UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional Ingeniería de Sistemas

Minería de opiniones basado en aprendizaje supervisado en la evaluación de destinos turísticos de la región de Puno

Por:

Roger Ticona Nina

Asesor:

Mg. Nilton Omar Santillan Aching

Co-Asesor:

Dra. Sc. Nélida Gladys Maquera Sosa

Juliaca, julio de 2019

DECLARACION JURADA
DE AUTORIA DEL INFORME DE TESIS

Mg. Nilton Omar Santillan Aching, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que el presente informe de investigación titulado: **"MINERÍA DE OPINIONES BASADO EN APRENDIZAJE SUPERVISADO EN LA EVALUACIÓN DE DESTINOS TURÍSTICOS DE LA REGIÓN DE PUNO"** constituye la memoria que presenta el **Bachiller Roger Ticona Nina** para aspirar al título Profesional de Ingeniero de Sistemas, cuya tesis ha sido realizada en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este informe son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en Juliaca a los dos días del mes de julio del año dos mil diecinueve.

Mg. Nilton Omar Santillan Aching

Minería de opiniones basado en aprendizaje supervisado en la evaluación de destinos turísticos de la región de Puno

TESIS

Presentada para optar el título profesional de Ingeniero de Sistemas

JURADO CALIFICADOR

Mg. Henry Lennin Centurión Julca
Presidente

Dr. Jorge Alejandro Sánchez Garcés
Secretario

Mg. Abel Angel Sullon Macalupu
Vocal

Mg. Nilton Omar Santillan Aching
Asesor

Juliaca, 02 de Julio de 2019

DEDICATORIA

Este trabajo se la dedico principalmente a Dios por su compañía y bendición divina.

A mi familia por ser el motor para alcanzar mis logros, en especial a ti "mamita" **Luisa Nina Velasque** por ser ese apoyo incondicional en cada etapa de mi vida, por motivarme en todo momento a que siga con mis sueños sin importar mis errores y enseñarme la valentía de seguir adelante a pesar de las adversidades.

A **la Dra. Sc. Nélida Gladys Maquera Sosa** por inculcarme el espíritu investigador e iniciar en la investigación científica, además de motivarme en mi superación personal.

AGRADECIMIENTOS

Doy gracias a Dios por protegerme durante mi camino y darme fuerzas para superar obstáculos y dificultades.

Al Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC), por promover e impulsar la investigación científica

A mi madre que, con su apoyo incondicional, cuidado y ejemplo me ha inculcado que los anhelos y metas con esfuerzo y dedicación se llegan alcanzar.

A mis hermanos y hermanas quienes con su apoyos y consejo me brindan fortaleza para seguir adelante.

A mi asesor de tesis**, Mg. Nilton Omar Santillan Aching** por brindarme su conocimiento, por su guía en el desarrollo de este proyecto con disposición, confianza y apoyo profesional.

Un profundo agradecimiento a **Dra.Sc. Nélida Gladys Maquera Sosa** por el apoyo y colaboración en el desarrollo de este proyecto.

A mis amigos y compañeros que de una u otra manera con su colaboración y apoyo formaron parte de este logro. Les deseo todo el éxito en el campo profesional, quiénes permanecieron pendientes pese a la distancia. Gracias por su amistad y cariño, mil gracias…

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

ABREVIATURAS Y ACRÓNIMOS

- AS: Abreviatura de Análisis de sentimientos
- MO: Minería de Opiniones
- SWBC : Abreviatura en ingles de Spanish Billion Word Corpus
- PLN: Procesamiento de lenguaje natural
- IA: inteligencia Artificial
- SVM: Abreviatura en ingles de Support Vector Machines
- POS: Abreviatura en ingles de Part Of Speech Tagging
- CV: Abreviatura en ingles de Cross Validation
- MINCETUR: Abreviatura de Ministerio de Comercio Exterior y Turismo

