Universidad Internacional de la Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

Creación de un clasificador de sentimientos de alumnos mediante minería de opinión

Trabajo Fin de Máster

presentado por: Ignacio Belzunegui Gabilondo

Dirigido por: Álvaro Alexánder Martínez Navarro

Ciudad:Madrid

Fecha: 06 de febrero de 2023

Índice de Contenidos

Índice de Ilustraciones

Índice de Tablas

Resumen

Para un profesor, reconocer la opinión de un alumno sobre él puede ser bastante difícil. Cuando hay mucha información, leerla toda puede ser complicado, y se necesita una herramienta que permita extraer todos estos datos. En este trabajo se propone una aplicación que permite a los profesores conocer la opinión de sus alumnos.

Como parte de la operación de minería de opinión, los comentarios han pasado por varios procesos: se han corregido palabras mal escritas, se han convertido en sinónimos algunas relacionadas al ámbito educativo, y se han convertido en sus respectivos lemas.

Para representar los comentarios como datos numéricos se han analizado dos algoritmos de extracción de características: bolsa de palabras y TF-IDF. Por otro lado, para aprender a clasificar los datos, se han explorado tres algoritmos: SVC, Naïve Bayes multinomial y Pasivo Agresivo.

El modelo de aprendizaje automático en el que se basa el proyecto ha sido entrenado utilizando comentarios de la Universidad Mariana de Colombia, y alcanza más de un 90 % de precisión.

Palabras Clave: Bolsa de palabras, Clasificación, Comentarios, Extracción de características, Minería de opinión, Naïve Bayes Multinomial, Pasivo Agresivo, Sentimientos, SVC, TF-IDF, Validación cruzada, Aprendizaje automático.

Abstract

For a professor, acknowledging a student's opinion about him can be quite difficult. When there is a lot of information, reading all of it can be complicated, and a tool to mine all this data is required. In this work an application that allows teachers to know their students' opinion is proposed.

As part of the opinion mining operation, comments have gone through various processes: misspelt words have been corrected, some related to the educational field have been turned to synonyms, and they have been turned into their respective lemmas.

With regard to represent comments as numerical data two feature extraction algorithms have been analyzed: Bag of Words and TF-IDF. On the other hand, in order to learn how to classify data, three algorithms have been explored: SVC, Multinomial Naïve Bayes and Passive Aggressive.

The machine learning model on which the project is based has been trained using comments from the Mariana University in Colombia, and achieves over 90 % accuracy.

Key words: Bag of words, Classification, Comments, Cross validation, Feature extraction, Multinomial Naïve Bayes, Opinion mining, Passive aggresive, Sentiments, SVC, TF-IDF, Machine learning.

Capítulo 1

Introducción

1.1. Motivación detrás del proyecto

1.1.1. Síntomas

Las grandes cantidades de información con las que cuentan las instituciones a día de hoy posibilitan crear modelos que perfilan la realidad de forma precisa. De acuerdo a [7], diariamente se generan 2,5 quintillones de bytes, y su empleo hoy en día es crucial para el correcto funcionamiento del mundo. La minería de opinión es una de las técnicas que facilitan crear estos modelos, y posibilita, por ejemplo, saber si una película ha sido exitosa o no basándose en las millones de críticas alojadas en la red. Esta información permite hacer saber a las productoras si su producto ha sido bien recibido por el público, o por el contrario, mal recibido. Asimismo, con este conocimiento, la junta directiva puede decidir de forma más efectiva qué elementos incluir o excluir en sus próximos proyectos teniendo en cuenta las críticas de los consumidores. Este ejemplo es extrapolable a cualquier negocio a día de hoy, y reporta a las empresas enormes beneficios [18].

En esta línea, las instituciones académicas, a través de los portales donde los alumnos expresan su opinión acerca del docente, el contenido de las asignaturas y la calidad de las clases, pueden emplear estos datos para mejorar su oferta educativa [33].

1.1.2. Causas

Existen estudios que demuestran que el recibimiento de la opinión del alumnado por parte del profesorado es beneficioso a la hora de mejorar la calidad educativa [15]. Muchos centros educativos poseen una cantidad de información que, empleada de forma correcta,

podría ser beneficiosa tanto para ellos como para los alumnos a los que enseñan. Este fénomeno ha dado lugar a una rama especializada dentro de la minería de datos; la minería de datos educacional [33] [40].

La mayoría de literatura acerca de estas técnicas está más enfocada en el aspecto metodológico, dejando a un lado la implementación de las mismas [33]. Si bien existen herramientas capaces de realizar minería de datos [40] y de opinión [3] educacional, todavía no existe una que ofrezca esta posibilidad de manera fácil y accesible.

Las instituciones no poseen un software intuitivo que les acerque esta tecnología de una manera amigable y completa; muchas veces no sirve con simplemente poseer la información, pues para ser de utilidad esta también ha de ser transformada, filtrada y procesada de forma correcta, lo que exige herramientas especializadas [40].

1.1.3. Diagnóstico

Como se ha mencionado anteriormente, existen instrumentos creados con el objetivo de mejorar la calidad de la educación mediante el uso de minería de opinión.

Debido a ello, en este trabajo se creará un software que, mediante el uso de inteligencia artificial, permitirá a las diferentes organizaciones educativas obtener un diagnóstico completo de la calidad de su producto a partir de la minería de opinión de los comentarios de sus estudiantes. A partir de un fichero que contenga las críticas, se mostrará de forma gráfica la percepción que los escolares tienen de las clases.

Se emplearán técnicas de minería y aprendizaje automáticos ya existentes, incidiendo en la accesibilidad y en el empleo de procedimientos y recursos eficientes.

1.1.4. Pronóstico

Este software democratizará el uso del procesamiento del lenguaje natural en el mundo de la educación. Los profesores podrán contar con una herramienta que les permita llevar un seguimiento del estado de sus clases de manera intuitiva, independientemente de lo familiarizados que estén con las nuevas tecnologías.

El aporte de este proyecto será poner a disposición de diferentes instituciones educativas un aliado que les permita redefinir su manera de impartir clase, contribuyendo a una mejora en la calidad en la forma de enseñar. Esto se traduce en beneficios para la organización y para el alumnado, lo que indirectamente resulta en un impacto positivo para la sociedad.

1.2. Planteamiento del trabajo

Este software se basa en dos ejes vertebradores para llevar a cabo su cometido: la minería de opinión y el aprendizaje automático.

Existen tres elementos fundamentales que hacen funcionar a esta aplicación: en primer lugar, el archivo que contiene los comentarios que serán evaluados; en segundo lugar, el módulo que analiza los comentarios, extrae las opiniones y evalúa los resultados obtenidos a partir de ellos; en tercer y último lugar, la interfaz gráfica de la aplicación, la cual conecta el segundo módulo con el usuario, además de convertir las predicciones en contenido gráfico fácilmente interpretable por el mismo. Estas tres piezas trabajan en conjunto para hacer que la aplicación trabaje de manera eficiente, efectiva y completamente autónoma.

El lenguaje de programación utilizado será *Python*, y se hará uso de librerías tales como *sklearn*, *nltk* y *Pandas*. Para realizar el apartado gráfico de la aplicación se utilizarán *HTML*, *CSS* y *Javascript*.

Se cuenta con una gran base de datos de opiniones para que el componente encargado del aprendizaje pueda llevar a cabo aprendizaje supervisado, obtenido por parte de la Universidad Mariana de Colombia.

Se cuenta con alrededor de 8700 comentarios, los cuales serán empleados para entrenar un modelo y desplegarlo para que el usuario pueda realizar sus predicciones. No obstante, para poder crearlo, se utilizarán estos mismos datos para evaluar que los algoritmos empleados y el calibrado de los hiperparámetros seleccionados funcionen correctamente a través de la estrategia K-Fold. Siguiendo esta filosofía, se puede optar por utilizar esta gran cantidad de datos para crear un modelo con un buen aprendizaje y para validar su funcionamiento, sin tener que renunciar a ningún comentario. Se buscará una precisión en las predicciones igual o superior al 90 %.

1.3. Estructura del trabajo

En primer lugar, en el estado del arte se presenta la situación actual de los algoritmos, técnicas y programas que resuelven problemas parecidos al propuesto en este trabajo. Consiste, mayoritariamente, en una recopilación de artículos e investigaciones acerca de algoritmos de minería de opinión y *machine learning* aplicados al procesamiento del lenguaje natural, pues son los pilares fundamentales sobre los que se sostiene este proyecto y constituyen la faceta más científica del mismo. Se comparan las diferentes técnicas y se

establece cuáles son las que mejor resuelven el problema a tratar.

En el apartado de objetivos y metodologías de trabajo se explica de forma más detallada cuáles son los problemas a los que se enfrenta este proyecto y la forma en la que los resuelve. Se exponen de manera ordenada los objetivos generales y específicos que se pretenden resolver así como las metodologías empleadas para ello.

En la sección de desarrollo específico de la contribución se explica qué diseño se ha seguido para desarrollar el programa así como su justificación. Se comentan los detalles más técnicos acerca de la implementación de los algoritmos, los conjuntos de datos empleados, las técnicas empleadas así como otras herramientas.

En el apartado de evaluación se lleva a cabo una medición de la usabilidad y de la funcionalidad del software junto con un análisis de las soluciones técnicas exploradas en el ámbito del aprendizaje automático.

En las conclusiones y trabajo futuro se discuten los resultados obtenidos, la aportación del proyecto dentro del problema que se trata así como posibles mejoras a realizar.

Capítulo 2

Contexto y Estado del Arte

2.1. Procesamiento del lenguaje natural

2.1.1. Conceptualización

Antes de comprender qué es el procesamiento del lenguaje natural o sus implicaciones, conviene tener presente la definición del lenguaje; este se define como un conjunto de reglas o símbolos que se combinan y utilizan para transmitir o difundir información [25].

El procesamiento del lenguaje natural se sitúa en el corazón de la resolución del problema planteado como origen de este proyecto. Consiste en el uso de la computación para procesar o entender el lenguaje humano en su estado natural, con el fin de desempeñar tareas útiles [10]. Puede ser entendido también como el conjunto de métodos que hacen accesible el lenguaje humano a los ordenadores [11].

En el ámbito científico, se entiende como una forma de modelar los mecanismos cognitivos subyacentes en la comprensión y producción del lenguaje humano; en el ingenieril, como un método capaz de desarrollar aplicaciones prácticas que permiten facilitar la interacción entre ordenadores y el habla de las personas [10].

Sus aplicaciones incluyen reconocimiento de voz, detección de spam, implicaciones médicas, resumen, comprensión del lenguaje hablado, sistemas de diálogo, análisis léxico, parseo, traducción, obtención de información o respuesta de preguntas [10] [25]. Por ejemplo, en el caso del marketing puede ser empleado para determinar qué aspectos de un producto son populares así como identificar a qué demografías les gusta o no ciertos aspectos [42].

El lenguaje natural puede ser definido como un sistema construido con el fin de trans-

mitir significado o semántica, y puede ser simbólico o discreto [10]. La señal físicamente observable del mismo es el texto, y se representa en forma simbólica [10].

Los sistemas de PLN (Procesamiento del Lenguaje Natural) pueden ser divididos en dos partes: la comprensión del lenguaje natural y la generación del mismo [25].

Un sistema de comprensión puede ser, por ejemplo, una red neuronal convolucional (CNN) [44], mientras que una red generativa antagónica (GAN) puede constituir un sistema de generación [28].

En esta línea, los modelos de aprendizaje automático pueden ser categorizados como generativos (generando datos sintéticos a partir de modelos de distribuciones de probabilidad) o discriminativos (estimando probabilidades posteriores basándose en observaciones) [34].

Algunos métodos que permiten crear modelos de aprendizaje automático discriminativos son la regresión logística, campos aleatorios condicionales (CFRFs o *Conditional Random Fields*) o SVM (*Support Vector Machines*), mientras que de generativa lo son Naïve Bayes y modelos ocultos de Markov (HMMs o *Hidden Markov Models*) [25] [34].

2.1.2. Comprensión del lenguaje natural

La información contenida en textos escritos de manera natural presenta el inconveniente de no seguir un modelo estructurado, pues están escritos libremente [33]. Por este motivo, es necesario preparar el contenido textual mediante procesos de múltiples fases para obtener datos estructurados que puedan ser tratados por métodos estadísticos [33].

De acuerdo a [33], el procedimiento llevado a cabo para poder convertir información textual en datos procesables numéricamente es el siguiente:

En primer lugar, el texto es analizado y *tokenizado* para obtener un conjunto de secuencias alfanuméricas que se corresponden a las expresiones empleadas en el texto. Estas palabras *tokenizadas* suelen estar separadas por espacios en blanco o caracteres especiales como *hashtags* o signos de exclamación, pero esto depende de la naturaleza del proyecto. Este esquema toma el nombre de bolsa de palabras (*Bag of Words*), y designa a la agrupación no ordenada de estos términos sin tener en cuenta las relaciones sintácticas o gramaticales pero manteniendo la multiplicidad. De esta forma, el grupo de diferentes palabras contenidas en este conjunto toma el nombre de tipo, mientras que cada instancia individual de las mismas, *token*. Todos los tipos conforman el vocabulario.

En segundo lugar, se aplica un pre-procesado al texto, pasando todas las palabras

a minúscula. Otras operaciones realizadas consisten en la eliminación de números o la corrección de palabras mal escritas.

Existen alternativas al *Bag of Words*, como por ejemplo, *Part of speech tagging*, el cual es utilizado en [12]. Esta técnica, a diferencia de BoW, permite obtener la estructura de las frases y conocer cuáles son las relaciones sintácticas entre palabras. Una *Part of speech* o categoría gramatical permite clasificar las palabras según su tipo, y revelan información sobre las palabras vecinas y la estructura sintáctica [17]. En español son 9: Nombre, Determinante, Adjetivo, Pronombre, Verbo, Adverbio, Preposición, Conjunción e Interjección [17]. De esta forma, para poseer información más refinada acerca de las palabras del texto, se puede realizar un etiquetado morfosintáctico de la oración; se identifican las diferentes partes de la oración y se asigna una etiqueta sobre la categoría gramatical a cada una de las palabras de un texto de entrada [9]. Los modelos ocultos de Markov (HMM) permiten llevar a cabo un etiquetado de la categoría de las palabras [17] [38], aunque existen alternativas que funcionan mejor, como *Net-Tagger*, expuesta por [38] y donde se emplean redes neuronales. De esta forma, a partir de una secuencia de palabras se obtiene una secuencia de pares formados por la palabra y la etiqueta correspondiente indicando la categoría gramatical a la que pertenece dicha palabra [9].

Continuando con el procedimiento expuesto en [33], posteriormente, se trata la inflexión de los términos de dos maneras diferentes: mediante derivación o lematización. La derivación reduce los términos inflexionados a su raíz eliminando sus sufijos, mientras que la lematización emplea el lema de un término para transformar cada palabra inflexionada a su forma canónica.

Puede realizarse otra poda más a fin de evitar términos que no informan (los conocidos como *stop-words*) eliminando las palabras más comunes utilizadas en la lengua en la que está escrita el texto o en el ámbito al que pertenece. De la misma forma, aquellas palabras con un bajo número de apariciones también son eliminadas del vocabulario. De esta manera, el contenido textual es convertido en una estructura de datos procesable.

En el modelo de espacio vectorial, cada texto puede ser representado como un vector

$$v_i = \{t_{i1}, ..., t_{ij}, ..., t_{ip}\} \tag{2.1}$$

donde t_{ij} define a la importancia del j-ésimo término en el vector de texto v_i. De este modo, el vector puede estar contenido como parte de una matriz D con n filas (correspondientes a los textos) y p columnas (palabras). La importancia de un término j se suele dar por la

frecuencia en la que aparece (*term frequency*) en un texto *i*, aunque también se emplean otras medidas para establecer la importancia, como N-grama o la Frecuencia de término-Frecuencia de documento inversa [16]. Esta matriz de textos * términos es la estructura más empleada a día de hoy en muchas estrategias analíticas.

2.2. Análisis de sentimientos

2.2.1. Conceptualización

El análisis de sentimientos consiste en el procesamiento computacional de opiniones, sentimientos y subjetividad en un texto [32]. Su objetivo es encontrar opiniones, identificar sentimientos y clasificar su polaridad [32].

De acuerdo a [14], existen 5 enfoques que permiten modelar un sistema de análisis de sentimientos:

- Análisis de sentimientos a nivel de documento: El sistema analiza y descubre la polaridad de todo el documento, aunque no logra extraer opiniones específicas [39].
- Análisis de sentimientos a nivel de oración: El sistema analiza y descubre la polaridad de todas las oraciones, dando una opinión positiva, negativa o neutral acerca de cada oración [39]. Esta es la opción que se utilizará en este proyecto.
- Análisis de sentimientos basado en aspectos: El sistema se centra en el reconocimiento de todas las expresiones de sentimiento dentro de un documento y en los aspectos a los que se refieren [14]. Tiene como objetivo extraer los sentimientos acerca de entidades sentimentales y sus aspectos [39].
- Análisis de sentimientos comparativo: Se pretende identificar oraciones que contienen opiniones comparitivas, así como extraer las entidades preferibles en cada opinión [14].
- Adquisición de lexicón de sentimientos: Consiste en obtener el lexicón de sentimientos, el cual constituye el elemento más importante para los algoritmos de análisis de sentimientos [14].

No obstante, de acuerdo a [39], se podrían resumir en 3: AS basado en documentos, en oraciones y en entidades y aspectos.

