INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Análisis de explicabilidad de redes neuronales en tareas de procesamiento de texto

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

Ing. José Ángel Avelar Barragán

DIRECTORES DE TESIS:

Dr. Rolando Méchaca Méndez
Dr. Grigori Sidorov

Estados Unidos Mexicanos
Ciudad de México
2022

SIP-13
REP 2017

INSTITUTO POLITÉCNICO NACIONAL
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REGISTRO DE TEMA DE TESIS
Y DESIGNACIÓN DE DIRECTOR DE TESIS

Ciudad de México, a 10 de julio del 2020

El Colegio de Profesores de Posgrado del **Centro de Investigación en Computación** en su Sesión
(Unidad Académica)

Ordinaria No. 5 celebrada el día 29 del mes mayo de 2020, conoció la solicitud presentada por el (la) alumno (a):

Apellido Paterno:	AVELAR	Apellido Materno:	BARRAGÁN	Nombre (s):	JOSÉ ÁNGEL

Número de registro: A 2 0 0 3 6 1

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

Referente al registro de su tema de tesis; acordando lo siguiente:

1.- Se designa al aspirante el tema de tesis titulado:

"Análisis de explicabilidad de redes neuronales en tareas de procesamiento de texto"

Objetivo general del trabajo de tesis:

Desarrollar y caracterizar experimentalmente un marco de trabajo para interpretar modelos entrenados aplicados al problema de clasificación de textos.

2.- Se designa como Directores de Tesis a los profesores:

Director: **Dr. Rolando Menchaca Méndez** 2° Director: **Dr. Grigori Sidorov**

No aplica: ☐

3.- El Trabajo de investigación base para el desarrollo de la tesis será elaborado por el alumno en:

Centro de Investigación en Computación

que cuenta con los recursos e infraestructura necesarios.

4.- El interesado deberá asistir a los seminarios desarrollados en el área de adscripción del trabajo desde la fecha en que se suscribe la presente, hasta la aprobación de la versión completa de la tesis por parte de la Comisión Revisora correspondiente.

Director de Tesis

Dr. Rolando Menchaca Méndez

2º Director de Tesis

Dr. Grigori Sidorov

Aspirante

C. José Ángel Avelar Barragán

Presidente del Colegio

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

Dr. Marco Antonio Moreno Ibarra

SIP-14
REP 2017

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 17:00 horas del día 02 del mes de junio del 2022 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Posgrado de: Centro de Investigación en Computación para examinar la tesis titulada:

"Análisis de explicabilidad de redes neuronales en tareas de procesamiento de texto"

del (la) alumno (a):

Apellido Paterno:	AVELAR	Apellido Materno:	BARRAGÁN	Nombre (s):	JOSÉ ÁNGEL

Número de registro: A 2 0 0 3 6 1

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 09 % 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** ☒ **SE CONSTITUYE UN POSIBLE PLAGIO.**

JUSTIFICACIÓN DE LA CONCLUSIÓN: *(Por ejemplo, el % de similitud se localiza en metodologías adecuadamente referidas a fuente original)*
El porcentaje de similitud se localiza en frases de uso común en documentos de tesis y textos científico.

****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** ☒ **SUSPENDER** ☐ **NO APROBAR** ☐ la tesis por **UNANIMIDAD** ☒ o **MAYORÍA** ☐ en virtud de los motivos siguientes:
Cumple con los requisitos académicos reglamentarios.

COMISIÓN REVISORA DE TESIS

Dr. Rolando Menchaca Méndez
Director de Tesis

Dr. Francisco Hiram Calvo Castro

Dr. Ricardo Menchaca Méndez

Dr. Grigori Sidorov
2° Director de Tesis

Dr. Rolando Quintero Téllez

Dr. Erik Zamora Gómez

INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
DIRECCIÓN

Dr. Francisco Hiram Calvo Castro
PRESIDENTE DEL COLEGIO DE PROFESORES

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE AUTORIZACIÓN DE USO DE OBRA PARA DIFUSIÓN

En la Ciudad de México el día 16 del mes de Junio del año 2022, el (la) que suscribe José Ángel Avelar Barragán alumno(a) del programa Maestría en Ciencias de la Computación (MCC) con número de registro A200361, adscrito(a) a Centro de Investigación en Computación (CIC) manifiesta que es autor(a) intelectual del presente trabajo de tesis bajo la dirección de Dr. Rolando Menchaca Méndez y y cede los derechos del trabajo intitulado "Análisis de explicabilidad de redes neuronales en tareas de procesamiento de texto", 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 expresado del autor y/o director(es). Este puede ser obtenido escribiendo a las siguiente(s) dirección(es) de correo angelavelarb.94@gmail.com. Si el permiso se otorga, al usuario deberá dar agradecimiento correspondiente y citar la fuente de este.

José Ángel Avelar Barragán

Nombre completo y firma autografá del (de la) estudiante

Resumen

Últimamente ha habido un aumento en la implementación de algoritmos de inteligencia artificial en áreas donde repercuten en la toma de decisiones que realizará un humano basado en sus resultados. La predicción de enfermedades, identificación de individuos por biométricos, detección de noticias falsas entre otras son tareas cuyo resultado impacta directamente en los individuos. La responsabilidad de la implementación de estos algoritmos cae en su empleador, pero como hacerlo responsable si este no conoce las razones detrás de la decisión tomada.

En respuesta tenemos el surgimiento de la inteligencia artificial explicable y los algoritmos de aprendizaje máquina interpretables, alternativas o apoyos a los algoritmos anteriores donde el usuario final es provisto de una explicación sobre las decisiones tomadas por el modelo.

En particular la importancia de las características nos ayuda a saber cuán tan importante es una característica para el resultado de nuestro modelo y nos permite entender el funcionamiento de una forma más literal en lugar de abstracta.

El presente documento pretende realizar un marco de trabajo para explicar mediante modelos sustitutos lineales el comportamiento de clasificadores de texto a nivel local al encontrar la importancia de las características, así como realizar una presentación de la explicación con dichas características.

Este marco de trabajo incluye la evaluación del impacto de características consideradas importantes, así como la similitud de los modelos sustitutos empleados por los explicadores con respecto el modelo que se busca explicar.

Palabras clave: Explicabilidad, Importancia de características, Modelos sustitutos lineales

ABSTRACT

The implementation of artificial intelligence algorithms has recently reached tasks affecting human decision-making. Illness prediction, biometric authentification, and fake news detections, among others, are tasks that directly concern the population. The responsibility for implementing these algorithms falls on the user of them, but how to hold responsible to them if they do not know the reasoning behind the decision taken.

In response, the fields of explainable artificial intelligence and interpretable machine learning have been developed, proposing an alternative or improving previous algorithms. With them, the final user is provided with an explanation of the reasoning behind the model's decision.

Specifically, the importance of feature importance assists in knowing how relevant a feature is for the model's prediction and allows us to understand how the model works using interpretable knowledge.

This work presents a framework using linear surrogate models to calculate the variation of models' prediction after finding the feature importance and presenting an explanation with them. This framework includes a feature impact evaluation, including a similarity measure between the surrogate models and the model which is wanted to explain.

Key words: Explainability, feature importance, linear surrogate models

AGRADECIMIENTOS

Al Instituto Politécnico Nacional, por permitirme considerarla mi *Alma mater*.
Al Centro de Investigación en Computación por ofrecerme un espacio para el estudio en el cuál me sentí apreciado.
Al CONACYT por proveer apoyo económico y fomentar la investigación y divulgación en México.
A los docentes, estudiantes y miembros del personal administrativo con quienes sentí se formó el sentimiento de comunidad.
Al Dr. Rolando Menchaca Méndez y al Dr. Grigori Sidorov, por siempre estar presentes y por su apoyo en el desarrollo de este trabajo.
A mis padres, por esos momentos de debilidad en los que fueron un primer apoyo.
A mis amigos del centro, por esos buenos momentos que se tuvieron y aquellos que quedaron pendientes.

ÍNDICE

LISTA DE FIGURAS

FIGURA **Pág.**

Lista de Tablas

CAPÍTULO 1

INTRODUCCIÓN

1.1 Entender el resultado de clasificadores: obteniendo una explicación

En la actualidad el uso de la inteligencia artificial en diferentes áreas como son el diagnostico médico, procesamiento de imágenes, generación automática de textos, entre muchos otros, parece imprescindible para el desarrollo tecnológico [25].

Esta tendencia ha sido acompañada con el uso de modelos cada vez más complejos y con una mayor cantidad de parámetros a optimizar, y aunque la cantidad de dichos parámetros es menor que el número de muestras que se tienen, se logra un error pequeño en pruebas locales, aunque no necesariamente se garantiza su correcto funcionamiento en la práctica o en la implementación del modelo. [31],[43].

El área de procesamiento de lenguaje natural (NLP) no es la excepción, ya que como se menciona en [35] se ha pasado de modelos transparentes como los tradicionales árboles de decisión, regresión logística y modelos de Markov , a modelos cada vez más opacos como *embeddings* y modelos de aprendizaje profundo como los transformadores [35].

Para referirnos al fenómeno nombrado en el párrafo anterior decimos que se trata de un "modelo opaco" a aquel modelo que si bien se entiende su desarrollo y fundamento matemático (funciona como se espera), el desarrollo de su toma de decisiones no queda claro en términos interpretables es decir, no es posible indicar las razones especificas detrás de cada resultado.

La aplicación de modelos opacos impacta de manera diferente dependiendo de qué se esté trabajando, este impacto podría tener repercusiones al presentar tendencias no percibidas, falta de causalidad, poca generalización o falta de confianza por parte del usuario final. Estos problemas pueden acarrear conflictos relacionados con la utilidad y reproducibilidad o incluso conflictos éticos y legales [33], [27] .

Como ejemplos representativos tenemos los casos de [20] donde el corpus presentaba un sesgo que asociaba la tez negra con términos desagradables en comparación con la tez blanca, y [14] donde una red neuronal artificial empleada para distinguir entre tanques enemigos y aliados diferenciaba en su lugar el clima detrás de los tanques (soleado, nublado).

La necesidad de una explicación sobre el funcionamiento de los modelos opacos ha llevado a realizar acciones como las del Parlamento Europeo [22], donde se implementó la regulación GDPR (*General Data Protection Regulation*) donde se aboga por los derechos de las personas de las cuales se han tomado sus datos y están siendo usados para tomar decisiones respecto ellas. La discusión sobre la existencia, posibilidad y alcance de dichos derechos se profundiza en [24]
En esta tesis se presenta una marco de trabajo para explicar el funcionamiento de clasificadores de texto, empleando las características significativas. Entendiendo por modelo explicable a aquel del cual tenemos una explicación sobre las causas detrás de sus decisiones (explicabilidad), y esta explicación es entendible por un humano (interpretabilidad). En particular se considera utilizar explicadores lineales basados en la importancia de las características de entrada utilizando dicha importancia como explicación de la clasificación de textos de noticias por tópico.

1.2 Planteamiento del problema

La clasificación de textos, es un tema clásico en el área de procesamiento de lenguaje natural y ha sido abordado mediante diferentes técnicas. Pero la evaluación de estas clasificaciones está asociada con un parámetro numérico como la precisión (*accuracy*), la exhaustividad (*recall*) o la norma F1, que si bien nos indican qué tanto desempeño se tuvo respecto el conjunto de datos empleados, no nos indican el comportamiento del clasificador y la razón detrás de sus decisiones.

Obtener una representación interpretable del funcionamiento del clasificador, así como las razones detrás de cada salida entregada por este son necesarias para poder determinar si se logró la tarea especifica con el conocimiento adecuado del problema, fue resuelto por casualidades y sesgos propios de la base de datos, o si incluso el sistema puede proveer conocimiento nuevo sobre la clasificación.

A continuación se darán definiciones sobre los términos que se han mencionado y utilizarán en adelante, partiendo de los conceptos más básicos y terminando con composiciones de estos.

Muestras y características:
Dado un conjunto de entrada X con una cantidad de elementos N, llamamos a cada elemento x como una muestra (o entrada) de X. Cada una de estas muestras posee una cantidad de características que le definen, por simplicidad supondremos que son la misma cantidad m de características para cada muestra y que todas las muestras poseen algún valor asociado a cada característica.
En nuestro caso nuestro corpus (X) contiene N documentos compuestos del mismo vocabulario y podemos representar a cada documento en un modelo de espacio vectorial donde a cada característica tiene un valor numérico.
Definimos el concepto de predictor como todo modelo del cual obtengamos una predicción (numérica o categórica) según sus entradas. En particular nos enfocaremos en el clasificador y el problema de clasificación.

Clasificador:
Sea un clasificador b encargado de la tarea de mapear instancias x provenientes de un conjunto X que contienen cada una una cantidad de características m (representado como X^m en adelante), a un elemento $\hat{y}$ del conjunto Y de categorías: $b : X^m \rightarrow Y$. Donde $\hat{y}$ es la categoría predicha por b para la instancia especifica x.

Muestras de entrada simplificadas:
Dada una muestra x definimos una muestra simplificada x' al vector binario que indica un subespacio de características a emplear de x.

Espacio de muestras generadas:
Llamaremos Z al espacio de muestras generado a partir de x, en particular Z representa una vecindad de elementos entorno a x obtenidos mediante algún proceso propio del algoritmo explicador. Los elementos z de Z poseen a su vez una versión simplificada nombrada z'.

Función de mapeo (de entradas simplificadas):
Los explicadores lineales que utilizaremos a continuación emplean un mapeo de características simplificadas, en general nombrado como h_x que dado una x' nos retorna su x, es decir $x = h_x(x')$

Modelo lineal sustituto:
Definimos un modelo lineal sustituto g miembro del conjunto de modelos lineales interpretables G que tiene la forma $g(z') = \sum_{i=0}^{N} w_i z'_i$ o también $w_0 + \sum_{i=1}^{N} w_i z'_i$ dejando explicito que w_0 hace

referencia a un peso de ajuste independiente de las características.

Formalmente podemos definir los problemas a considerar como:

Problema de la explicación del modelo global:
Sea un clasificador b y el conjunto de instancias X, la explicación del clasificador E (que se encuentra dentro del conjunto de explicaciones ϵ que son interpretables por un humano) se obtiene mediante la aplicación una lógica de explicación o abstracción ε_g a un predictor interpretable global c_g descrito como un proceso f que opera sobre b, X: $c_g = f(b, X)$, es decir: $E = \varepsilon_g(c_g, X)$

Problema de la explicación del modelo localmente:
Sea un clasificador b y una instancia x del conjunto X, la explicación del clasificador e se obtiene mediante la aplicación de una lógica de explicación o abstracción ε_l a un predictor interpretable local c_l que se define como un proceso f que opera sobre b, x: $c_l = f(b, x)$, es decir: $e = \varepsilon_l(c_l, x)$. En nuestro caso el predictor interpretable local se trata de un modelo lineal sustituto g.
A continuación se resumen a manera de tabla los términos y expresiones nombrados:

Notación	Expresión
x, X	Muestra de entrada y conjunto de estas
y, Y	Categoría de clasificación y conjunto de categorías
c, V	característica y vocabulario
m	Cantidad de características ($m = \|V\|$)
N	Número de muestras de entrada ($N = \|X\|$)
b	Predictor
$\hat{y}$	Predicción del clasificador, modelo opaco.
$\hat{y}_*$	Predicción del modelo sustituto interpretable
g	Modelo lineal sustituto interpretable
w_i	Pesos del modelo lineal sustituto
h_x^D	Función de decodificadora del mapeo de muestras simplificadas
h_x^C	Función de codificadora de las muestras a muestras simplificadas
x'	Muestra simplificada
z, Z	Muestra generada y espacio de muestras interpretables generadas
z'	Muestra interpretable simplificada
f	Proceso para obtener predictores interpretables
c_g, c_l	Predictor interpretable global y predictor interpretable local
ϵ_g, ϵ_l	abstracción local o global de predictores interpretables
E, e	Explicación global y explicación local

Tabla 1.1: Símbolos y significado

En este trabajo se busca encontrar y optimizar explicaciones e empleando diferentes explicadores lineales ε_l sobre diferentes clasificadores b y comparar las explicaciones obtenidas para saber si

son congruentes entre sí o no y porqué lo son.

1.3 Objetivos

1.3.1 Objetivo General

- Desarrollar y caracterizar experimentalmente un marco de trabajo para interpretar modelos no lineales entrenados aplicados al problema de clasificación de textos.

1.3.2 Objetivos Específicos

1. Implementar diferentes clasificadores no lineales al problema de clasificación de noticias del conjunto de datos 20Newspaper.
2. Desarrollar un análisis experimental comparativo del desempeño los algoritmos de clasificación.
3. Desarrollar un análisis experimental del desempeño de los explicadores lineales al aplicarlos a cada uno de los algoritmos de clasificación.
4. Interpretar el funcionamiento de los clasificadores con el marco de trabajo propuesto.

1.4 Justificación

Realizar un marco de trabajo que permita interpretar los resultados de los clasificadores empleados en la tarea de clasificación de texto por categoría resulta de utilidad debido a que permite tener una aproximación del funcionamiento del sistema, nos indica la forma en la que las características contribuyen a las salidas obtenidas y detectar posibles sesgos en la base de datos permitiendo entender la razón detrás del resultado del clasificador.

Además se debe presentar una forma de satisfacer requerimientos en un marco legal dónde se requiera una explicación en la cual el usuario final pueda confiar.

1.5 Organización de la Tesis

Los contenidos de esta tesis son:

- **capitulo 1**: Introducción.

  Un panorama general sobre los problemas a abordar y su justificación.

- **capitulo 2**: Marco teórico.

  Conceptos del problema de clasificación de textos por categoría.

- **capitulo 3**: Explicadores.

  Conceptos del problema de explicadores de modelos sustitutos lineales basados en importancia de características.

- **capitulo 4**: Trabajo relacionado.

  Mención de trabajos relacionados y diferencias con la propuesta.

- **capitulo 5**: Propuesta de solución.

  Metodología realizada en el trabajo.

- **capitulo 6**: Resultados.

  Desarrollo de la solución propuesta, junto con interpretaciones de ejemplos específicos.

- **capitulo 7**: Conclusiones.

  Conclusiones del trabajo, interpretaciones generales de los resultados y propuestas de trabajo a futuro.

CAPÍTULO 2

MARCO TEÓRICO

Dentro de este capitulo trataremos las bases para entender el problema de clasificación de textos, la terminología y algoritmos empleados. De estos últimos tendremos los algoritmos clasificadores y los modelos explicadores.

2.1 Problema de clasificación de textos

Dada una descripción, elemento o muestra $x \in X$ de un documento, donde X es el espacio de documentos, es decir una instancia o documento x tiene una cantidad de características m y pertenece al conjunto X y dado un conjunto fijo de categorías o clases Y con etiquetas o elementos y se tiene que a cada elemento de X se le es asigna al menos un elemento de Y.
Definimos al problema de clasificación de textos como el problema de asignar las etiquetas que corresponden a cierta categoría y a las instancias x mediante un algoritmo. Así, dicho algoritmo o método de aprendizaje debe cumplir con el mapeo $b : X \rightarrow Y$ [11] donde la predicción de dicho mapeo $\hat{y}$ debe coincidir con la categoría verdadera del documento y. El error o diferencia entre dichas categorías busca minimizarse para que nuestro clasificador prediga correctamente dicho etiquetado.

Entre las diferentes formas que se puede abordar el problema de clasificación tenemos la clasificación multi-etiqueta (a cada elemento se le pueden asignar diferentes etiquetas de categoría), el problema del conjunto parcialmente etiquetado (tenemos elementos sin etiqueta y hay que etiquetarlos) y el problema de conjunto de clasificación abierto (no se tiene una restricción de cuantas, ni cuales son las etiquetas).

En el caso especifico de un clasificador supervisado (del cual conocemos todas las categorías y para cada entrada x) multi-categoría se siguen los procesos de entrenamiento y predicción, ilustrados en la Figura 2.1 .

