grateful experiences I have had during these years.

My deepest gratitude to the *Centro Nacional de Ciencia y Tecnología* (CONACyT) and the federal goverment for the financial support that empowered my postgraduate education in Mexico.

Special thanks to my advisors, Dr. Hiram Calvo and Dr. Alexander Gelbukh for all the patience, advices, experiences, and for all the words that gave me encouragement and motivation to develop this work.

My profound admiration to my comittee members for all the opinions, knowledge and experience provided.

To my family, specially my mother, my brothers and my girlfriend Lupita who mean the most to me for supporting me through this years and being the special motivation to study my master's degree improving my life trajectory.

To my friends, for their magnificent company and friendship during my stay at *Centro de Investigación en Computación*.

- Sergio Arturo Damian Sandoval, May 2021

Contents

List of Figures

List of Tables

1 Introduction

The rise of social media has given the opportunity to users to publish and share content online in a very fast way. This easiness of publishing content has led to an increase in the amount of misinformation that is shared among users [2]. In fact, the propagation of fake news has been proven to be faster than real news and is causing several negative consequences in several areas of society such as economics, politics, medicine, culture, etc [3]. Information consumers usually do not detect or prevent sharing false information because no fact-checking systems are usually applied in social media and tend to be manipulated for, most of the time, strategic reasons. Or just they are not concerned of receiving this type of information from relatives and friends. One example of this manipulation is when an American citizen irrupted in a restaurant with a gun. He argued that place usually committed infant abuse managed by president ex-candidate Hillary Clinton. Of course this information was false, but the bad habit of no further research drove to a critical situation [4].

Understanding whether an article or a post is related with fake news content can be a complex task. Actually, fact-checking experts categorize fake information in multiple groups, according to the content type. Another problem is the time when the article was written. Some information could be truth last year but today is uncertain or fake. For example, the debate about if planet Pluto is a planet or not. Changes in society can affect the durability of a solution that aims to help with the problem.

But there are some other characteristics that do not change at all when time goes by. An author is used to redact with a certain vocabulary, expressions and/or sentiments when share information. All of these characteristics can be detected as a pattern that could help to determine if a new article or post is related with *misinformation*; hence, this work focuses in two main classification tasks: First, given a set of texts, to identify word patterns that are related with fake news. Second, given a set of texts per a certain amount of users, to determine if the user is prone to share false information. Having this mind, it is important to define the focus of this work and some general definitions that are relevant to comprehend this work's panorama.

1.1 Author Profiling

Author Profiling is the analysis of features and patterns that can be related to a group of people in order to classify them according to specific categories. Considering this definition, this work pretends to analyze and identify user's write-style and content-based features

in order to divide users in two groups: those who spread fake news and those who do not. One important aspect to mention is that human users can have multiple profiles or accounts in social media. Also, multiple human users can share a single account, so, this work classifies user accounts instead of individual human user's profiles.

1.2 Misinformation and Fake News

According to [5], misinformation is every false statement that can manipulate people hiding true facts. There are many concepts associated with misinformation, such as rumor, spam, disinformation and fake news. We can find various examples of every kind of misinformation in websites, and works that focus on one or some of this concepts. To define what this work examines, the following definitions are considered:

1. Fake news is a news article that intentionally misleads the readers, and it is actually false.
2. Misinformation can be broadly used to treat information as false information.
3. Disinformation is a piece of inaccurate information that is spread intentionally to mislead people.

This work defines "Fake news" as all text content that misleads readers or information that is inaccurate. Articles, posts or texts that contain some information related to identified fake news are considered fake news as well. On the other hand, articles, posts or texts that warn about fake news are considered real news. In addition, from this section to the rest of this work, *texts* are referred to articles, posts, messages or any other kind of written information in social media.

1.3 Fake News Spreaders

It usually cannot be assumed that a single human user is the only one who manages an account or a profile in social media. In fact, users can share multiple accounts or even have multiple accounts for many reasons. Therefore, this work defines a fake news spreader as a user account in a social media platform that has posted at least one text related to fake news (intentionally or unintentionally) and with any other intention that clarifies it. This can be processed retrieving a certain amount of posts for an specific user and examining if the texts are related with fake news using fact-checking websites or by expert's opinions about the topic.

1.4 Problem Statement

Understanding whether a text is fake or not is a very challenging task for users who in their majority are not experts [2]. Due to general social media users are not used to

investigate through reliable sources the information they consume, they tend to increase the misinformation sharing through society in an intentional or an unintentional way.

With this in mind, this work considers the following analysis:

Accounts that do not spread fake news may have a set of different characteristics compared to accounts that tend to share fake news. As a hypothesis, they may use different linguistic patterns when they share posts compared to fake news spreaders. This can be considered as a binary classification task because it does not matter how many fake texts an account has shared, the objective is to be able to determine if a text is fake and by this, determine if an account shared fake content.

1.5 Thesis Focus

The present work considers the following hypothesis: user accounts that spread misinformation through social media have stylistic or content-based features that are different from accounts that do not tend to share fake content and therefore, can be detected using Artificial Intelligence and Natural Language Processing tools in order to classify them correctly. The present work focuses on English and Spanish texts, with an individual process treatment. Some solution approaches than can be found in the State of the Art consider additional features such as social media metadata, images, etc., however, a deep study of such features is out of the scope of this work.

1.6 General Objective

The general objective of this work is to propose a solution approach that can detect fake news and fake news spreaders in social media using machine learning tools and to interpret the fake news phenomenon through this solution in order to prevent this kind of behavior.

1.7 Specific Objectives

In order to accomplish our general objective, the present thesis aims to achieve the following specific objectives:

1. To analyze the State of the Art methods and solutions
2. To analyze and experiment with different text based features
3. To examine feature selection tools
4. To propose a stacking ensemble model attempting to combine the different characteristics that multiple classifiers have and produce a solution approach
5. To interpret results obtained by the solution proposal using Explainable Artificial Intelligence (XAI)
6. To compare results with the State of the Art

1.8 Expected Contributions

The expected contributions of this work are the following:

1. To examine and propose interpreting tools as feature selection methods
2. To compare different feature sets in English and Spanish corpora
3. To compare results for different machine learning algorithms
4. To interpret results, and explain the model's behavior

1.9 System Specifications

The solution approach was developed in the Google Colaboratory platform, using Python 3.7 and machine learning, and natural language processing libraries: Scikit-learn, NLTK. SHAP and LIME are considered for model's interpretations.

1.10 Thesis Outline

The present work is organized as follows:
Chapter 2 focuses on the technical definitions that are part of the present work.
Chapter 3 presents the State of the Art of the problem.
Chapter 4 describes the steps followed in order to build the solution approach of this work.
Chapter 5 presents the results obtained by developing the solution in multiple corpora and how the final models are built.
Chapter 6 analyzes the final models using test datasets, presents the interpretation for every model and compare results with the state of the art.
Chapter 7 expresses the final conclusions and future work.

2 Theoretical Framework

2.1 Text Representations for Machine Learning

Machine learning algorithms are mostly fed only with numerical representations, hence there are some text transformations that are commonly used for this kind of tasks. This chapter describes the concepts and definitions of methods and techniques used by the present work.

2.1.1 N-Grams

Tokenization is the process of dividing the text in subsets of characters. N-grams is one way that tokenization can be applied and it is the procedure of splitting text into consecutive tokens, (tokens are small representations of text that represent a concept in a context, such as words, ideas, etc.) usually using spaces or punctuation signs as the token dividers. The n in n-grams represents the number of consecutive tokens that are joined and can be interpreted as a single one. N-grams can be built from word and characters and they can be combined in order to get better performance. N-grams can contribute to analyze the text and transform it in a numerical representation that can be processed with a machine learning algorithm. Depending of the problem, some n-grams are going to be relevant, and others may cause noise to the problem. The n-grams analyzed in this work are the following:

Word Unigrams

Unigrams are usually the main n-grams because any other n-gram can be constructed from them. In English and Spanish language, space is the character that splits words and it is usually the most common splitting character when building unigrams. An example is presented in Figure 2.1.

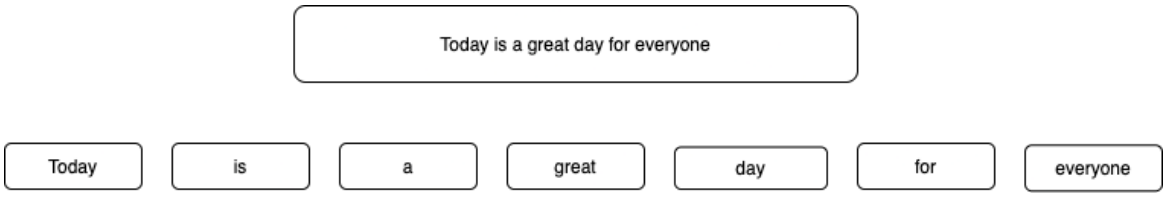

Figure 2.1: Example of unigrams

Word Bigrams

Bigrams consists on a pair of consecutive unigrams found in a text. It pretends to correlate unigrams and start making general ideas about the context of the work. One disadvantage is the huge dimensionality they can add to the problem. An example is presented in Figure 2.2.

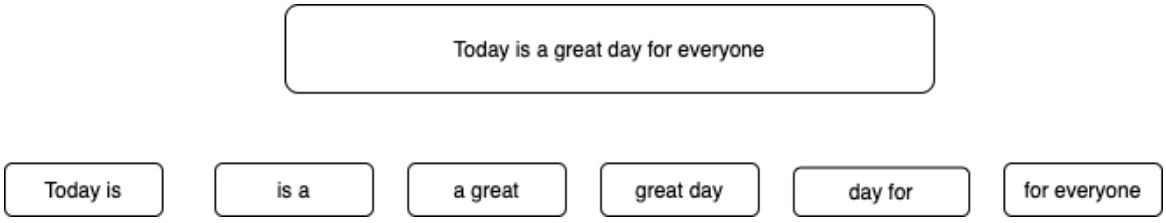

Figure 2.2: Example of bigrams

Word Skipgrams

Skipgrams can be of size two and further. They are defined as n-grams that appear in a certain number of gaps. Gaps are sequence n-grams that are skipped. An example of some types of skipgrams are presented in Figure 2.3.

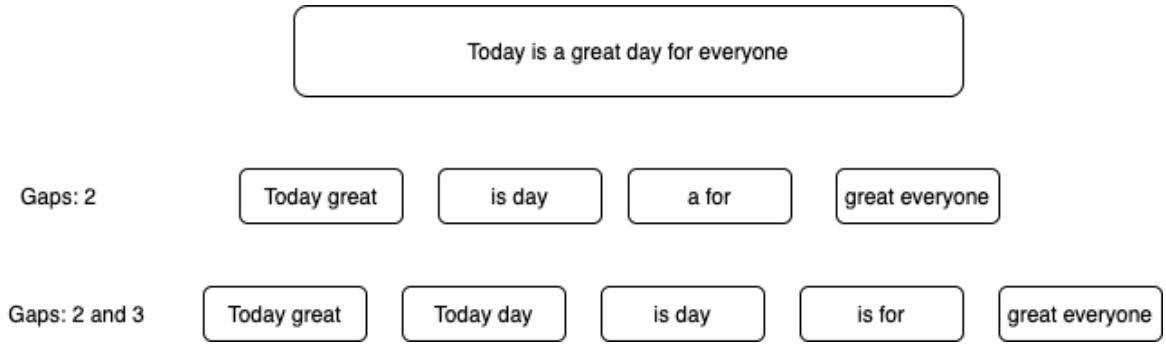

Figure 2.3: Example of 2-skipgrams using 2 and 2,3 gaps

When working with character n-grams it is difficult to interpret why certain features are important and why others does not, for this reason, they were not considered for this work.

2.1.2 TF-IDF

The concept TF-IDF stands for term frequency-inverse document frequency. This concept helps to understand how important a word is to a given document in a corpus. TF-IDF has two parts: Term Frequency and Inverse Document Frequency. The term frequency indicates the frequency of each of the words present in the document or dataset [6]. Thus, its definition is given as follows:

$$TF(t) = \frac{\text{Number of times term } t \text{ appears in a document}}{\text{Total number of terms in the document}} \tag{2.1}$$

IDF define some weighing down of the frequent terms while scaling up the rare ones, which decides the importance of each word [6]. We will achieve this with the following definition:

$$IDF(t) = log10(\frac{\text{Total number of documents}}{\text{Number of documents with term } t \text{ in them}}) \tag{2.2}$$

So, the calculation of TF-IDF is given as follows:

$$\text{TF-IDF} = TF(t) \cdot IDF(t) \tag{2.3}$$

2.2 Feature Engineering

One of the most popular techniques in machine learning is Feature Engineering. It consists in looking for additional features which can be calculated and discovered through experts in the area. The feature engineering process can be described as follows [7]:

1. Brainstorm about which features are relevant
2. Decide what features might improve the model performance
3. Create new features
4. Determine if the new features add to the model performance; if not, drop them
5. Go back to Step 1 until the performance of the model meets its designer's expectations

2.3 Feature Selection

It is common for a few features to be responsible for the majority of the information signal and the rest of the features are just mostly noise. It is important to lower the amount of input features for a variety of reasons including [7]:

1. Reducing the multi collinearity of the input features will make the machine learning model parameters easier to interpret
2. Reducing the time required to run the model and the amount of storage
3. The smaller number of input features a model requires, the easier it is to explain it
4. As the model has more features to describe the target, it might be able to describe the data more precisely, but it will not generalize with new data points, so the model will overfit the data. This is known as the curse of dimensionality.

Although there are many feature selection methods that can be found in literature, this work experiments with model interpretation methods (see Section 2.5) as feature selectors. There are three strategies for feature selection, and this work focuses on the one called *Iterative Feature Selection*. In this strategy, a series of models are built, with varying numbers of features. There are two basic methods: starting with no features and adding features one by one until some stopping criterion is reached, or starting with all features and removing features one by one until some stopping criterion is reached. Because a series of models are built, these methods are much more computationally expensive than other strategies but tend to be more precisely, depending on the problem [8].

2.4 Machine Learning Algorithms

The tasks that are analyzed in this work are treated as classification problems, so the following concepts of machine learning algorithms are treated as classifiers from this chapter to the rest of this work.

2.4.1 Logistic Regression

Logistic regression (LR) consists in finding the best-fit parameters to a nonlinear function called the sigmoid which is highly suitable for binary problems. It uses optimization methods such as gradient ascent or in a more efficient way, the stochastic gradient ascent. Some advantages of having stochastic gradient ascent as the optimization algorithm is that it can learn from new data through multiple batches and iterations [9]. Sigmoid function is presented in the next equation, its output is between 0 and 1 and it can be interpreted as a probablity value.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

2.4.2 Support Vector Machine

A Support Vector Machine (SVM) as a classifier, generates a binary decision (that is why they are called "machines"). They have good generalization error because they can generalize what they have learned. Support vector machines try to maximize margin by solving a quadratic optimization problem. They use Kernel methods that map data (which is usually nonlinear data) to a higher dimension where it can solve a linear problem. The radial-bias function is a popular kernel method that measures the distance between two vectors [9].

2.4.3 Naive Bayes

A Naive Bayes classifiers is a linear classifier that is known for being simple yet very efficient. The probabilistic model of naive Bayes classifiers is based on Bayes' theorem, and the adjective naive comes from the assumption that the features in a dataset are mutually independent. In practice, the independence assumption is often violated, but

naive Bayes classifiers still tend to perform very well under this unrealistic assumption. Especially for small sample sizes, naive Bayes classifiers can outperform the more powerful alternatives [10].

2.4.4 K Nearest Neighbors

K-Nearest Neighbors (KNN) as a supervised classification method can make predictions calculating the distance of the input to the k-nearest training data inputs and evaluating which class is the most common among them. There are many distance metrics that can be used, and for example, when working with text representations euclidean distance can be useful.

The main advantage of such a memory-based approach is that the classifier immediately adapts as we collect new training data. However, the downside is that the computational complexity for classifying new samples grows linearly with the number of samples in the training dataset in the worst-case scenario [11].

2.4.5 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a classification algorithm, but during training it learns the most discriminative axes between the classes, and these axes can then be used to define a hyperplane onto which to project the data. The benefit is that the projection will keep classes as far apart as possible, so LDA is also a good technique to reduce dimensionality before running another classification algorithm such as an SVM classifier [12].

2.4.6 Ensemble Algorithms

When working with multiple machine learning algorithms, it comes the idea of which one is better to use. This task usually is very expensive due to the multiple trainings that must be done. Other idea is to try to combine multiple classifiers in order to get better results. In this technique, the algorithms that are going to be combined are called weak learners. This kind of algorithms can be grouped in three types: Bagging, Boosting and Stacking.

Bagging

Bagging is a technique where the data is processed with multiple homogeneous weak learners (Multiple copies of the same classifier but with different hyperparameters) in parallel and combines their learning following a deterministic rule such as majority vote. One advanced bagging algorithm is Random Forest [12].

Boosting

Boosting process the data using homogeneous weak learners also, but they learns sequentially (a learner depends on the previous ones results). Every consecutive learner focuses on what

the previous ones could not learn. The result is the combination of a weighted sum of all learners. The algorithm this work focuses on is one of the the most popular ones, called AdaBoost.

Stacking

Stacking often considers heterogeneous weak learners (Different types of classifiers) that can learn in parallel from the data and the final prediction is produced using a meta-model with the output of all the weak learners. This work is focusing on building a stacking ensemble model, combining the results of multiple machine learning algorithms.

2.4.7 Random Forest

Random Forest works as an ensemble of Decision Tree models and uses a votation system to generate predictions. As a bagging model, it can analyze different feature subsets per weak learner and improve the result by this method. The Random Forest algorithm introduces extra randomness when growing trees; instead of searching for the very best feature when splitting a node, it searches for the best feature among a random subset of features. This results in a greater tree diversity, which (once again) trades a higher bias for a lower variance, generally yielding an overall better model [12].

2.4.8 AdaBoost

Adaptive Boosting (AdaBoost) is a boosting algorithm that trains its ensemble classifiers in sequence. The new predictor corrects its predecessor paying attention to the training instances that the predecessor underfitted [12]. By default, AdaBoost is an ensemble of Decision Tree classifiers, but in this work, Random Forest is selected as the base classifier.

2.4.9 Multi-Layer Perceptron

A Multiplayer Perceptron (MLP) is composed of one (passthrough) input layer, one or more layers of threshold logic units (TLUs), called hidden layers, and one final layer of TLUs called the output layer. TLU is a function that inputs an array of weighted quantities, sums them, and if this sum meets or surpasses some threshold, outputs a quantity. The layers close to the input layer are usually called the lower layers, and the ones close to the outputs are usually called the upper layers. Every layer except the output layer includes a bias neuron and is fully connected to the next layer. When an ANN contains a deep stack of hidden layers, it is called a deep neural network (DNN). The field of Deep Learning studies DNNs, and more generally models containing deep stacks of computations. However, many people talk about Deep Learning whenever neural networks are involved (even shallow ones) [12].

2.5 Explainable Artificial Intelligence (XAI)

When working with a huge data amount and a black box model to solve a problem or get a solution for a task, interpretation can be important. If humans cannot interpret how the model works, it could affect some important decisions excluding or including irrelevant information. For example, if a model predicts when a interviewer can be hired or not, it can consider irrelevant information, like gender, skills not considered to the position, age, etc. As it can be observed, ignoring how the problem is solved can present additional problems in real life. Explainable Artificial Intelligence (XAI) is a collection of methods which are capable of interpret predictions of any kind of machine learning model. This is why they are called model-agnostic methods [13].

2.5.1 LIME

LIME is short for Local Interpretable Model-Agnostic Explanations. Local refers to local fidelity, which means the explanation has to really reflect the behaviour of the classifier "around" the instance being predicted. This explanation is useless unless it is interpretable - that is, unless a human can make sense of it. LIME is able to explain any model without needing to 'peak' into it, so it is model-agnostic.

Some classifiers use representations that are not intuitive to users at all (e.g. word embeddings). LIME explains those classifiers in terms of interpretable representations (words), even if that is not the representation actually used by the classifier. In order to figure out what parts of the interpretable input are contributing to the prediction, LIME perturbs the input around its neighborhood and see how the model's predictions behave. Then it weights these perturbed data points by their proximity to the original example and learn an interpretable model on those and the associated predictions. For example, if a user tries to explain the prediction for the sentence "I hate this movie", LIME will perturb the sentence and get predictions on sentences such as "I hate movie", "I this movie", "I movie", "I hate", etc. Even if the original classifier takes many more words into account globally, it is reasonable to expect that around this example only the word "hate" will be relevant [14].

2.5.2 SHAP

SHAP values (an acronym from SHapley Additive exPlanations) are based on Shapley values, a concept coming from game theory. In this case, the features represent the players which need to contribute to an outcome (the prediction of the model). What SHAP does is quantifying the contribution that each feature brings to the prediction made by the model. To determine the importance of a single feature, SHAP values are based on the idea that the outcome of each possible combination of features should be considered. This corresponds to each possible combination of f features, where f represents a possible subset of features [15].

For example, a model says a bank shouldn't loan someone money, and the bank is legally required to explain the basis for each loan rejection, in this case, SHAP values

interpret the impact of having a certain value for a given feature in comparison to the prediction the model would make if that feature took some baseline value.

SHAP values do this in a way that guarantees a nice property. In general, SHAP decompose a prediction with the following equation:

$$sum(\text{SHAP values for all features}) = \text{prediction for team} - \text{prediction for baseline values} \tag{2.5}$$

Due to the complexity of calculating 2^n times a classifier in order to get an interpretation of the results, SHAP library contains functions to calculate approximations for the SHAP values for some types of machine learning algorithms: linear based algorithms and tree based algorithms. Algorithms such as K-Nearest Neighbors and Support Vector Machines do not have an efficient method to calculate the SHAP values, but LIME can evaluate them because it does not have to consider all features.

3 State of the Art

The study of fake news detection and fake news user spreaders detection has been a complex task through recent years. Many possible solutions with different characteristics can be found and most of them produces accurate predictions and results. This chapter presents some solution approaches found in the State of the Art. Moreover, this chapter describes the case study this work is focused on and the proposed solutions with which this work is compared.

3.1 Fake News and Fake News User Spreaders

Most user spreaders tend to hide their intentions and lie using an specific vocabulary and phrases. Although they have this in consideration, some patterns cannot be hidden, due to author habits or even the way that information is expressed [16]. They often catch the public by using sensationalism, or by using expressions that can captive user's attention. In [17], these kind of features are linguistic ones, and can be classified as follows:

Data Representation

Perhaps the simplest method of text representation is bag of words. Individual words or n-grams frequencies are aggregated and analyzed.

Deep Syntax

This analysis is implemented in this kind of works through Probability Context Free Grammars (PCFG). Sentences are transformed to a set of rewrite rules to describe syntax structure, for example noun and verb phrases.

Semantic Analysis

Although restricted to the domain of application, the intuition is that a deceptive writer with no experience with an event, person or object may include contradictions or omission of facts present in profiles on similar topics.

3.2 Text Preprocessing and Representation

Text processing usually consists of the following steps: remove stop words, spaces, punctuation and lemmatize/stemming the text [18], then select an appropriate representation that can be used in a machine learning algorithm. One of the most common text representations used for these tasks is TF-IDF. The reason can be observed in [19] where through several experiments using multiple datasets, they observed TF-IDF has stronger consistency and can give better and more efficient results in these tasks than a more advanced representation, such as word embeddings. Although we can find multiple works which use TF-IDF transformations with n-grams such as [20, 21, 22, 23], there are some works which have a different approach to these tasks. Some other representations found in recent works are the use of statistical features [19, 24], Part of speech Tagging (POS Tag) [25], sentimental detection approaches [26], metadata from social media and user accounts (when working with user profiling) [21], BERT [27], and even a combination of multiple features like in [28], where embeddings, emotions, and stylometry techniques are joined in a single text representation.

3.3 Machine Learning Solutions

Table 3.1 shows a list of machine learning and deep learning classifiers found in recent state of the art. In this work, 18 solution approaches were analyzed, being 12 machine learning approaches [18, 29, 30, 24, 20, 31, 32, 22, 33, 34, 35, 23, 36] and 6 deep learning approaches [37, 38, 39, 28, 40, 41]. It can be observed that SVM and LR are preferred by researchers due to its easiness to implement and they can obtain competitive results in these tasks.

Table 3.1: List of machine learning classifiers and their number of works that experimented with them

Classifier	Number of works
Support Vector Machine	6
Logistic Regression	5
Naive Bayes	3
Random Forest	1
XGBoost	2
LSTM	5
Bi-LSTM	2

3.4 Case study

This work focuses in corpora developed for two recent fake news detection events: first, PAN@CLEF 2020 event has a task for fake news user spreaders profiling, where all

participants worked with an English corpus and a Spanish corpus using NLP and machine learning techniques (see Section 4.1.1); and second, FakeDes@Iberlef 2021 is an event which fake news posted in multiple Spanish websites have to be detected (see Section 4.1.2). By the time this work is developed, FakeDes@Iberlef 2021 is in process, hence no participant solution approaches are described in this work yet.

3.4.1 Proposed Solutions at PAN@CLEF 2020

The winner was selected measuring the mean accuracy for both corpora. The conclusion of the event was a tie of an ensemble model and the use of a fine-tunned SVM solution. In [31], the ensemble model proposes the use of N-grams with the combination of LR, RF, SVM, and XGBoost. On the other hand, in [23], the single use of word n-grams with an SVM was enough to get the best results of the event. Some other approaches which worked with N-grams and TF-IDF used machine learning classifiers, mainly SVM and LR are described in [22, 35, 36]. Surprisingly, deep learning approaches did not reach the top 10 best solution proposals and it confirmed TF-IDF can solve this task in a better way as mentioned in [42]. Transformers were not used by any of the participants, but other kinds of deep learning architectures like LSTMs variations such as LSTM with Universal Sentence Encoder [41] and LSTM with attention [40] took part in the event. Table 3.2 shows the top 30 best solutions for the task at the event.

Table 3.2: Top 20 results obtained at PAN@CLEF 2020 Event expressed in the overview [1]. Some papers were not trackable so some classifiers are not described

Participant	**Classifier**	**EN Accuracy**	**ES Accuracy**	**Mean Accuracy**
bolonyai20 [31]	Ensemble	**0.750**	0.805	**0.7775**
pizarro20 [23]	SVM	0.735	**0.820**	**0.7775**
koloski20	LR	0.715	0.795	0.7550
deborjavalero20		0.730	0.780	0.7550
vogel20	SVM	0.725	0.785	0.7550
higueraporras20		0.725	0.775	0.7500
tarela20		0.725	0.775	0.7500
babaei20	MLP	0.725	0.765	0.7450
staykovski20		0.705	0.775	0.7400
hashemi20	RF	0.695	0.785	0.7400
estevecasademunt20		0.710	0.765	0.7375
castellanospellecer20		0.710	0.760	0.7350
shrestha20	SVM	0.710	0.755	0.7325
espinosagonzales20	SVM	0.690	0.760	0.7250
ikae20	Similarity	0.725	0.725	0.7250
morenosandoval20	RF	0.715	0.730	0.7225
majumder20	LSTM	0.640	0.800	0.7200

4 Solution Proposal Description

As discussed before, this work focuses on two main tasks to research: to detect fake news in social media such as websites and to detect fake news user spreaders. This tasks are pretending to be solved following the event's rules that consider working with content-based features only. Other features and metadata such as attached images, account profile, and so on, are out of the scope for the analysis and experiments related to this work. Since the text is the main source of information to solve these tasks, this work proposes a system that can detect fake news through word n-grams, TF-IDF frequencies and statistical features.

The following sections of this chapter describe the corpora used and the process of the solution proposal development.

4.1 Corpora Description

The corpora considered in this work was selected from two events, where each one is focusing in each task of this work respectively. The first one is the corpus generated at the PAN@CLEF 2020 event, which was created to detect fake news user spreaders in English and Spanish languages; and the second one is the corpus generated at the FakeDes@Iberlef 2021 event which was created to detect fake news in the Spanish language in news websites.

4.1.1 PAN@CLEF corpus

This corpus was created by experts and members of the PAN@CLEF 2020 event extracting Spanish and English posts from multiple user accounts that have shared or written texts related to fake news in Twitter respectively. These posts were analyzed using fact-checking websites such as Snopes and Politifact. Thus, they selected additional user accounts that have not shared content related to fake news as they could figure out about it. The last part was to select a set of posts for each user account (One hundred posts per user). Finally, two class balanced corpora (each language has its own corpus) were created and tagged according to the following rules:

1. Fake news user accounts spreaders are labeled as 1
2. Real news user accounts spreaders are labeled as 0
3. A fake news user account is defined as an account that spreads posts (or even a single one) that are related to fake news

Some additional characteristics of this corpus that are important to mention are described in table 4.1:

Table 4.1: Characteristics of PAN@CLEF 2020 Corpus

Characteristic	Description
Language(s)	English and Spanish
Number of user accounts	500 user accounts per language
Number of posts per user	100 messages per user account
Train and Test Splitting	60% training 40% testing

4.1.2 FakeDes@Iberlef 2021 Corpus

This corpus was created by organizers of the FakeDes@Iberlef 2021 event retrieving news articles from dedicated Spanish websites (either those that spread fake news and also true news). Thus, these texts have larger size than the previous corpora. Another difference is that this corpus is already split in train and development data. This corpus was also tagged using the labels "Fake" and "True" for each input. Some additional characteristics are described in Table 4.2.

Table 4.2: Characteristics of FakeDes@Iberlef 2020 Corpus

Characteristic	Description
Language(s)	Spanish
Number of news for training	676
Number of news for development	295
Number of news for testing	572

4.2 Model description

The solution approach is the analysis of N-gram and statistical frequencies for each corpus using machine learning algorithms in an ensemble model. Through preprocessing and feature selection methods, the solution attempts to solve both tasks described previously in the introduction of this chapter. The diagram 4.1 represents the general steps of the solution approach which are described in the following subsections.

Figure 4.1: Main steps of the proposal solution

4.2.1 Text Preprocessing

When working with PAN@CLEF corpus, texts are concatenated per user account. Also, all characters and symbols that are not related to the English or Spanish language are deleted. Then, multiple versions of the preprocessed data are created. The following lists describes the data versions created for each corpus:

PAN@CLEF Corpus

1. Version 1: All words are converted into lowercase. Twitter entities are tagged through tags as listed in table 4.3.
2. Version 2: Same as version 1, but punctuation is removed.
3. Version 3: Same as version 1, but numbers are tagged as "zznumberzz" and punctuation is removed.

FakeDes Corpus

1. Version 1: All words are converted into lowercase. Numbers' tag is changed to "zznumberzz".
2. Version 2: Same as version 1, but punctuation is removed.

Table 4.3: Characteristics of PAN@CLEF 2020 Corpus

Twitter Entity	Tag
User's mention	zzuserzz
URL	zzurlzz
Hashtag	zzhashzz
RT (Retweet)	zzrtzz

The process of tokenization is similar for all corpora. Word unigrams, bigrams and skipgrams are considered as features in this work. Stop words selected are those n-grams which appear in only one document and those which have the same number of appearances in both classes. Thus, if token A appears in 3 fake documents and 3 real documents, then it is considered as stop word. This is done in order to reduce the dimensionality of the data and to have more contextual interpretations. The skipgrams are built with size two, using one, two, three, and four gaps. These three n-gram types are examined individually, therefore three different models of each classifier are created.

Another preprocessing step is to extract statistical features from the texts and feed an individual machine learning algorithm using them. This contribution is evaluated at the same level of the other models in the final ensemble solution approach. The statistical features are described in Section 5.2. Figure 4.2 presents a brief summary about this part of the solution development.