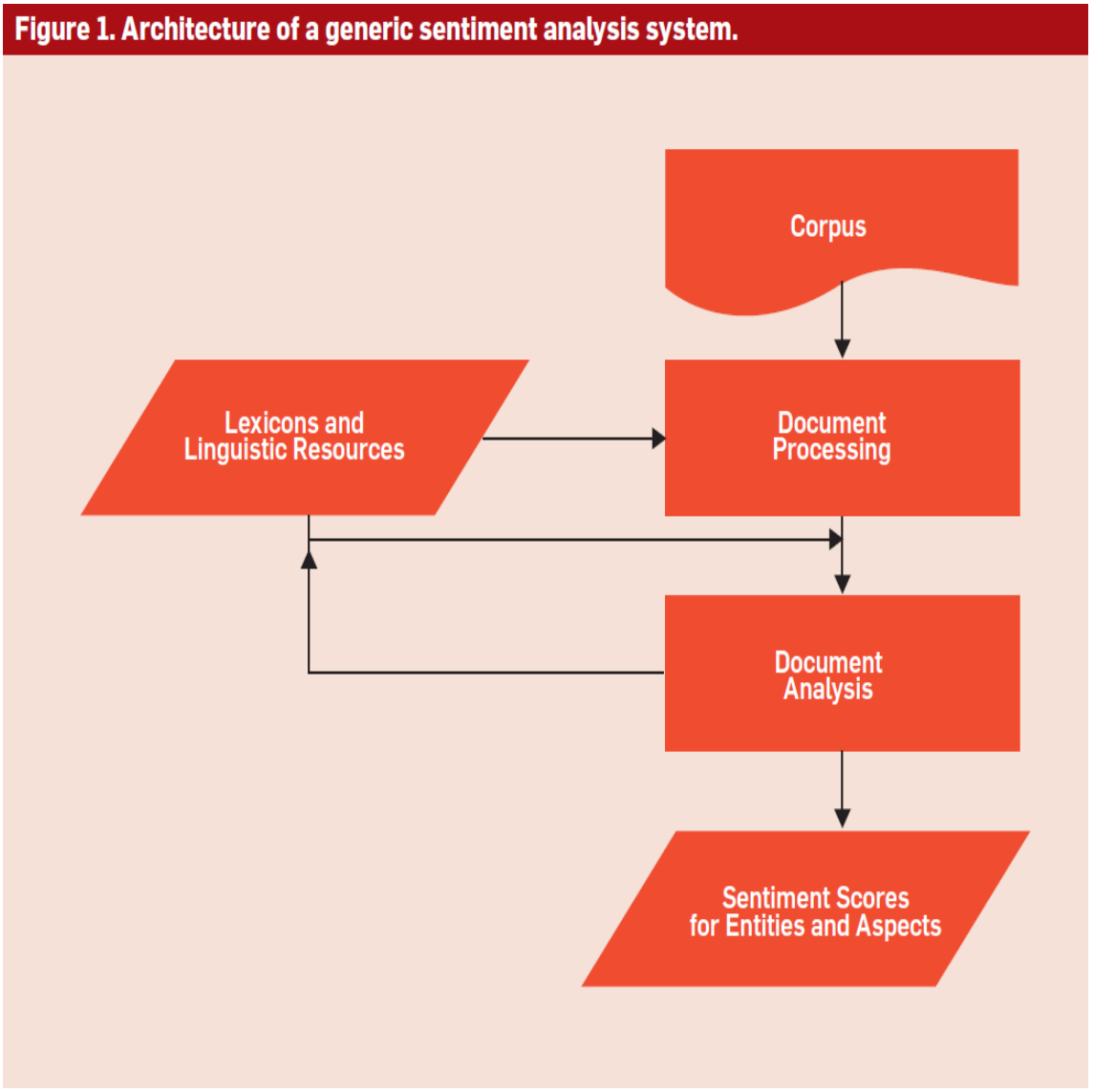


Figura 2.1: Sistema de análisis de sentimientos genérico. Imagen tomada de [14]

En [14] se presenta una arquitectura general de los sistemas de análisis de sentimiento, la cual puede ser observada en la imagen 2.1.

De acuerdo a [14], los documentos en el corpus son convertidos en texto y pre-procesados. El sistema utilizará un conjunto de léxicos y recursos lingüísticos, siendo el principal componente del sistema el módulo de análisis de documento. Este emplea los recursos lingüísticos para anotaciones de sentimientos en los documentos pre-procesados. Dependiendo del enfoque empleado, estas anotaciones se adjuntarán a documentos completos (en el caso de AS basado en documentos), a oraciones individuales (en el caso de AS basado en oraciones) o a aspectos o entidades concretas (para AS basado en aspectos). Estas anotaciones

serán el producto que devuelva el sistema, y pueden ser visualizadas utilizando diferentes herramientas.

2.2.2. Minería de opinión

La mayor aplicación del análisis de sentimientos es la minería de opinión (MO), la cual consiste en la extracción de sentimientos y opiniones a partir de un texto escrito en lenguaje natural a través de métodos computacionales [29]. Está compuesta por un conjunto de técnicas que aúnan la estadística, la lingüística y las ciencias de la computación, haciendo especial hincapié en la orientación semántica de los comentarios [33]. Cabe destacar que, pese a que normalmente se usan indistintamente debido a su estrecha relación conceptual, los términos minería de opinión y análisis de sentimientos no son exactamente lo mismo. El primero está orientado a la detección de la polaridad, mientras que el segundo al reconocimiento de los sentimientos. No obstante, debido a que ambas técnicas buscan cumplir el mismo objetivo y a que la primera depende de la segunda, estos conceptos se suelen utilizar como sinónimos [5]. A fin de evitar ambigüedades, en este trabajo se definirá minería de opinión bajo la definición expuesta al principio de este capítulo; la minería de opinión como detección de la polaridad, por tanto, será definida mediante esas mismas palabras: detección de la polaridad.

Los procedimientos de MO permiten transformar información cualitativa en cuantitativa, haciendo más fácil su tratamiento y permitiendo trabajar con datos objetivos. En los últimos años la minería de opinión ha comenzado a gozar de popularidad entre la comunidad científica y empresarial [5], pues es un campo que puede reportar múltiples beneficios a nivel de conocimiento y económico.

En el ámbito de las ciencias de la computación, la minería de opinión se trata de un problema NLP muy restringido, por lo que el progreso en este ámbito supone un avance considerable en otros problemas NLP [5]. Algunos sistemas lo emplean para realizar recomendaciones a sus usuarios (como los de los comercios en línea) o para llevar a cabo sistemas anti *spam*.

2.2.3. Análisis de sentimientos en el ámbito educativo

Como se ha explicado a lo largo de la introducción, la detección de la opinión general de los alumnos permite identificar problemas en el método educativo, así como potenciar los aspectos positivos.

Según [37], es necesario tener en cuenta las emociones de los estudiantes, pues están relacionadas con el aprendizaje e influyen en el conocimiento. Este hecho motiva y justifica enormemente la creación de software especializado en su detección en el ámbito educativo.

La teoría de la valoración, la lingüística funcional sistemática, el procesamiento del lenguaje natural y la concordancia de esquemas estadísticos han contribuido al desarrollo de la minería de opinión, así como el uso de técnicas de aprendizaje supervisado (como Naïve Bayes) y no supervisado (SO-PMI). [4]. De acuerdo a este artículo acerca de la minería de opinión orientada a la *e-learning*, la teoría de valoración es empleada para identificar la orientación semántica tratando palabras como grupos de atributos, mientras que la concordancia de esquemas estadísticos ha sido usada para identificar características en reseñas.

En [43] se emplea el análisis de sentimiento colectivo para explorar la relación entre las opiniones expresadas por los estudiantes y sus tasas de abandono en la plataforma de enseñanza en línea *Coursera*. Para recolectar las opiniones acerca de herramientas empleadas en el curso, se extraen las palabras de sentimiento positivas y negativas que están más asociadas con las palabras clave del tema de dichas herramientas. En este trabajo también se hace uso del análisis de supervivencia para estudiar cómo los sentimientos de los alumnos en una semana particular permiten predecir su continuidad en el curso.

2.2.4. Algoritmos de aprendizaje automático empleados

Existen dos enfoques a la hora de realizar análisis de sentimientos: el basado en *machine learning* y el basado en lexicones [2]. El primero usa técnicas de clasificación para clasificar el texto; el segundo usa diccionarios de sentimientos con palabras de opinión (tales como *SentiWordNet* [12]) y los compara con los datos para poder determinar la polaridad [2].

En el enfoque basado en lexicones las palabras son divididas en dos categorías: positivas y negativas. En él, un mensaje de texto es representado como una bolsa de palabras y se hace uso de un diccionario de palabras positivas y negativas a las que se les asigna un valor de sentimiento. Tras ser representado el mensaje, a todas las palabras positivas y negativas del mismo se les adjudican valores de sentimiento procedentes del diccionario [20].

Según [24], en el aprendizaje automático un algoritmo tiene que proponer una solución tras seguir los siguientes dos pasos:

1. Aprender de acuerdo con un corpus de datos de entrenamiento.

2. Clasificar datos de test (no utilizados) a partir de dicho modelo.

[24] discute varios métodos de clasificación, la cual consiste en el mapeo entre los patrones y las etiquetas. En este artículo se comparan la clasificación Naïve Bayes, la entropía máxima y SVM (*Support Vector Machine*), concluyendo que esta última es la que mejores resultados reporta actualmente.

Por su parte, [23] propone un nuevo método de minería de opinión llamado *OMLML*, el cual combina el enfoque basado en lexicón y el *machine learning*. Se hace uso de un procedimiento que, utilizando un método basado en el lexicón y las características textuales de las palabras y oraciones, determina la polaridad de las opiniones hacia una palabra objetivo. Tras haber mapeado el espacio de características en un vector de tres dimensiones, las opiniones se analizan y clasifican en base a un método de aprendizaje automático que utiliza una red neuronal propuesta en el artículo.

[16] implementa diez algoritmos de aprendizaje automático diferentes con dos algoritmos de extracción de características en cuatro bases de datos orientadas al análisis de sentimientos. Los algoritmos de *machine learning* son Naïve Bayes, Descenso de Gradiente Estocástico, SVM, Pasivo Agresivo, Entropía Máxima, AdaBoost, Naïve Bayes Multinomial, Naïve Bayes Bernouilli, Regresión de Cresta y Regresión Logística. Los algoritmos de extracción de características son N-grama y la Frecuencia de término–Frecuencia de documento inversa.

En [16] se sigue una metodología consistente en 4 pasos:

1. Los datos son pre-procesados: se eliminan palabras repetitivas además de caracteres y signos de puntuación no existentes en el idioma inglés .

2. Se extraen las características: se sustraen atributos confiables del contenido textual que derivan en una clasificación correcta. Se emplean N-grama y la Frecuencia de término–Frecuencia de documento inversa. El objetivo de este paso es mapear información textual a datos (o características) numéricos que puedan ser procesados por los algoritmos de *machine learning*.

3. Se emplean diferentes algoritmos de aprendizaje automático sobre los vectores de características para evaluar su desempeño.

4. Se lleva a cabo la validación cruzada de K pliegues para determinar qué algoritmo funciona mejor.

En [16] se concluye que los algoritmos de aprendizaje automático que mejor funcionan son Pasivo Agresivo y Regresión de Cresta, pues arrojan una precisión que va desde el 87 % hasta el 99,96 %. No obstante, Regresión Lineal y SVM también tienen un rendimiento aceptable. No obstante, en [21] se indica que usando el modelo *Bag of words* en problemas de clasificación binaria a nivel de documento, los clasificadores que mejor funcionan son Naïve Bayes, SVMs y la Entropía máxima. En esta línea, es destacable el trabajo realizado en [13], en el que se crea un corpus de opiniones etiquetadas con polaridades de nombre *SentiTEXT* (entre otros corpus), el cual está orientado al ámbito educativo y es evaluado mediante diferentes modelos de *Machine learning* y *deep learning*. Las dos implementaciones de SVM, SVC y SVC lineal, proporcionan precisiones del 90 %, mientras que las dos de Naïve Bayes, Multinomial NB y Bernouilli NB, del 87 %. De acuerdo a [19], donde se hace un estudio acerca del rol que desempeña *SentiWordNet* en la minería de opinión, de entre 102 trabajos científicos SVM y Naïve Bayes son los dos clasificadores más utilizados, siendo empleados en 36 y 14 trabajos respectivamente. Esto demuestra que estos dos clasificadores funcionan bien en tareas que involucran a la naturaleza de este proyecto.

2.2.5. Vacío de conocimiento en el análisis de sentimientos orientado al ámbito académico

De acuerdo a [35], a día de hoy son necesarias investigaciones que cubran el aprendizaje de conceptos. De la misma manera, también lo es el desarrollo de más sistemas que se encarguen de detectar y resumir opiniones de forma automática; esta es la principal carencia que cubrirá este proyecto. Además, en el ámbito de la minería de opinión, se tienen que desarrollar bases de conocimiento de sentido común y métodos de razonamiento mejores.

En [1] se realiza un prototipo de arquitectura de análisis de sentimientos orientada a la enseñanza superior de nombre *LADEL*, la cual cumple con objetivos muy parecidos a los de este proyecto. En este trabajo se han encontrado los siguientes problemas:

- Desarrollo de un módulo que informe acerca de tendencias tales como emociones a lo largo del tiempo.
- Comprensión de cómo determinados estados cognitivos pueden provocar o bloquear el camino de aprendizaje del modelo.

De acuerdo a la investigación llevada a cabo en [22], estas son las mayores necesidades

que enfrenta el análisis de sentimientos en los siguientes campos:

- Estructura y tamaño de los conjuntos de datos: Actualmente existe la necesidad de un formato general para dar forma a la información. El empleo de un formato estructurado normalizado (ya sea en un archivo de tipo *XML* o *JSON*) sería de gran utilidad para estandarizar la generación de *datasets* para el análisis de sentimientos en el ámbito educativo.

- Detección de emociones: Hay que tener en cuenta las emociones contenidas en las opiniones a fin de poder identificar y abordar las cuestiones relacionadas con la materia que está siendo tratada. De la misma forma, es necesario poder tratar emoticonos, pues estos poseen un gran significado emocional.

- Métricas de evaluación: Como respuesta a la escasa documentación acerca de las métricas empleadas para evaluar el comportamiento de los modelos de análisis de sentimientos en la literatura actual, se propone el uso de métricas tales como el Coeficiente Kappa de Cohen o el coeficiente de correlación de Pearson para medir la correlación entre los resultados del sistema y los de los datos etiquetados.

- Soluciones estandarizadas: Muchas de las soluciones propuestas en en el análisis de sentimiento son aún muy jóvenes y sirven para tareas muy específicas en diferentes escenarios. Todavía falta un proceso de estandarización que garantice la calidad, seguridad y fiabilidad de las soluciones y de los sistemas desarrollados para análisis de sentimientos.

- Contextualización y conceptualización del sentimiento: Las soluciones que usan técnicas de *Machine Learning* y *Deep Learning* tienen que incorporar contexto semántico mediante recursos léxicos tales como *Wordnet*, *SentiWordNet* y *SenticNet*. También deben implementar representación semántica mediante ontologías, con el objetivo de capturar opiniones, pensamientos y actitudes de usuarios a partir de un texto.

Capítulo 3

Objetivos

3.1. Objetivo general del proyecto

El objetivo de este trabajo es ofrecer a las instituciones educativas una solución que les permita obtener información gráfica acerca de las percepciones de los estudiantes frente a la labor de sus docentes a partir de las opiniones de sus alumnos. Para ello, se emplearán técnicas de minería de opinión y de aprendizaje automático, concretamente, técnicas de aprendizaje supervisado. Mediante una interfaz gráfica, el usuario podrá pedirle al sistema que evalúe un conjunto de comentarios contenidos en un archivo de extensión *.csv*. Tras haberse realizado el análisis, se le mostrará de manera gráfica cuántos comentarios han sido etiquetados como positivos, negativos y neutros, así como el porcentaje del total que representa cada cantidad y cuál de ellos es el más frecuente.

Se utilizará validación cruzada con K = 5 para comprobar el correcto funcionamiento del algoritmo que vertebra la funcionalidad de la aplicación, debido a que es la técnica que mejor se ajusta a su naturaleza. El objetivo principal es llegar a tener una precisión del 90 % como mínimo.

3.2. Objetivos específicos

1. Establecer el mejor algoritmo entre 3 de los que mejores resultados han reportado dentro de los revisados en el estado del arte y que han sido aplicados a contextos universitarios.

2. Construir el sistema de minería de opinión, el cual incluye el análisis de sentimientos junto a su modelo de IA correspondiente; es decir, se construirá el software. Este

objetivo incluye la realización de la interfaz gráfica.

3. Evaluar la usabilidad y funcionalidad del sistema construido a partir de métricas ya establecidas, así como determinar la calidad de las predicciones proporcionadas por los modelos estudiados.

3.3. Metodología de trabajo

En este apartado se explicará la metodología de trabajo que se llevará cabo para completar este proyecto.

3.3.1. Objetivo 1: Establecer el mejor algoritmo entre los 3 mejores

En primer lugar, para determinar qué método de minería de opinión funciona mejor, se seleccionarán tres algoritmos diferentes que, de acuerdo al estado del arte, son los que mejor funcionan: Pasivo Agresivo, SVM y Naïve Bayes. Si bien Regresión de Cresta también ofrece resultados muy buenos de acuerdo a [16], SVM y Naïve Bayes son empleados en un gran número de trabajos [19], por lo que también serán tenidos en cuenta. Junto con estos algoritmos se emplearán también los algoritmos de extracción de características Bag of Words y Frecuencia de término–Frecuencia de documento inversa. En esta línea, se tomará un enfoque fuertemente inspirado en [16]. El objetivo principal, de no tener ningún contratiempo, es construir tres modelos diferentes con estos algoritmos y optar por el que mejores resultados reporte; no obstante, de no ser posible, se optará por utilizar directamente SVM, pues queda avalado por el estado del arte como el mejor candidato [16] [13] [19]. En caso de no poseer el tiempo de evaluar todas las combinaciones de algoritmos posibles, se emplearía Frecuencia de término – Frecuencia de documento inversa, pues contiene información acerca de la importancia de las palabras, ideal para el análisis de sentimientos.

3.3.2. Objetivo 2: Construcción del sistema de minería de opinión

Troncalmente relacionado con el primer objetivo, se construirá el modelo de minería de opinión empleando SVM o tres modelos con Pasivo Agresivo, SVM y Naïve Bayes; dependerá de las restricciones técnicas y temporales. De construir 3 modelos, se optaría finalmente por el que mejores resultados proporcione. Se hará uso del lenguaje de programación *Python* para construir el modelo de minería de opinión; se empleará la librería

scikit-learn para el aprendizaje automático y *Natural Language Toolkit* (*NLTK*) para el procesamiento del lenguaje natural. La estructura del modelo se basará en la indicada en la imagen 2.1. Se emplearán archivos de extensión *csv* para contener la información que será procesada, además de utilizar *HTML* para construir la interfaz gráfica. Siguiendo el procedimiento llevado a cabo en [16], se utilizará validación cruzada con 5 grupos para asegurar que los algoritmos de *machine learning* funcionan correctamente.

3.3.3. Objetivo 3: Evaluación del sistema construido

En último lugar, se evaluará con un caso práctico el rendimiento de la aplicación. Se utilizarán dos criterios fundamentales: la usabilidad y la funcionalidad. Con el primero, se define como la gente usa el programa; con el segundo, qué se puede hacer con él. Se hará uso del sistema *SUS*, el cual permite medir estos parámetros a partir de preguntas. Para dar la aplicación por correcta, se buscará tener una puntuación de 70 o superior, y se utilizará la página web `https://uiuxtrend.com/sus-calculator/`. Adicionalmente, se emplearán curvas ROC para evaluar el funcionamiento de los modelos de *Machine Learning*. Para complementar estos resultados, se inicializará desde cero un modelo que haga uso de la mejor combinación de algoritmos de representación y clasificación, se entrenará con un 80 % de los datos y se testeará con el 20 % restante. El objetivo último es verificar el correcto funcionamiento de la aplicación mediante la armonía y coherencia de todos los resultados.

Capítulo 4

Identificación de Requisitos

Para poder identificar los requisitos por parte de un posible usuario de la aplicación, se hará uso de historias de usuario. En cada una de ellas se identifica qué usuario utilizaría la aplicación, qué desearía hacer y con qué objetivo. Posteriormente se define el criterio de aceptación de la historia, es decir, qué condiciones se han de cumplir para darla por realizada. En este caso, al ser una aplicación orientada a la minería de opinión de alumnos, el papel del usuario lo toma un profesor.