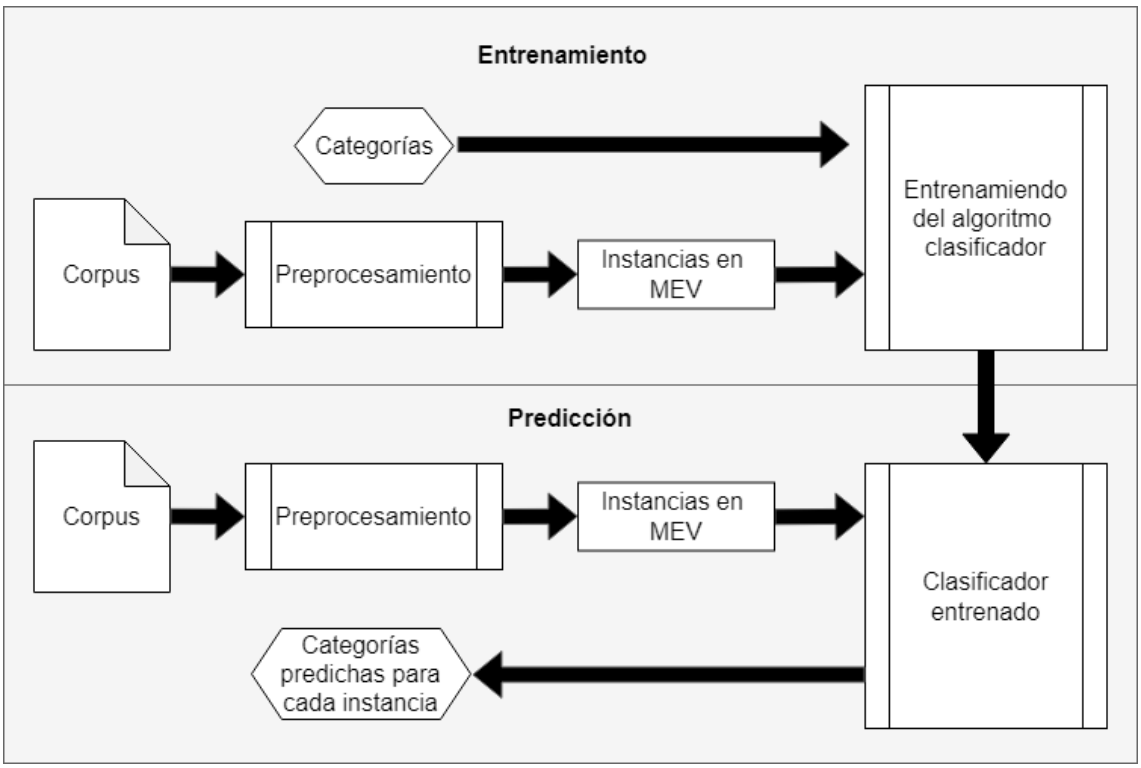


Figura 2.1: Diagrama que resume el proceso de entrenamiento y predicción de un clasificador, en el primero conocemos las etiquetas de categorías de cada una de las muestras y en el segundo el clasificador debe asignar las etiquetas de categorías que corresponden a cada instancia de entrada representada es su modelo de espacio vectorial (MEV).

Donde el modelo clasificador es el resultado de la implementación de un algoritmo de aprendizaje máquina que ha sido alimentado con características obtenidas de la información de entrada. Más adelante se detallará la obtención de características en preprocesado 2.3.
Un problema al obtener estos modelos es el cómo saber qué características son útiles y cómo codificarlas para ser procesadas por el algoritmo. La incorrecta selección de características puede derivar en problemas asociados a tendencias del corpus o que no generalice correctamente para nuevos ejemplos *(overfitting)* [10]. La selección de dichas características puede abordarse desde los selectores de características (detallados más adelante en este mismo capitulo), pero en la siguiente sección detallaremos la metodología típica para manejar el corpus o los conjuntos de datos.

2.2 Conjuntos de trabajo

Un método para refinar el conjunto de características es el análisis del error, donde dividiremos el corpus en tres conjuntos diferentes: conjunto de entrenamiento (*training*), conjunto de validación (*dev-test*) y conjunto de prueba (*test*). El conjunto formado por los primeros dos se conoce como conjunto de desarrollo (*development*) [10] y la Figura 2.2 ilustra esta separación de conjuntos.

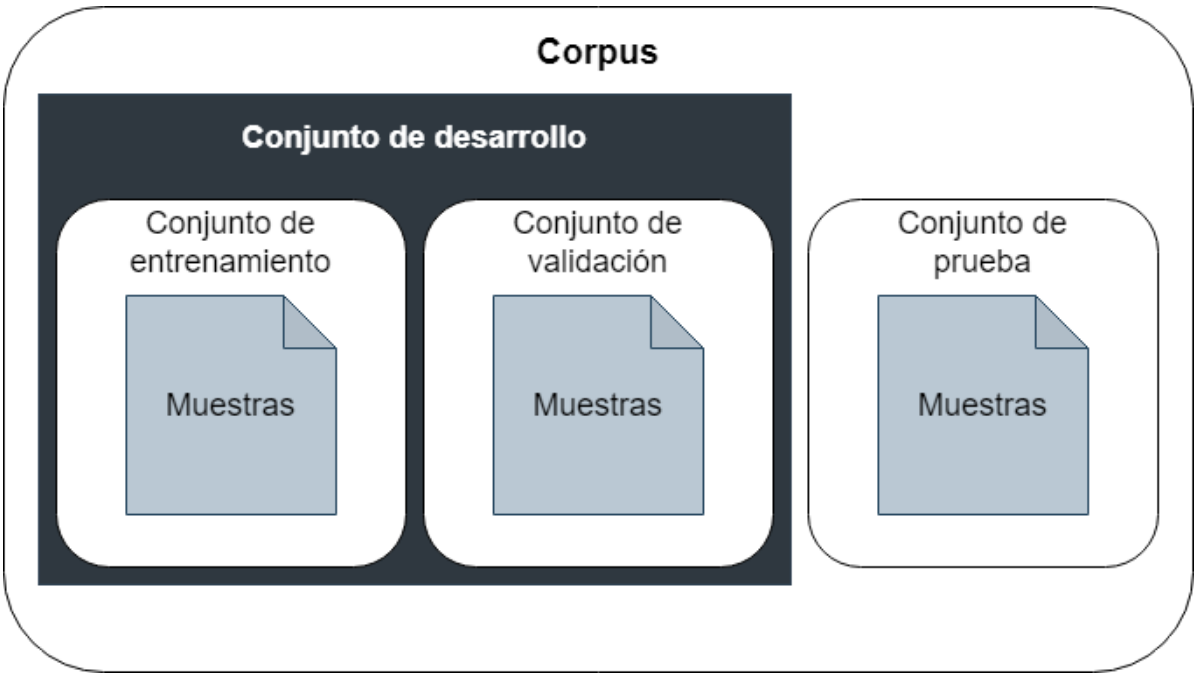


Figura 2.2: Conjuntos en los que se divide el corpus para poder realizar un análisis del error, evaluando y reportando el desempeño del clasificador con muestras no observadas durante el entrenamiento.

El objetivo de estos conjuntos es quedarnos con el clasificador que haya tenido un mejor desempeño en la generalización definiendo dicho desempeño acorde a una métrica.
El procedimiento de este análisis del error consiste en entrenar nuestro clasificador con el conjunto de entrenamiento, ajustar sus hiperparámetros y selección de características al evaluar el clasificador con el conjunto de validación y finalmente presentar los resultados del clasificador con el conjunto de prueba. [10]

2.3 Preprocesado

En la sección anterior hablamos sobre como separaríamos nuestro corpus en subconjuntos para la evaluación de nuestros clasificadores, en esta sección hablaremos sobre qué tomamos como

característica en el contexto de procesamiento de lenguaje natural.

Antes de poder trabajar con características debemos determinar qué unidad consideraremos como tales (palabras, oraciones, etc.) y para llegar a dicho punto de partida debemos realizar un preprocesado del texto en crudo.

Nuestro corpus o conjunto de datos debe considerarse como un conjunto que cumple cierto criterio de obtención, de forma que los resultados no necesariamente pueden extrapolarse a otros conjuntos con diferentes criterios [3].

El primer paso de nuestro preprocesado consiste en retirar todos los elementos considerados irrelevantes o que estorban en nuestros documentos: cabeceras, separadores, tablas, diagramas, figuras, etc. son ejemplos de elementos retirados en este paso.

Posteriormente debemos aplicar una normalización del texto, dependiendo de la tarea se debe considerar cambiar el texto a minúsculas, retirar sufijos y prefijos (*stemmers*), revisar si las palabras están incluidas en un diccionario (*lemmatization*), retirar caracteres no alfanuméricos y finalmente retirar o no palabras funcionales de alta frecuencia (*stopwords*)[10].

Para el siguiente paso debemos recortar las cadenas de texto resultantes para tener unidades lingüísticas identificables con las que trabajaremos, que denominaremos *Tokens*, la forma de obtener estos dependerá de lo que consideremos como unidad de trabajo y la definición que le demos[3]. En nuestro caso una 'palabra' no considera signos de puntuación, apostrofes ni guiones y considera que los espacio en blanco separan a una palabra de otra.

Otra operación a realizar sería la referente a remplazar las palabras por la parte a la que corresponden en el discurso, es decir un etiquetado gramatical *part-of-speech tagging*[3], lo cual es un problema en sí mismo y no se incluye en este trabajo. Pero puede considerarse la tarea de etiquetar parcialmente ciertos elementos que no se consideren de interés descritos de diferente forma, como lo podrían ser números (telefónicos), direcciones de correo, etc.

2.4 TF-IDF

Empleando un modelo de bolsa de palabras, tenemos una descripción de los documentos que depende si está presente o no cada palabra en cuestión, pero podemos hacer una ponderación en dependencia de la frecuencia con la que el termino aparece a lo largo de cada documento.

Además, considerando la cantidad de documentos en los que aparece el termino como una medida de su no relevancia (si aparece en todos los documentos no es de utilidad para diferenciar una clase de otra) obteniendo así una ponderación inversa a la frecuencia con la que el término aparece en los documentos.

Empleando estos dos conceptos obtenemos la ponderación $tf - idf_{t,d}$ (*term frequency, inverse document frequency*) para cada termino t a lo largo de todos los documentos d, siendo definida como:

$$tf - idf_{t,d} = tf_{t,d} \times idf_t =$$

$$(\#\text{ de ocurrencias de t en el documento d}) \times log\left(\frac{\#\text{ de documentos en total}}{\#\text{ de documentos donde aparece el término t}}\right)$$

Finalmente, con estas descripciones numéricas por término y por documento tenemos una representación en un espacio vectorial con el cual podemos operar y medir la similitud de sus elementos[11]. Siendo estos *tokens* nuestras características en adelante.

2.5 Selección de características

En la sección anterior decidimos como obtener unidades de trabajo o *tokens* que representan las características y forman el vocabulario con el que describiremos los documentos (representados como c y V respectivamente), sin embargo no necesariamente debemos utilizar todos los elementos que se obtienen. En el preprocesado sugerimos que se pueden retirar o sustituir algunas características, este proceso es conocido como selección de características y también se puede realizar empleando algún algoritmo en lugar de un criterio particular.

El objetivo general de la selección de características es retirar aquellas características que se consideran "ruido" (*noise feature*), es decir aquellas que cuando se incluyen en la representación de los documentos aumentan el error de la clasificación de nueva información, logrando así evitar el *overfitting* debido a características circunstanciales [11].

Aunque parezca que utilizar una menor cantidad de características resultaría en un desempeño menor del clasificador resulta preferible trabajar con modelos reducidos (revisar *bias-variance tradeoff*) al tener estos una menor tendencia al *overfitting* [11] y (si se emplean características significativas) representar mejor el fenómeno de manera abstracta.

Dada una métrica que nos indique la utilidad de una característica c para cierta clase y a clasificar nos quedaremos con los primeros K términos que tengan mayor utilidad[11]. El problema empieza con cómo se obtiene dicha utilidad o importancia de las características y la viabilidad de calcular la utilidad de todos los términos para quedarnos con solo los K mejores.

2.5.1 Selección mediante prueba χ^2

Un método estadístico a emplear es la prueba χ^2. Esta se utiliza para medir la independencia entre dos eventos, y en su aplicación de selección de características se utilizan como eventos la ocurrencia del término y la ocurrencia de la clase. De forma numérica esto se logra con la expresión:

$$\chi^2(X,c,y) = \sum_{e_c \in \{0,1\}} \sum_{e_y \in \{0,1\}} \frac{N_{e_c e_y} - E_{e_c e_y}}{E_{e_c e_y}}$$

Donde obtenemos el coeficiente χ^2 de dado el conjunto de documentos X, de una característica c en una clase y. Sumando la contribución de cuando el término se encuentra o no en el documento ($e_c\{1,0\}$) y de cuando el documento pertenece o no a la clase de la que se busca su contribución ($e_y\{0,1\}$). Obteniendo en cada caso una fracción de el valor esperado E y el valor observado N. Un alto valor de χ^2 implica una hipótesis de independencia, y que los valores observados y esperados distan entre sí, permitiéndonos decir que la clase y el termino son independientes entre sí con cierto grado de confianza. Por el caso contrario un valor pequeño significaría que el término y la clase tienen cierta dependencia y la ocurrencia de uno viene acompañada de la ocurrencia del otro.

Como último apartado de esta sección mencionaremos solo para completar otros dos métodos de selección de características utilizados en la clasificación de textos: Información mutua (*Mutual Information*) y selección de características basada en frecuencia (*Frequency-based feature selection*) [11].

2.6 Clasificadores

En esta sección hablaremos sobre los diferentes clasificadores que serán empleados en este trabajo, en general se ha utilizado la notación b para hacer referencia a estos y como su nombre lo indica son algoritmos que nos regresarán un etiquetado de las entradas a partir de ciertas consideraciones sobre las características de estas.

2.6.1 K-vecinos

El algoritmo K-vecinos (*K-nearest neighbor*) realiza un procedimiento de clasificación a nivel local: dado un elemento a clasificar se revisan los K elementos más cercanos a éste, y se le asigna la clase a la cual pertenecen la mayoría de sus vecinos.

K-vecinos hace uso de la hipótesis de contigüidad, donde se considera que si se tiene un elemento x de categoría y entonces los elementos cercanos entorno a este deben pertenecer a dicha categoría.

Para que el retorno del algoritmo sea una probabilidad de pertenencia a dicha clase, simplemente se devuelve la proporción de los vecinos de la clase mayoritaria respecto todos los vecinos considerados.

Además, consideremos que no todas las contribuciones son iguales al momento de decidir la clase. Podemos ponderar los votos en dependencia de alguna distancia, siendo así que los elementos más cercanos contribuyan más que aquellos que estén más lejos.

La elección del valor del hiperparámetro K queda en dependencia de la naturaleza del problema, pero esta suele ser impar [11] y es posible ajustarla en el proceso de validación tratado en la sección 2.8. En el caso particular de cuando $K = 1$ se tiene que la frontera de decisión del clasificador coincide con el teselado de Voronoi [11] como se ilustra en la Figura 2.3 junto con la superficie de decisión de una $K = 10$. Mientras que la decisión del clasificador para asignar las etiquetas en la predicción es entendible e interpretable con el teselado para el caso $K = 1$, el caso de $K > 1$ depende de la interpretación del algoritmo empleado para encontrar los siguientes puntos más cercanos después del primero (usualmente algoritmos de árbol de búsqueda binaria).

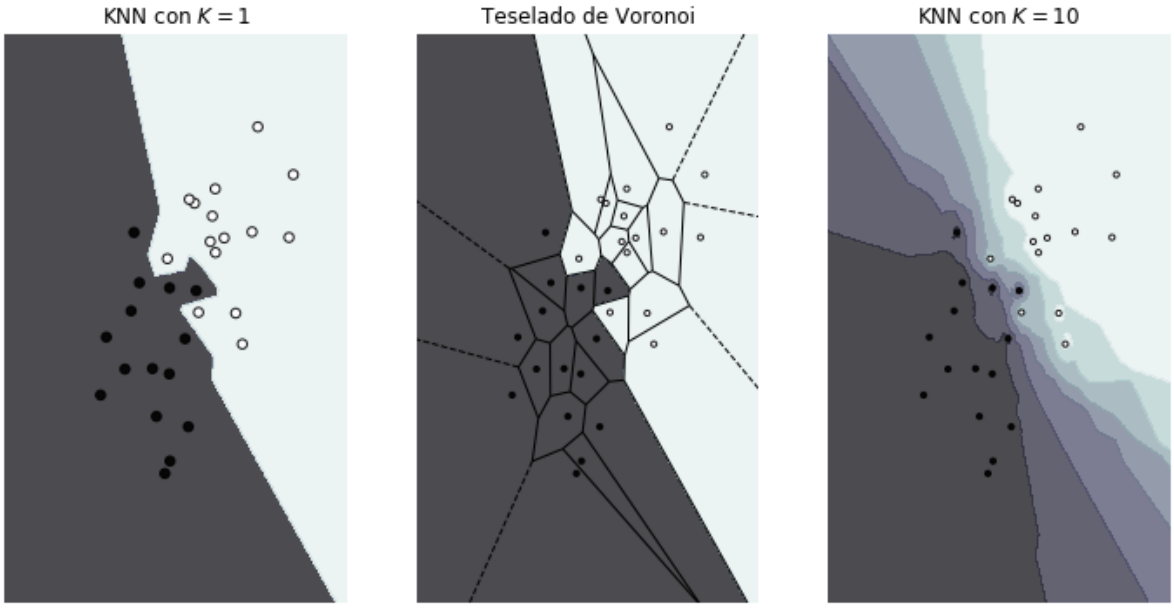


Figura 2.3: Frontera de decisión de un clasificador binario empleando KNN con $K = 1$ (izquierda). Teselado de Voronoi (centro). Fronteras de decisión con $K = 10$ (derecha).

2.6.2 SVM

Las *Support Vector Machines* o Máquinas de Vector Soporte (SVM en adelante) son modelos típicos para la clasificación y regresión originalmente propuestos por Vapnik en [1].

Estas parten de los clasificadores de máximo margen (*Maximum margin classifers*) cuyo objetivo es encontrar la distancia mínima entre las fronteras de decisión y cualquier muestra a clasificar.

Empleando un modelo lineal de clasificación binaria de la forma:

$$\hat{y}(x) = w^T \phi(x) + b$$

Donde $\phi(x)$ representa alguna transformación del espacio de características aplicada a los vectores de entrada x_i que tendrán una etiqueta objetivo binaria respectiva $y_i \in \{-1, 1\}$.

La distancia mínima entre las fronteras de decisión se obtiene partiendo primero de únicamente las instancias correctamente clasificadas, es decir $y_i \hat{y}(x_i) > 0$ la etiqueta correspondiente coincide con la predicción del clasificador. Dicha distancia queda definida como la distancia perpendicular del hiperplano de frontera de decisión y la instancia en cuestión. Buscando optimizar los parámetros w, b obtenemos:

$$arg\,max_{w,b}\Big\{\min_i \Big[\frac{y_i(w^T\phi(x_i)+b)}{||w||}\Big]\Big\}$$

Reescalando la distancia normal, es decir restringiendo el problema para que los puntos que satisfagan $y_i(w^T\phi(x_i)+b) \geq 1$, nuestro problema se reduce a maximizar $\frac{1}{|||w||}$, que equivale a minimizar $||w||^2$.

Si buscamos que pueda haber elementos mal clasificados empleamos las llamadas *slack variables* ξ_i que son definidas como $\xi_i = |y_i - \hat{y}(x_i)|$, obtenemos las funciones a optimizar:

$$\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{i=0}^{n} \xi_i$$

restringido por:

$$y_i(w^T\phi(x_i)+b) \geq 1-\xi_i,\ \xi_i \geq 0,\ \forall\ i$$

Donde C hace referencia a un parámetro de regulación. Esta formulación es conocida como *C-Support Vector Classification* [12].

Ahora, en lugar de resolver el problema con $\phi(x_i)$, resolvemos lo que se conoce como el problema dual, es decir:

$$\min_{\alpha} \frac{1}{2}\alpha^T Q \alpha^T - I^T \alpha$$

restringido por:

$$y^T\alpha = 0,\ 0 \geq \alpha \geq C$$

donde I es el vector identidad, Q es una matriz de $n \times n$ positiva semidefinida, de la forma: $Q_{ij} = y_i y_j K(x_i, x_j)$ donde $K(x_i, x_j)$ es una función conocida como *kernel* [8], [12].