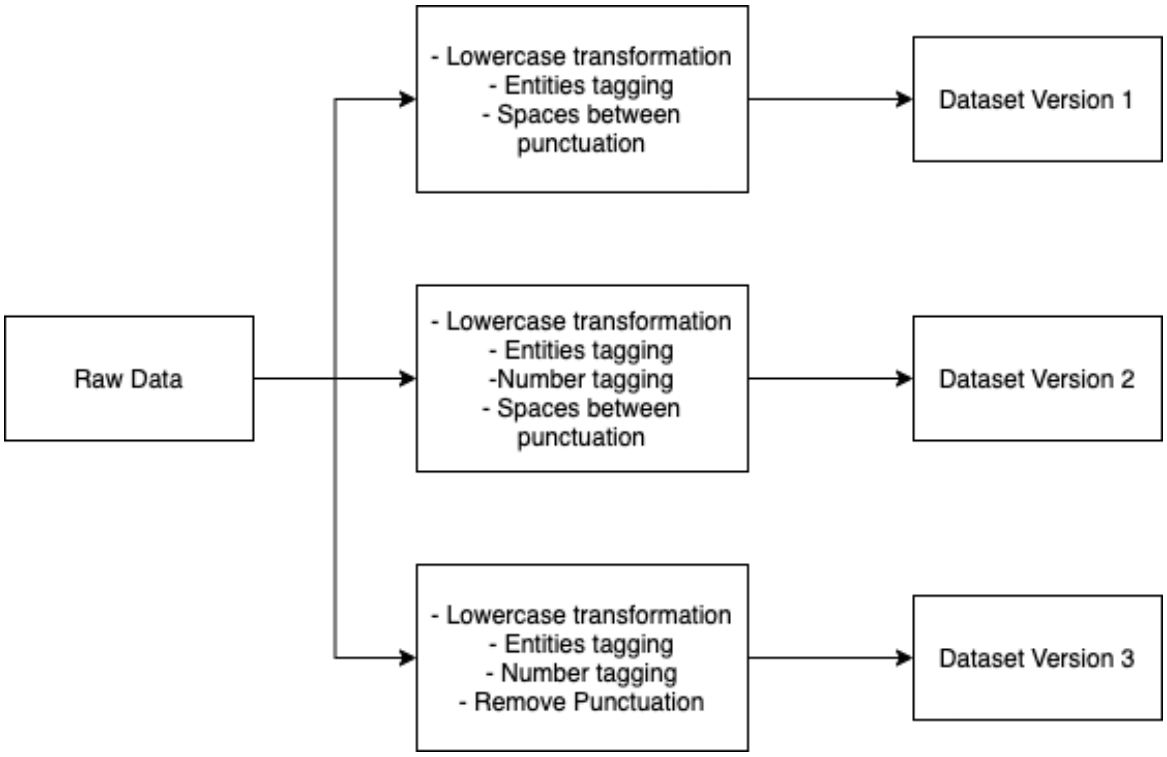

Figure 4.2: Representation of the Text Preprocessing step

4.2.2 Hyperparameter Selection

Each tokenized dataset version is transformed in a TF-IDF matrix, looking for the best n-gram frequencies that can achieve the highest accuracy during the training process for each classifier. In the case of PAN@CLEF corpora, no development data is provided, so each model is evaluated in a 3-repeated 5-fold cross-validation method. In the case of FakeDes corpus, the development data is provided. In this step, some hyperparameters of each machine learning model are evaluated and the model with the highest accuracy is selected. Figure 4.3 is a diagram that describes this process.

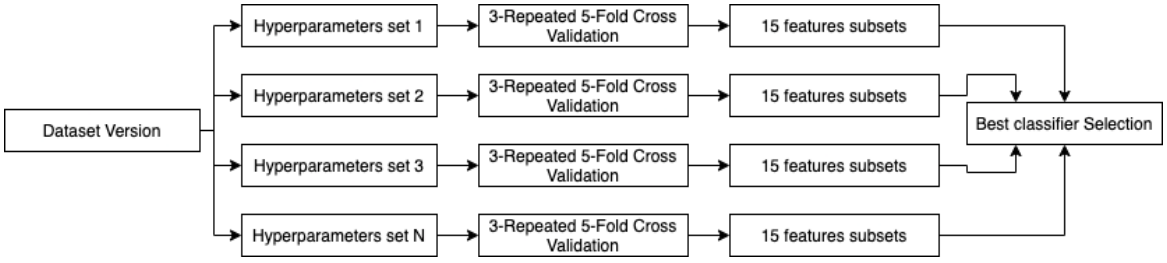

Figure 4.3: Representation of the Hyperparameter Selection Process

4.2.3 Feature Selection

When working with a huge amount of features or a high dimensionality feature dataset, it can be difficult to work and to interpret the obtained results. Feature selection techniques

provide a way to manage this problem and bring some other benefits, such as avoiding overfitting, decreasing the model training time, and improving the model's interpretation. The technique considered in this work is the use of SHAP and LIME, which not only are used to interpret model results but can also be a way to select the best features that contribute to the best solution. Figure 4.4 is a diagram representation of this process.

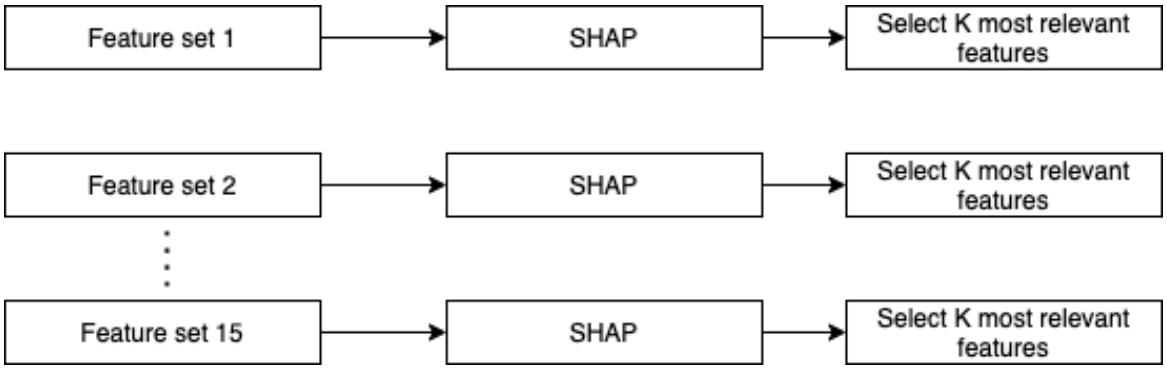

Figure 4.4: Representation of the Feature Selection Process

As mentioned before, the SHAP technique considers the contribution of each feature in the final prediction, that is, each feature has a different SHAP value in each prediction the model works out. The value can be either positive or negative, and its absolute value represents the degree of contribution the feature has in the model solution. In order to work with SHAP as a feature selector, all calculated absolute values in a prediction set are aggregated and they are sorted in decreasing order. Once this sorting is obtained, the selection of the features is represented as an optimal k number of features, where k is determined by training the model with different numbers of features and by examining the following aspects:

1. When the model achieves its maximum mean accuracy when predicting results for a dataset.

2. If there are multiple subsets when the model achieves its maximum accuracy, the subset with the minimum number of features is selected.

3. Some other alternatives of *optimal* number of features are selecting: (i) the maximum number of features when training data reaches its maximum accuracy result; (ii) the second subset with the minimum number of features, or (iii) a certain percentage amount of features.

4.2.4 Ensemble Algorithm

The main algorithm is an ensemble solution approach, using multiple classifiers that contribute to the final solution. The selected algorithms are: Logistic Regression (LR), Random Forest (RF), Support Vector Machines (SVM), AdaBoost (Ada), Multilayer Perceptron (MLP), Linear Discriminant Analysis (LDA), Naive Bayes (NB), K-Nearest Neighbors (KNN), and Logistic Regression for the statistical features dataset.

Once the feature selection is done for every machine learning model, the next step is to combine their predictions when testing the complete system. All machine learning models can retrieve a probabilistic value according to this binary classification problem, and in this way, their results can be submitted to a voting scheme. The two voting schemes that this project evaluates are soft voting (where the mean of all probabilities is calculated to get the final result) and hard voting (counting each binary value and the major is going to be the final result). It is a stacking ensemble approach which is conformed by a Bagging ensemble model (Random Forest) and a Boosting ensemble model (AdaBoost). The representation of the Random Forest classifier is described in Figure 4.5, the AdaBoost classifier is described in Figure 4.6 and the ensemble model in Figure 4.7. The ensemble model considers 8 different classifiers trained with different n-gram combinations: unigrams, bigrams, skipgrams, unigrams and bigrams, unigrams and skipgrams and unigrams, bigrams and skipgrams, having a total of 49 classifiers considering the Logistic Regression classifier with statistical features.

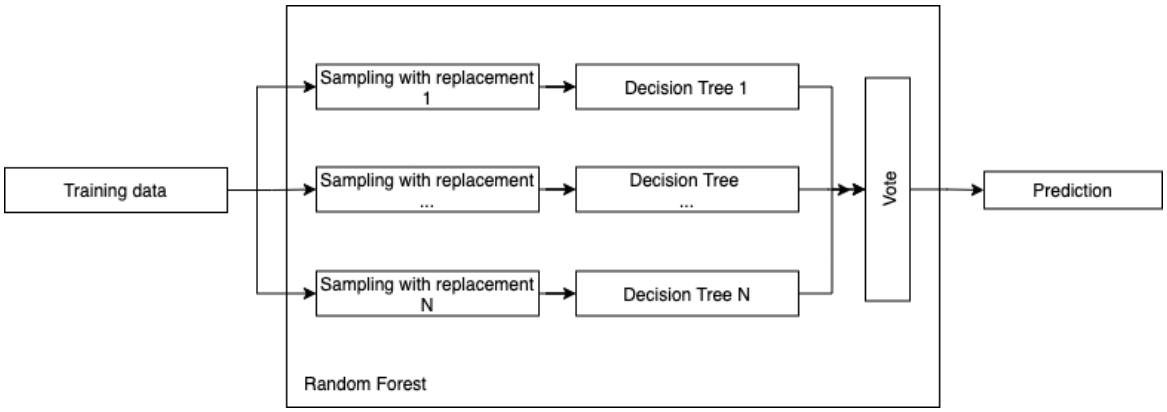

Figure 4.5: Representation of the Random Forest Classifier

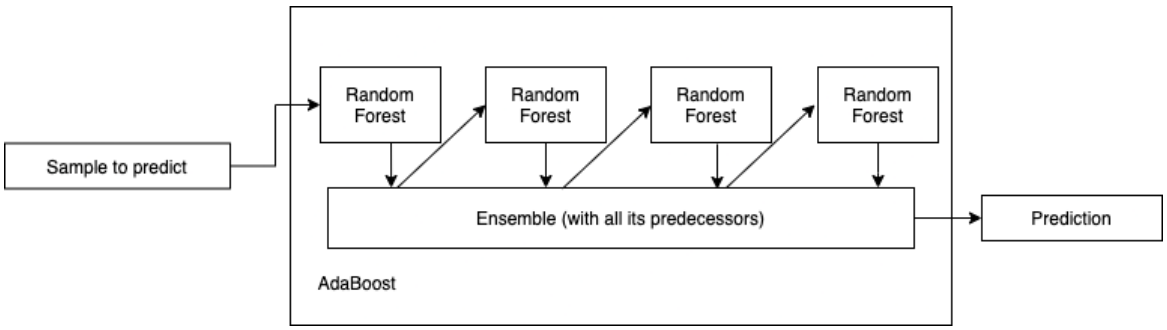

Figure 4.6: Representation of the AdaBoost Classifier

4.2.5 Solution Interpretation

The last part is to interpret the results based on how the model is working by their feature contributions. A top of the most relevant features per each model is examined and, in addition, it is possible to evaluate which models are the most relevant for the final results.

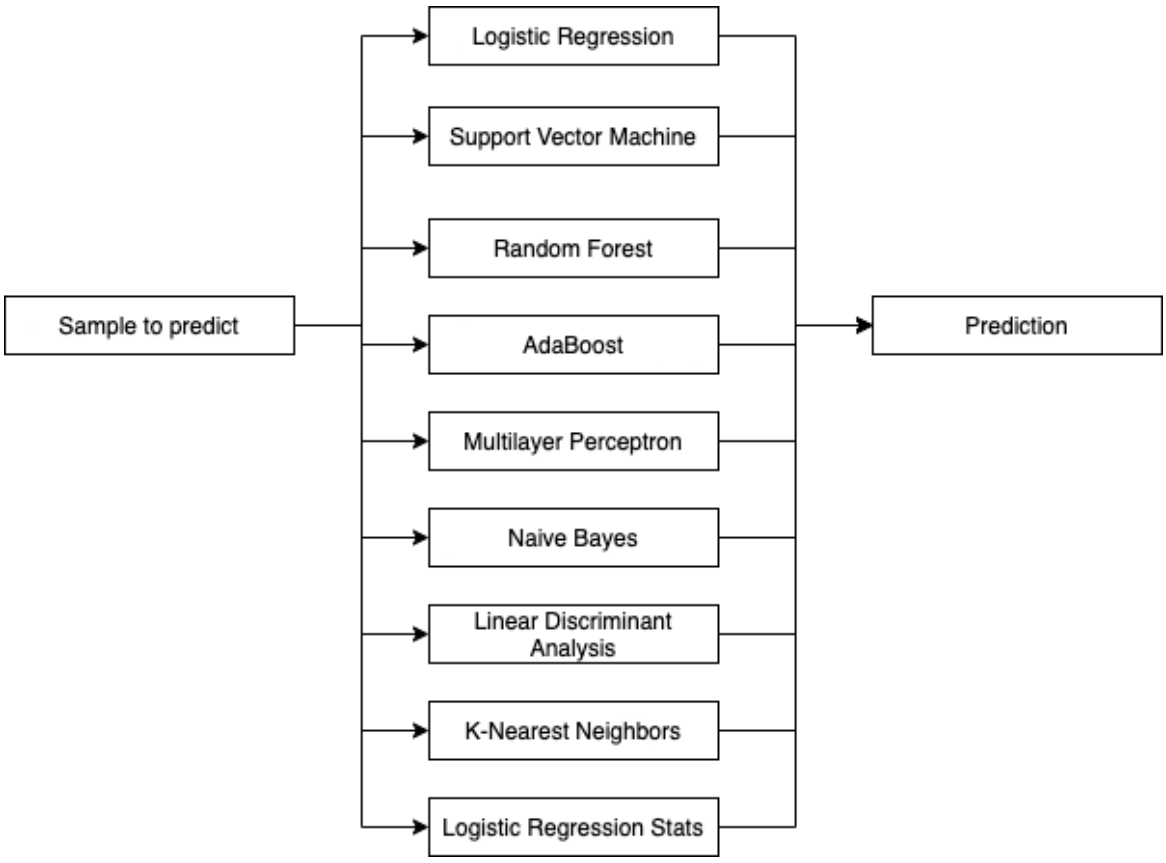

Figure 4.7: Representation of the Ensemble Solution

With this, it is possible to conclude what the model learned from the features and it is able to establish a set of rules the model detected aiming to explain the system to general users.

4.2.6 Evaluation Metrics

The evaluation metrics considered to analyze this work are the Accuracy and F1 Score, due to the fact that they are used in the previous events described in previous sections. By this means it is possible to compare results with the competitors of the events and make some conclusions about the performance of this approach.

5 Solution Development

This chapter presents the characteristics and results obtained during the process of building the final solution approaches for each corpus when applying the processes mentioned in the last chapter. All classifiers are evaluated individually, and a classifier per n-gram type is created. Finally, combined-features classifiers are created and evaluated.

5.1 Fake News Spreaders Profiling - English

5.1.1 Text Preprocessing

The solution for this task is developed using the English corpus from PAN@CLEF 2020 event. Table 5.1 presents an example for every dataset version created. This work considers as a hypotesis that having multiple dataset versions can improve the final results due to each classifier can focus on different features obtained from different sources.

Table 5.1: Corpus Processing for English PAN@CLEF 2020

Dataset Version	**Transformed text**
Original	BUSTED: Hillary Clinton Got $\$145,920,412$ Richer Thanks To Russia... #URL#
Version 1	busted : hillary clinton got $\$145,920,412$ richer thanks to russia . . . zzurlzz
Version 2	busted hillary clinton got 145 920 412 richer thanks to russia zzurlzz
Version 3	busted hillary clinton got zznumberzz zznumberzz zznumberzz richer thanks to russia zzurlzz

5.2 Statistical Features

In addition to the frequency features creation, a descriptive statistical features dataset is created. This statistical features are also based on the frequency of specific words and symbols. The following list enumerates all features used for this dataset:

1. Character count and standard deviation

2. Word count and standard deviation
3. Word count with size less than 5 characters
4. Punctuation count and standard deviation
5. Number count and standard deviation
6. Twitter entities count and standard deviation
7. Emoji count and standard deviation
8. Uppercase letters count and standard deviation
9. Lexical diversity (a ratio of how many different words the account uses)

5.2.1 Hyperparameter Selection

The hyperparameter selection is done with a 3-repeated 5-fold cross-validation method in order to get more accurate results when training every classifier. Customized stop words described in the previous chapter are excluded. Some algorithms have elevated computational cost when using all features for training, so some features are excluded based on their TF-IDF value using *min_ df* and *max_ df* parameters. Table 5.2 and Table 5.3 shows the best hyperparameters and dataset version found for each classifier.

Table 5.2: Best hyperparameters found for English PAN@CLEF classifiers, including dataset version and the *min_ df* and *max_ df* parameters for TF-IDF. AdaBoost Classifiers are ensembles of Random Forest with standard hyperparameters. - Part 1/2

Classifier	**N-gram Type**	**version**	**min**	**max**	**Classifier hyperparameters**
LR	unigrams	1	1	1.0	C=1000
LR	bigrams	3	1	1.0	C=1
LR	skipgrams	1	1	1.0	C=1000
RF	unigrams	1	1	1.0	n_estimators=200, max_features=1.0
RF	bigrams	1	1	1.0	n_estimators=200, max_features=0.5
RF	skipgrams	1	3	1.0	n_estimators=200, max_features=0.5
SVM	unigrams	3	1	1.0	C=100
SVM	bigrams	3	1	1.0	C=1
SVM	skipgrams	1	1	1.0	C=100
AdaBoost	unigrams	3	1	1.0	learning_rate=0.1
AdaBoost	bigrams	1	1	1.0	learning_rate=0.01
AdaBoost	skipgrams	3	3	1.0	learning_rate=0.01

Table 5.3: Best hyperparameters found for English PAN@CLEF classifiers, including dataset version and the *min_df* and *max_df* parameters for TF-IDF. Statistical LR is the classifier which works with statistical features only, so no N-gram is required. - Part 2/2

Classifier	N-gram Type	version	min	max	Classifier hyperparameters
MLP	unigrams	2	1	1.0	hidden_layer_sizes=(200), activation=logistic
MLP	bigrams	3	1	1.0	hidden_layer_sizes=(200), activation=logistic
MLP	skipgrams	3	1	1.0	hidden_layer_sizes=(200), activation=logistic
Naive Bayes	unigrams	2	1	1.0	alpha=1
Naive Bayes	bigrams	3	1	1.0	alpha=1
Naive Bayes	skipgrams	1	1	1.0	alpha=0
KNN	unigrams	3	1	1.0	p=2, n_neighbors=3
KNN	bigrams	2	1	1.0	p=2, n_neighbors=13
KNN	skipgrams	2	2	1.0	p=2, n_neighbors=9
LDA	unigrams	3	1	1.0	n_components=None
LDA	bigrams	3	1	1.0	n_components=None
LDA	skipgrams	1	2	1.0	n_components=None
Statistical LR	34 features				C=1

5.2.2 Feature Selection

The process of feature selection proposed is the following:

1. Evaluate the classifier for each k-fold using SHAP or LIME
2. Get the absolute contribution value per feature, adding the absolute value obtained for each input from validation data and sort them in decreasing order
3. Filter features with the lowest value of contribution. A value of zero means the feature is not relevant for the classifier and it can be removed. The standard threshold value used in this part of the work is 0.01
4. For each k-fold, select k sorted features that were remained from previous step and train a new classifier
5. Get the accuracy and repeat the previous step until all features were explored
6. Analyze the results and select the best subset of features. It is determined with the maximum accuracy found
7. Join all subsets of features found in order to get the best features set for the classifier

Some classifiers have elevated computational cost to evalute SHAP values. LIME is a less expensive method although is less accurate than SHAP, but can give similar results. Due to this reason SVM, NN, and KNN are evaluated using LIME. To see further details about the feature selection behaviour, see Appendix A. The step 4 of the previous process is complex since there is no optimal threshold that can be applied in order to get the best features. This work explores several heuristics which are described in the following list:

1. The minimum subset of features where the maximum accuracy is found (min)
2. The maximum subset of features where the maximum accuracy is found (max)
3. The second minimum subset of features where the maximum accuracy is found (sec)
4. The subset of features where the train dataset achieves its maximum accuracy score (acc)
5. Some percentage of the top features, which are from 10% to 60% of them

Figure 5.1 represents an example of accuracy behavior when selecting k most important features, sorted with SHAP or LIME method. In this case, the selection is applied per k-fold where previous rules are applied. For the first k-fold, the heuristics are applied when the model reaches the highest accuracy (0.7). Thus, *min* rule selects 2900 features, *max* and *sec* rule select 3480 features, and *acc* selects 1740 features.

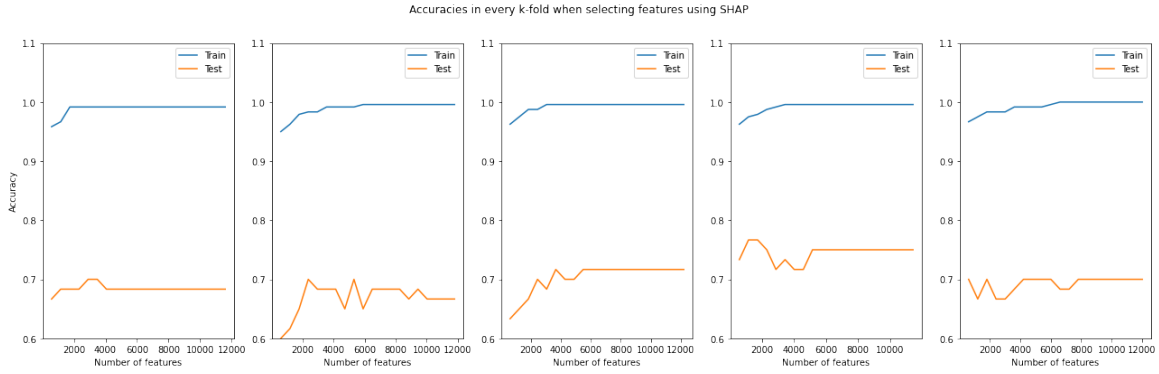

Figure 5.1: Accuracy per subset of relevant Features for Logistic Regression using bigrams

Table 5.4 shows the results obtained when training with different k number of relevant features. Each classifier selects its best feature set based on the mean accuracy value when training their different N-gram variations.

The best k value is found according to the highest mean accuracy value found per classifier. Table 5.5 and Table 5.6 describes the number of features selected per classifier according to their best feature set. Classifiers which have combinations of features are developed once their individual N-gram features are selected. This is considered as a joined combination of features.

Table 5.4: Mean Accuracy per k feature selection proposal

Feature Set	LR	SVM	RF	AdaBoost	MLP	LDA	NB	KNN
min	0.8661	0.9188	0.7133	0.6961	0.9366	**0.9088**	0.9116	0.7966
sec	0.8683	0.9019	0.7088	0.6938	0.9308	0.9044	0.9138	0.7969
max	0.8638	0.8827	0.7096	0.6931	0.9283	0.8972	0.9140	0.7962
acc	0.8730	0.9069	0.7158	0.7037	0.9402	0.8869	**0.9148**	0.8086
10%	**0.8745**	0.9202	**0.7162**	0.7098	0.9447	0.8848	0.8994	0.8140
20%	0.8743	0.9253	0.7148	0.7123	0.9474	0.8862	0.8948	0.8156
30%	0.8742	**0.9262**	0.7137	0.7126	0.9492	0.8877	0.8955	0.8173
40%	0.8738	0.9240	0.7129	**0.7139**	0.9500	0.8894	0.8968	0.8186
50%	0.8733	0.9193	0.7118	0.7127	**0.9504**	0.8889	0.8990	0.8193
60%	0.8724	0.9124	0.7099	0.7115	0.9503	0.8875	0.9010	**0.8197**

Table 5.5: Feature Selection for English PAN@CLEF corpus - Part 1/2

Classifier	N-gram Type	Features	Selected features	Percentage
LR	unigrams	30,479	1,630	5.35%
LR	bigrams	192,640	3,504	1.82%
LR	skipgrams	714,657	14,386	2.01%
LR	unigrams+bigrams	223,119	5,134	2.30%
LR	unigrams+skipgrams	745,136	16,016	2.15%
LR	unigrams+bigrams+skipgrams	937,776	19,520	2.08%
SVM	unigrams	29,915	5,541	18.52%
SVM	bigrams	192,640	11,671	6.06%
SVM	skipgrams	714,657	6,912	0.97%
SVM	unigrams+bigrams	222,555	17,212	7.73%
SVM	unigrams+skipgrams	744,572	12,453	1.67%
SVM	unigrams+bigrams+skipgrams	937,212	24,124	2.57%
RF	unigrams	30,479	1,999	6.56%
RF	bigrams	205,964	5,421	2.63%
RF	skipgrams	66,256	7,969	12.03%
RF	unigrams+bigrams	236,443	7,420	3.14%
RF	unigrams+skipgrams	96,735	9,968	10.30%
RF	unigrams+bigrams+skipgrams	302,699	15,389	5.08%
AdaBoost	unigrams	29,915	4,931	16.48%
AdaBoost	bigrams	205,964	12,616	6.13%
AdaBoost	skipgrams	53,106	14,416	27.15%
AdaBoost	unigrams+bigrams	235,879	17,547	7.44%
AdaBoost	unigrams+skipgrams	83,021	19,347	23.30%
AdaBoost	unigrams+bigrams+skipgrams	288,985	31,963	11.06%

Table 5.6: Feature Selection for English PAN@CLEF corpus - Part 2/2

Classifier	N-gram Type	Features	Selected features	Percentage
MLP	unigrams	30,447	8,880	29.17%
MLP	bigrams	192,640	18,295	9.50%
MLP	skipgrams	657,830	28,271	4.30%
MLP	unigrams+bigrams	223,087	27,175	12.18%
MLP	unigrams+skipgrams	688,277	37,151	5.40%
MLP	unigrams+bigrams+skipgrams	880,917	55,446	6.29%
LDA	unigrams	29,915	6,764	22.61%
LDA	bigrams	192,640	17,412	9.04%
LDA	skipgrams	137,437	73,266	53.31%
LDA	unigrams+bigrams	222,555	24,176	10.86%
LDA	unigrams+skipgrams	167,352	80,030	47.82%
LDA	unigrams+bigrams+skipgrams	359,992	97,442	27.07%
NB	unigrams	30,447	11,402	37.45%
NB	bigrams	192,640	22,387	11.62%
NB	skipgrams	714,657	28,165	3.94%
NB	unigrams+bigrams	223,087	33,789	15.15%
NB	unigrams+skipgrams	745,104	39,567	5.31%
NB	unigrams+bigrams+skipgrams	937,744	61,954	6.61%
KNN	unigrams	29,915	9,741	32.56%
KNN	bigrams	198,455	21,094	10.63%
KNN	skipgrams	116,154	79,736	68.65%
KNN	unigrams+bigrams	228,370	30,835	13.50%
KNN	unigrams+skipgrams	146,069	89,477	61.26%
KNN	unigrams+bigrams+skipgrams	344,524	110,571	32.09%

5.2.3 Trained Classifiers

Table 5.7 shows the results of the training step of each classifier. Since unigrams usually get the best results of the n-grams used, they are combined with the others in order to obtain additional classifiers that can be used for the final ensemble algorithm. The ensemble algorithm is created using all classifiers described in the table mentioned. According to this results, NB classifier with unigrams+skipgrams gets the best mean accuracy result of 0.9900. RF and AdaBoost get the worst mean accuracy results of 0.7500.

Table 5.7: Accuracy of Train dataset (cross-validation average) for English PAN@CLEF corpus obtained by selecting the best k features according to Table 5.4

Features	LR	SVM	RF	AdaBoost	MLP	LDA	NB	KNN
unigrams	0.8766	**0.9533**	0.7200	0.7000	**0.9766**	0.8600	0.7633	0.7533
bigrams	0.8000	0.9133	0.6766	0.6933	0.9666	**0.9666**	0.8200	0.8033
skipgrams	**0.9166**	0.8833	0.7033	0.7166	0.9033	0.8933	0.9566	0.8233
uni+bi	0.8933	0.9533	0.7200	0.7433	0.9766	0.9333	0.9866	0.8433
uni+ skip	0.8933	0.9466	**0.7500**	**0.7500**	0.9400	0.8933	**0.9900**	0.8533
uni + bi + skip	0.9033	0.9400	0.7366	0.7366	0.9566	0.9066	0.9866	**0.8600**

5.3 Fake News Spreaders Profiling - Spanish

Due to the solution approach for this corpus is very similar to the previous one, this section describes the results obtained from the development of this solution approach and the changes that are relevant to explain.

5.3.1 Text Preprocessing

Table 5.8 presents the dataset versions for this corpus. In this part, Spanish accents are considered, and the corpus considers the "n" character instead of the "ñ" character. For the statistical feature set, accent counting and its standard deviation calculation are included.

Table 5.8: Corpus Processing for Spanish PAN@CLEF corpus

Dataset Version	Transformed text
Original	Qué 20 amigos significan el mundo para ti? #URL#
Version 1	qué 20 amigos significan el mundo para ti ? zzurlzz
Version 2	qué 20 amigos significan el mundo para ti zzurlzz
Version 3	qué zznumberzz amigos significan el mundo para ti zzurlzz

5.3.2 Hyperparameter Selection

Table 5.9 and Table 5.10 shows the best hyperparameters found for each classifier and the dataset version.

Table 5.9: Best hyperparameters found for Spanish PAN@CLEF classifiers, including dataset version and the *min_df* and *max_df* parameters for TF-IDF. - Part 1/2

Classifier	N-gram Type	version	min	max	Classifier hyperparameters
LR	unigrams	3	1	1.0	C=1000
LR	bigrams	3	1	1.0	C=1000
LR	skipgrams	2	1	1.0	C=1000
RF	unigrams	1	1	1.0	n_estimators=200, max_features=1.0
RF	bigrams	1	1	1.0	n_estimators=200, max_features=0.5
RF	skipgrams	1	2	1.0	n_estimators=200, max_features=1.0
SVM	unigrams	2	1	1.0	C=1
SVM	bigrams	3	1	1.0	C=100
SVM	skipgrams	2	2	1.0	C=100
AdaBoost	unigrams	2	1	1.0	learning_rate=1
AdaBoost	bigrams	3	1	1.0	learning_rate=1
AdaBoost	skipgrams	1	3	1.0	learning_rate=1

Table 5.10: Best hyperparameters found for Spanish PAN@CLEF classifiers, including dataset version and the *min_df* and *max_df* parameters for TF-IDF. Statistical LR is the classifier which works with statistical features only, so no N-gram is required. - Part 2/2

Classifier	N-gram Type	version	min	max	Classifier hyperparameters
MLP	unigrams	2	1	1.0	hidden_layer_sizes=(200,), activation=logistic
MLP	bigrams	3	1	1.0	hidden_layer_sizes=(200, 100), activation=logistic
MLP	skipgrams	3	1	1.0	hidden_layer_sizes=(200,), activation=logistic
Naive Bayes	unigrams	3	1	1.0	alpha=0.1
Naive Bayes	bigrams	1	1	1.0	alpha=0.1
Naive Bayes	skipgrams	1	1	1.0	alpha=0.1
KNN	unigrams	2	1	1.0	p=2, n_neighbors=13
KNN	bigrams	1	3	1.0	p=2, n_neighbors=5
KNN	skipgrams	2	2	1.0	p=2, n_neighbors=9
LDA	unigrams	3	1	1.0	n_components=None
LDA	bigrams	3	2	1.0	n_components=None
LDA	skipgrams	2	2	1.0	n_components=None
Statistical LR	36 features				C=1

5.3.3 Feature Selection

Table 5.11 shows the train accuracy results when k features are selected using SHAP or LIME as a feature selection method. The operation for selecting k value is the same as the English corpus section.

Table 5.11: Mean Accuracy per feature selection proposal

Feature Set	LR	SVM	RF	AdaBoost	MLP	LDA	NB	KNN
min	0.9100	0.9116	0.7650	0.7622	0.9672	0.8072	0.9155	0.8583
sec	0.9119	0.9058	0.7650	0.7641	0.9641	0.8130	0.9216	0.8591
max	0.9100	0.9033	0.7653	0.7655	0.9640	**0.8153**	0.9288	0.8587
acc	0.9187	0.9256	0.7693	**0.7663**	0.9708	0.8019	**0.9316**	0.8656
10%	0.9218	0.9356	0.7708	0.7648	**0.9725**	0.7923	0.9041	0.8720
20%	0.9233	**0.9375**	0.7712	0.7633	0.9723	0.7879	0.8944	0.8762
30%	**0.9241**	0.9361	**0.7717**	0.7616	0.9714	0.7868	0.8936	**0.8770**
40%	0.9235	0.9342	0.7709	0.7602	0.9704	0.7865	0.8960	0.8769
50%	0.9228	0.9322	0.7708	0.7596	0.9700	0.7867	0.8990	0.8762
60%	0.9220	0.9303	0.7700	0.7580	0.9697	0.7885	0.9022	0.8756

Table 5.12 and Table 5.13 describes the number of features selected per classifier. Classifiers with joined features are developed once their individual N-gram features are selected.