Historia	**Criterio de aceptación**
Como profesor, quiero poder introducir mi archivo *csv* con los comentarios que los alumnos han dejado acerca de mí y poder obtener los comentarios agrupados por cada sentimiento, así como verlos todos con sus predicciones adjuntas.	▪ Se muestra una tabla que contenga todos los comentarios sin agrupar pero con sus sentimientos predichos. ▪ Se muestra una tabla que muestre primero los comentarios positivos, después los neutros y finalmente los negativos.

Tabla 4.1: Historia 1

Historia	**Criterio de aceptación**
Como usuario, quiero utilizar una aplicación que me ofrezca la máxima fiabilidad en las predicciones.	▪ Se entrenan tres modelos que hacen uso de los mejores algoritmos de clasificación descritos en el estado del arte, probando tanto Bag of Words como TF-IDF en cada uno. ▪ Se obtiene un modelo que proporciona un mínimo de 90 % de precisión empleando la métrica *f1_macro* de entre los 3 mencionados anteriormente. ▪ Se emplea un mínimo de 6000 comentarios para entrenar a los modelos.

Tabla 4.2: Historia 2

Historia	**Criterio de aceptación**
Como profesor, quiero poder introducir mi archivo *csv* con los comentarios que los alumnos han dejado acerca de mí y poder obtener un resumen de cuántos comentarios son negativos, neutros y positivos, así como conocer cuál es el más frecuente.	▪ Se muestra el número de comentarios positivos y el porcentaje que suponen respecto al total. ▪ Se muestra el número de comentarios neutros y el porcentaje que suponen respecto al total. ▪ Se muestra el número de comentarios negativos y el porcentaje que suponen respecto al total. ▪ Se indica cuál es el sentimiento más frecuente entre todos los comentarios.

Tabla 4.3: Historia 3

Historia	**Criterio de aceptación**
Como profesor, quiero poder hacer uso de la aplicación de forma sencilla para evitar complicaciones.	▪ Todos los elementos gráficos están contenidos en una misma página web. ▪ Solo existen dos botones. ▪ Las instrucciones están contenidas en un párrafo al comienzo de la página. ▪ Se designará el color verde a los comentarios positivos, el azul a los neutros y el rojo a los negativos.

Tabla 4.4: Historia 3

Capítulo 5

Desarrollo del trabajo

A lo largo de este capítulo se detallarán qué procedimientos se han llevado a cabo para completar este proyecto. Se abarcará desde las tecnologías y metodologías empleadas hasta casuísticas y casos de prueba. Se detallará el tratamiento de datos, los algoritmos empleados y la justificación de su elección así como el recorrido general de un caso de uso. El diagrama de la imagen 5.1 explica cómo funciona la aplicación, describiendo el ciclo de vida de una petición. En el repositorio `https://github.com/belzu/TFM-IA-UNIR-Students-Opinion-Mining-WebApp` puede encontrarse el código que ha hecho posible esta aplicación.

5.1. Tecnología empleada

A continuación se enunciarán todas las tecnologías que han sido utilizadas para desarrollar esta aplicación; desde frameworks, hasta lenguajes de programación pasando por librerías. Se mencionarán las más importantes en virtud de la sintetización. Cabe destacar que en el alcance de esta sección modelos de *Machine Learning* o modelos de análisis de sentimientos hacen referencia a los modelos de procesamiento de texto y de clasificación; se hace uso de estas expresiones pues conceptualmente son uno mismo en el modelo que se despliega en esta aplicación, además de para evitar repeticiones.

5.1.1. Python

Su facilidad de uso, la enorme oferta de librerías que permiten llevar a cabo modelos de *Machine Learning* y la portabilidad han sido los motivos principales detrás del empleo de este lenguaje de programación en este proyecto. Se ha hecho uso de la versión 3.8.16 y se

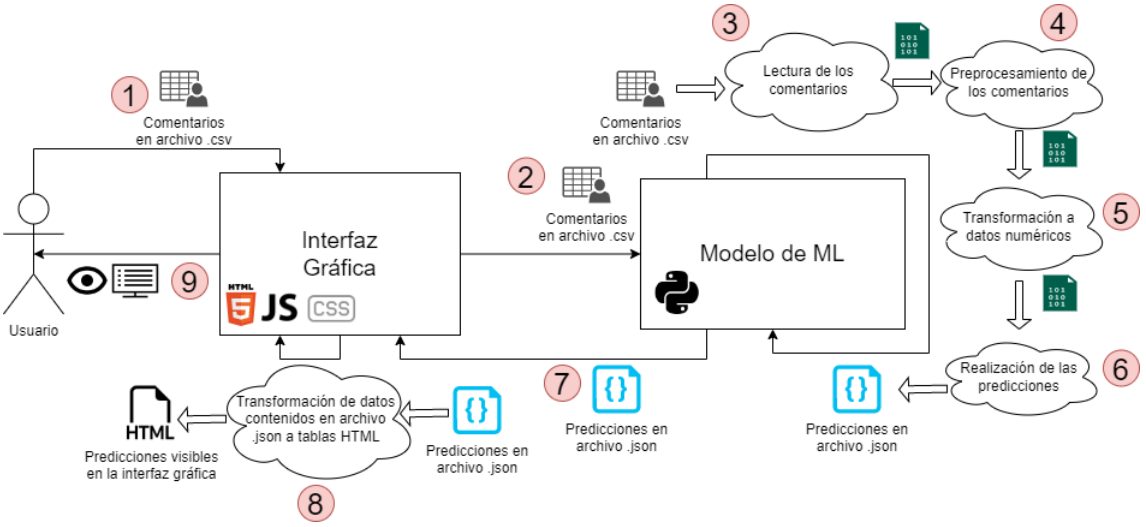


Figura 5.1: Esquema de la aplicación mostrando el ciclo de vida de una petición así como los lenguajes que soportan cada módulo.

ha utilizado para crear los modelos de *Machine Learning* así como para desplegarlos. En resumen, se ha empleado para crear todo el sistema de minería de opinión. Adicionalmente, también ha sido utilizado para obtener los comentarios en formato *.csv* a partir de archivos *.xlsx*.

5.1.2. Google Colab

Este servicio proporcionado por Google consiste en un *notebook* de *Jupyter* que se ejecuta en la nube. Al tratarse de un cuaderno, permite ejecutar fracciones de código de forma individual, compartiendo una memoria común; de esta manera, se puede separar el código en diferentes secciones, lo cual es ideal para realizar tests y desarrollar modelos. Además, permite realizar ejecuciones mediante GPU, acelerando enormemente los cálculos implicados en los procesos de aprendizaje, predicción y testeo. Se ha empleado para desarrollar, entrenar y testear los modelos de *Machine Learning*.

5.1.3. Pandas

Librería de Python orientada a la manipulación, análisis y tratamiento de datos. Ha sido utilizada para obtener los comentarios contenidos en los archivos de extensión *.xlsx* y para generar los ficheros de extensión *.csv* que permiten al modelo leerlos y serializar sus predicciones.

5.1.4. NLTK

Conjunto de librerías para el procesamiento de lenguaje natural en Python, orientado a trabajar con datos lingüísticos humanos. Ha sido utilizado para descargar *stop words* en castellano y para *tokenizar* los comentarios, y ha ayudado a corregir palabras mal escritas.

5.1.5. SpellChecker

Esta librería, como su propio nombre inidica, está orientada a la corrección ortográfica, y es para lo que ha sido empleada.

5.1.6. Stanza Language

Esta librería consiste en un paquete de análisis de lenguaje natural en Python; se ha utilizado para lematizar palabras.

5.1.7. Sklearn

Consiste en una biblioteca orientada al aprendizaje automático en Python. Posee algoritmos de clasificación, regresión y análisis de grupos además de ser compatible con otras bibliotecas como *numpy* y *scipy*. No solo permite crear modelos de aprendizaje automático, si no que también posibilita testearlos e incluso probar diferentes combinaciones de hiperparámetros, eligiendo la mejor. Ha sido utilizada para crear los modelos de minería de opinión (tanto de clasificación como representación de datos), probarlos mediante diferentes técnicas de validación y escoger el mejor. Ha permitido instanciar Bag of Words, TF-IDF, SVC, Naïve Bayes Multinomial y Pasivo Agresivo, testearlos mediante validación cruzada y escoger el mejor algoritmo en función de los resultados.

5.1.8. Javascript

Se trata de un lenguaje de programación interpretado que permite implementar funcionalidades complejas en páginas web. Ha sido utilizado para conectar la interfaz gráfica con el modelo desplegado enviando el archivo *.csv* con los comentarios, para leer las respuestas proporcionadas por el modelo y para mostrar los resultados.

5.1.9. Flask

Es un microframework de Python que permite crear aplicaciones web para este lenguaje de programación. Es decir, permite convertir un archivo de extensión *.py* en una aplicación web fácilmente desplegable. Ha sido utilizado para desplegar el modelo de aprendizaje automático.

5.1.10. HTML

Consiste en un lenguaje de marcado que permite estructurar páginas web mediante etiquetas. En un archivo *.html* se definen los diferentes elementos que van a componer una página web, las características que tienen y la posición que ocuparán así como qué harán cada uno de ellos. Ha sido utilizado junto a *CSS* para crear la interfaz gráfica con la que el usuario interactuará.

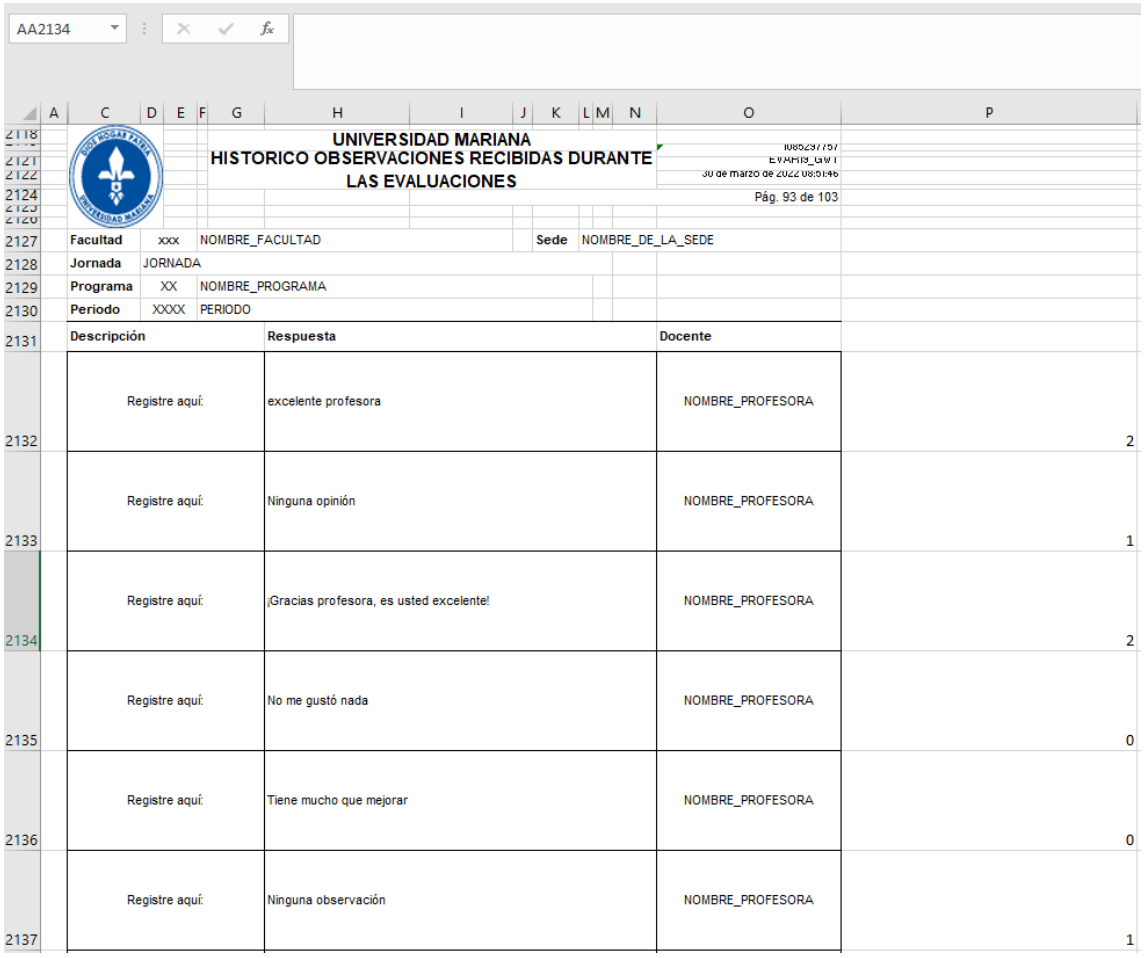
5.2. Datos utilizados

Para poder entrenar el modelo de aprendizaje automático se han utilizado aproximadamente 8700 comentarios provenientes de alumnos acerca de sus profesores en la Universidad Mariana de Colombia. Para preservar el anonimato de los profesores, se han modificado los datos. Se ha contado con un total de 20 hojas de cálculo de extensión *.xlsx* que contienen los comentarios de los alumnos y el docente al que van dirigido. Se han introducido uno a uno los sentimientos adjuntos a cada comentario manualmente, siendo:

- 0: Comentario negativo; descontento con el profesor, la asignatura y/o la institución académica.
- 1: Comentario neutro; nada que comentar o la ausencia de un sentimiento positivo o negativo.
- 2: Comentario positivo; conformidad con el profesor, la asignatura y/o la institución académica.

Lamentablemente, debido a restricciones temporales, solo se han podido utilizar los comentarios y sus sentimientos; no obstante, la aplicación está preparada para trabajar también con los siguientes datos también presentes en las hojas de cálculo:

- Facultad



AA2134

UNIVERSIDAD MARIANA
HISTORICO OBSERVACIONES RECIBIDAS DURANTE LAS EVALUACIONES

Pág. 93 de 103

Facultad	xxx	NOMBRE_FACULTAD	Sede	NOMBRE_DE_LA_SEDE	
Jornada	JORNADA				
Programa	XX	NOMBRE_PROGRAMA			
Periodo	XXXX	PERIODO			

Descripción	Respuesta	Docente	
Registre aquí:	excelente profesora	NOMBRE_PROFESORA	2
Registre aquí:	Ninguna opinión	NOMBRE_PROFESORA	1
Registre aquí:	¡Gracias profesora, es usted excelente!	NOMBRE_PROFESORA	2
Registre aquí:	No me gustó nada	NOMBRE_PROFESORA	0
Registre aquí:	Tiene mucho que mejorar	NOMBRE_PROFESORA	0
Registre aquí:	Ninguna observación	NOMBRE_PROFESORA	1

Figura 5.2: Ejemplo de los comentarios contenidos en excel con sus sentimientos adjuntos

- Jornada
- Programa
- Periodo
- Sede

En la imagen 5.2 se puede observar un ejemplo de los datos en su formato original, con la unica adición de los sentimientos a los que se refieren. Para poder trabajar con ellos ha sido necesario transformarlos a formato *.csv*, para lo que se ha creado un script. En la imagen 5.3 se pueden distinguir estos mismos datos en formato *.csv*.

```
NOMBRE_FACULTAD JORNADA NOMBRE_PROGRAMA PERIODO NOMBRE_DE_LA_SEDE   excelente profesora     NOMBRE_PROFESORA    2.0
NOMBRE_FACULTAD JORNADA NOMBRE_PROGRAMA PERIODO NOMBRE_DE_LA_SEDE   Ninguna opinión NOMBRE_PROFESORA    1.0
NOMBRE_FACULTAD JORNADA NOMBRE_PROGRAMA PERIODO NOMBRE_DE_LA_SEDE   ¡Gracias profesora, es usted excelente! NOMBRE_PROFESORA    2.0
NOMBRE_FACULTAD JORNADA NOMBRE_PROGRAMA PERIODO NOMBRE_DE_LA_SEDE   No me gustó nada    NOMBRE_PROFESORA    0.0
NOMBRE_FACULTAD JORNADA NOMBRE_PROGRAMA PERIODO NOMBRE_DE_LA_SEDE   Tiene mucho que mejorar NOMBRE_PROFESORA    0.0
NOMBRE_FACULTAD JORNADA NOMBRE_PROGRAMA PERIODO NOMBRE_DE_LA_SEDE   Ninguna observación NOMBRE_PROFESORA    1.0
```

Figura 5.3: Ejemplo de los comentarios contenidos en csv con sus sentimientos adjuntos

5.3. Preprocesamiento de los datos

En primer lugar, se han realizado correcciones a palabras o frases mal escritas en los archivos de extensión *.xlsx*, a fin de disminuir posibles problemas en el procesamiento de datos. Este proceso incluyó sustituir expresiones mal escritas, como por ejemplo, *exelente* por *excelente*. Cabe mencionar que, al tratarse de miles de comentarios, no se han podido realizar correcciones individualizadas, si no generales. En segundo lugar, se han sustituido comentarios consistentes en símbolos no numéricos o expresiones carentes de sentido por vacío (resultando en comentarios en blanco), a fin de ser deshechados por la aplicación en la fase de procesamiento. Posteriormente, se han etiquetado manualmente los comentarios, siguiendo las siguientes normas:

- Si el comentario se muestra de forma clara e inequívoca como positivo, se califica como tal.
- Si el comentario se muestra de forma clara e inequívoca como negativo, se califica como tal.
- Si el comentario incide únicamente en la necesidad de mejora en uno o varios aspectos, se califica como negativo.
- Si el comentario indica que no hay nada que decir o que no posee comentarios o no muestra una tendencia clara, se califica como neutro.
- Si el comentario consiste únicamente en un número del 0 al 10 se considera que designa una valoración, por lo que si este número es inferior a 5, se califica como negativo; si está entre 5 y 7, como neutro; si es mayor o igual a 7, como positivo.
- Si el número de aspectos positivos mencionados en el comentario es mayor que el número de elementos negativos, se califica como positivo; si es menor, como negativo; si es el mismo, como neutro.

La imagen 5.4 indica que el dataset no está balanceado; sin embargo, el modelo no cae en

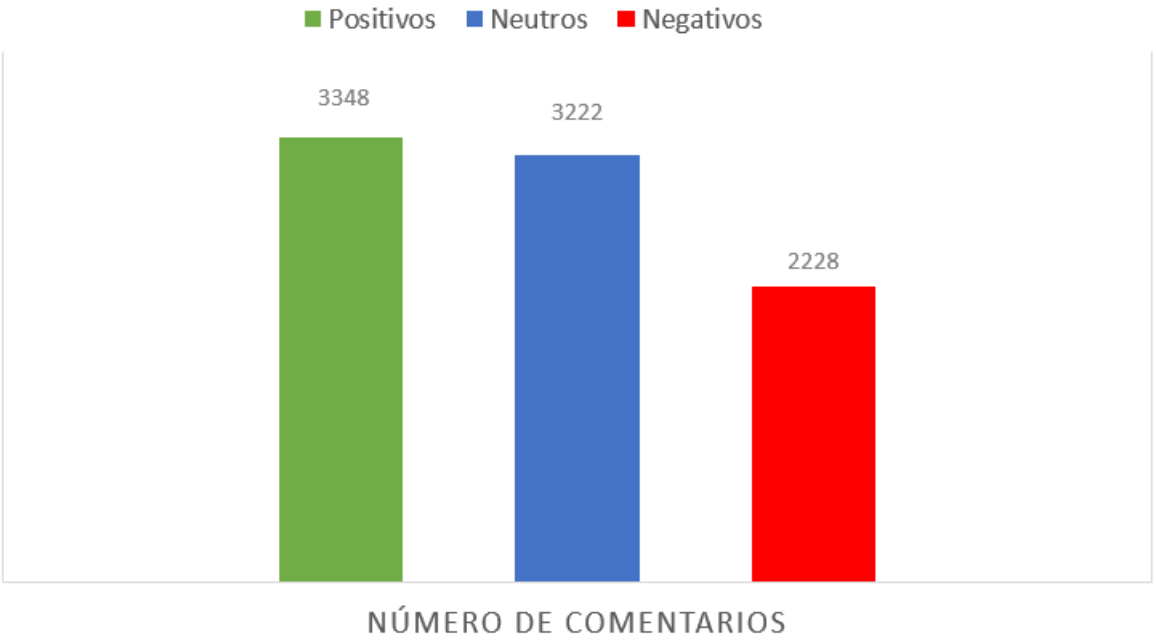


Figura 5.4: Número de comentarios agrupados por el sentimiento que transmiten

overfitting, por lo que la proporción de cada clase respecto al total no supone un problema en este caso.