Anteriormente se mencionó la aplicación de una función de mapeo ϕ, con la cual se podía definir la relación: $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, que sigue de la definición de que el kernel k debe ser una función simétrica.

El uso de kernel emplea el producto interior para generar extensiones empleando un *kernel trick*, cuyo principio es remplazar el producto escalar por cualquier otro producto (definido por el kernel) y evaluar este nuevo producto en su lugar.

Considerando a la función de mapeo $\phi(x) = x$ tenemos al kernel lineal: $k(x_i, x_j) = x_i^T x_j$, la más simple de las funciones de kernel. En el caso de que el kernel sea una función que dependa únicamente de la distancia de sus argumentos, digamos $k(x_i, x_j) = k(||x_i - x_j||)$ entonces se denominarán como kernels homogéneos o *radial basis functions* (RBF).

Dichos kernels RBF emplean la distancia radial para definirse, en particular un kernel RBF gaussiano está definido por la relación $k(x_i, x_j) = e^{-\gamma||x_i - x_j||^2}$ [12].

La aplicación del clasificador binario del C-SVM como un clasificador multiclase tiene 2 posibles aproximaciones: *one vs rest* o *one vs one*.

En el primero se utilizan clasificadores binarios que distinguen entre la clase a clasificar y los elementos que no pertenecen, obteniendo por definición conjuntos de entrenamiento desbalanceados. Por otro lado el segundo enfoque emplea $M(M-1)/2$ clasificadores binarios para M clases diferentes, se elige aquella clase que indique la mayoría de clasificadores binarios [12].

2.6.3 Perceptrón Multicapa

La Perceptrón Multicapa o MLP por sus siglas en inglés (*Multi-Layer Perceptron*) se trata de un modelo de red neuronal compuesto de múltiples capas de modelos de regresión logística. Y se trata de un modelo más compacto que las SVM pero cuya función de verosimilitud no se trata de una función convexa [8].

Para entender el funcionamiento de una MLP, consideremos un modelo lineal donde dada la descripción de una muestra x siendo x_i la i-ésima característica:

$$a_j = \sum_{i=1}^{M} \omega_{ji}^{(1)} x_i + \omega_{j0}^{(1)}$$

Donde el coeficiente ω_{ji} hace referencia a un parámetro a ajustar del modelo, perteneciente a la capa (1) y donde se obtiene una activación a_j.

Posteriormente dicha activación es transformada utilizando una función de activación que es una función diferenciable no lineal $\sigma()$ (típicamente se denota por $h()$, pero se prefiere esta notación para evitar confusiones con conceptos que se presentan en secciones posteriores) obteniendo así la salida del modelo: $z_j = \sigma(a_j)$.

Ahora, cada una de estas salidas corresponden a la entrada de una nueva capa de modelos lineales que se denomina capa oculta, definiendo ahora una activación de la forma:

$$a_k = \sum_{j=1}^{M'} \omega_{kj}^{(2)} x_i + \omega_{k0}^{(2)}$$

Donde la cantidad de modelos de la capa anterior corresponde con M' y las salidas de estos, este proceso se repite apilando capas ocultas de manera consecutiva hasta llegar a la última capa, denominada de salida [8]. En esta ultima capa se realiza la comparación de la predicción hecha por el clasificador con la clase a predecir $\hat{y} - y$ y se utiliza dicho error para el proceso de actualización de los parámetros ω.

El proceso de entrenamiento del modelo emplea una función de error dependiente de los parámetros del modelo, sea por ejemplo la función de error cuadrático medio:

$$E(\omega) = \frac{1}{2} \sum_{n=1}^{N} ||\hat{y} - y||^2$$

Obteniendo la contribución del error de cada característica mediante aplicaciones sucesivas de regla de la cadena podemos modificar el valor de cada parámetro ω. La obtención de dichas contribuciones es conocida como el algoritmo *Backpropagation* mientras que la forma de actualizar los coeficientes se le conoce como función de aprendizaje y suele emplear algún método basado en el descenso del gradiente para obtener el conjunto de parámetros óptimo.

2.7 Evaluación de clasificadores

En esta sección se presenta la forma en que los clasificadores de texto son evaluados mediante diferentes métricas basadas en el conteo de muestras correcta o incorrectamente clasificadas. La evaluación de los etiquetados de los clasificadores varia si se evalúa el conjunto de entrenamiento u otro de los conjuntos restantes.

En el primer caso podría ser que nuestro clasificador memorizara las etiquetas de entrada, y en el segundo caso dependería de la similitud de este segundo conjunto y el conjunto de entrenamiento. Si ambos conjuntos son muy similares tendremos una menor posibilidad de alcanzar la generalización [10].

La exactitud (*acc*) es la proporción de elementos correctamente etiquetados respecto el total de elementos clasificados, entre más cercana sea a 1 más exacto es el clasificador. Mientras que otras métricas empleadas son la precisión (*prec*), la exhaustividad (*rec*) y la norma-F1. Para entender la forma en que se calculan debemos entender los cuatro posibles casos de resultados de clasificación:

- Verdaderos Positivos (TP): Elementos correctamente clasificados de la clase buscada.
- Verdaderos Negativos (TN): Elementos correctamente clasificados de la clase no buscada.
- Falsos Positivos (FP): Elementos que no pertenecen a la clase buscada clasificados como miembros de esta.
- Falsos Negativos (FN): Elementos que pertenecen a la clase buscada clasificados como pertenecientes a otra.

Así podemos definir a las métricas con las expresiones ilustradas en la Tabla 2.1:

Exactitud	*acc*	$\frac{TP+TN}{TP+TN+FN+FP}$
Precisión	*prec*	$\frac{TP}{TP+FP}$
Exhaustividad	*rec*	$\frac{TP}{TP+FN}$
Norma F1	F1	$\frac{2\times prec\times rec}{(prec+rec)}$

Tabla 2.1: Métricas comunes para evaluar la tarea de clasificación de textos y sus respectivas expresiones.

De estas medidas debe entenderse el significado de cada una para el problema de clasificación de texto multi-categoría, y bajo este contexto la *acc* podría no ser una buena métrica por el desbalance que existe entre las clases al considerar que cada categoría emplea el uso de clasificadores binarios contra el resto de categorías.

Bien podemos entender como *prec* como la medida que nos indica de cuantos de los elementos devueltos con la etiqueta de interés son realmente miembros de dicha categoría. Entre más cercana a 1 mejor.

Así mismo la *rec* sería la medida que indica cuantos de los elementos que debían ser devueltos con la etiqueta han sido devueltos correctamente por el clasificador. Entre más cercana a 1 mejor. La ponderación de estas dos métricas se realiza mediante las medidas F, cuyo caso mencionado corresponde a la medida F1 que asigna igual importancia a cada una de ellas [11].
Usualmente los clasificadores no binarios emplean múltiples instancias de clasificadores binarios

para realizar la clasificación para poder tener un resultado promedio de las métricas anteriores sobre los diferentes clasificadores tenemos dos métodos: El *macro-averaging* devolverá el promedio de las métricas por cada una de las clases como si su contribución fuera idéntica. Y el *micro-averaging* considerará que cada documento contribuye equitativamente al resultado. Si existe un desvalance entre la cantidad de miembros de una clase respecto de otra esta se notará en el *micro-averaging* [11].

Una forma de ilustrar el desempeño de un clasificador considerando todas las clases a la vez es mediante una matriz de confusión, con la cual se comparan los resultados de todas las posibles combinaciones de clasificaciones. Entre más cercano sean los resultados a aparecer en la diagonal de la matriz, mejor será el desempeño de nuestro clasificador [10]. En la Figura 2.4 se ilustra la forma de una matriz de confusión.

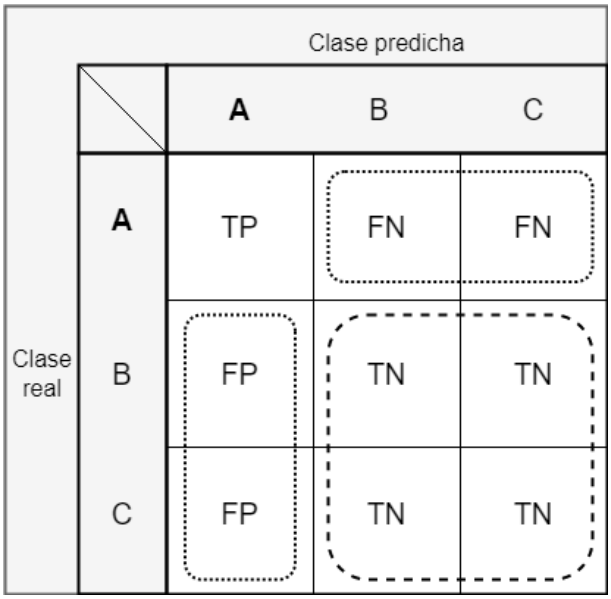


Figura 2.4: Ilustración de una matriz de confusión multi-clase donde se revisa el clasificador para la clase A, los documentos asignados como *TP, TN, FN y FP* varían dependiendo de qué clase se esté revisando del clasificador.

2.8 Validación Cruzada

Anteriormente mencionamos la forma de separar en subconjuntos nuestro conjunto de muestras originales, pero esto nos lleva a preguntarnos la proporción ideal de dichos subconjuntos: un conjunto de prueba pequeño podría derivar en evaluaciones sesgadas, y un conjunto de entrenamiento pequeño podría no llegar a generalizar.

Una forma de evitar estos conflictos es empleando validación cruzada (*cross validation*), donde la idea general es obtener diferentes subconjuntos de desarrollo y prueba en cada iteración, y valorar el desempeño de nuestro algoritmo a lo largo del proceso.

Al subdividir el corpus en K subconjuntos, de forma que empleamos $K-1$ subconjuntos para el desarrollo y un subconjunto para la prueba, variando en cada iteración dicho subconjunto entre los K posibles estaremos empleando *K-folding cross validation* o validación cruzada de K-iteraciones [10].

Si el desempeño de nuestro algoritmo es similar a lo largo de las iteraciones tendremos una idea aproximada de su buen funcionamiento, sin embargo una disparidad en el desempeño podría indicar que el comportamiento del algoritmo depende inherentemente de la porción de información empleada.

CAPÍTULO 3

EXPLICADORES

En esta sección se presentan los conceptos de interpretabilidad y explicabilidad, indicando los tipos de problemas de explicación que se pueden abordar (taxonomía), qué es lo que se entiende por un modelo interpretable y un modelo opaco, y finalmente tres algoritmos de explicación lineales basados en la importancia características.

3.1 Conceptos generales

Entendemos como interpretabilidad a la habilidad de explicar o indicar el significado en términos entendibles por un humano según los autores [21]. Otra definición general asociada con los predictores la define como el grado de entendimiento con el cual un humano puede discernir la causa de una decisión tomada por el predictor [32].

A partir de estas definiciones podemos decir que un modelo es interpretable si tenemos una unidad entendible (interpretable) que nos da un conocimiento sobre las razones de un procedimiento, en especifico, las causas por las que un predictor nos da cierto resultado en especifico.

Por otro lado, la explicabilidad puede definirse como la capacidad que se tiene para formar explicaciones a partir de elementos interpretables, entendiendo por explicación al intermediario entre los humanos y los predictores [35].

Así, tendremos diferentes ejemplos específicos sobre el método que se considera para elaborar las explicaciones de un sistema que parten de la forma en que se extraen unidades interpretables (o se deciden cuales son dichas unidades), que derivan en toda una familia de problemas y modelos.

3.2 Taxonomía

Para poder entender los diferentes modelos de acuerdo con si los consideramos interpretables o no, primero debemos indicar qué características o dimensiones son las que se toman en cuenta para clasificar dichos algoritmos y así caracterizar la taxonomía de la que forman parte. Esta se resume en la Figura 3.1.

3.2.1 Global vs local

La interpretación y explicación puede ser obtenida y expresada para un único ejemplo de entrada (local) o para todo el modelo (global). Entender un modelo como interpretable globalmente implica que se comprende del todo (algoritmo, modelo obtenido e importancia de las características). En la práctica suele ser más común entender los modelos por sus partes o de manera más modular. En este trabajo se emplearan explicadores locales y con ellos ilustrar por sus partes el comportamiento del clasificador [33].

3.2.2 Intrínseco o *Post-hoc*

Si el proceso de unidades interpretables y generación de una explicación pueden obtenerse directamente del modelo obtenido, se trata de un explicador intrínseco, por ejemplo, árboles de decisión. Si al contrario necesita un procesamiento posterior o la aplicación de otro algoritmo se definirá como un explicador *Post-hoc* [33], [35].

Al emplear modelos sustitutos que requieren un procesamiento posterior al entrenamiento del modelo clasificador los explicadores presentados en este trabajo entran en la segunda categoría.

3.2.3 Metodología de la explicación

La clasificación del algoritmo explicador también depende de la forma en la que los resultados son presentados al usuario final, lo cual depende del problema a resolver y de los elementos interpretativos del mismo. En [33] se listan los siguientes métodos:

- Resumen de características, numérico (Importancia de característica).
- Resumen de características, visual (Dependencia parcial entre características, prominencia).
- Elementos del modelo (Nodos de un árbol de decisión).
- Conjunto de puntos de información (La explicación regresa una muestra generada, contraejemplos).
- Intrínseco

3.2.4 Problema a resolver

Además, en [27] se considera también cuál es el problema que se busca resolver con el algoritmo explicador para categorizar a este. Los problemas que proponen que se pueden abordar son:

- Diseño de modelos transparentes.
- Explicación del modelo obtenido.
- Explicación del resultado del modelo.
- Inspección del funcionamiento del modelo.

Figura 3.1: Diagrama de la taxonomía de los modelos interpretables, dividido en tres ramas principales.

Además de las categorizaciones mencionadas anteriormente en [33] podemos encontrar tres categorizaciones adicionales que nos presenta [27] que valoran: la naturaleza de la explicación, el tiempo que tiene el usuario para entender dicha explicación y la complejidad algorítmica de obtener la explicación. Estas tres categorías no son detalladas este trabajo, pero se mencionan por completes de la taxonomía presentada.

El enfoque de este trabajo se limita a algoritmos que buscan explicar el resultado del modelo partiendo de explicaciones locales por medio de un modelo lineal sustituto (local *Post-hoc*), presentando la explicación como una lista de características.

3.3 Modelos interpretables y opacos

En secciones anteriores se ha hablado de la transparencia del modelo, lo cual se refiere al entendimiento del algoritmo con el cual se entrena u obtiene dicho modelo. El entendimiento de este proceso no necesariamente resulta en un modelo interpretable.
Aquel modelo cuyo proceso de decisión no quede claro (características y modelo resultante) se denominará en adelante como modelo opaco.
En [27] se considera que los árboles de decisión, modelos lineales y sistemas basados en reglas son modelos interpretables por sí mismos. Lo cual coincide con lo encontrado en [33], donde además se agregan los GLM (*Modelos lineales generalizados*), GAM (*Modelos aditivos generalizados*, el clasificador *Naive Bayes* y KNN (*K-Nearest Neighbors*).

3.4 Modelos sustitutos

Una metodología para obtener explicaciones sobre el comportamiento de un modelo es emplear un modelo sustituto interpretable del modelo opaco a explicar, dicho modelo se entrena para emular el resultado de un modelo opaco.

Para evaluar qué tan bien un modelo sustituto replica el resultado de un modelo opaco podemos medir la similitud de sus predicciones mediante la medida *R-squared*:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_*^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{n}(\hat{y}_*^{(i)} - \overline{\hat{y}}^{(i)})^2}$$

Si la medida se aproxima a 1 entonces nuestro modelo sustituto replica el comportamiento del modelo opaco, mientras que si se aproxima a 0, el modelo sustituto tiene un comportamiento diferente al modelo opaco.

3.5 Métodos basados en la importancia de características

Definimos a la importancia de una característica como la variación de error en una predicción de un modelo ante la permutación de los valores de las características [33]. El concepto fue introducido al valorar la importancia de una característica por permutar sus valores y comparar la salida, pero se ha extendido y desarrollado dado algoritmos como LIME y SHAP.

3.5.1 Importancia de características por permutación

Refiriéndonos a *Permutation Feature Importance* como importancia de características por permutación, concepto introducido por [6] para encontrar cual característica contribuía más (

nombrado como "*variable importance*") a la clasificación de un *Random Forest*.

Si al modificar los valores de una característica el error de la predicción del modelo no cambia, se considera que la característica en cuestión no es importante para el modelo. En cambio si el error se reduce (o aumenta) la característica contribuye a la predicción de forma negativa (o positiva).

Conocer las características que son importantes para un clasificador nos ayuda a entender cuales de estás son aquellas en las que se fundamenta su proceso de decisión.

Dependiendo de la forma en que se realice el proceso también puede arrojarnos la importancia mutua de características, es decir, mostrar cómo la variación de una característica repercute en el impacto de otra característica en el proceso de clasificación.

El resultado de aplicar está metodología para obtener unidades interpretables es una lista de características, dependientes del modelo del cual se obtuvieron y que nos permiten comparar modelos de forma cualitativa.

Finalmente, para entender el funcionamiento de los siguientes algoritmos nos referiremos a cuatro conjuntos diferentes: X, X', Z, Z' y sus respectivos elementos $x \in X$, $x' \in X'$,$z \in Z$ y $z' \in Z'$.

Tanto X como X' hacen referencia al espacio de características originales y una codificación aplicadas a éstas mediante el mapeo $h_x^C()$ que equivale al $h_x^{-1}()$ en la literatura.

Mientras que Z y Z' son espacios de características interpretables generados mediante algún proceso de selección de características de una muestra X. El mapeo decodificador que devuelve los elementos no primados teniendo como entrada los elementos primados se representará como $h_x^D()$ y equivale al $h_x()$ de la literatura. En la Figura 3.2 se condensa la información sobre los elementos mencionados.

3.5.2 LIME

La primer metodología (explicador) a emplear es *Local Interpretable Model-agnostic Explanations* (LIME), el cual se trata de un algoritmo *pos-hoc* agnóstico que emplea modelos lineales para entregar explicaciones locales para una muestra de entrada y su correspondiente salida por parte del clasificador.

Los dos criterios en los que se basa LIME son que la explicación debe ser interpretable y la fidelidad local. El primero se logra proponiendo que los pesos de un modelo lineal (interpretable) $g = w_g z$ cuyas entrada es un mapeo binario de una entrada original x_i mediante la cual se

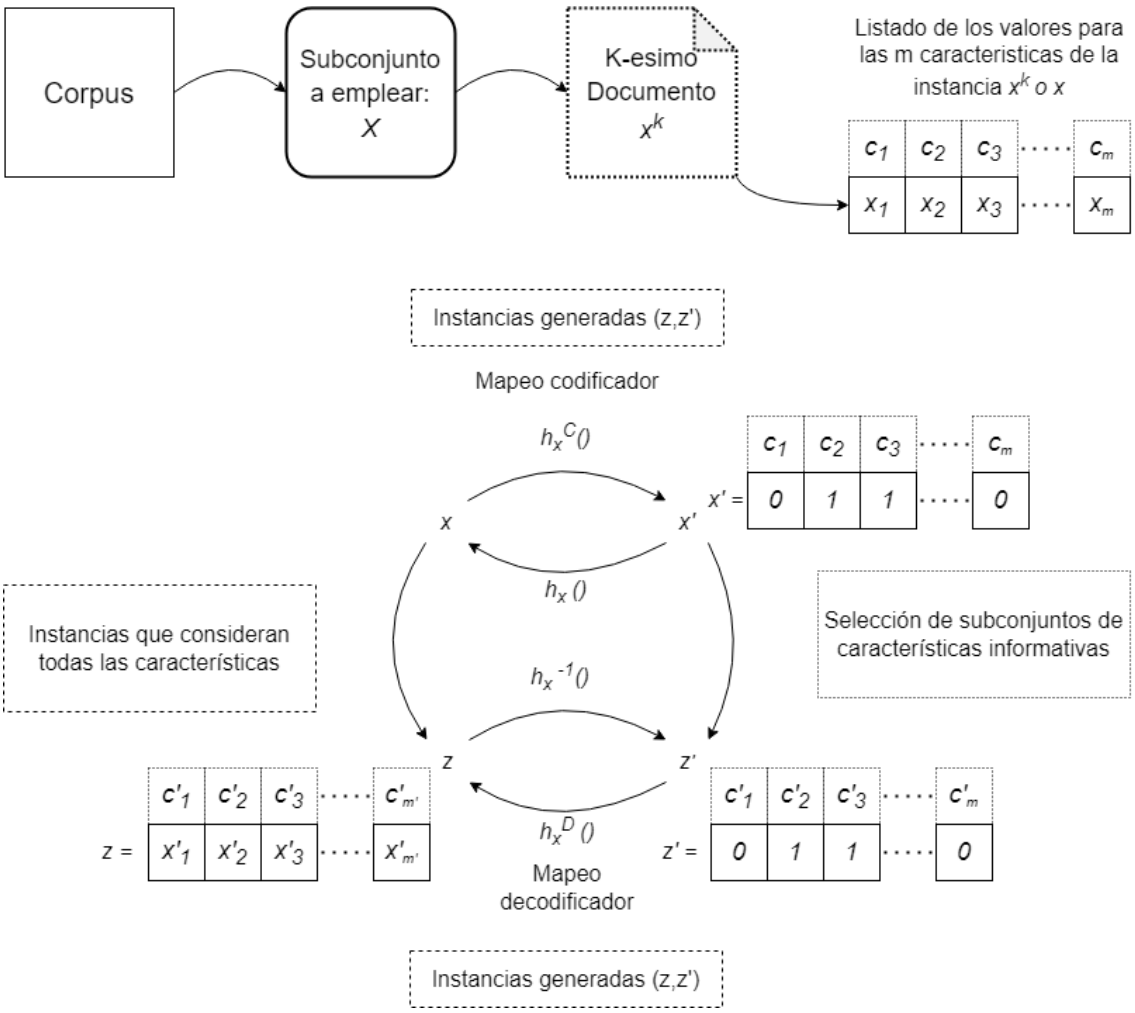


Figura 3.2: Comparativa de los elementos empleados por los algoritmos de importancia de características, con ejemplos del modelo vectorial que representan.

obtienen unidades interpretables z' que bien pueden ser las características de las entradas o subconjuntos de estas. Son estas unidades interpretables las que serán evaluadas en un vecindario de características perturbado Z_i donde cierto número de características interpretables serán removidos para valorar su contribución al modelo lineal sustituto g [18].