RESUMEN

La minería de Opiniones o Analisis de sentimento, se encuentra dentro del área de investigación del porcesamiento de Lenguaje natural, y consiste en la extracción, identificación y análisis de opiniones emitidas por ususarios en forma de textos sobre una determinada entidad, con el fin de clasificarlos mediante un procesamiento computacional, según la polaridad del sentmiento expresado. Esta tesis tiene como objetivo de investigación la utilización de las técnicas y procedimientos necesarios para realizar Minerías de opiniones basado en aprendizaje supervisado con el fin de realizar una evaluación de las opiniones en línea elaboradas por los usuarios de diversos destinos turísticos de la región de Puno. Esta tarea se realiza utilizando la metodología CRISP-DM, cuyos procesos permitieron realizar las acciones de comprensión del negocio, extracción de los datos procedentes de la web de viajes tripadvisor, comprensión de los datos, realizar el pre procesamiento de los datos, modelamiento de los modelos wor2vec necesarios para la extracción de representaciones vectoriales de los comentarios, evaluar la clasificación de polaridad entre "positiva" y "negativa" con la utilización del algoritmo SVM y finalmente analizar los resultados. De las cuales se vio que gracias a la utilización de representaciones vectoriales de los comentarios se puede obtener buenas clasificaciones de polaridad, además de que los modelos wor2vec son versátiles a la hora de adaptarse a diferentes dominios dependiendo de la cantidad de textos de entrenamiento. Luego del estudio se vio que los destinos turísticos de la región de Puno tienden a tener comentarios positivos en su mayoría, reflejando la satisfacción del turista luego de visitar Puno.

Palabras clave: Minería de opiniones, Word2vec, Support Vector Machines, Crisp-DM, Procesamiento de lenguaje natural, Destinos turísticos

ABSTRACT

The Opinion mining or sentiment analysis, is within the area of investigation of the natural language processing, and consists of the extraction, identification and analysis of opinions issued by users in the form of texts on a certain entity, in order to classify them through computational processing, according to the polarity of the expressed feeling. This thesis aims to research the use of techniques and procedures necessary to conduct Mining opinions based on supervised learning in order to make an assessment of online opinions prepared by users of various tourist destinations in the Puno region. This task is carried out using the CRISP-DM methodology, whose processes allowed to carry out the business comprehension actions, extraction of the data coming from the tripadvisor travel website, understanding of the data, pre-processing of the data, modeling of the wor2vec models needed for the extraction of vector representations of the comments, evaluate the polarity classification between "positive" and "negative" with the use of the SVM algorithm and finally analyze the results. Of which it was seen that thanks to the use of vectorial representations of the comments, good polarity classifications can be obtained, besides that the wor2vec models are versatile when adapting to different domains depending on the amount of training texts. After the study it was seen that the tourist destinations of the Puno region tend to have positive comments in their majority, reflecting the satisfaction of the tourist after visiting Puno.

Keywords: Opinions Mining, Word2vec, Support Vector Machines, Crisp-DM, Natural language processing, Turism Destinations

CAPÍTULO I. El problema

1.1. Descripción del problema

En el Perú una de las actividades económicas principales es el turismo, debido a que esta actividad proporciona un gran aporte en la generación de empleos, y la sostenibilidad de comunidades, por esto es considerada por los expertos como uno de los países con mayor producto turístico disponible, junto con México, España, Francia, Italia y Egipto (Villanueva, 2007).

En este sentido la región de Puno siendo la 5ta más grande del país, tiene una diversidad de productos agropecuarios, culturas pre-incas, como Tiahuanaco, Lupaca, además de paisajes imponentes y una arquitectura colonial manifestada en templos, casonas, balcones, etc., que hacen de la región un potencial turístico de nivel internacional (Gobierno Regional de Puno, 2011). Es por esto que, en el informe elaborado por el PROMPERU, acerca del perfil del visitante extranjero para el 2017, sitúa a Puno como la tercera región más visitada, gracias a sus diferentes destinos turísticos que posee. Siendo un destino turístico como un conjunto de productos que integran la experiencia turística, suministrados por un conjunto de actores (tour operadores, hoteleros, transportistas, restauranteros, etc.) que cooperan entre sí (Molinar, Espinoza, & Llamas, 2017).