Table 5.12: Feature Selection for Spanish PAN@CLEF corpus - Part 1/2

Classifier	N-gram Type	Features	Selected features	Percentage
LR	unigrams	41,593	5,478	13.17%
LR	bigrams	214,690	14,772	6.88%
LR	skipgrams	781,801	47,809	6.12%
LR	unigrams+bigrams	256,283	20,250	7.90%
LR	unigrams+skipgrams	823,394	53,287	6.47%
LR	unigrams+bigrams+skipgrams	1,038,084	68,059	6.56%
SVM	unigrams	42,317	4,574	10.81%
SVM	bigrams	214,690	9,393	4.38%
SVM	skipgrams	133,340	11,538	8.65%
SVM	unigrams+bigrams	257,007	13,967	5.43%
SVM	unigrams+skipgrams	175,657	16,112	9.17%
SVM	unigrams+bigrams+skipgrams	390,347	25,505	6.53%
RF	unigrams	42,350	6,081	14.36%
RF	bigrams	231,833	14,252	6.15%
RF	skipgrams	152,619	17,809	11.67%

Table 5.13: Feature Selection for Spanish PAN@CLEF corpus - Part 2/2

Classifier	N-gram Type	Features	Selected features	Percentage
RF	unigrams+bigrams	274,183	20,333	7.42%
RF	unigrams+skipgrams	194,969	23,890	12.25%
RF	unigrams+bigrams+skipgrams	426,802	38,142	8.94%
AdaBoost	unigrams	42,317	2,917	6.89%
AdaBoost	bigrams	214,690	4,808	2.24%
AdaBoost	skipgrams	77,238	173	0.22%
AdaBoost	unigrams+bigrams	257,007	7,725	3.01%
AdaBoost	unigrams+skipgrams	119,555	3,090	2.58%
AdaBoost	unigrams+bigrams+skipgrams	334,245	7,898	2.36%
MLP	unigrams	42,317	2,773	6.55%
MLP	bigrams	214,690	5,389	2.51%
MLP	skipgrams	757,327	6,907	0.91%
MLP	unigrams+bigrams	257,007	8,162	3.18%
MLP	unigrams+skipgrams	799,644	9,680	1.21%
MLP	unigrams+bigrams+skipgrams	1,014,334	15,069	1.49%
LDA	unigrams	41,593	11,373	27.34%
LDA	bigrams	45,321	16,074	35.47%
LDA	skipgrams	133,340	59,934	44.95%
LDA	unigrams+bigrams	86,914	27,447	31.58%
LDA	unigrams+skipgrams	174,933	71,307	40.76%
LDA	unigrams+bigrams+skipgrams	220,254	87,381	39.67%
NB	unigrams	41,593	12,886	30.98%
NB	bigrams	231,833	34,266	14.78%
NB	skipgrams	828,484	109,578	13.23%
NB	unigrams+bigrams	273,426	47,152	17.24%
NB	unigrams+skipgrams	870,077	122,464	14.08%
NB	unigrams+bigrams+skipgrams	1,101,910	156,730	14.22%
KNN	unigrams	42,317	7,830	18.50%
KNN	bigrams	27,073	9,920	36.64%
KNN	skipgrams	131,632	96,477	73.29%
KNN	unigrams+bigrams	69,390	17,750	25.58%
KNN	unigrams+skipgrams	173,949	104,307	59.96%
KNN	unigrams+bigrams+skipgrams	201,022	114,227	56.82%

5.3.4 Trained Classifiers

Table 5.14 shows the results the accuracy obtained by feature selection for every classifier. As the previous section, additional classifiers are trained using combinations of n-gram features. As it is seen, MLP gets the best accuracy results among all others.

Table 5.14: Mean Train Accuracy per Classifier for Spanish PAN@CLEF corpus

Features	LR	SVM	RF	AdaBoost	MLP	LDA	NB	KNN
unigrams	0.9200	**0.9800**	0.7666	0.7500	**1.0000**	0.8000	0.8733	0.8900
bigrams	0.9300	0.9333	0.7700	0.7633	0.9800	**0.8566**	0.9000	0.8166
skipgrams	0.9266	0.8766	0.7600	**0.7933**	0.9033	0.8166	0.8666	**0.9200**
uni + bi	0.9300	0.9700	0.7733	0.7600	0.9966	0.8366	**1.0000**	0.8666
uni + skip	0.9300	0.9633	0.7866	0.7766	1.0000	0.8033	1.0000	0.9133
uni+bi+skip	**0.9366**	0.9566	**0.7933**	0.7700	0.9966	0.8066	1.0000	0.8866

5.4 Fake News Detection - Spanish

The solution approach for this corpus is very similar to the PAN@CLEF Spanish corpus, only a few details are changed. This section describes the results obtained from the development of this solution approach and the changes that are relevant to explain.

5.4.1 Text Preprocessing

Table 5.15 presents the differences of some posts according to the version used for this corpus. This corpus contains the headline and the body of the news articles and are concatenated during this step. There are only two dataset versions since numbers are already tagged.

Table 5.15: Dataset versions for FakeDes@Iberlef 2021

Dataset Version	Transformed text
Original	el Chocolate Abuelita, el cual existe desde *NUMBER*, teniendo como imagen a dona Sara García desde *NUMBER*
Version 1	el chocolate abuelita , el cual existe desde zznumberzz , teniendo como imagen a dona sara garcía desde zznumberzz
Version 2	el chocolate abuelita el cual existe desde zznumberzz teniendo como imagen a dona sara garcía desde zznumberzz

5.4.2 Hyperparameter Selection

Table 5.16 and Table 5.17 shows the best hyperparameters found for each classifier and the dataset version. This selection is done with a 3-repeated 5-fold cross-validation method in order to get more accurate results when training every classifier. Some algorithms have elevated computation cost when using all features for training, so TF-IDF restriccions such as min_df and max_df were required. Since this corpus is not related with Twitter, some statistical features are removed. The following list describes the 11 features considered for the statistical classifier:

1. Number of words
2. Number of characters
3. Number of punctuation characters
4. Number of "zznumberzz"
5. Number of dot characters
6. Number of comma characters
7. Number of quotes
8. Number of line spaces
9. Number of accents
10. Number of uppercase letters
11. Lexical Diversity

Table 5.16: Best hyperparameters found for FakeDes@Iberlef 2021 classifiers, including dataset version and the *min_df* and *max_df* parameters for TF-IDF. - Part 1/2

Classifier	**N-gram Type**	**version**	**min**	**max**	**Classifier hyperparameters**
LR	unigrams	1	1	1.0	C=100
LR	bigrams	1	1	1.0	C=1000
LR	skipgrams	1	1	1.0	C=1000
RF	unigrams	2	1	1.0	n_estimators=200, max_features=1.0
RF	bigrams	1	1	1.0	n_estimators=200, max_features=1.0
RF	skipgrams	1	2	1.0	n_estimators=200, max_features=0.5
SVM	unigrams	1	1	1.0	C=100
SVM	bigrams	1	1	1.0	C=1
SVM	skipgrams	3	3	1.0	C=1
AdaBoost	unigrams	1	1	1.0	learning_rate=0.1
AdaBoost	bigrams	1	1	1.0	learning_rate=0.01
AdaBoost	skipgrams	1	3	1.0	learning_rate=0.01

Table 5.17: Best hyperparameters found for FakeDes@Iberlef 2021 classifiers, including dataset version and the *min_ df* and *max_ df* parameters for TF-IDF. Statistical LR is the classifier which works with statistical features only, so no N-gram is required. - Part 2/2

Classifier	N-gram Type	version	min	max	Classifier hyperparameters
MLP	unigrams	1	1	1.0	hidden_layer_sizes=(200), activation=logistic
MLP	bigrams	2	1	1.0	hidden_layer_sizes=(200, 100), activation=logistic
MLP	skipgrams	2	1	1.0	hidden_layer_sizes=(200,100), activation=logistic
Naive Bayes	unigrams	1	1	1.0	alpha=0.1
Naive Bayes	bigrams	1	1	1.0	alpha=0.1
Naive Bayes	skipgrams	2	1	1.0	alpha=0.1
KNN	unigrams	1	1	1.0	p=2, n_neighbors=13
KNN	bigrams	1	1	1.0	p=2, n_neighbors=11
KNN	skipgrams	2	2	1.0	p=2, n_neighbors=11
LDA	unigrams	2	1	1.0	n_components=None
LDA	bigrams	1	1	1.0	n_components=None
LDA	skipgrams	1	1	1.0	n_components=None
Statistical LR	11 features				C=1

5.4.3 Feature Selection

Table 5.18 shows the difference of features size when using SHAP as a feature selection method. The process of selection is the following:

1. Evaluate the classifier for each k-fold using SHAP
2. Get the absolute SHAP value per feature, adding the absolute value obtained for each input from validation data and sort them in decreasing order
3. Filter features with the lowest value of contribution. A value of zero means the feature is not relevant for the classifier and it can be removed. The standard threshold value used in this part of the work is 0.01
4. For each k-fold, select k sorted features that were remained from previous step and train a new classifier
5. Get the accuracy and repeat the previous step until all features were explored
6. Analyze the results and select the best subset of features. It is determined with the maximum accuracy found
7. Join all subsets of features found in order to get the best features for the classifier

The step 4 of the previous process is complex since there is no optimal threshold that can be applied in order to get the best features. This work explores several heuristics which are described in the following list:

1. The minimum subset of features where the maximum accuracy is found
2. The maximum subset of features where the maximum accuracy is found
3. The second minimum subset of features where the maximum accuracy is found
4. The subset of features where the train dataset achieves its maximum accuracy score
5. Some percentage of features, which are from 10% to 60% of the features

Table 5.18: Mean Accuracy per feature selection proposal

Feature Set	LR	RF	SVM	AdaBoost	MLP	LDA	NB	KNN
min	**0.8158**	**0.8129**	0.7468	0.7740	**0.8067**	**0.7604**	**0.7898**	0.7282
sec	0.8149	0.8127	0.7468	0.7740	0.8067	0.7604	0.7898	0.7282
max	0.8141	0.8126	0.7468	0.7740	0.8067	0.7604	0.7898	**0.7282**
acc	0.8100	0.8053	0.7524	**0.7759**	0.7985	0.7507	0.7865	0.7100
10%	0.8120	0.8025	0.7544	0.7753	0.7949	0.7481	0.7838	0.6892
20%	0.8122	0.8041	0.7548	0.7754	0.7938	0.7464	0.7807	0.6641
30%	0.8129	0.8043	0.7555	0.7732	0.7928	0.7468	0.7782	0.6448
40%	0.8139	0.8044	**0.7557**	0.7730	0.7925	0.7474	0.7773	0.6455
50%	0.8142	0.8052	0.7546	0.7731	0.7925	0.7488	0.7762	0.6465
60%	0.8141	0.8062	0.7524	0.7725	0.7928	0.7498	0.7754	0.6487

5.4.4 Trained Classifiers

Table 5.21 shows the results of the training step of each classifier. Since unigrams get the best results of the n-grams used, they are combined with the others in order to obtain a new classifier that can be used for the final ensemble algorithm. The ensemble algorithm is created using the predicted results of each classifier.

Table 5.19: Feature Selection for FakeDes@Iberlef 2021 corpus - Part 1/2

Classifier	N-gram Type	Features	Selected features	Percentage
LR	unigrams	24,408	4,590	18.81%
LR	bigrams	126,829	8,704	6.86%
LR	skipgrams	537,084	21,528	4.01%
LR	unigrams+bigrams	151,237	13,294	8.79%
LR	unigrams+skipgrams	561,492	26,118	4.65%
LR	unigrams+bigrams+skipgrams	688,321	34,822	5.06%
SVM	unigrams	24,408	5,544	22.71%
SVM	bigrams	126,829	14,420	11.37%
SVM	skipgrams	44,120	22,600	51.22%
SVM	unigrams+bigrams	151,237	19,964	13.20%
SVM	unigrams+skipgrams	68,528	28,144	41.07%
SVM	unigrams+bigrams+skipgrams	195,357	42,564	21.79%
RF	unigrams	24,384	31	0.13%
RF	bigrams	126,829	2,702	2.13%
RF	skipgrams	88,557	2,619	2.96%
RF	unigrams+bigrams	151,213	2,733	1.81%
RF	unigrams+skipgrams	112,941	2,650	2.35%
RF	unigrams+bigrams+skipgrams	239,770	5,352	2.23%
AdaBoost	unigrams	24,408	308	1.26%
AdaBoost	bigrams	126,829	1,500	1.18%
AdaBoost	skipgrams	44,120	2,260	5.12%
AdaBoost	unigrams+bigrams	151,237	1,808	1.20%
AdaBoost	unigrams+skipgrams	68,528	2,568	3.75%
AdaBoost	unigrams+bigrams+skipgrams	195,357	4,068	2.08%
MLP	unigrams	24,408	6,160	25.24%
MLP	bigrams	126,902	12,600	9.93%
MLP	skipgrams	533,510	22,500	4.22%
MLP	unigrams+bigrams	151,310	18,760	12.40%
MLP	unigrams+skipgrams	557,918	28,660	5.14%
MLP	unigrams+bigrams+skipgrams	684,820	41,260	6.02%

Table 5.20: Feature Selection for FakeDes@Iberlef 2021 corpus - Part 2/2

Classifier	N-gram Type	Features	Selected features	Percentage
LDA	unigrams	24,384	3,289	13.49%
LDA	bigrams	126,829	5,120	4.04%
LDA	skipgrams	537,084	4,968	0.92%
LDA	unigrams+bigrams	151,213	8,409	5.56%
LDA	unigrams+skipgrams	561,468	8,257	1.47%
LDA	unigrams+bigrams+skipgrams	688,297	13,377	1.94%
NB	unigrams	24,408	4,845	19.85%
NB	bigrams	126,829	8,096	6.38%
NB	skipgrams	533,510	14,272	2.68%
NB	unigrams+bigrams	151,237	12,941	8.56%
NB	unigrams+skipgrams	557,918	19,117	3.43%
NB	unigrams+bigrams+skipgrams	684,747	27,213	3.97%
KNN	unigrams	24,408	6,160	25.24%
KNN	bigrams	126,829	4,326	3.41%
KNN	skipgrams	79,262	42,168	53.20%
KNN	unigrams+bigrams	151,237	10,486	6.93%
KNN	unigrams+skipgrams	103,670	6,166	5.95%
KNN	unigrams+bigrams+skipgrams	230,499	10,492	4.55%

Table 5.21: Accuracy of Train dataset for FakeDes@Iberlef 2021 corpus

Features	LR	SVM	RF	AdaBoost	MLP	LDA	NB	KNN
unigrams	0.8067	0.7966	0.7491	0.7728	0.8000	0.7593	0.7762	0.7762
bigrams	0.8203	**0.8271**	0.7186	0.7898	0.8135	0.7288	0.8033	0.7559
skipgrams	0.8033	0.8033	0.7491	0.7457	0.7932	0.7661	0.7627	0.4813
uni + bi	0.8135	0.8135	0.7627	0.7966	**0.8237**	0.7457	**0.8135**	0.7898
uni + skip	0.8237	0.8169	**0.7864**	0.7796	0.8033	**0.7864**	0.7796	0.7762
uni+bi+skip	**0.8271**	0.8203	0.7796	**0.8067**	0.8067	0.7762	0.8033	**0.7898**

6 Results and Interpretations

6.1 Fake News Spreaders Profiling - English

6.1.1 Results

Table 6.1 and Table 6.2 show the accuracy obtained for all individual classifiers. SVM classifier with unigrams and bigrams and LR with unigrams get the best results, with an accuracy of 0.745. Table 6.3 presents the results of the final ensemble model when working with a LR meta classifier, and the use of majority voting, including soft and hard voting. When working as a majority hard voting classifier, the model reaches its best accuracy result of 0.730, which is lower than the best individual classifier result.

Table 6.1: Accuracy of Test dataset for English PAN@CLEF classifiers - Part 1/2

Classifier	N-gram Type	Feature Selection	Accuracy
LR	unigram	SHAP	**0.745**
LR	uni-bigram	SHAP	**0.745**
SVM	uni-bigram	LIME	**0.745**
SVM	uni-skipgram	LIME	0.740
LR	uni-bi-skipgram	SHAP	0.735
SVM	uni-bi-skipgram	LIME	0.735
AdaBoost	uni-bi-skipgram	SHAP	0.735
RF	uni-skipgram	SHAP	0.730
RF	uni-bigram	SHAP	0.725
LR	uni-skipgram	SHAP	0.720
SVM	bigram	LIME	0.720
RF	bigram	SHAP	0.715
RF	uni-bi-skipgram	SHAP	0.715
MLP	unigram	LIME	0.715
MLP	uni-bigram	LIME	0.715
MLP	uni-bi-skipgram	LIME	0.715
SVM	unigram	LIME	0.710
AdaBoost	unigram	SHAP	0.710

Table 6.2: Accuracy of Test dataset for English PAN@CLEF classifiers - Part 2/2

Classifier	N-gram Type	Feature Selection	Accuracy
MLP	uni-skipgram	LIME	0.710
LDA	unigram	SHAP	0.710
LDA	uni-bi-skipgram	SHAP	0.710
Naive Bayes	bigram	SHAP	0.710
LR	skipgram	SHAP	0.705
RF	unigram	SHAP	0.705
LDA	uni-skipgram	SHAP	0.705
LR	bigram	SHAP	0.700
RF	skipgram	SHAP	0.700
SVM	skipgram	LIME	0.700
LDA	skipgram	SHAP	0.700
AdaBoost	uni-skipgram	SHAP	0.695
MLP	bigram	LIME	0.695
LDA	bigram	SHAP	0.695
LDA	uni-bigram	SHAP	0.695
Naive Bayes	uni-bigram	SHAP	0.695
Naive Bayes	unigram	SHAP	0.690
AdaBoost	skipgram	SHAP	0.685
MLP	skipgram	LIME	0.685
AdaBoost	bigram	SHAP	0.680
AdaBoost	uni-bigram	SHAP	0.675
Naive Bayes	uni-bi-skipgram	SHAP	0.665
KNN	uni-skipgram	LIME	0.660
KNN	skipgram	LIME	0.655
KNN	uni-bigram	LIME	0.650
LR	stats	SHAP	0.645
KNN	bigram	LIME	0.635
KNN	uni-bi-skipgram	LIME	0.630
KNN	unigram	LIME	0.625
Naive Bayes	uni-skipgram	SHAP	0.590
Naive Bayes	skipgram	SHAP	0.550

Table 6.3: Metric Results for Test dataset - English PAN@CLEF classifiers

Classifier	Precision	Recall	Accuracy	F1-Score
Ensemble Model with LR	0.6667	0.7000	0.6750	0.6829
Ensemble Model using Soft Voting	0.7157	0.7300	0.7200	0.7228
Ensemble Model using Hard Voting	**0.7170**	**0.7600**	**0.7300**	**0.7379**

6.1.2 Model Interpretation

The Hard Voting model gets the best results of the ensemble model approaches. It gives equal importance for every classifier since it works as a majority voting classifier. The generalization resulted better because it considers the best classifiers for the final prediction. In addition, with a deeper analysis of the behavior of each individual classifier, the theoretical maximum accuracy that the best ensemble approach can get is 0.975. Two hundred predictions are evaluated and 5 of them cannot be evaluated correctly with none of the classifiers described in Section 6.1.1.

The Ensemble Model with LR gets the worst results of the ensemble model approaches. Figure 6.1 is the interpretation of which models have better importance to get its final predictions. Comparing these classifiers with Table 5.7, the conclusion is that this ensemble classifier is considering the classifiers with the best train accuracy results and apparently, they are not generalizing the problem sufficiently enough to get a better score. The best individual classifiers which generalized better the problem were not considered in the top 15 of the most important classifiers, this could be the main reason its final results are not as good as they should be. The red tone represents a greater value for the feature, which in this case means a high probability.

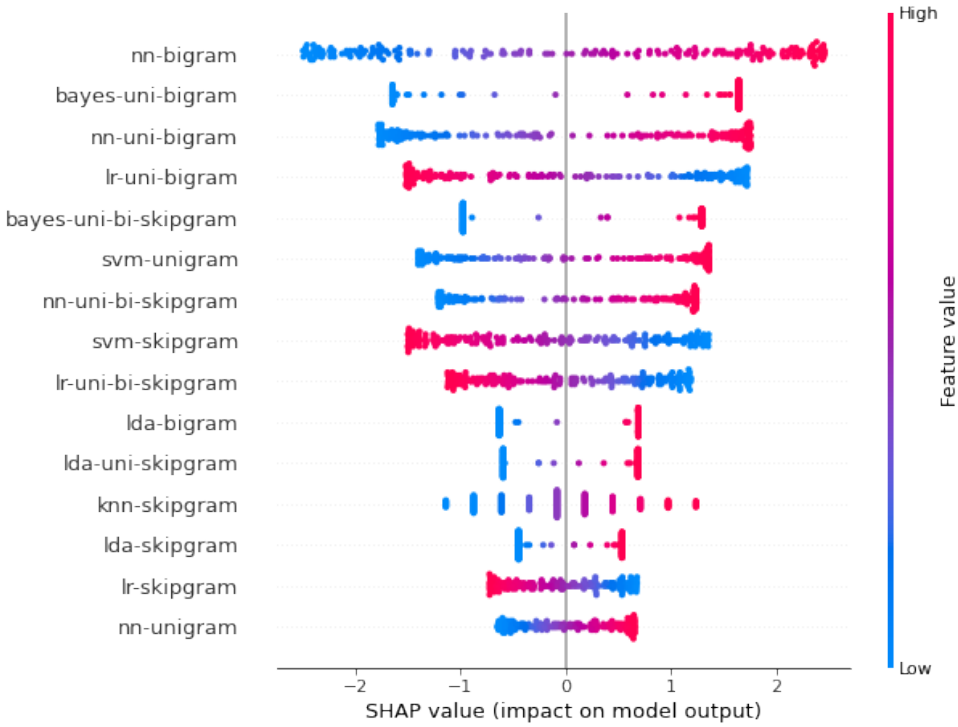

Figure 6.1: Top 15 of most important classifiers for ensemble model

The top 15 most important features used by the top 5 best classifiers are joined and described in Table 6.4. The interpretation for these features is: the more they increase their TF-IDF value, the more prediction is affected, according to the column they belong to. Top relevant features for all classifiers can be analyzed in Appendix B.

Table 6.4: Most important features used by English PAN@CLEF classifiers

Real news	Fake news
zzuserzz	trump
zzhashzz zzurlzz	obama
zzhashzz	trump s
zzrtzz	donald
zzuserzz zzuserzz	video
zzrtzz zzuserzz	meghan
zzurlzz zzurlzz	watch
zzemojizz	she
zzuserzz zzurlzz	report
2020	birthday
review	clinton
tv	hillary
-	democrats
series	
season	
ps zznumberzz	
everyone	
ps	
via	
how to	

Figures from 6.2 to 6.5 represent the interpretation of the predictions that none of the classifiers can predict correctly. They are based on the best classifier developed.

Figure 6.2 shows the user is not a fake news spreader. According to Table 6.4, the use of Twitter entities like user's mentions, URLs, etc. have an impact towards real news, but apparently, this input is an exception to the learned rule and that is why it cannot be predicted correctly. The same happens with Figure 6.3, Figure 6.4 and Figure 6.6. Figure 6.5 is the opposite, due to the absence of Twitter entities and the presence of features such as "hillary" and "trump", the user account is labeled as fake news spreader. As a general conclusion, these classifiers learned how to detect fake news spreaders accounts considering the presence or absence of Twitter entities mainly. Also, texts related to political people such as "trump", "hillary" or "obama" are considered factors to fake news texts.

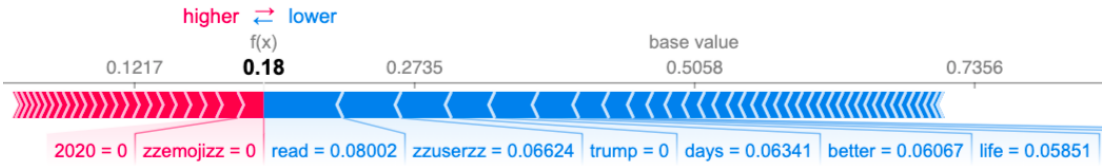

Figure 6.2: Interpretation of the best classifier's prediction 1

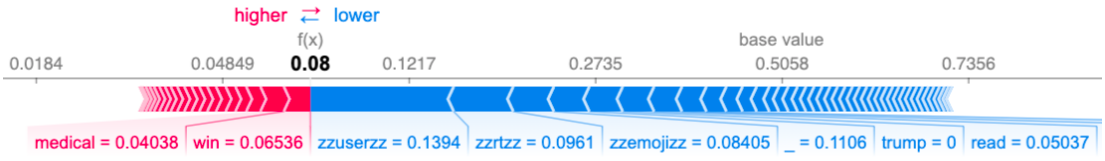

Figure 6.3: Interpretation of the best classifier's prediction 2 - English

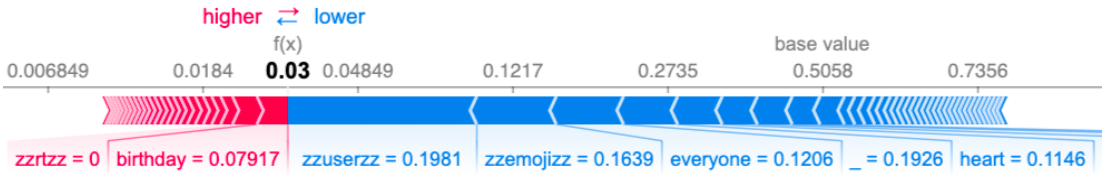

Figure 6.4: Interpretation of the best classifier's prediction 3 - English

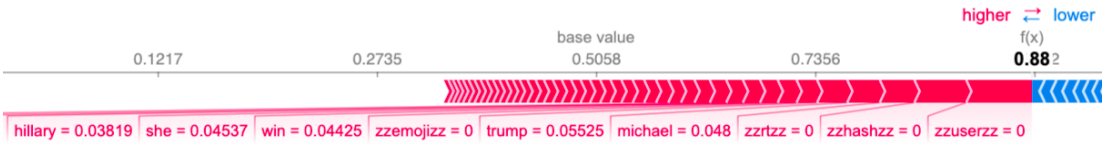

Figure 6.5: Interpretation of the best classifier's prediction 4 - English

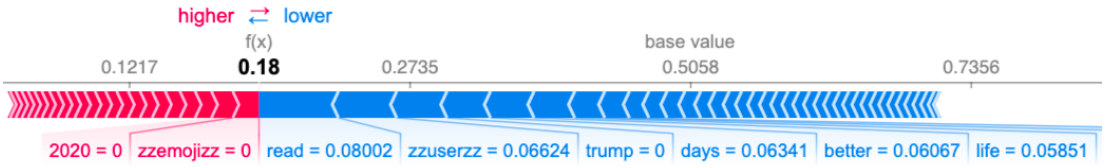

Figure 6.6: Interpretation of the best classifier's prediction 5 - English

6.2 Fake News Spreaders Profiling - Spanish

6.2.1 Results

Table 6.5 and Table 6.6 shows the results obtained for all individual classifiers, when predicts the test dataset. SVM classifier for combined unigrams and bigrams gets the best result, with an accuracy of 0.82. Table 6.7 presents the results of three final ensemble models. The first one is a LR meta classifier, and the second and third use soft and hard voting respectively. When working with the LR classifier, the ensemble model reaches its best accuracy result of 0.795, which is lower than the best individual classifier.

Table 6.5: Accuracy of Test dataset for Spanish PAN@CLEF classifiers - Part 1/2

Classifier	N-gram Type	Feature Selection	Accuracy
SVM	uni-bigram	LIME	**0.820**
SVM	uni-skipgram	LIME	0.800
MLP	uni-bigram	LIME	0.800
MLP	uni-skipgram	LIME	0.800
SVM	uni-bi-skipgram	LIME	0.795
MLP	uni-bi-skipgram	LIME	0.795
LR	uni-bigram	SHAP	0.785
LR	unigram	SHAP	0.780
LR	uni-skipgram	SHAP	0.780
LR	uni-bi-skipgram	SHAP	0.780
SVM	bigram	LIME	0.780
MLP	unigram	LIME	0.780
MLP	skipgram	LIME	0.780
SVM	unigram	LIME	0.775
KNN	uni-skipgram	LIME	0.775
RF	uni-bigram	SHAP	0.770
KNN	skipgram	LIME	0.770
LR	skipgram	SHAP	0.765
RF	uni-bi-skipgram	SHAP	0.765
SVM	skipgram	LIME	0.765
AdaBoost	uni-skipgram	SHAP	0.765
LR	bigram	SHAP	0.760
AdaBoost	uni-bi-skipgram	SHAP	0.760
MLP	bigram	LIME	0.760
LDA	bigram	SHAP	0.760

Table 6.6: Accuracy of Test dataset for Spanish PAN@CLEF classifiers - Part 2/2

Classifier	N-gram Type	Feature Selection	Accuracy
RF	uni-skipgram	SHAP	0.755
KNN	uni-bigram	LIME	0.755
KNN	uni-bi-skipgram	LIME	0.755
AdaBoost	uni-bigram	SHAP	0.750
LR	stats	SHAP	0.750
AdaBoost	skipgram	SHAP	0.745
LDA	uni-bigram	SHAP	0.745
RF	unigram	SHAP	0.740
AdaBoost	bigram	SHAP	0.740
RF	skipgram	SHAP	0.735
KNN	unigram	LIME	0.735
KNN	bigram	LIME	0.735
AdaBoost	unigram	SHAP	0.730
Naive Bayes	bigram	SHAP	0.730
RF	bigram	SHAP	0.725
LDA	unigram	SHAP	0.725
LDA	uni-bi-skipgram	SHAP	0.720
Naive Bayes	uni-bi-skipgram	SHAP	0.720
Naive Bayes	unigram	SHAP	0.715
LDA	skipgram	SHAP	0.710
Naive Bayes	uni-bigram	SHAP	0.705
LDA	uni-skipgram	SHAP	0.700
Naive Bayes	skipgram	SHAP	0.700
Naive Bayes	uni-skipgram	SHAP	0.685

Table 6.7: Metric Results for Test dataset - Spanish PAN@CLEF classifiers

Classifier	Precision	Recall	Accuracy	F1-Score
Ensemble Model using LR	0.8242	**0.7500**	**0.7950**	**0.7853**
Ensemble Model using Soft Voting	**0.8500**	0.6800	0.7800	0.7556
Ensemble Model using Hard Voting	0.8415	0.6900	0.7800	0.7582

6.2.2 Model Interpretation

In this case, the Ensemble Model with LR obtains the best results of the ensemble model approaches. Figure 6.7 shows which models have more importance to get the final predictions. Comparing these classifiers with Table 5.14, the conclusion is that this ensemble classifier is considering the classifiers with the best train accuracy results (the top 4 is integrated by MLP versions) and apparently, they are generalizing the problem sufficiently enough to get a better score. Surprisingly, the best individual classifier is not considered at the top most important classifiers for this ensemble model.

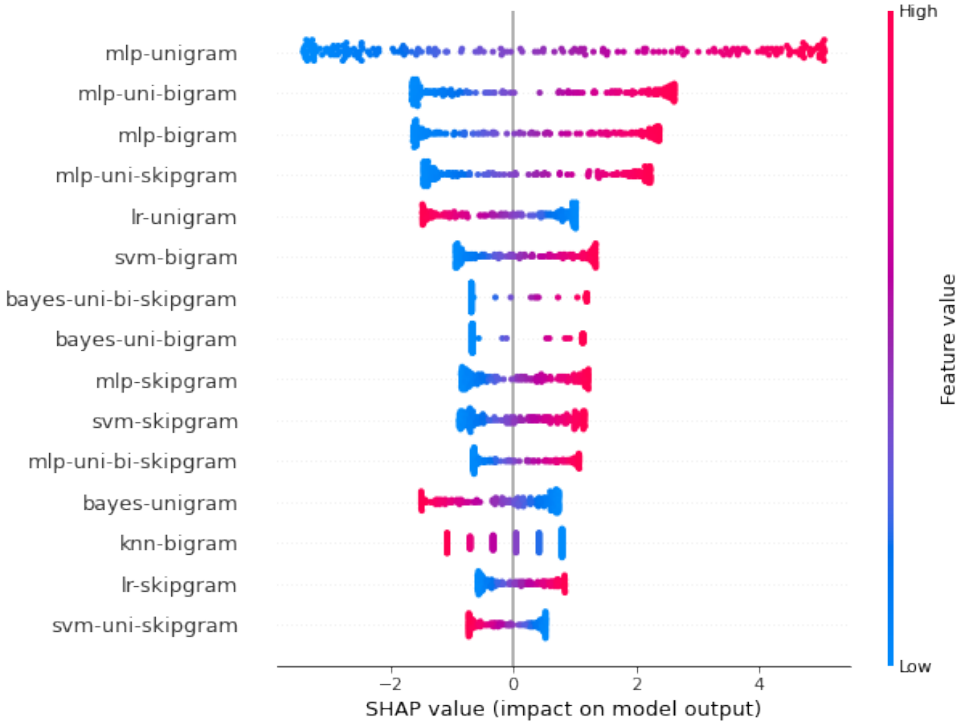

Figure 6.7: Top 15 of most important classifiers for ensemble model

With a deeper analysis of the behavior of each individual classifier, the theoretical maximum accuracy that the best ensemble approach can get is 0.96 using all classifiers. Two hundred predictions are evaluated and 7 of them cannot be evaluated correctly with none of the classifiers. The top 15 most important features used by the top 5 best classifiers are joined and described in Table 6.8. The interpretation of these features is: the more they increases their TF-IDF value, the more affect the prediction, according to the column they belong to. Top relevant features for all classifiers can be analyzed in Appendix B.

Figures from 6.8 to 6.11 represents some predictions that none of the classifiers can predict correctly. They are based on the best classifier developed which is SVM using unigrams and bigrams. Due to SVM has elevated computational cost when SHAP is applied, LIME is used instead.