La cantidad u orden de elementos interpretados se nombra como *complejidad* del modelo y se denota como $\Omega(g)$.

La fidelidad local se logra mediante una medida de similitud del modelo lineal y el modelo original considerando también la similitud del conjunto de unidades interpretables con la entrada original. Esta similitud entre los elementos del espacio Z_i y el elemento x_i se calcula mediante un kernel gaussiano $\Pi_x(z) = e^{\frac{-D(x,z)^2}{\sigma^2}}$ donde el operador D representa una función de distancia (distancia

coseno, normal L2, etc.). Para medir la similitud del modelo sustituto con el modelo opaco se emplea entonces la relación:

$$\mathscr{L}(b,g,\Pi_x) = \sum_{z,z' \in Z} \Pi_x(z)\big(b(z) - g(z')\big)^2$$

El algoritmo entonces busca encontrar el modelo lineal g que minimice $\mathscr{L}$, pues entre más cercano sea su valor a 0, significa que el modelo sustituto encontrado y el modelo opaco cumplen con la fidelidad local. Finalmente son los pesos de g aquellos que nos dan una medida de la importancia de las características interpretables z_i [18].

A continuación se incluye en la Figura 3.3 una ilustración de los pasos realizados por el algoritmo LIME para explicar una muestra en el caso de un algoritmo clasificador b, empezando con la función de decisión de b, con su respectiva probabilidad de pertenencia a una clase especifica y elementos de entrenamiento (superior izquierda). La selección de una muestra a explicar x (superior centro). Generación de muestras z' entorno a x que se consideran poseen características interpretables c' (superior derecha). Obtención de un modelo lineal sustituto (linea roja punteada) para Z' que no pondera la distancia de los elementos de Z' con la muestra elegida x (inferior izquierda). Obtención de un modelo lineal sustituto g (linea roja punteada) que sí pondera y es el empleado por LIME (inferior centro). Resultado comparativo de g y b incluyendo la fidelidad local de g (inferior derecha).

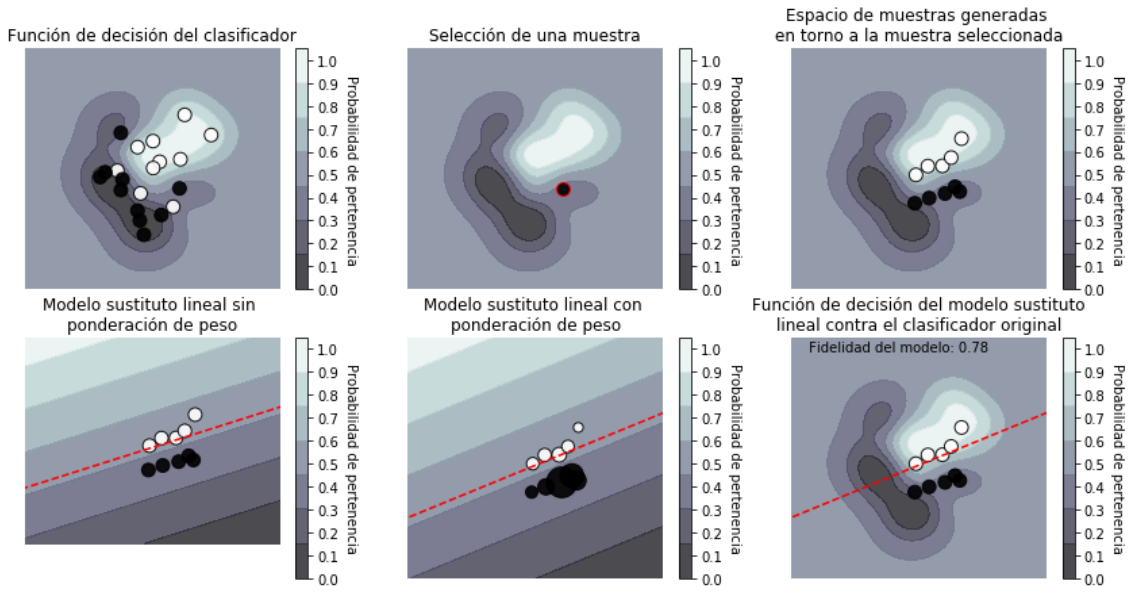


Figura 3.3: Diagrama del funcionamiento de LIME por pasos.

3.5.3 SHAP

El explicador *SHapley Additive exPlanations* (SHAP), se trata también de un algoritmo local que emplea un modelo lineal sustituto y aproxima los valores de Shapley de cada característica de dicho modelo. Es decir, de un modelo lineal de la forma $g(z') = w_0 + \sum_{i=1}^{M} w_i z'_i$ que busca sustituir un clasificador b entorno a una muestra x, se obtienen los coeficientes ϕ de que son aproximaciones de los valores de Shapley para cada característica de z' [23].

A diferencia de LIME, el vecindario a considerar por este algoritmo es el formado por todos los posibles subconjuntos de características de la(s) muestra(s) de entrada a considerar, esto porque emplea una ponderación de las características basada en los valores de Shapley de teoría de juegos cooperativos [23], [33].

La definición del valor de Shapley para la i-ésima característica de un modelo lineal g con características x queda definida mediante:

$$\phi_i = \sum_{S \subseteq F\{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \Big[g_{S \cup \{i\}}(x_{S \cup \{i\}}) - g_S(x_S) \Big]$$

Donde S representa un subconjunto del conjunto de características F que emplea el modelo g, $g_{S \cup \{i\}}$ contempla a S junto con la i-ésima característica mientras que g_S no incluye la i-ésima característica, así mismo las muestras $x_{S \cup \{i\}}$ y x_S respectivamente. Otra notación equivalente es $x_{\setminus i}$ y x_{-i} que indican que la i-ésima característica no es incluida y x_{+i} para indicar que sí está incluida.

En el caso que se emplee un modelo lineal sustituto g para un clasificador b se considera la fidelidad local del modelo sustituto al valorar la diferencia entre las predicciones pues se presupone una fidelidad local: $b(h_x^D(z')) \approx g(z')$ siempre que se cumpla una cercanía entre las muestras $z' \approx x'$.

En este caso el mapeo binario $x = h_x^D(x')$ vuelve a reducir las características por su presencia o ausencia del modelo de forma que x' equivale a una codificación *one-hot* de las entradas x. De esta forma los elementos interpretables z dependen de cómo se definan del algoritmo que obtenga a g. Ahora,la definición del valor de Shapley para la i-ésima característica del modelo lineal sustituto g que sustituye un clasificador b entorno a una muestra x queda definido como:

$$\phi_i(b, x) = \sum_{z' \subseteq x'} \frac{|z'|!(|x'| - |z'| - 1)!}{|x'|!} \Big[b_x(z') - b_x(z' \setminus i) \Big]$$

Donde la expresión $|z'|$ representa el número de características que no han sido enviadas a 0 en z', $z' \setminus i$ es el subconjunto de z' que no contienen la i-ésima característica interpretable y la diferencia $b_x(z') - b_c(z' \setminus i)$ muestra la variación del resultado de la predicción del modelo opaco ante la inclusión de una característica i que no está presente en z'.

Debido a que la complejidad del orden 2^M de obtener los valores Shapley exactos se emplea la aproximación SHAP, una atribución aditiva de características, es decir: al aproximar la salida del modelo $b(z) = b(h_x^D(z'))$ con $E[b(z)|z_S]$ es posible aproximar los valores de Shapley como:

$$\phi_i(b,x) = w_j(x_i - E[x_i])$$

donde w_j corresponde al peso del modelo lineal g que considera hasta la característica j y $E[x_j]$ el valor esperado de dicha característica, es decir la contribución en valores de SHAP de la i-ésima característica es la contribución de la i-ésima característica en el modelo sustituto menos el valor esperado de dicha característica [33].

Para representar mejor el funcionamiento de SHAP dado un modelo lineal sustituto g tenemos el siguiente Pseudocódigo 3.4:

Entradas:
Número de iteraciones K.
Muestra a explicar x.
Índice de la i-ésima característica a explicar i
Modelo lineal g
Conjunto de muestras Z.

Por cada $k \in 1 \ldots K$ **hacer**
  Seleccionar z, un elemento de Z aleatoriamente.
  Elegir una permutación aleatoria del orden de las características.
  Ordenar las muestras x y z respecto dicho orden.
  Construir las muestras x_i y $x_{\setminus i}$ con z
  Calcular la diferencia de la predicción $\phi_j^k = g(x_i) - g(x_{\setminus i})$
fin
Calcular el promedio de aproximaciones de valores de Shapley $\phi_j = \frac{1}{K}\sum_{k=1}^{K}\phi_j^k$
Regresar ϕ_j

Figura 3.4: Pseudocódigo del algoritmo SHAP para obtener la aproximación del valor de Shapley de la i-ésima característica de una muestra x

En general podemos decir que SHAP aproxima los valores de Shapley considerando independencia entre las características, promediando ordenes aleatorios de estas en un entorno con un número de muestras determinado por el modelo lineal sustituto.

3.5.4 AOPC

Para poder evaluar cuán tan informativas son los *tokens* obtenidos por los métodos antes mencionados existen diferentes aproximaciones basadas en retirar palabras del documento y evaluar

cierto cambio del clasificador, por ejemplo el *Switching point* representa el porcentaje de palabras que deben ser eliminadas del documento para que la predicción cambie de la originalmente predicha por el clasificador.

En este trabajo se toma como métrica principal el *Area Over the Perturbation Curve* (AOPC en adelante) propuesta en [19] por W. Samek et. al., y empleada para LIME en la tarea de clasificación de textos en el artículo [29]. La definición de AOPC se ilustra en la siguiente ecuación:

$$AOPC = \frac{1}{K+1}\left\langle \sum_{k=1}^{K} \hat{y}(x) - \hat{y}(x_{\setminus 1\ldots k}) \right\rangle_{p(x)}$$

Donde la expresión $x_{\setminus 1\ldots k}$ representa el subconjunto de características sin K características, siguiendo siguiendo un ordenamiento especifico $1\ldots k$ que considera las características más importantes primero (*MoRF* en adelante) tal como se indica en [19]. Además la expresión $\langle\dot{\rangle}_{p(x)}$ significa el promedio sobre todas las instancias consideradas. Donde el subconjunto $x_{\setminus 1\ldots k}$ puede no ser el mismo para cada instancia en particular.

En síntesis, el AOPC mide la variación de la probabilidad predicha del clasificador al retirarle a la instancia a clasificar las características más importantes una por una y en orden. Entre más variación haya por parte de la predicción del clasificador $\hat{y}$ se confirma que dicha característica era importante.

3.6 Discusión

A lo largo de este capitulo se han desarrollado algoritmos entorno a la importancia de características, sin embargo no se ha hecho la aclaración si este procedimiento debe realizarse sobre el conjunto de entrenamiento, de validación o prueba.

La diferencia entre obtener la importancia de características de cada uno de estos conjuntos reside en qué información buscamos saber. La forma en la que el modelo atribuye importancia a cada característica para realizar sus predicciones se obtiene al explicar el conjunto de entrenamiento. Mientras que explicar cualquiera de los otros dos conjuntos nos indica qué características contribuyen más en el desempeño del clasificador pese a no necesariamente ser vistas antes [33].

Debe tenerse en cuenta que aplicar un explicador sobre del conjunto de entrenamiento tiene un fenómeno similar a tratar de evaluar el desempeño del clasificador sobre el conjunto de entrenamiento: puede existir la tendencia de atribuir importancia a características que debido a su distribución en el conjunto de entrenamiento terminan siendo relevantes.

Si bien esto último pareciera indicar que no vale la pena utilizar un explicador en el conjunto de entrenamiento sería una forma de saber si nuestro clasificador considera características inútiles como relevantes pues sería un indicador del mal entrenamiento de este, pese los resultados obtenidos en métricas de evaluación.

CAPÍTULO 4

TRABAJO RELACIONADO

En la primer parte de este capitulo abordaremos el estado del arte de del problema de clasificación de noticias enfocado al corpus *20Newsgroups* o *20News* para abreviar. El uso de diferentes explicadores lineales basados en la importancia de características aplicados a la tareas de clasificación será abordado en la segunda parte.

4.1 Relacionados a la clasificación de 20NewsPapers

La tarea de clasificación de textos tiene como finalidad asignar etiquetas predefinidas a documentos dependiendo de la información recabada de documentos de los cuales se conoce su etiqueta. De entre estas tareas de clasificación de textos la clasificación automática de noticias por tópico (*NC, News clasification* en adelante) tiene entre otros objetivos regresar una forma de indexar noticias y presentarlas al lector de acuerdo a sus intereses [16].

La *NC* es un problema clásico del área de procesamiento de lenguaje natural como se aprecia en el artículo de [2] al cual se le atribuye la recopilación del corpus *20 News* según [44] [13] del cual es extraído actualmente.

En esta sección abordaremos los diferentes algoritmos con los que se ha abordado el problema, el propósito de dichos trabajos, metodologías, resultados y finalmente una comparativa con el trabajo presentado.

4.1.1 Algoritmos empleados

A lo largo de la literatura nos encontramos con trabajos que emplean *Lineal regresion*[2], *Logistic regresion*[41], [37], *Decision trees*[7], [37], *Naive Bayes*[45], [41], *Self organizing maps*[7], *Clasificador Rocchio* [9], *KNN* [4], [9], [15], *SVM*[4], [5], [45], [16], [37], [41], *MLP* [4], [7], [37], *LSTM*[38], [39] y *Transformers*[34], [42] combinaciones de algoritmos (Tal como lo es [9], [45], [30], [42]), como algoritmos típicos para la solución de *NC*.

Algunos de estos trabajos en especifico trabajan con el corpus *20 News* para tratar de clasificarlo o lo emplean para presentar contraste con otros corpus, lo que deriva en comparativas de resultados principalmente enfocadas a las métricas.

Dentro de estos trabajos encontramos diversidad en cuanto la selección de subconjuntos *20 News* para trabajar con 2 clases [29], [39],[5], 4 clases [38], 5 clases [16], 8 clases [5] y 20 clases[9],[45],[15],[32],[37],[38],[42],[41]. Así como diferentes formas de preprocesar el texto, obtener de las características a usar, elegir cantidad de características a usar y determinar proporción adecuada al dividir los conjuntos de datos. Junto con variaciones de qué métrica reportar para la evaluación y el procedimiento adecuado para la selección de los hiperparámetros de los clasificadores.
Si bien en el trabajo de [4] y [9] se menciona la necesidad de emplear alguna métrica particular para la evaluación entre diferentes clasificadores y diferentes corpus trabajados como lo son *t-test* y *resampled t-test* es común presentar una tabla con alguna métrica como forma de mostrar los resultados independientemente del clasificador, corpus y características empleadas dentro del mismo trabajo.

Otro elemento en común en la mayoría de los trabajos revisados en el uso de tf-idf como mecanismo para la vectorización del documento, incluso en trabajos recientes como [41]. Sin embargo el trabajo de [15] compara en el uso de *embeddings* como alternativa a este proceso.

4.1.2 Resultados

Los resultados y objetivos de los trabajos se presentan de manera diversa: determinar exactitud en dependencia del tamaño del conjunto de entrenamiento [5],[45], el error de la clasificación [7], comparar *embeddings* con tf-idf [15], comparar modelos híbridos [9] y modelos clásicos [37],[41]. Además de las diferentes métricas reportadas e incluso la versión de *20 News*[1] o de otros corpus empleados.
De forma general podemos encontrar que modelos basados en SVM logran una exactitud en el conjunto de prueba de entre 70% [37] y 96.5% [41], para el caso de algoritmos KNN 56% de error

[1]Existen 3 versiones del dataset, todas disponibles en http://qwone.com/ jason/20Newsgroups/

en las clasificaciones [15] [2] y las MLP logran entre 70% [37] y 75% [7]. Empleando en todos los casos las 20 clases disponibles.

4.2 Relacionados a la explicación de clasificadores

En esta sección de trabajo relacionado se abordarán diferentes algoritmos explicadores que buscan explicar modelos de clasificadores de texto no necesariamente enfocados al problema de etiquetar *20News*.

4.2.1 Trabajos, empleo y uso

Los trabajos enfocados a explicar algoritmos de clasificación surgen por la necesidad de proveerle al usuario razones para que este confié en las predicciones o que pueda explicarlas legalmente como se menciona en [14]. En dicho trabajo se describe como los algoritmos de *Decision Trees*, *Classification Rules*, *Decision Tables*, *KNN* y *Bayesian Network Classifiers* son modelos interpretables.

Si bien se menciona cómo los clasificadores que emplean *KNN* pueden ser explicados listando las características de los vecinos más cercanos cuando el problema tiene una alta dimensionalidad dicha lista resulta ambigua. La aproximación sugerida emplea una ponderación de las características en función de su impacto en la predicción (importancia) para obtener solo las características relevantes.

Entre trabajos tenemos aquellos enfocados a la clasificación de textos explicable [17], [28], [30], [29], [39] con diferentes aproximaciones sobre distintos algoritmos clasificadores. Y también trabajos con un enfoque agnóstico respecto el modelo opaco [18],[23],[17],[29].

Anteriormente mencionamos enfoques agnósticos como lo son LIME y SHAP, algoritmos explicadores de propósito general que buscan ofrecer una lista de características importantes para la clasificación de una salida empleando un modelo sustituto. Similar a estos se encuentra LORE *LOcal Rule-based Explanations* [26] donde dada una instancia de entrada particular se genera un conjunto de instancias en el vecindario entorno a esta mediante un algoritmo genético y posteriormente se emplea un *tree classifier* para extraer la explicación en forma de las reglas seguidas por dicho algoritmo interpretable.

En dicho trabajo se pone en duda la funcionalidad de LIME, mencionando como el vecindario que se genera debe de cumplir ciertas condiciones y cómo el hiperpárametro de cuantas instancias se

[2] Este trabajo emplea solo las 500 palabras más frecuentes de cada documento para *20 News*

generan entorno afecta al funcionamiento del algoritmo.

