UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

TRABAJO DE TITULACIÓN

Minería de texto para el análisis de sentimientos, en procesos de enseñanza aprendizaje al utilizar la red social Facebook

Autor: Astudillo Bustamante, Ivo Andrés

Directora: Jara Roa, Dunia Inés

LOJA – ECUADOR

2021

2021

Aprobación del director del Trabajo de Titulación

Loja, 30 de junio de 2021

Magister.
Fernanda Maricela Soto Guerrero
Coordinadora de Titulación

Ciudad. -

De mi consideración:

El presente Trabajo de Titulación denominado: Minería de texto para el análisis de sentimientos, en procesos de enseñanza aprendizaje al utilizar la red social Facebook realizado por Ivo Andrés Astudillo Bustamante ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo. Así mismo, doy fe que dicho Trabajo de Titulación ha sido revisado por la herramienta antiplagio institucional.

Particular que comunico para los fines pertinentes.

Atentamente,

Dunia Inés Jara Roa.
C.I.: 1102005095

Declaración de autoría y cesión de derechos

"Yo, Ivo Andrés Astudillo Bustamante, declaro y acepto en forma expresa lo siguiente:

- Ser autor del Trabajo de Titulación denominado: Minería de texto para el análisis de sentimientos, en procesos de enseñanza aprendizaje al utilizar la red social Facebook, específicamente de los contenidos comprendidos en: Introducción, Capítulo 1. Contexto de la investigación, Capítulo 2. Marco teórico, Capítulo 3. Metodología de desarrollo, Capítulo 4. Aplicación de algoritmos y análisis de resultados, siendo Dunia Inés Jara Roa, directora del presente trabajo; y, en tal virtud, eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones judiciales o administrativas, en relación a la propiedad intelectual. Además, ratifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo son de mi exclusiva responsabilidad.
- Que mi obra, producto de mis actividades académicas y de investigación, forma parte del patrimonio de la Universidad Técnica Particular de Loja, de conformidad con el artículo 20, literal j, de la Ley Orgánica de Educación Superior; y, artículo 91 del Estatuto Orgánico de la UTPL, que establece: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad".
- Autorizo a la Universidad Técnica Particular de Loja para que pueda hacer uso de mi obra con fines netamente académicos, ya sea de forma impresa, digital y/o electrónica o por cualquier medio conocido o por conocerse, sirviendo el presente instrumento como la fe de mi completo consentimiento; y, para que sea ingresada al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública, en cumplimiento del artículo 144 de la Ley Orgánica de Educación Superior.

Autor: Ivo Andrés Astudillo Bustamante

C.I.: 1104902174

Dedicatoria

Dedico el presente trabajo de fin titulación a mi familia, especialmente a mi madre que ha creído en mí en todo momento y ha sido soporte fundamental para no dejarme decaer en el proceso de formarme como profesional.

Agradecimiento

Agradezco a mis padres por todo el esfuerzo y apoyo incondicional que me han dado en estos años de estudio. Agradezco muy especialmente a mi directora de trabajo de fin de titulación Mgtr. Dunia Inés Jara Roa, por su preocupación y por compartirme sus conocimientos a lo largo del desarrollo del presente trabajo de fin de titulación.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Resumen

Los algoritmos de aprendizaje automático son ampliamente utilizados en problemas de minería de texto, en este trabajo en específico se los utilizó para realizar análisis de sentimientos. El corpus textual en el cual se probaron los algoritmos fue generado a partir de interacciones que tenían los estudiantes y docente en grupos de estudio creados en la red social Facebook. Se probaron dos algoritmos: Support Vector Machine y Complement Naïve Bayes. Los resultados obtenidos mostraron que, basados en métricas como la sensibilidad, precisión y el puntaje F1, el algoritmo que tiene un mejor rendimiento es el Complement Naïve Bayes.

Palabras claves: Análisis de sentimientos, Support Vector Machine, Complement Naïve Bayes.

Abstract

Machine learning algorithms are widely used in text mining problems, in this specific work they were used to perform sentiment analysis. The textual corpus in which the algorithms were tested was generated from interactions that students and teachers had in study groups created on the social network Facebook. Two algorithms were tested: Support Vector Machine and Complement Naïve Bayes. The results obtained showed that, based on metrics such as Recall, precision and the F1 score, the algorithm with the best performance is the Complement Naïve Bayes.

Keywords: Sentiment Analysis, Support Vector Machine, Complement Naïve Bayes.

Introducción

Las empresas e instituciones tanto públicas como privadas generan una basta cantidad de datos a diario que necesariamente tienen que ser procesados con el fin de obtener información que sea relevante para la toma de decisiones. Debido a la inmensa cantidad de datos este proceso tiene que ser automatizado. En los últimos años esto ha sido posible gracias a campos como la minería de datos, minería de texto y ciencias a fines. Una de las muchas aplicaciones que se pueden desarrollar con la aplicación de estos campos es el análisis de sentimientos. Esta en particular permite que las empresas, instituciones y otras entidades tengan una visión más profunda de lo que los clientes, empleados, estudiantes y personas en general piensan de ellas y así poder tomar decisiones concretas que sirvan para la mejora de procesos, servicios o productos dependiendo el caso. En esta investigación en particular, la empresa es una institución de educación superior y los datos generados pertenecen a estudiantes de esta universidad. Uno de los objetivos es determinar si es correcto o no el uso de redes sociales tradiciones en cuanto a enseñanza – aprendizaje se refiere y de esta manera tomar decisiones que mejoren los procesos educativos con el fin de que se brinde el mejor servicio de educación posible.

Para dar cumplimiento a los objetivos planteados en el presente trabajo de fin de titulación, el documento fue dividido en cuatro capítulos. En el capítulo uno se explica el contexto de la investigación y se abarcan temas como el planteamiento del problema y el alcance de este. El capítulo dos corresponde al marco teórico en el cual se abordaron todas las temáticas necesarias para resolver de mejor manera los problemas de esta investigación. En el capítulo tres se empieza a dar forma de cómo se abordó la problemática. Para ello fue necesaria la elección de una metodología de minería de texto que sea adecuada a los datos recopilados y la cual se desglosa en fases que están explicadas en este capítulo. Finalmente el capítulo cuatro corresponde a la fase final de la metodología en la cual se abarca la aplicación de los algoritmos Support Vector Machine y Complement Naïve Bayes y el respectivo análisis de resultados.

Para finalizar esta introducción es necesario acotar que este trabajo tiene una gran importancia para la institución ya que se deja como base un corpus textual al cual se lo puede seguir alimentando con nuevos comentarios de tal manera que se puedan seguir probando distintos algoritmos a parte de los que se evaluaron en esta investigación. De esta manera la institución podrá automatizar procesos que permitan el análisis de texto con el objetivo primordial de tomar decisiones que mejoren la calidad en lo que al servicio de educación se refiere.

Capítulo uno

Contexto de la investigación

La finalidad de este capítulo es presentar una introducción al presente trabajo de titulación. Se empieza con la sección 1.1. en la cual se describe el tema de la investigación. Seguido se tiene el planteamiento del problema, en la sección 1.2. En la sección 1.3. se describe el alcance de la investigación. En la sección 1.4. se plantean los objetivos tanto general como específicos. En la sección 1.5 se analiza la metodología y estrategia de desarrollo y se finaliza este capítulo con una explicación de la estructura del documento en la sección 1.6.

1.1. Tema

Minería de texto para el análisis de sentimientos, en procesos de enseñanza aprendizaje al utilizar la red social Facebook.

1.2. Planteamiento del problema

En la mayoría de las universidades del mundo, los entornos virtuales de aprendizaje y sistemas de gestión del aprendizaje se han convertido en las principales herramientas informáticas en procesos de enseñanza – aprendizaje. Estos sistemas permiten que el docente realice un seguimiento exhaustivo de distintas actividades como: tareas, evaluaciones, foros, chats de discusión, etcétera. Además, sirven como una especie de red social educativa, en donde el docente y los alumnos interactúan constantemente, pero... ¿qué hay de las redes sociales tradicionales?, ¿es posible llevar a cabo procesos de enseñanza – aprendizaje en alguna red social, por ejemplo Facebook?

De ser así, ¿qué actitudes muestran los estudiantes al utilizarla?, La presente investigación pretende responder a estas interrogantes, partiendo de la base que ya se tiene grupos de Facebook que están dirigidos por un docente, en los cuales se desarrollan actividades académicas de una materia en concreto. Particularmente las actividades planteadas por la docente son: chats de discusión de algún tema en específico y el desarrollo colaborativo de tareas.

De esta manera, la presente investigación está orientada al análisis computacional de los datos generados en los grupos de estudio de Facebook con el fin de determinar si los estudiantes están conformes al utilizar esta red social como herramienta de apoyo en procesos de enseñanza – aprendizaje.

1.3. Alcance

La investigación se centra en el análisis de sentimientos de estudiantes de la carrera de Ingeniería Informática de la Universidad Técnica Particular de Loja, específicamente de la modalidad Abierta y a Distancia, que interactúan en grupos de estudio de la materia de Inteligencia Artificial en la red social Facebook. El dataset será generado con textos de aproximadamente treinta grupos conformados por cinco estudiantes cada grupo. El texto será pre - procesado con técnicas de NLP (procesamiento de lenguaje natural) para posteriormente evaluar los algoritmos de aprendizaje automático. Para llevar a cabo la experimentación se va a trabajar con el lenguaje de programación Python.

1.4. Objetivos

Para el presente trabajo de titulación se han definido los siguientes objetivos:

1.4.1. General

Identificar las actitudes de los estudiantes cuando se utiliza la red social Facebook en procesos de enseñanza aprendizaje, para detectar la pertinencia de uso en procesos formativos reglados.

1.4.2. Específicos

- Creación de un corpus textual con las intervenciones de cada uno de los grupos en la red social Facebook.
- Elegir la metodología más apropiada para la realización de este tipo de minería de texto.
- Seleccionar el algoritmo de aprendizaje automático que mejor se ajuste a los datos recolectados a fin de obtener la mayor tasa de aciertos posibles.
- Realizar la evaluación e interpretación de los resultados obtenidos.

1.5. Metodología y estrategia de desarrollo

El presente trabajo de titulación costa de tres etapas principales; la primera etapa, corresponde a la elaboración del marco teórico acerca de la minería de texto. Así como, de las diferentes técnicas usadas en problemas de minería de texto, poniendo especial énfasis en el análisis de sentimientos. La segunda etapa, corresponde al proceso de extracción de los datos, procesamiento y aplicación de los algoritmos de minería de texto. Finalmente, en la última etapa se llevará a cabo el análisis de los resultados obtenidos.

1.6. Estructura del documento

El presente trabajo de titulación está estructurado en cuatro capítulos. Se plantean de tal manera que están ordenados lógicamente con la finalidad de responder adecuadamente a los objetivos de la investigación.

En el **capítulo uno** se presenta el contexto de la investigación, en el cual se incluye; el tema, planteamiento del problema, alcance, objetivos, metodología y estrategia de desarrollo y estructura del documento.

El **capítulo dos** constituye el marco teórico de la investigación en cuestión. Por un lado, a manera de introducción se empieza explicando lo que es la minería de datos en general. Seguido de esto se aborda a profundidad la minería de texto y sus diferentes etapas, así como también las principales técnicas utilizadas en procesos de clasificación. También se hace alusión a los distintos usos que tiene a la minería de textos, profundizando especialmente en el análisis de sentimientos. Luego de esto se analizan las principales herramientas existentes en el mercado. Para finalizar esta sección se presenta un apartado con cuatro trabajos relacionados y un estudio comparativo de estos.

El **capítulo tres** está constituido por la metodología de desarrollo. La metodología inicia con el propósito de la minería de texto, seguido de la explicación de cómo se obtuvo el corpus textual. Luego se explica todos los pasos realizados en lo que al pre – procesamiento de texto se refiere. Finalmente en este capítulo se indica el proceso de transformación del texto a vectores numéricos.

En el **capítulo cuatro** se presenta la fase final de la metodología de desarrollo, la cual corresponde al proceso de aplicación del método de minería de texto. Como fase final y para dar por terminado este capítulo se realiza un análisis de los resultados obtenidos.

Capítulo dos

Marco teórico

En el presente capítulo se define el marco teórico de la investigación. Se inicia con el estudio de la minería de datos en la sección 2.1. En la sección 2.2 se analiza el descubrimiento de conocimiento. En la sección 2.3 se estudia los datos estructurados y los datos no estructurados. Seguido de esto, en la sección 2.4 se aborda la minería de texto. En la sección 2.5 se revisan las principales técnicas de la minería de texto usadas en problemas de clasificación. En la sección 2.6 se presentan distintas aplicaciones que tiene la minería de textos. En la sección 2.7 se comparan las principales herramientas existentes en el mercado para resolver problemas de minería de textos. En la sección 2.8 se estudian cuatro trabajos relacionados. Finalmente, en la sección 2.9 se realiza un estudio comparativo de estos trabajos.

2.1. Minería de datos

En la actualidad la cantidad de datos que generan las aplicaciones y sistemas informáticos es abrumadora. Desde una pequeña tienda de zapatos hasta las grandes empresas y organizaciones almacenan todo tipo de información. Imágenes, videos, datos numéricos, texto, etcétera; se guardan en las conocidas bases de datos las cuales permiten almacenar cantidades muy grandes de datos a costos relativamente bajos y que su integración y administración resulte cada vez más sencilla.

Ahora bien, Weiss *et al*. (2005) explican que los datos que se almacenan en tablas y que se transforman a una representación numérica se los conoce como datos estructurados. A estos datos se los puede procesar con técnicas estadísticas y obtener cierta información. El problema es que hoy en día no solo se almacenan datos estructurados, existe una gran cantidad de datos que se encuentran en un formato no estructurado por lo que si solo se trabaja con los datos que están almacenados en un formato de tablas, se estaría limitando la posibilidad de encontrar información relevante del otro conjunto de datos. Los datos estructurados y no estructurados se analizan en el apartado 2.3.

Con el paso del tiempo los investigadores se dieron cuenta que los datos contienen patrones e información que puede ser crucial para la toma de decisiones, es por eso que a partir de la década de los noventa, empezaron a aplicar nuevos métodos y técnicas que integran la estadística, la inteligencia artificial, el aprendizaje automático y los sistemas de bases de datos. Estos métodos y técnicas tienen el potencial de manejar y procesar grandes cantidades de datos con el fin de encontrar patrones ocultos y potencialmente útiles con el objetivo de que las empresas y organizaciones tomen mejores decisiones tanto para su beneficio, como para el de sus consumidores o la sociedad en general. Estos nuevos métodos y técnicas crearon un nuevo campo de investigación que se denomina minería de datos.

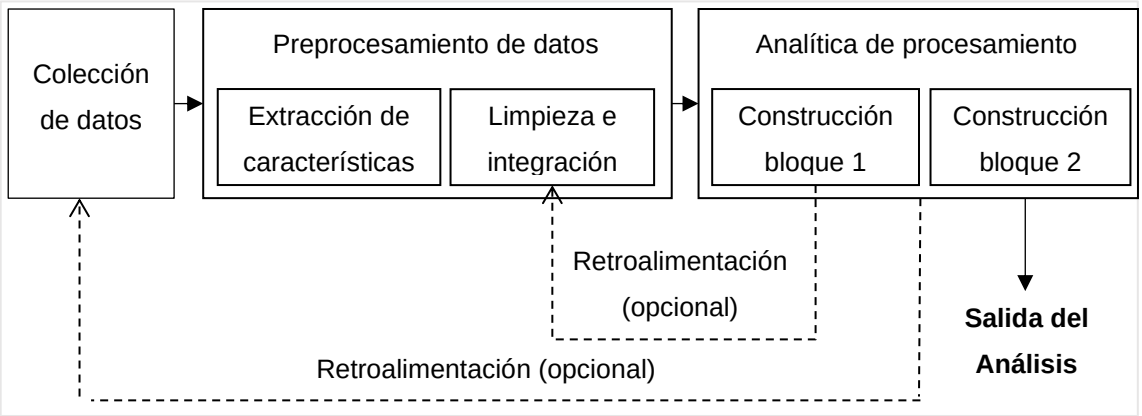
Para Aggarwal, como para muchos otros investigadores: “la minería de datos es el estudio de recopilar, limpiar, procesar, analizar y obtener información útil de los datos” (2015, p. 27).

Para Mourya & Gupta (2012) la minería de datos es el uso de tecnologías inteligentes que ayudan a los usuarios finales a extraer conocimiento oculto que reside en las grandes bases de datos.

La minería de datos es un proceso que contiene algunas fases, comenzando con la recopilación de los datos, seguido por la extracción de características y limpieza de los datos, el procesamiento analítico y el diseño de algoritmos (ver figura 1).

Figura 1

Pipeline del procesamiento de datos



Nota. Adaptado de *Data Mining* (p.4) [Diagrama de flujo], por Aggarwal, C. 2015. (https://doi.org/10.1007/978-3-319-14142-8). Springer International Publishing Switzerland

La explicación de estas fases está fuera del alcance de esta investigación ya que como veremos posteriormente los datos que se recopilaron para la presente, son datos en formato texto no estructurados, y para el manejo de estos se utilizan técnicas de minería de texto que están bastante relacionadas con la minería de datos es por esto que es conveniente iniciar esta investigación haciendo una breve explicación de la minería de datos.

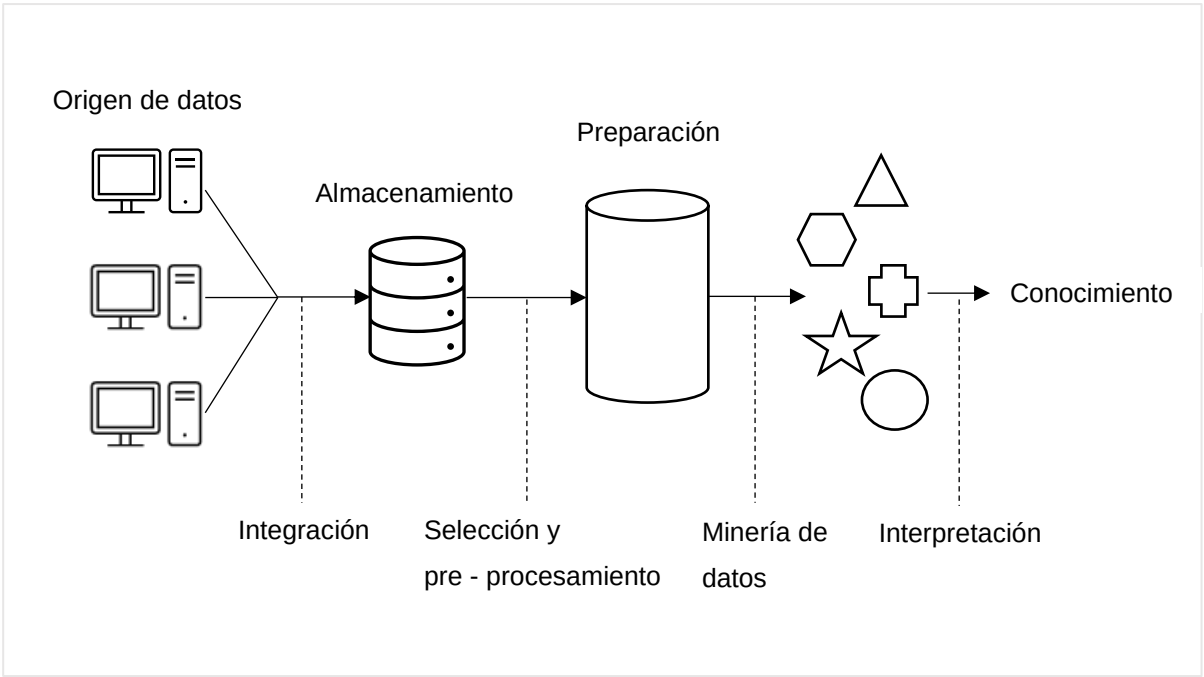
2.2. Descubrimiento de conocimiento

A diferencia de Aggarwal, Bramer (2016) centra la definición de minería de datos en algoritmos que permiten descubrir patrones o información oculta de los datos. Además, señala que la minería de datos es la parte central de un proceso mayor llamado descubrimiento del conocimiento.

Bramer define al descubrimiento de conocimiento como “la extracción no trivial de información implícita, previamente desconocida y potencialmente útil de los datos” (2016, p. 18).

Figura 2

El proceso de descubrimiento del conocimiento



Nota. Adaptado de *Principles of Data Mining* (p. 18) [Diagrama de flujo], por Bramer, M. 2016. (https://doi.org/10.1007/978-1-4471-7307-6). Springer-Verlag London Ltd.

En la figura 2 se observa los distintos enfoques de los investigadores con respecto a la minería de datos. Para Aggarwal la minería de datos abarca todo el proceso de descubrimiento de conocimiento, mientras que para Bramer la minería de datos es una fase de este proceso.

Esta diferencia la hacen notar Han *et al*. (2012) quienes indican que la minería de datos es tratada por muchos como un proceso de descubrimiento de conocimiento en bases de datos (KDD por sus siglas en inglés Knowledge Discovery in Databases), mientras que para otros, la minería de datos es un simple paso esencial en este proceso de descubrimiento de conocimiento.

2.3. Datos estructurados y datos no estructurados

Como se señaló en el apartado anterior, las técnicas de minería de datos trabajan con datos estructurados y para ello una vez que han sido recopilados, integrados y pre - procesados, estos deben de estructurarse mediante el proceso de “Preparación de datos” que se lo ve representado en la figura 2. O mejor aún como nos dicen Weiss *et al*. (2005) es posible recopilar datos ya estructurados en función de un diseño previo de minería de datos que se vaya a utilizar.

La mayoría de los datos que utilizan las empresas y organizaciones son de tipo texto y de tipo numérico. Weiss *et al*. (2005) mencionan que por regla general la minería de datos espera dos tipos de información: (a) ordenada numérica y (b) categórica. Los atributos numéricos corresponden, por ejemplo, al peso o los ingresos, que claramente tienen valores mayores o menores y por ende están ordenados. Mientras que, los atributos categóricos son códigos numéricos desordenados que tienen una definición para que puedan ser interpretados por las personas. Por ejemplo, los atributos de verdadero y falso, pueden ser representados por un uno o un cero; así como también, los géneros femenino y masculino que pueden ser representados por un código para su interpretación.

Weiss *et al*. (2005) concluyen que si los datos pueden describirse mediante una hoja de cálculo con su formato tabular, estos están en una fase altamente estructurada. (ver tabla 1).

Tabla 1

Ejemplo de una hoja de cálculo de datos médicos

	Altura	**Peso**	**Código**
M	175	65	3
F	141	72	1
. . .	. . .	. . .	. . .
F	160	59	2

Nota. Adaptado de *Text Mining (p.3)*, por Weiss et al., 2005, (https://doi.org/10.1007/978-1-84996-226-1). Springer Science+Business Media, Inc.

A diferencia de los datos estructurados, los datos no estructurados son textos. “El texto suele ser una colección de documentos no estructurados sin requisitos especiales para la composición de documentos”. (Weiss *et al*., 2005, p.12)

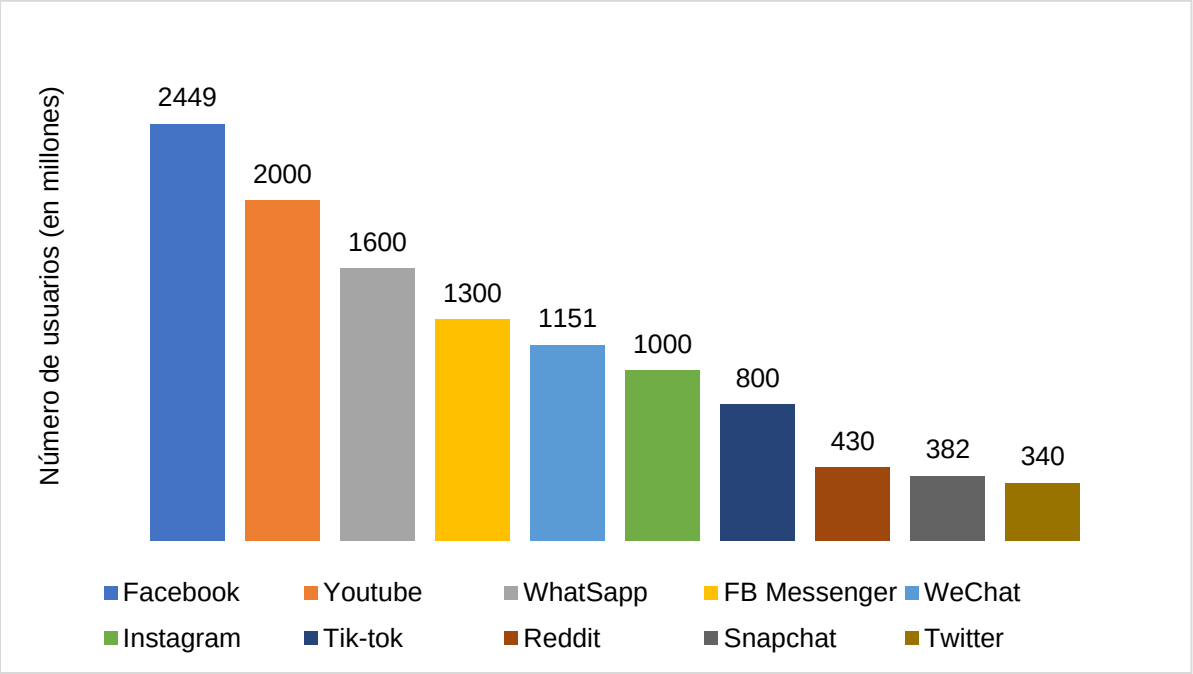
Todo tipo de revistas, artículos, documentos, e-books, blogs, sitios de noticias, post y conversaciones en redes sociales; son información no estructurada en lo concerniente a la minería de datos y sabemos bien que esta información es la que más abunda hoy en día.