La aplicación permite 3 transformaciones esenciales a los comentarios, los cuales se representan como listas de cadenas de caracteres o *Strings*, donde cada cadena representa un comentario. Estos procesos pueden ser desactivados o activados a voluntad de forma individual.

5.3.1. Corrección de palabras

Los comentarios no siempre están bien escritos, y en muchas ocasiones padecen de la ausencia de una letra o de permutación de varias. Este hecho se traduce en el análisis por parte del modelo de varias palabras que en realidad son la misma, y no solo puede incurrir en que este pierda precisión, si no también en el uso de una cantidad de memoria excesiva.

A fin de evitar en la medida de lo posible esta situación, cada comentario de la lista de comentarios se desglosa en tokens (instancias individuales de palabras y signos de puntuación), y se corrigen términos mal escritos sustituyéndolos por la corrección propuesta por la librería *SpellChecker*.

5.3.2. Conversión de palabras a sinónimos

Existen casos donde una palabra puede compartir su significado con otras muchas. Al trabajar con técnicas de aprendizaje supervisado, los modelos tienen una dependencia estrecha con las palabras contenidas en el dataset de aprendizaje para realizar predicciones correctas. Sin embargo, no siempre es posible contar con conjuntos de datos que contengan un vocabulario extenso, por lo que la aparición de palabras o expresiones desconocidas puede suponer un problema a la hora de realizar predicciones, aunque estos términos sean sinónimos de otros que aparecen con frecuencia.

Para solventar ese problema, se sustituyen en cada comentario un conjunto discreto de palabras y expresiones por un sinónimo contenido en el dataset de entrenamiento. Por ejemplo, el vocablo *maestro* o cualquiera de sus sinónimos es sustituido por *docente*, mientras que *desorganizado* por *malo*, una palabra muy recurrente en el dataset de entrenamiento. De esta forma, una frase que podría resultar desconocida para el modelo como "*es un maestro muy desorganizado*"se convierte en "*es un docente muy malo*", el cual se clasifica como negativo. No obstante, esta técnica está orientada a ser utilizada en la predicción y no en el entrenamiento, pues reduce el vocabulario del modelo; por ese mismo motivo, no ha sido empleada en dicha fase.

5.3.3. Lematización de palabras

En el ámbito de la lexicografía y morfología se define como lema a la forma básica de una palabra, constituyendo normalmente las entradas de los diccionarios. Por ejemplo, los lemas de las palabras *profesores* y *profesoras* son *profesor* y *profesora* respectivamente, mientras que la de *viajará*, *viajando*, *viajó* y *viajaron* es *viajar*. Este proceso permite condensar palabras en una sola, y, de la misma forma que en la conversión de palabras a sinónimos, simplificar los conceptos contenidos en cada comentario y reducir la memoria empleada.

De esta manera, se ha sustituido cada palabra de cada comentario por su lema mediante el uso de la librería *StanzaLanguage*.

5.4. Modelos empleados

En esta sección se comentarán qué algoritmos de extracción de características y qué algoritmos de clasificación han sido utilizados para construir los modelos de representación

y clasificación de la información respectivamente.

Se mostrarán las comprobaciones de qué algoritmos han funcionado en cada caso, a fin de determinar la combinación de algoritmos que resulta en la construcción del modelo de minería de opinión óptimo.

5.4.1. Modelos de representación

Existe la necesidad de transformar los comentarios representados mediante texto en datos numéricos interpretables por los algoritmos de aprendizaje automático. De acuerdo a lo mencionado en el estado del arte y en los objetivos, los algoritmos de extracción de características que se utilizarán serán *Bag of Words* y Frecuencia de término - Frecuencia de documento inversa (*TF-IDF*). En el contexto de este proyecto, un documento se refiere a un comentario; concretamente, a cualquier texto con más de 40 caracteres.

Bag of Words

Consiste en una representación de texto donde se muestra el número de veces que aparece una palabra dentro de un documento; en este caso, en cada comentario. De esta forma, se mide la presencialidad de cada palabra dentro del conjunto de comentarios. Los datos se alojan en una matriz M de dimensión $D * N$, donde D es el número de documentos (comentarios) y N el número de diferentes palabras en todo el conjunto de documentos. De esta manera, siendo $0 \leqslant i \leqslant D$ y $0 \leqslant j \leqslant N$, el elemento $M_{i,j}$ representa el número de veces que aparece la palabra de índice j en el documento de índice i. En la figura 5.5 se puede ver un ejemplo de esta casuística. Dentro de la librería *Sklearn*, la clase encargada de implementarlo es *CountVectorizer*.

TF-IDF

Expresa cuán importante es una palabra dentro de un documento en comparación con el resto, lo cual es muy importante a la hora de clasificar texto. La fórmula [30] que lo

```python
#Inicialización del objeto donde se aloja la estructura "Bag of Words"
bag_of_words = CountVectorizer()
#Conjunto de comentarios (siendo cada comentario un documento)
comentarios = ["El perro pasea feliz por el bosque",
               "El perro ya no pasea feliz por el bosque"]
#Se transforman los comentarios a datos numéricos
count_matrix = bag_of_words.fit_transform(comentarios)
#Se imprime el diccionario que identifica mediante su índice correspondiente
#a cada palabra presente en el conjunto de comentarios
print(sorted(bag_of_words.vocabulary_.items()))
#Se imprime la matriz (m) donde se indica cuántas veces (m(i,j)) aparece cada
#palabra (j) en cada documento (i)
print(count_matrix)
#Se transforma la matriz en un array
count_array = count_matrix.toarray()
#Se imprime cada fila de la matriz, donde se indica cuántas veces está presente
#en cada comentario cada una de las palabras del conjunto de comentarios
for i,j in zip (comentarios, count_array):
    print(str(i) + " " + str(j))
```

```
[('bosque', 0), ('el', 1), ('feliz', 2), ('no', 3), ('pasea', 4), ('perro', 5), ('por', 6), ('ya', 7)]
  (0, 1)        2
  (0, 5)        1
  (0, 4)        1
  (0, 2)        1
  (0, 6)        1
  (0, 0)        1
  (1, 1)        2
  (1, 5)        1
  (1, 4)        1
  (1, 2)        1
  (1, 6)        1
  (1, 0)        1
  (1, 7)        1
  (1, 3)        1
El perro pasea feliz por el bosque [1 2 1 0 1 1 1 0]
El perro ya no pasea feliz por el bosque [1 2 1 1 1 1 1 1]
```

Figura 5.5: Ejemplo de bag of words con Sklearn

define es la 5.1.

$$TF(i,j) = \frac{\text{Frecuencia del término } i \text{ en el documento } j}{\text{Número de palabras totales en el documento } j},$$
$$IDF(i) = \log_2\left(\frac{\text{Número total de documentos}}{\text{Número de documentos con la palabra } i}\right),$$

Para el término i y documento j :

$$TF - IDF(i,j) = TF(i,j) * IDF(i) \tag{5.1}$$

En la figura 5.6 se puede ver un ejemplo de esta casuística. Dentro de la librería *Sklearn*, la clase encargada de implementarlo es *TfidfVectorizer*.

```
tf_idf = TfidfVectorizer()
#Conjunto de comentarios (siendo cada comentario un documento)
comentarios = ["El perro pasea feliz por el bosque",
               "El perro ya no pasea feliz por el bosque"]
#Se transforman los comentarios a datos numéricos
count_matrix = tf_idf.fit_transform(comentarios)
#Se imprime el diccionario que identifica mediante su índice correspondiente
#a cada palabra presente en el conjunto de comentarios
print(sorted(tf_idf.vocabulary_.items()))
#Se imprime la matriz (m) donde se indica la puntuación TF-IDF de cada
#palabra (j) en cada documento (i)
print(count_matrix)
#Se transforma la matriz en un array
count_array = count_matrix.toarray()
#Se imprime cada fila de la matriz, donde se indica la importancia de cada
#palabra en cada comentario
for i,j in zip (comentarios, count_array):
    print(str(i) + " " + str(j))
```

```
[('bosque', 0), ('el', 1), ('feliz', 2), ('no', 3), ('pasea', 4), ('perro', 5), ('por', 6), ('ya', 7)]
  (0, 0)        0.3333333333333333
  (0, 6)        0.3333333333333333
  (0, 2)        0.3333333333333333
  (0, 4)        0.3333333333333333
  (0, 5)        0.3333333333333333
  (0, 1)        0.6666666666666666
  (1, 3)        0.39054766417182263
  (1, 7)        0.39054766417182263
  (1, 0)        0.2778778796561673
  (1, 6)        0.2778778796561673
  (1, 2)        0.2778778796561673
  (1, 4)        0.2778778796561673
  (1, 5)        0.2778778796561673
  (1, 1)        0.5557557593123346
El perro pasea feliz por el bosque [0.33333333 0.66666667 0.33333333 0.         0.33333333 0.33333333
 0.33333333 0.        ]
El perro ya no pasea feliz por el bosque [0.27787788 0.55575576 0.27787788 0.39054766 0.27787788 0.27787788
 0.27787788 0.39054766]
```

Figura 5.6: Ejemplo de TF-IDF con Sklearn

5.4.2. Modelos de clasificación

Tras representar los datos de forma numérica, queda crear modelos que aprendan a relacionarlos entre sí para poder clasificarlos correctamente. De acuerdo al estado del arte y como se cita en los objetivos, los 3 algoritmos seleccionados para cumplir con este fin son *SVC*, *Naïve Bayes Multinomial* y *Pasivo Agresivo*.

SVC

Se trata de las *Support Vector Classifiers*, un subtipo de *SVM* (*Support Vector Machines*). Tal y como se indica en [8], se trata de un modelo que representa los puntos de muestra en el espacio, separando las clases en espacios lo más amplios posibles empleando un hiperplano. La imagen 5.7 representa gráficamente este concepto. Dentro de la librería

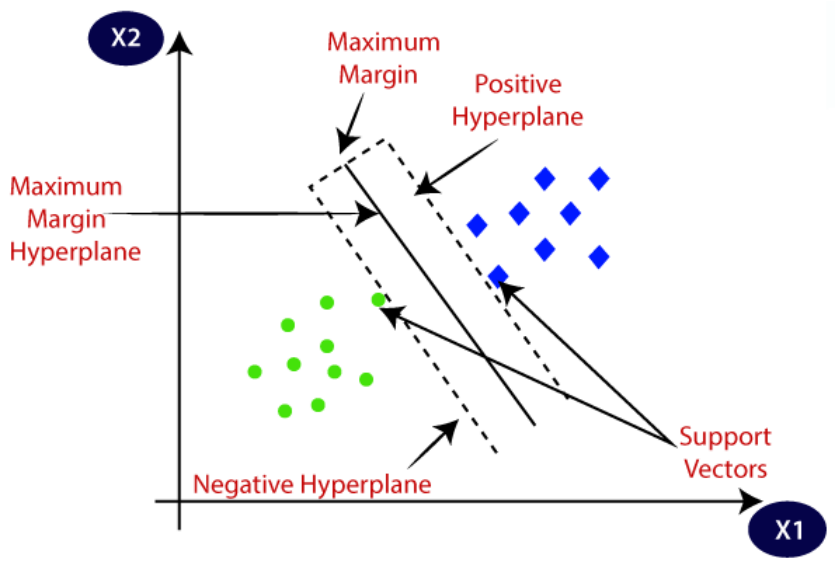


Figura 5.7: Ejemplo gráfico de la separación de dos clases en el espacio mediante un hiperplano. Ejemplo obtenido de `https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm`

Sklearn, la clase encargada de implementarlo es *SVC*.

Multinomial NB

Se trata de una forma modificada del clasificador *Naïve Bayes*, la cual también posee un enfoque probabilístico similar a este [41]. Está especialmente diseñada para tratar con documentos de texto y para calcular la ocurrencia de cada palabra [41]. Mientras que *Naïve Bayes* funciona basándose en la probabilidad condicional, *Multinomial NB* lo hace basándose en la distribución multinomial [41]. Se cálcula la probabilidad individual de que un documento pertenezca a cada una de las clases, y se le asigna aquella cuya mayor probabilidad posea [36]. Calcula las hipótesis de que los términos presentes en un documento ya se encuentren en otros pertenecientes a una clase concreta [36]. El modelo multinomial depende de la evaluación de la función de decisión, la cual se concluye a partir del teorema de Bayes [36]. La probabilidad bayesiana se calcula según la ecuación 5.2. Dentro de la librería *Sklearn*, la clase encargada de implementarlo es *MultinomialNB*.

Siendo W el vector con los términos y C el vector con las clases:

$$p(C_k|W) = \frac{p(C_k) * p(W|C_k)}{p(W)}, \tag{5.2}$$
$$p(W|C_k) = \prod_{i=1}^{n} p(w_i|C_k)$$

Passive Aggressive (Pasivo agresivo)

Es un algoritmo de aprendizaje en línea; esto significa que puede actualizar sus parámetros conforme recibe nuevos datos [6]. Es un paradigma de aprendizaje consistente en entrenar un modelo introduciéndole datos de forma secuencial; de esta manera, los modelos de aprendizaje en línea siguen aprendiendo aún cuando ya están desplegados en producción. De acuerdo a [26], es una modificación del algoritmo Perceptrón estándar, donde se hace uso de un único parámetro, C. Este se encarga de encontrar un equilibrio entre el tamaño del margen y el número de instancias mal clasificadas. Si una instancia es clasificada correctamente, no se efectua ningún cambio en el modelo; de lo contrario, se ajustan los pesos [31]. El grado en el que se produce este ajuste depende del parámetro C [26]. Dentro de la librería *Sklearn*, la clase encargada de implementarlo es *PassiveAggressiveClassifier*.

5.4.3. Selección de los mejores modelos

Para seleccionar qué modelos han sido los óptimos se ha hecho uso de la clase *GridSearchCV* de *Sklearn*. Con este uso se han cumplido dos objetivos principales: en primer lugar, determinar qué combinaciones de hiperparámetros son las que mejor funcionan para cada modelo; en segundo lugar, mediante validación cruzada con $K = 5$, determinar qué precisión arroja esa combinación. Debido a limitaciones técnicas, no se han probado todas las combinaciones posibles, por lo que los resultados obtenidos se han ajustado a las posibilidades materiales y temporales. Para estudiar los modelos se ha utilizado la métrica *f1_macro* en todo momento, pues es la más precisa en relación a su complejidad computacional. Esto se debe a que la métrica F1 se calcula llevando a cabo la media armónica entre la exhaustividad (*recall*) y la precisión (*precision*), por lo que se combinan ambas métricas en un solo valor. La precisión mide cuántas clasificaciones tienen realmente la etiqueta que les ha sido asignada respecto al número de instancias que se han declarado con dicha etiqueta; la exhaustividad mide cuántas clasificaciones tienen realmente la etiqueta que les ha sido asignada respecto al número de instancias que realmente la poseen.

Combinar ambas métricas es relevante debido a que de esta manera se tienen pocos falsos positivos y pocos falsos negativos, obteniendo una perspectiva más realista del funcionamiento del modelo. Se ha empleado la variante *macro* pues se trata de un cálculo muy sencillo, no ponderado y porque trata a todas las clases por igual; además, no se tienen en cuenta falsos positivos o negativos [27].

$$F1 = 2 * \frac{recall * precision}{recall + precision} \tag{5.3}$$

Cabe destacar que se han utilizado ambos modelos de representación de datos con unigramas y bigramas (es decir, se han tomado en cuenta las palabras de forma individual y cada secuencias de dos palabras seguidas que ofrece cada documento). Se han eliminado todas las *stop words* en castellano.

En primer lugar, se han leido los comentarios y se han preprocesado, tal y como se puede ver en la imagen 5.8. En segundo lugar, se han seleccionado los hiperparámetros a

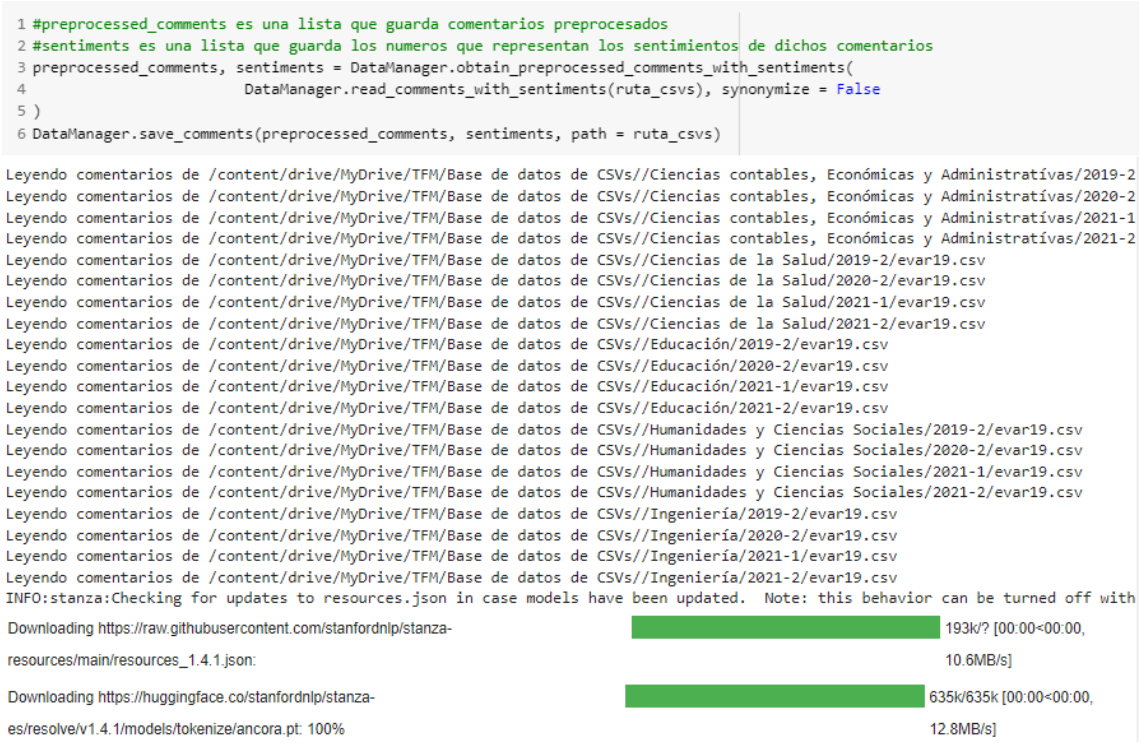


Figura 5.8: Ejemplo de obtención de los comentarios preprocesados junto a sus sentimientos

calibrar en cada modelo de clasificación junto a sus posibles valores. En cada algoritmo han sido los siguientes:

- *SVC*
  - C: Parámetro de penalización del término de error. Se dedica a compensar las clasificaciones correctas y la maximización de la función de decisión. Sus posibles valores han sido 1, 10 y 100.
  - *kernel*: Función que proyecta los datos a un nuevo espacio de características con mayor dimensionalidad. Su único valor ha sido *linear*, pues es el kernel que mejor funciona para problemas de clasificación de texto.
- *Multinomial NB*:
  - *alpha*: Parámetro de suavizado aditivo; calibra el valor de los parámetros para que no haya *overfitting* o *underfitting*. Sus posibles valores han sido 1, 0.1, 0.01, 0.001, 0.0001 y 0.00001.
  - *fit_prior*: Parámetro para establecer si se aprenden las probabilidades a priori de la clase. Sus valores han sido *True* y *False*.
- *Passive Aggresive*:
  - *warm_start*: Establece si se reutiliza la solución de la llamada anterior en la inicialización de la siguiente. Su único valor ha sido *True*.
  - C: Máximo tamaño del paso. Sus posibles valores han sido 0.003, 0.01, 0.03 y 0.1.
  - *loss*: Función de pérdida a emplear. Sus posibles valores han sido *hinge* y *squared_hinge*.