A diferencia de otros métodos no se emplea la importancia de características y la forma de presentar la explicación son dos conjuntos de reglas, aquellas que deben cumplirse para que la predicción sea como el modelo opaco b (r en la notación del artículo) y aquellas que de ocurrir invertirían la clasificación del modelo opaco $\hat{y}$ (Φ en la notación del artículo). La Figura 4.1 ilustra un ejemplo de dichas reglas.

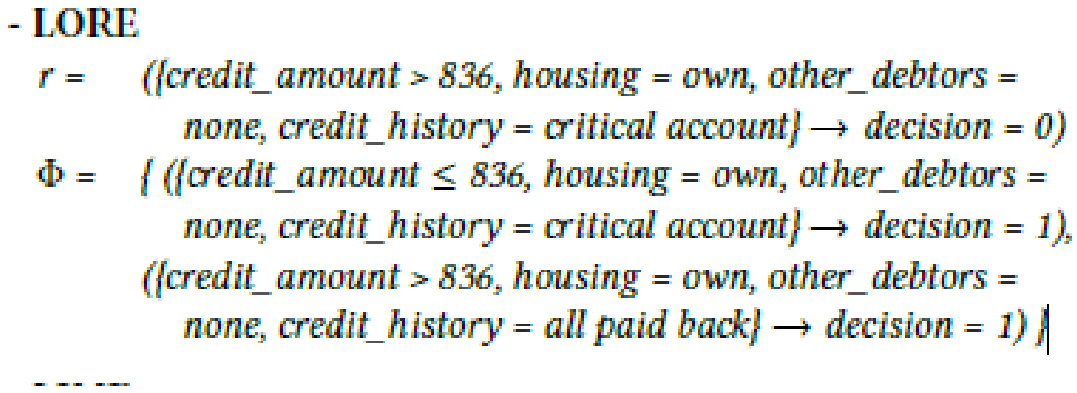


Figura 4.1: Ejemplo de reglas obtenidas mediante LORE

Propuestas de arquitecturas de redes neuronales con un *enc* y *gen* donde el primero se refiere al clasificador y el segundo a un algoritmo que extrae subconjuntos de las entradas para entregar aquel subconjunto que produzca una respuesta del *enc* lo más semejante a la entrada con ciertas condiciones como se realiza en [17]. En la Figura 4.2 se ilustra un ejemplo de cómo se presenta la explicación.

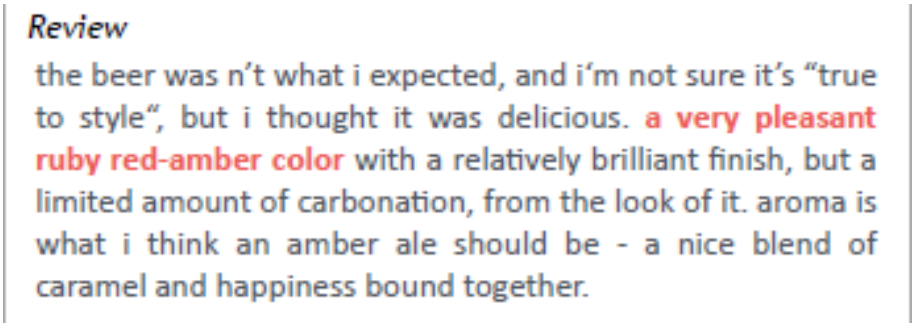


Figura 4.2: Explicación generada por un modelo *enc* y *gen* donde el texto remarcado representa la razón detrás del resultado del clasificador.

Otra aproximación para el problema de multietiqueta se presenta en [28], donde se propone el algoritmo de *CALM* o *Convolutional Attention for Multi-Label classification*. Este algoritmo sugiere que en lugar de emplear un *pooling* en la capa convolucional es posible emplear una

capa de atención y con los pesos de dicha capa ponderar las características para listarlas como importantes para el clasificador en su toma de decisiones.

Se mencionó antes como *KNN* podría ser un algoritmo interpretable y dicha idea se emplea en [30] para obtener cuales son las características de importancia para un modelo de red neuronal convolucional y un modelo de LSTM. Se emplea el algoritmo *DkNN* donde la predicción no se realiza mediante un clasificador *softmax* en la salida de la red. Sino mediante la similitud de los elementos más cercanos (Se aplica *KNN*). El resultado de la explicación es presentado mediante un *Saliency Map* como se ilustra en la Figura 4.3

Prediction	Input	Saliency Map
Contradiction	Premise	a young boy reaches for and touches the propeller of a vintage aircraft.
	Hypothesis	a young boy swims in his pool.
Contradiction	Premise	a brown dog and a black dog in the edge of the ocean with a wave under them boats are on the water in the background.
	Hypothesis	the pets are sleeping on the grass.
	Premise	man in a blue shirt standing in front of a structure painted with geometric designs.
Entailment	Hypothesis	a man is wearing a blue shirt.
Entailment	Hypothesis	a man is wearing a black shirt.

Color Legend Positive Impact Negative Impact

Figura 4.3: Resultado de la explicación del algoritmo CALM, donde se remarcan las palabras que tuvieron un impacto positivo o negativo.

Se termina esta sección mencionando los trabajos relacionados con la clasificación de *20News* que buscan explicar el resultado de la clasificación.

En [29] se emplean *MLP* y *Linear Regresor* para la clasificación binaria de las clases *Christianity* y *Atheism* proponiendo LIME como algoritmo explicador y comparándolo con *word omission*, *First derivative salency* y selección aleatoria como baseline. La robustes de LIME es puesta a prueba al variar los hiperparámetros del número de características empleadas para explicar (complejidad) y el número de instancias generadas para obtener la explicación.

Respecto métodos basados en atención tenemos que [39] se menciona cómo los encoders de una LSTM y los pesos de la capa de atención no equivalen a características importantes ni a una ponderación de de estas respectivamente. Como alternativa a dicha arquitectura LSTM+atención se propusieron las arquitecturas *Diversity LSTM* y *Orthogonal LSTM* que solventan el problema de 'alta conicidad'. Entre las diferentes tareas de NLP revisadas, para la clasificación binaria de

texto se emplean las categorias de *Baseball* y *Hockey* del corpus de *20News*.

4.3 Enunciado de propuesta

El enfoque del presente trabajo está orientado a aproximarse al desempeño reportado de los diferentes algoritmos clasificadores para el corpus *20News* incluyendo en el reporte todas las métricas mencionadas en el capitulo 2. Empleando validación cruzada con búsqueda sobre una cuadricula de hiperparámetros considerando como criterio de selección la métrica de exactitud.

La finalidad de esta primer etapa es tener un modelo cuyo funcionamiento se considere "correcto" desde la visión de estas métricas y el análisis de error con validación cruzada para poder emplear los algoritmos explicadores en clasificadores que serían "aceptados".

Obtenidos los clasificadores se emplearan los algoritmos explicadores LIME y SHAP sobre del conjunto de entrenamiento, obteniendo así explicaciones locales para cada instancia explicada. La evaluación y selección de qué conjunto de hiperparámetros para cada explicador se realizará comparando las métricas de *R-squared* y AOPC.

Se representarán las explicaciones de ciertos elementos a nivel local, junto con la tabla de *tokens* y sus valores de importancia obtenidos.

CAPÍTULO 5

PROPUESTA DE SOLUCIÓN

En este capitulo se describirán los métodos y modelos empleados para el preprocesado, clasificación y obtención de explicaciones, así como las especificaciones de estos. El procedimiento, a grandes rasgos, se describe a continuación.
Tras la carga y preprocesado de los archivos de entrada se obtienen los documentos a clasificar, considerando los casos con *stopwords* y sin *stopwords*. Dichos documentos serán clasificados comparando tres diferentes clasificadores que logran los resultados obtenidos en el estado del arte.
De los clasificadores se obtienen las características más importantes de manera local mediante LIME y SHAP, aplicándolos en el conjunto de entrenamiento. Los resultados de los explicadores son evaluados mediante las métricas de *R-squared* y AOPC definidos en 3.4 y 3.5.4 respectivamente.

Finalmente se ilustran los resultados mediante el remarcado de palabras en el texto preprocesado mediante un *Heatmap*, junto con los resultados de las métricas y el texto sin preprocesar.

5.1 Preprocesado

Como se mencionó en el capitulo anterior, el corpus *20News* contiene 20 categorías diferentes con una distribución casi homogénea de elementos por etiqueta, como se resume en la Tabla 5.1.

Categoría	Número de noticias	Categoría	Número de noticias
alt.atheism	799	rec.sport.hockey	999
comp.graphics	973	sci.crypt	991
comp.os.ms-windows.misc	985	sci.electronics	981
comp.sys.ibm.pc.hardware	982	sci.med	990
comp.sys.mac.hardware	961	sci.space	987
comp.windows.x	980	soc.religion.christian	977
misc.forsale	972	talk.politics.guns	910
rec.autos	990	talk.politics.mideast	940
rec.motorcycles	994	talk.politics.misc	775
rec.sport.baseball	994	talk.religion.misc	628

Tabla 5.1: Distribución de documentos por categoría en *20News*

Algunas de las clases tienen más semejanza entre sí que otras. En la página *20 Newsgroups*[44] se presenta la siguiente Tabla 5.2 que ilustra un agrupamiento con 6 categorías generales: computación, recreación, ciencia, política, religión y miscelánea.

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Tabla 5.2: Agrupamiento de categorías presentado en *20 Newsgroups*

Existen tres diferentes versiones de este corpus. La versión original contiene 19,997 noticias. La segunda versión contiene 18,846 documentos que han sido ordenados por fecha y se han retirado algunos duplicados así como algunos *headers*. La tercer versión (y empleada en este trabajo) tiene 18,828 noticias, no contiene duplicados y únicamente tiene los *headers From* y *Subject*.

Todos los documentos coinciden con tener los *headers From* y *Subject* en las primeras dos lineas. Dentro de la paquetería de `scikit learn` se ofrecen versiones sin las *signature* y *footer* pero como mencionan, son versiones no "exactas" ya que estos últimos elementos no son homogéneos a lo largo de los documentos.

La siguiente etapa es el preprocesamiento de los documentos, cuyo objetivo es obtener un modelo de espacio vectorial. La Figura 5.1 ilustra de forma general este proceso.

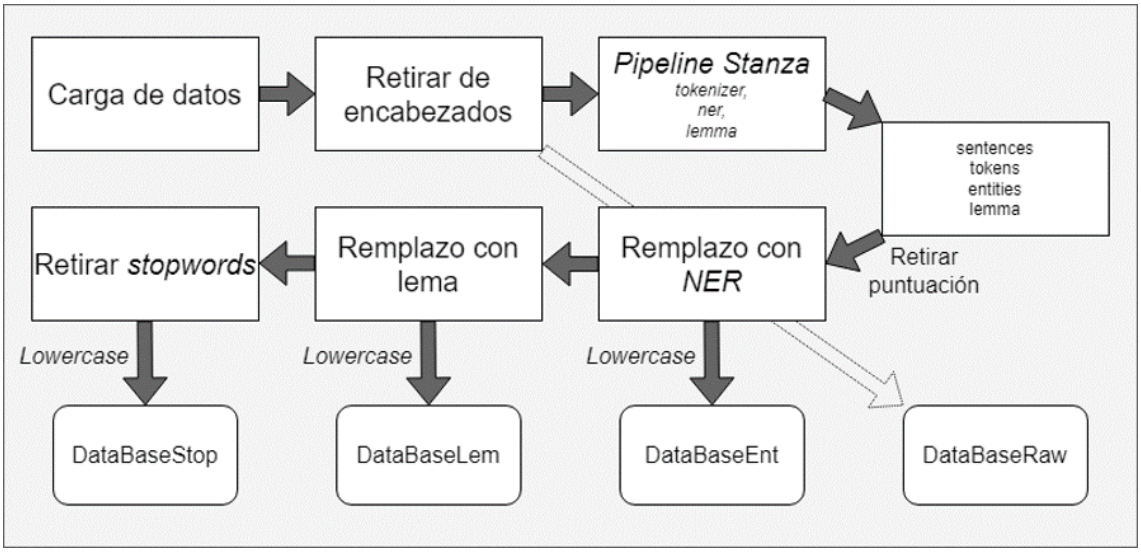


Figura 5.1: Preprocesamiento aplicado a *20News*

Los documentos son cargados con su codificación *'windows-1252'* y posteriormente se aplica un *pipeline* con `Stanza` [40] con los procesos de `tokenize, ner y lemma` (tokenizador, reconocedor de entidades nombradas y lematizador respectivamente), con el cual obtenemos el documento dividido en oraciones (*sentences*), *tokens*, palabras, entidades y lemas.

A cada documento se le retiran los signos de puntuación y se guardan en tres diferentes bases de datos (`DataBaseEnt, DataBaseLem` y `DataBaseStop`) con codificación `utf-8` que contienen respectivamente las entidades remplazadas por entidad_ETIQUETA, las palabras por su respectivo lema y retirando las *stopwords* incluidas en `Spacy` [36]. Los modelos empleados para cada proceso son: OntoNotes (NER), EWT (Lemmatizador) y en_core_web_sm (Stopwords)

Tras cargar la base de datos de los documentos preprocesados, se divide en diferentes conjuntos: Entrenamiento, Validación y Prueba. El conjunto de desarrollo corresponde al 85% de los documentos, dejando 15% de los documentos para el conjunto de prueba. El conjunto de desarrollo se divide en un 72.25% para el entrenamiento de los clasificadores y un 12.75% para el conjunto de validación.

El trabajo incluye dos etapas que serán evaluadas de forma diferente (clasificación y explicación) de forma que el conjunto de validación se reserva para validar los resultados de la explicación, mientras que el conjunto de desarrollo será dividido nuevamente en la etapa de validación del clasificador al emplear validación cruzada.

El proceso de vectorizar nuestros documentos emplea el `CountVectorizer` y un `TfidfTransformer` de la paquetería `scikit learn` [13]. Específicamente el `CountVectorizer` nos regresa una ma-

triz dispersa con el conteo por palabra que toma en consideración solo unigramas y una *document frequency* de 1 (toda palabra que no esté presente en el conjunto de entrenamiento será descartada posteriormente). Mientras que el `TfidfTransformer` nos regresa una ponderación que corresponde a la siguiente expresión para el i-ésimo token:

$$tfifd_i = \left(1 + \log(tf_i)\right) \times \left(\log\left(\frac{1+n}{1+df_i}\right) + 1\right)$$

Donde tf_i e idf_i se refieren a la *term frequency* y la *inverse document frequency* del token i y n es el número de documentos en cuestión. Notemos que se han empleado los ajustes *sublinear term frequency* y *smooth inverse document frequency* al aplicarle los respectivos logaritmos a los valores originales. Finalmente, a la expresión anterior se le aplica la norma L2 obteniendo una matriz dispersa sin divisiones entre cero y normalizada. Obteniendo así el modelo de espacio vectorial de nuestros documentos.

5.2 Entrenamiento de clasificadores

Implementando un *pipeline* que incluya el preprocesamiento anterior se entrenan tres diferentes clasificadores para las bases de datos con *stopwords* y sin *stopwords* de manera que la entrada sean lineas de texto. La selección de los hiperparámetros de cada clasificador se realiza mediante una exploración de cuadricula de hiperparámetros tomando aquel modelo que tuviera mejor desempeño respecto la precisión. El entrenamiento de los diferentes modelos de clasificadores da como resultado diferentes modelos opacos b que explicaremos más adelante.

5.2.1 Hiperparámetros de cada clasificador

Los hiperparámetros considerados para optimizar cada clasificador junto con sus respectivos valores se resumen en la siguiente tabla, dónde $|tokens|$ representa la cantidad de *tokens* y σ la varianza de los valores.

Modelo - Hiperparámetro		Valores	
KNN - número de vecinos	3	5	15
KNN - tamaño de la hoja	25	50	100
SVM - penalización "c"	1	10	100
SVM - γ	$\frac{1}{\|tokens\|}$	$\frac{1}{\|tokens\|*\sigma}$	
MLP - neuronas de en la capa oculta	100	128	
MLP - función de activación	*ReLU*	*tanh*	
MLP - tamaño del *batch*	200	500	

Tabla 5.3: Hiper-parámetros de los clasificadores empleados.

En cada modelo de clasificador con ciertos hiper-parámetros se obtienen las métricas de precisión, exhaustividad y norma F1 con sus mediciones micro y macro, además de la exactitud y del soporte.

5.2.2 Validación Cruzada

La selección de hiper-parámetros emplea una validación cruzada con 10 iteraciones considerando como criterio de selección la métrica de exactitud. Si bien se guardan todos los modelos entrenados obtenidos, solo se trabajará con el que haya reportado una mayor precisión. Esta validación utiliza el subconjunto de entrenamiento nombrado anteriormente y lo subdivide en cada iteración entrenando con $\frac{9}{10}$ y reportando métricas con el $\frac{1}{10}$ restante. Con esto obtenemos modelos de clasificadores validados, de forma que los hiperparámetros correspondan a ser los que mejor se desempeñen en el conjunto de validación.

5.3 Explicadores

Se proponen realizar dos pruebas con los resultados de los explicadores, la primera para obtener explicaciones válidas a nivel local de nuestro modelo clasificador, y la segunda para obtener explicaciones significativas. Dichas pruebas son necesaria debido a que no todo resultado del explicador es consistente (no se obtiene el mismo subconjunto de características por iteración), ni significativo (las características obtenidas no necesariamente afectan a la predicción), ni fiel localmente (el modelo sustituto representa al modelo opaco).

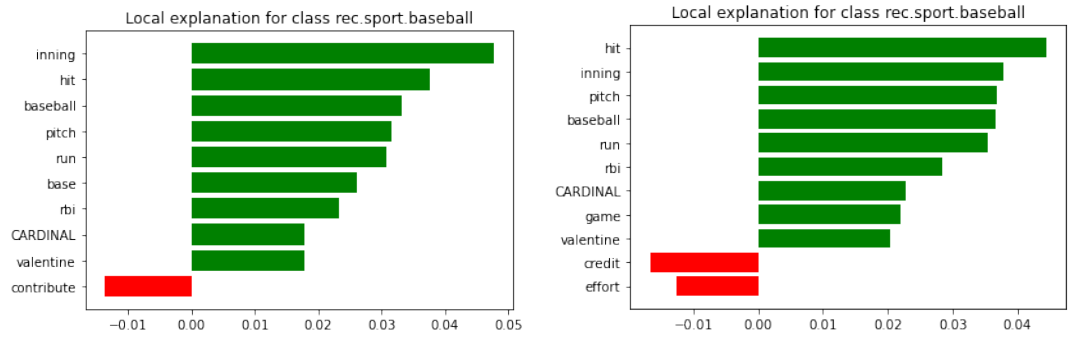
Finalmente se ilustran las formas en las que pueden ser presentados los resultados de los explicadores. Ya que estos resultados pueden mostrarse por instancia (localmente) o por categoría (aproximación global). Estás representaciones deben incluir las métricas de evaluación de los explicadores así como el texto original explicado.
Esta última prueba es una comparativa cualitativa de cómo varía el resultado del explicador entre dos diferentes instancias y cómo varían las instancias contra el promedio de las mismas y su propósito está en comparar la diferencia de las explicaciones de cada explicador.

5.3.1 Obtención de características importantes por instancia

Al implementar explicadores para obtener la importancia de características por permutación, encontramos que su resultado varía de ejecución en ejecución, incluso en una misma instancia. Por esto se busca una validación o ajuste de los hiper-parámetros de los explicadores de forma que se tenga consistencia.

En el caso de LIME, los hiperparámetros a ajustar consisten en el número de características resultantes objetivo y el número de muestras perturbadas generadas entorno a la instancia a explicar. Por su parte, SHAP tiene como hiperparámetros cuáles y cuántas instancias serán utilizadas para la perturbación de la instancia a explicar, el número de repeticiones que el algoritmo permutará las características y el coeficiente de regularización del algoritmo para obtener su modelo sustituto.

La repercusión que tienen la variación de estos hiperparámetros varía entre explicadores, a continuación, en las Figuras 5.2, 5.3 y 5.4 se ilustran de manera visual la variación de número de perturbaciones en LIME y la variación de las instancias vecinas para generar las perturbaciones.



$|Z|$ = 10 $|Z|$ =11

Figura 5.2: Variación del número de características a emplear varía la magnitud de Z. Notemos que los resultados han variado entre iteraciones.