Table 6.8: Most important features used by Spanish PAN@CLEF classifiers

Real news	**Fake news**
zzhashzz	unete
zzuserzz zzuserzz	video
zzuserzz	sánchez
zzhashzz zzhashzz	unete zzuserzz
zzurlzz zzhashzz	zzurlzz unete
zzrtzz	iglesias
zzrtzz zzuserzz	pedro
en zzhashzz	zzurlzz vía
zzuserzz y	dinero
madrid	facebook
de zzhashzz	podemos
qué	amor
zzemojizz	dios
sí	venezuela
algo	secreto
ha	
creo	
os	

Figure 6.8 shows the user is not a fake news spreader. According to Table 6.8, the use of Twitter entities like user's mentions, URLs, etc., have an impact towards real news, but apparently, this input is an exception to the learned rule and that is why it cannot be predicted correctly. The same happens with Figure 6.9, Figure 6.10 and Figure 6.11. All 7 incorrect predictions have this problem, and in conclusion, this rule related with the presence and absence of Twitter entities is less general than English corpus. Another consideration is related with the most important features, where no specific topic can be seen at first sight, hence, this model can be considered more general than English one, and it can be an explanation about why in PAN@CLEF 2020, the English corpus is more complicated to classify than Spanish corpus.

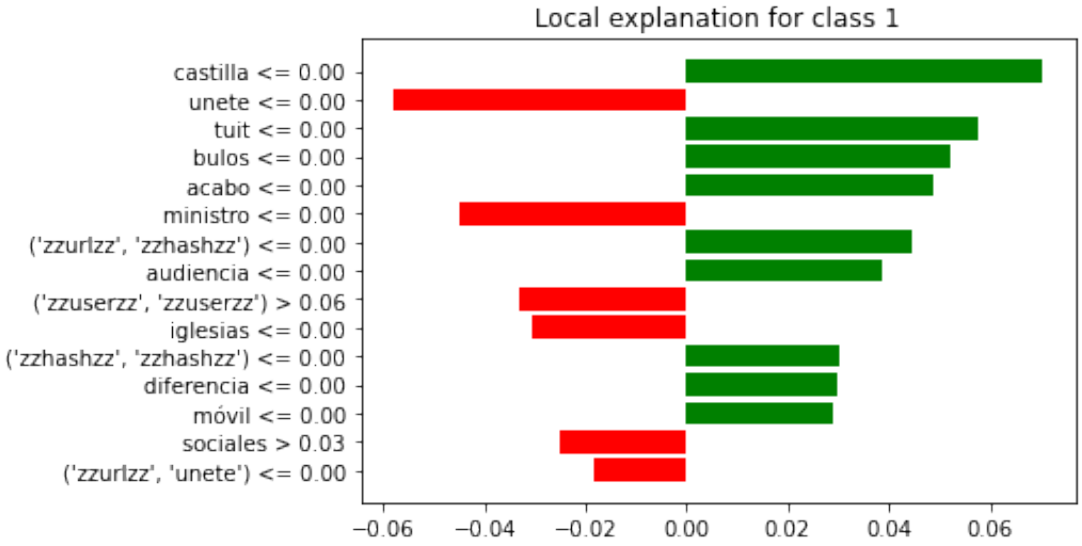

Figure 6.8: Interpretation of the best classifier's prediction 1 - Spanish

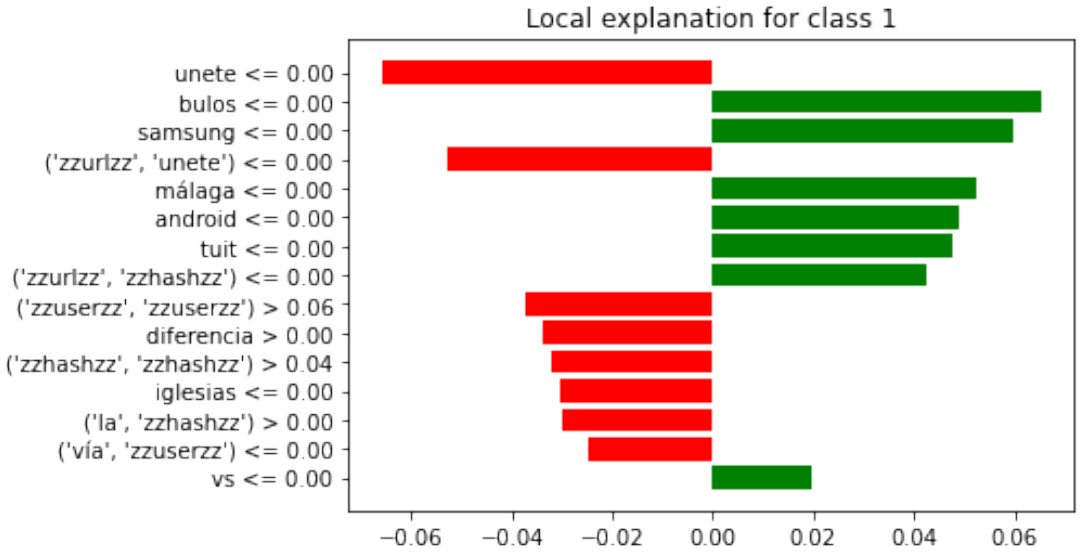

Figure 6.9: Interpretation of the best classifier's prediction 2 - Spanish

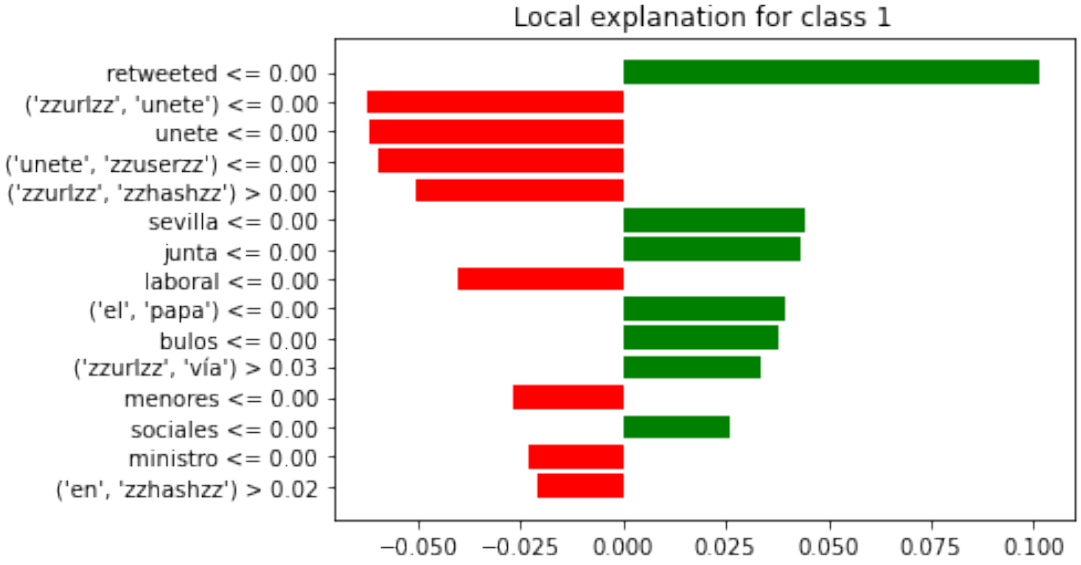

Figure 6.10: Interpretation of the best classifier's prediction 3 - Spanish

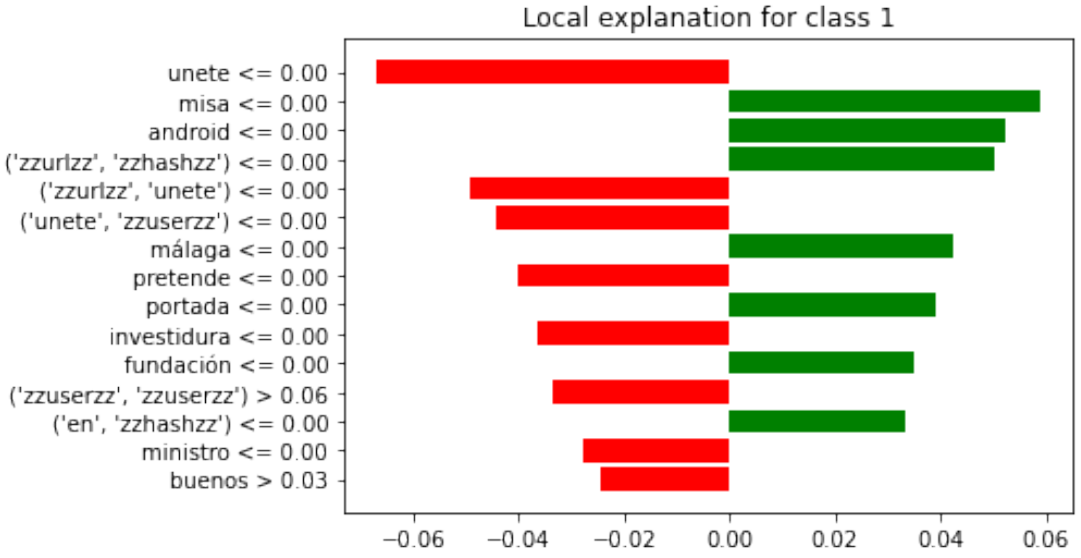

Figure 6.11: Interpretation of the best classifier's prediction 4 - Spanish

6.3 Comparison with PAN@CLEF 2020 Results

Table 6.9 presents the top 25 better results for the event, and they are compared to the results obtained by this work. The best classifier developed in this work surpass the event's first places. Considering that over 60 teams participated in this event, having solutions placed at the top 10 is meaningful. This proves that SHAP values and LIME values can be used as feature selection methods and they can get competitive results when extracting important features.

Table 6.9: Comparison with PAN@CLEF 2020 Results

Participant	EN Accuracy	ES Accuracy	Mean Accuracy
SVM+uni-bigrams	0.745	**0.820**	**0.7825**
bolonyai20 [31]	**0.750**	0.805	0.7775
pizarro20 [23]	0.735	0.820	0.7775
LR+uni-bigrams	0.745	0.785	0.7650
SVM+uni-bi-skipgrams	0.735	0.795	0.7650
LR+unigrams	0.745	0.780	0.7625
LR+uni-bi-skipgrams	0.735	0.780	0.7575
koloski20	0.715	0.795	0.7550
deborjavalero20	0.730	0.780	0.7550
vogel20	0.725	0.785	0.7550
Ensemble Hard Voting	0.730	0.780	0.7550
higueraporras20	0.725	0.775	0.7500
tarela20	0.725	0.775	0.7500
Ada+uni-bi-skipgrams	0.735	0.760	0.7475
babaei20	0.725	0.765	0.7450
staykovski20	0.705	0.775	0.7400
hashemi20	0.695	0.785	0.7400
estevecasademunt20	0.710	0.765	0.7375
castellanospellecer20	0.710	0.760	0.7350
shrestha20	0.710	0.755	0.7325
tommasel20	0.690	0.775	0.7325
johansson20	0.720	0.735	0.7275
murauer20	0.685	0.770	0.7275
espinosagonzales20	0.690	0.760	0.7250
ikae20	0.725	0.725	0.7250
morenosandoval20	0.715	0.730	0.7225
majumder20	0.640	0.800	0.7200
sanchezromero20	0.685	0.755	0.7200
lopezchilet20	0.680	0.755	0.7175
nadalalmela20	0.680	0.755	0.7175
carrodve20	0.710	0.725	0.7175
gil20	0.695	0.735	0.7150

6.4 Fake News Detection - Spanish

6.4.1 Results

Table 6.10 and Table 6.11 shows the results obtained for all classifiers, when predicts individually the test dataset outcomes. LR classifier for unigrams, bigrams and skipgrams combined gets the best result, with an F1 score of 0.8197. Table 6.12 presents the results of the final ensemble model when working with the use of majority voting, including soft and hard voting. When working as a majority hard voting classifier, the model reaches its best F1 score of 0.8028 which is lower than the best individual classifier score. These scores In this particular case, individual results cannot be compared, by the time this work is written, this event is taking place and there are several limitations to test multiple times different solution proposals.

Table 6.10: F1 Score for Development Dataset at FakeDes@Iberlef 2021 - Part 1/2

Classifier	N-gram Type	Feature Selection	Accuracy
LR	uni-bi-skipgrams	SHAP	**0.8197**
LR	uni-skipgrams	SHAP	0.8156
SVM	bigrams	LIME	0.8089
SVM	uni-bi-skipgrams	LIME	0.8086
SVM	uni-skipgrams	LIME	0.8085
MLP	bigrams	LIME	0.8070
AdaBoost	uni-bigrams	SHAP	0.8068
LR	bigrams	SHAP	0.8058
LR	uni-bigrams	SHAP	0.8042
MLP	uni-bi-skipgrams	LIME	0.8028
MLP	uni-bigrams	LIME	0.8027
NB	uni-bigrams	SHAP	0.8000
LR	unigrams	SHAP	0.7985
SVM	uni-bigrams	LIME	0.7985
MLP	skipgrams	LIME	0.7932
MLP	unigrams	LIME	0.7929
MLP	uni-skipgrams	LIME	0.7916
LR	skipgrams	SHAP	0.7913
SVM	skipgrams	LIME	0.7913
NB	uni-bi-skipgrams	SHAP	0.7883

Table 6.11: F1 Score for Development Dataset at FakeDes@Iberlef 2021 - Part - Part 2/2

Classifier	N-gram Type	Feature Selection	Accuracy
RF	uni-bi-skipgrams	SHAP	0.7880
SVM	unigrams	LIME	0.7810
NB	bigrams	SHAP	0.7786
RF	skipgrams	SHAP	0.7781
AdaBoost	uni-bi-skipgrams	SHAP	0.7781
AdaBoost	uni-skipgrams	SHAP	0.7770
LDA	uni-skipgrams	SHAP	0.7741
AdaBoost	skipgrams	SHAP	0.7718
NB	uni-skipgrams	SHAP	0.7670
RF	uni-skipgrams	SHAP	0.7661
AdaBoost	unigrams	SHAP	0.7642
NB	unigrams	SHAP	0.7642
KNN	uni-bigrams	LIME	0.7633
KNN	uni-bi-skipgrams	LIME	0.7633
RF	unigrams	SHAP	0.7612
RF	uni-bigrams	SHAP	0.7606
LDA	uni-bi-skipgrams	SHAP	0.7555
KNN	unigrams	LIME	0.7555
KNN	uni-skipgrams	LIME	0.7555
AdaBoost	bigrams	SHAP	0.7544
LDA	skipgrams	SHAP	0.7544
LDA	unigrams	SHAP	0.7491
NB	skipgrams	SHAP	0.7482
RF	bigrams	SHAP	0.7405
KNN	bigrams	LIME	0.7352
LDA	uni-bigrams	SHAP	0.7232
LDA	bigrams	SHAP	0.6992
LR	statistical features	SHAP	0.6863
KNN	skipgrams	LIME	0.6498

Table 6.12: Ensemble Model Results for FakeDes@Iberlef 2021 Development Dataset

Classifier	Precision	Recall	Accuracy	F1-Score
Ensemble Model using LR	**0.8154**	0.7465	0.7966	0.7794
Ensemble Model using Soft Voting	0.8058	0.7887	0.8068	0.7972
Ensemble Model using Hard Voting	0.8028	**0.8028**	**0.8102**	**0.8028**

6.4.2 Model Interpretation

In this case, the Ensemble Model with Hard Voting gets the best results of the ensemble model approaches. This approach considers all individual classifiers with the same importance and the prediction is a majority voting. With a deeper analysis of the behavior of each individual classifier, the theoretical maximum accuracy that the best ensemble approach can get using the development dataset is 1.0, which means all development inputs can be predicted correctly using the individual classifiers developed. The top 15 most important features used by the top 5 best classifiers are joined and described in Table 6.13. The interpretation of these features is: the more they increases their TF-IDF value, the more affect the prediction, according to the column they belong to. Top relevant features for all classifiers can be analyzed in Appendix B.

Table 6.13: Most important features used by Spanish FakeDes@Iberlef classifiers

Real news	**Fake news**
en el	: "
. en	, y
zznumberzz de	, pues
en zznumberzz	no se
. el	ya que
de zznumberzz	ya
" ,	!
. la	todo
el zznumberzz	"
zznumberzz .	esto
zznumberzz ,	:
, "	ha
zznumberzz mil	pues
zznumberzz y	lo
en la	será
zznumberzz	se
tiene	nos
’	que
aseguró	de acuerdo
mil	
;	
sobre	
dijo	
- zznumberzz	
. zznumberzz	
?	
méxico	
gobierno	
contra	

According to Table 6.13, the classifiers are focusing in punctuation and numbers' presence. No topic can be related at first sight to these features, and although there are no test results, it is possible to interpret some test inputs in order to understand the model's behaviour. The following text is an input of the test dataset which is labeled incorrectly, due to it is clarifying fake information:

'Agua con limón y bicarbonato no previene el Covid-19 Un té de limón con bicarbonato no elimina el coronavirus. Se dice que esta mezcla "mata de manera inmediata el virus" y "lo elimina completamente del cuerpo".Tal afirmación viene sostenida por el siguiente argumento: "Estos dos componentes alcalinizan el sistema inmunológico, ya que, cuando cae la noche, el sistema se acidifica y bajan las defensas". Datos La mezcla de limón y bicarbonato en agua caliente no sirve para combatir el coronavirus ni alcaliniza el organismo. No se ha demostrado que ningún alimento o bebida proteja contra el coronavirus. El doctor Jaime Barrio, del Consejo Científico del Colegio Oficial de Médicos de Madrid (Icomem), recuerda que la propia Organización Mundial de la Salud (OMS) y otros organismos oficiales se han pronunciado en este sentido. Del mismo modo, las autoridades sanitarias espanolas precisan que "no hay que tomar precauciones especiales" con los alimentos...'

Figure 6.12: Interpretation of an incorrect prediction 4 - FakeDes@Iberlef

Figure 6.12 shows the most important features used to predict that the text is related to fake news. Some features considered as fake news indicators shown in Table 6.13 are presented in this example (features in red color), and they provoke this result.

On the other hand, here is an example when the classifier is working correctly:

'Exempleada de Facebook denuncia nulas medidas para evitar manipulación política Sabemos que además de ser un buen lugar para compartir con nuestros seres queridos, Facebook es una herramienta de comunicación poderosa para hacer llegar todo tipo de mensajes a millones de personas, por lo que también puede ser un arma poderosa. Así lo sabe Sophie Zhang, una extrabajadora de la empresa que aseguró en un informe sobre las irregularidades de la red social estar segura de "tener sangre en las manos" luego de haber colaborado con la compañía. Según la ingeniera, Facebook ha sido lento en reaccionar o ha ignorado la existencia de campanas organizadas para distribuir desinformación política e influir en los resultados de elecciones en países de todo el mundo, según el informe interno de la exempleada de la compañía. Te recomendamos: Facebook elimina cuentas que buscaban influir de forma política en América Latina "La verdad es que simplemente no nos preocupábamos lo suficiente como para'

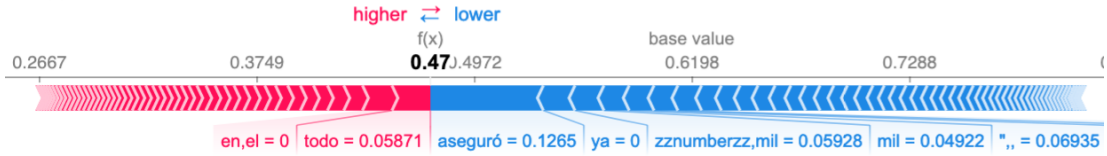

Figure 6.13: Interpretation of a correct prediction 4 - FakeDes@Iberlef

Here the classifier predicts this text as real information, and in fact it is. According to Table 6.13, the presence of *zznumberzz mil* and de ausence of *ya* have an impact to the result towards real news detection. The real article can be found in [43].

6.5 Comparison with FakeDes@Iberlef 2021 Event

Table 6.14 presents the results evaluated at CodaLab Platform. The event has a limitation of 2 submissions per user and the evaluation method is using the F1-Score for "Fake" class. The best approaches can compete with the other participants, and as a conclusion, the solution approach for this model can solve both tasks as well as other works found in the State of the Art.

Table 6.14: Comparison with FakeDes@Iberlef 2021 - Development Dataset Results

Classifier	F1-Score
ClaudiaPorto	**0.8498**
GuanZhengyi	0.8497
vitiugin	0.8198
LR+Uni-bi-skipgrams	0.8197
Ensemble Model	0.8028
spalenza	0.7973
Maoqin	0.7865
gamallo	0.7655
luiso91	0.7576
jorge.reyes	0.7402

6.6 Summary of Best Classifiers per Corpus

As a summary, Figure 6.14, Figure 6.15 and Figure 6.16 presents the top 10 best classifiers developed in this work for each corpus when applying SHAP and LIME as feature selector methods. The figures present the accuracy and the F1 score respectively.

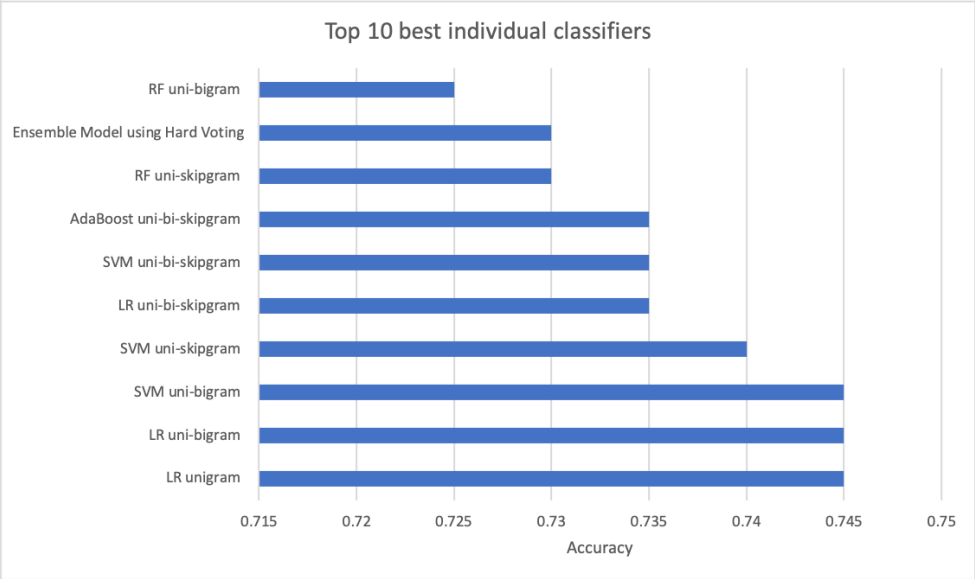

Figure 6.14: Best classifiers for PAN@CLEF 2020 - English

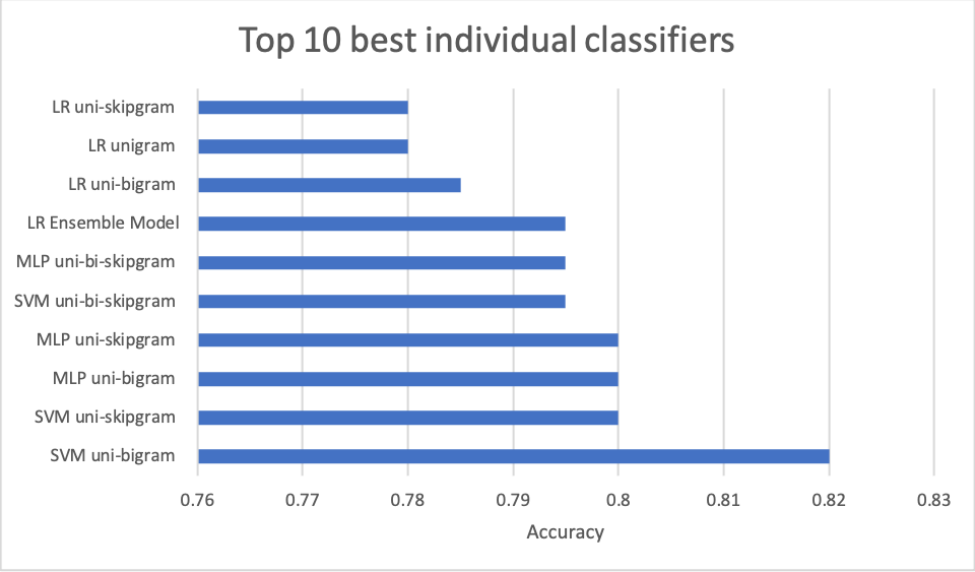

Figure 6.15: Best classifiers for PAN@CLEF 2020 - Spanish

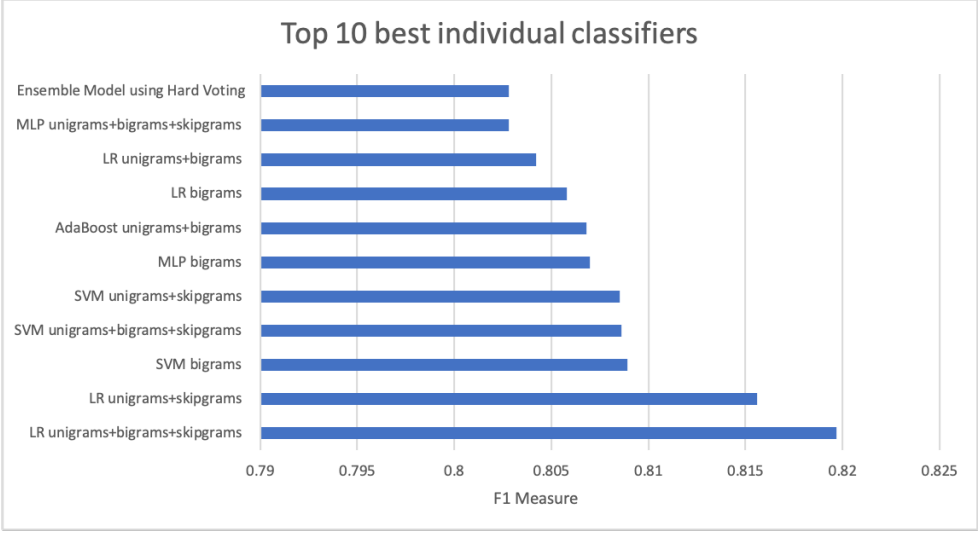

Figure 6.16: Best classifiers for FakeDes@Iberlef

7 Conclusions and Future Work

7.1 Conclusions

7.1.1 Learning from experiment failures

Fake news identification can be a complex problem due to high data representations. This is the main reason a feature selection method using SHAP and LIME is considered. They provide a way to interpret results and focus on relevant features in order to give a better solution approach. About features used in this work, emojis and emoticons do not provide good results when using them as individual features. The same happens with stop words with length less than 5 characters and, many classifiers do not consider punctuation characters as features. Skipgrams are more complex than bigrams and unigrams, so additional experiments are analyzed in order to select a good representation when using them. One of them is to consider reversed texts, and by this approach, the importance of having two n-grams in a text does not take into consideration the sequence of them. Unfortunately, this approach did not give better results than using normal skipgrams, hence it was not contemplated in the final solution.

7.1.2 Ensemble Model

The purpose of developing an ensemble model in this work, is to compare multiple classifiers found in literature and take advantages from each one of them. Although it did not have the best performance in results, it cannot be discarded because it can provide more generalization given the different characteristics each classifier takes in consideration.

7.1.3 Interpretations

One of the main goal of this work is the interpretation of the final model's outcomes, using a XAI. SHAP and LIME techniques provide the contribution per feature when predicting an outcome, and by this way, users can interpret the model's knowledge about the task. This particular interpretation can provide enough information to give a set of rules the model is considering. SHAP time cost is high when working with classifiers that does not have an efficient way to calculate their feature contributions. SVM, KNN and MLP are the most expensive models and an exact interpretation of them are analyzed using LIME. For the PAN@CLEF 2020 corpus, the interpretation is that the models are learning which topics are most common for fake news spreaders in both languages, which it might imply

this is not a general solution for the problem. For the FakeDes@Iberlef 2021 corpus, the interpretation is done through specific words and punctuation for a single topic which is politics. In this case, the conclusion is that the model has learned specific features for fake news detection.

7.1.4 Comparison between English and Spanish

In PAN@CLEF event, English is the most difficult language to work with. Assuming the interpretation previously described, it might be due to the difficulty of split users by the topics they talk about. Assuming that a topic can be relevant to the tasks, it is possible to build a bilingual approach using the properties of this work, or a new corpus where topic is not relevant could be considered in order to get a more general solution for both tasks.

7.1.5 Considerations

To consider a user account as fake spreader could not be ethical. Every user in most of social media platforms have the right of publish what they want (of course with their respective limitation and political rules of the platform) and being tagged as a "spread fake news user" can be something to treat in a delicate way. The publication of this work could help fake news spreaders to change their way they share information, and it could be a explanation why this problem is evolving through time.

7.1.6 Applications

When there is a tendency about a topic that is relevant for general society, users want to know as many information as they can obtain, but not all information is useful. If someone wants to inform reliable information, he/she can use this work as an approach to detect information that are fake for an specific problem. Of course, this work can improve its results if more data is provided. This solution can complement a whole system where fake news spreaders and fake news are detected, where each module focuses on different feature representations in order to difficult users to hack the system detector.

7.2 Future Work

The analysis of other features such as sentiments, POS tagging, or embeddings can improve the results of the present work. According to the State of the Art, all solution approaches that considered this kind of data representation did not obtain better results than the ones that used statistical and frequency features, but by using any XIA tool, it could be possible to explain or at least have an idea about why they could not reach a better score.

The comparison with other feature selection methods can be a good justification about why SHAP and LIME can be used for that kind of process, in addition, SHAP and LIME methods can be compared and analyzed in order to select the better method for a task.

Evaluating the predictions of this solution approach with other datasets and other languages can explain deeper the process this work proposes, hence this model could work with multilingual texts.

Bibliography

[1] J. Bevendorff, B. Ghanem, A. Giachanou, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. Rangel, P. Rosso, *et al.*, "Overview of PAN 2020: Authorship Verification, Celebrity Profiling, Profiling Fake News Spreaders on Twitter, and Style Change Detection," in *International Conference of the Cross-Language Evaluation Forum for European Languages*, pp. 372–383, Springer, 2020.

[2] F. Rangel, A. Giachanou, B. Ghanem, and P. Rosso, "Overview of the 8th Author Profiling Task at PAN 2020: Profiling Fake News Spreaders on Twitter," in *CLEF*, 2020.

[3] T. Beysolow II, *Applied Natural Language Processing with Python: Implementing Machine Learning and Deep Learning Algorithms for Natural Language Processing*. Apress, 2018.

[4] C. Kang and A. Goldman, "In Washington pizzeria attack, fake news brought real guns," *New York Times*, vol. 5, 2016.

[5] M. R. Islam, S. Liu, X. Wang, and G. Xu, "Deep learning for misinformation detection on online social networks: a survey and new perspectives," *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 1–20, 2020.

[6] J. Thanaki, *Python natural language processing*. Packt Publishing Ltd, 2017.

[7] A. Artasanchez and P. Joshi, *Artificial Intelligence with Python: Your complete guide to building intelligent apps using Python 3. x*. Packt Publishing Ltd, 2020.

[8] A. C. Müller and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[9] P. Harrington, *Machine learning in action*. Manning Publications Co., 2012.

[10] S. Raschka, "Naive Bayes and text classification I: introduction and theory. arXiv e-prints," *arXiv preprint arXiv:1410.5329*, 2014.

[11] S. Raschka, *Python machine learning*. Packt publishing ltd, 2015.

[12] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[13] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[14] M. T. Ribeiro, S. Singh, and C. Guestrin, "" Why should I trust you?" Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[15] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *arXiv preprint arXiv:1705.07874*, 2017.

[16] Á. Hernández-Castañeda, H. Calvo, A. Gelbukh, and J. J. G. Flores, "Cross-domain deception detection using support vector networks," *Soft Computing*, vol. 21, no. 3, pp. 585–595, 2017.

[17] N. K. Conroy, V. L. Rubin, and Y. Chen, "Automatic deception detection: Methods for finding fake news," *Proceedings of the Association for Information Science and Technology*, vol. 52, no. 1, pp. 1–4, 2015.

[18] V. Agarwal, H. P. Sultana, S. Malhotra, and A. Sarkar, "Analysis of Classifiers for Fake News Detection," *Procedia Computer Science*, vol. 165, pp. 377–383, 2019.

[19] P. Meel and D. K. Vishwakarma, "Fake news, rumor, information pollution in social media and web: A contemporary survey of state-of-the-arts, challenges and opportunities," *Expert Systems with Applications*, vol. 153, p. 112986, 2020.

[20] H. E. Wynne and Z. Z. Wint, "Content based fake news detection using N-gram models," in *Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services*, pp. 669–673, 2019.

[21] S. Helmstetter and H. Paulheim, "Weakly supervised learning for fake news detection on Twitter," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 274–277, IEEE, 2018.

[22] D. Y. Espinosa, H. Gómez-Adorno, and G. Sidorov, "Profiling fake news spreaders using character and words n-grams," in *CLEF*, 2020.

[23] J. Pizarro, "Using N-grams to detect Fake News Spreaders on Twitter," in *CLEF*, 2020.

[24] J. C. Reis, A. Correia, F. Murai, A. Veloso, and F. Benevenuto, "Explainable machine learning for fake news detection," in *Proceedings of the 10th ACM conference on web science*, pp. 17–26, 2019.