En la imagen 5.9 se puede apreciar un ejemplo del entrenamiento y la creación de un modelo a partir de estos hiperparámetros y los comentarios preprocesados.

En tercer lugar, se han guardado los modelos en formato *.joblib* conforme estos eran obtenidos, tal y como se observa en la última línea de código en 5.9.

En último lugar, para determinar de forma definitiva qué modelo es el que mejor funciona, se ha optado por realizar una validación cruzada invocando a la combinación de hiperparámetros que mejores resultados han arrojado en la fase anterior, tal y como se puede ver en 5.10. Según se puede contemplar en la tabla 5.1, la cual detalla el porcentaje de la métrica *F1_macro*, el modelo que mejor funciona es *Passive Aggressive* con $C = 0.03$, $loss = hinge$ y $warm\_start = True$ en combinación con *TF-IDF*; no obstante, cabe

```
repres_model_tfidf = TfidfVectorizer(ngram_range=(1,2)) #Tfidf
param_grid = {'C': [1, 10, 100],
              'kernel': ['linear']}
grid_classif_model = GridSearchCV(svm.SVC(), param_grid, refit = True, verbose = 3, scoring = "f1_macro
sentim_model = SentimentManager(repres_model_tfidf, grid_classif_model)
```

```
sentim_model.train_with_comments(preprocessed_comments, sentiments)
sentim_model.save_model(path=ruta_modelos, filename="BEST_F1_SVC_GRID_SEARCH_CV_TFIDF")
```

```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
[CV 1/5] END ................C=1, kernel=linear;, score=0.892 total time= 8.8min
[CV 2/5] END ................C=1, kernel=linear;, score=0.936 total time= 8.8min
[CV 3/5] END ................C=1, kernel=linear;, score=0.900 total time= 8.3min
[CV 4/5] END ................C=1, kernel=linear;, score=0.898 total time= 8.5min
[CV 5/5] END ................C=1, kernel=linear;, score=0.914 total time= 8.8min
[CV 1/5] END ...............C=10, kernel=linear;, score=0.880 total time= 9.8min
[CV 2/5] END ...............C=10, kernel=linear;, score=0.928 total time=10.3min
```

Figura 5.9: Ejemplo de la obtención del modelo SVC con TF-IDF

```
svc_K_fold = SentimentManager(TfidfVectorizer(ngram_range=(1,2)),
                              svm.SVC(C = 1,
                              kernel = "linear"))
X = svc_K_fold.data_representation_model.fit_transform(preprocessed_comments).toarray()
cv = KFold(n_splits=5, random_state = None, shuffle = True)
print("Accuracies with " + str(cv))
scores = cross_val_score(svc_K_fold.data_classification_model, X, sentiments, cv=cv, scoring = "f1_macro")
print("%0.4f accuracy with a standard deviation of %0.4f" % (scores.mean(), scores.std()))
```

```
Accuracies with KFold(n_splits=5, random_state=None, shuffle=True)
0.9142 accuracy with a standard deviation of 0.0073
```

Figura 5.10: Ejemplo de cross validation con los mejores hiperparámetros, en este caso, con SVC

recalcar que los resultados han sido parejos en todos los algoritmos de clasificación cuando han empleado *TF-IDF*, por lo que cualquiera podría ser utilizado.

	SVC	MultinomialNB	Passive Aggresive
BoW	88.54 %	90.41 %	89.38 %
TF-IDF	91.42 %	90.94 %	91.59 %

Tabla 5.1: Resultados (*f1_macro*) proporcionados por la combinación de los algoritmos de extracción de características (filas) y los de clasificación (columnas)

Finalmente, se ha probado el mejor modelo con comentarios escritos a mano, guardando las predicciones en un archivo *.json*. Este será el formato en el que será enviado a la interfaz gráfica. En la imagen 5.11 se puede apreciar como en un par de líneas de código el modelo de minería de opinión es capaz de preprocesar los comentarios, predecir los sentimientos que denotan y guardar los resultados. El resultado de las predicciones puede ser observado

```
"Tiene que mejorar",
"Puede mejorar, pero es bueno",
"es bueno",
"Excelente profesor",
"Gran docente!",
"No me gusto como dio clase",
"El peor docente",
"El peor maestro",
"El mejor docente",
"El mejor maestro",
"El peor docente, pero tiene mucho que mejorar",
"El peor profesor, pero tiene mucho que mejorar",
"El mejor docente, pero tiene mucho que mejorar",
"El mejor profesor, pero tiene mucho que mejorar",
"muchas gracias por todo profe!",
"un profe muy impuntual!",
"malo",
"bueno",
"poco bueno",
"nada malo",
"muy malo",
"muy bueno",
"espero no volver a verle",
"espero volver a verle",
"calidad baja",
"baja calidad",
"tiene muy baja calidad para enseñar",
"cambiadlo",
"cambienlo",
"lo tienen que cambiar",
"no lo cambieis",
"no lo cambien",
"mantenedlo",
"mantenganlo"
]
best_model = SentimentManager.load_model("/content/drive/MyDrive/TFM/Modelos/BEST_F1_PASSIVE_AGGRESIVE_GRID_SEARCH_CV_TFIDF_V2.joblib")
best_model.predict_comments_and_save(handmade_comments)
```

Figura 5.11: Predicción de comentarios escritos a mano empleando el algoritmo SVC

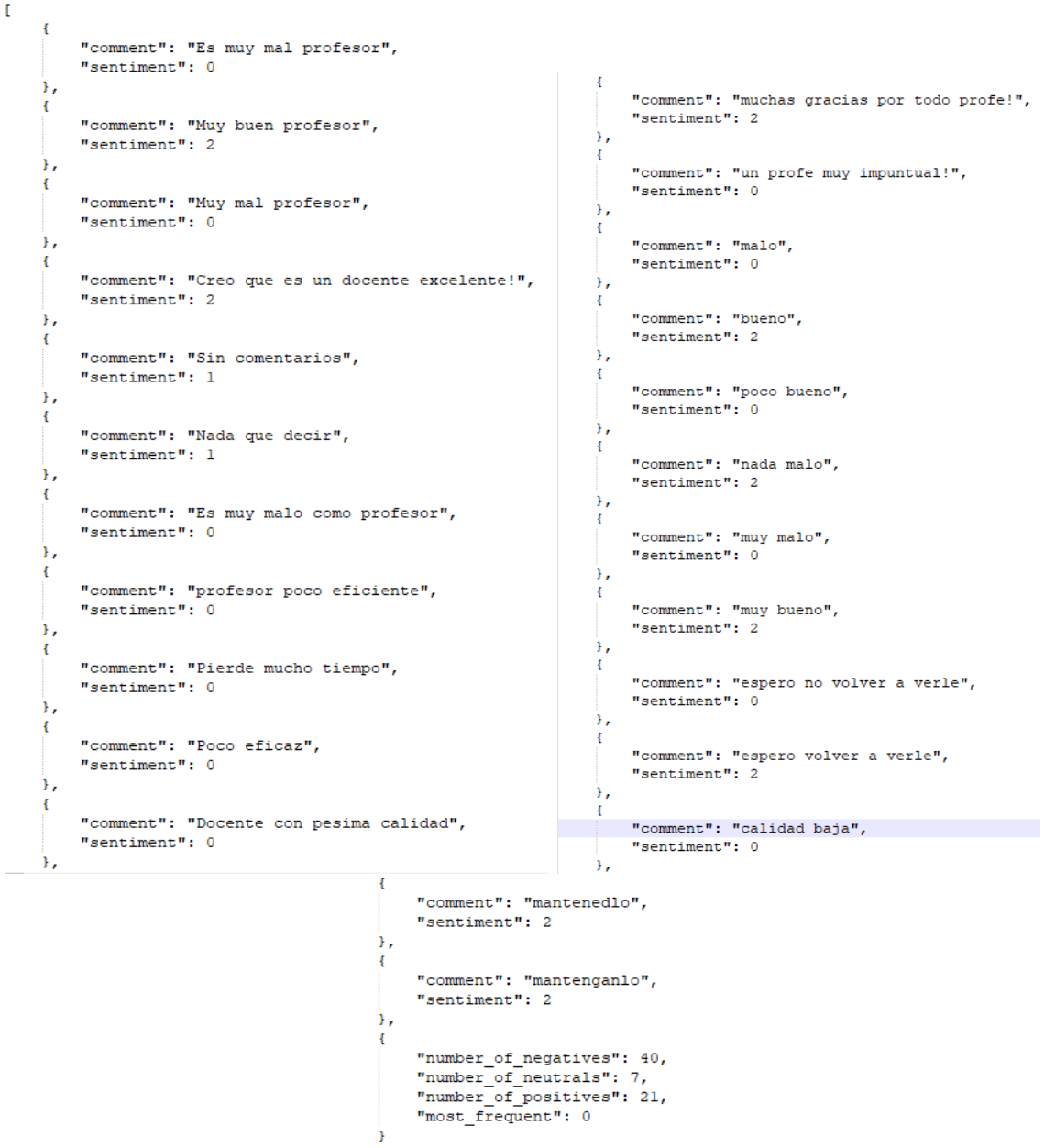
en las imágenes 5.12.

5.5. Despliegue del modelo

Para desplegar el modelo, se ha empleado el framework *Flask*. El procedimiento ha sido muy sencillo, y puede ser estudiado en la imagen 5.13.

En primer lugar, se declaran las configuraciones necesarias para el funcionamiento de *Flask*. Posteriormente, se inicializa el modelo leyéndolo del archivo *.joblib* que lo contiene. Se ha empleado *Passive Aggresive*, pues es el que mejor resultados presenta.

Como se puede ver, al llamar al endpoint */predict* se busca el archivo *.csv* que contiene los comentarios. Estos han de venir únicamente con *Comentario* como única cabecera. Una vez leído, se guarda en un directorio temporal de nombre *input*, desde donde lo leerá el modelo mediante su función *read_comments_without_sentiments*. Tras realizar la lectura de los comentarios, se procesan en *obtain_predictions*, la cual devuelve una lista que contiene cada comentario con su sentimiento adjunto así como el número de comentarios para cada sentimiento y el más frecuente. Una vez obtenidos los resultados, elimina el archivo. En último lugar, devuelve las predicciones en formato *.json*, lo cual le resultará de utilidad a



```
[
    {
        "comment": "Es muy mal profesor",
        "sentiment": 0
    },
    {
        "comment": "Muy buen profesor",
        "sentiment": 2
    },
    {
        "comment": "Muy mal profesor",
        "sentiment": 0
    },
    {
        "comment": "Creo que es un docente excelente!",
        "sentiment": 2
    },
    {
        "comment": "Sin comentarios",
        "sentiment": 1
    },
    {
        "comment": "Nada que decir",
        "sentiment": 1
    },
    {
        "comment": "Es muy malo como profesor",
        "sentiment": 0
    },
    {
        "comment": "profesor poco eficiente",
        "sentiment": 0
    },
    {
        "comment": "Pierde mucho tiempo",
        "sentiment": 0
    },
    {
        "comment": "Poco eficaz",
        "sentiment": 0
    },
    {
        "comment": "Docente con pesima calidad",
        "sentiment": 0
    },
    {
        "comment": "muchas gracias por todo profe!",
        "sentiment": 2
    },
    {
        "comment": "un profe muy impuntual!",
        "sentiment": 0
    },
    {
        "comment": "malo",
        "sentiment": 0
    },
    {
        "comment": "bueno",
        "sentiment": 2
    },
    {
        "comment": "poco bueno",
        "sentiment": 0
    },
    {
        "comment": "nada malo",
        "sentiment": 2
    },
    {
        "comment": "muy malo",
        "sentiment": 0
    },
    {
        "comment": "muy bueno",
        "sentiment": 2
    },
    {
        "comment": "espero no volver a verle",
        "sentiment": 0
    },
    {
        "comment": "espero volver a verle",
        "sentiment": 2
    },
    {
        "comment": "calidad baja",
        "sentiment": 0
    },
    {
        "comment": "mantenedlo",
        "sentiment": 2
    },
    {
        "comment": "mantenganlo",
        "sentiment": 2
    },
    {
        "number_of_negatives": 40,
        "number_of_neutrals": 7,
        "number_of_positives": 21,
        "most_frequent": 0
    }
```


Figura 5.12: Resultado de la predicción en un fichero json

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
from Sentiment_evaluator import SentimentManager
from Sentiment_evaluator import DataManager
from werkzeug.utils import secure_filename
import os

app = Flask(__name__)
CORS(app)

PORT = 8000
DEBUG = True
ALLOWED_EXTENSIONS = set(['csv'])

model = SentimentManager.load_model("./models/BEST_F1_PASSIVE_AGGRESIVE_GRID_SEARCH_CV_TFIDF_V2.joblib")

def allowed_file(filename):
    return '.' in filename and \
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


@app.errorhandler(404)
def not_found(error):
    return "Not found!"


@app.route('/predict', methods = ['POST'])
def process():
    file = request.files['file']
    print(file.filename)
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = os.path.join('input', filename)
        file.save(filepath)
        predictions = predict(filepath)
        os.remove(filepath)
        return jsonify(predictions)
    return ""


def predict(filepath):
    comments = DataManager.read_comments_without_sentiments(filepath)
    return model.obtain_predictions(comments)