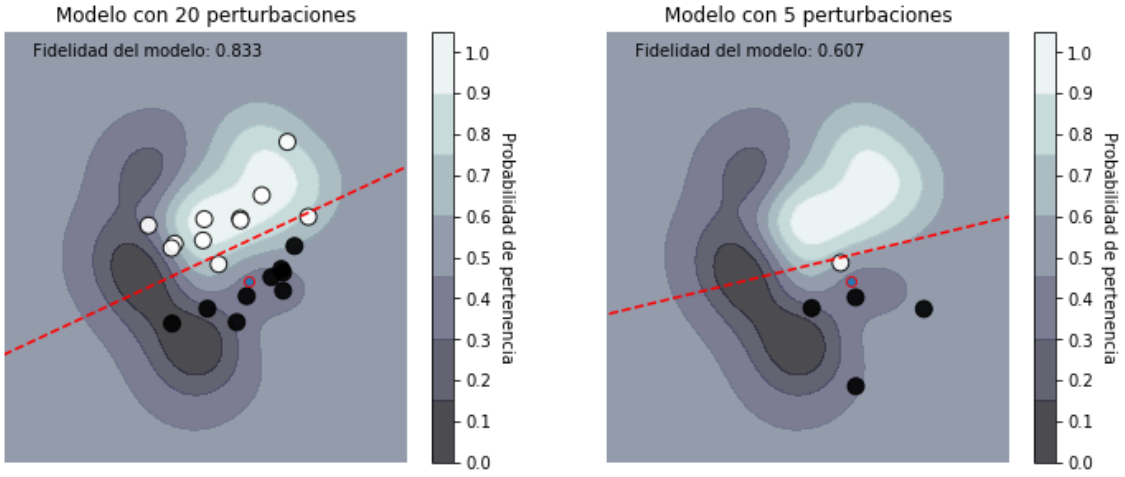


Figura 5.3: Variación del número de perturbaciones a emplear en LIME, notemos la variación los modelos sustitutos

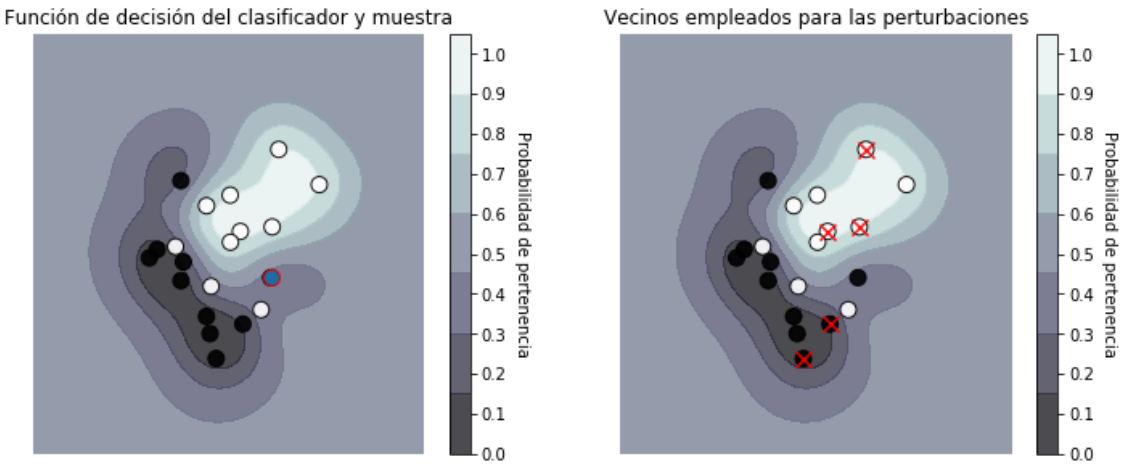


Figura 5.4: Variación de vecinos para la perturbación en SHAP, empleando solo aquellos marcados. Notemos que cuántas y cuáles instancias vecinas a la instancia son usadas para a explicar son un hiperparámetro a ajustar

En el caso de la variación del hiperparámetro del número de repeticiones de SHAP entre mayor número de repeticiones se obtiene una mejor aproximación de los valores de Shapley mediante los *Shap values*.

5.3.2 Métricas para evaluar la explicación

Para obtener un resultado reproducible y válido se evaluarán las explicaciones obtenidas por instancia. Las métricas empleadas son la contribución de cada instancia al AOPC, que nos indica cuánta influencia tienen las características obtenidas para la predicción del clasificador y el *R-squared* para obtener una medida de fidelidad local, para indicarnos si nuestro modelo sustituto se comporta como nuestro clasificador para dicha instancia.

5.3.3 Implementación de los explicadores

Dado un subconjunto de documentos (entrenamiento, prueba, validación) podemos obtener las características que más afectan la predicción del clasificador por cada documento (instancia) mediante los algoritmos LIME y SHAP. El resultado consiste en una lista de *tokens*, cada uno de ellos asociado con un número que representa de forma numérica la importancia del *token* en cuestión.

Dadas estas listas por cada documento, podemos obtener las métricas *AOPC* y *R-squared*, y dado cierto conjunto de hiperparámetros podemos evaluar cuál explicador funciona mejor para un subconjunto de documentos. Es decir, podemos aplicar una validación obteniendo el promedio,

varianza, mínimos y máximos de las métricas.

Un problema sobre de esta validación es sobre qué conjuntos debería realizarse, ya que el conjunto de entrenamiento nos refleja el comportamiento del clasificador (qué es lo que el clasificador considera importante al ser entrenado), mientras que los otros conjuntos nos muestran qué es lo que se considera importante por el clasificador aunque no se trate de una muestra de entrenamiento.

En este trabajo se toman ambos enfoques a manera de comparativa, es decir, obtener las listas de características importantes de cada categoría del conjunto de entrenamiento de los clasificadores y obtener las métricas sobre dicho conjunto, es decir, explicar el entrenamiento. De igual manera, obtener dichas listas de un conjunto de validación y obtener las métricas sobre el conjunto de prueba.

No se toman todos los documentos de cada subconjunto, sino se utiliza una muestra como aproximación del comportamiento. La cantidad de documentos a considerar es un hiperparámetro adicional.
Otra consideración tomada en este trabajo es que la importancia obtenida puede ser tanto positiva (el *token* contribuye a la predicción de la categoría correcta) como negativa (el *token* actúa como ruido para el clasificador). De forma que de la lista de *tokens* obtenidos por cada instancia utilizaremos un ordenamiento *MoRF* para el calculo del AOPC. Un efecto de esta consideración es que si bien podemos indicar el número de características a obtener, no podremos asegurar que todas ellas sean positivas, de forma que tendremos una cota superior de la cantidad de nuestros conjuntos de *tokens* más no un valor mínimo.

En la Tabla 5.4 se lista la cuadricula de los hiperparámetros sugeridos de los explicadores, algunos hiperparámetros son algoritmos y tienen a su vez otros hiperparámetros.

Hiperparámetro	LIME	Hiperparámetro	SHAP
Número de características objetivo	20	Número de instancias para generar las perturbaciones	2, 8
Número de muestras generadas	500, 1000, 5000	Número de repeticiones del algorítmo aproximador de Shapley Values	100, 800

Hiperparámetro	LIME	SHAP
Algoritmo para obtener el modelo sustituto g	regresión Ridge	regresión Lasso
Coeficiente de regularización	1	1e-4
Ordenamiento para AOPC	MoRF	MoRF
Instancias explicadas	300	300

Tabla 5.4: Hiperparámetros implementados en diferentes explicadores.

5.4 Presentación de las explicaciones

Finalmente, la última parte del desarrollo se enfoca en presentar explicaciones al usuario implementando las características interpretables consideradas importantes (listas de *tokens* obtenidas por instancias y por categorías).

La primer representación consiste en remarcar la importancia de cada *token* en el texto original según el resultado obtenido por instancia o por categoría, es decir la prominencia de los *tokens*. Esta representación debe incluye la fidelidad local y el AOPC obtenido. Las figuras 6.5 y 6.6 son ejemplos de estás.
El resto de representaciones consisten en gráficos que ilustren la importancia obtenida por el método explicador de cada *token* por documento o por categoría de documentos. El gráfico de barras permite comparar de forma visual la diferencia de la importancia obtenida entre diferentes *tokens* en un mismo documento o categoría como se ilustran en las Figuras 6.3, 6.4 en el capitulo siguiente.

CAPÍTULO 6

RESULTADOS EXPERIMENTALES

En este capitulo se desarrollarán las propuestas de solución descritas en el capitulo anterior, mostrando los resultados y una interpretación de los mismos. Al igual que el capitulo anterior, las secciones de éste cubrirán el preprocesado, la implementación de los clasificadores y la implementación de los explicadores con sus respectivas pruebas.

6.1 Preprocesado

Tras realizar el preprocesado al corpus *20News* se obtuvieron cuatro diferentes bases de datos, de las cuales se trabajará con *DataBaseStop* que no contiene *stopwords* y con *DataBaseLem* que contiene *stopwords*. Esto con la finalidad de que en pruebas posteriores sea posible evaluar la importancia de las *stopwords* para nuestros clasificadores.

La reducción del vocabulario era uno de los objetivos de la aplicación de este proceso y puede ser resumida en la Tabla 6.1. Además en las Figuras 6.1 se ilustra una comparativa de un documento original y su versión en *DataBaseStop*, si bien la cabecera se retira sin problemas, el pie de página persiste a lo largo de los documentos. Notemos también el resultado del etiquetado de entidades concatenando los tokens de la entidad y agregando la etiqueta correspondiente en mayúsculas.

Versión de la base de datos	Tamaño del vocabulario	Máximo de tokens por documento	Tokens promedio por documento
DataBaseRaw	197,791	13,706	148
DataBaseEnt	146,314	3,337	125
DataBaseLem	143,597	3,346	115
DataBaseStop	143,301	3,282	78

Tabla 6.1: Resultados del preprocesado

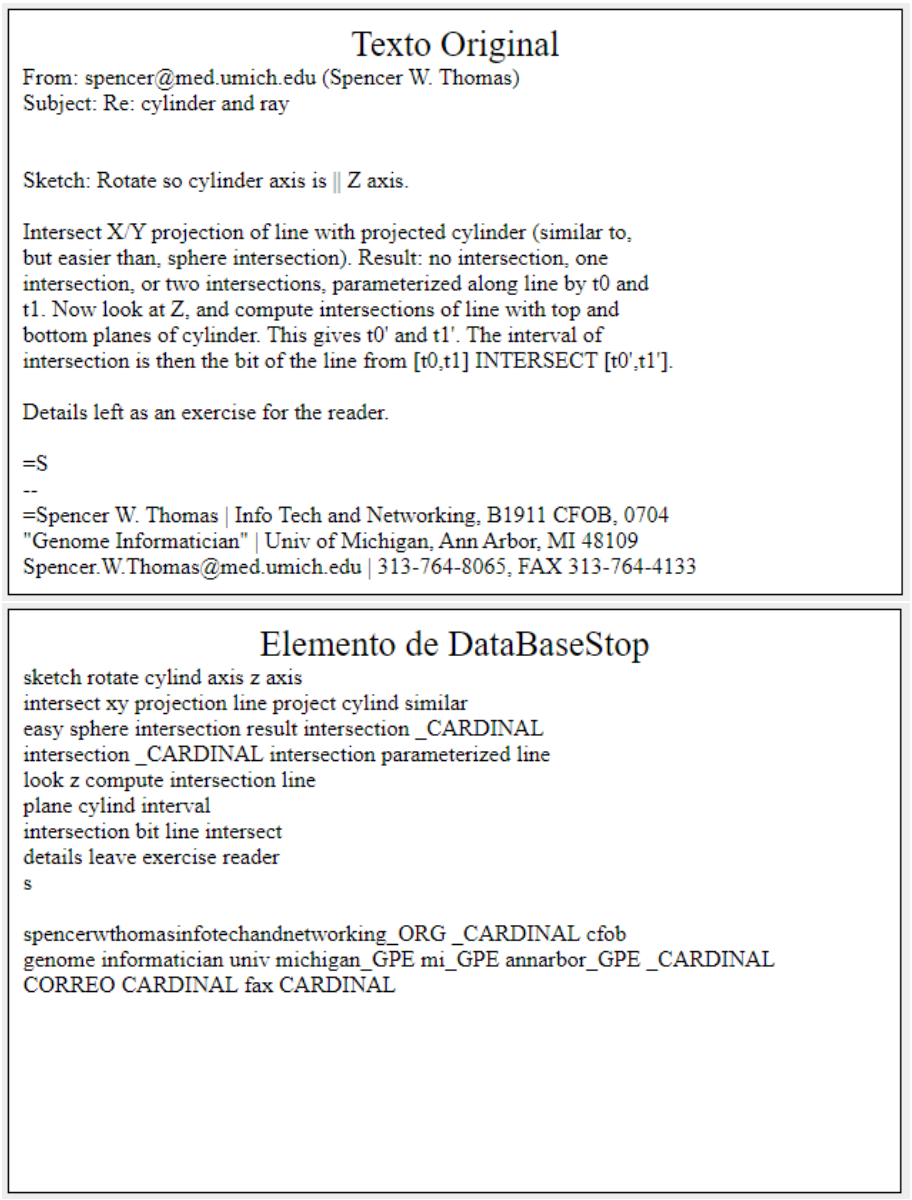


Figura 6.1: Resultado del preprocesamiento comparando original y *DataBaseStop* del documento con índice 3233.

6.2 Clasificadores

En esta sección se mostrarán los resultados de los clasificadores seleccionados, mediante validación cruzada con sus respectivas métricas.

Utilizando como métrica de evaluación del método de validación cruzada la precisión obtenemos los resultados presentados en las Tablas 6.2, 6.3 donde se ilustran los mínimos y máximos de la puntuación obtenida redondeando a 2 decimales.

Métrica/Clasificador	KNN	SVM	MLP
peor puntaje	0.13	0.05	0.85
mejor puntaje (promedio máximo)	**0.41**	**0.87**	**0.90**
media del puntaje	0.29	0.48	0.87
desviación estándar del puntaje	0.01	0.39	0.02

Tabla 6.2: Tabla de resultados de la validación cruzada para *DataBaseStop*.

Métrica/Clasificador	KNN	SVM	MLP
peor puntaje	0.19	0.05	0.85
mejor puntaje (promedio máximo)	**0.39**	**0.86**	**0.87**
media del puntaje	0.30	0.46	0.86
desviación estándar del puntaje	0.07	0.39	0.01

Tabla 6.3: Tabla de resultados de la validación cruzada para *DataBaseLem* (Con *stopwords*).

Como se ilustra, aunque un criterio de selección entre diferentes modelos de clasificadores sea quedarnos con aquel con mayor puntaje, reportar el resultado del peor clasificador junto con la media y desviación estándar ayuda a entender si nuestro clasificador no funciona correctamente. En estos casos, los resultados parecen indicar que incluir *stopwords* produce una disminución de la precisión en los clasificadores.

Respecto las métricas del mejor clasificador obtenido por la validación cruzada se reportan tanto las micro como macro así como el soporte en las Tablas 6.4, 6.5 para el caso que no incluye *stopwords* y las Tablas 6.6, 6.7 cuando sí se incluyen. En todos los casos se resalta el clasificador que mejor se haya desempeñado para cada métrica.

Train	Macro			Micro	sup
	prec	rec	F1	acc	
KNN	0.997	0.997	0.997	0.997	13602
SVM	**0.998**	**0.998**	**0.998**	**0.998**	13602
MLP	0.988	0.988	0.988	0.988	13602

Tabla 6.4: Resultados de las métricas, para el subconjunto de entrenamiento y la DataBaseStop (sin*stopwords*)

Test	Macro			Micro	sup
	prec	rec	F1	acc	
KNN	**0.930**	0.420	0.531	0.416	2825
SVM	0.879	0.874	0.874	0.879	2825
MLP	0.899	**0.896**	**0.896**	**0.901**	2825

Tabla 6.5: Resultados de las métricas, para el subconjunto de prueba y la DataBaseStop (sin *stopwords*)

Train	Macro			Micro	sup
	prec	rec	F1	acc	
KNN	**0.963**	0.903	0.921	0.901	13602
SVM	0.942	0.935	0.937	0.935	13602
MLP	0.945	**0.943**	**0.944**	**0.943**	13602

Tabla 6.6: Resultados de las métricas, para el subconjunto de prueba y la DataBaseLem (con *stopwords*)

Test	Macro			Micro	sup
	prec	rec	F1	acc	
KNN	**0.928**	0.191	0.247	0.185	2825
SVM	0.801	0.765	0.768	0.772	2825
MLP	0.813	**0.804**	**0.805**	**0.810**	2825

Tabla 6.7: Resultados de las métricas, para el subconjunto de entrenamiento y la DataBaseLem (con *stopwords*)

En todos los casos las Tablas anteriores hacen referencia a las métricas en inglés, y se presenta el soporte (*sup*) de cada subconjunto empleado. El comportamiento respecto el conjunto de entrenamiento y prueba resulta como lo esperado. Mejor en entrenamiento que en prueba en general.

Además, los resultados parecen indicar que el uso de las *stopwords* reduce el rendimiento de los clasificadores sobre todas las métricas. Si dichas *stopwords* son una fuente de "ruido" o perturbación para los clasificadores tendrían una contribución negativa al momento de calcular

su importancia mediante los explicadores. Se espera entonces que pocas o ninguna *stopword* sea considerada como un *token* importante.

6.3 Explicadores

En está sección se explicará el uso de los algoritmos LIME y SHAP para obtener características interpretables que tengan importancia para la decisión del clasificador, permitiendo formar una explicación con estos.

6.3.1 Obtención de características importantes

Si bien tanto LIME como SHAP son algoritmos agnósticos que emplean modelos sustitutos, en este trabajo se emplearon ciertas versiones especificas de estos.

En el caso de LIME se empleó el `LimeTextExplainer` que permite trabajar con características que son *tokens* retornando un objeto que incluye la lista de *tokens* como características importantes junto con un número que representa dicha importancia. Además provee del cálculo del coeficiente de puntuación basado en la métrica de *R-squared*.

En particular, este algoritmo no utiliza un clasificador como modelo sustituto, sino un regresor, ya que es posible que al momento de generar las muestras perturbadas de una instancia se obtengan resultados que solo pertenezcan a una clase. De forma que la métrica de similitud del modelo sustituto g con nuestro modelo opaco b se realiza sobre una regresión lineal con regularización *Ridge*.

Esta propiedad se considera un hiperparámetro en sí, ya que podría emplearse otra regresión, pero permite entender porque el resultado de la regresión no se encuentra acotado entre 0 y 1 para indicar la pertenencia a cierta clase.

Esta propiedad también ocurre en el caso de SHAP, donde se empleó el `KernelExplainer` que genera modelos sustitutos g basados LIME, y posteriormente se aproximan los *Shapley values*.

De forma similar a `LimeTextExplainer` el `KernelExplainer` emplea una regresión en lugar de una clasificación, siendo en este una regresión `Lasso` retornándonos así mismo su fidelidad respecto nuestro modelo opaco b mediante la métrica de *R-squared*. Dicha regresión es declarada de la forma dentro del código: `l1_reg= 1e-4`, donde `l1_reg` se refiere al *key argument* donde se elige el regresor y `1e-4` representa el coeficiente de regularización.

A diferencia de `LimeTextExplainer` el explicador `KernelExplainer` no realiza un filtro de los *tokens* presentes en el texto a explicar, de forma que intenta calcular la importancia sobre todo el vocabulario. Si bien sí presenta la opción de entregar una mascara para indicar qué características puede tomar para calcular su importancia, dicha mascara debe ser igual para todos las instancias a explicar. En nuestro caso se modificó el código base de SHAP para permitirnos ingresar por instancia o documento la lista de *tokens* presentes en el documento para que solo calcule la importancia de dichos *tokens*.

Respecto el cálculo de la métrica AOPC, la forma de obtenerla se realiza por instancia y después promediando dichos resultados se obtendría el AOPC. En este trabajo se reporta además la contribución de cada instancia al AOPC, de aquí en adelante Instancia AOPC.

La Instancia AOPC es obtenida al remover en orden *MoRF* los *tokens* importantes obtenidos por cada explicador de la instancia y midiendo la diferencia entre la predicción del clasificador con y sin dicho *token*.