[25] H. S. Al-Ash and W. C. Wibowo, "Fake news identification characteristics using named entity recognition and phrase detection," in *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pp. 12–17, IEEE, 2018.

[26] C. Boididou, S. Papadopoulos, M. Zampoglou, L. Apostolidis, O. Papadopoulou, and Y. Kompatsiaris, "Detection and visualization of misleading content on Twitter," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 1, pp. 71–86, 2018.

[27] A. Shrestha, F. Spezzano, and A. Joy, "Detecting Fake News Spreaders in Social Networks via Linguistic and Personality Features," in *CLEF*, 2020.

[28] E. Fersini, J. Armanini, and M. D'Intorni, "Profiling fake news spreaders: stylometry, personality, emotions and embeddings," in *CLEF*, 2020.

[29] F. A. Ozbay and B. Alatas, "Fake news detection within online social media using supervised artificial intelligence algorithms," *Physica A: Statistical Mechanics and its Applications*, vol. 540, p. 123174, 2020.

[30] M. Aldwairi and A. Alwahedi, "Detecting fake news in social media networks," *Procedia Computer Science*, vol. 141, pp. 215–222, 2018.

[31] J. Buda and F. Bolonyai, "An Ensemble Model Using N-grams and Statistical Features to Identify Fake News Spreaders on Twitter," in *CLEF*, 2020.

[32] H. Rashkin, E. Choi, J. Y. Jang, S. Volkova, and Y. Choi, "Truth of varying shades: Analyzing language in fake news and political fact-checking," in *Proceedings of the 2017 conference on empirical methods in natural language processing*, pp. 2931–2937, 2017.

[33] I. Vogel and M. Meghana, "Detecting Fake News Spreaders on Twitter from a Multilingual Perspective," in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 599–606, IEEE, 2020.

[34] B. Koloski, S. Pollak, and B. Škrlj, "Multilingual Detection of Fake News Spreaders via Sparse Matrix Factorization," in *CLEF*, 2020.

[35] R. Manna, A. Pascucci, and J. Monti, "Profiling fake news spreaders through stylometry and lexical features. UniOR NLP@ PAN2020," in *CLEF*, 2020.

[36] I. Vogel and M. Meghana, "Fake News Spreader Detection on Twitter using Character N-Grams," in *CLEF*, 2020.

[37] N. Ruchansky, S. Seo, and Y. Liu, "Csi: A hybrid deep model for fake news detection," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 797–806, 2017.

[38] P. Bahad, P. Saxena, and R. Kamal, "Fake News Detection using Bi-directional LSTM-Recurrent Neural Network," *Procedia Computer Science*, vol. 165, pp. 74–82, 2019.

[39] K. Popat, S. Mukherjee, A. Yates, and G. Weikum, "Declare: Debunking fake news and false claims using evidence-aware deep learning," *arXiv preprint arXiv:1809.06416*, 2018.

[40] R. Labadie-Tamayo, D. Castro-Castro, and R. Ortega-Bueno, "Fusing Stylistic Features with Deep-learning Methods for Profiling Fake News Spreaders," 2020.

[41] S. B. Majumder and D. Das, "Detecting Fake News Spreaders on Twitter Using Universal Sentence Encoder," in *CLEF*, 2020.

[42] D. Katsaros, G. Stavropoulos, and D. Papakostas, "Which machine learning paradigm for fake news detection?," in *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 383–387, IEEE, 2019.

[43] J. Wakefield, "Facebook: la exempleada que denuncia la responsabilidad de la red en las campañas de manipulación de todo el mundo." url:https://www.bbc.com/mundo/noticias-54172189, 2020. Accessed 05-07-2021.

8 Appendix A

8.1 Feature Selection Plots for English PAN@CLEF Classifiers

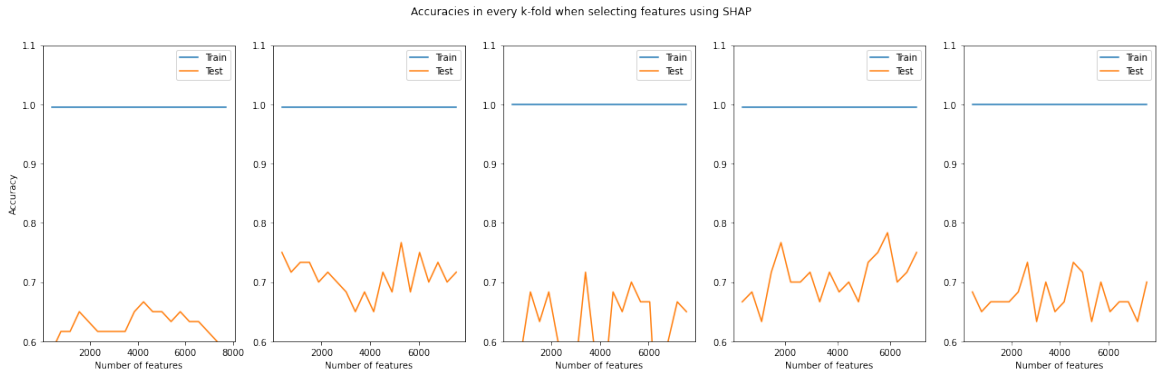

Figure 8.1: Accuracy per subset of relevant Features for AdaBoost using unigrams

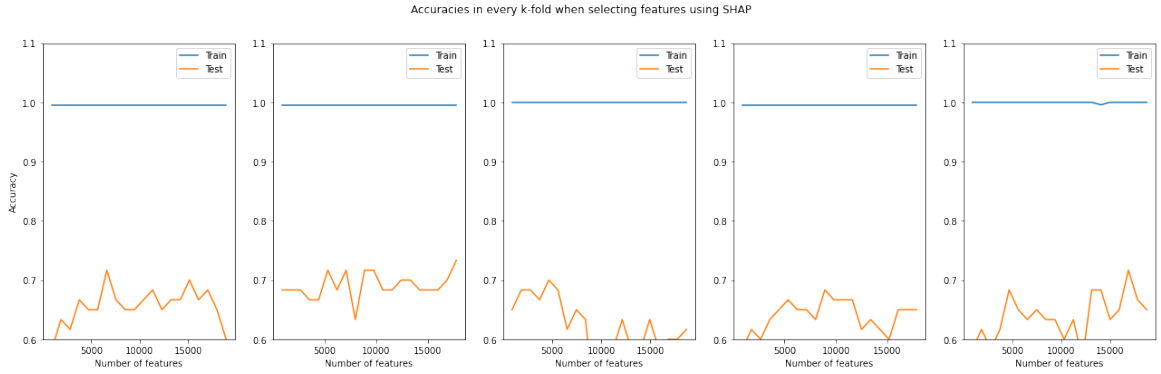

Figure 8.2: Accuracy per subset of relevant Features for AdaBoost using bigrams

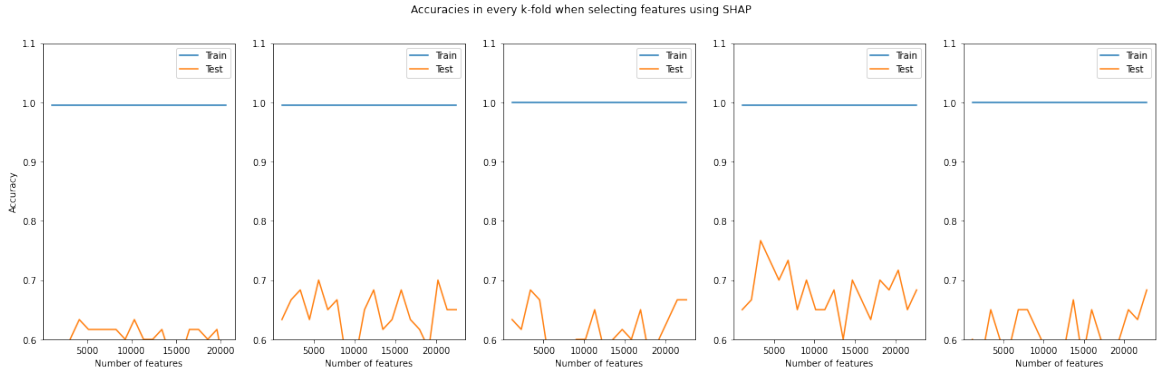

Figure 8.3: Accuracy per subset of relevant Features for AdaBoost using skipgrams

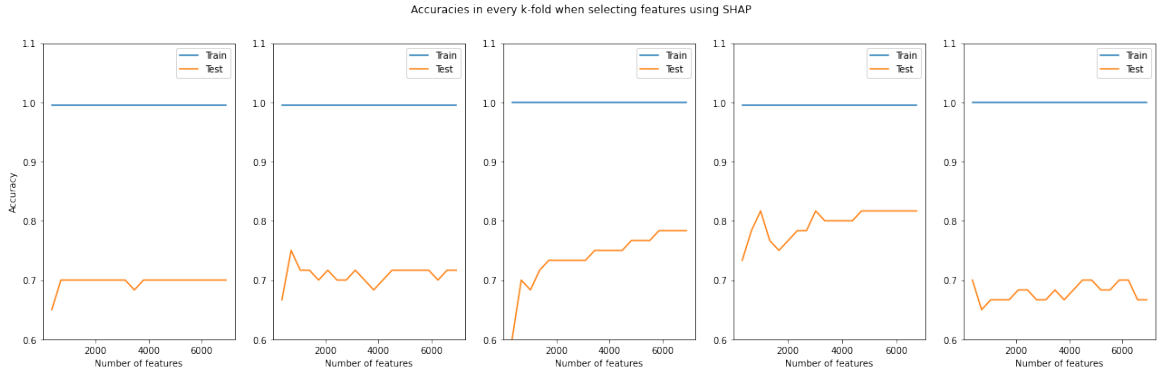

Figure 8.4: Accuracy per subset of relevant Features for Logistic Regression using unigrams

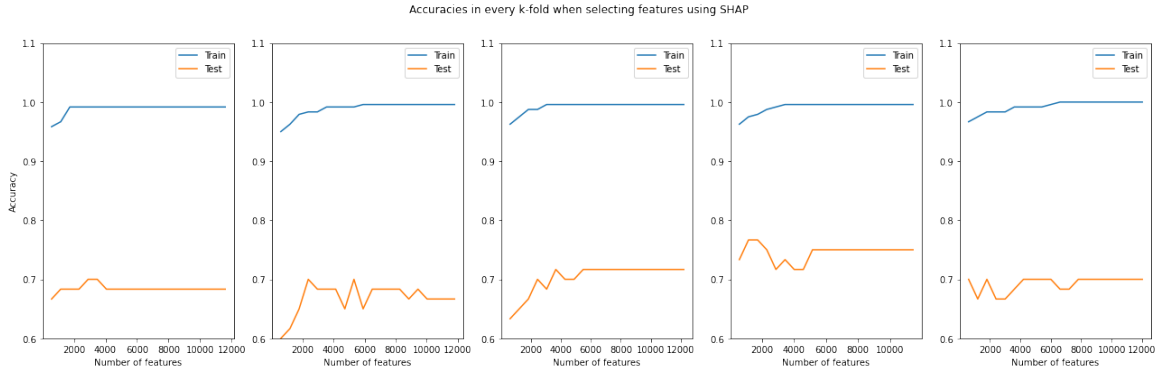

Figure 8.5: Accuracy per subset of relevant Features for Logistic Regression using bigrams

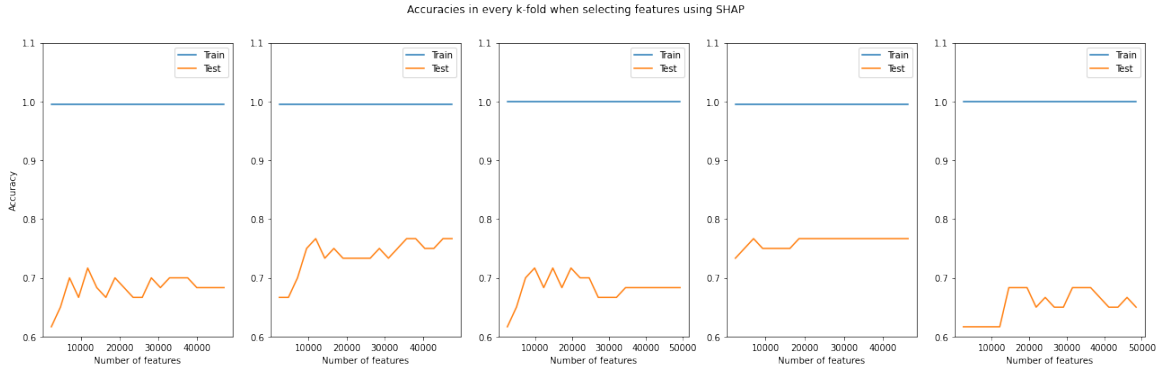

Figure 8.6: Accuracy per subset of relevant Features for Logistic Regression using skipgrams

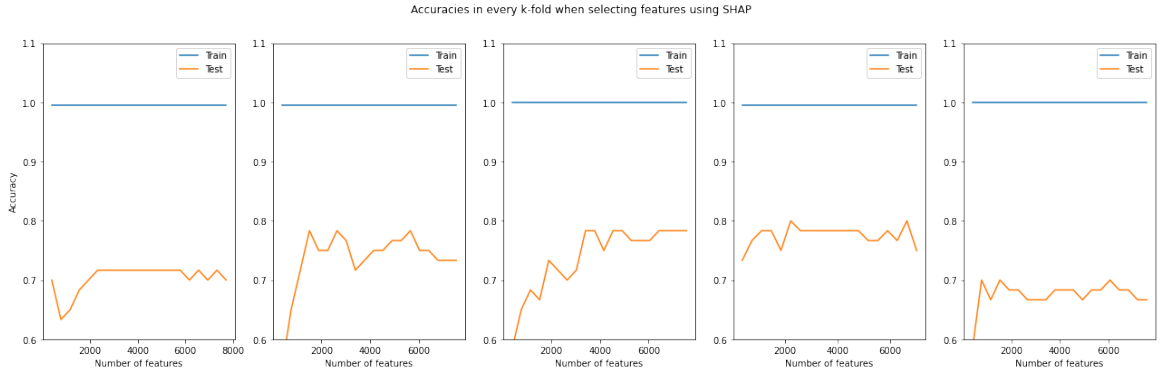

Figure 8.7: Accuracy per subset of relevant Features for Support Vector Machine using unigrams

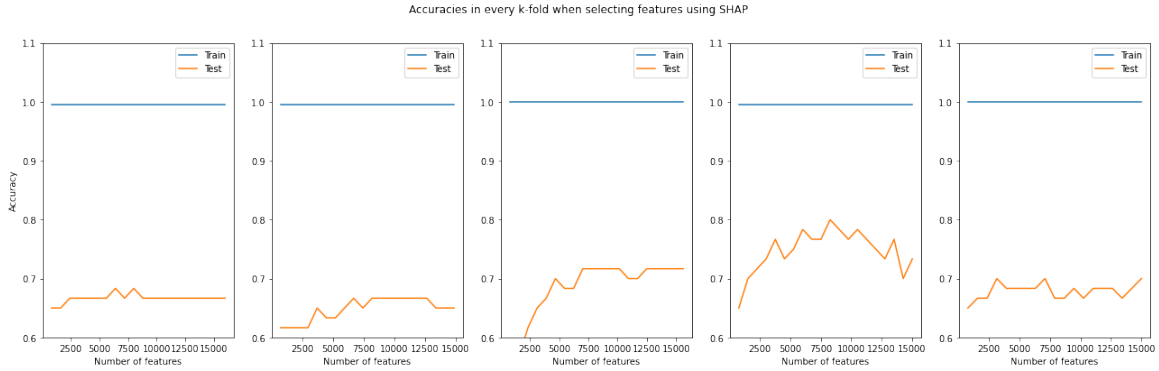

Figure 8.8: Accuracy per subset of relevant Features for Support Vector Machine using bigrams

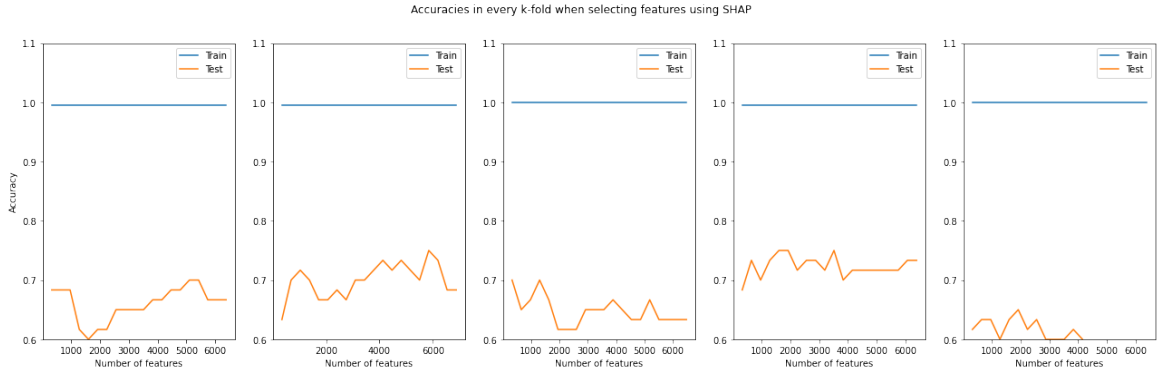

Figure 8.9: Accuracy per subset of relevant Features for Support Vector Machine using skipgrams

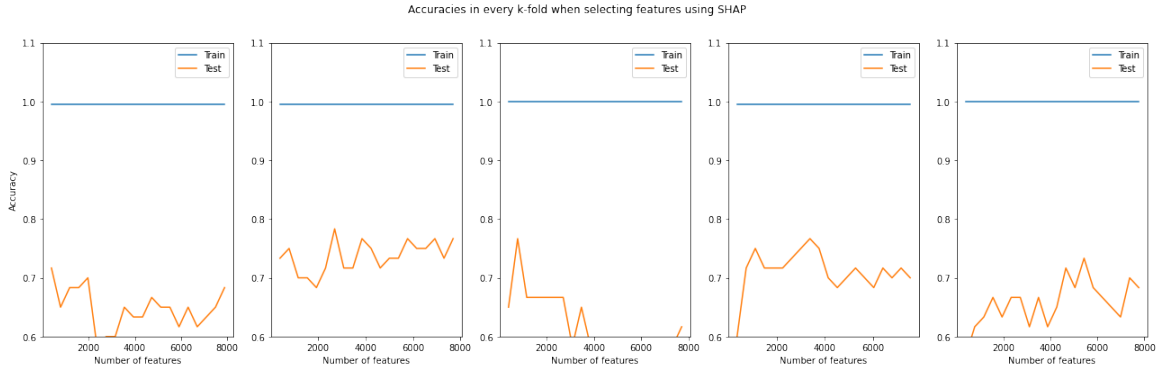

Figure 8.10: Accuracy per subset of relevant Features for Random Forest using unigrams

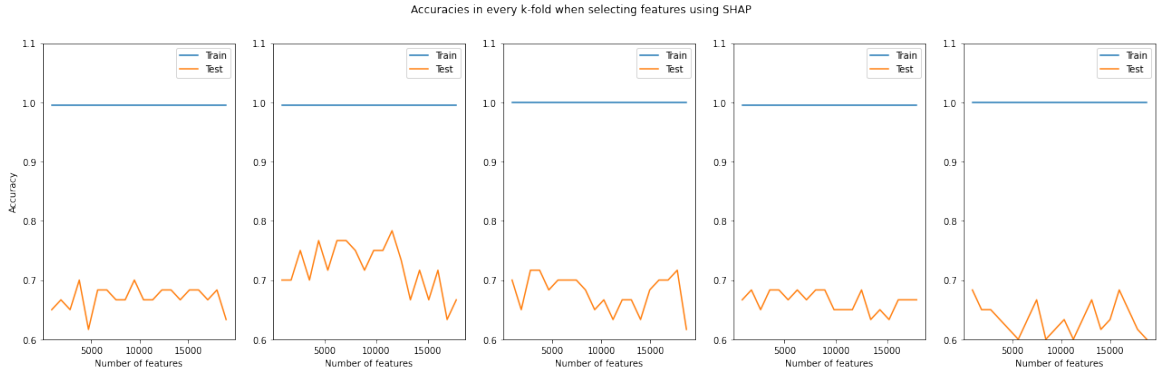

Figure 8.11: Accuracy per subset of relevant Features for Random Forest using bigrams

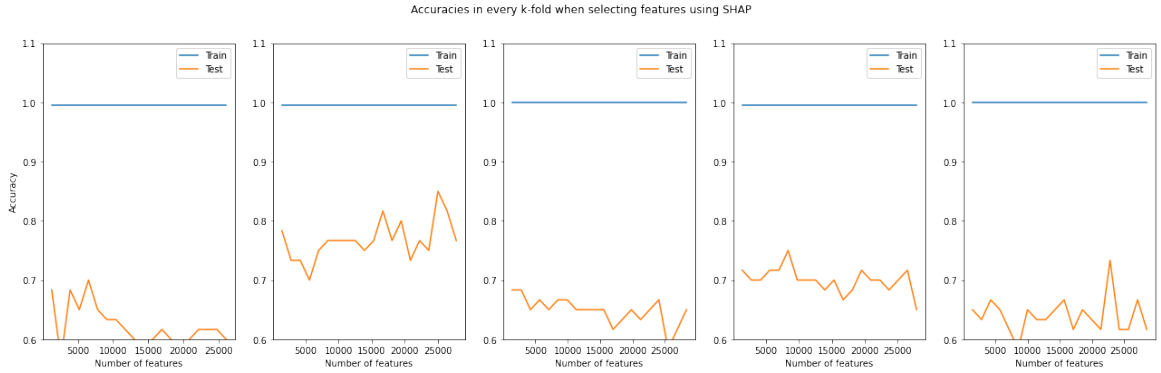

Figure 8.12: Accuracy per subset of relevant Features for Random Forest using skipgrams

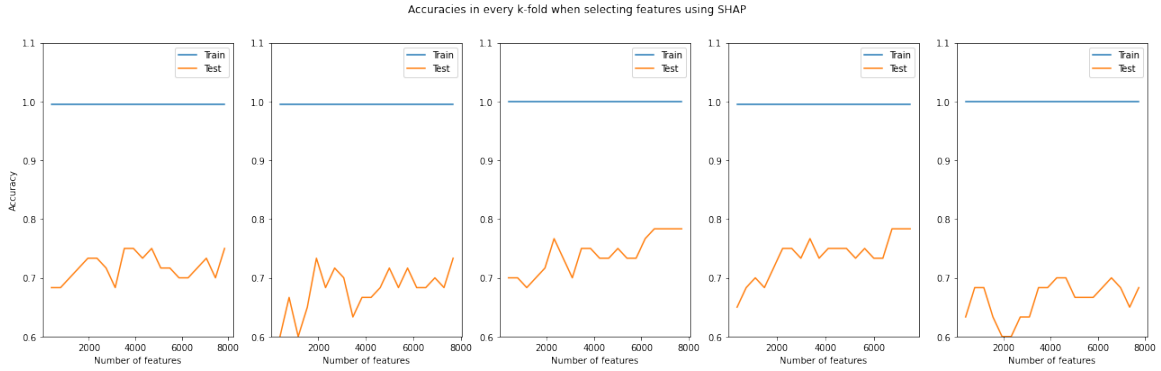

Figure 8.13: Accuracy per subset of relevant Features for Multilayer Perceptron using unigrams

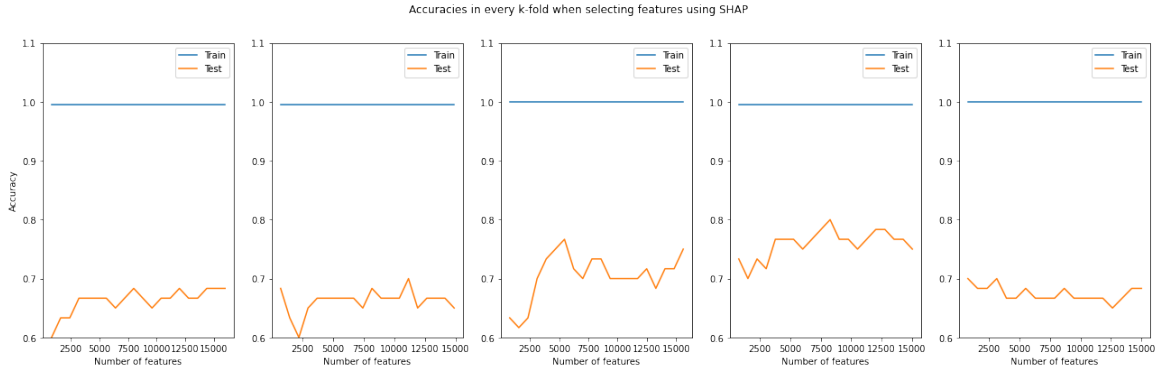

Figure 8.14: Accuracy per subset of relevant Features for Multilayer Perceptron using bigrams

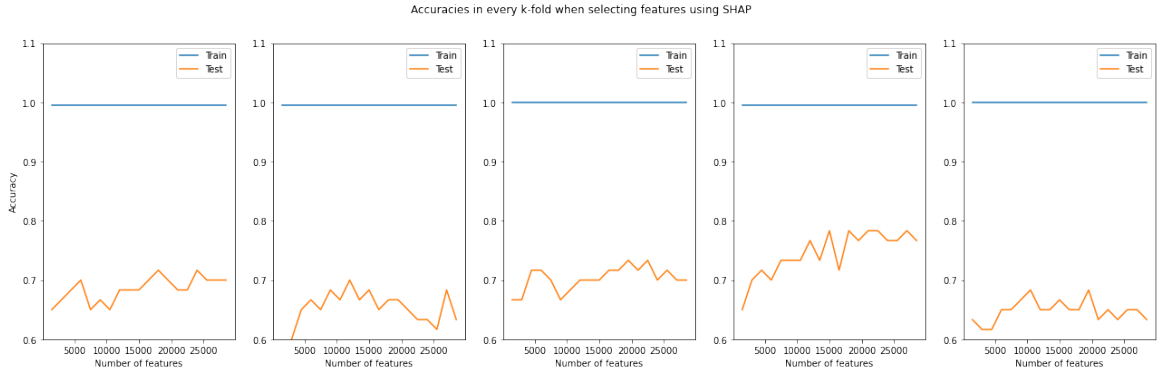

Figure 8.15: Accuracy per subset of relevant Features for Multilayer Perceptron using skipgrams

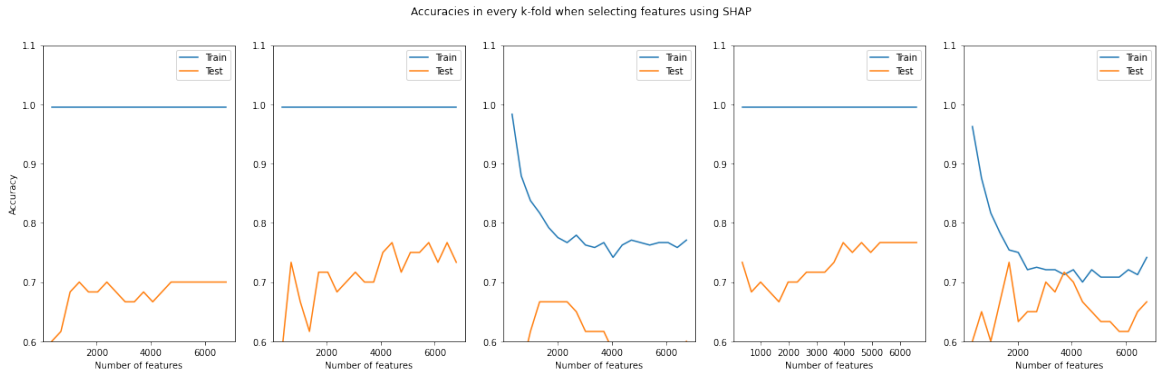

Figure 8.16: F1-Score per subset of relevant Features for Linear Discriminant Analysis using unigrams

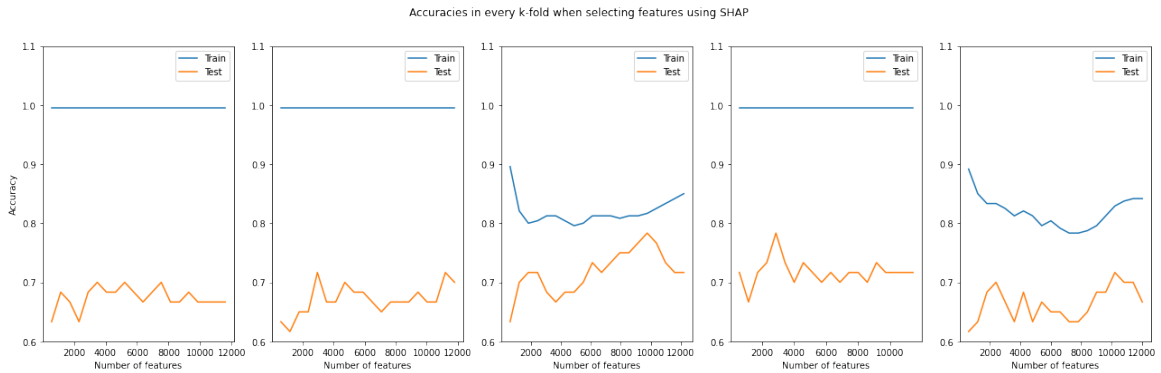

Figure 8.17: F1-Score per subset of relevant Features for Linear Discriminant Analysis using bigrams

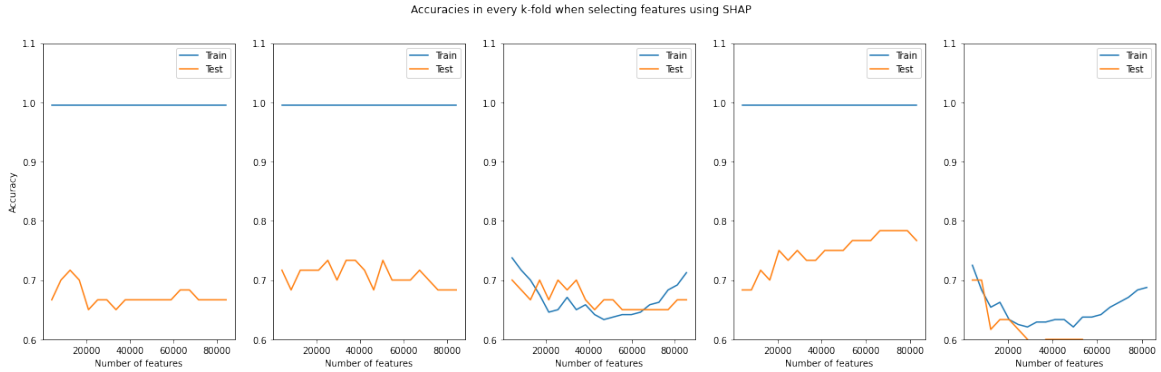

Figure 8.18: F1-Score per subset of relevant Features for Linear Discriminant Analysis using skipgrams

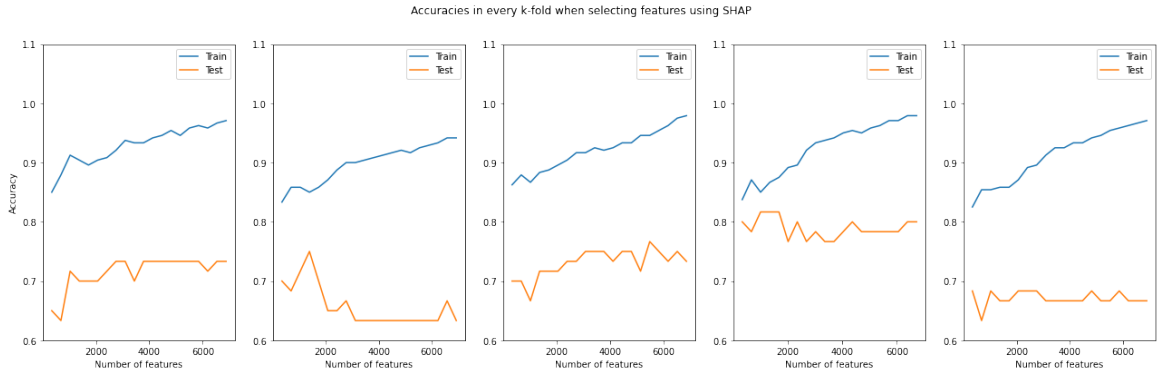

Figure 8.19: F1-Score per subset of relevant Features for Naive Bayes using unigrams