Armstrong (2019) en el sitio web *statista* publica que hasta octubre de 2019 existen 1,72 mil millones de sitios web, la gran mayoría de estos sitios contienen información no estructurada. Los reportes y estadísticas existentes sobre el uso de sitios web muestran que la mayor cantidad de tráfico se encuentra en las plataformas sociales.

> Más de 4.500 millones de personas usan Internet a principios de 2020, mientras que los usuarios de las redes sociales han superado la marca de los 3.800 millones.
>
> Casi el 60 por ciento de la población mundial ya está en línea, y las últimas tendencias sugieren que más de la mitad de la población total del mundo usará las redes sociales a mediados de este año (Kemp, 2020).

Figura 3

Plataformas sociales más usadas en el mundo



Nota. Adaptado de *DATAREPORTAL* [Gráfico de barras], por Kemp, S., 2020, (https://bit.ly/3fMYMVr). Kepios.

Como se aprecia en la figura 3, el número de usuarios activos por mes es realmente impresionante. Kemp destaca que las redes sociales que lideran las estadísticas a enero de 2020 son: Facebook con de 2.449 millones de usuarios activos mensualmente, seguida de YouTube con 2.000 millones de usuarios activos, luego se tiene a Whatsapp con 1300 millones.

En estas redes sociales las personas comparten datos en distintos formatos: fotos, videos y lo que más nos interesa en esta investigación los textos. Textos que viene en forma de comentarios, estados, conversaciones vía chats, tweets, discusiones, post; etcétera. En (Internet-live-stats) se puede observar estadísticas en tiempo real de algunas redes sociales, el día que se está redactando este documento (8 de mayo de 2020) tenemos que: se han escrito en Twitter un aproximado de 617 millones de tweets, se han enviado 208 mil millones de emails, se han publicado 117 millones de posts en Tumblr y se han escrito más de 5 millones de blogs.

Todos estos textos que se publican en blogs, chats, emails, redes sociales, etcétera, es información que los humanos la podemos entender; es decir, tienen un conocimiento explícito, pero como ya se había mencionado anteriormente, este tipo de datos no estructurados están en continuo crecimiento y se vuelve inhumano leer estas grandes cantidades de información. Es por eso que las empresas y organizaciones se ven en la necesidad de invertir en tecnologías que automaticen estos procesos con el fin de poder encontrar patrones o información potencialmente útil en los textos y es aquí en donde interviene otro campo de investigación que es la minería te texto que será abordada en el siguiente punto.

2.4. Minería de texto

Feldman & Sanger (2007) citados por Taeho (2019) mencionan que “La minería de texto se define como el proceso de extracción del conocimiento implícito de los datos textuales” (p. 17).

De la misma forma Banchs (2013) afirma que la minería de texto es el proceso de descubrir conocimiento cuando los datos de origen son texto.

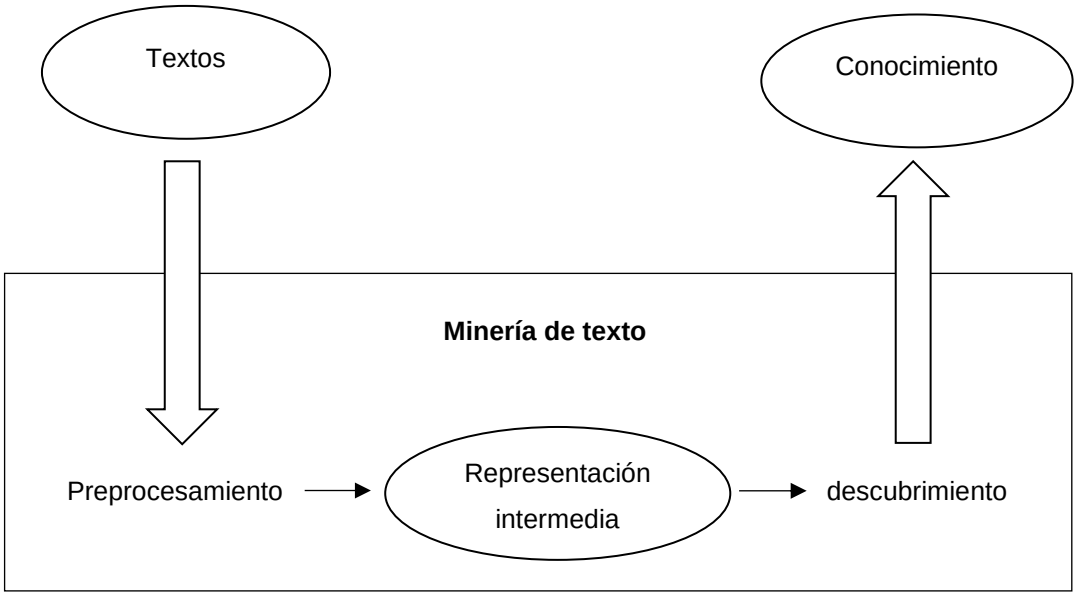
Estas definiciones tienen mucha similitud con el proceso KDD (descubrimiento de conocimiento en bases de datos) de minería de datos, tal como lo mencionan Justicia De La Torre *et al*. (2005). Ellos dicen que de manera similar a KDD, el descubrimiento de conocimiento en texto (KDT, por sus siglas en ingles Knowledge Discovery in Text) se puede definir como el proceso de extracción de información que previamente se desconocía y que puede ser potencialmente útil teniendo como fuente de datos un texto.

Para Weiss *et al.* (2005) los especialistas en minería de datos son conocidos como los “chicos de los números” porque sus datos están en forma numérica mientras que los especialistas en minería de texto esperan colecciones de documentos, donde los contenidos son legibles y tienen un significado obvio para las personas.

En la figura 4 se muestra un esquema general que se lleva a cabo en tareas de minería de texto.

Figura 4

Esquema general de la minería de texto

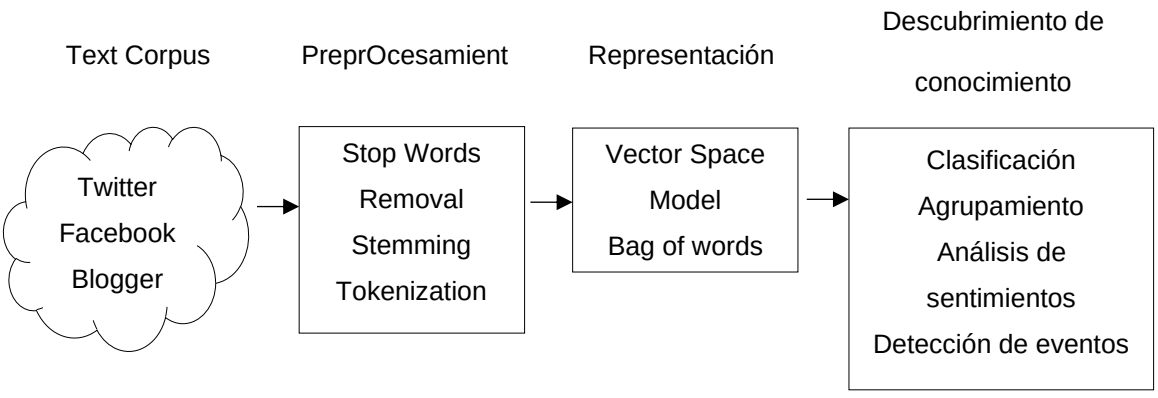


Nota. Adaptada de *Generador de los grafos conceptuales a partir del texto en español (p. 40)* [Diagrama de flujo], por Hernández Cruz, 2007, (https://bit.ly/3fPTto4).

De la misma forma Hu & Liu (2012) en el artículo “Text analytics in social media” hablan de un framework tradicional para el análisis de texto:

Figura 5

Framework tradicional para el análisis de texto



Nota. Adaptada de *Mining Text Data (p.389)* [Diagrama de flujo], por Hu & Liu, 2012, (https://doi.org/10.1007/978-1-4614-3223-4). Springer Science+Business Media, LLC.

Como se puede observar en la figura 5, existe un consenso por parte de los investigadores en lo que ha análisis de textos se refiere. La primera fase, consta de la obtención del texto, el cual puede ser documentos de un tamaño considerable o micropost que los usuarios comparten en sus redes sociales. La segunda fase, y quizá la más importante en este proceso es el preprocesamiento del texto con el cual se va a obtener una representación o forma intermedia del mismo. A esa representación o forma intermedia ya es posible aplicarle los algoritmos de minería de texto como tal, con los cuales se obtendrá el tan ansiado conocimiento.

2.4.1. Preprocesamiento

Los datos textuales están en formato no estructurado o mejor dicho se encuentran en un formato que es muy complejo para que lo entiendan las máquinas; por esta razón, no es posible aplicar directamente técnicas de minería de texto sobre ellos. El preprocesamiento es una fase esencial para el proceso de descubrimiento de conocimiento y consiste en transformar el texto a una forma intermedia que permita la aplicación de técnicas para la extracción de patrones.

En la figura 5, dentro de la fase de preprocesamiento se encuentran algunos métodos como la eliminación de palabras de detención (Stop Words Removal) que consiste en eliminar las preposiciones, los artículos, los pronombres personales y demás palabras que se las considera sin sentido que comúnmente se denominan palabras vacías. Otro método de preprocesamiento es el Stemming el cual consiste en expresar las diferentes formas de las palabras a una sola conocida como la raíz por ejemplo: “leí”, “leeré” y “leyendo” puede ser reducida a su forma raíz que es “leer”.

Hu & Liu (2012) dicen que las palabras con formas variantes se las puede considerar palabras que tienen la misma característica.

Otro método de preprocesamiento de textos es la tokenización (tokenization) y consiste en dividir el flujo de texto en palabras o frases. A estas palabras o frases separadas se las conoce como *tokens.* (Weiss *et al*., 2005).

2.4.2. Representación o forma intermedia

Una vez que el texto ha sido preprocesado con los métodos que se mencionó anteriormente, ahora es posible representarlo a una forma intermedia. En la figura 6 se mencionan las principales formas intermedias.

Modelo de espacio vectorial (Vector Space Model) que consiste en la representación del texto por medio de vectores de términos. “En el modelo de espacio vectorial, cada palabra está representada por una variable que tiene un valor numérico que indica el peso (importancia) de la palabra en el documento” (Allahyari et al., 2017, p. 4). Un método para elaborar un VSM es TF-IDF o frecuencia de término – frecuencia inversa de documento (Term frequency – Inverse document frequency), consiste en una medida numérica para expresar cuantas veces aparecen los términos en los documentos y de esta forma determinar su relevancia.

Otra forma de representación intermedia es la bolsa de palabras (Bag of Words). “Una representación basada en bolsa de palabras es aquella en la que se toman todas las palabras del documento, se eliminan sus relaciones tanto sintácticas como semánticas y se almacenan de tal modo que quede accesible para futuros tratamientos.” (Justicia de la Torre, 2017, p. 72). Hu & Liu (2012) dicen que en el modelo de bolsa de palabras, a las palabras se las representa como variables separadas que tienen un peso numérico de importancia variable.

Cifuentes (2016) menciona los N-Gramas para la representación de textos. En ellos los textos son separados en palabras únicas llamadas unigramas. En dos palabras llamadas Bi-gramas y en tres palabras llamadas Tri - gramas.

Otro método más avanzado de representar el texto en una forma intermedia, son las ontologías. Mizoguchi *et al.* (2006) citados en Justicia de la Torre (2017) dicen que una ontología es la forma de representar un marco conceptual, en el cual se incluye la descripción de los conceptos y las relaciones entre ellos.

Existen otras representaciones intermedias como: Jerarquía de conceptos o taxonomía de términos, grafos semánticos, red IS-A, grafos conceptuales, grafos conceptuales difusos, dependencia conceptual, ap-set, entre otras. Lo que hay que tener en

cuenta para la elección de una forma intermedia es que técnica de minería de texto se quiere aplicar en la fase siguiente; por ejemplo: “Si se desea utilizar técnicas de minería basadas en lógica, será necesario una representación intermedia del texto sobre la que se pueda realizar dicha inferencia. Si por el contrario, lo que se pretende es jugar con la mayor cantidad de información contenida en el documento, habrá que decidir primero qué representación intermedia nos conviene.” (Justicia de la Torre, 2017, p. 74).

Como se observa la minería de texto presenta grandes desafíos y a diferencia de la minería de datos, esta requiere de la aplicación de técnicas de recuperación de información, inteligencia artificial y Procesamiento de Lenguaje Natural (NLP por sus siglas en ingles Natural Language Processing) lo que hace que este campo de investigación sea un verdadero desafío. En el siguiente punto que corresponde a la fase previa de obtención de patrones o conocimiento, se revisarán las principales técnicas de minería de texto.

2.5. Técnicas de minería de textos

Técnicas de aprendizaje automático (Machine Learning) son ampliamente usadas en la minería de texto. Los modelos de aprendizaje automático se usan para reconocer patrones dentro de los datos textuales. Existen muchos algoritmos de aprendizaje automático, que de acuerdo con su utilidad o a la salida que producen se los clasifican en dos categorías principales que son; aprendizaje supervisado y aprendizaje no supervisado. Otra categoría es el aprendizaje por refuerzo.

Russell & Norvig dicen que; “el problema de aprendizaje supervisado consiste en aprender una función a partir de ejemplos de sus entradas y sus salidas.” Mientras que, “el problema de aprendizaje no supervisado consiste en aprender a partir de patrones de entradas para los que no se especifican los valores de sus salidas.” (2004, p. 740)

En otras palabras, se puede decir que los algoritmos de aprendizaje supervisado necesitan de conjuntos de datos etiquetados para ser entrenados, por ejemplo; si se quiere determinar si un comentario de cierto producto es positivo o negativo. El curso intensivo de aprendizaje automático impartido por Google menciona que primeramente se tiene que

entrenar el modelo con datos que previamente hayan sido clasificados, en nuestro caso de ejemplo comentarios clasificados en positivos o negativos, estos se pueden definir como:

$$datosEtiquetados = (x, y)$$

en donde x es el comentario, y es la categoría a la que pertenece.

En este ejemplo y identifica si el comentario es positivo o negativo. Lo que hace el modelo es aprender la relación que existe entre x e y, de esta manera se pueda deducir la categoría de ejemplos sin etiqueta, que en nuestro caso serían los nuevos comentaros que hacen los clientes. Un ejemplo no etiquetado se lo define como:

$$datosSinEtiqueta = (x, ?)$$

en donde x es el comentario, y ? es la etiqueta a inferir.

En el mismo curso intensivo de aprendiza automático de Google dicen que un modelo de aprendizaje automático supervisado se encarga de establecer la relación que existe entre los atributos y las etiquetas y consta de dos fases; entrenamiento e inferencia.

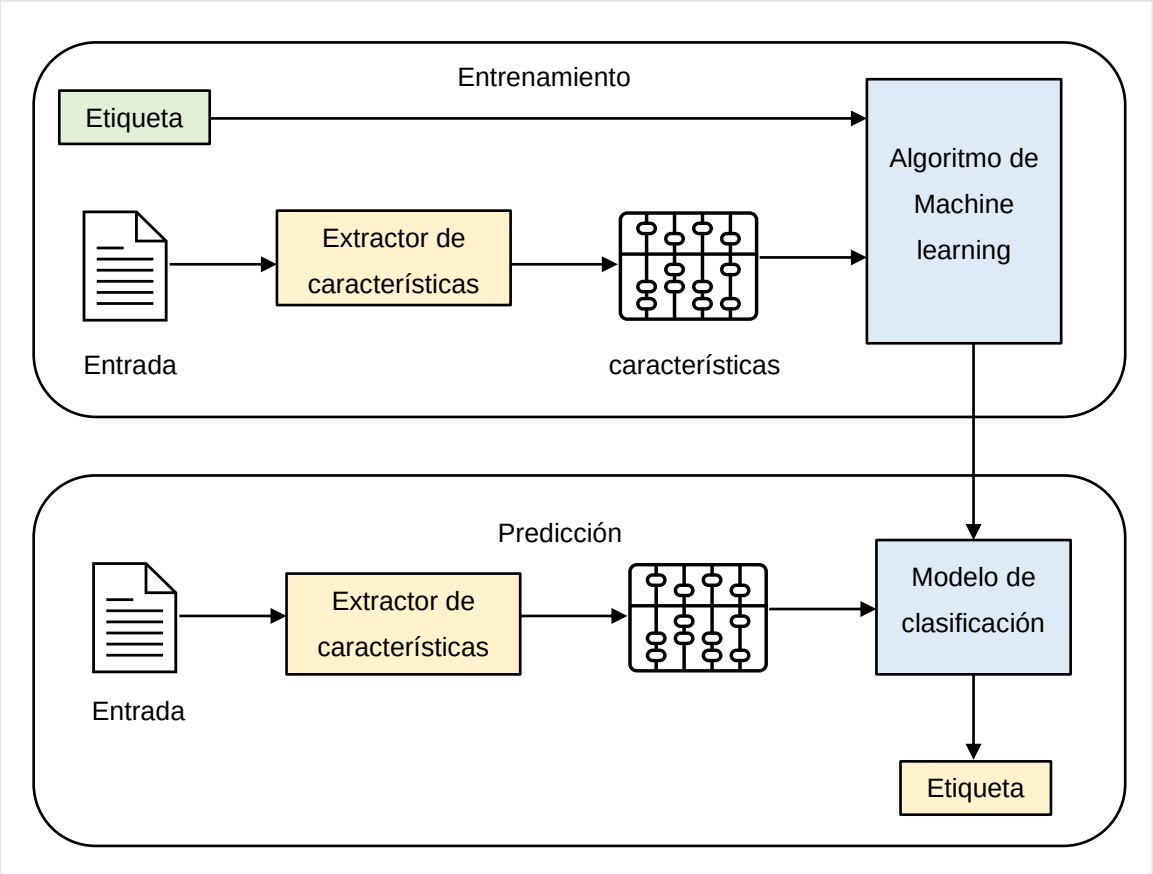
El *entrenamiento* es la fase en la que se le indica al modelo los ejemplos etiquetados para que este aprenda la relación que existen entre el atributo y la etiqueta.

La *inferencia* es la fase en la que se le muestra al modelo ejemplos sin etiqueta que nunca han sido vistos con el objetivo de que realice la inferencia de la etiqueta.

Como se observa en el ejemplo mencionado anteriormente, es un ejemplo de clasificación; es decir, dado un comentario identificar si este es positivo o negativo. Gran parte de las aplicaciones de minería de texto son problemas de clasificación, para resolver estos problemas generalmente se utilizan modelos de aprendizaje supervisado. Por otro lado, se tienen los problemas de agrupamiento (clustering), perfilado o profiling, agrupamientos de coocurrencias, entre otros, para los cuales se utilizan modelos de aprendizaje no supervisado. En la figura 6 se muestra un proceso general del aprendizaje automático supervisado aplicado a la clasificación.

Figura 6

Marco general para el aprendizaje automático supervisado aplicado a la clasificación



Nota. Adaptado de *Natural Language Processing with Python* [Diagrama de flujo], por Bird et al., 2009, (https://www.nltk.org/book/). CC BY 3.0.

El presente trabajo consta de un problema de clasificación, por este motivo en el siguiente punto se analizarán las principales técnicas de aprendizaje automático supervisado que se utilizan para la clasificación.

2.5.1. Clasificador Naïve Bayes (Naïve Bayes Classifier)

Los modelos Naïve Bayes son utilizados en aplicaciones de clasificación de documentos con el propósito de asignar automáticamente documentos electrónicos individuales a una o más categorías en función de sus contenidos.

Pitigala et al. (2011) dicen que los modelos de clasificación Naïve Bayes se basan en el modelo de probabilidad posterior, además dicen que es un método rápido y robusto para la clasificación de texto. Dado un documento (d), ¿su probabilidad de pertenecer a una clase

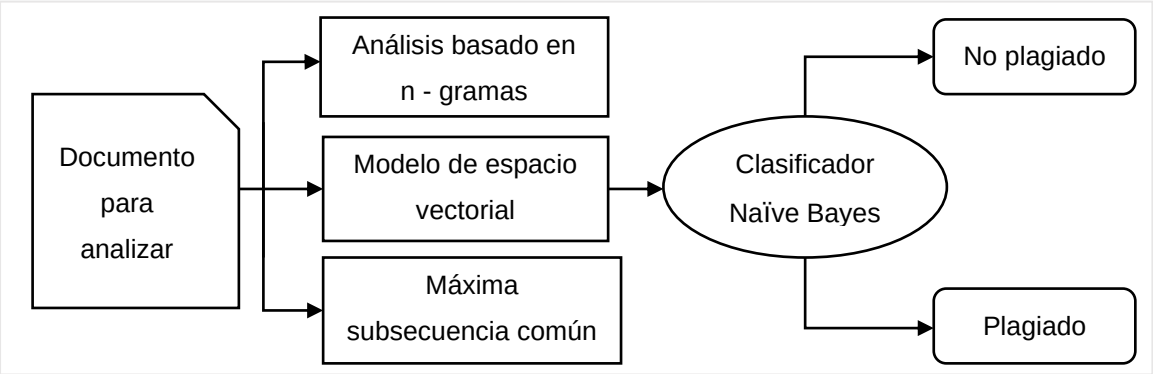
(c) es?. El objetivo de la clasificación de Naïve Bayes, es encontrar la clase óptima para un documento dado, es decir, la clase que ofrece la máxima probabilidad posterior.

Allahyari et al. (2017) dicen que el modelo Naïve Bayes es quizás el clasificador más simple y más utilizado. Modela la distribución de documentos en cada clase usando un modelo probabilístico, asumiendo que la distribución de diferentes términos son independientes entre sí. Dentro de la clasificación de texto, existen dos modelos de clasificadores Naïve Bayes. El primero es el Modelo Bernoulli Multivariado (Multi - Variate Bernoulli Model) en el cual un documento está representado por un vector de características binarias que indican la presencia o ausencia de palabras en el documento. El otro, es el Modelo Multinomial (Multinomial Model) en el que previamente un documento tiene que ser representado como una bolsa de palabras (Bag of Words) y el modelo se encarga de capturar la frecuencia de las palabras o términos. El objetivo de ambos modelos es encontrar la probabilidad posterior de una clase basada en la distribución de las palabras en el documento. La diferencia entre los dos modelos es que en el Modelo Bernoulli Multivariado no se toma en cuenta la frecuencia de las palabras; mientras que, en el Modelo Multinomial sí.

Un ejemplo se encuentra en el artículo de Reyes *et al*. (2014) en el cual se utiliza un modelo de clasificación bayesiana para determinar posibles plagios en textos científicos. En la figura 7 se presenta el flujo de su propuesta de solución.

Figura 7

Propuesta de solución



Nota. Adaptado de *Técnica de clasificación bayesiana para identificar posible plagio en información textual* [Diagrama de flujo], por Reyes et al., 2014, (https://bit.ly/2ViGoKw).

En la tabla 2 se muestran los resultados obtenidos del clasificador Naïve Bayes.

Tabla 2

Resultados de la ejecución del algoritmo Naïve Bayes

Parámetro	Valor
Porciento de instancias correctamente clasificadas	97.67%
Porciento de instancias incorrectamente clasificadas	2.34%
Promedio de verdaderos positivos (TP)	0.98
Promedio de falsos positivos (FP)	0.01

Nota. Adaptado de *Técnica de clasificación bayesiana para identificar posible plagio en información textual*, por Reyes et al., 2014, (https://bit.ly/2ViGoKw).