El resumen de los resultados de los diferentes explicadores se ilustra en la Tabla 6.8. Los resultados de las métricas por instancia se reportan más adelante.

	KNN		SVM		MLP	
Explicador	R-squared promedio	AOPC	R-squared promedio	AOPC	R-squared promedio	AOPC
LIME-500	**0.780**	0.524	**0.493**	0.203	**0.568**	0.168
LIME-1000	0.776	0.540	0.483	0.214	0.563	0.178
LIME-5000	0.773	**0.550**	0.473	**0.241**	0.555	**0.186**
SHAP-8-100	**0.561**	0.487	**0.584**	0.139	**0.540**	0.150
SHAP-2-100	0.372	0.473	0.381	0.132	0.361	0.133
SHAP-2-1000	0.221	**0.528**	0.199	**0.179**	0.185	**0.177**

Tabla 6.8: Resultados de los diferentes modelos de explicadores implementados, se remarcan los mejores resultados para cada caso.

Se aprecia cómo los resultados varían dependiendo del explicador usado, en cuanto a la métrica de fidelidad local *R-squared* en el que ambos explicadores aproximaron mejor para clasificador KNN. También la *R-squared* media sobre las instancias resulta mayor para LIME que para SHAP, confirmando que el primero posee modelos sustitutos que reflejan mejor el comportamiento del clasificador.

6.3.2 Presentación de las explicaciones

Debido a que podemos presentar la explicación de una predicción de una instancia de diferentes formas, a continuación se ilustran dos aproximaciones para una misma instancia:

La instancia en cuestión es el documento cuyo índice es 815 correspondiente al texto original ilustrado en la Figura 6.2.

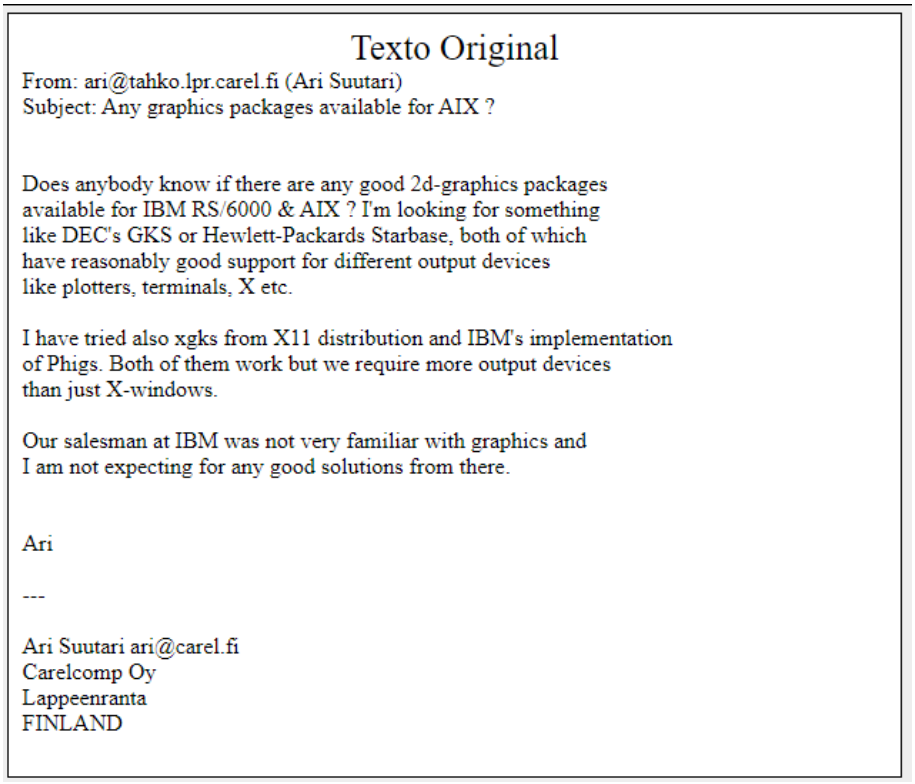


Figura 6.2: Texto original

A continuación mostraremos la comparativa de las explicaciones obtenidas de KNN en las Figuras 6.3 y 6.5, y de la MLP en las Figuras 6.4 y 6.6 con los explicadores LIME-5000 y SHAP-8-100.

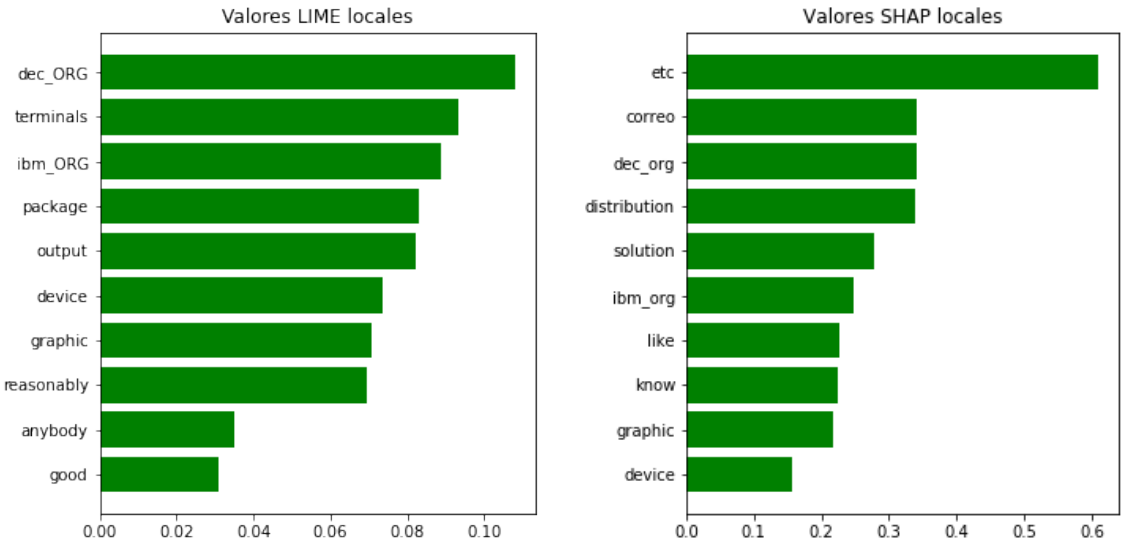


Figura 6.3: Tablas de valores para el clasificador KNN aplicando LIME-5000 y SHAP-8-100 sobre la instancia 815.

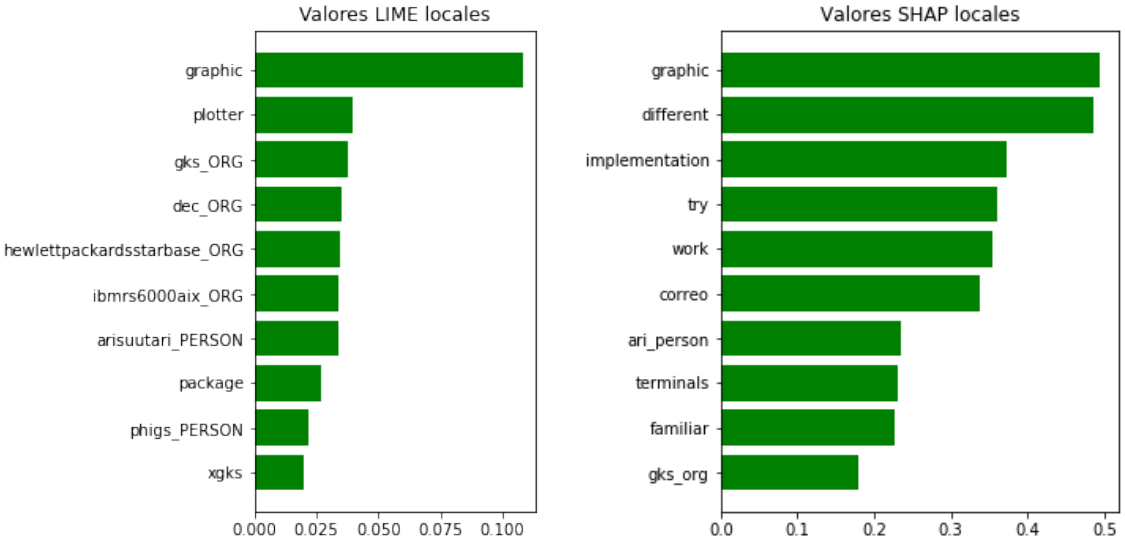


Figura 6.4: Tablas de valores para el clasificador MLP aplicando LIME-5000 y SHAP-8-100 sobre la instancia 815.

Notemos cómo cada explicador ha conseguido una lista de *tokens* (se muestran las primeras 10) y valores diferentes. Esto coincide con un comportamiento distinto entre las métricas de los clasificadores: KNN con una *acc* de 0.416 da importancia a diferentes *tokens* que una a una MLP con *acc* de 0.901 para el test según la Tabla 6.5. También existe una diferencia entre los *tokens* de cada explicador, y esta puede deberse a la diferencia de fidelidad local de los explicadores.

El resultado de cada *heatmap* incluye las métricas del explicador. Empleando la métrica de fidelidad local podemos decidir cual *heatmap* proviene de un modelo sustituto g más similar al modelo opaco b y con ambos AOPC cuán tan significativos son dichos *tokens* para el clasificador.

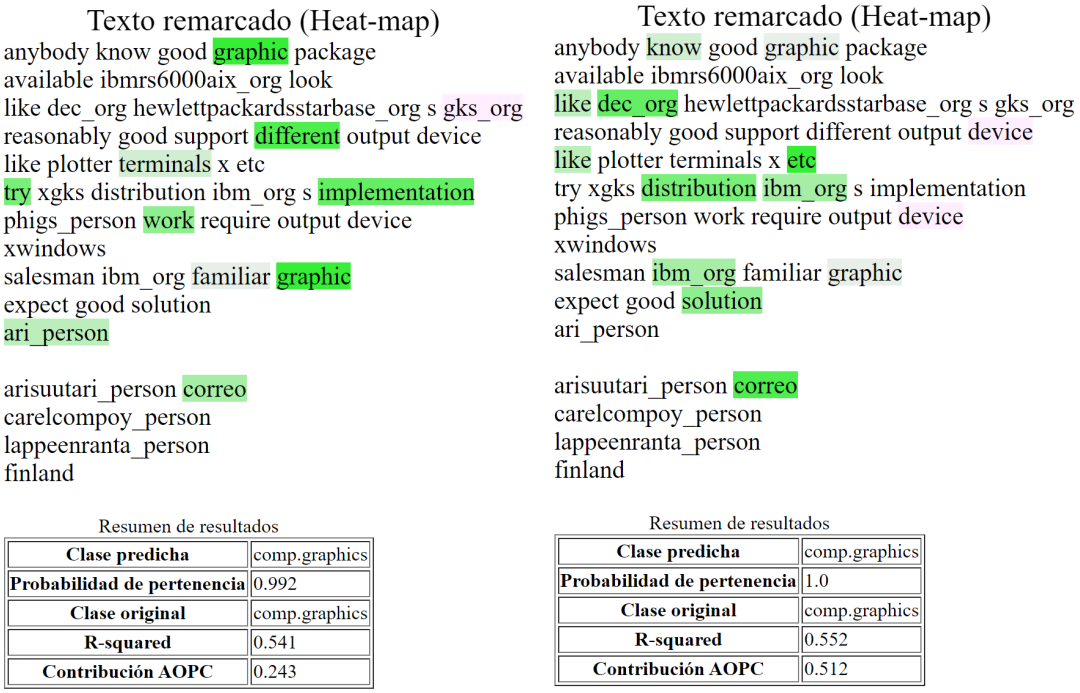


Figura 6.5: *Heatmap* de los explicadores (LIME-5000 izquierda, SHAP-8-100 derecha) para el clasificador KNN.

En la Figura anterior 6.5 se muestra un ejemplo de fidelidad local similar pero con diferente impacto de los *tokens*. Aunque la clasificación es correcta, la importancia obtenida por SHAP-8-100 indica cómo hay una mayor importancia en términos no necesariamente asociados a la categoría (La importancia de "etc", "correo" es mayor que "graphic"). La clasificación ha sido correcta, pero el clasificador ha priorizado características diferentes a las que se esperaría. Esto puede significar un conocimiento nuevo o una tendencia de nuestro clasificador, dado que el KNN tuvo baja *acc* en su validación podríamos tomar esto como un indicio de sobreajuste.

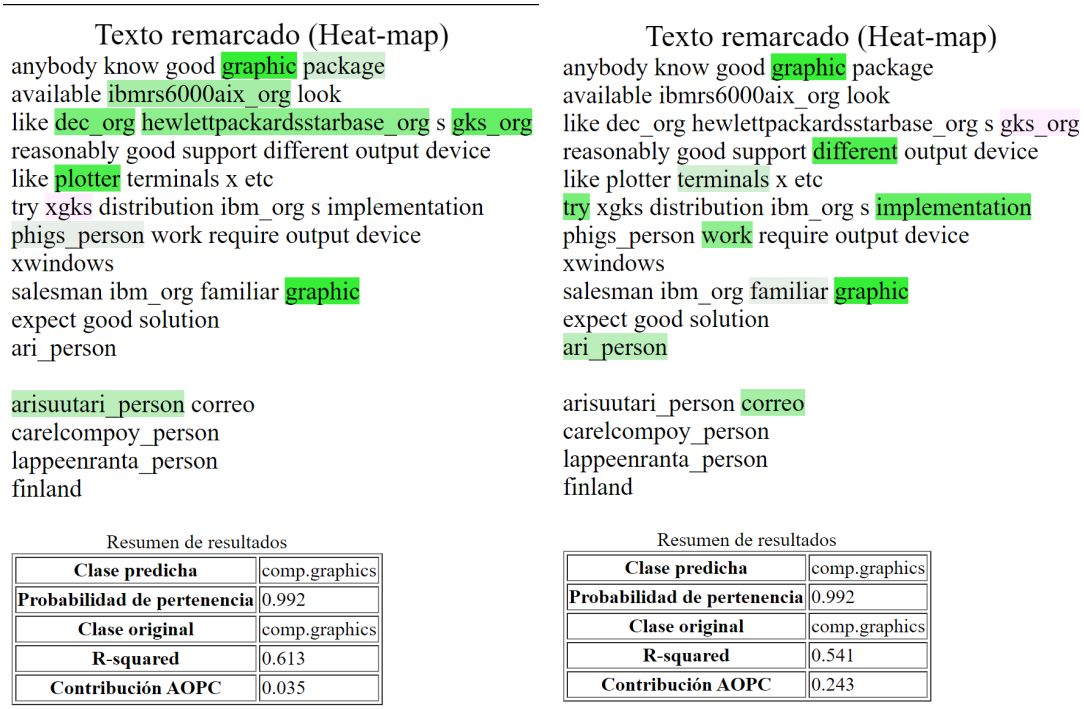


Figura 6.6: *Heatmap* de los explicadores (LIME-5000 izquierda, SHAP-8-100 derecha) para el clasificador MLP.

A diferencia del caso del clasificador KNN en la Figura 6.6 tenemos como los *tokens* son similares, y dado que el clasificador MLP tuvo un desempeño positivo en su etapa de validación se implica una clasificación correcta debida a características asociadas a la categoría.

Notemos como este proceso interpretativo es realizado de manera local y no es posible generalizar la decisión si el clasificador ha tomado las características que se esperarían como significativas para toda una categoría.

A continuación se ilustran en las Figuras 6.7, 6.8 y 6.9 tres casos sobre un mismo documento, instancia 10850. Donde la interpretación de la explicación tiene diferentes significados.

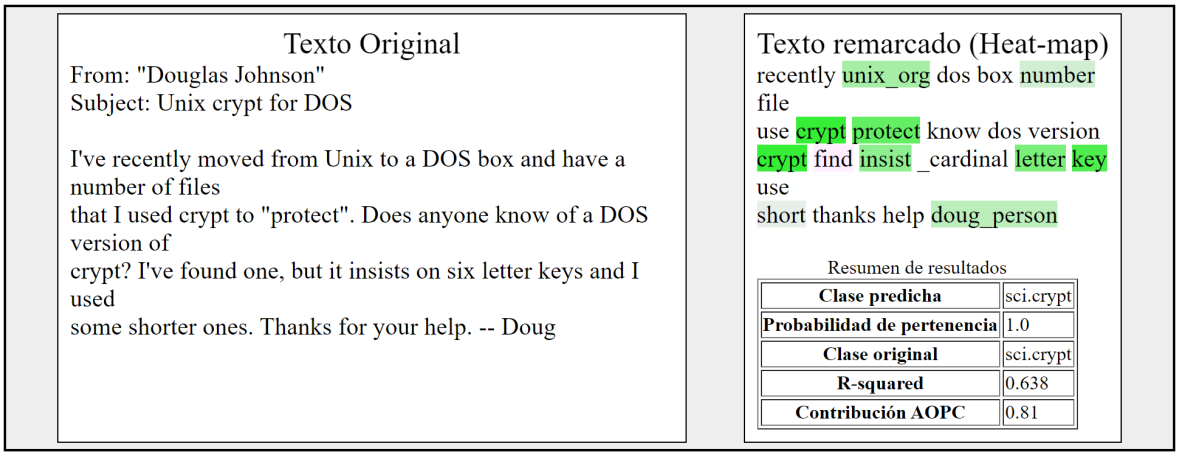


Figura 6.7: Primer ejemplo, clasificación correcta, explicación relevante.

En la Figura 6.7 se muestra un ejemplo de clasificación correcta por la SVM y con una explicación de *tokens* asociados a la categoría por LIME-5000. El resultado de las métricas del explicador coincide en indicar que las características son significativas y representan de forma local al modelo opaco.

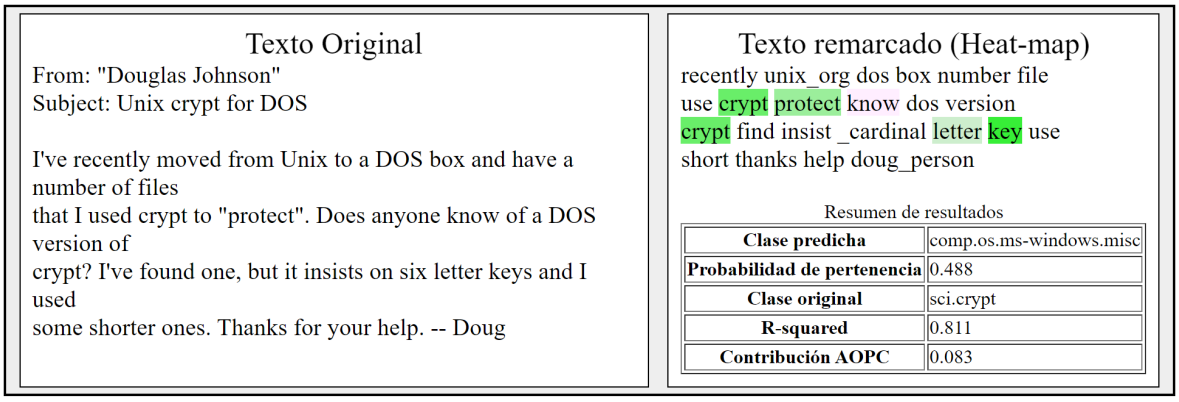


Figura 6.8: Segundo ejemplo, clasificación incorrecta, explicación relevante.

En la Figura 6.8 se presenta el caso en el que el clasificador (MLP) se ha equivocado aunque los *tokens* importantes coinciden con la categoría. En particular vemos cómo el rendimiento del explicador, LIME-5000, tiene una alta fidelidad local pero un bajo impacto en la categoría predicha por el clasificador, la contribución del AOPC es baja respecto la categoría incorrecta.
Si no conociéramos la categoría objetivo, que la importancia del término "key" fuera mayor que de la importancia de "crypt" indicaría que se ha cometido un error o que la muestra tiene un comportamiento anómalo. De igual forma el bajo AOPC de *tokens* que son relacionados con la categoría implican un error en la clasificación.