Por otra parte, el surgimiento de la denominada web 2.0 origino que, las personas puedan compartir y debatir libremente sus opiniones positivas o negativas sobre un determinado tema en sitios de internet como, redes sociales, redes de microblog, foros o sitios de revisión de productos (Ding, Li, Zhao, & Cheng, 2017). Y como resultado de, este uso masivo de redes sociales, fue que en el sector del turismo emerjan los sitios webs enfocadas a este dominio, convirtiéndose cada vez más en una fuente de conocimientos incalculable (Sangeetha, 2017).

Debido a que gracias estos comentarios en línea, realizados por viajeros que están dispuestos a compartir sus experiencias, los nuevos viajeros pueden recoger las opiniones de otros para que de esta manera encuentren destinos, consejos, ofertas y la posibilidad de planificar de forma integral sus vacaciones. Según PROMPERU (2017) el 72% de los

viajeros que arribaron a la región de Puno en el 2017 tomaron como medio más influyente para la elección del destino turístico a Internet, siendo en su mayoría personas pertenecientes a la generación “millennials”. Esto muestra el crecimiento acelerado del uso de las webs de turismo por los viajeros y el aumento diario de revisiones o comentarios en línea respecto a los destinos turísticos que visitaron.

Según Rosso, Locoro, Mascardi, & Balaguer (2010) la principal fuente de información para conocer los puntos fuertes y débiles de una organización es a través de las opiniones que generan los propios consumidores. Por ello hoy en día en el sector turístico y particularmente en la actividad turística de la región de Puno existe la necesidad del aprovechamiento del gran volumen de información contenida en los numerosos sitios webs de turismo existentes que ofrecen opiniones de los viajeros, además de que esta información sea analizada por procesos automáticos, ya que realizarla de manera manual es inviable. En este entorno una de las técnicas para este aprovechamiento del gran volumen de información existente es el análisis del sentimiento (SA) o minería de opiniones (OM) perteneciente a la rama del Procesamiento del Lenguaje Natural (PLN), disciplina que combina el proceso de Inteligencia Artificial y la Lingüística Computacional para la recuperación de información, extracción de texto y detectar masivamente el significado residente en el lenguaje natural o humano. Cuyo objetivo principal es determinar si la parte del contenido es positiva, negativa o neutra (Afzaal & Usman, 2016).

Se considera que la presente investigación proporcionara una alternativa de solución que ayudara a conocer la percepción del turista acerca del destino turístico.

1.2. Objetivos.

1.2.1. Objetivo general

Aplicar las técnicas de Minería de Opiniones basada en aprendizaje supervisado para la evaluación de los comentarios en línea de los destinos turísticos de la región Puno

1.2.2. Objetivos específicos

- Extraer los comentarios existentes de los destinos turísticos más visitados de la región de Puno mediante las técnicas de web scraping.

- Implementar el módulo de pre procesamiento de lenguaje, utilizando las técnicas de Segmentación, Normalización, Etiquetado, Lematización y eliminación de stopwords.

- Implementar el módulo de entrenamiento del modelo word2vec y extraer las representaciones vectoriales de los comentarios.
- Implementar el módulo de entrenamiento de Las Máquinas de Vectores Soporte.
- Evaluar resultados de clasificación.

1.3. Justificación de la investigación.

La región de Puno posee un número considerable de maravillosos destinos turísticos, razón por la cual son visitados por los viajeros de diferentes partes del mundo, según el (PROMPERU, 2017) en el 2017 se tuvo un total de 2,5 millones de arribos de vacacionistas al Perú, desde cuatro zonas primordiales "Norteamérica"," Latinoamérica", "Europa", "Asia" y "Oceanía". Además, en el mismo año se registró la mayor cantidad de visitas en los destinos turísticos de, la ciudad de Puno, Lago Titicaca, Isla de los Uros, Isla Taquile e Isla Amantaní con 80%, 73%, 72%, 42% y 17 % respectivamente. Al mismo tiempo la entidad estimo que el 72% de todos los viajeros que arribaron a la región en el mismo año, tomaron como medio más influyente para la elección del destino turístico a "INTERNET", así mismo el 31% de viajeros tomaron como otro medio influyente las recomendaciones de sus amigos.