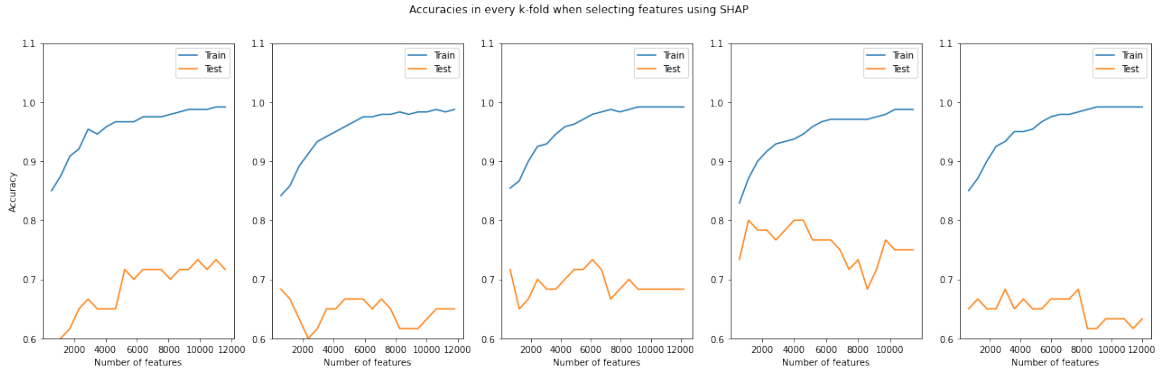

Figure 8.20: F1-Score per subset of relevant Features for Naive Bayes using bigrams

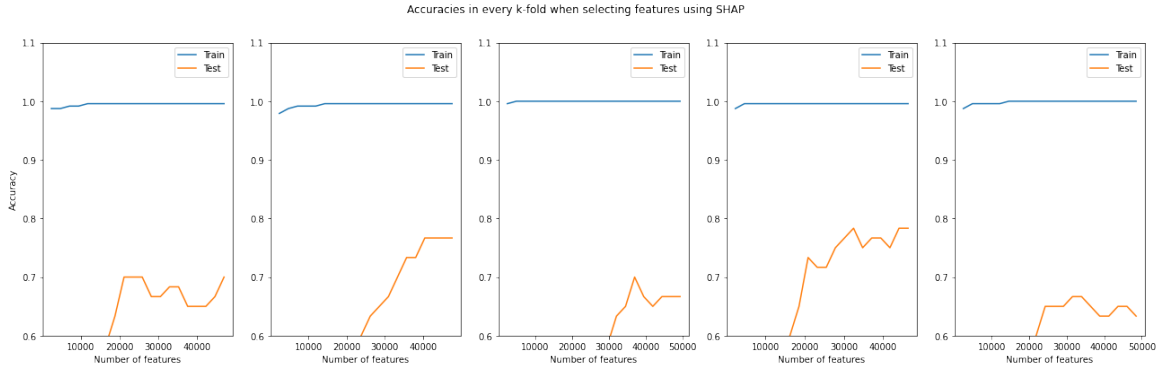

Figure 8.21: F1-Score per subset of relevant Features for Naive Bayes using skipgrams

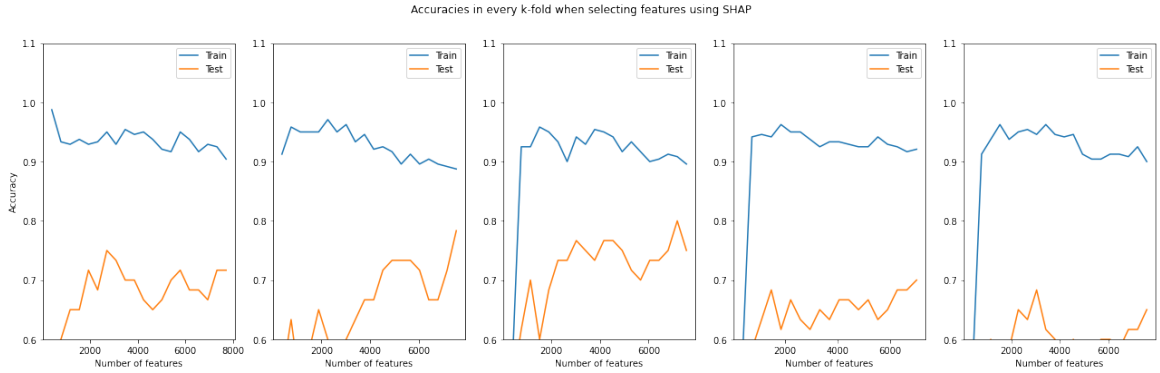

Figure 8.22: F1-Score per subset of relevant Features for K-Nearest Neighbors using unigrams

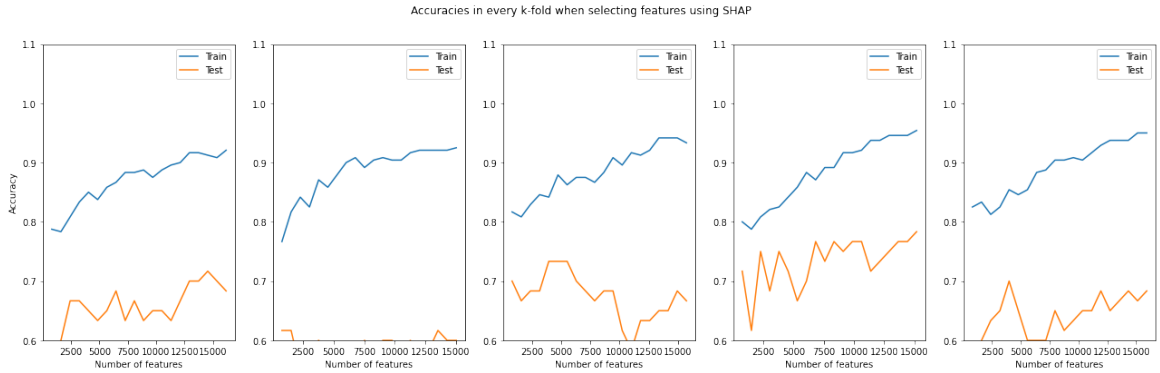

Figure 8.23: F1-Score per subset of relevant Features for K-Nearest Neighbors using bigrams

8.2 Feature Selection Plots for Spanish PAN@CLEF Classifiers

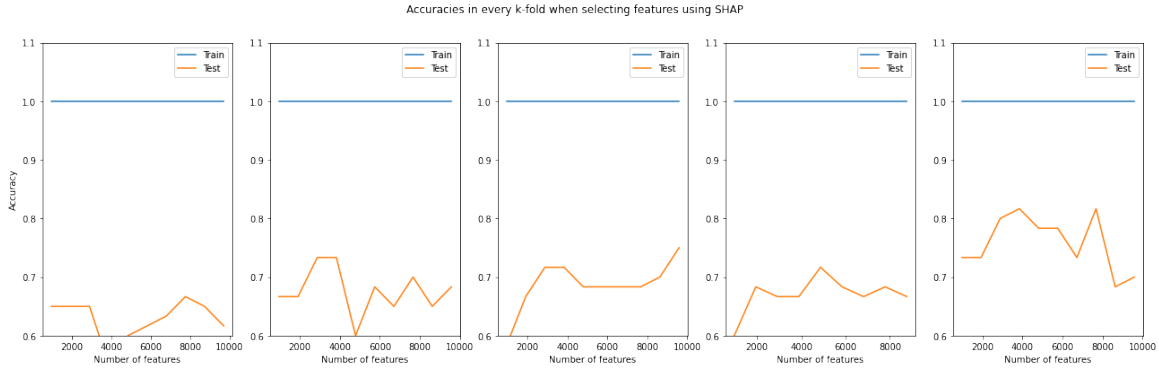

Figure 8.24: Accuracy per subset of relevant Features for AdaBoost using unigrams

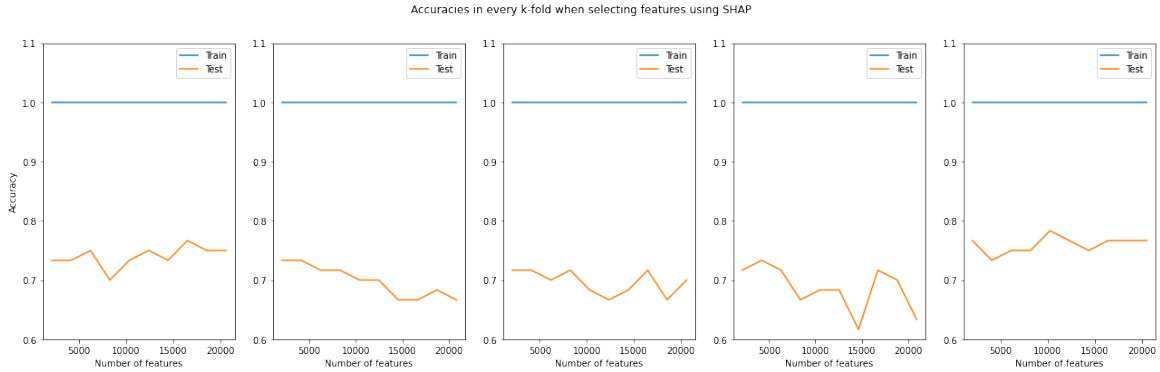

Figure 8.25: Accuracy per subset of relevant Features for AdaBoost using bigrams

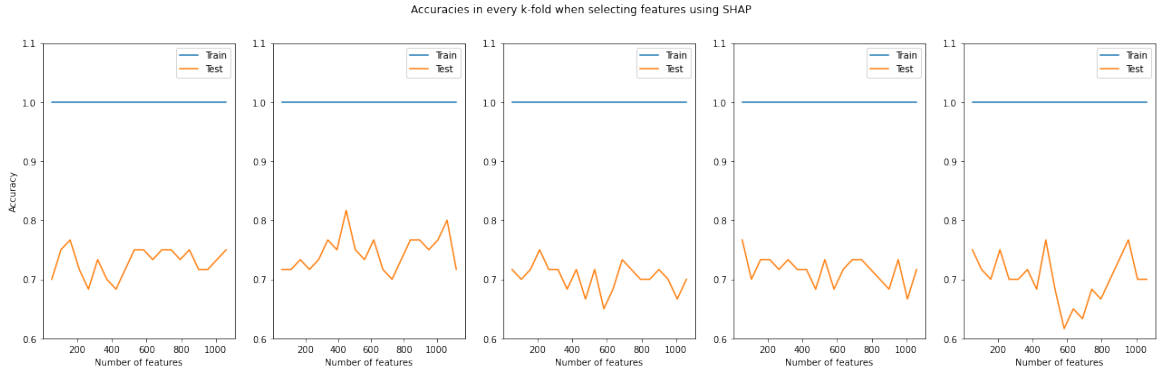

Figure 8.26: Accuracy per subset of relevant Features for AdaBoost using skipgrams

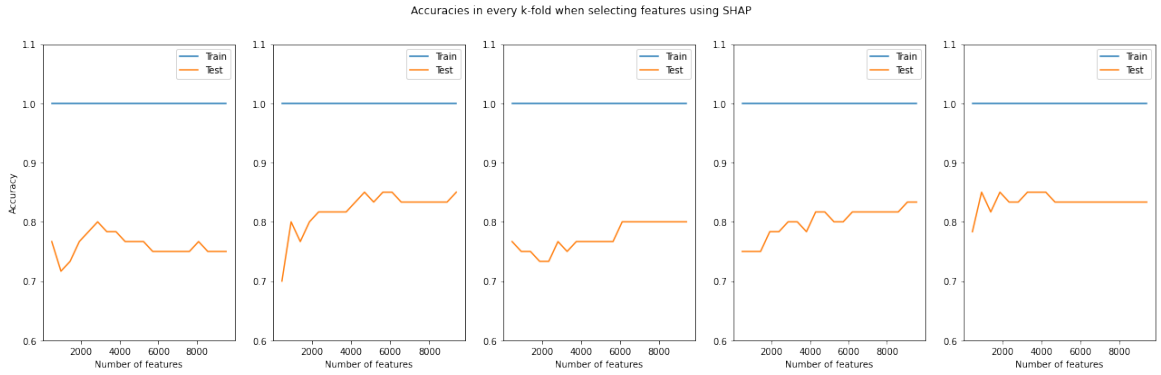

Figure 8.27: Accuracy per subset of relevant Features for Logistic Regression using unigrams

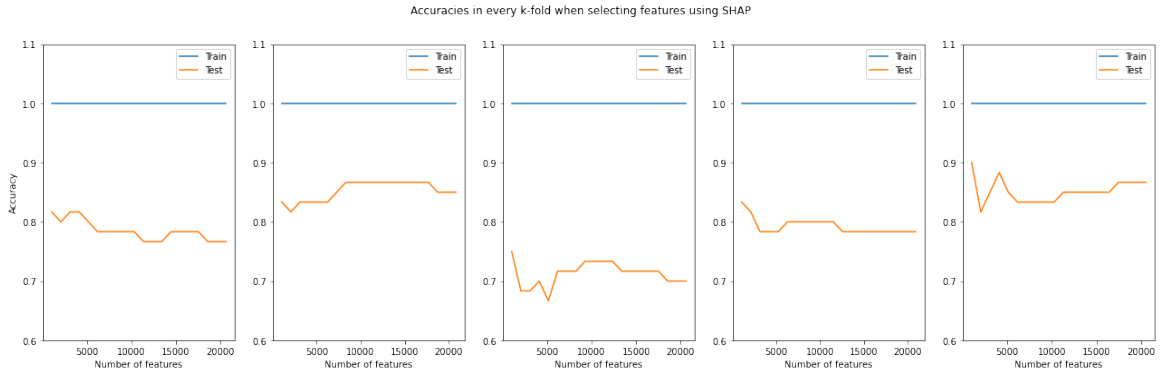

Figure 8.28: Accuracy per subset of relevant Features for Logistic Regression using bigrams

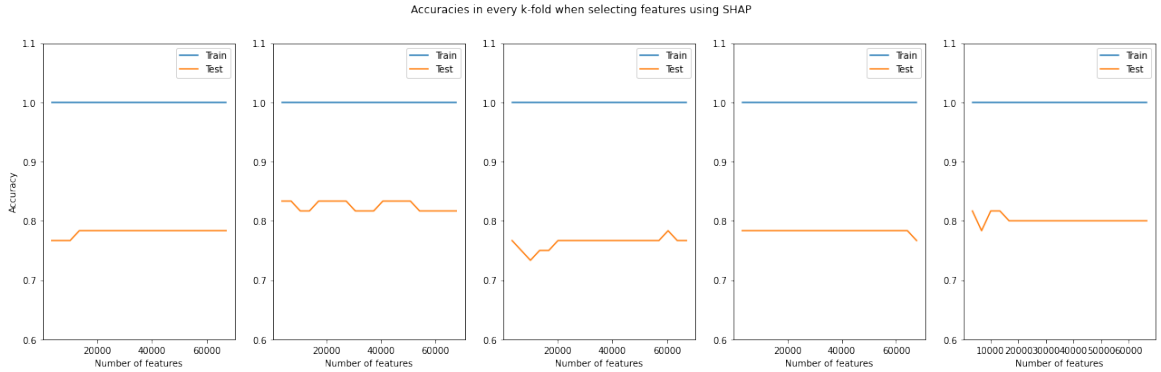

Figure 8.29: Accuracy per subset of relevant Features for Logistic Regression using skipgrams

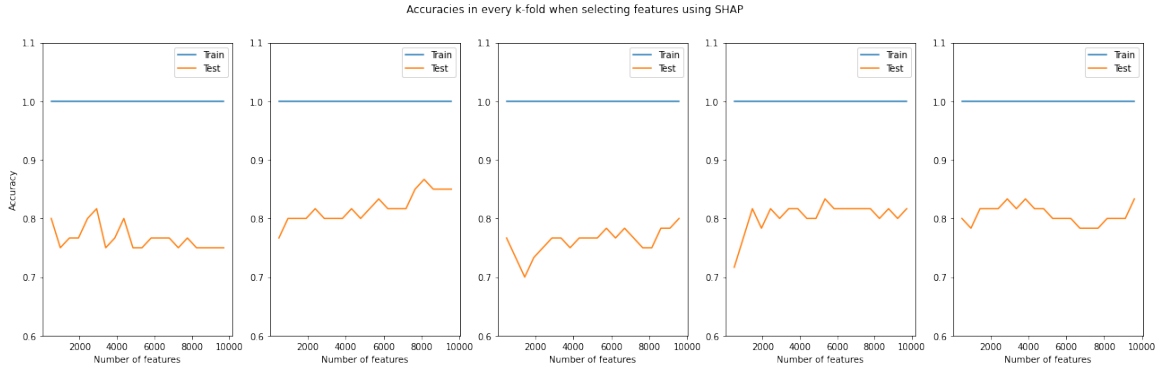

Figure 8.30: Accuracy per subset of relevant Features for Support Vector Machine using unigrams

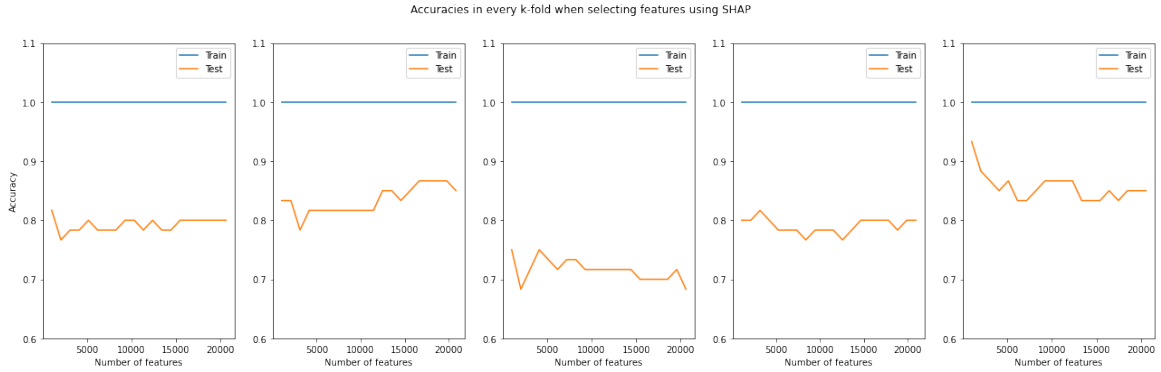

Figure 8.31: Accuracy per subset of relevant Features for Support Vector Machine using bigrams

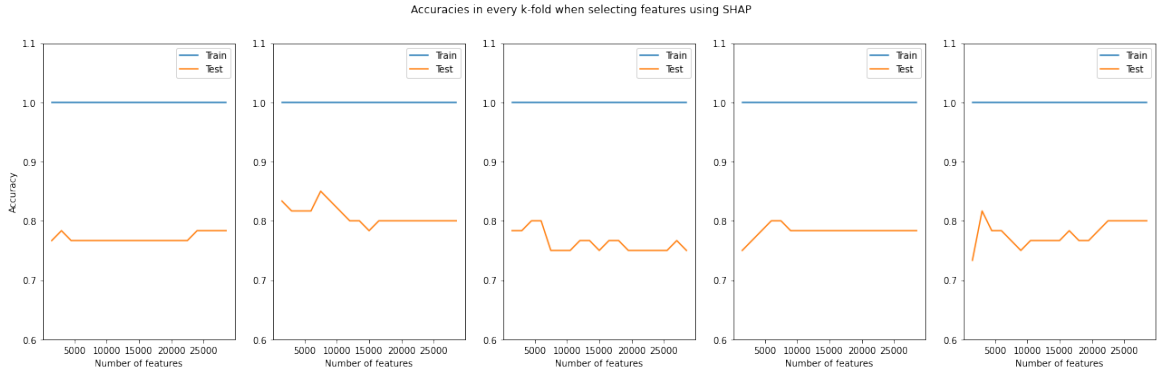

Figure 8.32: Accuracy per subset of relevant Features for Support Vector Machine using skipgrams

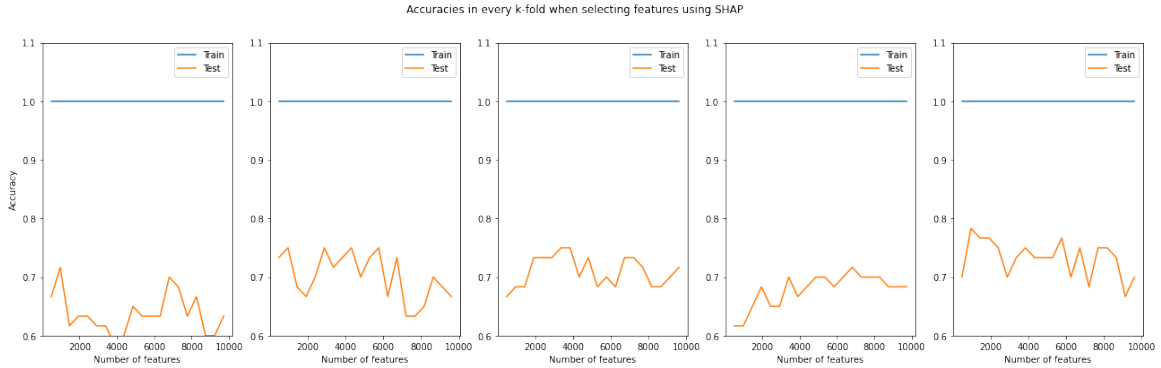

Figure 8.33: Accuracy per subset of relevant Features for Random Forest using unigrams

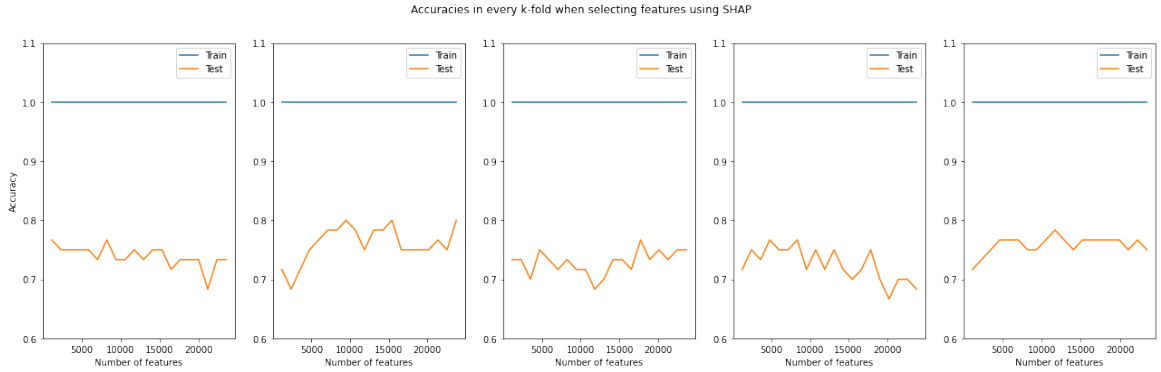

Figure 8.34: Accuracy per subset of relevant Features for Random Forest using bigrams

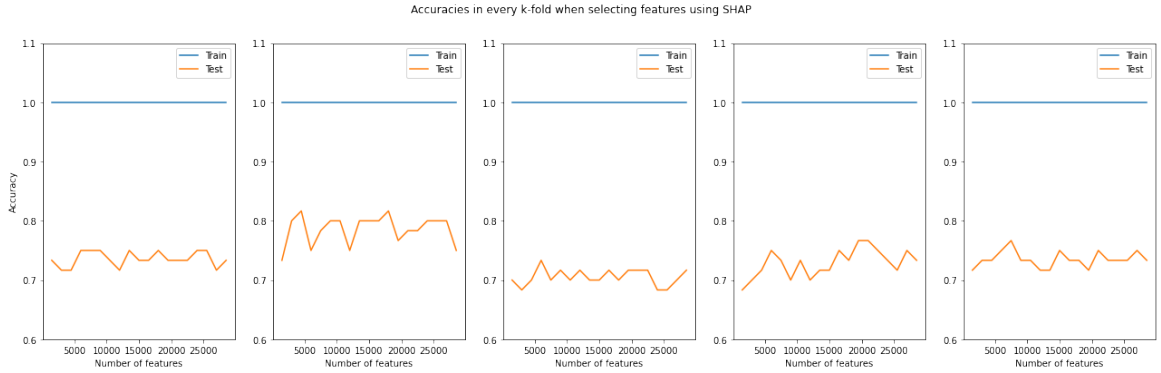

Figure 8.35: Accuracy per subset of relevant Features for Random Forest using skipgrams

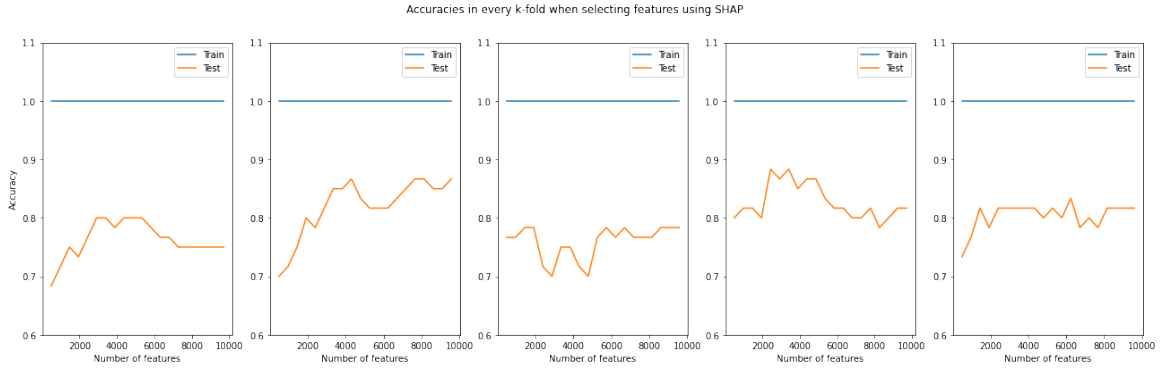

Figure 8.36: Accuracy per subset of relevant Features for Multilayer Perceptron using unigrams

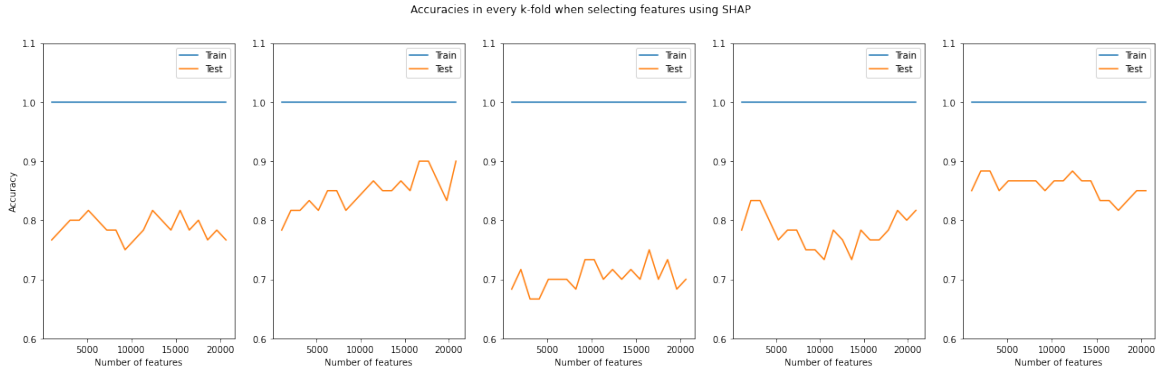

Figure 8.37: Accuracy per subset of relevant Features for Multilayer Perceptron using bigrams

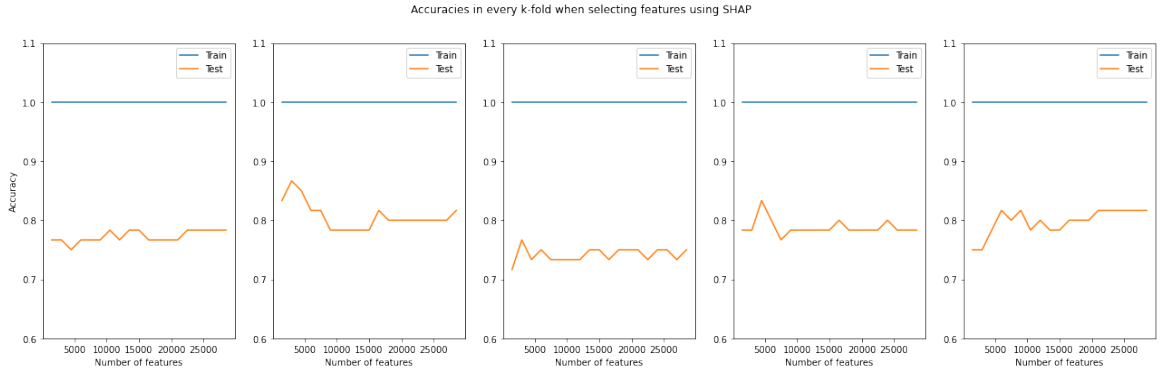

Figure 8.38: Accuracy per subset of relevant Features for Multilayer Perceptron using skipgrams

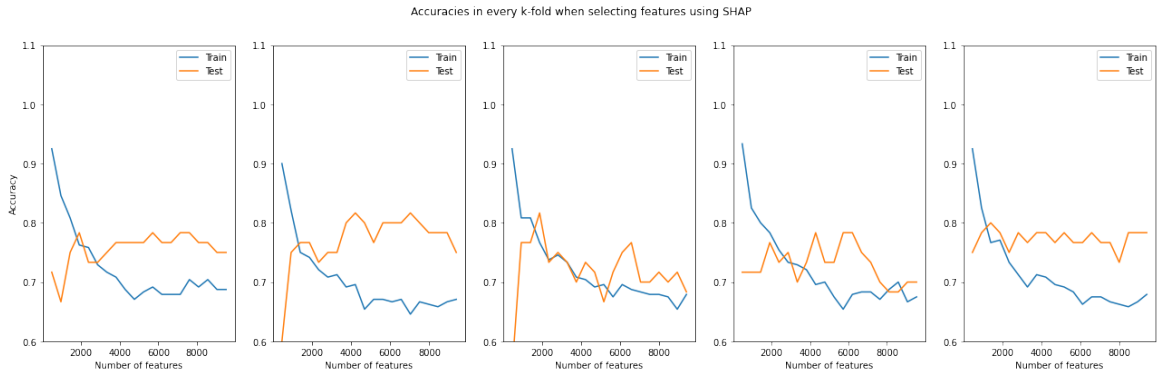

Figure 8.39: F1-Score per subset of relevant Features for Linear Discriminant Analysis using unigrams

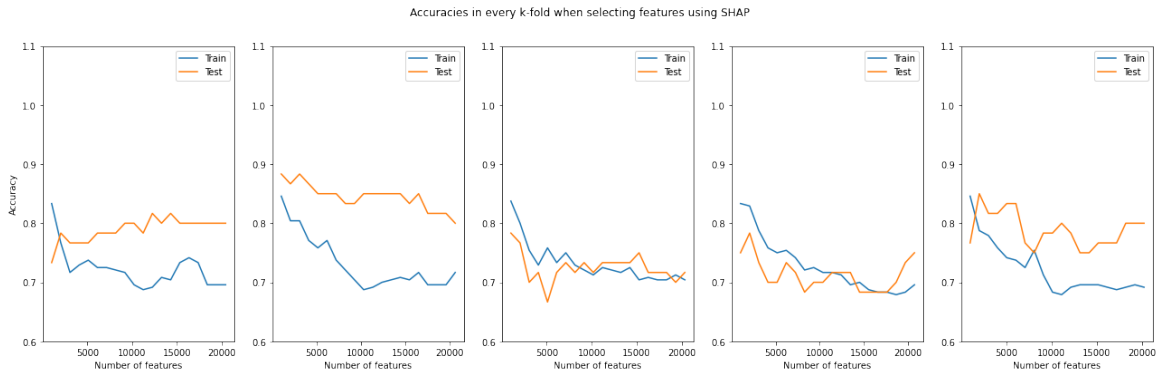

Figure 8.40: F1-Score per subset of relevant Features for Linear Discriminant Analysis using bigrams

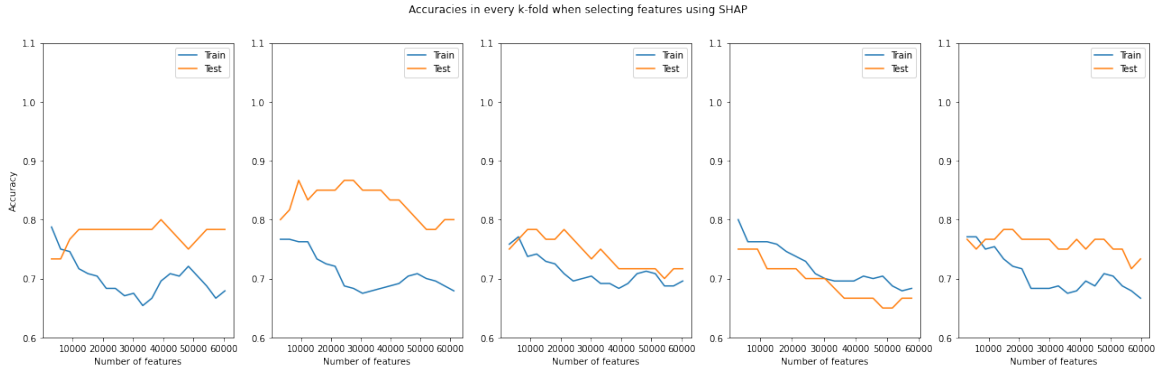

Figure 8.41: F1-Score per subset of relevant Features for Linear Discriminant Analysis using skipgrams