2.5.2. Clasificador Vecino Más Cercano (Nearest Neighbor Classifier or K-NN)

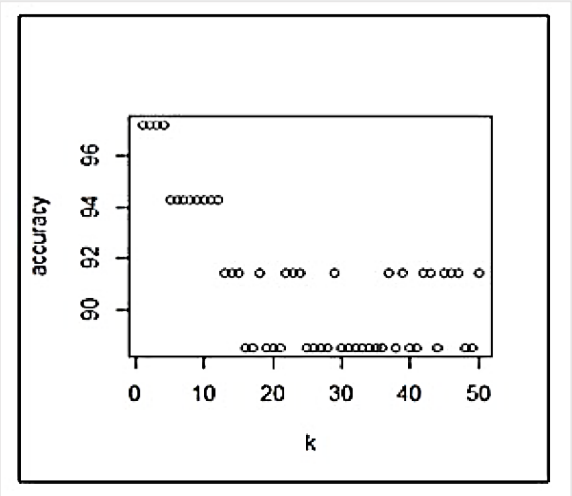
El clasificador vecino más cercano es un clasificador basado en la proximidad, que utiliza medidas basadas en la distancia para realizar la clasificación. Allahyari *et al*. (2017). Así mismo, Manning *et al*. citados por Godoy, (2017) mencionan que en el clasificador vecino más cercano cuando una nueva instancia necesita ser clasificada, a esta se la compara con los ejemplos existentes, para ello usa una métrica de distancia y los ejemplos que estén más próximos son utilizados para asignar la clase a la que pertenece la nueva instancia.

Yu & Song (2010), dicen que para la aplicación del clasificador vecino más cercano, el texto tiene que estar representado en la forma intermedia de Modelo de espacio Vectorial (Vector Space Model) además mencionan que es un modelo muy efectivo para la categorización de texto. Al generar un vector de características para un texto desconocido, el modelo KNN buscará en todos los ejemplos de entrenamiento y comparará la similitud de los vectores de características con el objetivo de encontrar el ejemplo de entrenamiento K más cercano y finalmente asignar el texto desconocido al vecino K más cercano con mayor valor de clasificación.

Un ejemplo de K-NN para la clasificación automatizada de documentos textuales en categorías predefinidas se propone en el artículo de Moldagulova y Sulaiman (2017).

Figura 8

Tendencia de precisión de la clasificación en K de 1 a 50



Nota. Recuperado de *Using KNN algorithm for classification of textual documents* [Imagen], por Moldagulova & Sulaiman, 2017, (https://doi.org/10.1109/ICITECH.2017.8079924). IEEE.

En la figura 8, se observa los resultados de precisión obtenidos en valores de K de 1 a 50. El peor caso de precisión es del 88% aproximadamente. Con esto se puede deducir que, mientras más bajo es el valor de K, mejor es la precisión del clasificador.

2.5.3. Clasificadores de árbol de decisión (Decision Tree classifiers)

Los árboles de decisión son reglas especiales de decisión que se organizan en una estructura de árbol. Un árbol de decisión divide el espacio del documento en regiones no superpuestas en sus hojas, y se hacen predicciones en cada hoja. (Weiss *et al*., 2010)

Russell y Norvig (2004) dicen que en un árbol de decisión se toma como entrada un objeto o una situación descrita, la que es representada como un conjunto de atributos, para finalmente devolver una decisión. Una decisión, es el valor previsto de la salida dada la entrada.

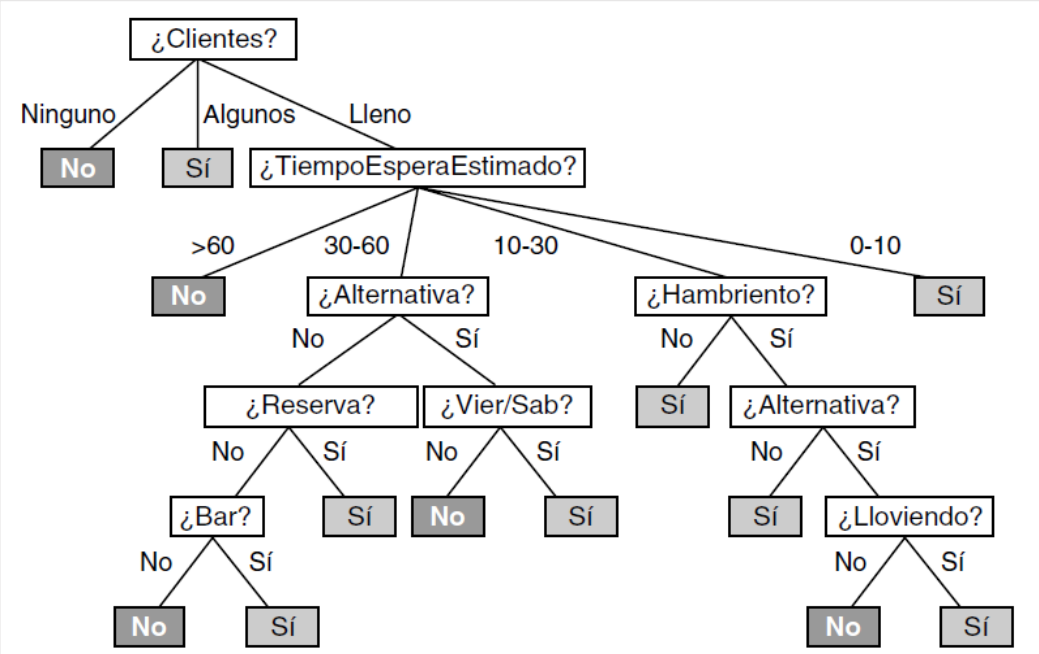
Allahyari *et al*. (2017), dicen que en el caso de los datos textuales, las condiciones en los nodos del árbol de decisión se definen comúnmente en términos de términos en los documentos de texto. Por ejemplo; un nodo puede subdividirse en otros nodos (hijos) dependiendo de la presencia o ausencia de un término particular en el documento.

Yu & Song (2010), mencionan dos fases principales en los clasificadores de árboles de decisión; la primera, es la fase de crecimiento del árbol, en la que el algoritmo empieza con todo el conjunto de datos en el nodo raíz y mediante algún criterio de división el conjunto de datos se divide en subconjuntos. La segunda fase, es la fase de poda, en la cual el árbol adulto se corta para evitar un ajuste excesivo y mejorar su precisión.

Un ejemplo sencillo de cómo funcionan los árboles de decisión para algo cotidiano de la vida como esperar por una mesa en un restaurante se encuentra en el libro “Inteligencia artificial: un enfoque moderno” de Russell y Norvig (2004).

Figura 9

Árbol de decisión para decidir si esperar por una mesa



Nota. Recuperado de *Inteligencia Artificial: Un enfoque moderno (p. 745)* [Mapa conceptual] , por Russell y Norvig, 2004. Pearson Educación, S.A.

Pranckevičius y Marcinkevičius (2017), proponen un estudio comparativo de modelos de clasificación en documentos textuales demostrando que los árboles de decisión tiene una precisión baja en comparación con otros modelos como el Naïve Bayes y las Maquinas de Vectores de Soporte analizados en este documento.

2.5.4. Máquinas de vectores de soporte (Support Vector Machines)

Las máquinas de vectores de soporte (SVM) son algoritmos de aprendizaje supervisado que han sido utilizados ampliamente para problemas de clasificación de datos. Suthaharan dice que SVM utiliza un modelo matemático simple y lo manipula para permitir la división lineal del dominio, además menciona que la máquina de vectores de soporte se puede dividir en modelos lineales y no lineales. Se llama máquina de vector de soporte lineal si el dominio de datos se puede dividir linealmente (por ejemplo, línea recta o hiperplano) para separar las clases en el dominio original. Si el dominio de datos se puede transformar en un espacio llamado espacio de características, entonces se llama máquina de vectores de soporte no lineal. Matemáticamente, el modelado de una máquina de soporte lineal adopta la ecuación lineal:

$$y = wx' + \gamma$$

y el modelo de una máquina de vector de soporte no lineal adopta la ecuación no lineal:

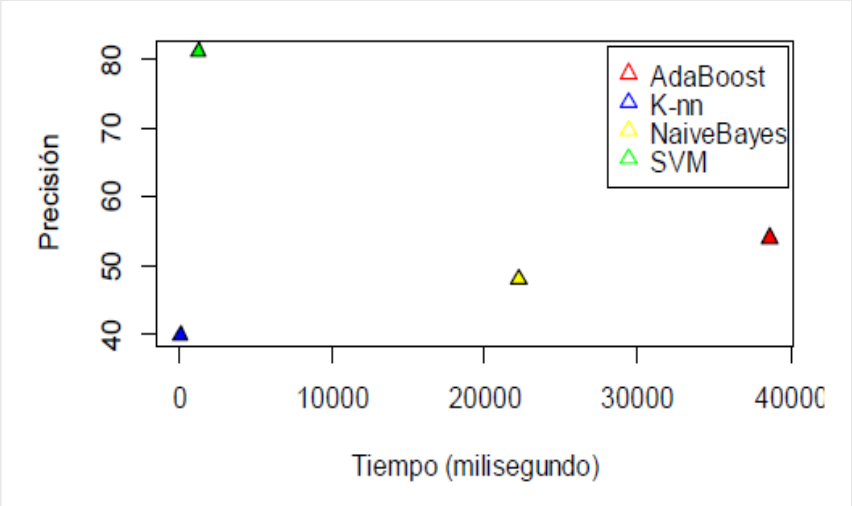
$$y = w\emptyset(x') + \gamma$$

Haykin citado por Shafiabady et al. (2016), menciona que SVM posee una sobresaliente capacidad de generalización y que esta es aportada por la implementación del principio de minimización del riesgo estructural (SRM) que consiste en encontrar un hiperplano de separación óptimo que garantiza el error de clasificación más bajo.

En el artículo de Castellanos *et al.* (2017), desarrollan una biblioteca digital en la cual implementan SVM para la clasificación de documentos y demuestran su precisión en comparación con otros clasificadores como Naïve Bayes y K-NN.

Figura 10

Precisión de los clasificadores en términos de precisión y tiempo



Nota. Recuperado de *Biblioteca digital con técnicas de clasificación automática de documentos* [Imagen], por Castellanos et al., 2017,

En la figura 10, se observa que el mejor clasificador es SVM con un promedio del 80% de precisión en la clasificación de documentos textuales. También ocupa el segundo lugar en términos del tiempo que se demora el modelo en la clasificación.

2.5.5. Redes neuronales artificiales (Artificial Neural Network)

Las redes neuronales artificiales son modelos de aprendizaje automático que tratan de simular el comportamiento de las neuronas biológicas. Para definir lo que es una red neural artificiales primero hay que definir lo que es una neurona. Dreyfus (2005), define a la neurona artificial como una función no lineal parametrizada y acotada. Mientras que una red de neuronas es la composición de las funciones no lineales de dos o más neuronas. La función de la neurona viene dada por:

$$y = f(x_1, x_2, \dots, ; w_1, w_2, \dots, w_p)$$

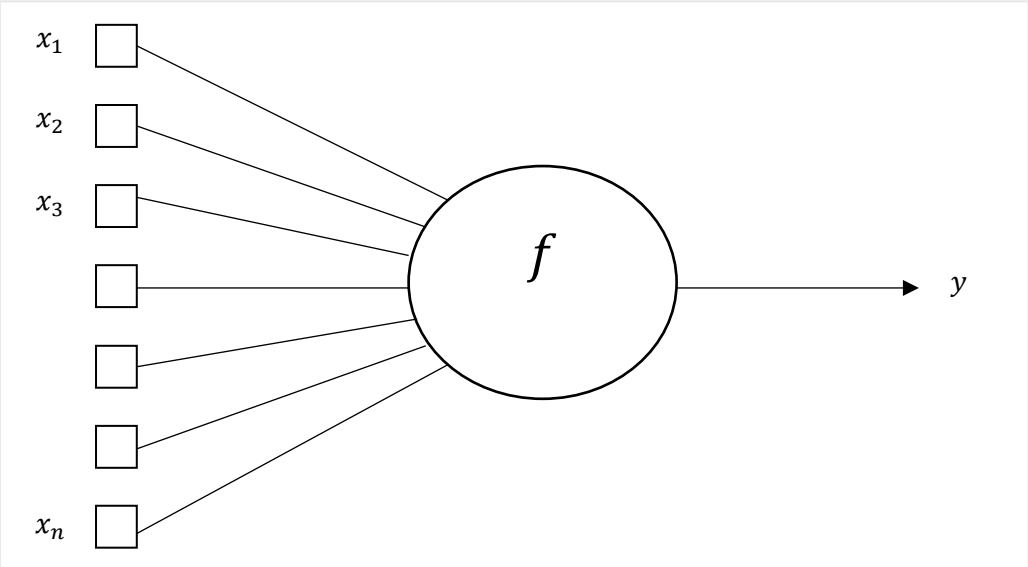
donde x_i son las variables y w_i son los parámetros de la neurona.

Los parámetros se asignan a las entradas de las neuronas. La salida de la neurona es una combinación no lineal de las entradas $\{x_i\}$, ponderada por los parámetros $\{w_i\}$, que a menudo se denominan pesos. La función f se denomina función de activación. Es aconsejable que la función de activación sea una función sigmoidea.

Una neurona representada gráficamente se muestra en la figura 11.

Figura 11

Estructura de una neurona artificial.

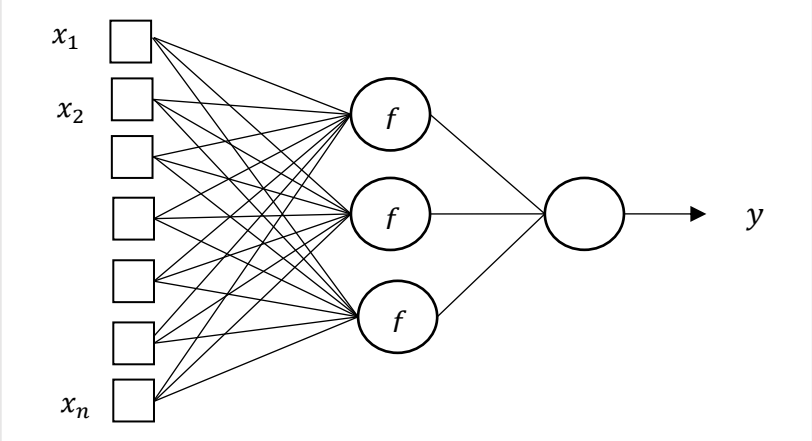


Nota. Adaptado de *Neural Networks: Methodology and Applications (p. 2)* [Imagen], por Dreyfus, 2005, (https://doi.org/10.1007/3-540-28847-3). Springer-Verlag Berlin

Una red neuronal representada gráficamente se encuentra en la figura 12.

Figura 12

Estructura de una red neuronal con una capa oculta



Nota. Adaptado de *Inteligencia Artificial: Un enfoque moderno (p. 847)* [Imagen], por Russell y Norvig, 2004. Pearson Educación, S.A.

Russell & Norvig, (2004), mencionan que la idea para el aprendizaje de las redes neuronales es ajustar los pesos de la red para así minimizar alguna medida del error que se

produce con el conjunto de entrenamiento. La medida del error clásica es la suma de los errores cuadrados o también conocido como el error cuadrático medio. Uno de los problemas de las redes neuronales artificiales es que cuando existen demasiados parámetros en el modelo, las redes neuronales artificiales son sujeto de sobreajuste (overfitting). El enfoque para evitar el sobreajuste es intentar varias veces y quedarnos con el mejor modelo. Existen redes neuronales de una capa y multicapa. En redes multicapa totalmente conectadas como la de la figura 12, tenemos que preocuparnos por el número de capas ocultas y por su tamaño es decir la cantidad de neuronas de la capa.

Feldman & Sanger, (2009) citados por Godoy, (2017), dicen que en los clasificadores de texto que están basados en redes neuronales artificiales la capa de entrada está formada por unidades que representan los términos existentes en los documentos, mientras que en las capas ocultas y de salida están representadas las categorías de interés. En los problemas de clasificación, el número de neuronas de la capa de salida está dado por la cantidad de categorías en las que se quieren clasificar los datos de entrada.

Un ejemplo se encuentra en el artículo de García *et al*. (2018), en el cual utilizan una red neuronal artificial y el modelo Naïve Bayes para clasificar tweets (ver tabla 3).

Tabla 3

Exactitud y error promedio por ejecución

	Clasificador Bayesiano		**Red Neuronal**	
Ejecución	**Exactitud**	**Error**	**Exactitud**	**Error**
1	0.842	0.157	0.917	0.082
2	0.842	0.157	0.915	0.084
3	0.841	0.158	0.922	0.077
4	0.841	0.158	0.919	0.080
5	0.848	0.151	0.920	0.079
6	0.842	0.157	0.919	0.080

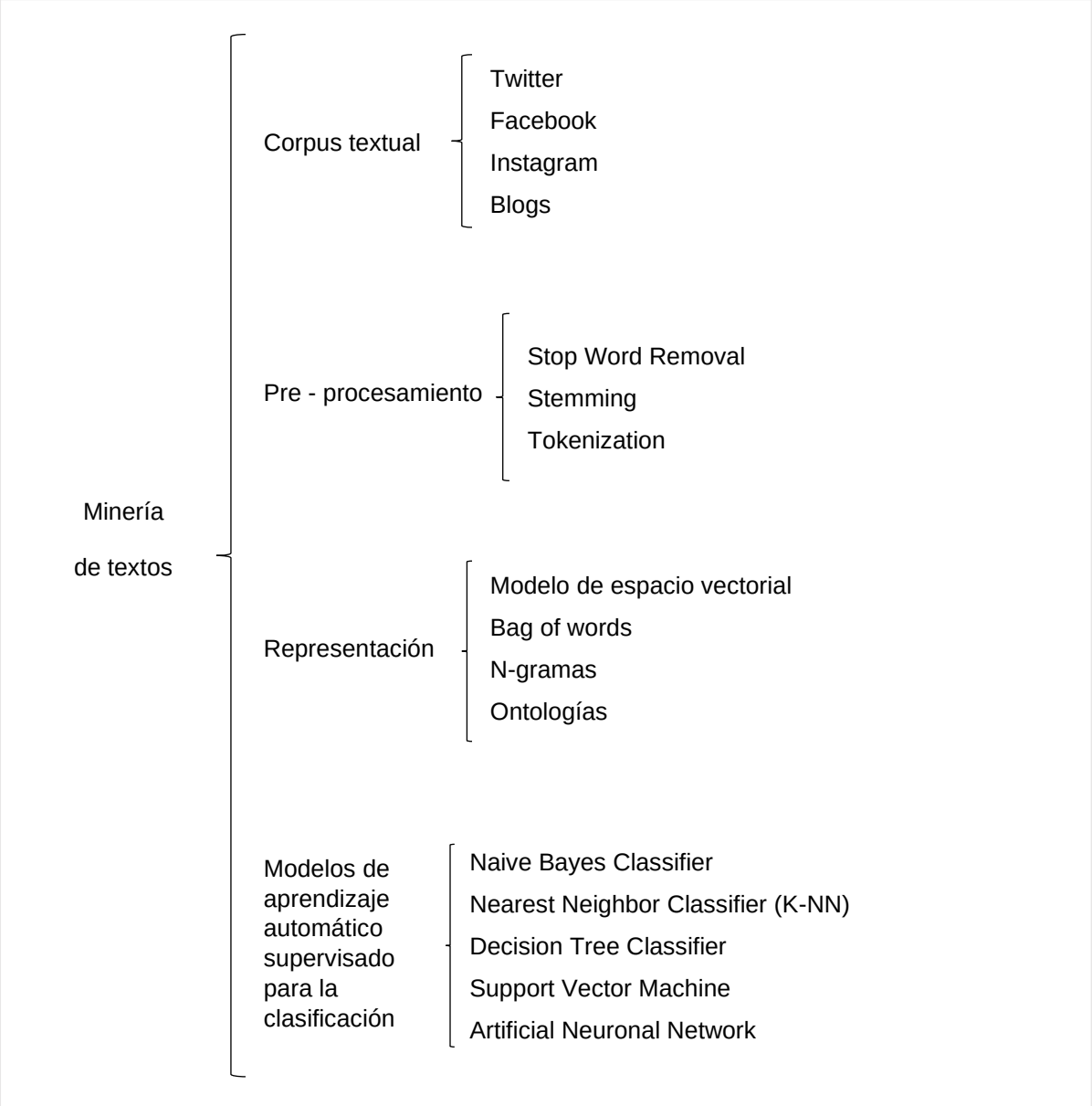
Nota. Adaptado de *Aplicación de una red neuronal artificial para la clasificación automática de tuits en español*, por García *et al.* 2018. (https://bit.ly/2VlC6C9)

Los resultados indican que las redes neuronales tienen mejor precisión y una menor tasa de error en todas las ejecuciones de la clasificación de los tweets en comparación con el modelo de Naïve Bayes.

La figura 13, representa un diagrama resumen del proceso a seguir para resolver problemas de clasificación de corpus textuales obtenidos de redes sociales.

Figura 13

Procesamiento de textos



Nota. Principales métodos para resolver problemas de clasificación de textos de redes sociales. [Diagrama de llaves].

2.6. Aplicaciones de la minería de texto

Las aplicaciones de la minería de texto son muy diversas: resúmenes de documentos, síntesis de información, filtrado de emails, personalización de perfiles web y publicaciones digitales, análisis de redes sociales, detección de comunidades web, extracción de ideas principales, obtención de principales tópicos de discusión de artículos científicos, detección de plagios, organización bibliográfica y de autores, análisis de patentes, sistemas inteligentes de mercado vía web, sistemas de detección automática de comportamiento antisocial, realización de pronósticos, entre otras. Una explicación detallada de algunas de estas aplicaciones se encuentra en Justicia de la Torre, (2017).

En la tabla 4 se presenta un resumen de los diferentes usos de la minería de texto clasificados por área de aplicación.

Tabla 4

Aplicaciones de la minería de texto.

Área de aplicación	Uso
Web Semántica	Servicios web de descubrimiento Aprendizaje de ontologías
Redes Sociales	Filtrado de Email Personalización de perfiles web Detección de Comunidades Web Teorías Sociales
Opinion Mining, Sentimental Analysis	Clasificación de Opiniones Hotspots en foros Predicciones Detección de Comportamiento Antisocial Encuestas de Opinión
Síntesis, Organización	Summarization Revisiones sistemáticas Obtención de Titulares Extracción de Ideas útiles y nuevas Discusión de temas principales Organización de documentos detección de plagios Búsqueda de contradicciones

Área de aplicación	Uso
Minería de Fuentes Abiertas, Tesauros	Identificación de temas Exploración de tesauros
e-Commerce	Toma de decisiones
Marketing	Localización de "trozos" de información Análisis de mercado
e-Learning	Herramientas colaborativas
Help Desk	Generación de casos modelo Detección de fallos

Nota. Adaptado de *Nuevas Técnicas de Minería de Textos: Aplicaciones,* por Justicia de la Torre, 2017. (https://digibug.ugr.es/handle/10481/46975). CC BY 3.0

2.6.1. Análisis de sentimientos

El análisis de sentimientos o minería de opinión resulta ser una tarea desafiante pero útil, por ejemplo; las empresas y organizaciones siempre quieren conocer las opiniones que los consumidores tienen acerca de sus productos o servicios, de la misma forma, los clientes también quieren conocer las opiniones de otros usuarios antes de comprar un producto o usar un servicio.

Las empresas podrían destinar personal para que se encargue de leer y analizar todas las opiniones y resulta ser una tarea sencilla ya que los seres humanos poseemos la capacidad de interpretar la sintaxis y semántica de las oraciones. Pero esto se vuelve inviable cuando la cantidad de información es exageradamente grande. Por esta razón surge la necesidad de automatizar este proceso mediante hardware y software que es capaz de realizar cálculos y procesar datos en cuestión de segundos.

No es una tarea sencilla para las computadoras ya que estas no tienen la capacidad de interpretar el lenguaje oral o escrito de los humanos. Pero esto ha ido cambiando con el paso del tiempo y gracias a los avances tecnológicos hoy en día se puede dotar a las computadoras de cierta inteligencia que permite la interpretación del lenguaje.

El análisis de sentimientos o la minería de opinión para Liu & Zhang (2012), es el estudio computacional de las opiniones, valoraciones, actitudes y emociones de las personas hacia entidades, individuos, problemas, eventos, temas y sus atributos.

Una definición muy parecida la dan Medhat *et al*. (2014), dicen que el análisis de sentimientos o la minería de opinión es el estudio computacional de las opiniones, actitudes y emociones de las personas hacia una entidad. La entidad puede representar individuos, eventos o temas.

Bravo *et al*. (2014), mencionan que el análisis de sentimientos es un problema de clasificación cuyos enfoques actuales se centran principalmente en una dimensión de opinión popular que es muy difícil de clasificar de forma independiente y para ello proponen una taxonomía basada en tres métodos: Polaridad (Polarity), Fuerza (Strength) y Emoción (Emotion).

- *Polaridad (Polarity).* – Los métodos orientados a la polaridad tienen como objetivo extraer información de polaridad de un pasaje de texto y normalmente devuelven una variable categórica cuyos valores posibles son: positivos, negativos y neutros.
- *Fuerza (Strength).* – Los métodos orientados a la fuerza devuelven puntajes numéricos que indican la intensidad de los sentimientos positivos y negativos expresados en un pasaje de texto.
- *Emoción (Emotion).* – Estos métodos se centran en extraer emociones o estados de ánimo de un pasaje de texto. Las categorías en las que se puede clasificar un mensaje utilizando un método orientado a la emoción son: tristeza, alegría, sorpresa, entre otras.