Esta Tendencia de utilizar "INTERNET" como medio más influyente ha ido en aumento los últimos años, gracias la adopción masiva de las nuevas tecnologías por parte de las nuevas generaciones, como los millennials, que son una generación nacida entre los años 1979 y 1994 y prefieren revisar en internet las experiencias y comentarios de las personas antes de consultar en medios masivos o agencias de viajes. Por lo que hace suponer que principales fuentes de las nuevas generaciones para la planificación de sus viajes son el gran cumulo de información que se encuentran en foros, redes sociales, redes microbloggin, etc.

Así que cuando un turista busca un destino turístico de la ciudad de Puno, a menudo hace uso del internet para encontrarlo, y recurren directamente a los comentarios que hicieron los turistas que ya visitaron el lugar, en foros dedicadas a este rubro como tripadvisor, booking, etc. Cuando el comentario es positivo usualmente el turista escogerá el lugar como un futuro destino a visitar , en cambio sí en los comentarios el destino está considerada con una valoración negativa el turista no lo tomara en cuenta como un futuro destino a visitar, este fenómeno sucede debido a que en la vida cotidiana, las personas tienden a tomar las

opiniones de los demás como referencia para ayudarles a tomar sus propias decisiones (Ding et al., 2017). De modo que gracias a este flujo de personas dispuestas a compartir su opinión acerca de sus experiencias de visita, el crecimiento de información es exponencial, con opiniones valiosas listos para su análisis.

Con el objetivo de aprovechar la gran cantidad de opiniones disponibles y tener el análisis de los destinos turísticos, este trabajo busca utilizar las técnicas de minería de opiniones, el cual nos permite el procesamiento del lenguaje natural para rastrear el estado de ánimo del público sobre un lugar turístico en particular (Afzaal & Usman, 2016).

Según Shein & Nyunt (2010) la minería de opinión se refiere a la identificación de opiniones y argumentos en un texto e intenta encontrar las declaraciones de acuerdo o desacuerdo que se refieren a positivo, negativo o neutral en los comentarios, esta enfoqué surgió debido al amplio crecimiento de los comentarios en línea hechas por los turistas. Por lo que las investigaciones en análisis de sentimiento o minería de opiniones, ha sido una tendencia en los últimos años, debido a que, las comunidades investigadoras han dirigido sus miradas en la necesidad de aprovechar la inmensa producción de opiniones y comentarios disponibles en diversas páginas web, blogs y redes sociales disponibles en internet. Los investigadores Piryani, Madhavi, & Singh (2017) realizaron un estudio de la tendencia existente en este tema en un periodo de 16 años 2000 al 2015, lo que demuestra la importancia tecnología en su desarrollo Figura 1.

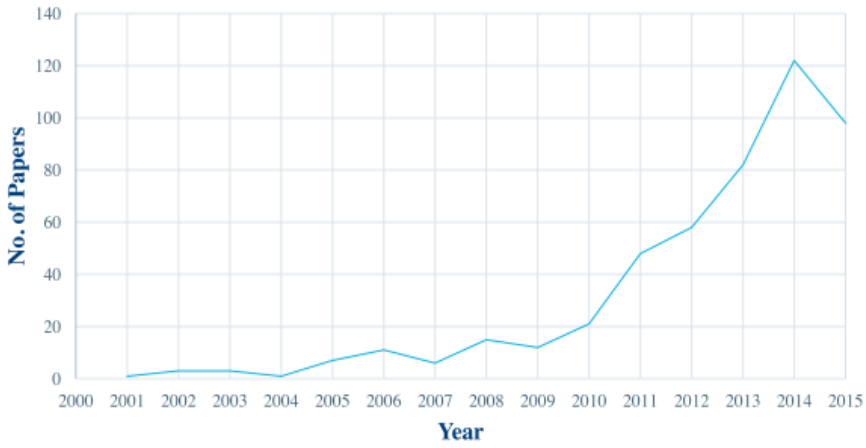


Figura 1. Cantidad de publicaciones en análisis de sentimientos, año 2000 - 2015.
Fuente: Piryani, Madhavi, & Singh, (2017), Analytical mapping of opinion mining and sentiment analysis research during 2000–2015.