if __name__ == '__main__':
    app.run(port = PORT, debug = DEBUG)
```

Figura 5.13: Código empleado para desplegar el modelo con *Flask*

la interfaz gráfica para poder leer los datos de forma sencilla.

Se ha ejecutado en local debido a restricciones temporales; no obstante, su sencillo diseño permite desplegarla en un entorno web con gran facilidad.

5.6. Interfaz gráfica

La interfaz gráfica consiste en una sencilla página *HTML* donde el usuario ha de introducir un archivo de extensión *.csv* que contenga los comentarios a procesar y con *Comentario* como única cabecera, como se muestra en la imagen 5.14.

Una vez posee dicho documento, simplemente tiene que hacer click en *Seleccionar archivo* (figura 5.15), seleccionarlo (figura 5.16), y hacer click en el botón *predecir* (figura 5.17).

Tras hacer click en el botón se indicará que los comentarios han sido enviados (imagen

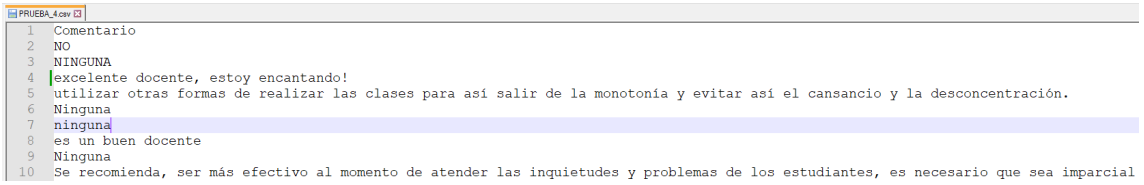


Figura 5.14: Contenido del archivo *.csv* con los comentarios

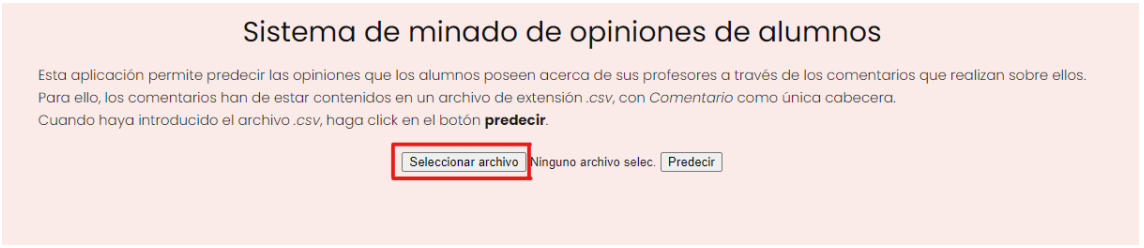


Figura 5.15: Botón de selección de archivo

5.18), aparecerá una anotación indicando que están siendo procesados y desaparecerá el botón de predecir, de la manera en la que se puede ver en 5.19.

Una vez se lleva a cabo el procesamiento por parte del modelo tal y como se ha visto en la sección 5.5, se reciben dos tablas: la primera contiene los comentarios con los sentimientos que les han sido asignados; la segunda posee dichos comentarios agrupados por sentimientos. Finalmente, se muestran cuántos comentarios son negativos, positivos y neutros así como el sentimiento más frecuente entre todos. Además, el botón de predecir vuelve a aparecer. Las imágenes 5.20 y 5.21 muestran extractos de la predicción de manera gráfica.

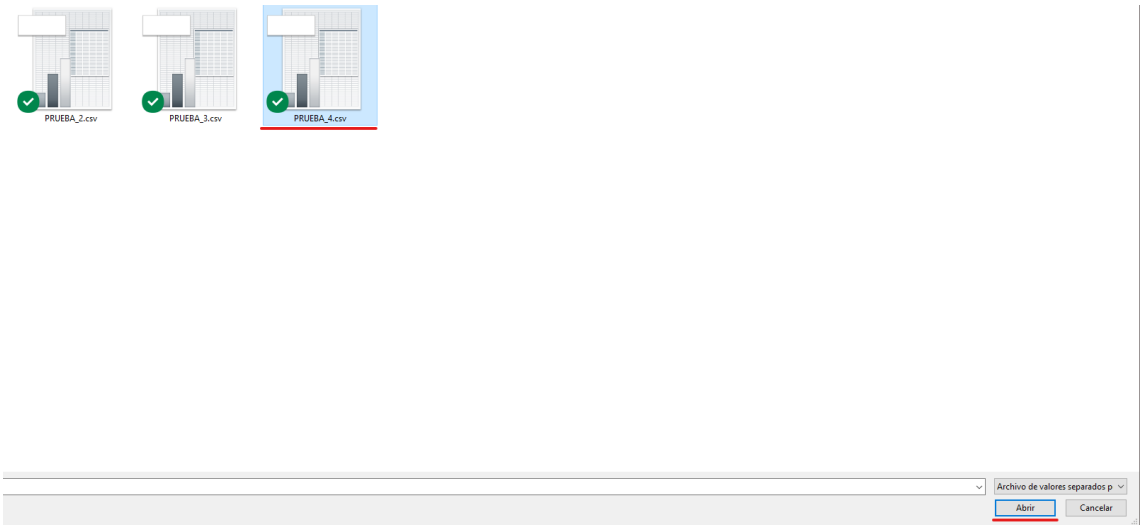


Figura 5.16: Selección del archivo

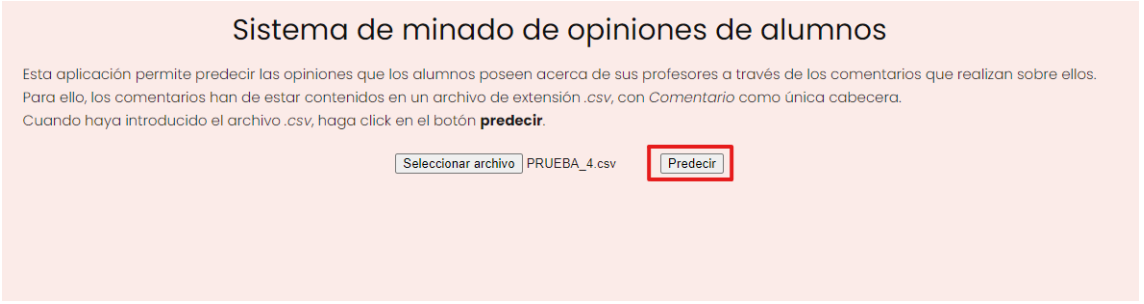


Figura 5.17: Botón de predecir

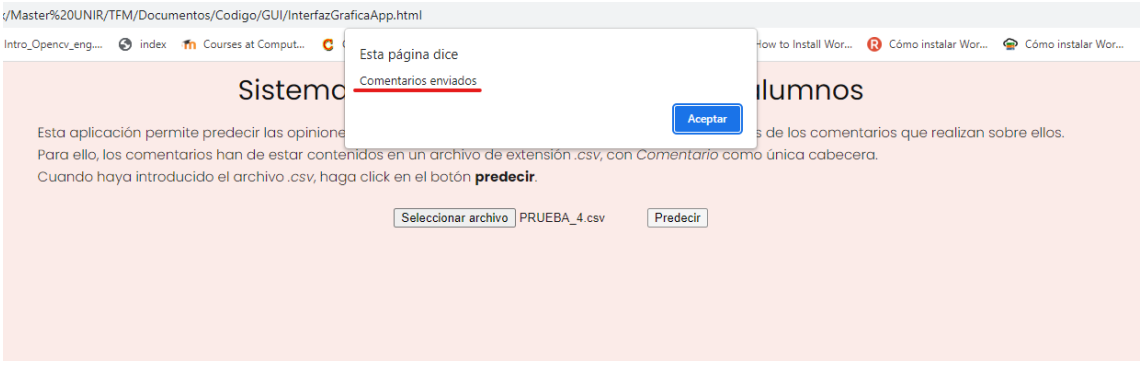


Figura 5.18: Indicación del envío de los comentarios

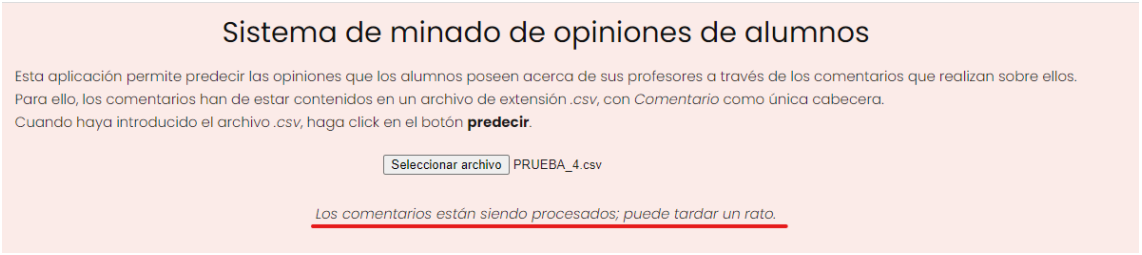


Figura 5.19: Indicación del procesamiento de los comentarios

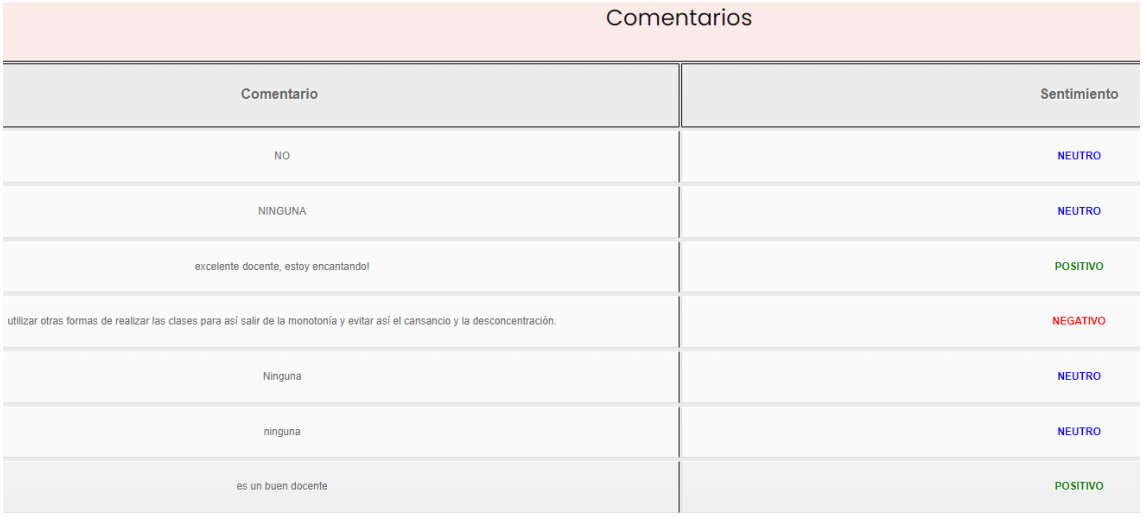


Figura 5.20: Extracto de los resultados de la predicción de manera gráfica: comentarios con sus sentimientos adjuntos

Figura 5.21: Extracto de los resultados de la predicción de manera gráfica: comentarios agrupados por sus sentimientos

Capítulo 6

Evaluación del software

6.1. Evaluación de los modelos de Machine Learning

Para evaluar el funcionamiento de los diferentes modelos de *Machine Learning* se ha optado por obtener las curvas ROC que proporcionan cada combinación de algoritmos mencionados, además de entrenar y testear un modelo desde 0. La curva ROC se utiliza para evaluar cómo funciona un modelo de Machine Learning a la hora de enfrentar problemas de clasificación binaria, y consiste en un trazado de la tasa de verdaderos positivos contra la tasa de falsos positivos. Si bien este no es un problema de clasificación binaria, se han trazado las curvas ROC para cada clase (0, 1 y 2), de forma que se evalua cómo funciona cada combinación de algoritmos en cada una.

En primer lugar, se instancian uno de los modelos haciendo uso de los hiperparámetros que mejores resultados han arrojado para su algoritmo correspondiente (obtenidos mediante *GridSearchCV* 5.9), de la misma forma en la que se hace en 5.10. Acto seguido, se transforman los comentarios pre-procesados en datos numéricos. Posteriormente, se divide este conjunto de datos en 2 partes: un conjunto de entrenamiento (80 % del total) y conjunto de test (20 % restante). Los datos se entremezclan para evitar *overfitting* y para reducir la varianza. La idea es que estén lo menos relacionados posibles, lo cual puede llegar a darse en casos donde hay muchos positivos/negativos seguidos, como por ejemplo, cuando se evalua a un mismo profesor y los alumnos poseen opiniones parecidas. Después, se entrena el modelo con el conjunto de datos de entrenamiento, para luego obtener los resultados. Estos resultados no son las predicciones como tal, si no las probabilidades adjuntas a cada clase en el caso de *SVC* y *Multinomial NB* o puntuaciones de confianza para *Passive Aggressive*, las cuales se refieren al grado de confianza que posee el modelo para

```
svc_model = SentimentManager(TfidfVectorizer(ngram_range=(1,2)),
                             svm.SVC(C = 1,
                             kernel = "linear", probability = True))
X = svc_model.transform_comments_into_count_array(preprocessed_comments)
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, sentiments, test_size = 0.2, random_state=None)
svc_model.train(X_train, y_train)

probabs = svc_model.predict_proba(X_test)
y_true = np.array(y_test)
y_probabs = copy.copy(probabs)
skplt.metrics.plot_roc_curve(y_true, y_probabs)
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_roc_curve is deprecat
  warnings.warn(msg, category=FutureWarning)
```

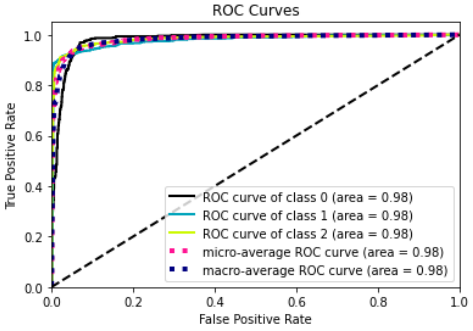


Figura 6.1: Ejemplo de la obtención de las curvas ROC para SVC con TF-IDF

cada clase de que esta sea la verdadera.

Tras obtener los resultados, se logra la gráfica con las curvas ROC para cada clase haciendo uso de la función *plot_roc_curve* de la librería *scikitplot*. La imagen 6.1 muestra el código que ha hecho posible la obtención de estos resultados, más concretamente para SVC con TF-IDF.

Las imágenes 6.26.36.4 muestran las soluciones conseguidas, y tal y como se puede advertir, todas las combinaciones superan el 90 % en todos los indicadores, evidenciando que los modelos funcionan positivamente. Para complementar esta evaluación, se ha optado también por entrenar un modelo y testearlo desde 0. La imagen 6.5 expone cómo se inicializa un modelo con los hiperparámetros óptimos, se entrena con un 80 % de los datos y se testea con el 20 % restante; el resultado, como se observa, es que se predicen correctamente el 91 % de los comentarios, en sintonía con las conclusiones extraidas de las curvas ROC.

6.2. Evaluación de la aplicación web

Para medir la funcionalidad y usabilidad de la aplicación web se ha calculado el índice *SUS* de la misma rellenando el formulario proporcionado por *SUS Calculator*. Se ha

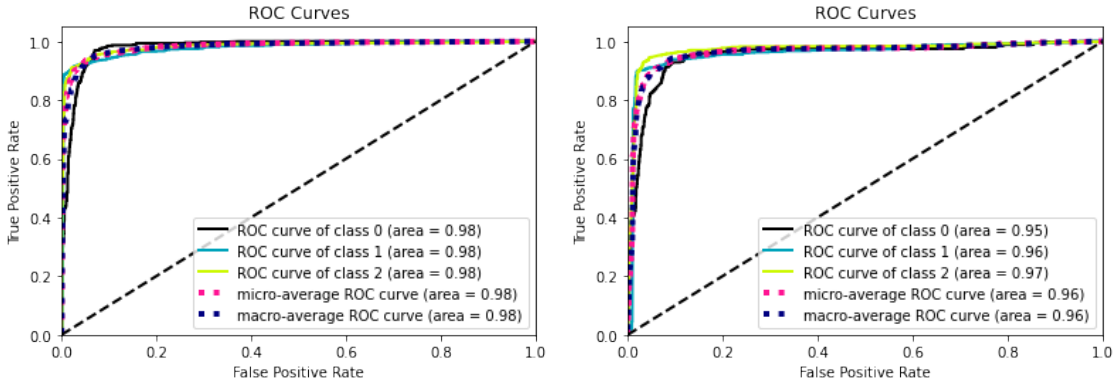


Figura 6.2: Curvas ROC de SVC para TF-IDF (izquierda) y Bag of Words (derecha)

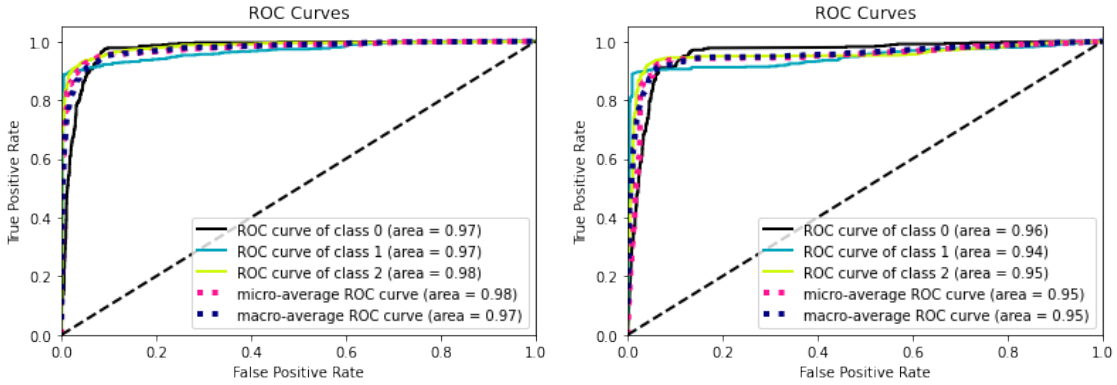


Figura 6.3: Curvas ROC de Naïve Bayes Multinomial para TF-IDF (izquierda) y Bag of Words (derecha)

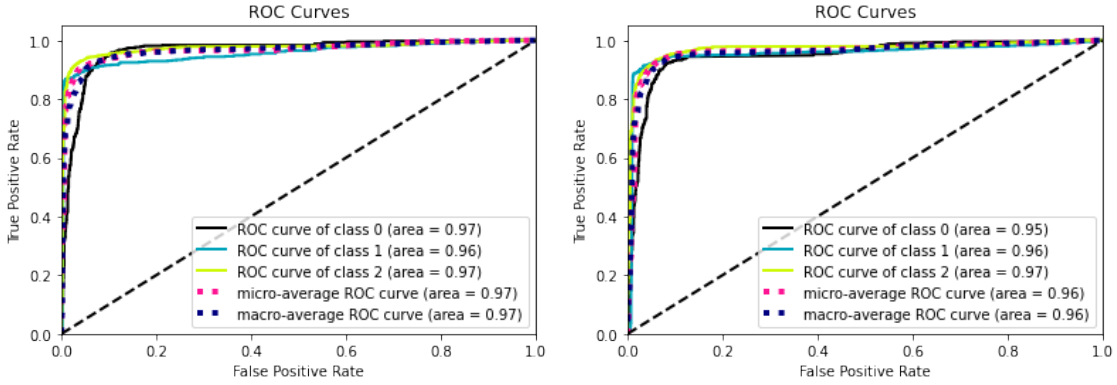


Figura 6.4: Curvas ROC de Pasivo Agresivo para TF-IDF (izquierda) y Bag of Words (derecha)

```
passive_agressive_model = SentimentManager(TfidfVectorizer(ngram_range=(1,2)),
                                           PassiveAggressiveClassifier(C=0.03,
                                           loss="hinge", warm_start= True))
X = passive_agressive_model.transform_comments_into_count_array(preprocessed_comments)
X_train, X_test, y_train, y_test = model_selection.train_test_split(
    X, sentiments, test_size = 0.2, random_state=None, shuffle = True)
passive_agressive_model.train(X_train, y_train)

correct_negative  = 0
correct_neutral  = 0
correct_positive  = 0
for i in zip(y_test, passive_agressive_model.predict(X_test)):
    if i[0] == i[1]:
        if i[0] == 0:
            correct_negative += 1
        elif i[0] == 1:
            correct_neutral += 1
        elif i[0] == 2:
            correct_positive += 1
correct = correct_negative + correct_neutral + correct_positive
print(f"Correct {correct} | Total {len(y_test)} | {correct/len(y_test)*100} %")
print(f"Correct negative: {correct_negative} / {y_test.count(0)} | {correct_negative/y_test.count(0)*100} %")
print(f"Correct neutral: {correct_neutral} / {y_test.count(1)}  | {correct_neutral/y_test.count(1)*100} %")
print(f"Correct positive: {correct_positive} / {y_test.count(2)}  | {correct_positive/y_test.count(2)*100} %")
```