Según Baviera (2017), el análisis de sentimientos aplicado a grandes volúmenes de datos tiene una gran dificultad ya que evaluar el tipo de emoción o la polaridad no siempre tiene la misma interpretación incluso si el proceso es realizado manualmente, los resultados varían según la persona que los analiza. Además menciona que el desarrollo de técnicas de análisis de sentimientos aplicadas al español aún se encuentra en fase de maduración.

2.7. Herramientas para la minería de texto

En la actualidad existe una gran variedad de herramientas software que han sido desarrolladas por gigantes tecnológicos. A continuación se describen las principales.

- *MonkeyLearn:* en la página oficial dan a conocer su software mediante el eslogan; “Capacite modelos personalizados de aprendizaje automático para obtener temas, sentimientos, intenciones, palabras clave y más” (MonkeyLearn, 2020)

  Esta es una herramienta de pago orientada a pequeñas, medianas y grandes empresas. Ofertan planes de 300$ y 999$ por año.
- *IBM Watson:* IBM Watson tiene un gran número de herramientas de pago desarrolladas con inteligencia artificial y aprendizaje automático. Watson Natural Language Understanding (https://www.ibm.com/cloud/watson-natural-language-understanding) es una herramienta que ofrece el servicio de procesamiento de lenguaje natural para analizar textos. Watson Natural Language Classifier (https://www.ibm.com/cloud/watson-natural-language-classifier) es una herramienta que permite personalizar modelos de aprendizaje automático para analizar y etiquetar datos textuales.
- *Amazon Comprehend:* El gigante tecnológico Amazon también ofrece herramientas para el análisis y procesamiento de corpus textuales. En su sitio web oficial (https://aws.amazon.com/es/comprehend/) mencionan que

  > Amazon Comprehend utiliza el aprendizaje automático para ayudar a descubrir la información y las relaciones en sus datos no estructurados.
  >
  > El servicio identifica el idioma del texto; extrae frases, nombres de lugares, personas, marcas o eventos clave; comprende el grado de positividad o negatividad del texto; lo analiza mediante tokenización y categorías gramaticales; y organiza automáticamente una colección de archivos de texto por tema (amazon, 2020)

  Igual que las herramientas anteriores, esta también es de pago y sus precios varían de acuerdo a la cantidad de texto que se desea procesar.

- *Google Cloud NLP:* Otro gigante de la tecnología como lo es Google también ofrece herramientas que permiten el análisis de textos. En su sitio web oficial se destaca que:

  > Natural Language usa el aprendizaje automático para revelar la estructura y el significado del texto.
  >
  > Puedes extraer información sobre personas, lugares y eventos, y comprender mejor las opiniones de las redes sociales y las conversaciones de los clientes. Natural Language te permite analizar texto y, también, integrarlo a tu almacenamiento de documentos en Cloud Storage. (Google, 2020)

  Los precios varían dependiendo de las funcionalidades que se usen.

Existen otras empresas que han desarrollado herramientas como Aylien, Thematic, MeaningCloud, Lexalytics entre otras. La gran mayoría de las herramientas mencionadas en este apartado son de pago y están orientadas a las PYMES y las grandes empresas. Aunque vale destacar que también ofrecen versiones o planes gratuitos con el objetivo de poder probar sus productos.

Por otro lado, se tiene la opción de desarrollar una aplicación propia para el análisis de textos. Para ello existen librerías de programación que permiten realizar el pre - procesamiento, la representación y el descubrimiento de conocimiento.

Entre otras, las principales herramientas de programación para el análisis de textos son:

- *Natural Language Toolkit (NLTK):* Es un conjunto de herramientas que permiten la escritura de programas en Python. En su sitio web oficial se menciona que NLTK está conformado por un gran conjunto de bibliotecas las que permiten realizar tokenización, derivación, etiquetado, procesamiento de texto para la clasificación, análisis y razonamiento semántico (NLTK Project, 2020).
- *TextBlob Simplified Text Processing:* Es una biblioteca de Python para el procesamiento de datos textuales. En su sitio web oficial indican que la librería “proporciona una API simple para sumergirse en tareas comunes de

Procesamiento del Lenguaje Natural (PNL) como el etiquetado de parte del discurso, extracción de frases nominales, análisis de sentimientos, clasificación, traducción y más" (TextBlob, 2020).

- *Scikit-learn:* es una biblioteca de Python para el aprendizaje automático. En su sitio web oficial describen que Scikit-learn cuenta con algoritmos para el aprendizaje supervisado como; modeles lineales, máquinas de vectores de soporte, k-vecinos más cercanos, naïve bayes, árboles de decisión, modelos de redes neuronales, entre otros (scikit-learn, s/f).
- *TensorFlow*: En su sitio web oficial destacan que tensorFlow es la principal biblioteca de código abierto para la construcción de modelos de aprendizaje automático. Cuenta con librerías para el procesamiento de textos y está disponible para los lenguajes de programación Python y JavaScript; así como, para dispositivos móviles y de IOT (TensorFlow, s/f).

Existen muchas más librerías de programación para el procesamiento y análisis de datos textuales como; Gensim, CoreNLP, spaCy, Pattern, VADER, Scikit-learn, PyNLPI.

Para finalizar esta sección, véase la tabla 5 en la cual se presenta un resumen comparativo de las herramientas de minería de texto ofertadas por las más grandes empresas tecnológicas.

Tabla 5

Herramientas para la minería de texto

Herramienta	Empresa	Que ofrece	Precio
MonkeyLearn	MonkeyLearn	Agrupación temática. Análisis de sentimientos. Análisis de texto. Clasificación taxonómica. Consultas booleanas. Detección del idioma. Etiquetado. Filtrado de documentos. Modelado predictivo. Presentación gráfica de datos. Resumen.	Para equipos $299 al mes. Para negocios $999 al mes.
IBM Watson	IBM	Análisis de texto. Extracción de metadatos de contenido. Categorización de contenido. Identificación de conceptos de alto nivel. Análisis de emociones. Análisis de sentimientos.	El precio se calcula de acuerdo a las unidades de datos que queramos procesar. Una unidad de datos es un grupo de 10000 caracteres o menos. $0.003 es el costo por unidad de dato. Hay que recalcar que el costo por unidad varía de acuerdo al servicio que necesitemos.
Amazon Comprehend	Amazon	Extracción de frases claves. Análisis de opiniones. Análisis sintáctico. Reconocimiento de entidad. Detección de idioma Clasificación personalizada. Modelado de tema. Compatibilidad con varios idiomas. Comprensión de datos médicos.	El precio se calcula de acuerdo con las unidades que necesitamos procesar. Una unidad corresponde a 100 caracteres. $0,0001 es el costo por unidad. Hay que recalcar que el costo por unidad varía de acuerdo al servicio que necesitemos.

Herramienta	Empresa	Que ofrece	Precio
Google Cloud NLP	Google	Análisis sintáctico. Análisis de entidades. Extracción de entidades personalizadas. Análisis de opiniones. Clasificación de contenido. Multilingüe. Información sobre la estructura espacial. Asistencia para conjuntos de datos grandes.	El precio varía de acuerdo a las unidades que necesitamos procesar y al servicio que necesitemos. Una unidad corresponde a 1000 caracteres o menos. Por ejemplo el precio para el análisis de opiniones en textos de 5001 a 1000000 caracteres cuesta $1.00.

Nota. Comparativa de herramientas disponibles en el mercado para minería de texto.

2.8. Trabajos relacionados

En este apartado se presentan algunos trabajos de investigación que utilizan técnicas de aprendizaje automático supervisado para realizar análisis de sentimientos en corpus textuales en español, específicamente corpus textuales de redes sociales.

Para la selección de los trabajos se realizó una búsqueda avanzada en la biblioteca virtual de la Universidad Técnica Particular de Loja. Las cadenas de búsqueda que se utilizaron fueron: análisis AND sentimientos AND redes AND sociales.

Los criterios de inclusión que se utilizaron fueron:

- Estudios realizados desde el año 2017 hasta el presente.
- Estudios cuyo idioma fuera el español.
- Estudios que pertenezcan a la categoría Academic Journals.
- Estudios más citados por otros investigadores.

2.8.1. TR01: Análisis de sentimientos de tweets en español utilizando SVM y CNN. (Rosa et al., 2017)

En este trabajo se presenta un clasificador híbrido de sentimientos de polaridad para tweets, el cual fue puesto a prueba en la competición SEPLN (TASS). El TASS se celebra desde el 2012 y es una propuesta de la Sociedad Española para el Procesamiento del

Lenguaje Natural (SEPLN). Para el desarrollo de su trabajo contaron con un aproximado de 8000 tweets y lo particionaron en: 85% para el entrenamiento y 15% para el desarrollo, (véase tabla 6). Luego de un preprocesamiento de los tweets realizaron la clasificación utilizando tres enfoques: Primero utilizaron un clasificador basado en máquinas de vectores de soporte (SVM), luego un clasificador basado en redes neuronales convolucionales (CNN) y finalmente realizaron una clasificación utilizando una combinación de estos dos algoritmos. Los resultados de estos experimentos se presentan en la tabla 7.

Tabla 6

Tamaño de los corpus utilizados

	Entrenamiento	**Desarrollo**
P	2782 (39 %)	576 (36 %)
N	2295 (32 %)	524 (33 %)
NEU	721 (10 %)	151 (10 %)
NONE	1346 (19 %)	338 (21 %)
Total	7114	1589

Nota. Adaptado de RETUYT in TASS 2017: Sentiment Analysis for Spanish Tweets using SVM and CNN, por Rosa et al., 2017, (http://arxiv.org/abs/1710.06393)

Tabla 7

Resultados sobre el corpus de validación

Clasificación	**M-F1**	**Acierto**
svm	49.9	61.7
cnn4	50.2	64.1
svm_cnn	50.9	64.7

Nota. Adaptado de RETUYT in TASS 2017: Sentiment Analysis for Spanish Tweets using SVM and CNN, por Rosa et al., 2017, (http://arxiv.org/abs/1710.06393)

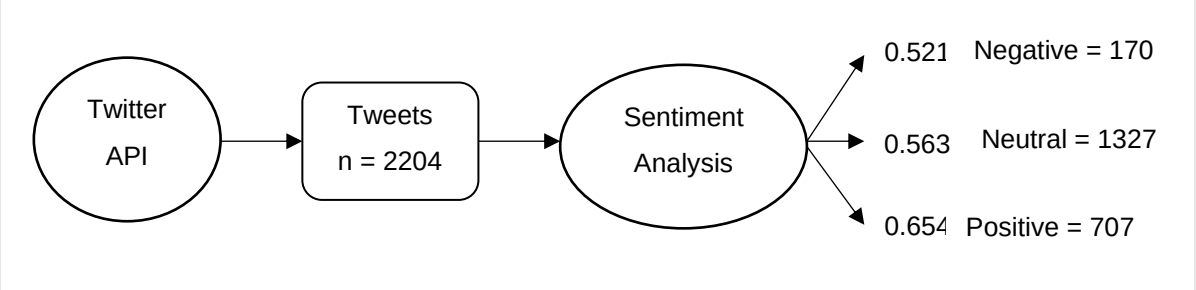
Como se observa en los resultados de la tabla 7, el clasificador híbrido es el que presenta una mejor taza de acierto por encima del clasificador basado en máquinas de vectores de soporte (SVM) y el clasificador basado en redes neuronales convolucionales (CNN).

2.8.2. TR02: Un Análisis de Sentimiento en Twitter con Machine Learning: Identificando el sentimiento sobre las ofertas de #BlackFriday. (Saura et al., 2018)

Ellos realizan un trabajo el cual consiste en medir los sentimientos de los consumidores que utilizan la red social Twitter en relación con el evento #BlackFriday que se celebra cada año. Para ello trabajaron con un corpus de aproximadamente 2200 tweets y utilizaron específicamente la librería MonkeyLearn que contiene algoritmos de para la clasificación de sentimientos. Después del análisis obtuvieron que el 60.2% de los tweets son categorizados como neutros, el 32.1% como tweets positivos y el 7.7% como negativos. En la figura 14 se muestra el proceso del desarrollo metodológico que implementaron.

Figura 14

Desarrollo del proceso metodológico



Nota. Adaptado de *Un Análisis de Sentimiento en Twitter con Machine Learning: Identificando el sentimiento sobre las ofertas de #BlackFriday*, por Saura et al., 2018 (https://bit.ly/37lu1Db). revistaESPACIOS.com

2.8.3. TR03: Maternidad en Perú a través del uso del Sentiment Analysis en Facebook. (Seperak et al., 2019)

En el trabajo presentado por Seperak *et al*. (2019), se propone el uso del aprendizaje automático supervisado para conocer la percepción actual que tienen los peruanos sobre la

maternidad y la mujer. Contaron con un corpus textual de aproximadamente 28.000 comentarios que hacían relación a la temática extraído de grupos abiertos al público general.

El conjunto de entrenamiento lo conformaron con alrededor de 500 comentarios positivos, 500 negativos y 500 neutros clasificados manualmente. Luego de la fase de preprocesamiento de los textos, utilizaron el algoritmo Conditional Random Fields para la clasificación. Los resultados se muestran en la tabla 8.

Tabla 8

Resultados de la clasificación

	Cantidad	**Porcentaje**
Comentarios positivos	20.858	72.16 %
Comentarios negativos	8.049	27.84 %
Total	28.907	100.00 %

Nota. Adaptado de *Maternidad en Perú a través del uso del Sentiment Analysis en Facebook*, por Seperak et al., 2019, (https://doi.org/10.4185/RLCS-2019-1370).

En la tabla 8, se puede observar que la mayoría de los comentarios acerca de la mujer y la maternidad en el Perú son positivos con un 70.16%, mientras que los comentarios negativos representan un 27.84% del total.

2.8.4. TR04: Análisis de sentimiento en Instagram: polaridad y subjetividad de cuentas infantiles (Arantxa & Aguaded, 2020)

Finalmente presentamos un trabajo muy actual de Arantxa y Aguaded en el cual se realiza un análisis de sentimientos en cuentas infantiles administradas por padres de la red social Instagram. Realizan una clasificación utilizando los métodos de polaridad y fuerza, es decir, aparte de determinar si un comentario es positivo o negativo a este se le asociado un peso, en donde un comentario muy negativo esta dado por (-1), un comentario muy positivo este dado por (+1) y (0) representa un comentario neutro. El corpus textual consta de 772 comentarios realizados en 100 fotos de 8 cuentas infantiles, 4 cuentas en el lenguaje inglés

y 4 en el lenguaje español. Los resultados que aquí presentamos son de las cuentas en español. Después de la fase de preprocesamiento de los textos utilizan la librería TextBlob para realizar la clasificación y los resultados se muestran en la tabla 9.

Tabla 9

Polaridad en cuentas infantiles en español

Cuenta	Número de entradas positivas	Número de entradas negativas	Número de entradas neuras	Media de polaridad positiva	Media de polaridad negativa	Valoración media de polaridad de la cuenta
1	46	4	22	0.43	-0.26	0.26
2	94	6	0	0.39	-0.17	0.36
3	80	7	13	0.37	-0.17	0.29
4	62	18	20	0.36	-0.18	0.19
Total	282	35	55	0.39	-0.19	0.27

Nota. Adaptado de *Análisis de sentimiento en Instagram: polaridad y subjetividad de cuentas infantiles*, por Arantxa y Aguaded, 2020, (https://doi.org/10.1387/zer.21454)

2.9. Comparación de trabajos relacionados

Para finalizar este capítulo véase la tabla 10, en la cual se presenta un resumen comparativo de los cuatro estudios analizados.

Tabla 10

Trabajos relacionados

	Objetivos	Herramientas para la obtención y procesamiento del texto	Algoritmos para la clasificación	Resultados	Inconvenientes
TR01	Construir un clasificador de sentimientos de tweets.	Twitter API. Scikit-learn.	Support Vector Machines (SVM). Convolutional Neural Networks (CNN). Híbrido entre SVM y CNN.	El clasificador con mejor desempeño es el que utiliza una combinación de SVM y CNN.	Debido a que la clase (tweets neutros) tiene pocos ejemplos en el corpus y a la similitud que tienen con los tweets negativos y positivos, el sistema es poco efectivo en la detección de estos.
TR02	Identificar el sentimiento sobre las ofertas de #BlackFriday.	Twitter API. MonkeyLearn.	Suppor Vector Machine	Tweets positivos: 32,1%. Tweets negativos: 7,7%. Tweets neutros: 60,2%.	El tamaño de la muestra y el número limitado de empresas que utilizan el hashtag #BlackFriday.

	Objetivos	Herramientas para la obtención y procesamiento del texto	Algoritmos para la clasificación	Resultados	Inconvenientes
TR03	Conocer cuál es la percepción y nuevos conceptos en la actualidad acerca de la maternidad en Perú.	Facebook API.	Conditional Random Fields	Comentarios con sentimientos positivos hacia la maternidad y a la imagen de madre: 72,16%. Comentarios negativos: 27,84%.	El proceso de extraer el corpus lo tuvieron que realizar dos veces ya que la primera vez se encontró que la mayoría de los resultados no correspondían al concepto “madre”.
TR04	Analizar la polaridad de comentarios publicados en fotos de cuentas infantiles gestionadas por padres en la red social Instagram.	TextBlob.	Naïve Bayes	Entradas positivas: 75,81%. Entradas negativas: 14,66%. Entradas neutras: 9,4%	Corpus textual limitado.

Nota. Tabla comparativa de los trabajos relacionados.

Capítulo tres

Metodología de desarrollo

En el presente capítulo se define la metodología que se va a aplicar para resolver la parte práctica de la siguiente investigación. En la sección 3.1. se plantean las fases de la metodología. En la sección 3.2 se definen las herramientas utilizadas. En la sección 3.3 se define la fase 1 de la metodología que consiste en indicar el propósito de la minería de texto. La sección 3.4 corresponde a la fase 2 de la metodología en la cual se indica el proceso de la recuperación de la información. Este capítulo finaliza con la sección 3.5 en la cual se describe la fase 3 de la metodología que corresponde al pre – procesamiento del texto.

3.1. Metodologías para la minería de texto

No existe un estándar metodológico para resolver problemas de minería de texto. Sin embargo, investigadores como Verma, Ranjan & Mishra (2015) citados en Barrera, (2016), proponen una metodología general que contempla dos fases; a) la fase de refinación del texto, en donde se transforman los documentos y se los representa en estructuras de datos; y b) la fase de destilación del conocimiento, en donde a partir de las estructuras de datos generadas en la primera fase, se identifican los patrones que servirán de base de conocimiento.

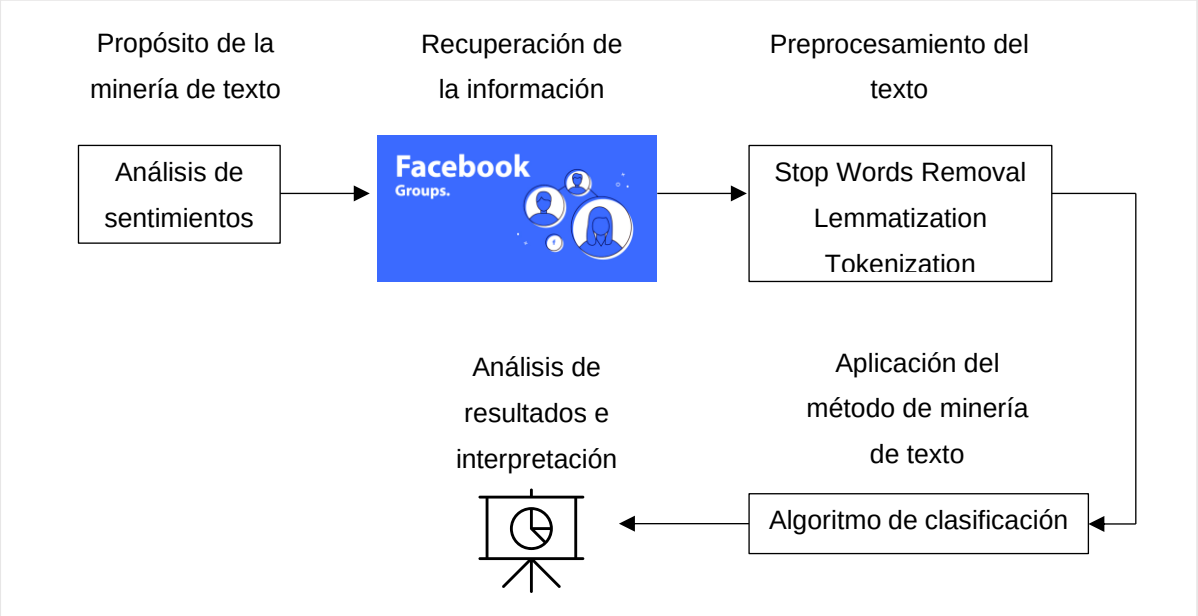
Por otro lado, M.Sukanya (2012), citado en el mismo estudio de Barrera, (2016), propone una metodología más específica en la cual el proceso para la resolución de problemas de minería de texto consta de varias fases. En la primera fase, se define el propósito de la minería de texto; en la segunda fase se realiza el proceso de recuperación de la información; en la tercera fase se realiza el preprocesamiento de los datos; en la cuarta fase se procede con la extracción de relaciones o patrones. Finalmente, como último paso se presentan los resultados para su posterior interpretación.

Para esta investigación se selecciona la metodología propuesta por M. Sukanya, es decir, la metodología más específica. Se hizo esta elección principalmente porque al dividir el problema en un conjunto de pasos, se tiene un mejor control del proceso de desarrollo lo

que facilita la obtención de resultados. En la figura 15 se muestra un diagrama de la metodología adaptada al contexto de esta investigación.

Figura 15

Metodología de desarrollo



Nota. Esta figura resume el proceso metodológico que se va a seguir para desarrollar la aplicación del presente trabajo de titulación [Diagrama de flujo].

3.2. Herramientas

Para desarrollar la parte experimental de la presente investigación se ha trabajado con el lenguaje de programación Python, específicamente con la versión 3.8.7. Se hizo esta elección porque Python es un lenguaje intuitivo, dinámico y fácil de interpretar lo cual permite resolver problemas más rápido que en otros lenguajes. Además cuenta con un sinnúmero de librerías y grandes comunidades que dan soporte. En lo que ha ciencia de datos y machine learning se refiere, Python es el lenguaje preferido por los científicos de datos.

Con el objetivo de organizar de mejor manera el código y que sea más entendible, se ha utilizado la herramienta JupyterLab, puesto que es un entorno de desarrollo interactivo muy usado en flujos de trabajo en ciencias de datos y machine learning Kluyver et al., (2016). La versión utilizada de JupyterLab es 6.1.11.

Por otro lado, Visual Studio Code es el editor elegido para escribir el código fuente. Debido a la gran cantidad de IDEs y editores de código que existen en el mercado todos ellos con muy buenas características. Cabe recalcar que VSC cuenta con soporte para trabajar con Jupyter Notebooks de manera nativa.

Las librerías con las que se ha desarrollado el pre - procesamiento del texto, la transformación a la forma intermedia y la aplicación de los algoritmos son las siguientes:

- Pandas 1.2.1: Librería utilizada para manejar las estructuras de datos.
- Unidecode 1.1.2: Librería que se la utiliza para eliminar los acentos.
- Regex 2020.11.13: Librería que maneja las expresiones regulares, se la utiliza para eliminar los signos de puntuación.
- NLTK 3.5: Paquete usado para la tokenización del texto, la eliminación de palabras vacías y el proceso de stemming
- SPACY 3.0.0: Paquete utilizado para realizar el proceso de lematización del texto.
- Scikit-Learn 0.24.2: Paquete que se lo utiliza para la implementación de los algoritmos.
- Matplotlib 3.3.4: Paquete utilizado para crear gráficas.