Por este motivo la solución desarrollada será la utilización de Minería de Opiniones bajo el enfoque a nivel de documentos, y aprendizaje supervisado aplicada en el análisis de sentimientos de los destinos turísticos más visitados de la región de Puno, permitiendo mostrar la percepción general del destino turístico como positiva, negativa o neutra, para que de esta manera se pueda contribuir en las investigaciones realizadas bajo el enfoque de análisis de sentimientos.

CAPÍTULO II. Marco teórico

2.1. Revisión de la Literatura.

La investigación realizada por Machado (2018), con el título de "Análisis de sentimientos basado en opiniones turísticas" realizo un estudio cuyo objetivo fue la elaboración de un sistema que permite realizar el análisis de opiniones elaboradas por los usuarios en línea de diversos lugares como restaurantes, hoteles, parques y playas de toda la isla de Tenerife, el cual utilizo los procedimientos de Recolección de datos, Pre procesado de texto, elaboración del modelo wor2vec y clasificación de opiniones. Cuyos resultados fueron la elaboración del sistema de clasificación de opiniones, además de evaluar la clasificacion del modelo ente 5 categorías numéricas.

Dubiau (2013) realiza una investigación titulada "Procesamiento de Lenguaje Natural en Sistemas de Análisis de Sentimientos", cuyo objetivo fue realizar la investigación, análisis, evaluación y comparación experimental de técnicas de Procesamiento de Lenguaje Natural para análisis de información subjetiva como opiniones, sentimientos y emociones en textos no estructurados. Para esto utilizo una metodología propia de obtención de datos, pre procesado, selección de atributos y clasificación. Como resultado se logró realizar un sistema de análisis de sentimientos que contemple diversas técnicas de preprocesado además de diversas técnicas de clasificación como Naıve Bayes, MaxEnt, SVM Y Decisión Trees, para de esta manera verificar la performance de los resultados.

En el trabajo de Sande (2018), titulada "Análisis de sentimientos en Twitter", tuvo el objetivo de, explicar los fundamentos teóricos sobre los que se asienta el análisis de sentimientos, su historia, aplicaciones y su relación con el procesamiento del lenguaje natural además de la implementación de un clasificador de sentimientos para los mensajes de Twitter basado en algoritmos de aprendizaje supervisado y llevar a cabo un estudio comparativo con las técnicas más populares para el análisis de sentimientos a nivel de documento. Los procedimientos utilizados fueron la obtención de datos, pre procesamiento de datos y clasificación de polaridad. Donde se obtuvo un estudio completo de las técnicas

utilizadas en análisis de sentimiento, la implementación de un clasificador que permite realizar la comparación de los mejores algoritmos de clasificación de textos.

La investigación realizada por García (2016) el cual titula "Minería de textos y análisis de sentimientos en sanidadysalud.com", tiene el objetivo de realizar un sistema automatico que permita clasificar las opiniones como negativas y no negativas de las opiniones sobre centros de salud, hospitales y farmacias españolas recogidas del portal web sanidadysalud.com. Para esta tarea utilizo la metodología Crisp- DM cuyos procesos son la comprencion del negocio, comprencion de los datos, análisis de datos, modelado, evaluación y puesta en producción. Como resultado se obtuvo la construcción de un sistema de analsis de sentimientos y se encarga de clasificar las opiniones del portal web como positivas y negativas para posteriormente ser publicadas autolmaticamente.

Miranda (2017) en su trabajo titulado "Un modelo integrado de técnicas de aprendizaje de máquinas no supervisadas y ontologías para la detección automática de sentimientos desde una estructura gramatical simple en español". Donde tuvo el objetivo de Construir un modelo integrado de técnicas de aprendizaje de máquinas no supervisadas y ontologías para el análisis de sentimientos a nivel de características que permita la detección automática de aspectos explícitos e implícitos en una estructura gramatical simple en español, ademas de elaborar un sistema que permita esta clasificacion de sentimientos. Se utilizo Fase teoría y Fase de Implementación como la metodología del estudio. Obteniendo los siguientes resultados: El sistema obtuvo un 73.07 de valor F1 en el proceso de extracción de aspectos, un 46.2 en la extracción de la categoría (Entidad-atributo) e identificación de aspectos conjuntamente y un 84.8% de exactitud en la clasificación de sentimientos, además el sistema desarrollado AspectSA obtuvo mejores resultados sobre los sistemas participantes en la competición en los tres aspectos antes mencionados.