```
Correct 1614 | Total 1760 | 91.70454545454545 %
Correct negative: 432 / 455 | 94.94505494505493 %
Correct neutral: 585 / 656  | 89.17682926829268 %
Correct positive: 597 / 649  | 91.98767334360555 %
```

Figura 6.5: Ejemplo de entrenamiento y testeo con el algoritmo pasivo agresivo

contado con la opinión de 5 usuarios que han probado presencialmente la aplicación para llevar a cabo este cálculo. Las preguntas y sus respuestas correspondientes han sido las siguientes:

1. Creo que me gustaría usar este sitio web con frecuencia:
    - Primer usuario: De acuerdo.
    - Segundo usuario: Muy de acuerdo.
    - Tercer usuario: Neutro.
    - Cuarto usuario: De acuerdo.
    - Quinto usuario: Neutro.
2. Encontré el sitio web innecesariamente complejo:
    - Primer usuario: Desacuerdo.
    - Segundo usuario: Desacuerdo.
    - Tercer usuario: Muy en desacuerdo.
    - Cuarto usuario: Muy desacuerdo.

- Quinto usuario: Desacuerdo.

3. Pensé que el sitio web era fácil de usar:
   - Primer usuario: Muy de acuerdo.
   - Segundo usuario: De acuerdo.
   - Tercer usuario: De acuerdo.
   - Cuarto usuario: Muy de acuerdo.
   - Quinto usuario: Muy de acuerdo.
4. Creo que necesitaría el apoyo de un técnico para poder utilizar este sitio web:
   - Primer usuario: Neutro.
   - Segundo usuario: Desacuerdo.
   - Tercer usuario: Neutro.
   - Cuarto usuario: Desacuerdo.
   - Quinto usuario: De acuerdo.
5. Encontré que las diversas funciones de este sitio web estaban bien integradas:
   - Primer usuario: De acuerdo.
   - Segundo usuario: Neutro.
   - Tercer usuario: De acuerdo.
   - Cuarto usuario: De acuerdo.
   - Quinto usuario: De acuerdo.
6. Pensé que había demasiada inconsistencia en este sitio web:
   - Primer usuario: Muy en desacuerdo.
   - Segundo usuario: Desacuerdo.
   - Tercer usuario: Neutro.
   - Cuarto usuario: Desacuerdo.
   - Quinto usuario: Muy en desacuerdo.
7. Me imagino que la mayoría de la gente aprendería a usar este sitio web muy rápidamente:

- Primer usuario: Muy de acuerdo.
- Segundo usuario: Muy de acuerdo.
- Tercer usuario: De acuerdo.
- Cuarto usuario: De acuerdo.
- Quinto usuario: Muy de acuerdo.

8. Encontré el sitio web muy engorroso de usar:
   - Primer usuario: Desacuerdo.
   - Segundo usuario: Muy en desacuerdo.
   - Tercer usuario: Neutro.
   - Cuarto usuario: Desacuerdo.
   - Quinto usuario: Desacuerdo.
9. Me sentí muy confiado al usar el sitio web:
   - Primer usuario: De acuerdo.
   - Segundo usuario: De acuerdo.
   - Tercer usuario: Muy de acuerdo.
   - Cuarto usuario: De acuerdo.
   - Quinto usuario: De acuerdo.
10. Necesitaba aprender muchas cosas antes de poder utilizar este sistema:
    - Primer usuario: Desacuerdo.
    - Segundo usuario: Muy en desacuerdo.
    - Tercer usuario: Muy en desacuerdo.
    - Cuarto usuario: Desacuerdo.
    - Quinto usuario: Desacuerdo.

Se han obtenido las siguientes puntuaciones:

- Primer usuario: 80.
- Segundo usuario: 82.5.

- Tercer usuario: 72.5.
- Cuarto usuario: 80.
- Quinto usuario: 75.

Se ha obtenido una nota media de 78, por lo que la aplicación web cumple con los requerimientos de usabilidad y funcionalidad requeridos.

Para evaluar su funcionamiento como usuario, se ha optado por predecir los siguientes comentarios:

- Es una profesora fantástica.
- No es bueno.
- Es un mal profesor.
- Es un buen profesor.
- Horrible como maestro.
- Nada que decir.
- Enseña muy bien.
- Muy contenta con la enseñanza.
- Debería tener más en cuenta la opinión de los alumnos.
- El mejor maestro que he tenido.
- No me gustó como impartió clase.
- Me gustó como impartió clase.
- Debería mejorar sus métodos.
- Ojalá volver a tenerle como docente.
- Sin comentarios.
- Es deficiente como profesor.
- Espero volver a verle.

- Espero no volver a verle.
- Es bueno como profesor, pero sus clases tienen que ser más amenas.
- Debería vocalizar mejor, no obstante, es bueno.
- Sus clases son aburridas.

Para ello, se ha creado un archivo *.csv* conteniendo dichos comentarios y se ha evaluado mediante el modelo desplegado, arrojando las predicciones visibles en las imágenes 6.6 y 6.7.

Los resultados van en línea con lo visto a lo largo de este capítulo, y evidencian que el software funciona correctamente y conforme a lo establecido en los objetivos. Las predicciones han resultado ser correctas aún cuando existen frases con un gran parecido entre sí donde la diferencia entre los sentimientos que designan reside en matices sutiles.

Comentarios

Comentario	Sentimiento
Es una profesora fantástica	POSITIVO
No es bueno	NEGATIVO
Es un mal profesor	NEGATIVO
Es un buen profesor	POSITIVO
Horrible como maestro	NEGATIVO
Nada que decir	NEUTRO
Enseña muy bien	POSITIVO
Muy contenta con la enseñanza	POSITIVO
Debería tener más en cuenta la opinión de los alumnos	NEGATIVO
El mejor maestro que he tenido	POSITIVO

Comentarios

No me gustó como impartió clase	NEGATIVO
Me gustó como impartió clase	POSITIVO
Debería mejorar sus métodos	NEGATIVO
Ojalá volver a tenerle como docente	POSITIVO
Sin comentarios	NEUTRO
Es deficiente como profesor	NEGATIVO
Espero volver a verle	POSITIVO
Espero no volver a verle	NEGATIVO
Es bueno como profesor, pero sus clases tienen que ser más amenas	NEUTRO
Debería vocalizar mejor, no obstante, es bueno	NEUTRO
Sus clases son aburridas	NEGATIVO

Figura 6.6: Predicciones de los comentarios de test definidos en el capítulo 6

Figura 6.7: Predicciones agrupadas y resumen de los comentarios de test definidos en el capítulo 6

Capítulo 7

Conclusiones y Trabajo Futuro

En este capítulo se extraerán las conclusiones acerca del trabajo realizado, así como posibles aspectos a mejorar y líneas de trabajo a llevar a cabo en el futuro.

En primer lugar, cabe destacar que la aplicación web ha demostrado ser de utilidad en los casos estudiados, y cumple con los objetivos definidos. En segundo lugar, se puede afirmar que los algoritmos evaluados en el estado del arte son de gran utilidad en el ámbito de la minería de opinión educacional, y se reafirman como de gran interés para los investigadores que decidan ahondar en esta disciplina. El algoritmo pasivo agresivo ha demostrado ser mejor que SVC y Naïve Bayes Multinomial, sin embargo, la diferencia es de décimas, por lo que cualquiera de los tres algoritmos de clasificación podría haber sido empleado en este proyecto. En lo que se refiere a los algoritmos de extracción de características, TF-IDF ha demostrado ser mejor que Bag of Words, por lo que debería ser utilizado tanto en cualquier posible evolución futura de esta aplicación como en otras relacionadas a la minería de opinión educacional.

No obstante, hay diversos aspectos a mejorar pese a los buenos resultados obtenidos:

- La aplicación debería dar la posibilidad al usuario de elegir qué algoritmos quiere emplear.
- El modelo de aprendizaje automático debería ser capaz de aprender con cada nuevo conjunto de comentarios predichos.
- Se debería poder entrenar un nuevo modelo o uno ya existente mediante un *.csv*, no simplemente realizar predecciones.
- Se debería poder tener un registro histórico de las predicciones realizadas.

- Se debería rediseñar la interfaz gráfica con un nuevo diseño para que sea más atractiva y eficiente visualmente (por ejemplo, teniendo tablas cuya anchura se ajuste de manera dinámica).
- Se deberían entrenar los modelos con más comentarios y con más diversidad léxica.
- Se debería contar con un número parejo de comentarios positivos, neutros y negativos.
- Se deberían explorar más algoritmos de extracción de características y de clasificación.
- Se debería llegar a un 95 % de precisión.
- Se deberían poder detectar más sentimientos.
- Se debería contar con datos que provengan de más instituciones educativas.
- Se debería poder realizar predicciones en más idiomas además del castellano.
- Se deberían poder predecir los sentimientos de comentarios escritos desde la propia interfaz gráfica.
- Se debería contar con una arquitectura e infraestructura de software más sofisticada.

Bibliografía

[1] Adinolfi, P., D'Avanzo, E., Lytras, M. D., Novo-Corti, I., and Picatoste, J. Sentiment analysis to evaluate teaching performance. *International Journal of Knowledge Society Research (IJKSR) 7*, 4 (2016), 86–107.

[2] Aung, K. Z., and Myo, N. N. Sentiment analysis of students' comment using lexicon based approach. In *2017 IEEE/ACIS 16th international conference on computer and information science (ICIS)* (2017), IEEE, pp. 149–154.

[3] Balahadia, F. F., Fernando, M. C. G., and Juanatas, I. C. Teacher's performance evaluation tool using opinion mining with sentiment analysis. In *2016 IEEE Region 10 Symposium (TENSYMP)* (2016), pp. 95–98.

[4] Binali, H. H., Wu, C., and Potdar, V. A new significant area: Emotion detection in e-learning using opinion mining techniques. In *2009 3rd IEEE international conference on digital ecosystems and technologies* (2009), IEEE, pp. 259–264.

[5] Cambria, E., Schuller, B., Xia, Y., and Havasi, C. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent systems 28*, 2 (2013), 15–21.

[6] Cesa-Bianchi, N., and Orabona, F. Online learning algorithms. *Annual review of statistics and its application* (2021).

[7] Charlie. How much data is created every day in 2022?

[8] de la Rioja, U. I. Tema 8. aprendizaje supervisado: clasificación con máquinas vector de soporte. *Universidad Internacional de la Rioja*.

[9] de la Rioja, U. I. Etiquetado morfosintáctico (pos tagging). *Universidad Internacional de la Rioja* (2017).

[10] Deng, L., and Liu, Y. A joint introduction to natural language processing and to deep learning. In *Deep learning in natural language processing*. Springer, 2018, pp. 1–22.

[11] Eisenstein, J. Natural language processing. *Jacob Eisenstein* (2018).

[12] Elyasir, A., and Anbananthen, K. Opinion mining framework in the education domain. *International Journal of Educational and Pedagogical Sciences 7*, 4 (2014), 1049–1054.

[13] Estrada, M. L. B., Cabada, R. Z., Bustillos, R. O., and Graff, M. Opinion mining and emotion recognition applied to learning environments. *Expert Systems with Applications 150* (2020), 113265.

[14] Feldman, R. Techniques and applications for sentiment analysis. *Communications of the ACM 56*, 4 (2013), 82–89.

[15] Flodén, J. The impact of student feedback on teaching in higher education. *Assessment & Evaluation in Higher Education 42*, 7 (2017), 1054–1068.

[16] Gamal, D., Alfonse, M., M. El-Horbaty, E.-S., and M. Salem, A.-B. Analysis of machine learning algorithms for opinion mining in different domains. *Machine Learning and Knowledge Extraction 1*, 1 (2018), 224–234.

[17] Garcia, E. M. Tema 3 -etiquetado morfosintáctico pos tagging (i). *Universidad Internacional de la Rioja* (2017).

[18] Gutierrez, Y., Tomas, D., and Fernandez, J. Benefits of using ranking skip-gram techniques for opinion mining approaches. In *eChallenges e-2015 Conference* (2015), pp. 1–10.

[19] Husnain, M., Missen, M. M. S., Akhtar, N., Coustaty, M., Mumtaz, S., and Prasath, V. A systematic study on the role of sentiwordnet in opinion mining. *Frontiers of Computer Science 15*, 4 (2021), 1–19.

[20] Jurek, A., Mulvenna, M. D., and Bi, Y. Improved lexicon-based sentiment analysis for social media analytics. *Security Informatics 4*, 1 (2015), 1–13.

[21] Kasthuriarachchy, B. H., De Zoysa, K., and Premaratne, H. Enhanced bag-of-words model for phrase-level sentiment analysis. In *2014 14th International Conference on Advances in ICT for Emerging Regions (ICTer)* (2014), pp. 210–214.

[22] Kastrati, Z., Dalipi, F., Imran, A. S., Pireva Nuci, K., and Wani, M. A. Sentiment analysis of students' feedback with nlp and deep learning: A systematic mapping study. *Applied Sciences 11*, 9 (2021), 3986.

[23] Keyvanpour, M., Karimi Zandian, Z., and Heidarypanah, M. Omlml: a helpful opinion mining method based on lexicon and machine learning in social networks. *Social Network Analysis and Mining 10*, 1 (2020), 1–17.

[24] Khairnar, J., and Kinikar, M. Machine learning algorithms for opinion mining and sentiment classification. *International Journal of Scientific and Research Publications 3*, 6 (2013), 1–6.

[25] Khurana, D., Koli, A., Khatter, K., and Singh, S. Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications* (2022), 1–32.

[26] Kumar, A. Passive aggressive classifier: Concepts amp; examples. `https://vitalflux.com/passive-aggressive-classifier-concepts-examples/#:~:text=The%20passive%20aggressive%20algorithm%20is,and%20the%20number%20of%20misclassifications.`, Oct 2022.

[27] Leung, K. Micro, macro weighted averages of f1 score, clearly explained. `https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f#2f35`, Jan 2022.

[28] Li, T., Liu, X., and Su, S. Semi-supervised text regression with conditional generative adversarial networks. In *2018 IEEE International Conference on Big Data (Big Data)* (2018), IEEE, pp. 5375–5377.

[29] Liu, B. *Sentiment analysis: Mining opinions, sentiments, and emotions.* Cambridge university press, 2020.

[30] Lok, E. J. Episode 1: Using tf-idf to identify the signal from the noise. "`https://mungingdata.wordpress.com/2017/11/25/episode-1-using-tf-idf-to-identify-the-signal-from-the-noise/`", Nov 2017.

[31] Mane, S. S., and Ali, S. M. Detection of forged news using passive aggressive classifier.

[32] Medhat, W., Hassan, A., and Korashy, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal 5*, 4 (2014), 1093–1113.

[33] Misuraca, M., Scepi, G., and Spano, M. Using opinion mining as an educational analytic: An integrated strategy for the analysis of students' feedback. *Studies in Educational Evaluation 68* (2021), 100979.

[34] Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. Natural language processing: an introduction. *Journal of the American Medical Informatics Association 18*, 5 (2011), 544–551.

[35] Qazi, A., Raj, R. G., Hardaker, G., and Standing, C. A systematic literature review on opinion types and sentiment analysis techniques: Tasks and challenges. *Internet Research* (2017).

[36] Ratz, A. V. Multinomial nave bayes' for documents classification and natural language processing (nlp). `https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6`, Apr 2022.

[37] Russell, J. A. Core affect and the psychological construction of emotion. *Psychological review 110*, 1 (2003), 145.

[38] Schmid, H. Part-of-speech tagging with neural networks. *arXiv preprint cmp-lg/9410018* (1994).

[39] Shirsat, V. S., Jagdale, R. S., and Deshmukh, S. Document level sentiment analysis from news articles. In *2017 international conference on computing, Communication, Control and Automation (ICCUBEA)* (2017), IEEE, pp. 1–4.

[40] Slater, S., Joksimović, S., Kovanovic, V., Baker, R. S., and Gasevic, D. Tools for educational data mining: A review. *Journal of Educational and Behavioral Statistics 42*, 1 (2017), 85–106.

[41] Surya, P. P., Seetha, L. V., and Subbulakshmi, B. Analysis of user emotions and opinion using multinomial naive bayes classifier. In *2019 3rd International con-*

ference on Electronics, Communication and Aerospace Technology (ICECA) (2019), IEEE, pp. 410–415.

[42] Vinodhini, G., and Chandrasekaran, R. Sentiment analysis and opinion mining: a survey. *International Journal 2*, 6 (2012), 282–292.

[43] Wen, M., Yang, D., and Rose, C. Sentiment analysis in mooc discussion forums: What does it tell us? In *Educational data mining 2014* (2014), Citeseer.

[44] Wróbel, K., Karwatowski, M., Wielgosz, M., Pietroń, M., and Wiatr, K. Compression of convolutional neural network for natural language processing. *Computer Science 21*, 1 (2020).

Apéndice A

Apéndices

A continuación se encuentra el artículo:

Creación de un clasificador de sentimientos de alumnos mediante minería de opinión

Ignacio Belzunegui Gabilondo

Universidad Internacional de la Rioja, Logroño (España)

06 de febrero de 2023

RESUMEN

Para un profesor, reconocer la opinión de un alumno sobre él puede ser bastante difícil. Cuando hay mucha información, leerla toda puede ser complicado, y se necesita una herramienta que permita extraer todos estos datos. En este trabajo se propone una aplicación que permite a los profesores conocer la opinión de sus alumnos.

Como parte de la operación de minería de opinión, los comentarios han pasado por varios procesos: se han corregido palabras mal escritas, se han convertido en sinónimos algunas relacionadas al ámbito educativo, y se han convertido en sus respectivos lemas.

Para representar los comentarios como datos numéricos se han analizado dos algoritmos de extracción de características: bolsa de palabras y TF-IDF. Por otro lado, para aprender a clasificar los datos, se han explorado tres algoritmos: SVC, Naïve Bayes multinomial y Pasivo Agresivo.

El modelo de aprendizaje automático en el que se basa el proyecto ha sido entrenado utilizando comentarios de la Universidad Mariana de Colombia, y alcanza más de un 90 % de precisión.

LA UNIVERSIDAD EN INTERNET

PALABRAS CLAVE

Bolsa de palabras, Clasificación, Comentarios, Extracción de características, Minería de opinión, Naïve Bayes Multinomial, Pasivo Agresivo, Sentimientos, SVC, TF-IDF, Validación cruzada, Aprendizaje automático.

I. INTRODUCCIÓN

Las grandes cantidades de información con las que cuentan las instituciones a día de hoy posibilitan crear modelos que perfilan la realidad de forma precisa. La minería de opinión es una de las técnicas que facilitan crear estos modelos, y posibilita, por ejemplo, saber si una película ha sido exitosa o no basándose en las millones de críticas alojadas en la red. En esta línea, las instituciones académicas, a través de los portales donde los alumnos expresan su opinión acerca del docente, el contenido de las asignaturas y la calidad de las clases, pueden emplear estos datos para mejorar su oferta educativa [1].

Las instituciones no poseen un software intuitivo que les acerque esta tecnología de una manera amigable y completa; muchas veces no sirve con simplemente poseer la información, pues para ser de utilidad esta también ha de ser transformada, filtrada y procesada de forma correcta, lo que exige herramientas especializadas [2].

Debido a ello, en este trabajo se creará un software que, mediante el uso de inteligencia artificial, permitirá a las diferentes organizaciones educativas obtener un diagnóstico completo de la calidad de su producto a partir de la minería de opinión de los comentarios de sus estudiantes. Se emplearán técnicas de minería y aprendizaje automáticos ya existentes, incidiendo en la accesibilidad y en el empleo de procedimientos y recursos eficientes.

El aporte de este proyecto será poner a disposición de diferentes instituciones educativas

un aliado que les permita redefinir su manera de impartir clase, contribuyendo a una mejora en la calidad en la forma de enseñar.

II. ESTADO DEL ARTE

El procesamiento del lenguaje natural se sitúa en el corazón de la resolución del problema planteado como origen de este proyecto. Consiste en el uso de la computación para procesar o entender el lenguaje humano en su estado natural, con el fin de desempeñar tareas útiles [3].