Los algoritmos de clasificación que se van a testear son:

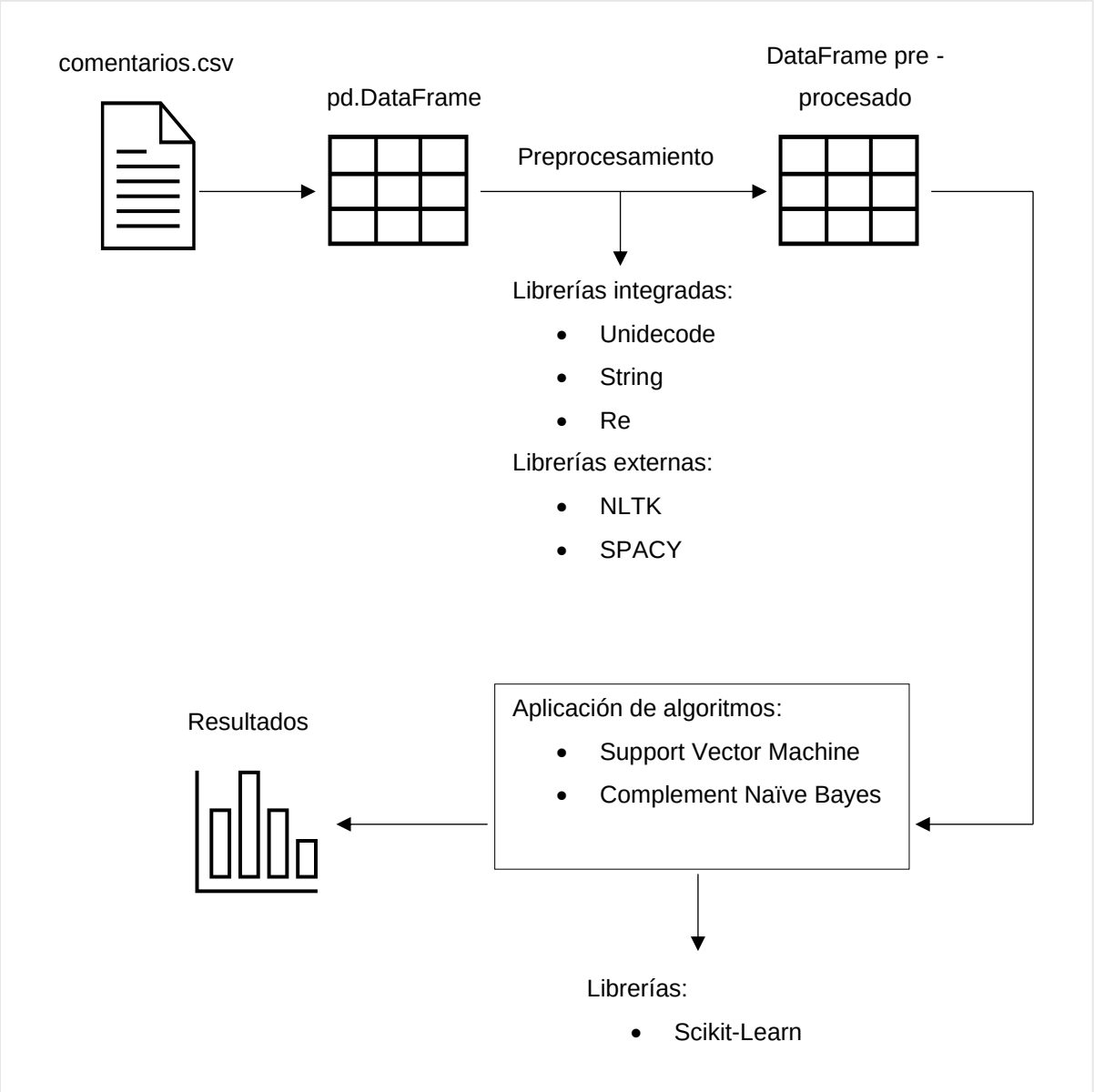
- Support Vector Machine
- Naive Bayes

Los algoritmos mencionados anteriormente fueron escogidos en base a los trabajos relacionados estudiados, en los que se encuentra un alto grado de usabilidad de estos algoritmos. En el apartado 2.8. se describen los trabajos relacionados.

En la figura 16 se presenta un diagrama de las herramientas anteriormente mencionadas.

Figura 16

Diagrama de las herramientas usadas.



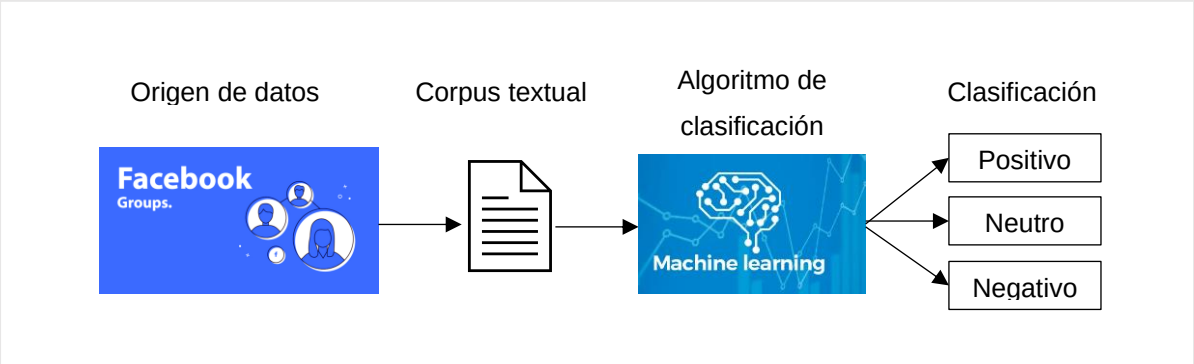
Nota. Esta figura muestra el diagrama de las herramientas utilizadas para medir el rendimiento de los algoritmos [Diagrama de flujo].

3.3. Fase 1: Propósito de la minería de texto

El propósito de la minería de texto es desarrollar un clasificador de sentimientos automatizado, el cual categorice comentarios de grupos de estudio de Facebook, como positivos, negativos o neutros (ver figura 17).

Figura 17

Clasificador de comentarios



Nota. Esta figura muestra a manera abstracta como va a funcionar el clasificador [Diagrama de flujo].

3.4. Fase 2: Recuperación de la información

Esta etapa empieza con la selección de la fuente de información. En este caso, se está trabajando con la red social Facebook. En esta plataforma se crearon grupos privados de 4 a 6 estudiantes cada uno. Los estudiantes pertenecen a la materia de Inteligencia Artificial de la Modalidad Abierta y a Distancia de la Universidad Técnica Particular de Loja. El objetivo de estos grupos es que en ellos se debatan, concuerden y organicen el desarrollo de los trabajos autónomos. Se identificaron un total de 45 grupos de estudio. En los 45 grupos de estudio se identificaron un total de 265 post. Entiéndase por post a cualquier texto que es publicado dentro del grupo. Los estudiantes interactúan respondiendo al post, a estas interacciones se las denomina comentarios. En todos los grupos de estudio existe un total de 1980 comentarios cuya distribución se muestra en la tabla 11.

Tabla 11

Distribución de los comentarios

Número de grupo	Número de post	Número de comentarios
1	4	5
2	17	51
3	4	6
4	29	63
5	9	28
6	12	17
7	27	276

Número de grupo	Número de post	Número de comentarios
8	26	105
9	5	22
10	26	94
11	7	10
12	3	9
13	13	29
14	16	97
15	5	10
16	13	43
17	9	30
18	4	3
19	6	14
20	6	16
21	8	201
22	10	48
23	14	35
24	23	24
25	4	3
26	4	18
27	4	15
28	15	33
29	17	38
30	3	4
31	17	60
32	18	34
33	6	9
34	2	4
35	3	9
36	3	1
37	12	23
38	11	15
39	3	2
40	9	15
41	2	20
42	3	7
43	9	25
44	8	7
45	10	17

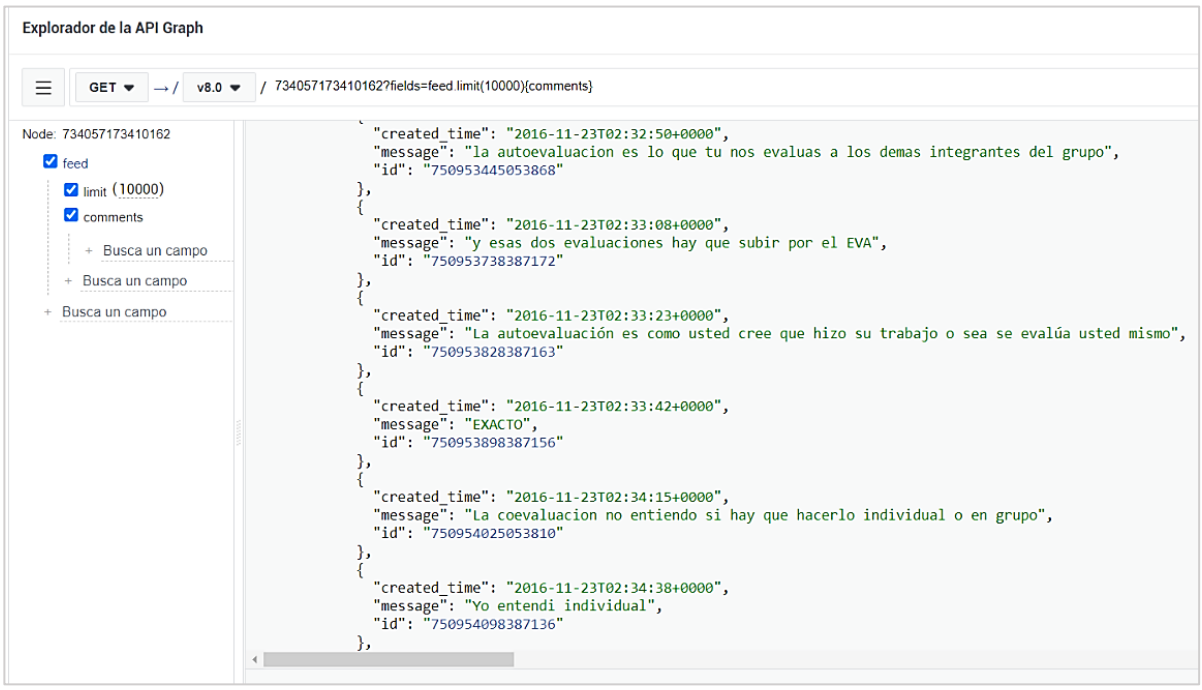
Nota. Distribución de los comentarios en los diferentes grupos de estudio en FB.

Para obtener la información se utiliza las herramientas para desarrollo que brinda la plataforma Facebook. Se ha creado una aplicación a la cual se la ha denominado *“GroupPost”.* Esto con el objetivo de acceder a la API Graph de Facebook Developers, ya que sin la creación de una aplicación esto no es posible. Una vez que se tiene acceso a la API Graph, se accede a la información de los grupos por medio de su ID. El ID de los grupos se lo encuentra en la URL de estos.

La API Graph de Facebook facilita en gran medida la descarga de la información ya que simplemente se necesita el ID del grupo y llenar unos cuantos parámetros para obtener los datos. En este caso en particular, los parámetros utilizados dentro de la API Graph, específicamente en el método GET fueron, *feed* para obtener todos los posts y *comments* para obtener todos los comentarios que tenía cada post. El formato elegido para la descarga de los datos fue JSON. Un ejemplo de cómo se muestran los datos una vez elegidos los parámetros y el formato de salida se muestra en la figura 18.

Figura 18

Obtención de los datos



Nota. Esta imagen muestra el resultado arrojado por la API Graph de Facebook.

Con el fin de facilitar la lectura de los datos, se ha procedido a integrarlos en un solo archivo y a convertirlos de un formato JSON a un formato CSV. El enlace al script para realizar este proceso se encuentra en el apéndice 1. Una vez que se tiene los datos en un formato más entendible se procede con la clasificación manual de los comentarios. Los posibles valores para cada comentario son: positivo, neutro y negativo. Estos valores corresponden a la clasificación basada en la polaridad, que es el objetivo de la presente investigación.

Este proceso resulta ser complejo y determinante a la hora de realizar el testeo de los algoritmos ya que como se mencionó en el apartado 2.6.1 los resultados de la clasificación van a depender de la persona que realice el proceso. Para mitigar este problema existen métodos los cuales van a permitir validar la clasificación manual. Uno de ellos es contar con una persona que sea totalmente ajena a la investigación. Esta persona deberá realizar el proceso de clasificación de forma totalmente autónoma sin intervención de los investigadores. Por su parte el investigador también realiza su clasificación y de esta forma se tiene al menos dos clasificaciones para el mismo conjunto de datos. Para la validación de la clasificación se aplica el Coeficiente Kappa de Cohen que es una medida estadística que permite medir el grado de acuerdo que existe entre dos mediciones. Hay que recalcar que esta es una tarea bastante tediosa y puede llevar mucho tiempo si el dataset es demasiado grande. Por otro lado, si no se cuenta con una persona externa que pueda ayudar con el proceso de clasificación, es posible utilizar alguna herramienta ya existente como las mencionadas en el apartado 3.2 para obtener una segunda clasificación con la cual se podrá realizar el proceso de validación aplicando el Coeficiente Kappa de Cohen.

Para esta investigación, se ha utilizado una poderosa herramienta del gigante tecnológico Google, específicamente *Google Cloud NLP* para realizar una segunda clasificación. Como se menciona en el apartado 3.2 el uso de esta API tiene un costo, pero también es posible realizar alrededor de 5000 peticiones al mes completamente gratis. El enlace del script desarrollado para la utilización de la herramienta se encuentra en el apéndice 2.

Una vez que se ha obtenido las dos clasificaciones para el mismo conjunto de datos se procede a realizar la validación por medio de la aplicación del Coeficiente Kappa de Cohen. Se ha utilizado la herramienta IBM SPSS Statistic en su versión 21 para automatizar este proceso. Los resultados obtenidos luego del procesamiento de los datos se muestran en las tablas 19, 20 y 21.

Tabla 12

Resumen del procesamiento de los casos

	Casos					
	Válidos		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
POLARIDAD_PERSONA* POLARIDAD_GOOGLE	17440	100,0%	0	0,0	1744	100,0 %

Nota. En la tabla se observa que todos los datos se procesaron correctamente.

Tabla 13

*Tabla de contingencia POLARIDAD_PERSONA * POLARIDAD_GOOGLE*

			POLARIDAD_GOOGLE			
			negativo	neutro	positivo	Total
POLARIDAD_PERSONA	negativo	Recuento	141	0	0	141
		% del total	8,1%	0,0%	0,0%	8,1 %
	neutro	Recuento	0	475	219	694
		% del total	0,0%	27,2 %	12,6%	39,8%
	positivo	Recuento	0	0	909	909
		% del total	0,0%	0,0%	52,1%	52,1%
Total		Recuento	141	475	1128	1744
		% del total	8,1%	27,2%	64,7%	100,0%

Nota. En la tabla se observa el desglose de resultados de ambas clasificaciones.

Tabla 14

Medidas simétricas

	Valor	Error típ. asint.[a]	T aproximada[b]	Sig. aproximada
Medida de acuerdo Kappa	,771	0,14	40,584	,000
N de casos válidos	1744			

Nota. En la tabla se observa que el índice de kappa es de 0,771

La tabla 12, corresponde a la primera salida que produce el programa SPSS en la cual se puede observar un resumen de los casos procesados. Se tiene un total de 1744 valores de clasificación lo que representa el 100%. No se tiene casos perdidos.

La tabla 13, corresponde a la tabla de contingencia, en ella se puede observar que tanto la clasificación personal, como la clasificación hecha por *Google Cloud NLP* tienen 141 instancias clasificadas como negativas; es decir, que la clasificación personal y la clasificación de la herramienta coinciden plenamente en esta instancia. Por otro lado, en la segunda fila se puede observar que existe discordancia en la clasificación. Para *Google Cloud NLP* 219 comentarios son positivos y para la persona estos mismos comentarios son considerados de polaridad neutra. También existe una concordancia de 475 comentarios en ambas clasificaciones. Finalmente, se tiene que ambas clasificaciones consideran que 909 comentarios son de polaridad positiva. Analizando estos valores se puede llegar a la conclusión que la clasificación de polaridad neutra y positiva representó un mayor reto para la persona.

En la tabla 14, se observa las medidas simétricas. Para esta validación lo que interesa es la medida de acuerdo Kappa. El valor que está arrojando la medición es 0.771. Este valor corresponde al rango de 0.60 – 0.80 que indica una buena concordancia de acuerdo a la interpretación que se muestra en la tabla 15.

Tabla 15

Coeficiente Kappa de Cohen

Índice de Kappa	Interpretación
0.00 – 0.20	Ínfima concordancia
0.00 – 0.40	Escasa concordancia
0.40 – 0.60	Moderada concordancia
0.60 – 0.80	**Buena concordancia**
0.80 – 1.00	Muy buena concordancia

Nota. En esta tabla se observa la interpretación que se la da al resultado arrojado por el Coeficiente Kappa de Cohen.

Este resultado obtenido demuestra que la clasificación que se hizo de manera personal es buena y de esta manera se puede seguir realizando las siguientes fases del proceso de análisis de sentimientos. Con esto se da por finalizada la Fase 2 de la metodología. La fase 3 tiene como entrada un archivo en formato CSV en el cual se encuentran los comentarios y su clasificación previamente validada.

3.5. Fase 3: Pre - procesamiento de texto

Es importante tener un conocimiento profundo de los datos con los que se está trabajando. Conocer su tipo, cuáles son sus valores, la relación que existe entre ellos y su distribución. Una forma de realizar esto, es trabajar con el paquete Pandas el cual cuenta con métodos que permiten analizar a profundidad el dataset.

Como entrada se tiene el conjunto de datos en un solo archivo de extensión CSV. En este caso en particular todos los valores son de tipo *string.* Se procede a cargar y a crear un dataframe. Esto se lo realiza con el siguiente código;

```
df = pd.read_csv('Data/comentarios.csv', sep=';')
```

df representa la variable en donde va a estar almacenado el dataframe. La biblioteca Pandas ha sido importada previamente como *pd.* El *método read_csv* es el que permite cargar los datos, en este caso se ha usado dos parámetros. El primero, corresponde a la ruta en donde están almacenados los datos y el segundo, corresponde al separador de los datos.

Por otro lado, para conocer la distribución de los datos se cuenta con el método *describe().* Con este método se puede observar la cuenta total por columnas, los valores únicos, cuáles son las instancias que más se repiten y la frecuencia con que lo hacen. Ejecutando el siguiente código

```
df.describe()
```

se tiene el siguiente resultado;

Tabla 16

Resultados de la ejecución del método describe()

	MENSAJES	**POLARIDAD**
count	1744	1744
unique	1694	3
Top	ok	positivo
freq	9	1128

Nota. En esta tabla se observa la descripción de los datos

El resultado indica que no se tiene valores perdidos, tanto la columna de MENSAJES como la de POLARIDAD cuentan con 1744 instancias, que coinciden con el tamaño del dataframe. Esto se lo puede verificar con el siguiente código;

```
df.shape
```

cuyo resultado es el siguiente;

(1744, 2)

El primer valor corresponde al número de filas y el segundo valor al número de columnas. La columna MENSAJES tiene 1694 instancias únicas y la columna de POLARIDAD 3. La instancia que más se repite en la columna de MENSAJES es *ok*, mientras que en cuanto a la polaridad se refiere el valor que más se repite es *positivo*. Finalmente se muestra la frecuencia con la que los valores antes mencionados se repiten. 9 veces para *ok* y 1128 para positivo.

Otro punto importante es, conocer el número de instancias clasificadas como positivas, cuantas como neutras y cuantas como negativas. Esto se lo puede hacer con el método *value_counts* de la biblioteca Pandas. El código es el siguiente;

```
pd.value_counts(df['POLARIDAD'], sort = True)
```

El resultado es el siguiente;

Tabla 17

Resultados de la ejecución del método value_counts()

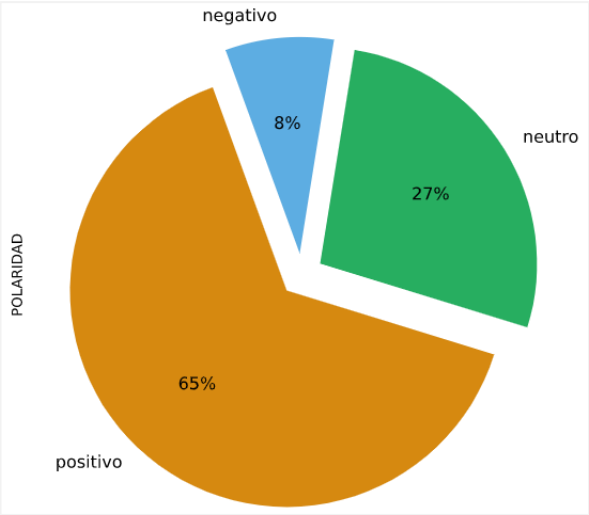
positivo	1128
negativo	475
neutro	141

Nota. En esta tabla se observa la distribución de los datos de acuerdo a la categoría.

Para visualizar de mejor manera estos resultados se ha creado una gráfica de pastel. Esto se lo hace con la biblioteca Matplotlib. El enlace al código de la implementación de estas funciones se encuentra en el apéndice 3.

Figura 19

Distribución de los valores.



Nota. En la gráfica de pastel se observa que tan solo el 8% de comentarios son negativos.

Se observa que existe un 65% de datos que corresponden a la clase positiva, seguidos de un 27% que pertenecen a la clase neutra y finalmente en la clase negativa se tiene un 8% de datos; por consiguiente, el 35% no son positivos. Estos resultados son realmente importantes y determinantes a la hora de entrenar un algoritmo de clasificación y es que en la mayoría de las situaciones lo ideal sería contar con un conjunto de datos balanceados, es decir un 33,3% de datos en cada clase. Pero como se puede observar en esta investigación ese no es el caso. Es por esta razón que se tiene que aplicar estrategias para mitigar este problema, las cuales serán aplicadas en la fase 4 de la metodología.

El siguiente paso que se realizó dentro de esta fase es la normalización de los datos con el objetivo de reducir la dimensión del vector que se crea al momento de transformar el texto a números. Como bien se sabe, los algoritmos de clasificación reciben como entrada datos numéricos y por eso es necesario llevar a cabo un proceso de transformación también conocido como representación o forma intermedia.

Las herramientas utilizadas para el preprocesamiento del texto son; el kit de herramientas de lenguaje natural NLTK, el paquete spacy y funciones integradas de Python. A continuación, se detallan los pasos realizados:

a) Se utilizada una función que viene integrada en el paquete de Python llamada *lower(),* que permite convertir el texto a minúsculas. El código es el siguiente:

```
df['minusculas'] = df['MENSAJES'].str.lower()
```

`df['minusculas']` es una nueva columna en el dataframe en la cual se va a almacenar el texto convertido. A continuación se muestra un ejemplo:

Tabla 18

Resultados del texto transformado a minúsculas

Mensaje	Minúsculas
Por favor pongamos en la portada algo que tenga relación con la asignatura.	por favor pongamos en la portada algo que tenga relación con la asignatura.