2.2. Procesamiento de Lenguaje Natural.

El termino Natural Language Processing (NLP) o procesamiento de lenguaje Natural (PLN) por sus siglas en inglés, es el enfoque de investigación que se basa en la utilización de herramientas y técnicas computacionales para el tratamiento automático del "lenguaje natural", diversos autores lo definen de la siguiente manera.

El PLN se concibe como el reconocimiento y utilización de la información expresada en lenguaje humano a través del uso de sistemas informático. Tanto desde un enfoque computacional como lingüístico se utilizan técnicas de inteligencia artificia (Sosa, 1997).

"El procesamiento del lenguaje natural es una gama de técnicas computacionales, motivadas teóricamente para analizar y representar textos naturales en uno o más niveles de análisis lingüístico, con el fin de lograr un procesamiento del lenguaje similar al humano para una gama de tareas o aplicaciones" (Liddy, 2001, p. 6).

"Procesamiento de Lenguaje Natural (PLN) es un área de investigación y aplicación que explora, como las computadoras pueden ser usados para entender y manipular el lenguaje natural, el texto o el habla para hacer cosas útiles" (Chowdhury, 2005, p. 7).

En definitiva, el principal objetivo de las tareas de PLN es aprovechar las expresiones del ser humano ya sea en el habla o en el texto, para su procesamiento, manipulación y análisis, por medio de las nuevas herramientas computacionales como la inteligencia artificial (IA).

2.2.1. Niveles del procesamiento de lenguaje natural.

Según Liddy (2001); Feldman (1999) el lenguaje natural posee 7 niveles de estudio, conocido también como "el modelo sincrónico del lenguaje" y son necesarios para que la extracción y comprensión de la información se realicen de una manera más adecuada, debido a que están relacionados entre sí, es decir existen niveles que necesitan de otros para permitir su análisis, a continuación, presentamos los niveles de análisis existentes Figura 2.

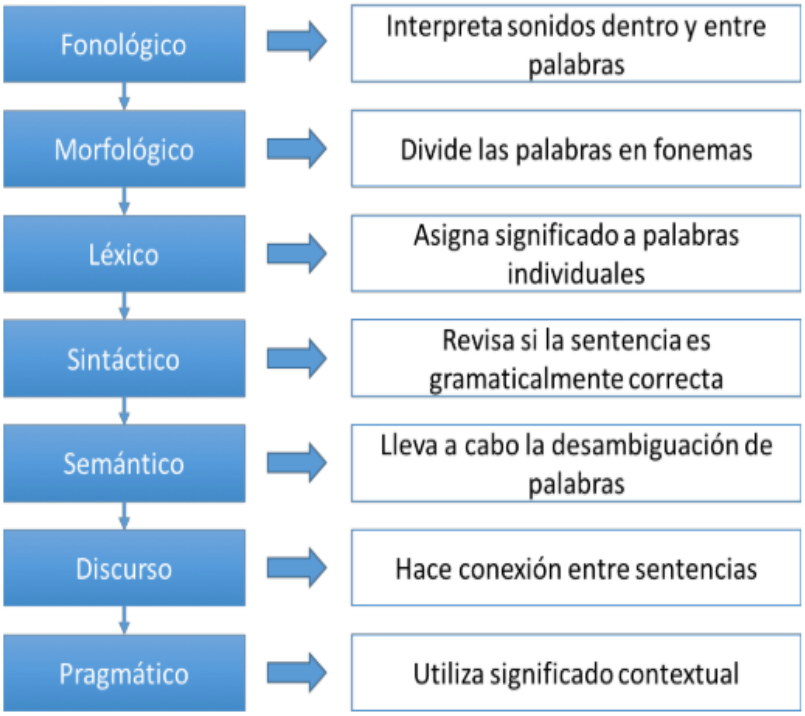


Figura 2. Niveles del procesamiento de lenguaje natural.
Fuente: Zarate, M. (2017) Detección de Patrones Psicolingüísticos para el Análisis de Lenguaje Subjetivo en español.