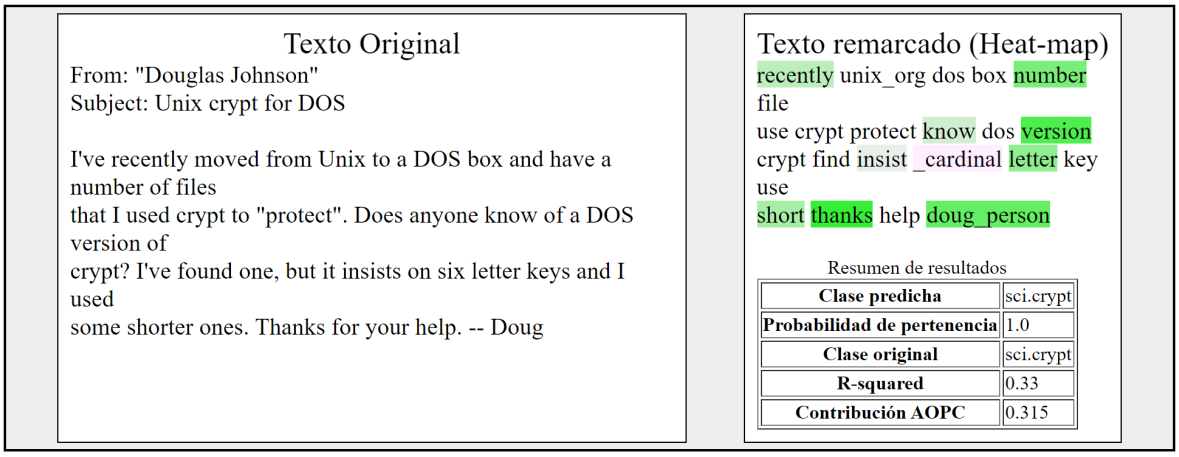


Figura 6.9: Clasificación correcta, explicación no relevante.

Finalmente, en la Figura 6.9 un ejemplo en el que la clasificación es correcta pero la explicación tiene un rendimiento bajo. Los términos encontrados no parecen tener relación y su impacto a la predicción es bajo. En comparación con el primero de estos tres casos, la explicación obtenida por LIME-5000 fue mejor que la obtenida por SHAP-2-1000.
Notemos que este comportamiento no se generaliza, sino que varía de muestra a muestra. En la Figura 6.6 se muestra un caso donde pese a la similitud de las explicaciones, el AOPC es mayor en el explicador SHAP-8-100 que el obtenido por LIME-5000.

6.3.3 Aproximación global

A manera de mostrar una aproximación de una representación global del modelo podemos obtener el promedio de importancia obtenido por cada *token* sobre todas las instancias explicadas de una misma categoría. El resultado de este procedimiento es una tabla con los *tokens* que más repercuten, como se ilustra en las Tablas 6.10 y 6.11.

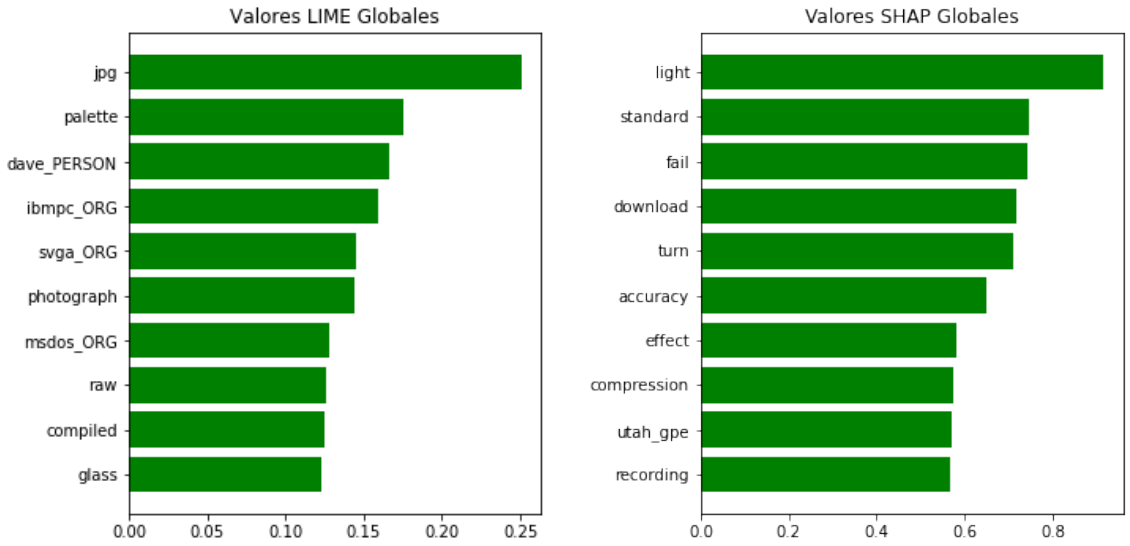


Figura 6.10: Aproximación de explicación global para el clasificador KNN con los explicadores LIME-5000 y SHAP-8-100.

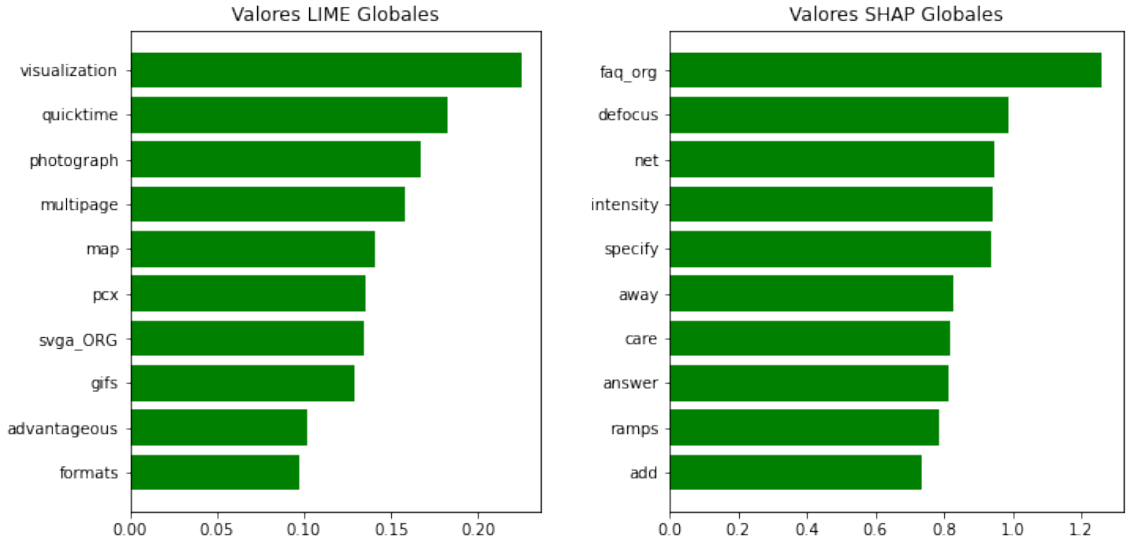


Figura 6.11: Aproximación de explicación global para el clasificador MLP con los explicadores LIME-5000 y SHAP-8-100

También podemos realizar un *heatmap* con dicha tabla sobre una instancia especifica. Resultando en la representación de la Figura 6.12

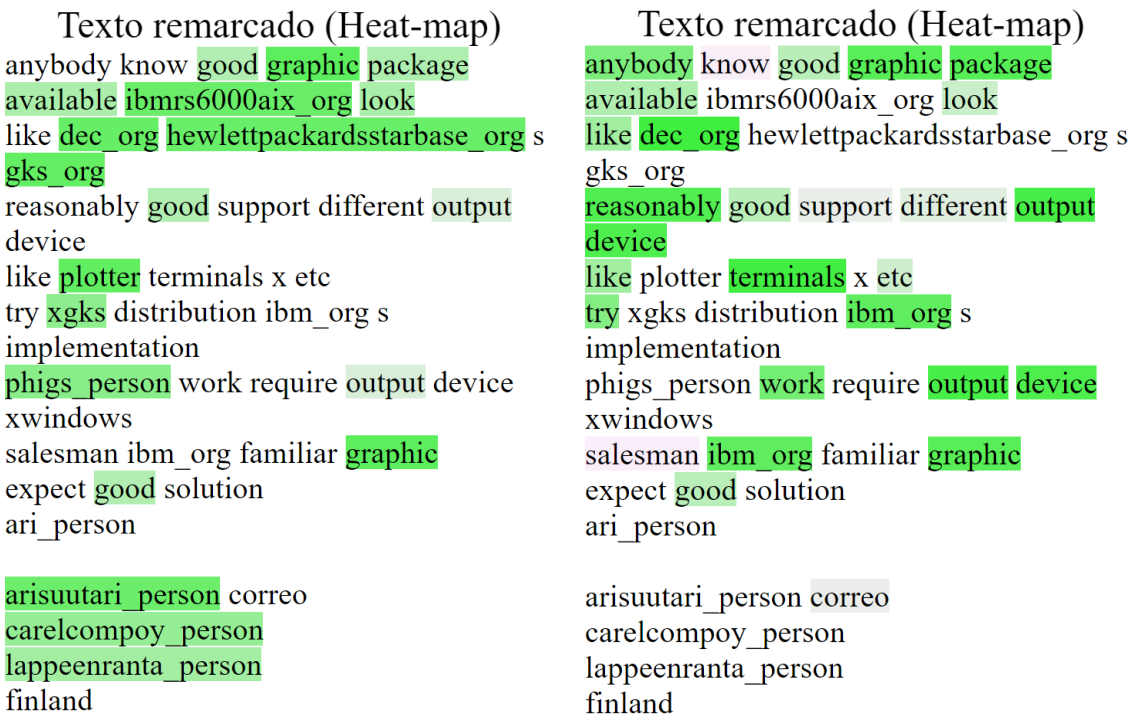


Figura 6.12: *Heatmaps* de aproximación global para la categoría *comps.graphics* para los clasificadores MLP y KNN obtenidas mediante LIME-5000.

La inclusión de la aproximación global por categoría del *heatmap* nos permite tener una idea de cuales *tokens* de nuestra instancia son significativos para el clasificador, aunque no necesariamente la explicación local los haya considerado. Por ejemplo, en la Figura 6.12 tenemos que el clasificador MLP toma como importantes diferentes *tokens* reconocidos como entidades que corresponder al pie de página y posiblemente correspondan a un sesgo de la base de datos. Mientras que el clasificador KNN no las ha tomado como importantes dentro de la muestra explicada.

CAPÍTULO 7

CONCLUSIONES Y TRABAJO A FUTURO

Se implementan y evalúan algoritmos clasificadores y algoritmos explicadores aplicados a la tarea de clasificación de textos.
Los clasificadores fueron entrenados y validados logrando resultados cercanos a los presentados en la literatura.
La evaluación de los algoritmos explicadores estos se realiza con las métricas propuestas y la representación de la explicación permite comparar las diferentes explicaciones obtenidas.
Los resultados de las métricas de los clasificadores coinciden en que estos tuvieron un mejor rendimiento no emplear las *stopwords*.
Se hicieron las modificaciones necesarias para emplear `KernelSHAP` al permitir el uso de una mascara de características por cada instancia a explicar, así como la integración del *pipeline* del clasificador.
Ocurre un intercambio de rendimiento entre la métrica *R-squared* promedio y el AOPC, entre más iteraciones se realizan se obtiene un mayor impacto de las características, pero una menor fidelidad local. Esto concuerda con que el algoritmo explicador debe converger a una única explicación entre más iteraciones se realicen.
De forma general, las explicaciones obtenidas mediante LIME tienen un mayor *R-squared* y un mayor AOPC que las obtenidas mediante SHAP.
De forma particular, el clasificador KNN tuvo un mayor desempeño en ambas métricas que los otros clasificadores para ambos explicadores al comparar las versiones con mayor número de iteraciones.
Se presentan casos donde las explicaciones no necesariamente coinciden entre los explicadores, dónde los resultados de las métricas contribuyen a entender el funcionamiento del clasificador en casos particulares.

Los algoritmos explicadores de modelo sustituto lineal que obtienen la importancia de características por permutación están restringidos por el costo derivado de la cantidad de características, en nuestro caso el vocabulario. Así como la suposición de independencia lineal de las características, la suposición de comportamiento lineal en la localidad y la suposición de distribución homogénea de las muestras generadas entorno la muestra a explicar.
El presente trabajo se limita a un modelo de espacio vectorial basado en bolsa de palabras cuyo alcance no permite revisar la importancia en función de la posición.

7.1 Trabajo a futuro

En este trabajo solo se exploraron dos métricas para evaluar las explicaciones obtenidas, sin embargo es posible implementar más métricas dependiendo de la naturaleza de los clasificadores y qué se busque medir de las explicaciones obtenidas. Implementar diferentes métricas así como evaluar su significado contribuiría a la explicación.
Dentro de los explicadores propuestos existen diferentes hiperparámetros que son algoritmos en sí (obtención de modelo lineal sustituto, generación de instancias sintéticas a partir de perturbaciones de una instancia a explicar y selección de características a modificar) el trabajo se limita únicamente a cierta combinación de estos hiperparámetros, una exploración más exhaustiva podría derivar en mejores explicadores.
El trabajo abordó el enfoque *MoRF* pero la propuesta de no solo retirar características significativas sino agregarlas y emplear otro enfoque podría permitir a estos algoritmos explicadores regresar instancias prototípicas.
Incluir tiempo y la complejidad de los algoritmos como criterio de evaluación para los algoritmos explicadores, el costo de una explicación también es un factor relevante.
Además, se sugiere implementar una evaluación de las explicaciones obtenidas mediante un proceso empírico donde usuarios indiquen su preferencia respecto la generalización de los explicadores y la utilidad de las explicaciones para identificar irregularidades de los clasificadores.
Emplear modelos de espacios vectoriales basados en *embeddings* para obtener la importancia de palabras en estos y posteriormente obtener la importancia de los elementos del *embedding* sobre el clasificador. Extender estos modelos a *embeddings* posicionales para poder obtener importancia en función de la posición.

REFERENCIAS

[1] C. Cortes and V. Vapnik, "Support-vector networks",
Machine learning, vol. 20, no. 3, pp. 273–297, 1995.

[2] K. Lang,
"Newsweeder: Learning to filter netnews",
In *Machine Learning Proceedings 1995*,
Elsevier, 1995,
Pp. 331–339.

[3] C. Manning and H. Schutze,
Foundations of statistical natural language processing.
MIT press, 1999.

[4] Y. Yang and X. Liu,
"A re-examination of text categorization methods",
In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*,
1999,
Pp. 42–49.

[5] G. Schohn and D. Cohn,
"Less is more: Active learning with support vector machines",
In *ICML*,
Citeseer,
Vol. 2, 2000,
P. 6.

[6] L. Breiman, "Random forests",
Machine learning, vol. 45, no. 1, pp. 5–32, 2001.

[7] R. F. Correa and T. B. Ludermir,
"Web documents categorization using neural networks",
In *International Conference on Neural Information Processing*,
Springer,
2004,

Pp. 758–762.

[8] C. M. Bishop, “Pattern recognition”,
Machine learning, vol. 128, no. 9, 2006.

[9] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, “Using knn model for automatic text categorization”,
Soft Computing, vol. 10, no. 5, pp. 423–430, 2006.

[10] S. Bird, E. Loper, and E. Klein, “Natural language processing with python o’reilly media inc”,
2009.

[11] I. Mogotsi,
Christopher d. manning, prabhakar raghavan, and hinrich schütze: Introduction to information retrieval,
2010.

[12] C.-C. Chang and C.-J. Lin, “Libsvm: A library for support vector machines”,
ACM transactions on intelligent systems and technology (TIST), vol. 2, no. 3, pp. 1–27, 2011.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python”,
Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[14] A. A. Freitas, “Comprehensible classification models: A position paper”,
ACM SIGKDD explorations newsletter, vol. 15, no. 1, pp. 1–10, 2014.

[15] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger,
“From word embeddings to document distances”,
In *International conference on machine learning*,
PMLR,
2015,
Pp. 957–966.

[16] S. M. H. Dadgar, M. S. Araghi, and M. M. Farahani,
“A novel text mining approach based on tf-idf and support vector machine for news classification”,
In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*,
IEEE,
2016,
Pp. 112–116.

[17] T. Lei, R. Barzilay, and T. Jaakkola, “Rationalizing neural predictions”,
arXiv preprint arXiv:1606.04155, 2016.

[18] 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*,
2016,
Pp. 1135–1144.

[19] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned",
IEEE transactions on neural networks and learning systems, vol. 28, no. 11, pp. 2660–2673, 2016.

[20] A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics derived automatically from language corpora contain human-like biases",
Science, vol. 356, no. 6334, pp. 183–186, 2017.

[21] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning",
arXiv preprint arXiv:1702.08608, 2017.

[22] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"",
AI magazine, vol. 38, no. 3, pp. 50–57, 2017.

[23] S. M. Lundberg and S.-I. Lee,
"A unified approach to interpreting model predictions",
In *Proceedings of the 31st international conference on neural information processing systems*,
2017,
Pp. 4768–4777.

[24] S. Wachter, B. Mittelstadt, and L. Floridi, "Why a right to explanation of automated decision-making does not exist in the general data protection regulation",
International Data Privacy Law, vol. 7, no. 2, pp. 76–99, 2017.

[25] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)",
IEEE access, vol. 6, pp. 52 138–52 160, 2018.

[26] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local rule-based explanations of black box decision systems",
arXiv preprint arXiv:1805.10820, 2018.

[27] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models",
ACM computing surveys (CSUR), vol. 51, no. 5, pp. 1–42, 2018.

[28] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein, “Explainable prediction of medical codes from clinical text”,
arXiv preprint arXiv:1802.05695, 2018.

[29] D. Nguyen,
“Comparing automatic and human evaluation of local explanations for text classification”,
In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*,
2018,
Pp. 1069–1078.

[30] E. Wallace, S. Feng, and J. Boyd-Graber, “Interpreting neural networks with nearest neighbors”,
arXiv preprint arXiv:1809.02847, 2018.

[31] V. Buhrmester, D. Münch, and M. Arens, “Analysis of explainers of black box deep neural networks for computer vision: A survey”,
arXiv preprint arXiv:1911.12116, 2019.

[32] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences”,
Artificial intelligence, vol. 267, pp. 1–38, 2019.

[33] C. Molnar,
Interpretable Machine Learning, A Guide for Making Black Box Models Explainable.
2019,
`https://christophm.github.io/interpretable-ml-book\/`.

[34] R. Pappagari, P. Zelasko, J. Villalba, Y. Carmiel, and N. Dehak,
“Hierarchical transformers for long document classification”,
In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*,
IEEE,
2019,
Pp. 838–844.

[35] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, and P. Sen, “A survey of the state of explainable ai for natural language processing”,
arXiv preprint arXiv:2010.00711, 2020.

[36] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, “spaCy: Industrial-strength Natural Language Processing in Python”,
2020.
DOI: `10.5281/zenodo.1212303`.

[37] B.-M. Hsu, “Comparison of supervised classification models on textual data”,
Mathematics, vol. 8, no. 5, p. 851, 2020.

[38] Q. Jiang, H. Zhang, J. Shang, I. Wesson, and E. Li,
"Densely connected bidirectional lstm with max-pooling of cnn network for text classification",
In *International Conference on Advanced Data Mining and Applications*,
Springer,
2020,
Pp. 98–113.

[39] A. K. Mohankumar, P. Nema, S. Narasimhan, M. M. Khapra, B. V. Srinivasan, and B. Ravindran, "Towards transparent and explainable attention models",
arXiv preprint arXiv:2004.14243, 2020.

[40] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning,
"Stanza: A Python natural language processing toolkit for many human languages",
In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*,
2020.
[Online]. Available: `https://nlp.stanford.edu/pubs/qi2020stanza.pdf`.

[41] S. Kumar, A. Gulati, R. Jain, P. Nagrath, and N. Sharma,
"Categorizing text documents using naive bayes, svm and logistic regression",
In *Data Management, Analytics and Innovation*,
Springer, 2021,
Pp. 225–235.

[42] R. Sukumaran, "Improved customer transaction classification using semi-supervised knowledge distillation",
arXiv preprint arXiv:2102.07635, 2021.

[43] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization",
Communications of the ACM, vol. 64, no. 3, pp. 107–115, 2021.

[44] *20 newsgroups*,
`http://qwone.com/~jason/20Newsgroups/`,
Accessed: 2021-11-22.

[45] L. Shih, Y.-H. Chang, J. Rennie, and D. Karger,
"Not too hot, not too cold: The bundled-svm is just right!",
Citeseer.