Mensaje	Minúsculas
Buenas tardes, que tal esta imagen. Compañero alguna sugerencia.	Buenas tardes, que tal esta imagen. Compañero alguna sugerencia.
A mí me gusta, esperemos a ver que dicen sus compañeros de equipo.	a mí me gusta, esperemos a ver que dicen sus compañeros de equipo.

b) Se realiza la eliminación de signos de puntuación, para ello se utiliza expresiones regulares. La biblioteca *re* que también está incluida en el paquete de Python permite la creación y aplicación de expresiones regulares. El código es el siguiente:

```
reg_signos = re.compile('[^\w\s]')
df['puntuacion'] = df[minusculas].replace(reg_signos, '')
```

A través del método compile se crea la expresión regular, la cual es almacenada en la variable `reg_signos`. Con el método `replace` aplicado a la columna `df[minusculas]`, se remplazan todas coincidencias de la expresión regular por `''`. El resultado es almacenado en una nueva columna de dataframe a la cual se la llama `df['puntuacion']`. A continuación, se presenta un ejemplo:

Tabla 19

Resultados del texto sin signos de puntuación

Minúsculas	Puntuación
por favor pongamos en la portada algo que tenga relación con la asignatura.	por favor pongamos en la portada algo que tenga relación con la asignatura
buenas tardes, que tal esta imagen. compañero alguna sugerencia.	buenas tardes que tal esta imagen compañero alguna sugerencia
a mí me gusta, esperemos a ver que dicen sus compañeros de equipo.	a mí me gusta esperemos a ver que dicen sus compañeros de equipo

c) Se realiza la eliminación de acentos con otra función que viene integrada en el paquete de Python que se denomina *unidecode()*. Esta función se la aplica a la columna `df['puntuacion'].` El código es el siguiente:

```
df['acentos'] = df['puntuacion'].apply(unidecode)
```

El resultado se almacena en una nueva columna llamada `df['acentos'].` Nótese que con esta función también se logra la eliminación de la virgulilla (ñ). A continuación, se muestra un ejemplo:

Tabla 20

Resultados del texto sin acentos

Puntuación	Acentos
por favor pongamos en la portada algo que tenga relación con la asignatura	por favor pongamos en la portada algo que tenga relacion con la asignatura
buenas tardes que tal esta imagen companero alguna sugerencia	buenas tardes que tal esta imagen companero alguna sugerencia
a mí me gusta esperemos a ver que dicen sus companeros de equipo	a mi me gusta esperemos a ver que dicen sus companeros de equipo

d) El siguiente paso en esta fase de preprocesamiento consiste en tokenizar el texto es decir, segmentar el texto. Para realizar esto se utiliza la función `word_tokenize()` del kit de herramientas de lenguaje natural NLTK. Se aplica esta función a la columna `df['acentos']` del dataframe. El código es el siguiente:

```
df['tokens'] = df['acentos'].apply(word_tokenize)
```

`df['tokens']` es la nueva columna en donde se almacena el resultado. A continuación, se presenta un ejemplo:

Tabla 21

Resultados del texto tokenizado

Acentos	Tokens
por favor pongamos en la portada algo que tenga relacion con la asignatura	'por', 'favor', 'pongamos', 'en', 'la', 'portada', 'algo', 'que', 'tenga', 'relacion', 'con', 'la', 'asignatura'
buenas tardes que tal esta imagen companero alguna sugerencia	'buenas', 'tardes', 'que', 'tal', 'esta', 'imagen', 'companero', 'alguna', 'sugerencia'
a mi me gusta esperemos a ver que dicen sus companeros de equipo	'a', 'mi', 'me', 'gusta', 'esperemos', 'a', 'ver', 'que', 'dicen', 'sus', 'companeros', 'de', 'equipo',

e) Se eliminan las stopwords. El kit de herramientas NLTK proporciona un corpus textual de palabras vacías a través del método `words()`. Como parámetro de este método se especifica el idioma, en este caso, `spanish`. El código es el siguiente:

```
stop_words = set(stopwords.words('spanish'))
```

El diccionario de palabras vacías es almacenado en la variable `stop_words.` Luego por medio de un ciclo repetitivo y de un condicional, se eliminan las palabras del dataframe. El código es el siguiente:

```
df['stopwords'] = df['tokens'].apply(lambda x: [item for item in x
                    if item not in stop_words])
```

El texto sin palabras vacías es almacenado en una nueva columna llamada `['stopwords'].` A continuación, se presenta un ejemplo:

Tabla 22

Resultados del texto sin stopwords

Tokens	Stopwords
'por', 'favor', 'pongamos', 'en', 'la', 'portada', 'algo', 'que', 'tenga', 'relacion', 'con', 'la', 'asignatura'	'favor', 'pongamos', 'portada', 'relacion', 'asignatura'
'buenas', 'tardes', 'que', 'tal', 'esta', 'imagen', 'companero', 'alguna', 'sugerencia'	'buenas', 'tardes', 'imagen', 'companero', 'alguna', 'sugerencia'
'a', 'mi', 'me', 'gusta', 'esperemos', 'a', 'ver', 'que', 'dicen', 'sus', 'companeros', 'de', 'equipo',	'gusta', 'esperemos', 'ver', 'dicen', 'companeros', 'equipo'

Aquí se puede observar claramente como el texto se reduce significativamente. El objetivo es reducir aún más la cantidad de características del corpus, por esta razón se aplican dos pasos más a esta fase de preprocesamiento.

f) Lematizar el texto. Para realizar este proceso se utiliza la biblioteca spaCy que cuenta con métodos avanzados que usan conocimiento lingüístico con el fin de procesar el texto. Lematizar consiste en reducir palabras a lemas. Un lema es la forma que se acepta como representación de todas las formas posibles de la palabra.

En este apartado no se muestra específicamente cómo se ha logrado realizar este proceso debido a que es más complejo y requiere más líneas de código. Pero puede visitar el apéndice 4 en el cual se encuentra el enlace a todo el código utilizado en la fase de pre - procesamiento.

A continuación, un ejemplo del resultado obtenido:

Tabla 23

Resultados del texto lematizado

Stopwords	Lemas
'favor', 'pongamos', 'portada', 'relacion', 'asignatura'	'favor', 'poner', 'portada', 'relacion', 'asignatura'
'buenas', 'tardes', 'imagen', 'companero', 'alguna', 'sugerencia'	'buena', 'tardes', 'imagen', 'companero', 'alguno', 'sugerencia'
'gusta', 'esperemos', 'ver', 'dicen', 'companeros', 'equipo'	'gustar', 'esperemos', 'ver', 'decir', 'companero', 'equipo'

g) Finalmente, y como último paso en esta fase de preprocesamiento se realiza el proceso de Stemming, como se menciona en el aparatado 2.4.1, consiste en reducir una palabra a su raíz. Para ello se utiliza funciones del kit de herramientas de lenguaje natural NLTK. A continuación, se muestra los resultados obtenidos:

Tabla 24

Resultado del texto realizado el proceso de stemming

Lemas	Stemming
'favor', 'poner', 'portada', 'relacion', 'asignatura'	'favor', 'pon', 'port', 'relacion', 'asignatur'
'buena', 'tardes', 'imagen', 'companero', 'alguno', 'sugerencia'	'buen', 'tard', 'imag', 'companer', 'algun', 'sugerent'
'gustar', 'esperemos', 'ver', 'decir', 'companero', 'equipo'	'gust', 'esper', 'ver', 'dec', 'companer', 'equip'

De la misma forma que con la lematización no se muestra específicamente como se realizó este proceso, pero pude ir al apéndice 4 en el cual se encuentra el enlace que contiene todo el código de la fase de preprocesamiento.

Finalmente, se procede a guardar el texto procesado en un archivo de extensión CSV. El enlace a ese archivo se encuentra en el apéndice 5.

3.5.1. Representación o forma intermedia

La librería scikit-learn cuenta con algoritmos que permiten transformar el texto a números. A esta transformación de texto a números se la conoce como representación o forma intermedia y es la entrada para los algoritmos de clasificación. En este caso en particular se ha utilizado el algoritmo Frecuencia del Término por Frecuencia Inversa del Documento (TF-IDF) cuyo método en es `TfidfVectorizer()`. Los hiper - parámetros utilizados en este método son los siguientes:

- *Input = 'content'*
- *encoding = 'utf-8'*
- *decode_error = 'strict'*
- *strip_accents = None*
- *lowercase = None*
- *preprocessor = None*
- *tokenizer = None*
- *analyzer = 'word'*
- *stop_words = None*
- *token_pattern = r'(?u)\b\w\w+\b'*
- *ngram_range = (1, 1)*
- *max_df = 0.8*
- *min_df = 7*
- *max_features = 50*
- *vocabulary = None*
- *binary = False*
- *dtype = np.float64*
- *norm = 'l2'*

- *use_idf = True*
- *smooth_idf = True*
- *sublinear_tf = False*

Para un mejor entendimiento de que hace cada parámetro y sus posibles valores se recomienda revisar la documentación oficial de scikit-learn. Con esto se da por finalizada la etapa de pre - procesamiento y se deja listo el texto transformado a vectores numéricos que sirven como entrada para los algoritmos de clasificación que se probaran en la siguiente fase de la metodología. La siguiente fase de la metodología corresponde al capítulo 4 del presente trabajo de fin de titulación.

Capítulo cuatro

Aplicación de algoritmos y análisis de resultados

En el presente capítulo se lleva a cabo la fase final de la metodología. En la sección 4.1 se explica la forma en la que se crean los modelos de clasificación. Dentro de esta sección, específicamente en el apartado 4.1.1 se explica cómo se eligieron los hiper – parámetros para el clasificador. En el apartado 4.1.2 y 4.1.3 se explica la forma en la que se realiza el entrenamiento y las pruebas de los algoritmos. Finalmente, en el apartado 4.1.4 se explican y se prueban técnicas para trabajar con datos desbalanceados.

4.1. Fase 4: Aplicación del método de minería de texto y análisis de resultados

En esta fase se aplican los algoritmos de clasificación. En este caso en particular los algoritmos seleccionados son: Support Vector Machine y Naïve Bayes.

Previamente al entrenamiento del modelo de clasificación se procede a dividir los datos en un conjunto de datos de entrenamiento y un conjunto de datos de prueba. Esto se logra por medio de la siguiente función:

```
def dividirDataSet(vector, labels):
    X_train, X_test, y_train, y_test = train_test_split(
        vector,
        labels,
        test_size=0.2,
        train_size=None,
        random_state=0,
        shuffle=True,
        stratify=None,
    )
return X_train, X_test, y_train, y_test
```

Específicamente se utiliza el método `train_test_split()` el cual recibe como parámetro el vector que representan los comentarios transformados a un vector numérico y

la etiqueta que corresponde al valor de la clasificación. El hiper - parámetro `test_size` cuyo valor es 0.2 representa el porcentaje de nuestro conjunto de prueba. 0.2 significa que los datos se dividirán en 20% para las pruebas y lo que queda que corresponde el 80% para el entrenamiento. El método retorna `X_train, y_train` que corresponden a los valores y la etiqueta que se las utilizará para el entrenamiento, y `X_test, y_test` que representan los valores y la etiqueta que se utilizarán para testear el modelo de clasificación.

Una vez que se tiene el conjunto de datos dividido, se procede con la creación del modelo. El primero algoritmo es el Support Vector Machine. Scikit-learn cuenta con el método `SVC()` para su creación. El segundo algoritmo que será puesto a prueba es el Complement Naïve Bayes. Este algoritmo es una variante del algoritmo Naïve Bayes el cual es adecuado para conjuntos de datos que tiene sus clases desequilibradas que es el caso de esta investigación, tal como se analizó en el apartado 3.5. Para la creación de este modelo de clasificación Scikit-learn cuenta con el método `ComplementNB().`

4.1.1. Optimización de los hiper – parámetros

Un punto clave a la hora de crear los modelos de clasificación son los valores de los hiper – parámetros ya que ajustando estos valores se puede conseguir el mejor rendimiento. Así por ejemplo SVC tiene un hiper – parámetro llamado kernel el cual representa la función con la que el algoritmo intenta dividir el conjunto de datos de acuerdo a las clases. Los posibles valores para kernel son ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’. Entonces para lograr el mejor resultado posible se tendría que probar cada uno de estos valores y todas las posibles combinaciones que existan. Esta tarea resulta compleja más aún si se tiene un conjunto de datos demasiado grande.

Scikit-learn brinda la posibilidad de automatizar este proceso por medio de una función llamada *GridSearch* que se encarga de analizar todas las posibles combinaciones de hiper – parámetros que se elijan para el algoritmo. El funcionamiento es el siguiente:

a) Se crea el modelo de clasificación

```
classifier = SVC()
```

b) Se crean los posibles hiper - parámetros en un diccionario

```
params = {
    'C': [1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'random_state': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
    }
```

c) El método `GridSearchCV` recibe el clasificador que en este caso es SVC y los posibles valores de los hiper – parámetros que vienen dados en un diccionario. Lo que hace este método es crear el modelo con todas las posibles combinaciones. El código es el siguiente

```
grid = GridSearchCV(classifier, params)
```

d) Se entrena el modelo con todas las posibilidades posibles

```
grid.fit(X_train, y_train)
```

e) Finalmente se obtienen los mejores parámetros para los datos por medio del método `best_estimator_.get_params()`. Que da como resultado lo siguiente:

```
{'C': 2.0,
 'break_ties': False,
 'cache_size': 200,
 'class_weight': None,
 'coef0': 0.0,
 'decision_function_shape': 'ovr',
 'degree': 3,
 'gamma': 'scale',
 'kernel': 'rbf',
 'max_iter': -1,
 'probability': False,
 'random_state': None,
```

```
 'shrinking': True,
 'tol': 0.001,
 'verbose': False}
```

El mismo procedimiento se lo realiza para el algoritmo Complement Naïve Bayes:

a) Se crea el modelo de clasificación

```
classifier = ComplementNB()
```

b) Se crean los posibles hiper - parámetros en un diccionario

```
params = {
    'alpha': [1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0],
}
```

c) El método `GridSearchCV` recibe el clasificador que en este caso es ComplementNB y los posibles valores de los hiper – parámetros que vienen dados en un diccionario. Como se menciona anteriormente lo que hace este método es crear el modelo con todas las posibles combinaciones de parámetros. El código es el siguiente

```
grid = GridSearchCV(classifier, params)
```

d) Se entrena el modelo

```
grid.fit(X_train, y_train)
```

e) Finalmente se obtienen los mejores parámetros para los datos por medio del método `best_estimator_.get_params()`. Que da como resultado lo siguiente:

```
{'alpha': 1.0,
'class_prior': None,
'fit_prior': True,
'norm': False}
```

4.1.2. Entrenamiento y pruebas Support Vector Machine

a) Una vez que se obtienen los mejores parámetros posibles se procede a la creación del modelo con estos parámetros. La función para la creación del modelo es la siguiente:

```
def SupportVectorMachine():
    classifier = SVC(
       'C': 2.0,
       'break_ties': False,
       'cache_size': 200,
       'class_weight': None,
       'coef0': 0.0,
       'decision_function_shape': 'ovr',
       'degree': 3,
       'gamma': 'scale',
       'kernel': 'rbf',
       'max_iter': -1,
       'probability': False,
       'random_state': None,
       'shrinking': True,
       'tol': 0.001,
      'verbose': False)
    return classifier
```

b) Luego se entrena el modelo. Para ello se utiliza la siguiente función:

```
def entrenarModelo(classifier, X_train, y_train):
    return classifier.fit(X_train, y_train)
```

La función `entrenarModelo()` recibe como parámetros el clasificador que previamente fue creado y `X_train, y_train` que corresponden al conjunto de datos de entrenamiento. La función retorna el mismo clasificador pero que ya ha sido entrenado.

c) Finalmente se procede a probar el modelo. Se lo realiza con la siguiente función:

```
def probarModelo(classifier, X_test):
```

```
return classifier.predict(X_test)
```

La función recibe como parámetros, el clasificador entrenado `X_test` que corresponde al conjunto de pruebas. La función devuelve las predicciones hechas por el algoritmo con respecto a `X_test`, es decir devuelve la clase a la que pertenece el comentario según el clasificador.

Ahora corresponde analizar los resultados obtenidos por el modelo. Como se menciona previamente, el conjunto de datos que se tiene es desequilibrado. De acuerdo a la figura 22 se tiene que el 65% de datos corresponde a la clase positiva, el 27% a la clase neutra y el 8% a la clase negativa. Es por esta razón que es conveniente analizar los resultados basándose en las distintas métricas existentes y no solo en la exactitud como en la mayoría de los casos. Las métricas utilizadas están descritas a continuación:

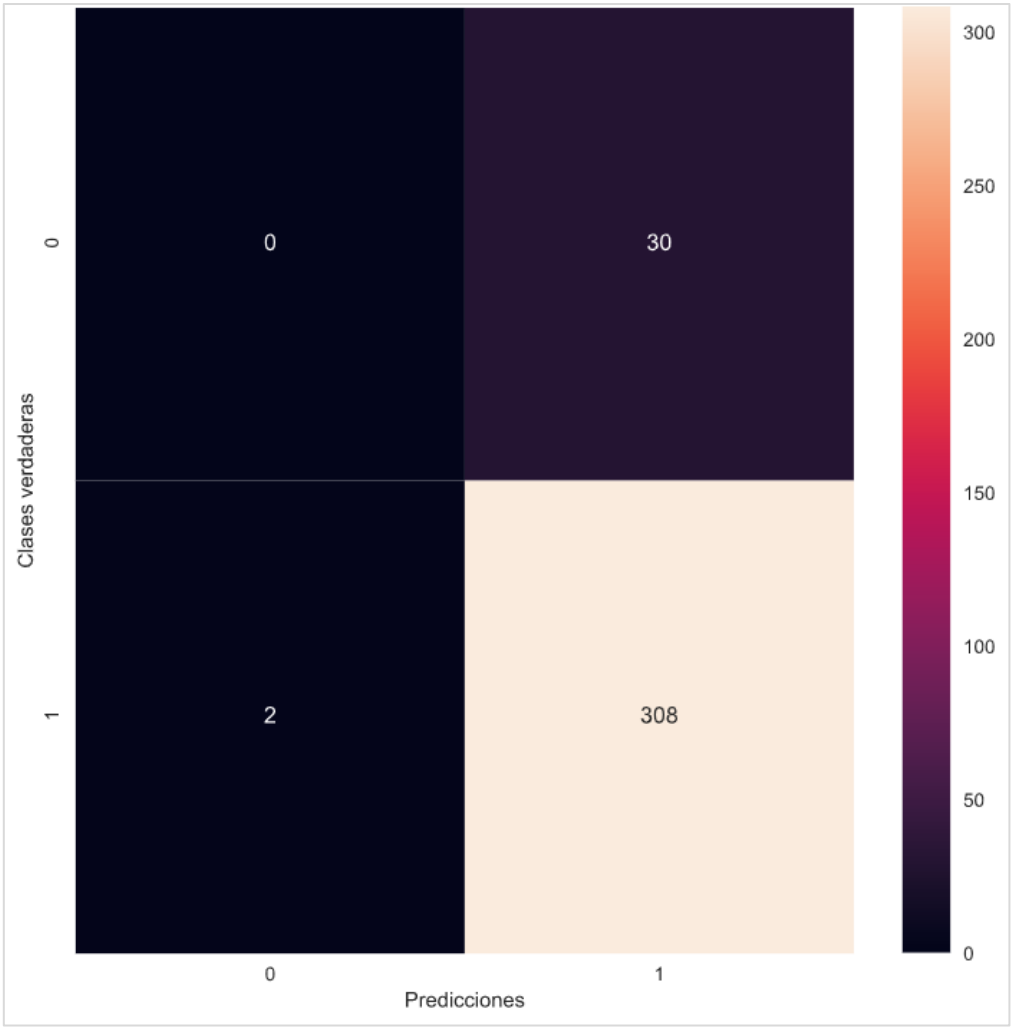
- *Matriz de confusión.* - Es una métrica sencilla, también se la denomina tabla de contingencia. Muestra cómo se distribuyen los valores reales y los valores predichos por el clasificador. Se la utiliza como entrada para otras métricas como la precisión, exactitud, puntaje F1.
- *Exactitud* – accuary.. - Es el número de predicciones correctas que realiza el modelo dividido por el número total de registros. Mientas más cerca de 1.0 es mejor.
- *Precisión:* es un valor que define si un modelo es confiable a la hora de responder si un valor corresponde a esa clase. Siempre que el valor se acerque a 1.0 será mejor.
- *Sensibilidad – Recall.* – Este valor expresa cuan bien el clasificador puede detectar todos los ejemplos en su clase correspondiente. Mientras más se aproxime a 1.0 será mejor.
- *Puntaje F1.* - Esta es una métrica que combina la precisión y la sensibilidad en un solo valor. Es la media armónica de ambas métricas.

Scikit-learn cuenta con métodos para calcular cada una de las métricas antes mencionadas. Se comienza analizando la matriz de confusión. Para ello se utiliza el método `confusion_matrix()` que recibe como parámetros las etiquetas de nuestro conjunto de

pruebas que en este caso están almacenadas en `y_test` y las predicciones realizadas por el clasificador. Para visualizar de mejor manera los resultados se ha graficado la matriz de confusión con un mapa de calor utilizando la librería matplotlib. En resultado es el siguiente:

Figura 20

Matriz de confusión del algoritmo Support Vector Machine



La matriz de confusión muestra que el algoritmo Support Vector Machine está detectando 308 verdaderos positivos para la clase 1. No está detectando verdaderos positivos para la clase 0, ni para la clase 2. También se puede ver que 30 instancias que corresponde a la clase 0, el modelo las está clasificando como si fuesen de la clase 1. A demás se observa que 2 instancias que pertenecen a la clase 1, el modelo las está clasificando como si fuesen

de la clase 0. La clase 0 hace referencia a la clase negativa, la clase 1 hace referencia a la clase positiva y la 2 a la clase neutra.

A partir de estos valores se pueden calcular las siguientes. Scky-learn permite automatizar este proceso a través del método `ClassificationReport()`. Los resultados se muestran a continuación:

Tabla 25

Reporte de clasificación con Support Vector Machine

	Precision	**Recall**	**F1-score**	**support**
Negativo	0.00	0.00	0.00	30
Neutro	0.00	0.00	0.00	9
Positivo	0.89	0.99	0.94	310
Accuary			0.88	349
Macro avg	0.30	0.33	0.31	349
Weighted avg	0.79	0.88	0.89	349

El clasificador obtuvo una exactitud de 0.88. Este es un resultado muy engañoso porque como se observó en la matriz de confusión el algoritmo no pudo detectar instancias negativas y neutras. En la literatura relacionada al análisis de sentimientos, se recomienda usar la métrica f1-score para medir el rendimiento de los clasificadores. Esta métrica es una media armónica de la precisión y la sensibilidad. Analizando los resultados obtenidos se observa que la clase negativa y la clase neutra tiene un valor de f1-score de 0.00 lo que indica que el clasificador es realmente malo para detectar estas clases. Por otro lado, la clase positiva tiene un valor de 0.94, lo cual indica que el modelo predice de mejor manera esta clase. El promedio macro de los valores f1-score es de 0,31.

4.1.3. Entrenamiento y pruebas Complement Naïve Bayes

a) Para crear el modelo se utiliza la siguiente función:

```
def ComplementNaiveBayes():
    classifier = ComplementNB(
```

```
            alpha=1.0,
            fit_prior=False,
            class_prior=None,
            norm=False
        )
        return classifier
```

b) La función para entrenar el modelo es la siguiente:

```
def entrenarModelo(classifier, X_train, y_train):
    return classifier.fit(X_train, y_train)
```

la función recibe como parámetros el clasificador creado previamente y el conjunto de datos de entrenamiento que está dado por `X_train, y_train.` La función retorna el modelo ya entrenado.

c) El siguiente paso es probar el modelo. Para realizar este paso se utiliza la siguiente función:

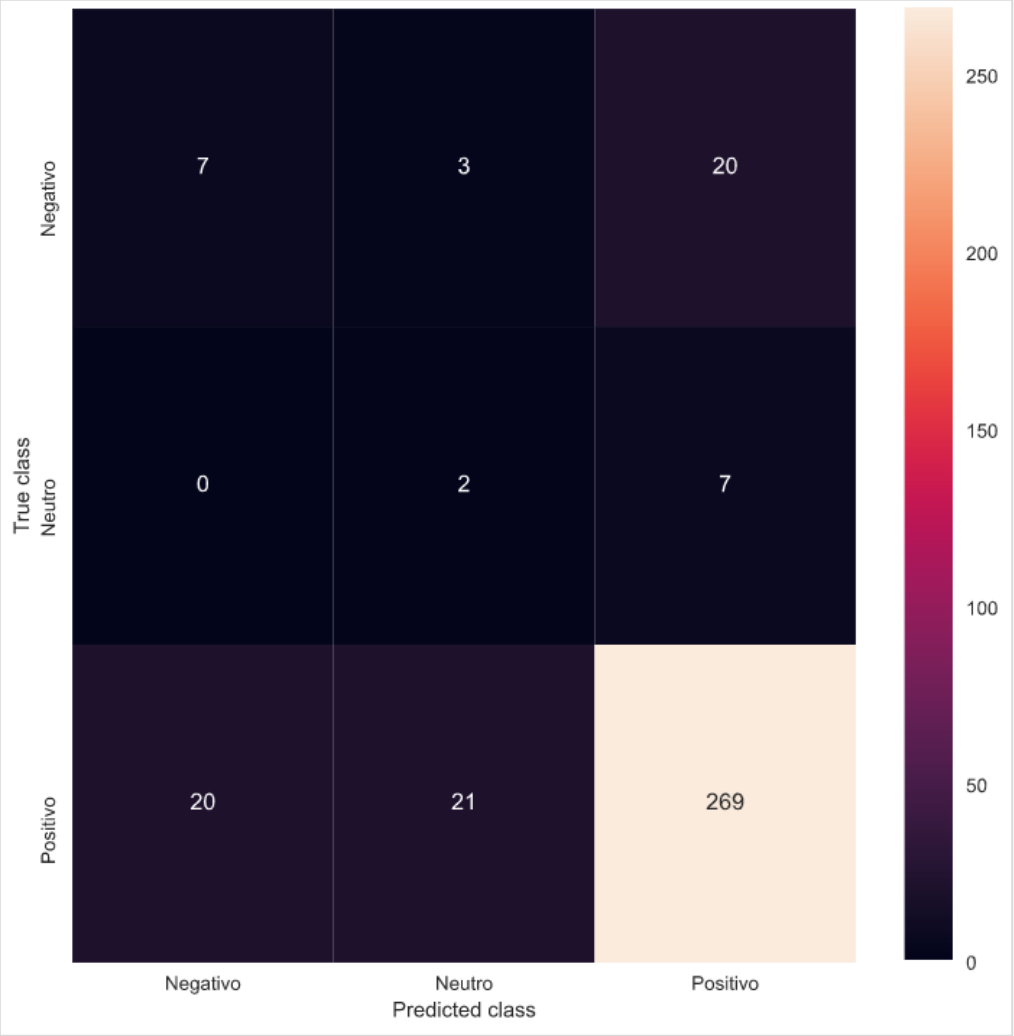
```
def probarModelo(classifier_fit, X_test):
    return classifier_fit.predict(X_test)
```

La función recibe como parámetro el clasificador que previamente ha sido entrenado, y los comentarios para las pruebas que vienen dados en `X_test.` La función retorna las predicciones hechas por el clasificador para los valores de `X_test.`

Luego se aplican los mismos métodos indicados en el apartado 4.1.2 para obtener y graficar los resultados. La matriz de confusión para este clasificador es la siguiente:

Figura 21

Matriz de confusión Complement Naïve Bayes



A primera vista observando los resultados de la matriz de confusión se puede deducir que el algoritmo Complement Naïve Bayes funciona mejor que Support Vector Machine. Ya que este si está detectando instancias negativas y neutras. Aquí se observa que el algoritmo detectó 269 positivos verdaderos. 2 neutros verdaderos y 7 negativos verdaderos. De la misma forma se observa que 20 instancias que son positivos, el algoritmo los clasificó como negativos y 21 instancias que son positivos las clasificó como neutras. También se tiene que 7 instancias que son neutros los clasificó como positivos. 3 instancias negativas las clasificó como neutras y 20 instancias negativas las clasificó como positivas.