Es necesario preparar el contenido textual mediante procesos de múltiples fases para obtener datos estructurados que puedan ser tratados por métodos estadísticos [1].

De acuerdo a [1], el procedimiento llevado a cabo para poder convertir información textual en datos procesables numéricamente es el siguiente:

En primer lugar, el texto es analizado y *tokenizado* para obtener un conjunto de secuencias alfanuméricas que se corresponden a las expresiones empleadas en el texto. En segundo lugar, se aplica un pre-procesado al texto, pasando todas las palabras a minúscula; otras operaciones realizadas consisten en la eliminación de números o la corrección de palabras mal escritas. Posteriormente, se trata la inflexión de los términos de dos maneras diferentes: mediante derivación o lematización. Puede realizarse otra poda más a fin de evitar términos que no informan (los conocidos como *stop-words*) eliminando las palabras más comunes utilizadas en la lengua en la que está escrita el texto o en el ámbito al que pertenece. Aquellas palabras con un bajo número de apariciones también son eliminadas del vocabulario. De esta manera, el contenido textual es convertido en una estructura de datos procesable.

La mayor aplicación del análisis de sentimientos es la minería de opinión (MO), la cual consiste en la extracción de sentimientos y opiniones a partir de un texto escrito en lenguaje natural a través de métodos computacionales [4].

[5] discute varios métodos de clasificación, la cual consiste en el mapeo entre los patrones y las etiquetas. En este artículo se comparan la clasificación Naïve Bayes, la entropía máxima y SVM (*Support Vector Machine*), concluyendo que esta última es la que mejores resultados reporta actualmente.

En [6] se concluye que los algoritmos de aprendizaje automático que mejor funcionan son Pasivo Agresivo y Regresión de Cresta, pues arrojan una precisión que va desde el 87 % hasta el 99,96 %. De acuerdo a [7], de entre 102 trabajos científicos SVM y Naïve Bayes son los dos clasificadores más utilizados, siendo empleados en 36 y 14 trabajos respectivamente.

III. OBJETIVOS Y METODOLOGÍA

El objetivo de este trabajo es ofrecer a las instituciones educativas una solución que les permita obtener información gráfica acerca de las percepciones de los estudiantes frente a la labor de sus docentes a partir de las opiniones de sus alumnos. Para ello, se emplearán técnicas de minería de opinión y de aprendizaje automático, concretamente, técnicas de aprendizaje supervisado.

Se utilizará validación cruzada con $K = 5$ para comprobar el correcto funcionamiento del algoritmo que vertebra la funcionalidad de la aplicación, debido a que es la técnica que mejor se ajusta a su naturaleza. El objetivo principal es llegar a tener una precisión del 90 % como mínimo.

Para determinar qué método de minería de opinión funciona mejor, se seleccionarán tres algoritmos diferentes que, de acuerdo al estado del arte, son los que mejor funcionan: Pasivo Agresivo, SVM y Naïve Bayes. Junto con estos algoritmos se emplearán también los algoritmos de extracción de características Bag of Words y Frecuencia de término–Frecuencia de documento inversa. El objetivo principal es construir tres modelos diferentes con estos algoritmos y optar por el que mejores resultados reporte.

Se utilizará validación cruzada con 5 grupos y curvas ROC para asegurar que los algoritmos de *machine learning* funcionan correctamente. Para complementar estos resultados, se inicializará desde cero un modelo que haga uso de la mejor combinación de algoritmos de representación y clasificación, se entrenará con un 80 % de los datos y se testeará con el 20 % restante. Para evaluar la usabilidad y la funcionalidad del software, se hará uso del sistema *SUS*, el cual permite medir estos atributos a partir de preguntas. Para dar la aplicación por correcta, se buscará tener una puntuación de 70 o superior.

IV. CONTRIBUCIÓN

Para poder entrenar el modelo de aprendizaje automático se han utilizado aproximadamente 8700 comentarios provenientes de alumnos acerca de sus profesores en la Universidad Mariana de Colombia. Se ha contado con un total de 20 hojas de cálculo de extensión *.xlsx* que contienen los comentarios de los alumnos y el docente al que van dirigido. Cada opinión tiene adjunta un número que designa el sentimiento que evoca:

- 0: Comentario negativo.
- 1: Comentario neutro.
- 2: Comentario positivo.

A. Preprocesamiento de los datos

En primer lugar, se han realizado correcciones a palabras o frases mal escritas en los archivos de extensión *.xlsx*, a fin de disminuir posibles problemas en el procesamiento de datos. En segundo lugar, se han sustituido comentarios consistentes en símbolos no numéricos o expresiones carentes de sentido por vacío. Posteriormente, se han etiquetado manualmente los comentarios.

A.1. Corrección de palabras

Cada comentario de la lista de comentarios se desglosa en tokens (instancias individuales de palabras y signos de puntuación), y se corrigen términos mal escritos sustituyéndolos por la corrección propuesta por la librería *SpellChecker*.

A.2. Conversión de palabras a sinónimos

Se sustituyen en cada comentario un conjunto discreto de palabras y expresiones por un sinónimo contenido en el dataset de entrenamiento. No obstante, esta técnica está orientada a ser utilizada en la predicción y no en el entrenamiento, pues reduce el vocabulario del modelo; por ese mismo motivo, no ha sido empleada en dicha fase.

A.3. Lematización de palabras

Se ha sustituido cada palabra de cada comentario por su lema mediante el uso de la librería *StanzaLanguage*.

B. Modelos empleados

B.1. Modelos de representación

Existe la necesidad de transformar los comentarios representados mediante texto en datos numéricos interpretables por los algoritmos de aprendizaje automático. Los algoritmos de extracción de características que se utilizarán serán *Bag of Words* y Frecuencia de término - Frecuencia de documento inversa (*TF-IDF*).

- Bag of Words: Consiste en una representación de texto donde se muestra el número de veces que aparece una palabra dentro de un documento; en este caso, en cada comentario. De esta forma, se mide la presencialidad de cada palabra dentro del conjunto de comentarios.

- TF-IDF: Expresa cuán importante es una palabra dentro de un documento en comparación con el resto. La fórmula [8] que lo define es la 1.

$$TF(i,j) = \frac{\text{Frec. término } i \text{ en doc } j}{\text{Nº palabras totales en doc } j}$$

$$IDF(i) = \log_2(\frac{\text{Nº total de docs}}{\text{Nº de docs con la palabra } i})$$

Para el término i y documento j:

$$TF - IDF(i,j) = TF(i,j) * IDF(i) \quad (1)$$

B.2. Modelos de clasificación

Tras representar los datos de forma numérica, queda crear modelos que aprendan a relacionarlos entre sí para poder clasificarlos correctamente. Los 3 algoritmos seleccionados para cumplir con este fin son *SVC*, *Naïve Bayes Multinomial* y *Pasivo Agresivo*.

- SVC: Se trata de las *Support Vector Classifiers*, un subtipo de *SVM* (*Support Vector Machines*). Es un modelo que representa los puntos de muestra en el espacio, separando las clases en espacios lo más amplios posibles empleando un hiperplano [9].
- Naïve Bayes Multinomial: Se trata de una forma modificada del clasificador *Naïve Bayes*, la cual también posee un enfoque probabilístico similar a este [10]. Está especialmente diseñada para tratar con documentos de texto y para calcular la ocurrencia de cada palabra. Se cálcula la probabilidad individual de que un documento pertenezca a cada una de las clases, y se le asigna aquella cuya mayor probabilidad posea [11]. Calcula las hipótesis de que los términos presentes en un documento ya se encuentren en otros pertenecientes a una clase concreta[10].
- Pasivo Agresivo: Es un algoritmo de aprendizaje en línea; esto significa que puede actualizar sus parámetros conforme recibe nuevos datos [12]. Es un paradigma de aprendizaje consistente en entrenar un modelo introduciéndole datos de forma secuencial; de esta manera, los modelos de aprendizaje en línea siguen aprendiendo aún cuando ya están desplegados en producción. De acuerdo a [13], es una modificación del algoritmo Perceptrón estándar.

	SVC	MultinomialNB	Passive Aggresive
BoW	88.54 %	90.41 %	89.38 %
TF-IDF	91.42 %	90.94 %	91.59 %

Cuadro 1: Resultados (*f1_macro*) proporcionados por la combinación de los algoritmos de extracción de características (filas) y los de clasificación (columnas)

B.3. Selección de los mejores modelos

Para seleccionar qué modelos han sido los óptimos se ha hecho uso de la clase *GridSearchCV* de *Sklearn.*.

Según se puede contemplar en la tabla B.3, la cual detalla el porcentaje de la métrica *F1_macro*, el modelo que mejor funciona es *Passive Aggressive* con $C = 0.03$, $loss = hinge$ y $warm\_start = True$ en combinación con *TF-IDF*.

C. Despliegue del modelo

En primer lugar, se declaran las configuraciones necesarias para el funcionamiento de *Flask*. Posteriormente, se inicializa el modelo leyéndolo del archivo *.joblib* que lo contiene. Se ha empleado *Passive Aggresive*, pues es el que mejor resultados presenta. Al llamar al endpoint */predict* se busca el archivo *.csv* que contiene los comentarios. Tras realizar la lectura de los comentarios, se procesan en *obtain_predictions*. En último lugar, devuelve las predicciones en formato *.json*.

D. Interfaz gráfica

La interfaz gráfica consiste en una sencilla página *HTML* donde el usuario ha de introducir un archivo de extensión *.csv* que contenga los comentarios a procesar y con *Comentario* como única cabecera. Una vez posee dicho documento, simplemente tiene que hacer click en *Seleccionar archivo*, seleccionarlo, y hacer click en el botón *predecir*. Tras ello, se indicará que

los comentarios han sido enviados y aparecerá una anotación indicando que están siendo procesados. Una vez se lleva a cabo el procesamiento por parte del modelo se reciben dos tablas: la primera contiene los comentarios con los sentimientos que les han sido asignados; la segunda posee dichos comentarios agrupados por sentimientos. Finalmente, se muestran cuántos comentarios son negativos, positivos y neutros así como el sentimiento más frecuente entre todos.

V. RESULTADOS O EVALUACIÓN

A. Evaluación de los modelos de Machine Learning

Para evaluar el funcionamiento de los diferentes modelos de *Machine Learning* se ha optado por obtener las curvas ROC que proporcionan cada combinación de algoritmos mencionados, además de entrenar y testear un modelo desde 0.

En primer lugar, se instancian uno de los modelos haciendo uso de los hiperparámetros que mejores resultados han arrojado para su algoritmo correspondiente. Acto seguido, se transforman los comentarios pre-procesados en datos numéricos. Posteriormente, se divide este conjunto de datos en 2 partes: un conjunto de entrenamiento (80 % del total) y conjunto de test (20 % restante). Los datos se entremezclan para evitar *overfitting* y para reducir la varianza. Después, se entrena el modelo con el conjunto de datos de entrenamiento, para luego obtener los resultados. Tras adquirirlos, se logra la gráfica con las curvas ROC para cada clase haciendo uso de la función *plot_roc_curve* de la librería *scikitplot*. Las imágenes 123 muestran las soluciones conseguidas, y tal y como se puede advertir, todas las combinaciones superan el 90 % en todos los indicadores, evidenciando que los modelos funcionan positivamente. Para complementar esta evaluación, se ha optado también por entrenar un modelo con los algoritmos TF-IDF y Pasivo Agresivo y testearlo desde 0. Se entrena con un 80 %

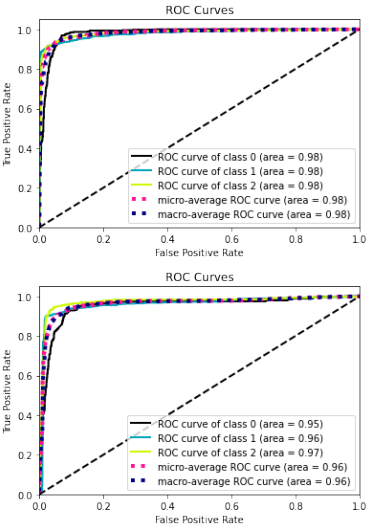


Figura 1: Curvas ROC de SVC para TF-IDF (arriba) y Bag of Words (debajo)

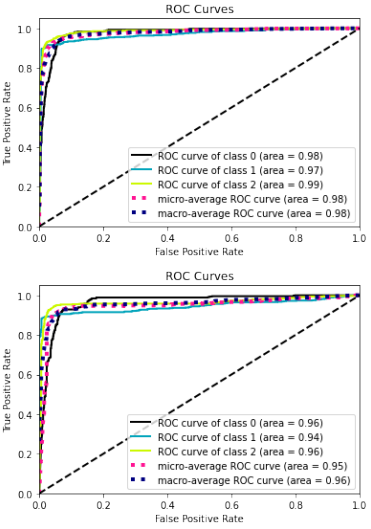


Figura 2: Curvas ROC de Naïve Bayes Multinomial para TF-IDF (arriba) y Bag of Words (debajo)

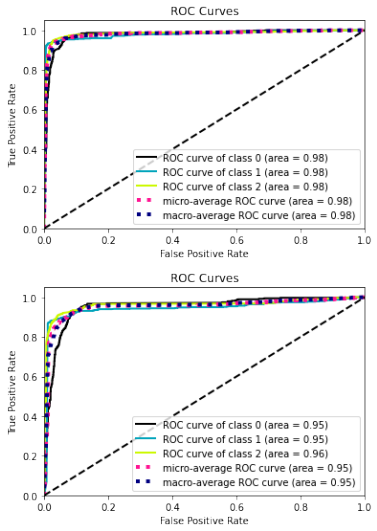


Figura 3: Curvas ROC de Pasivo Agresivo para TF-IDF (arriba) y Bag of Words (debajo)

de los datos y se testea con el 20 % restante. Las instancias totales correctamente clasificadas son el 91.70 %, los comentarios negativos el 94.94 %, los neutros el 89.17 % y los positivos el 91.98 %.

B. Evaluación de la aplicación web

Para medir la funcionalidad y usabilidad de la aplicación web se ha calculado el índice *SUS* de la misma rellenando el formulario proporcionado por *SUS Calculator*. Tras haberse evaluado la aplicación con 5 usuarios, se ha obtenido una nota de 78, por lo que la aplicación web cumple con los requerimientos de usabilidad y funcionalidad requeridos. Para evaluar su funcionamiento como usuario, se ha optado por predecir los siguientes comentarios, siendo rojo una predicción de sentimiento negativo, azul neutro y verde positivo:

- Es una profesora fantástica.
- No es bueno.
- Es un mal profesor.
- Es un buen profesor.
- Horrible como maestro.
- Nada que decir.
- Enseña muy bien.
- Muy contenta con la enseñanza.
- Debería tener más en cuenta la opinión de los alumnos.
- El mejor maestro que he tenido.
- No me gustó como impartió clase.
- Me gustó como impartió clase.
- Debería mejorar sus métodos.
- Ojalá volver a tenerle como docente.
- Sin comentarios.
- Es deficiente como profesor.
- Espero volver a verle.
- Espero no volver a verle.
- Es bueno como profesor, pero sus clases tienen que ser más amenas.
- Debería vocalizar mejor, no obstante, es bueno.
- Sus clases son aburridas.

VI. DISCUSIÓN O ANÁLISIS DE RESULTADOS

De acuerdo a los resultados obtenidos, se puede concluir que la aplicación funciona correctamente. Las curvas ROC indican una excelente relación entre las tasas de falsos y verdaderos positivos; en esa línea, existe una sintonía con los resultados arrojados por el entrenamiento del modelo desde 0, donde se predicen correctamente el 91 % de los casos, llegando alrededor del 90 % de clasificaciones correctas para cada clase. Finalmente, las clasificaciones

realizadas por la interfaz gráfica, donde los comentarios cuentan con una variedad léxica propia de un caso de uso real, devuelve unos resultados que se alinean con los demás, evidenciando el buen funcionamiento de la aplicación. Además, la puntuación de 78 en SUS muestra que el programa cumple con los requerimientos de funcionalidad y usabilidad.

VII. CONCLUSIONES

En primer lugar, cabe destacar que la aplicación web ha demostrado ser de utilidad en los casos estudiados, y cumple con los objetivos definidos. En segundo lugar, se puede afirmar que los algoritmos evaluados en el estado del arte son de gran utilidad en el ámbito de la minería de opinión educacional, y se reafirman como de gran interés para los investigadores que decidan ahondar en esta disciplina. El algoritmo pasivo agresivo ha demostrado ser mejor que SVC y Naïve Bayes Multinomial. En lo que se refiere a los algoritmos de extracción de características, TF-IDF ha demostrado ser mejor que Bag of Words, por lo que debería ser utilizado tanto en cualquier posible evolución futura de esta aplicación como en otras relacionadas a la minería de opinión educacional. No obstante, hay diversos aspectos a mejorar pese a los buenos resultados obtenidos:

- La aplicación debería dar la posibilidad al usuario de elegir qué algoritmos quiere emplear.
- El modelo de aprendizaje automático debería ser capaz de aprender con cada nuevo conjunto de comentarios predichos.
- Se debería poder entrenar un nuevo modelo o uno ya existente mediante un *.csv*, no simplemente realizar predecciones.
- Se debería poder tener un registro histórico de las predicciones realizadas.
- Se debería rediseñar la interfaz gráfica con un nuevo diseño para que sea más atractiva y eficiente visualmente (por ejemplo, teniendo tablas cuya anchura se ajuste de manera dinámica).
- Se deberían entrenar los modelos con más comentarios y con más diversidad léxica.
- Se debería contar con un número parejo de comentarios positivos, neutros y negativos.
- Se deberían explorar más algoritmos de extracción de características y de clasificación.
- Se debería llegar a un 95 % de precisión.
- Se deberían poder detectar más sentimientos.
- Se debería contar con datos que provengan de más instituciones educativas.
- Se debería poder realizar predicciones en más idiomas además del castellano.
- Se deberían poder predecir los sentimientos de comentarios escritos desde la propia interfaz gráfica.
- Se debería contar con una arquitectura e infraestructura de software más sofisticada.

Referencias

[1] Michelangelo Misuraca, Germana Scepi, and Maria Spano. Using opinion mining as an educational analytic: An integrated strategy for the analysis of students' feedback. *Studies in Educational Evaluation*, 68:100979, 2021.

[2] Stefan Slater, Srećko Joksimović, Vitomir Kovanovic, Ryan S Baker, and Dragan Gasevic. Tools for educational data mining: A review. *Journal of Educational and Behavioral Statistics*, 42(1):85–106, 2017.

[3] Li Deng and Yang Liu. A joint introduction to natural language processing and to deep learning. In *Deep learning in natural language processing*, pages 1–22. Springer, 2018.

[4] Bing Liu. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge university press, 2020.

[5] Jayashri Khairnar and Mayura Kinikar. Machine learning algorithms for opinion mining and sentiment classification. *International Journal of Scientific and Research Publications*, 3(6):1–6, 2013.

[6] Donia Gamal, Marco Alfonse, El-Sayed M. El-Horbaty, and Abdel-Badeeh M. Salem. Analysis of machine learning algorithms for opinion mining in different domains. *Machine Learning and Knowledge Extraction*, 1(1):224–234, 2018.

[7] Mujtaba Husnain, Malik Muhammad Saad Missen, Nadeem Akhtar, Mickaël Coustaty, Shahzad Mumtaz, and VB Prasath. A systematic study on the role of sentiwordnet in opinion mining. *Frontiers of Computer Science*, 15(4):1–19, 2021.

[8] Eu Jin Lok. Episode 1: Using tf-idf to identify the signal from the noise. "https://mungingdata.wordpress.com/2017/11/25/episode-1-using-tf-idf-to-identify-the-signal-from-the-noise/", Nov 2017.

[9] Universidad Internacional de la Rioja. Tema 8. aprendizaje supervisado: clasificación con máquinas vector de soporte. *Universidad Internacional de la Rioja*.

[10] Prabha PM Surya, Lakshmi V Seetha, and B Subbulakshmi. Analysis of user emotions and opinion using multinomial naive bayes classifier. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 410–415. IEEE, 2019.

[11] Arthur V. Ratz. Multinomial na - ve bayes' for documents classification and natural language processing (nlp). https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6 Apr 2022.

[12] Nicolò Cesa-Bianchi and Francesco Orabona. Online learning algorithms. *Annual review of statistics and its application*, 2021.

[13] Ajitesh Kumar. Passive aggressive classifier: Concepts amp; examples. https://vitalflux.com/passive-aggressive-classifier-concepts-examples#:~:text=The%20passive%20aggressive%20algorithm%20is,and%20the%20number%20of%20misclassifications., Oct 2022.