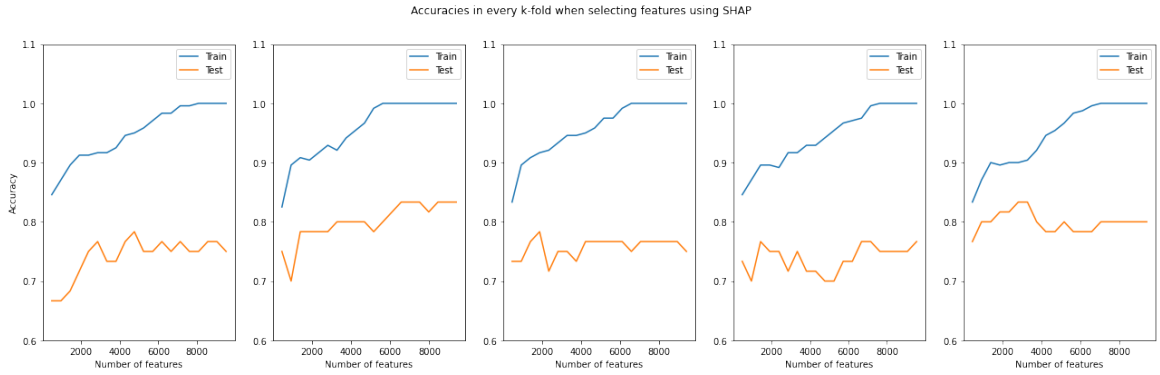

Figure 8.42: F1-Score per subset of relevant Features for Naive Bayes using unigrams

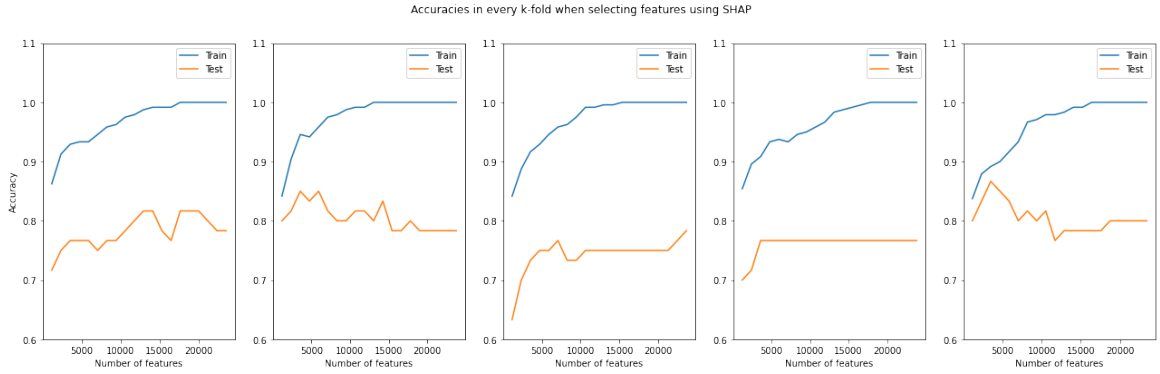

Figure 8.43: F1-Score per subset of relevant Features for Naive Bayes using bigrams

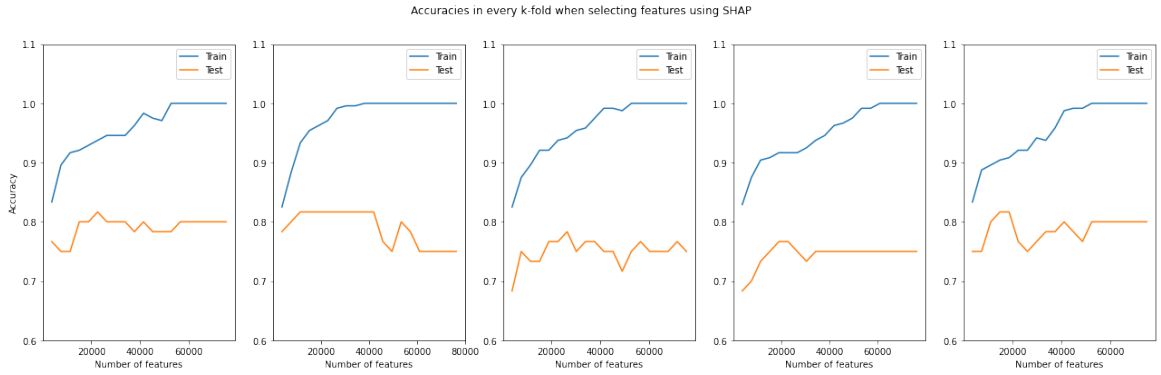

Figure 8.44: F1-Score per subset of relevant Features for Naive Bayes using skipgrams

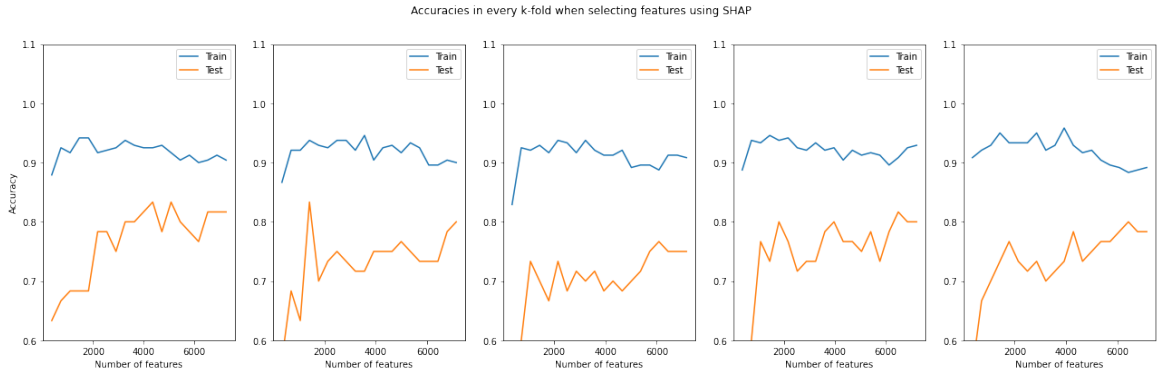

Figure 8.45: F1-Score per subset of relevant Features for K-Nearest Neighbors using unigrams

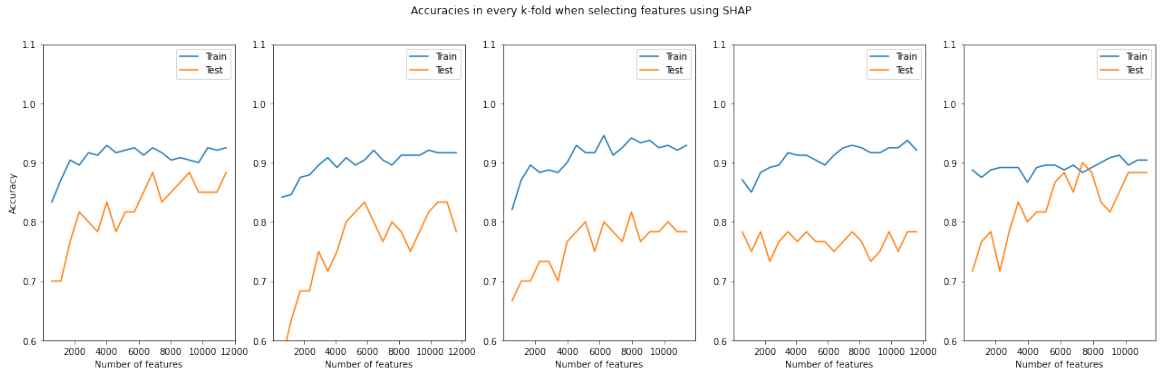

Figure 8.46: F1-Score per subset of relevant Features for K-Nearest Neighbors using bigrams

8.3 Feature Selection Plots for FakeDes@Iberlef Corpus

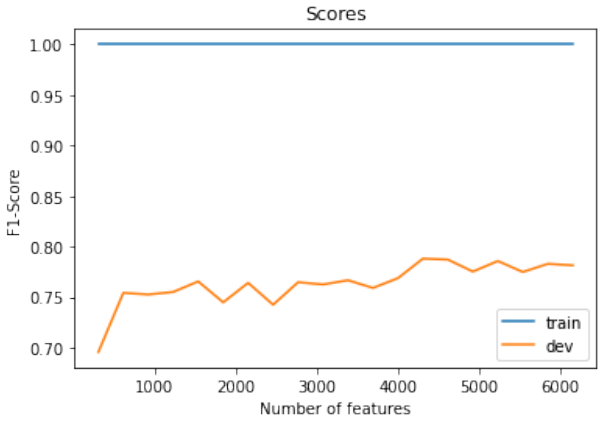

Figure 8.47: F1-Score per subset of relevant Features for AdaBoost using unigrams

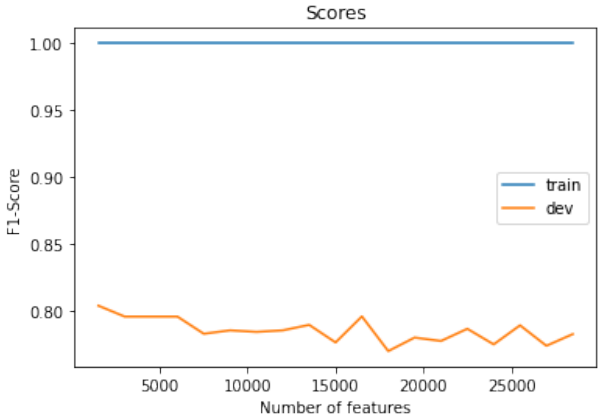

Figure 8.48: F1-Score per subset of relevant Features for AdaBoost using bigrams

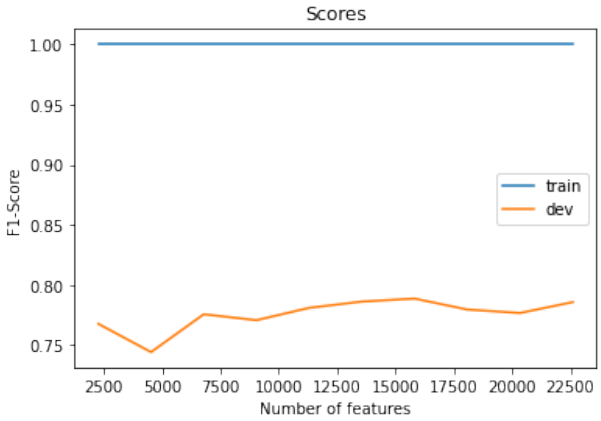

Figure 8.49: F1-Score per subset of relevant Features for AdaBoost using skipgrams

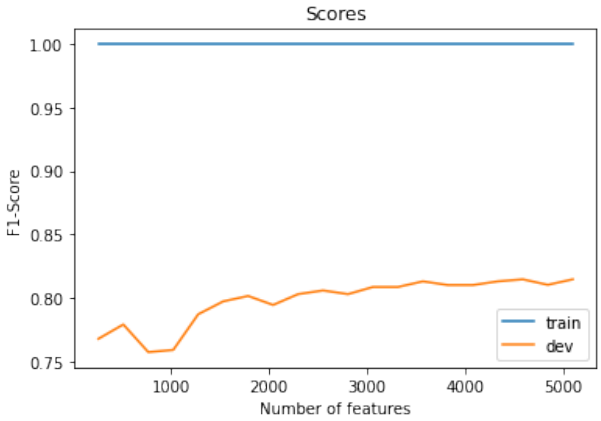

Figure 8.50: F1-Score per subset of relevant Features for Logistic Regression using unigrams

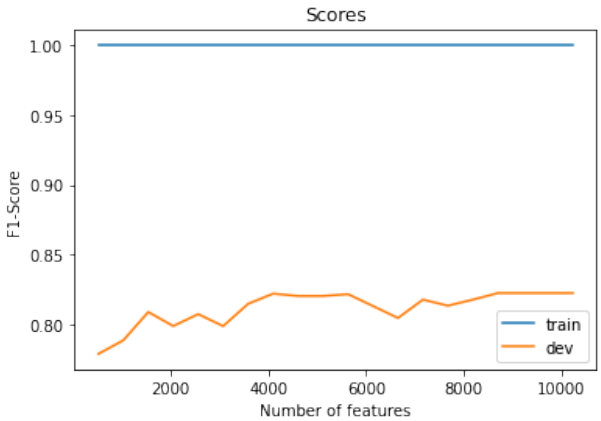

Figure 8.51: F1-Score per subset of relevant Features for Logistic Regression using bigrams

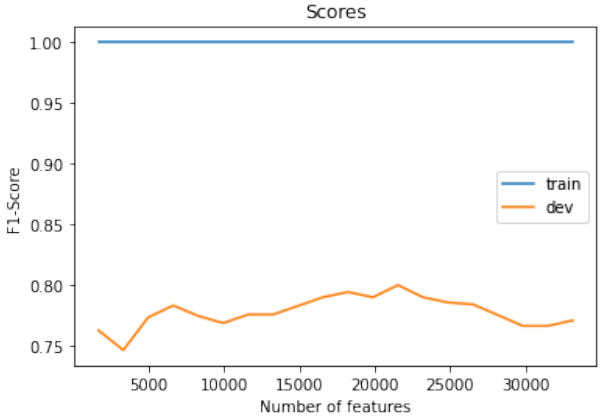

Figure 8.52: F1-Score per subset of relevant Features for Logistic Regression using skipgrams

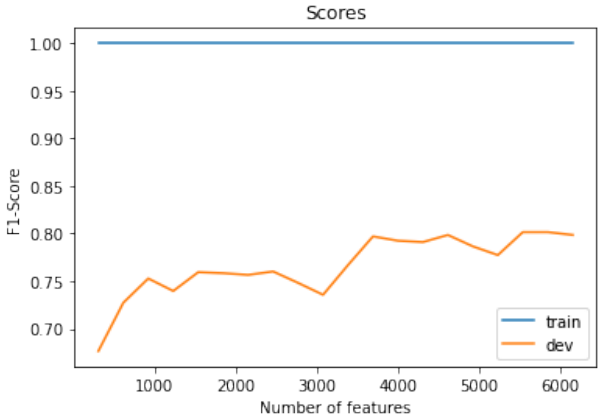

Figure 8.53: F1-Score per subset of relevant Features for Support Vector Machine using unigrams

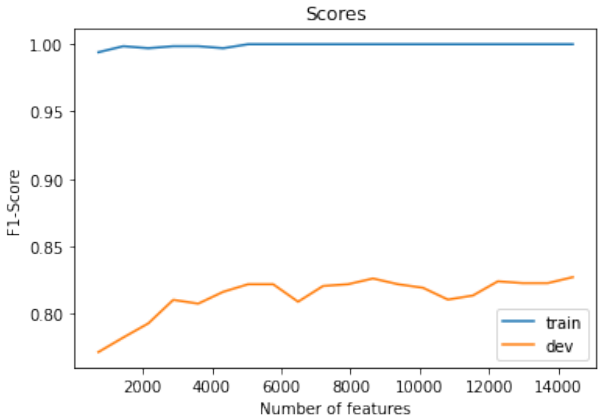

Figure 8.54: F1-Score per subset of relevant Features for Support Vector Machine using bigrams

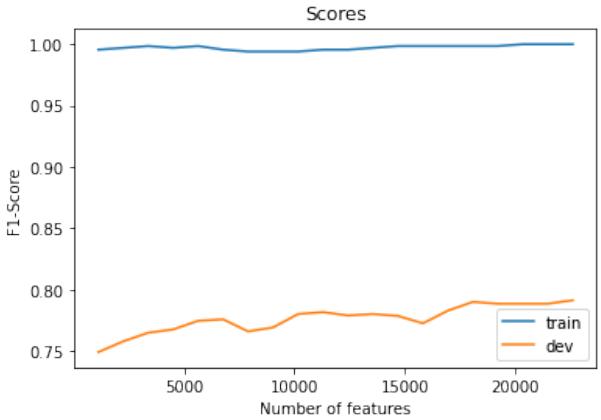

Figure 8.55: F1-Score per subset of relevant Features for Support Vector Machine using skipgrams

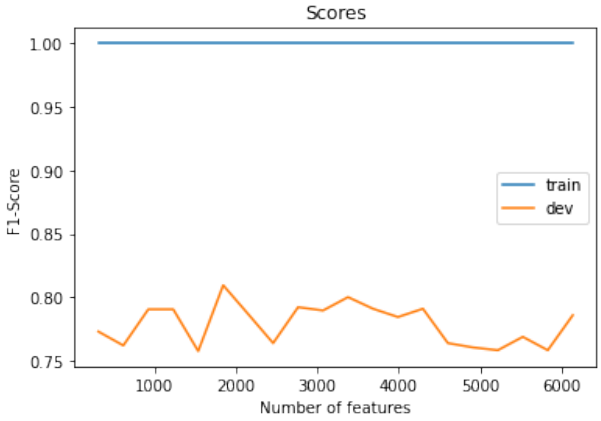

Figure 8.56: F1-Score per subset of relevant Features for Random Forest using unigrams

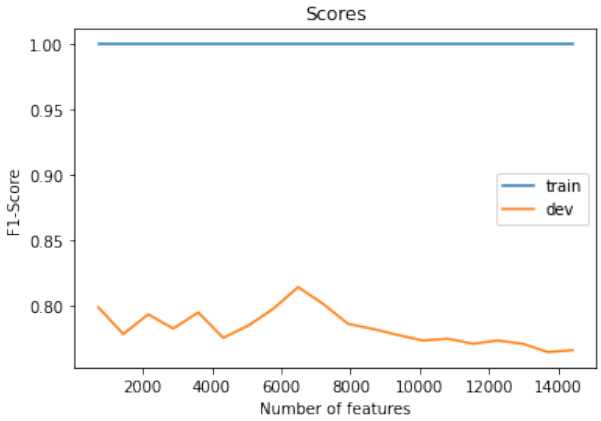

Figure 8.57: F1-Score per subset of relevant Features for Random Forest using bigrams

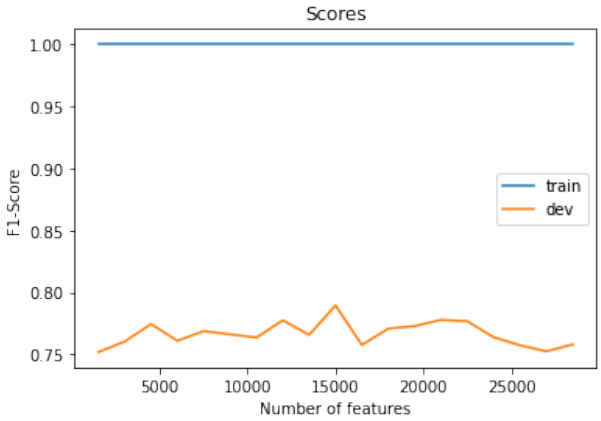

Figure 8.58: F1-Score per subset of relevant Features for Random Forest using skipgrams

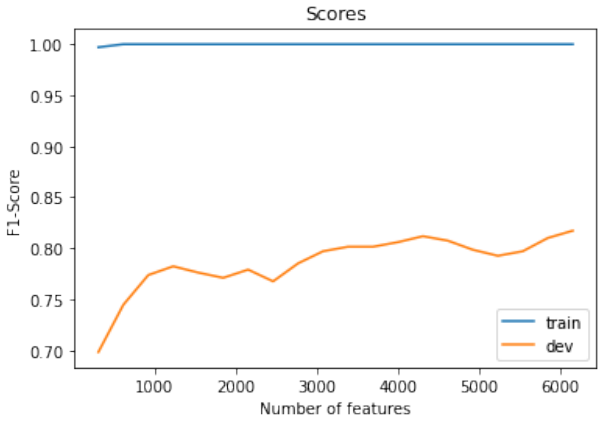

Figure 8.59: F1-Score per subset of relevant Features for Multilayer Perceptron using unigrams

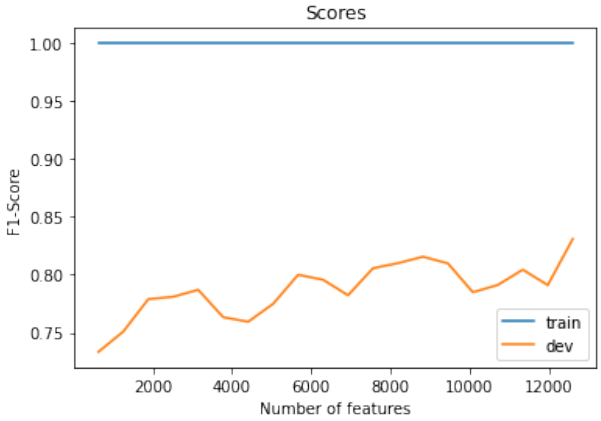

Figure 8.60: F1-Score per subset of relevant Features for Multilayer Perceptron using bigrams

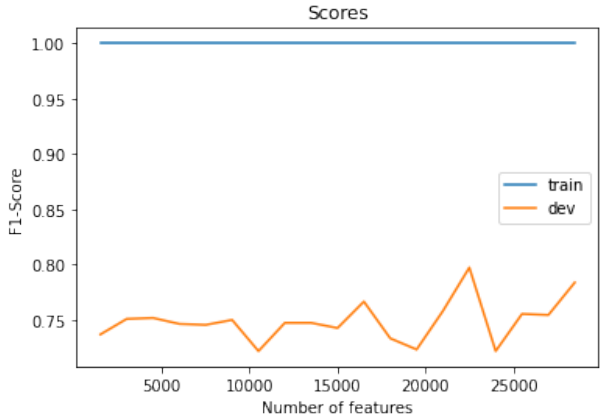

Figure 8.61: F1-Score per subset of relevant Features for Multilayer Perceptron using skipgrams

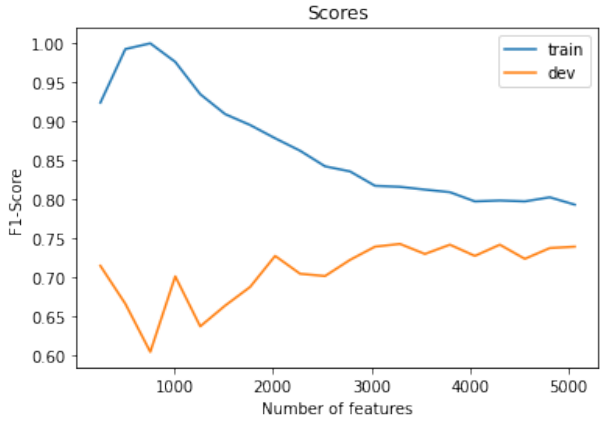

Figure 8.62: F1-Score per subset of relevant Features for Linear Discriminant Analysis using unigrams

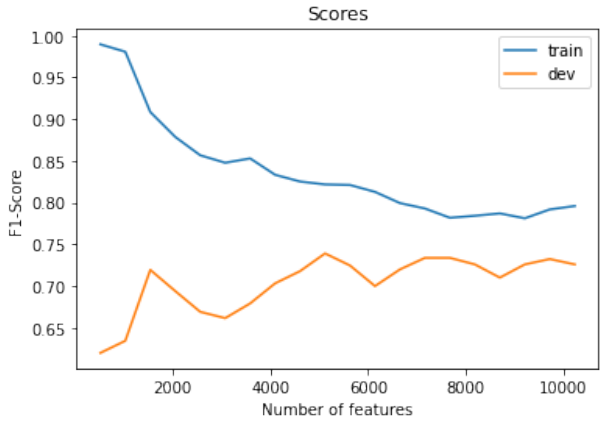

Figure 8.63: F1-Score per subset of relevant Features for Linear Discriminant Analysis using bigrams

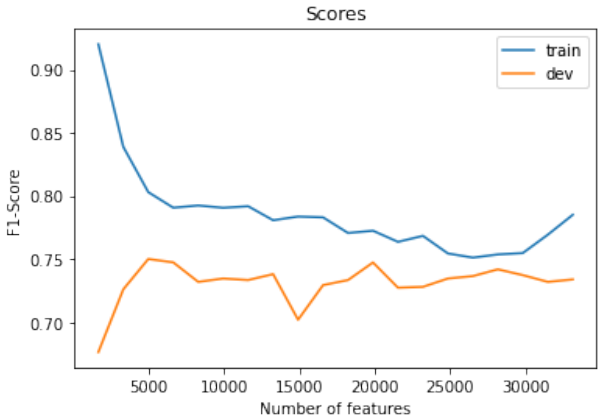

Figure 8.64: F1-Score per subset of relevant Features for Linear Discriminant Analysis using skipgrams

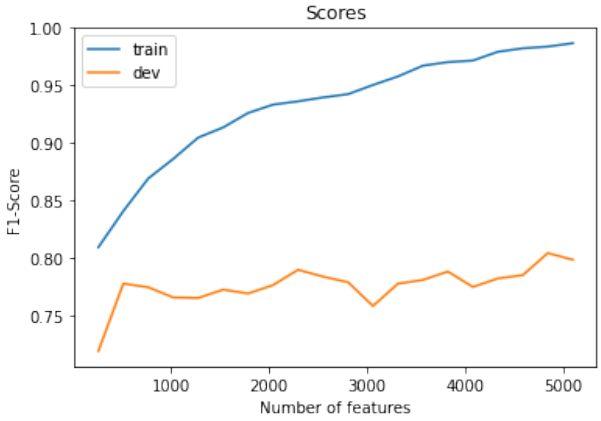

Figure 8.65: F1-Score per subset of relevant Features for Naive Bayes using unigrams

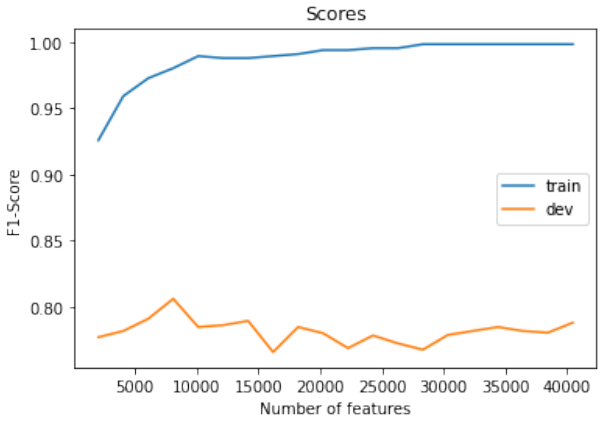

Figure 8.66: F1-Score per subset of relevant Features for Naive Bayes using bigrams

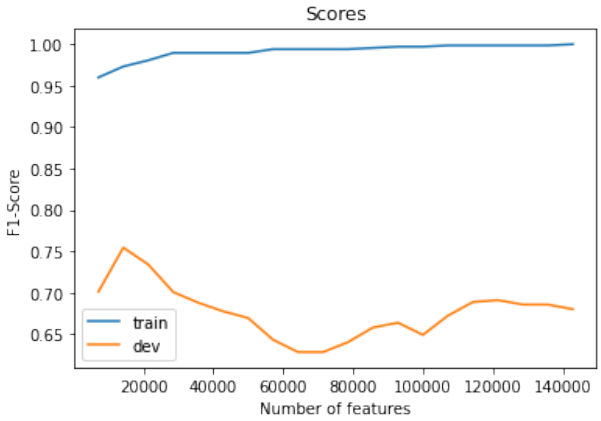

Figure 8.67: F1-Score per subset of relevant Features for Naive Bayes using skipgrams

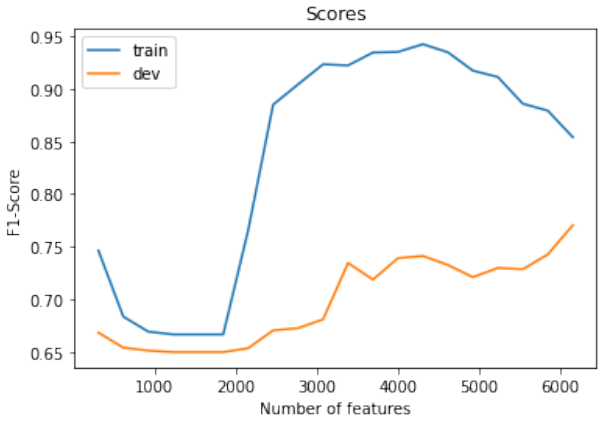

Figure 8.68: F1-Score per subset of relevant Features for K-Nearest Neighbors using unigrams

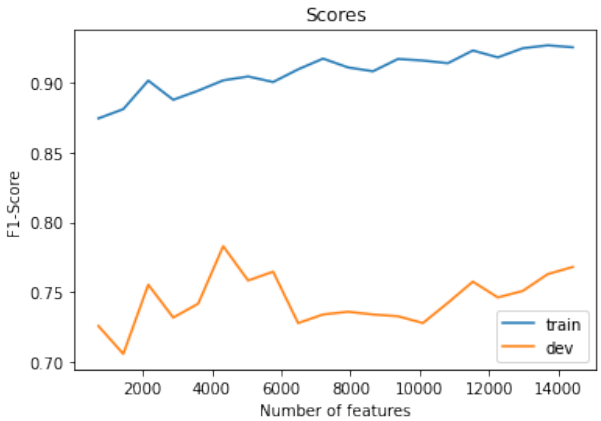

Figure 8.69: F1-Score per subset of relevant Features for K-Nearest Neighbors using bigrams

9 Appendix B

9.1 Intepretation SHAP Plots for English PAN@CLEF Classifiers

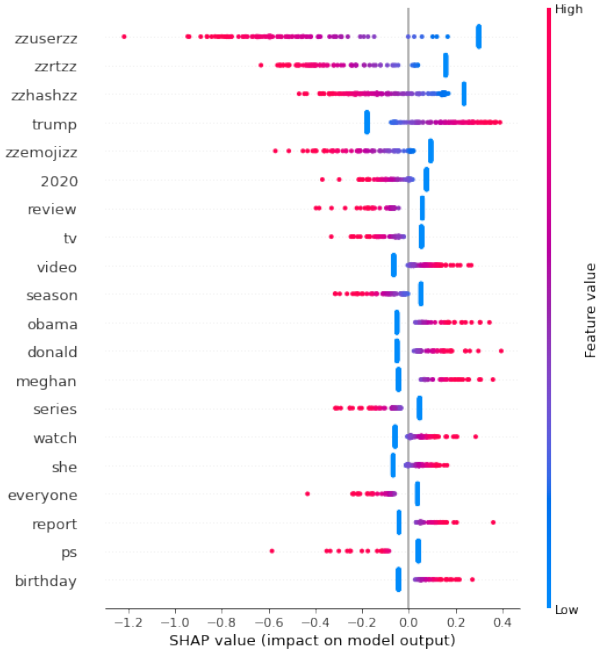

Figure 9.1: Top 20 of Most Important Features for Logistic Regression using unigrams

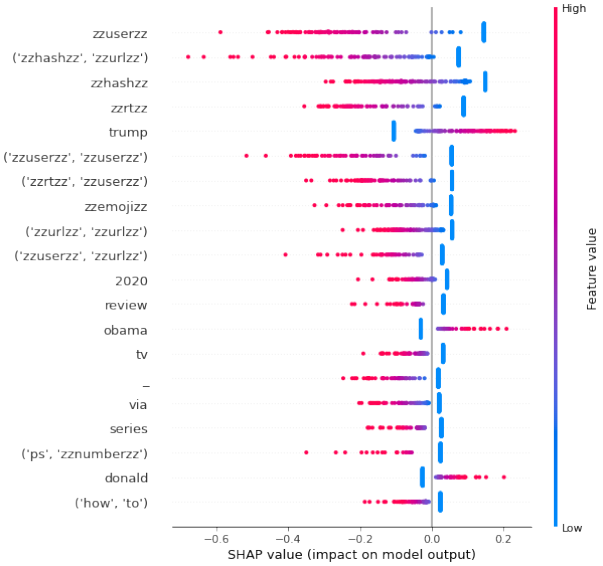

Figure 9.2: Top 20 of Most Important Features for Logistic Regression using uni-bigrams

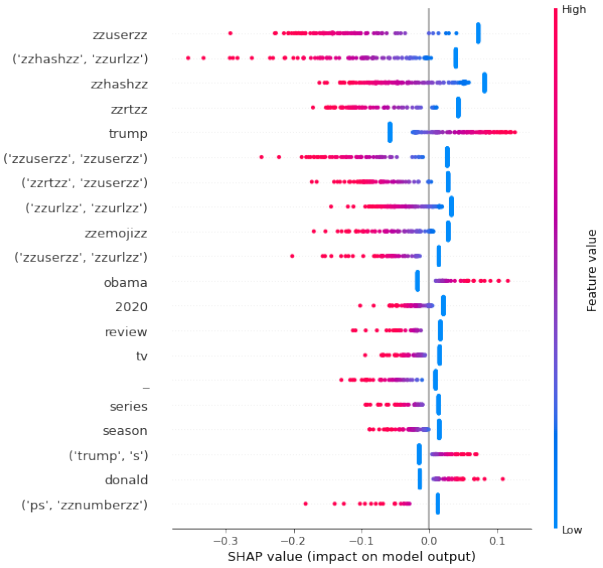

Figure 9.3: Top 20 of Most Important Features for Logistic Regression using uni-bi-skipgrams