A continuación, se presenta un reporte con las otras métricas para este clasificador:

Tabla 26

Reporte de clasificación con Complement Naïve Bayes

	Precision	Recall	F1-score	support
Negativo	0.26	0.23	0.25	30
Neutro	0.08	0.22	0.11	9
Positivo	0.91	0.87	0.89	310
Accuary			0.80	349
Macro avg	0.41	0.44	0.42	349
Weighted avg	0.83	0.80	0.81	349

Los valores de precisión, sensibilidad y por ende f1-score mejor notablemente con respecto al clasificador Support Vector Machine. Se observa que la clase negativa obtiene un valor de precisión de 0.26, la clase neutra un valor de 0.8 y la clase negativa 0.91. Con respecto a la sensibilidad, la clase negativa obtuvo un valor de 0.23, la clase neutra 0.22 y la clase positiva 0.87. La métrica f1-score marca que la clase negativa tiene un valor de 0.25, la clase neutra 0.11 y la clase positiva un valor de 0.89. El promedio macro de la métrica f1 score es de 0.42

A continuación, se presentan las gráficas comparativas de los resultados obtenidos:

Figura 22

Comparativa de la métrica de precisión

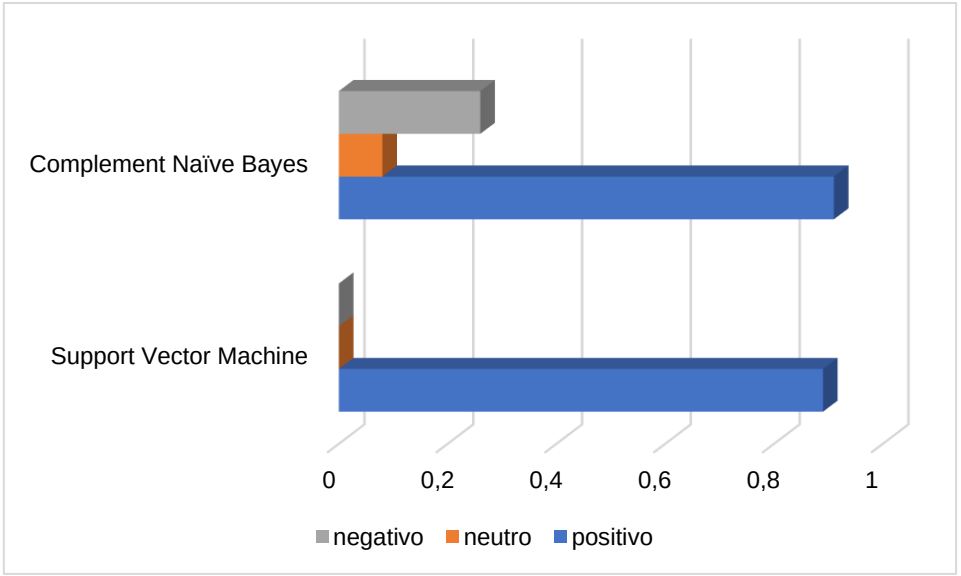


Figura 23

Comparativa de la métrica de sensibilidad o recall

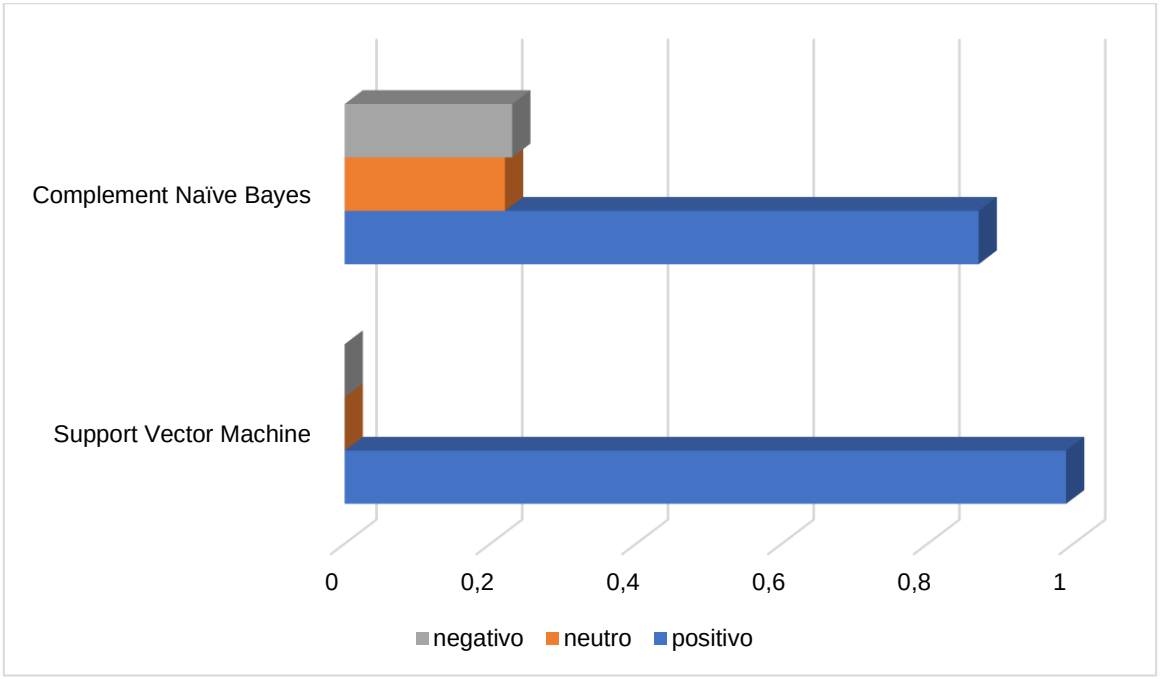


Figura 24

Comparativa de la métrica F1 score

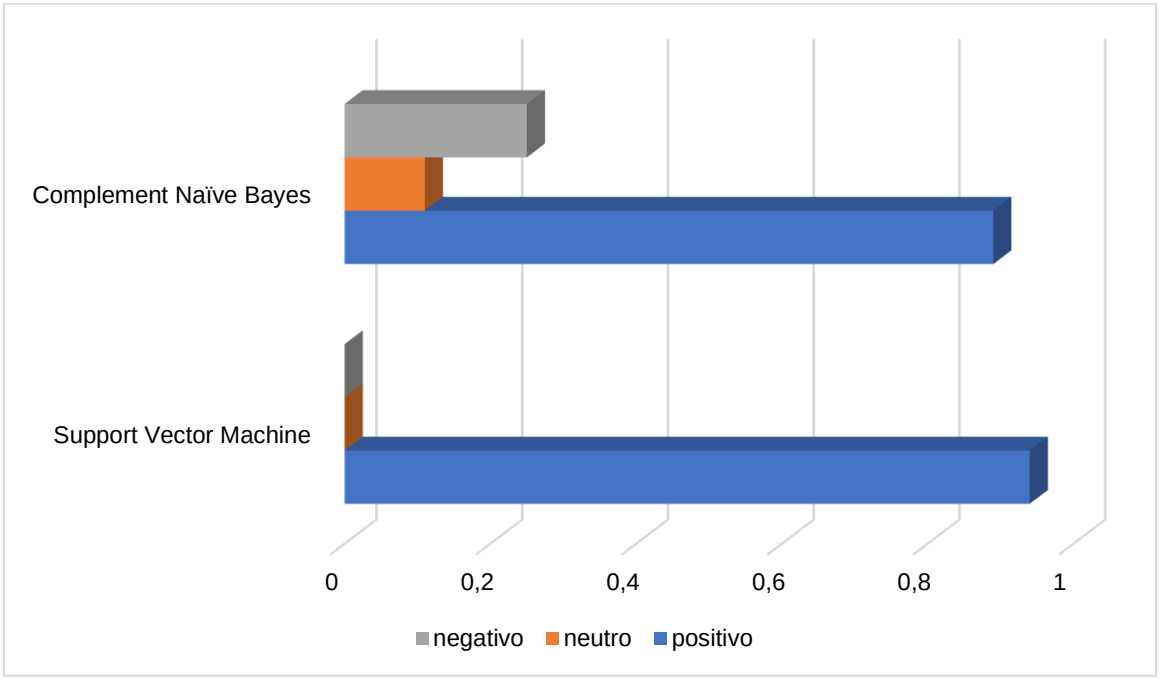


Tabla 27

Comparación de los promedios de las métricas.

	Precisión	Recall	F1- Score
Support Vector Machine	0.30	0.33	0.31
Complement Naïve Bayes	0.41	0.44	0.42

Nota. En esta tabla se observa que Complement Naïve Bayes tiene mejor rendimiento.

En el apéndice 6 y 7 se encuentran los enlaces que contienen el código de estas implementaciones. Una vez analizados los resultados se puede concluir que el algoritmo que mejor rendimiento tiene es el Complement Naïve Bayes. Esto tiene bastante lógica con la teoría que se mencionó anteriormente. Este algoritmo en concreto ha sido optimizado para que funcione con clases desbalanceadas. ¿Pero qué pasa si se aplican técnicas para resolver el problema de desequilibrio de datos? ¿los resultados seguirán siendo favorables para el algoritmo Complement Naïve bayes?. Esta comparativa se la resuelve en el siguiente apartado.

4.1.4. Estrategias para resolver desequilibrio de datos

a) La primera estrategia es conseguir más datos de las clases desbalanceadas, en este caso en particular, esta estrategia no es posible ya que el corpus textual es limitado y no se tiene más fuentes de datos.

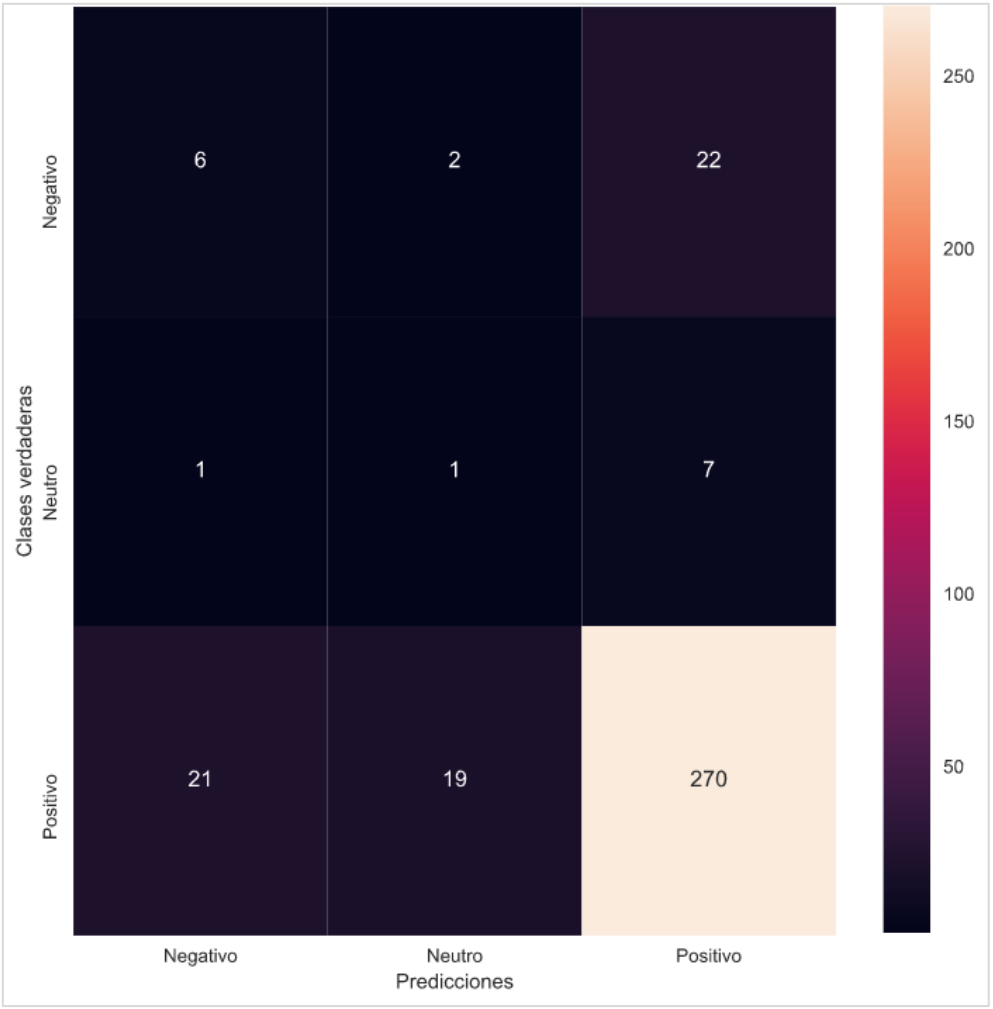
b) Una segunda estrategia que si es aplicable a este contexto es añadir un hiper – parámetro al clasificador Support Vector Machine, con el cual automáticamente el algoritmo realiza penalizaciones para compensar las clases minoritarias durante el entrenamiento. El hiper – parámetro es el siguiente:

```
class_weight = "balanced"
```

Una vez que se ha creado el clasificador con este hiper – parámetro adicional, se procede a entrenar y probar el modelo. La realización de estos pasos ya ha sido explicada anteriormente. Los resultados aplicando esta estrategia al algoritmo Suppport Vector Machine son los siguientes:

Figura 25

Matriz de confusión del algoritmo SVM con hiper – parámetro class_weight



En comparación con los resultados mostrados en la figura 23, se ve que hay una notable mejora. En esta ocasión el algoritmo clasifica 270 verdaderos positivos, 1 verdadero neutro y 6 verdaderos negativos. Por otro lado 21 instancias que son positivas el modelo las clasifica como negativas, 19 instancias que son positivas las clasifica como neutras. 1 instancias que es neutra la clasifica como negativa, 7 instancias que son neutras las clasifica como positivas. 2 instancias negativas las clasifica como neutras y 22 instancias negativas las clasifica como positivas.

A continuación, se presenta el reporte con las otras métricas:

Tabla 28

Reporte de clasificación con estrategia de hiper - parámetros

	Precision	Recall	F1-score	support
Negativo	0.21	0.20	0.21	30
Neutro	0.05	0.11	0.06	9
Positivo	0.90	0.87	0.89	310
Accuary			0.79	349
Macro avg	0.39	0.39	0.39	349
Weighted avg	0.82	0.79	0.81	349

Al finalizar la sección se compararán todas las métricas obtenidas.

c) Otra estrategia para mitigar el desequilibrio de los datos es el Submuestreo Aleatorio, la cual consiste en eliminar instancias de las clases más grandes e igualarla con la clase que menos instancias tiene. Para ello se utiliza la biblioteca *imbalanced-learn* específicamente el método `NearMiss()`. El código de todo este procedimiento se encuentra en el apéndice 8. Los resultados de cómo queda distribuido el dataset se muestran a continuación:

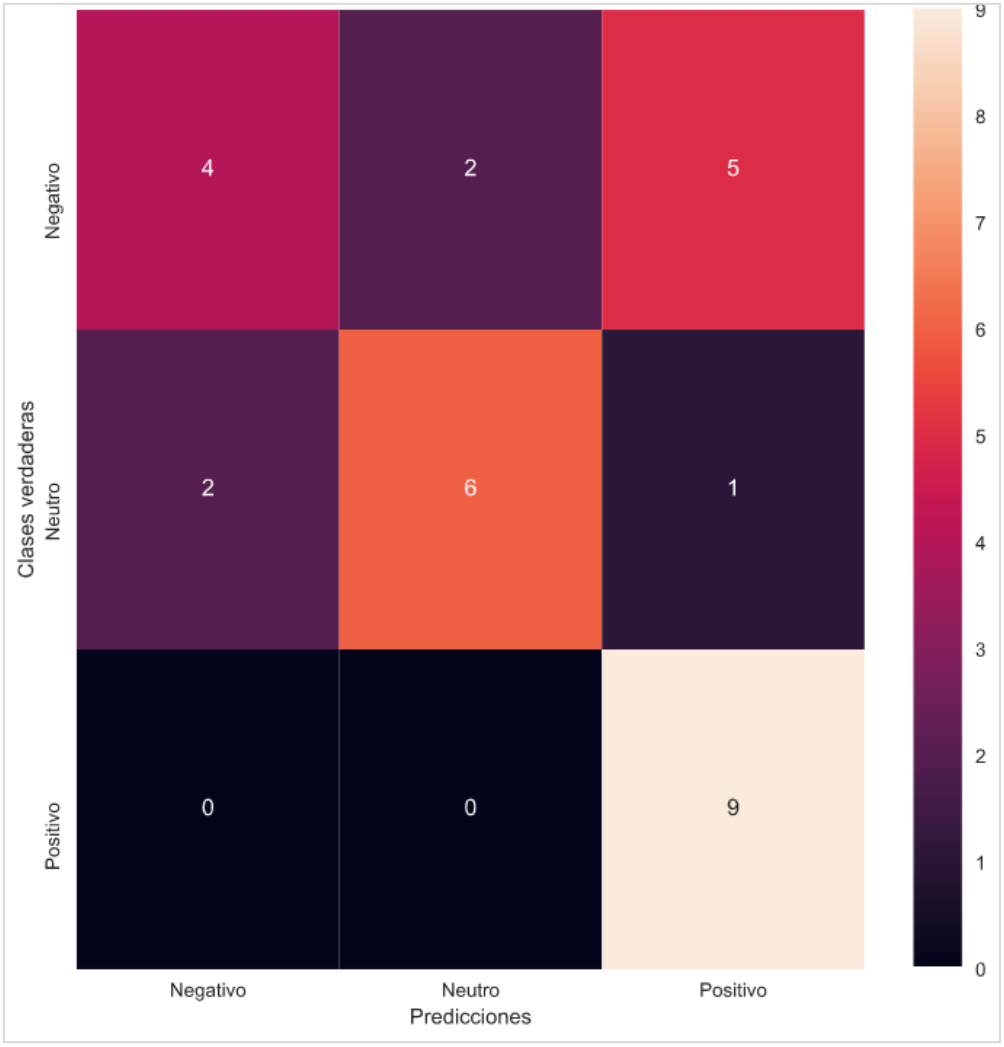
Distribución antes de submuestreo ({'positivo': 1563, 'negativo': 134, 'neutro': 47})

Distribución después de submuestreo ({'negativo': 47, 'neutro': 47, 'positivo': 47})

Con esta nueva distribución del dataset se procede a dividir los datos en conjunto de entrenamiento y conjunto de pruebas, luego se entrena y prueba el clasificador. Los resultados son los siguientes:

Figura 26

Matriz de confusión de SVM aplicada la técnica de submuestreo



En la matriz de confusión se observa que el modelo clasificó 9 instancias verdaderas positivas, 6 instancias verdaderas neutras y 4 instancias verdaderas negativas. Por otro lado, se tiene que el modelo clasificó 2 instancias neutras como negativas y una instancia neutra como positiva. Finalmente, el modelo clasificó 2 instancias negativas como neutras y 5 instancias negativas como positivas.

El reporte de los otros parámetros generados a partir de los valores de la matriz de confusión se presentan a continuación:

Tabla 29

Reporte de clasificación con submuestreo aleatorio

	Precision	Recall	F1-score	support
Negativo	0.67	0.36	0.47	11
Neutro	0.75	0.67	0.71	9
Positivo	0.60	1.00	0.75	9
Accuary			0.66	29
Macro avg	0.67	0.68	0.64	29
Weighted avg	0.67	0.66	0.63	29

d) Otra técnica es el sobremuestreo aleatorio. Consiste en crear copias de las clases minoritarias con el objetivo de igualarse con la clase mayoritaria. Este proceso se lo hace por medio del método `RandomOverSampler()` de la biblioteca *imbalanced-learn.* Los resultados obtenidos luego de realizar este proceso son los siguientes:

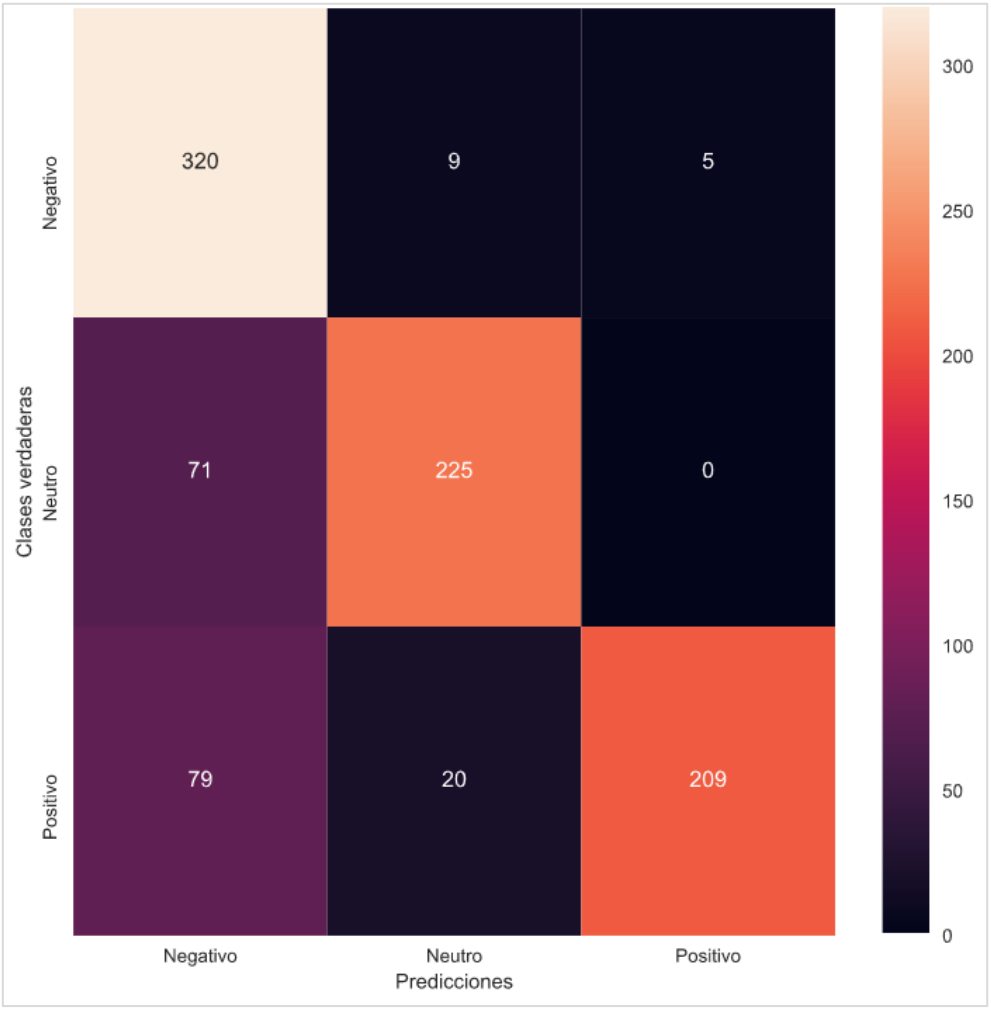
Distribución antes del sobremuestreo ({'positivo': 1563, 'negativo': 134, 'neutro': 47})

Distribución después del sobremuestreo ({'neutro': 1563, 'positivo': 1563, 'negativo': 1563})

Luego de esto se realiza el proceso de dividir los datos en un conjunto de entrenamiento y otro de pruebas. Finalmente, se entrena y prueba el clasificador. Los resultados son los siguientes.

Figura 27

Matriz de confusión de SVM aplicada la técnica de sobremuestreo



La matriz de confusión muestra que con la técnica de sobremuestreo el modelo clasificó 209 instancias verdaderas positivas, 225 verdaderas neutras y 320 verdaderas negativas. 20 instancias positivas el modelo las clasificó como neutras, 79 instancias positivas el modelo las clasificó como negativas. 71 instancias neutras fueron clasificadas como negativas. 9 instancias negativas fueron clasificadas como neutras y 5 instancias negativas fueron clasificadas como positivas.

El reporte de las métricas de clasificación es el siguiente:

Tabla 30

Reporte de clasificación con sobremustreo aleatorio

	Precision	Recall	F1-score	support
Negativo	0.68	0.96	0.80	334
Neutro	0.89	0.76	0.82	296
Positivo	0.98	0.68	0.80	308
Accuary			0.80	938
Macro avg	0.85	0.80	0.80	938
Weighted avg	0.84	0.80	0.80	938

e) Otra técnica consiste en combinar el submuestreo con el sobremuestreo, esto se logra con el algoritmo *Smote-Tomek* el método para utilizar es `SMOTETomek().` La nueva distribución una vez que se aplica este método es la siguiente:

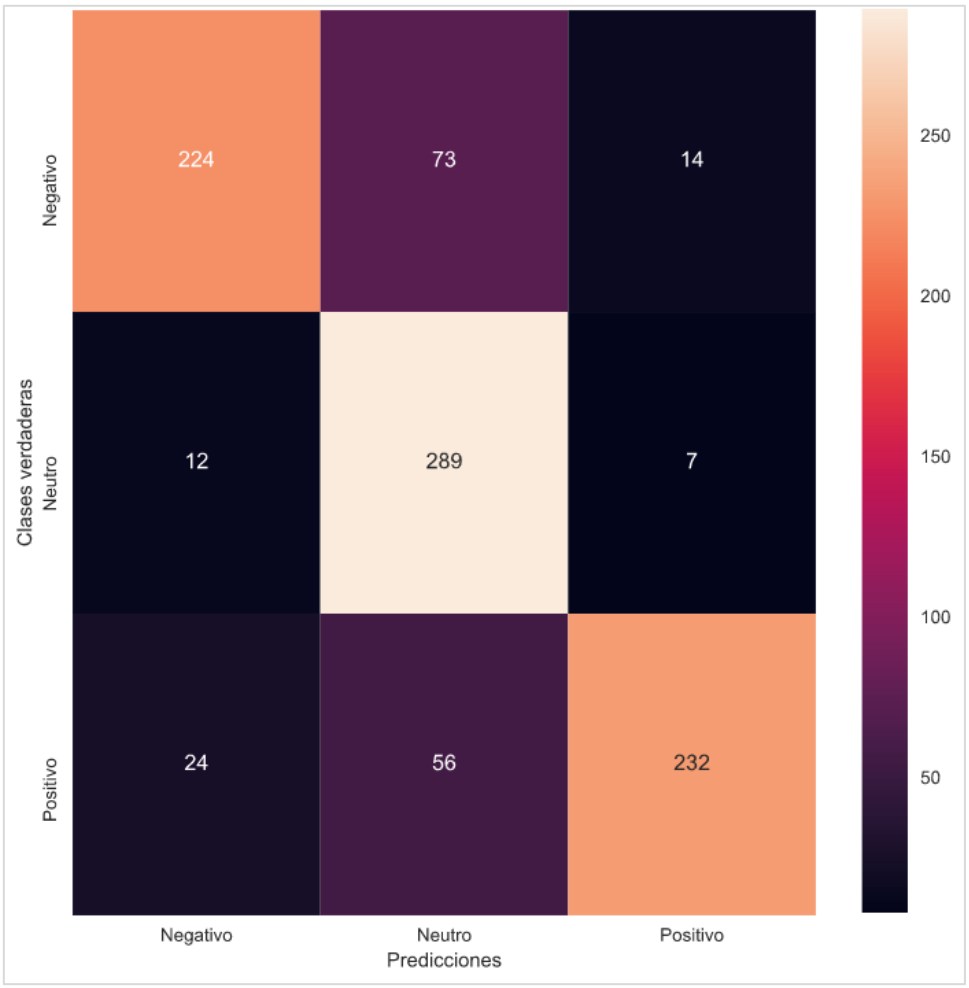
Distribución antes del remuestreo ({'positivo': 1563, 'negativo': 134, 'neutro': 47})

Distribución después del remuestreo ({'positivo': 1561, 'neutro': 1547, 'negativo': 1545})

Luego de esto se realiza el proceso de dividir los datos en un conjunto de entrenamiento y otro de pruebas. Finalmente se entrena y prueba el clasificador. Los resultados son los siguientes.

Figura 28

Matriz de confusión de SVM aplicado el remuestreo Smote-Tomek



Los resultados indican que 232 instancias han sido verdaderos positivas, 289 verdadero neutras y 224 Verdaderos negativas. 56 instancias positivas han sido clasificadas como neutras, 24 instancias positivas han sido clasificadas como negativas. 12 instancias neutras han sido clasificadas como negativas y 7 instancias neutras como positivas. Finalmente 73 instancias negativas han sido clasificadas como neutras y 14 instancias negativas como positivas. El reporte de las métricas se muestra a continuación:

Tabla 31

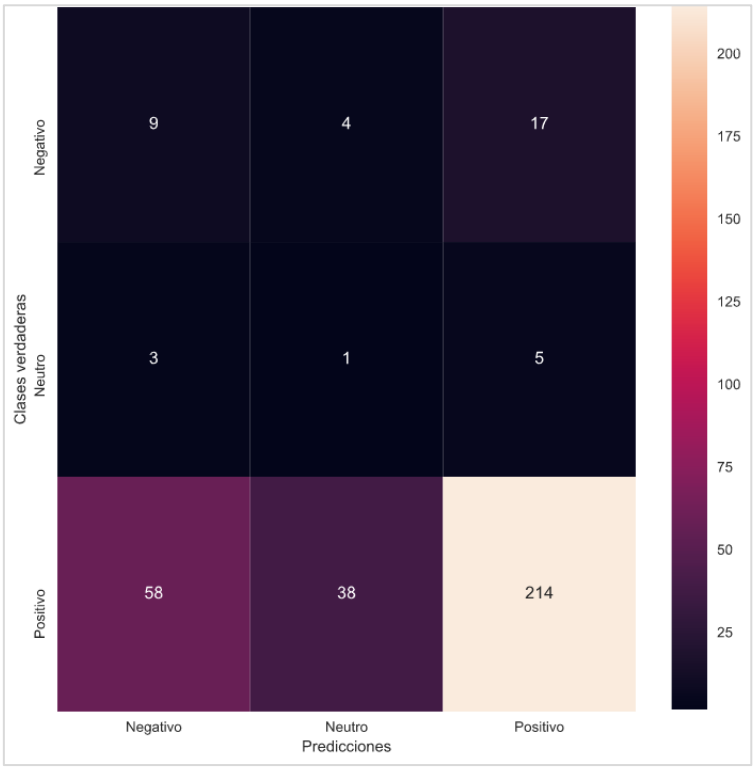
Reporte de clasificación con Smote - Tomek

	Precision	Recall	F1-score	support
Negativo	0.86	0.72	0.78	311
Neutro	0.69	0.94	0.80	308
Positivo	0.92	0.74	0.82	312
Accuary			0.80	931
Macro avg	0.82	0.80	0.80	931
Weighted avg	0.82	0.80	0.80	931

f) Finalmente, una última técnica aplicada al desbalanceo de los datos es la de *ensamblar modelos con balanceo.* Consiste en crear el modelo dentro de un método, específicamente el método `BalancedBaggingClassifier()` el cual se encarga de penalizar las clases que están desbalanceadas. Los resultados una vez que se aplica esta técnica se muestran a continuación:

Figura 29

Matriz de confusión aplicando BalancedBaggingClassifier()



A continuación, se presenta el reporte de las métricas del clasificador:

Tabla 32

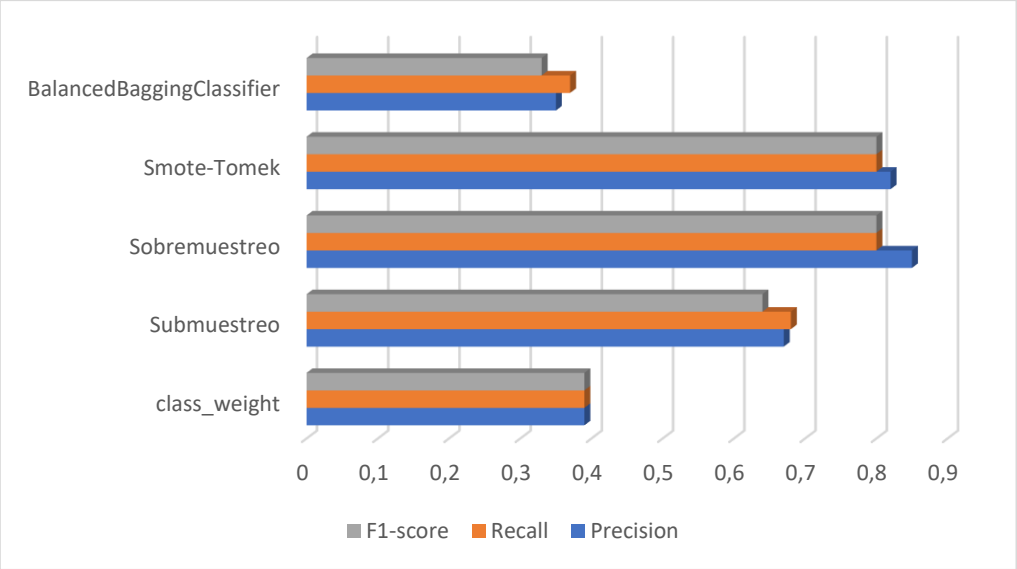
Reporte de clasificación con técnica de ensamblar modelos con balanceo

	Precision	Recall	F1-score	support
Negativo	0.13	0.30	0.18	30
Neutro	0.02	0.11	0.04	9
Positivo	0.91	0.69	0.78	310
Accuary			0.80	931
Macro avg	0.35	0.37	0.33	931
Weighted avg	0.82	0.64	0.71	931

Ahora es conveniente comparar los resultados obtenidos:

Figura 30

Comparativa de las técnicas para datos desbalanceados

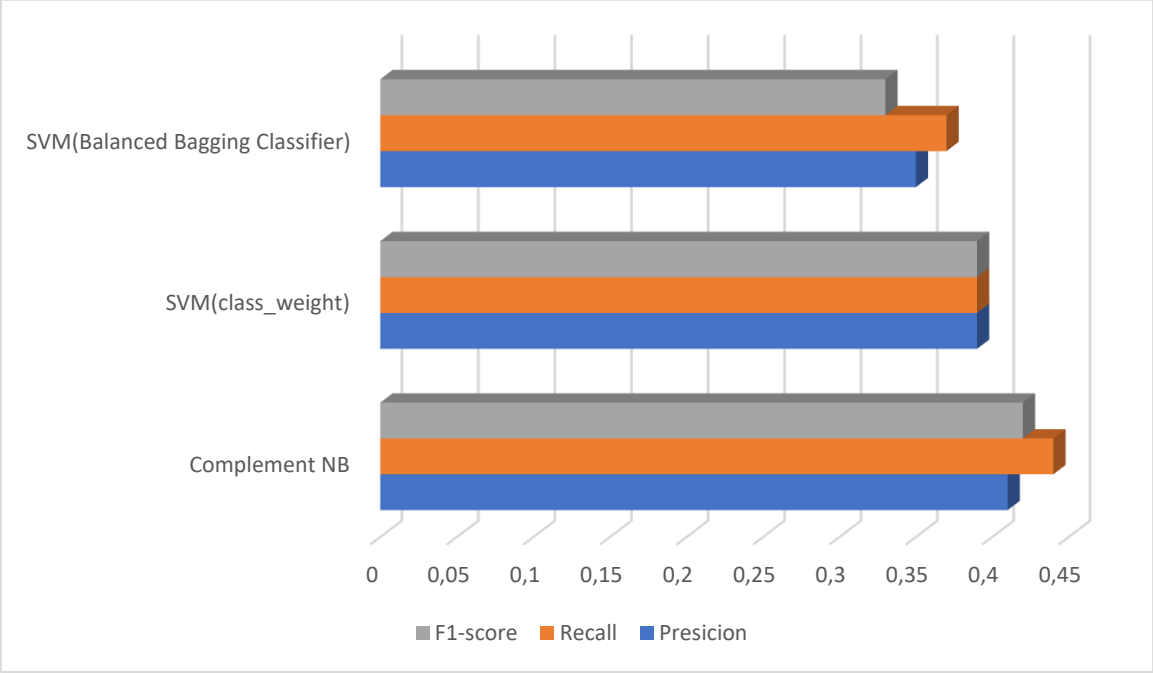


Con estos resultados se concluye que la técnica con la que se obtiene el mejor rendimiento para el algoritmo SVM es el sobremuestreo. Ahora bien hay un punto importante a tener en cuenta, tanto el sobremuestreo como el submuestreo y el método Smote-Tomek crean o eliminan instancias de las clases para lograr el balanceo deseado por lo que si comparamos la columna support que son las instancias totales de nuestro conjunto de datos

de prueba no coinciden en ninguna de estas técnicas por lo que los resultados para una comparación con otro algoritmo puede ser engañosas. Es por este motivo que se utiliza la técnica del hiper – parámetro class_weight y el ensamble de modelos con balanceo para comparar sus resultados contra el algoritmo Complement Naïve Bayes, ya que estás técnicas no eliminan ni crean instancias. El gráfico comparativo final se muestra a continuación:

Figura 31

Comparativa final de los resultados de clasificación



Observando los resultados anteriores se concluye que basados en la precisión, sensibilidad y el puntaje F1, el algoritmo que mejor rendimiento obtuvo es el Complement Naïve Bayes.

Conclusiones

Una vez que se ha finalizado el presente trabajo de fin de titulación las conclusiones son las siguientes:

El proceso de clasificación manual de los comentarios de acuerdo a la polaridad (positivos, negativos, neutros) fue bastante bueno. Esto se lo pudo validar con la herramienta de Google *Cloud Natural Language.*

La metodología elegida permitió desarrollar la investigación de manera satisfactoria, para esto al problema general se lo desglosó en distintas fases que facilitaron su desarrollo.

El corpus textual obtenido es pequeño y desbalanceado, con una clara tendencia de comentarios clasificados como positivos. Se aplicaron técnicas de desbalanceo de datos para mitigar este inconveniente; sin embargo, los resultados siguen siendo poco fiables ya que al probar el algoritmo con nuevos comentarios, hay una alta probabilidad de que su resultado sea positivo.

Para el corpus textual el algoritmo que mejor rendimiento obtuvo es el Complement Naïve Bayes ya que presenta valores más cercanos a uno en métricas de: precisión, sensibilidad y puntaje F1.

Los estudiantes muestran actitudes positivas al utilizar la red social Facebook en procesos de enseñanza – aprendizaje, puesto que el 8% de comentarios son negativos.

Recomendaciones

Finalizado el presente trabajo de fin de titulación las recomendaciones son las siguientes:

Para realizar el proceso de clasificación de datos es recomendable que al menos dos personas ajenas a la investigación lo ejecuten; además, es conveniente realizar su validación con alguna herramienta existente en el mercado, como por ejemplo: Google Cloud NLP.

Para hacer análisis de sentimientos se recomienda seleccionar la metodología específica propuesta por M.Sukanya (2012) que consta de cuatro fases, obteniéndose así un mejor control del proceso de desarrollo.

Se recomienda recopilar más datos con polaridad negativa y neutra, con el fin de balancear el dataset.

Para aplicaciones sobre análisis de sentimientos con corpus textuales en español se recomienda probar algunos algoritmos y seleccionar el que mejor se ajuste al corpus; así como, al objetivo de la investigación.

Para hacer un verdadero análisis de sentimientos usando redes sociales, en procesos de enseñanza – aprendizaje, se recomienda que el docente no participe en los grupos de estudio puesto que su presencia puede condicionar las expresiones de los estudiantes.

Para la continuidad de esta investigación se recomienda nutrir el dataset con más datos de tal manera que sea posible la utilización de otros algoritmos como por ejemplo las redes neuronales.

Referencias

Aggarwal, C. (2015). *Data Mining*. Springer, Cham. https://doi.org/10.1007/978-3-319-14142-8

Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). *A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques*. http://arxiv.org/abs/1707.02919

amazon. (2020). *Amazon Comprehend Descubra información y relaciones en un texto*. https://aws.amazon.com/es/comprehend/

Arantxa, V., & Aguaded, I. (2020). Análisis de sentimiento en Instagram: polaridad y subjetividad de cuentas infantiles. *ZER - Revista de Estudios de Comunicación*, *25*(48), 213–229. https://doi.org/10.1387/zer.21454

Armstrong, M. (2019). *How Many Websites Are There?* | *Statista*. https://www.statista.com/chart/19058/how-many-websites-are-there/

Banchs, R. E. (2013). *Text Mining with MATLAB®* (1.ª ed.). Springer-Verlag New York. https://doi.org/10.1007/978-1-4614-4151-9

Barrera, M. C. (2016). Minería de texto en la clasificación de material bibliográfico. *Biblios*, *64*, 33–43. https://doi.org/doi.org/10.5195/biblios.2016.309

Baviera, T. (2017). Técnicas para el Análisis de Sentimiento en Twitter: Aprendizaje Automático Supervisado y SentiStrength. *Dígitos: Revista de Comunicación Digital*, *1*(3), 33–50. https://doi.org/10.7203/rd.v1i3.74

Bramer, M. (2016). *Principles of Data Mining* (3.ª ed.). https://doi.org/10.1007/978-1-4471-7307-6

Bravo, F., Mendoza, M., & Poblete, B. (2014). Meta-level sentiment models for big social data analysis. *Knowledge-Based Systems*, *69*(1), 86–99. https://doi.org/doi.org/10.1016/j.knosys.2014.05.016

Castellanos Domínguez, M. I., Pérez Perdomo, E., Rivas Méndez, A., & Góngora Zaldívar, V. J. (2017). Biblioteca digital con técnicas de clasificación automática de documentos. *VIII Conferencia Científica Internacional de la Universidad de Holguín, Cuba*, *April*. https://www.researchgate.net/publication/316860869_Biblioteca_digital_con_tecnicas_de_clasificacion_automatica_de_documentos

Cifuentes, F. (2016). *Clasificación automática de Tweets utilizando K-NN y K-Means como algoritmos de clasificación automática, aplicando TF-IDF y TF-RFL para las ponderaciones* [Pontificia Universidad Católica de Valparaíso]. http://opac.pucv.cl/pucv_txt/Txt-8500/UCD8528_01.pdf

Dreyfus, G. (2005). *Neural Networks: Methodology and Applications* (1.ª ed.). Springer-Verlag Berlin Heidelberg. https://doi.org/10.1007/3-540-28847-3

García, A., Ríos, A., Tello, E., Barrón, J., & Díaz, A. (2018). *Aplicación de una red neuronal artificial para la clasificación automática de tweets en español*. *40*(130). http://www.itcelaya.edu.mx/ojs/index.php/pistas/article/view/1737

Godoy, A. (2017). Técnicas de aprendizaje de máquina utilizadas para la minería de texto. *Investigacion Bibliotecologica*, 24. https://doi.org/10.22201/iibi.0187358xp.2017.71.57812

Google. (2020). *Natural Language*. https://cloud.google.com/natural-language

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques* (3.ª ed.). https://doi.org/10.1016/C2009-0-61819-5

Hu, X., & Liu, H. (2012). *Mining Text Data* (C. C. Aggarwal & C. Zhai (eds.); 1.ª ed.). Springer-Verlag New York. https://doi.org/10.1007/978-1-4614-3223-4

Internet-live-stats. (s/f). *Internet Live Stats - Internet Usage & Social Media Statistics*. https://www.internetlivestats.com/

Justicia De La Torre, C., Martín-Bautista, M. J., Síanchez, D., & Vila, M. A. (2005). *Text mining: intermediate forms on knowledge representation*. 7. https://www.researchgate.net/publication/221399082_Text_mining_intermediate_forms_on_knowledge_representation

Justicia de la Torre, M. del C. (2017). *Nuevas Tecnicas De Minería De Textos: Aplicaciones* [Universidad de Granada]. https://digibug.ugr.es/handle/10481/46975

Kluyver, T., Benjamin Ragan-Kelley, Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). *Project Jupyter | Home*. Jupyter Notebooks -- a publishing format for reproducible computational workflows. https://jupyter.org/

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Moldagulova, A., & Sulaiman, R. B. (2017). Using KNN algorithm for classification of textual documents. *ICIT 2017 - 8th International Conference on Information Technology, Proceedings*, 665–671. https://doi.org/10.1109/ICITECH.2017.8079924

MonkeyLearn. (2020). *Create new value from your data*. https://monkeylearn.com/

Mourya, S. K., & Gupta, S. (2012). *Data Mining and Data Warehousing*. https://search.proquest.com

NLTK Project. (2020). *NLTK documentation*. https://www.nltk.org/

Pitigala, S., Li, C., & Seo, S. (2011). A comparative study of text classification approaches for personalized retrieval in PubMed. *2011 IEEE International Conference on Bioinformatics and Biomedicine Workshops, BIBMW 2011*, 3. https://doi.org/10.1109/BIBMW.2011.6112503

Pranckevičius, T., & Marcinkevičius, V. (2017). Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Baltic Journal of Modern Computing*, *5*(2), 221–232. https://doi.org/10.22364/bjmc.2017.5.2.05

Reyes, G. C., González, Y. G., & Farias, G. L. (2014). Técnica de clasificación bayesiana para identificar posible plagio en información textual. *Revista Cubana de Ciencias Informáticas*, *8*(4), 130–144. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992014000400008&lng=es&tlng=es

Rosa, A., Chiruzzo, L., Etcheverry, M., & Castro, S. (2017). *RETUYT in TASS 2017: Sentiment Analysis for Spanish Tweets using SVM and CNN*. *October*, 77–83. http://arxiv.org/abs/1710.06393

Russell, S. J., & Norvig, P. (2004). *Inteligencia Artificial: Un Enfoque Moderno* (2.ª ed.). PEARSON EDUCACIÓN, S.A.,.

scikit-learn. (s/f). *scikit-learn Machine Learning in Python*. https://scikit-learn.org/

Seperak, R., Cerellino, L., Ochoa, J., Torres-Valer, A., & Dianderes, C. (2019). Maternidad en Perú a través del uso del Sentimient Analysis en Facebook. *Revista Latina de Comunicacion Social*, *74*, 1031–1055. https://doi.org/10.4185/RLCS-2019-1370

Shafiabady, N., Lee, L. H., Rajkumar, R., Kallimani, V. P., Akram, N. A., & Isa, D. (2016). Using unsupervised clustering approach to train the Support Vector Machine for text classification. *Neurocomputing*, *211*, 4–10. https://doi.org/10.1016/j.neucom.2015.10.137

Taeho, J. (2019). *Text Mining* (1.ª ed.). Springer International Publishing. https://doi.org/10.1007/978-3-319-91815-0

TensorFlow. (s/f). *Una plataforma de extremo a extremo de código abierto para el aprendizaje automático*. tensorflow.org

TextBlob. (2020). *TextBlob: Simplified Text Processing*. https://textblob.readthedocs.io/en/dev/

Weiss, S. M., Indurkhya, N., & Zhang, T. (2010). *Fundamentals of Predictive Text Mining* (1.ª ed.). Springer-Verlag London. https://doi.org/10.1007/978-1-84996-226-1

Weiss, S. M., Indurkhya, N., Zhang, T., & Damerau, F. (2005). *Text Mining* (1.ª ed.). Springer-Verlag New York. https://doi.org/10.1007/978-0-387-34555-0

Yu, W., & Song, X. (2010). Research on text categorization based on machine learning. *ICAMS 2010 - Proceedings of 2010 IEEE International Conference on Advanced Management Science*, 3. https://doi.org/10.1109/ICAMS.2010.5552917

Apéndice

Apéndice 1: Script desarrollado para crear el archivo CSV a partir de los datos en formato JSON.

https://github.com/iaastudillo/proyectoTT/blob/main/Code/infoJSON.py

Apéndice 2: Código desarrollado para el análisis de sentimientos con la herramienta Google Cloud NLP.

https://github.com/iaastudillo/proyectoTT/blob/main/Code/Sentiments_analysis_google.py

Apéndice 3: Código desarrollado para la exploración de los datos.

https://github.com/iaastudillo/proyectoTT/blob/main/Code/1)explorar_datos.ipynb

Apéndice 4: Código desarrollado para la fase de pre – procesamiento del texto.

https://github.com/iaastudillo/proyectoTT/blob/main/Code/2)pre-procesamiento.ipynb

Apéndice 5: Archivo que contiene el texto procesado.

https://github.com/iaastudillo/proyectoTT/blob/main/Code/Data_procesada/datos_procesados.csv

Apéndice 6: Código desarrollado para la implementación del algoritmo Support Vector Machine.

https://github.com/iaastudillo/proyectoTT/blob/main/Code/3)support_vector_machine.ipynb

Apéndice 7: Código desarrollado para la implementación del algoritmo Complement Naïve Bayes.

https://github.com/iaastudillo/proyectoTT/blob/main/Code/3)complement_naive_bayes.ipynb

Apéndice 8: Código desarrollado para la implementación de las estrategias para datos desbalanceados.

https://github.com/iaastudillo/proyectoTT/blob/main/Code/3)support_vector_machine.ipynb