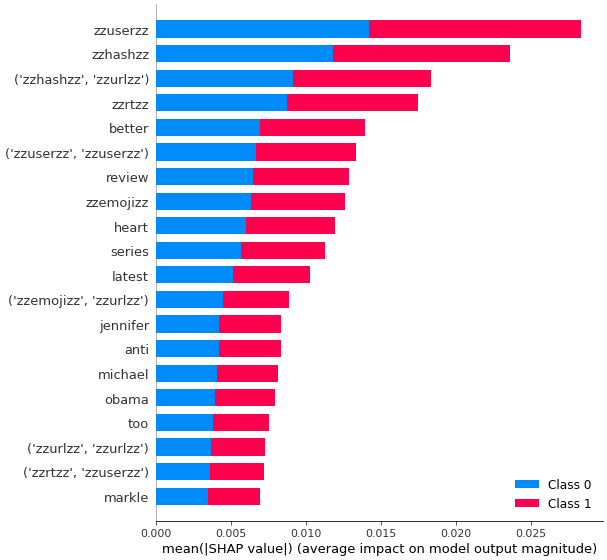

Figure 9.4: Top 20 of Most Important Features for Support Vector Machine Classifier using uni-bigrams

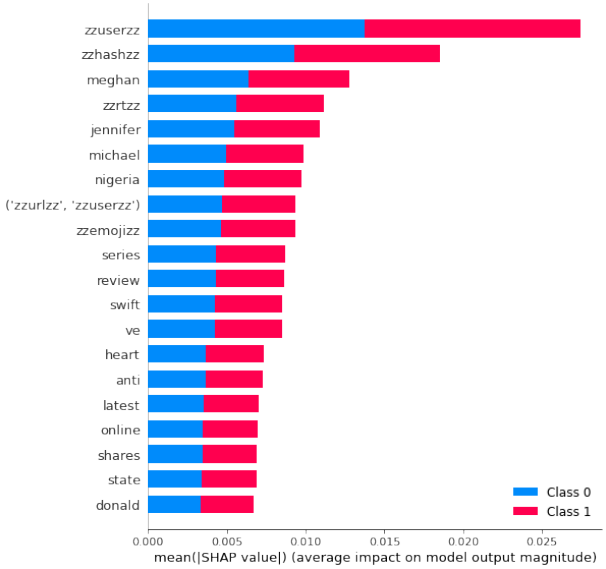

Figure 9.5: Top 20 of Most Important Features for Support Vector Machine Classifier using uni-skipgrams

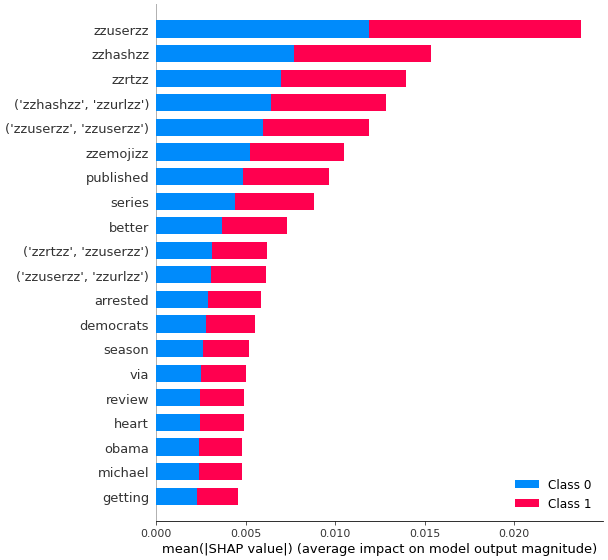

Figure 9.6: Top 20 of Most Important Features for Support Vector Machine Classifier using uni-bi-skipgrams

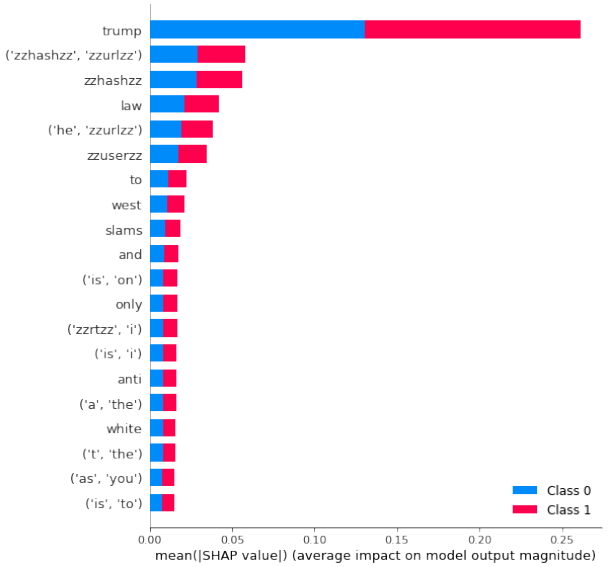

Figure 9.7: Top 20 of Most Important Features for AdaBoost using uni-skipgrams

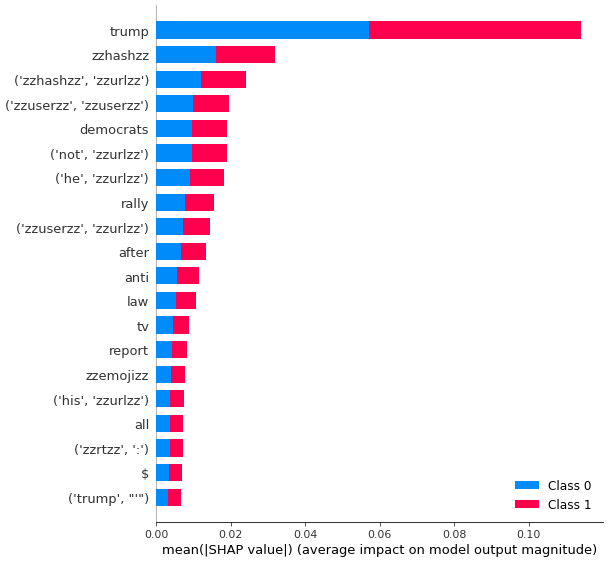

Figure 9.8: Top 20 of Most Important Features for Random Forest using uni-skipgrams

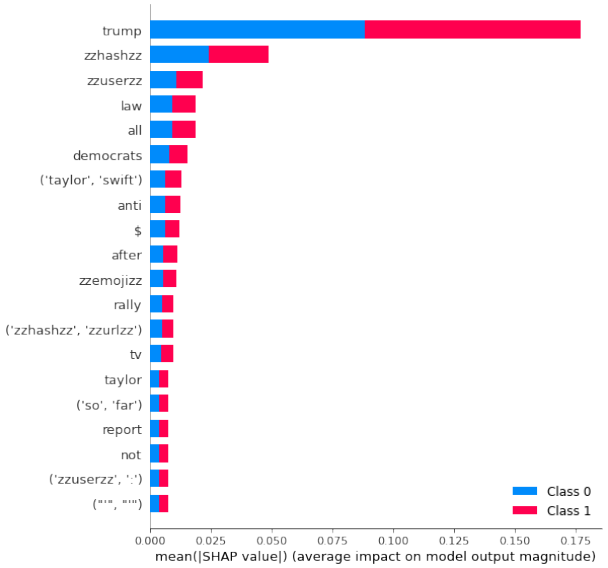

Figure 9.9: Top 20 of Most Important Features for Random Forest using uni-bigrams

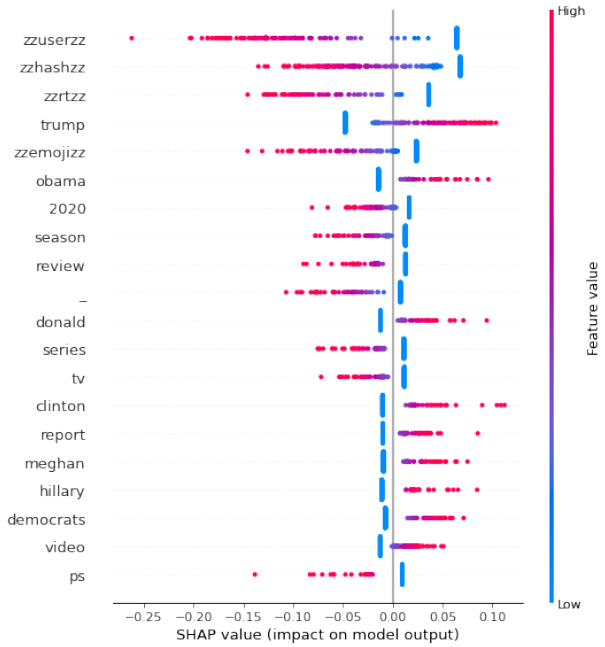

Figure 9.10: Top 20 of Most Important Features for Logistic Regression using uni-skipgrams

9.2 Intepretation SHAP Plots for Spanish PAN@CLEF Classifiers

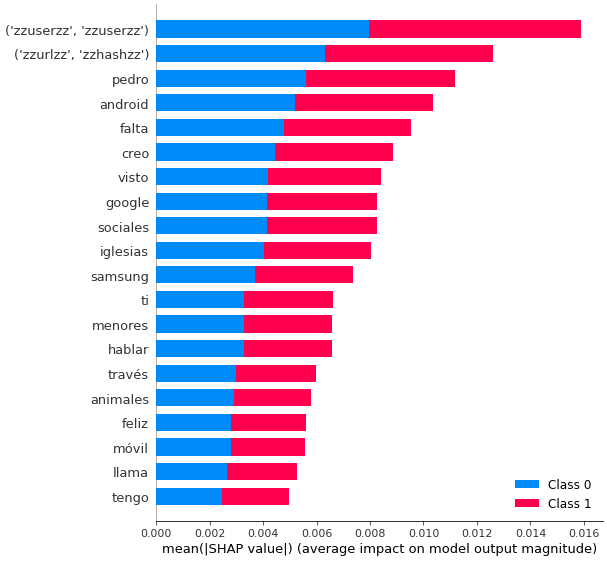

Figure 9.11: Top 20 of Most Important Features for Support Vector Machine Classifier using uni-bigrams

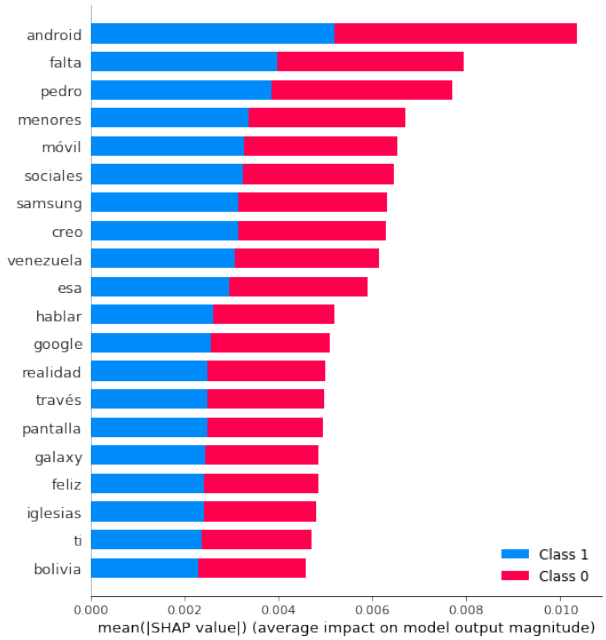

Figure 9.12: Top 20 of Most Important Features for Support Vector Machine Classifier using uni-skipgrams

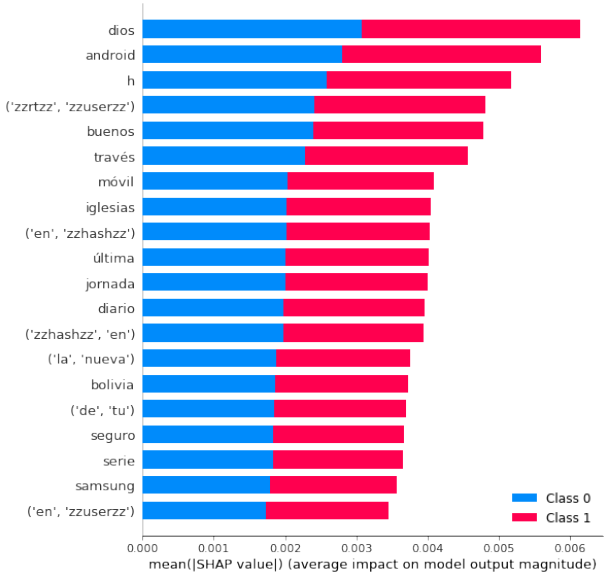

Figure 9.13: Top 20 of Most Important Features for MultiLayer Perceptron Classifier using uni-bigrams

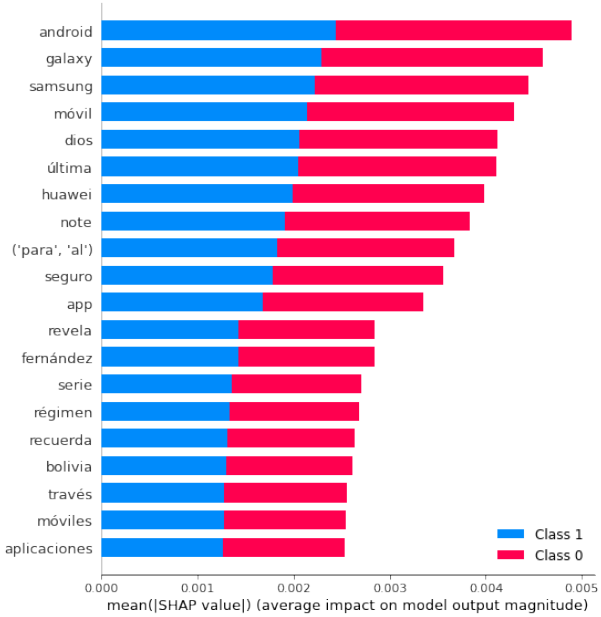

Figure 9.14: Top 20 of Most Important Features for MultiLayer Perceptron Classifier using uni-skipgrams

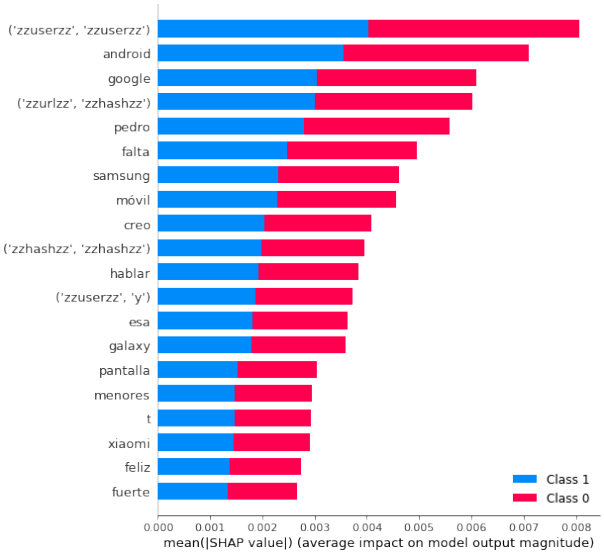

Figure 9.15: Top 20 of Most Important Features for Support Vector Machine Classifier using uni-bi-skipgrams

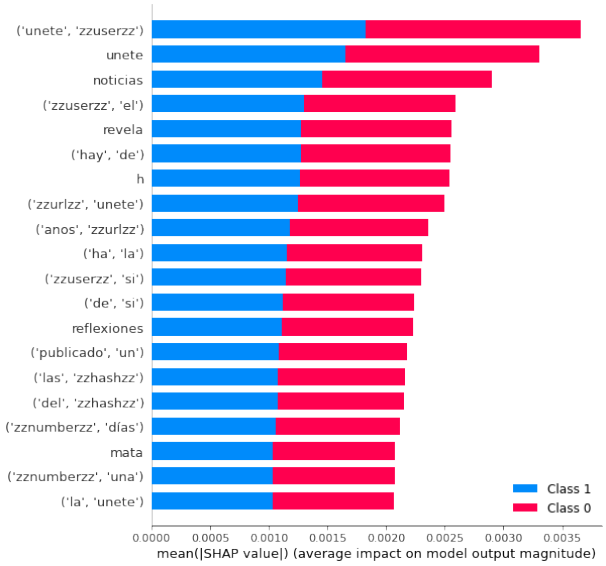

Figure 9.16: Top 20 of Most Important Features for MultiLayer Perceptron Classifier using uni-bi-skipgrams

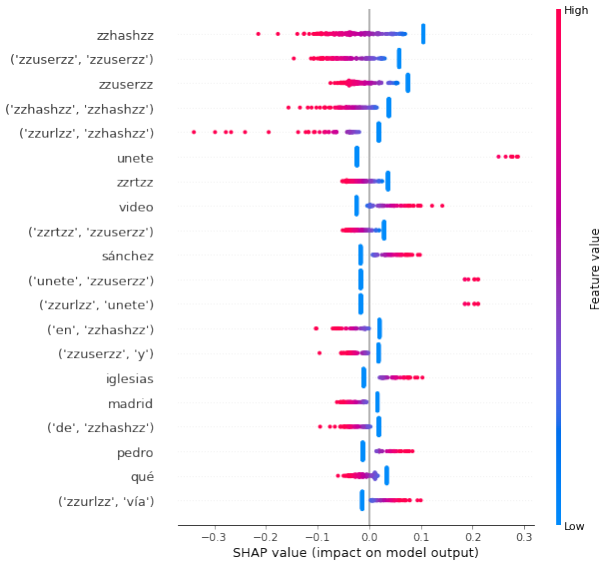

Figure 9.17: Top 20 of Most Important Features for Logistic Regression using uni-bigrams

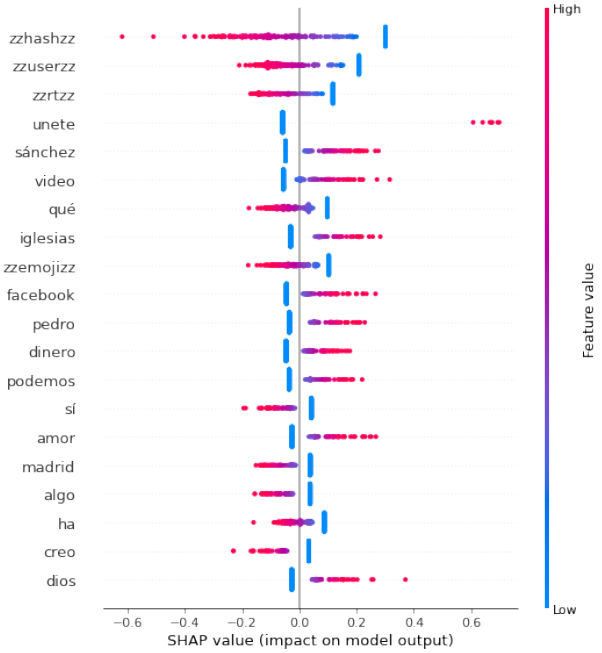

Figure 9.18: Top 20 of Most Important Features for Logistic Regression using unigrams

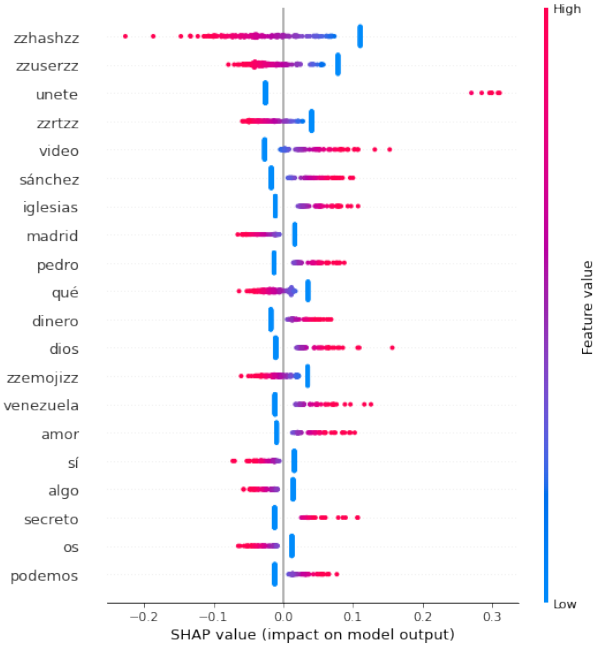

Figure 9.19: Top 20 of Most Important Features for Logistic Regression using uni-skipgrams

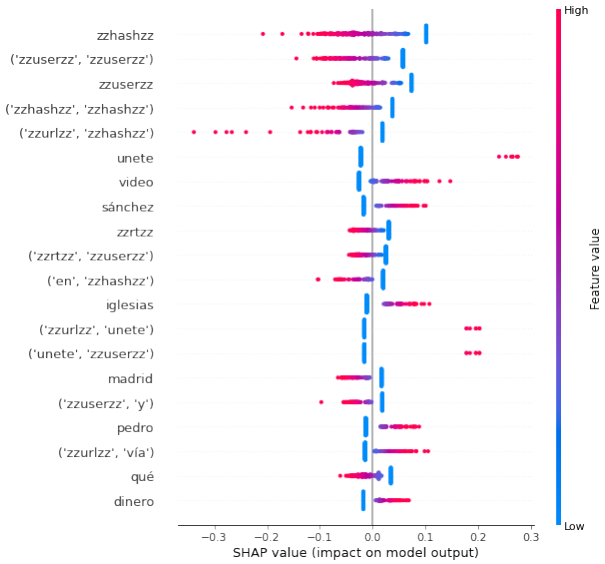

Figure 9.20: Top 20 of Most Important Features for Logistic Regression using uni-bi-skipgrams

9.3 Intepretation SHAP Plots for Spanish FakeDes@Iberlef Classifiers

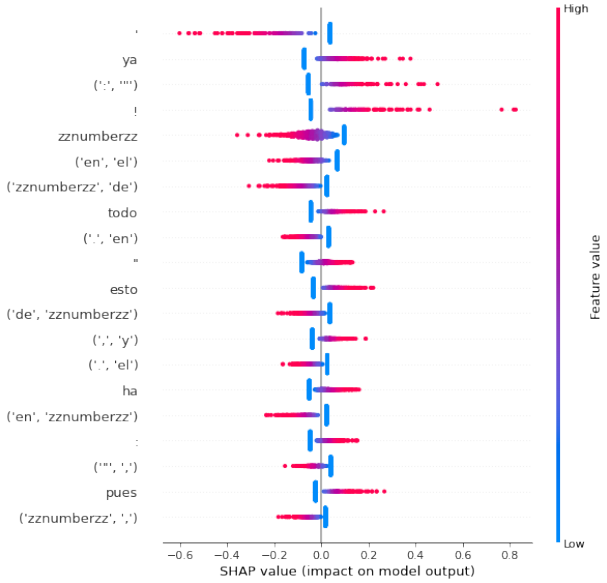

Figure 9.21: Top 20 of Most Important Features for Logistic Regression using uni-bi-skipgrams

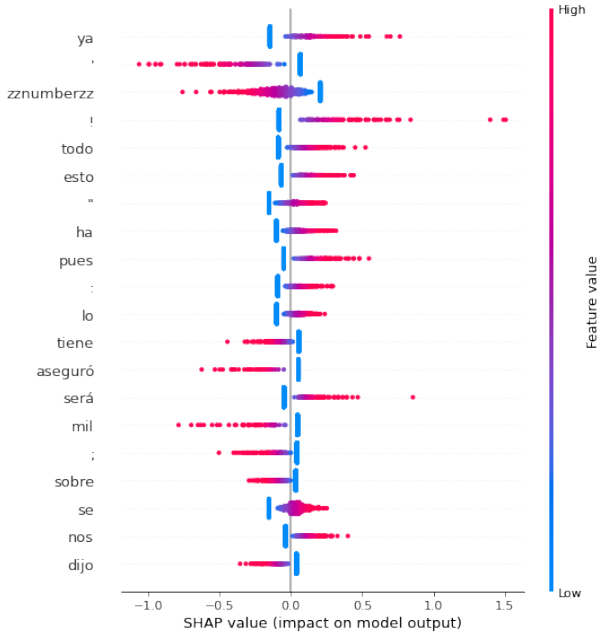

Figure 9.22: Top 20 of Most Important Features for Logistic Regression using uni-skipgrams

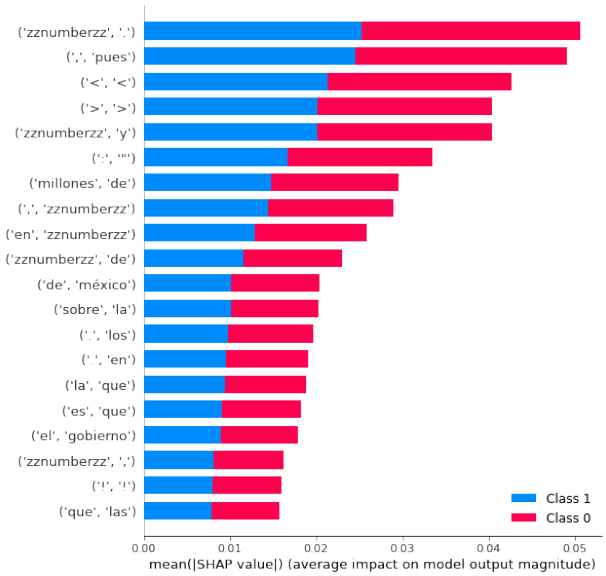

Figure 9.23: Top 20 of Most Important Features for Support Vector Machine using bigrams

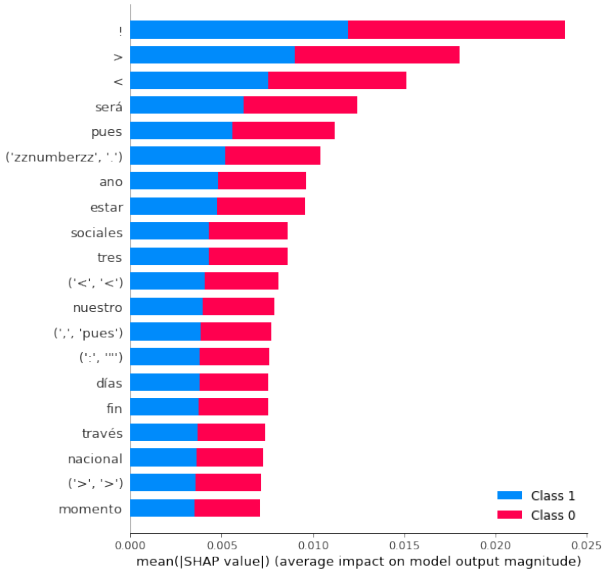

Figure 9.24: Top 20 of Most Important Features for Support Vector Machine using uni-bi-skipgrams

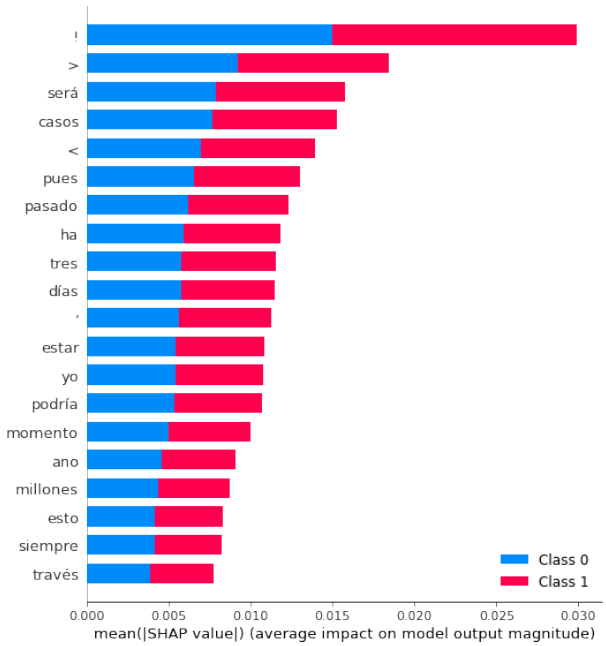

Figure 9.25: Top 20 of Most Important Features for Support Vector Machine using uni-skipgrams

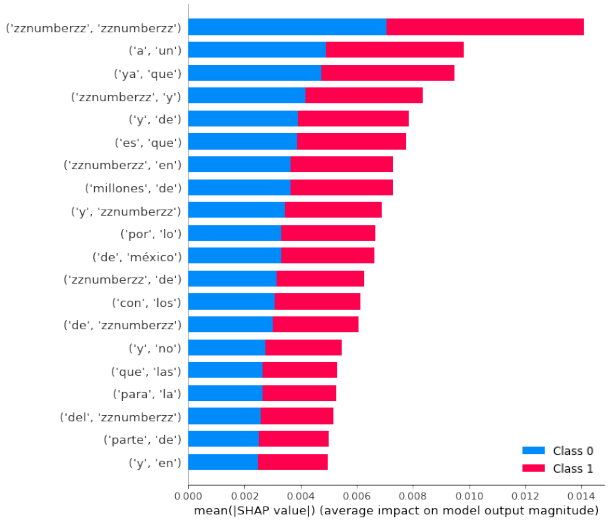

Figure 9.26: Top 20 of Most Important Features for Multi-Layer Perceptron using bigrams

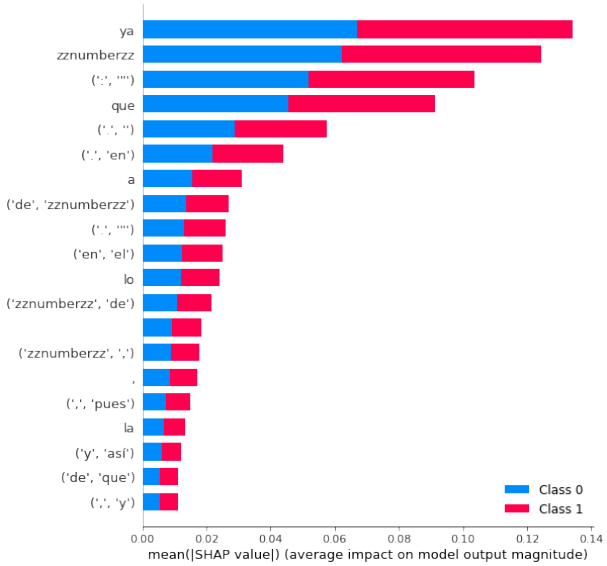

Figure 9.27: Top 20 of Most Important Features for AdaBoost using uni-bigrams

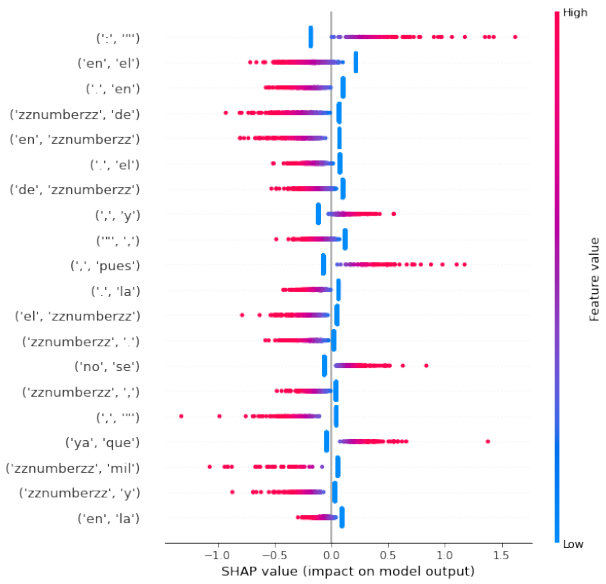

Figure 9.28: Top 20 of Most Important Features for Logistic Regression using bigrams

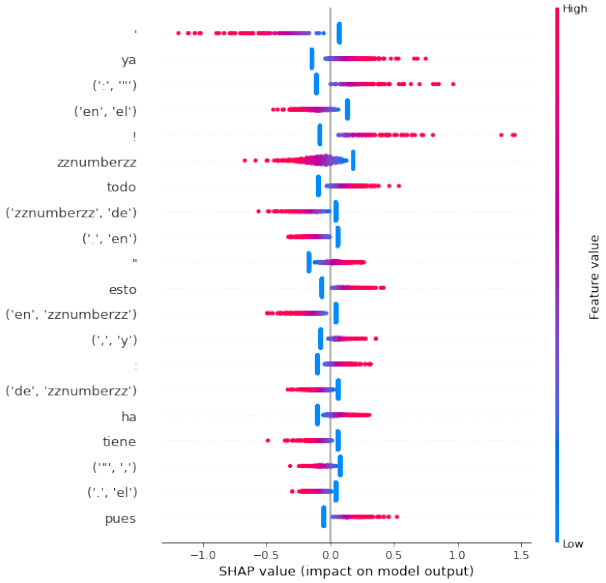

Figure 9.29: Top 20 of Most Important Features for Logistic Regression using uni-bigrams

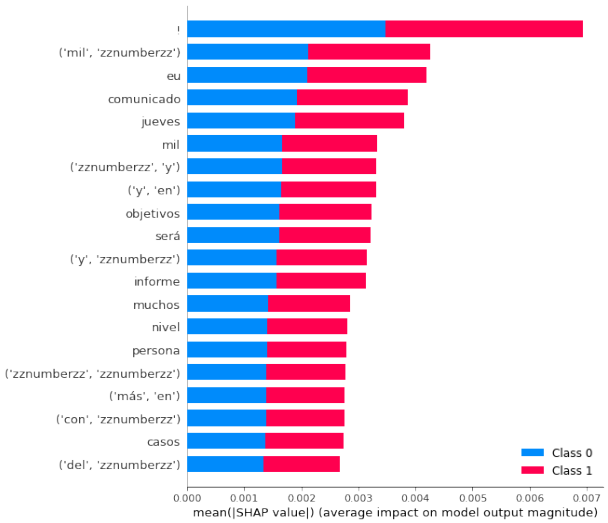

Figure 9.30: Top 20 of Most Important Features for Multi-Layer Perceptron using uni-bi-skipgrams