2.2.1.1. Nivel fonológico.

Comúnmente llamado fonética y se encarga del análisis de los sonidos del lenguaje humano producidas por la voz. Liddy (2001) en este nivel existen tres tipos de reglas usadas 1) Reglas fonéticas, utilizadas para los sonidos dentro de las palabras; 2) reglas de fonema, para las variaciones de pronunciación cuando las palabras se hablan juntas; y 3) reglas prosódicas, las cuales son utilizadas para la fluctuación del acento y entonación cuando se enuncia una sentencia. En este nivel la tarea de los sistemas PLN es aceptar como entrada la voz humana, analizarla y codificar las ondas producidas en una señal digitalizada para interpretar varias reglas o para establecer comparaciones con otro modelo de lenguaje.

2.2.1.2. Nivel Morfológico.

Este nivel del lenguaje humano tiene la finalidad de estudiar la composición y la derivación de las palabras, es decir buscar detectar las unidades mínimas o reglas gramaticales que forman una palabra, y para ello los sistemas de PLN buscan detectar y transformar cada carácter o palabra en un morfema con la utilización de herramientas como stemmers, lematizadores, o POS-taggers para detectar el significado de estas.

2.2.1.3. Nivel Léxico.

Este nivel del procesamiento del lenguaje natural tiene el objetivo de buscar interpretar el significado de las palabras individuales, y para ello los sistemas de PLN se soportan de varios tipos de procesamiento, una de ellas es la llamada comúnmente part-of-speech tag o la asignación de etiquetas individuales a cada palabra del texto, "estas etiquetas determinan la categoría léxica de la palabra y en el caso de que una palabra pueda tener diferentes funciones dentro de la sentencia, será etiquetado con la categoría léxica más probable en función del contexto" (Zárate, 2017, p. 23).

2.2.1.4. Nivel Sintáctico.

El objetivo de este nivel es, estudiar el orden y la relación de las palaras en una oración y la función que cumplen en esta, "trata de cómo las palabras pueden unirse para formar oraciones, fijando el papel estructural que cada palabra juega en la oración y que sintagmas son parte de otros sintagmas" (Cortez, Vega, & Pariona, 2009, p. 48).

2.2.1.5. Nivel Semántico.

Este nivel de análisis tiene el objetivo de determinar el significado de una oración centrándose en los significados de las palabras que forman la oración. este nivel de procesamiento puede requerir de técnicas de desambiguación semántica para aquellas palabras que tengan varios significados.

2.2.1.6. Nivel Discurso.

Los niveles de análisis semántica y sintáctica se enfocan en textos a nivel de sentencia mientras que este nivel a nivel de discurso se enfoca principalmente en textos mucho más largos, es decir, no interpreta los textos de varias frases como si fueran oraciones concatenatorias, cada una de las cuales puede interpretarse individualmente. Sino, el nivel de discurso se enfoca en las propiedades del texto como un todo que transmite significado haciendo conexiones entre oraciones componentes (Liddy, 2001).

2.2.1.7. Nivel Pragmático.

Según Zárate (2017) el análisis pragmático es uno de los niveles más complejos donde se analiza como las sentencias son usadas en distintas situaciones y de cómo el uso afecta al significado de las sentencias. El análisis que se realiza en este nivel se encuentra relacionado con los factores extralingüísticos que condicionan el uso del lenguaje en situaciones

concretas. Algunas aplicaciones PLN utilizan bases de conocimiento y módulos de inferencia que permiten interpretar las intenciones, los planes y los objetivos de un texto.

Tabla 1.

Niveles del lenguaje natral y herramientas PLN.

Nivel del procesamiento del lenguaje natural	Objetivo de estudio	Herramientas Utilizadas	Resultados del procesamiento
Fonético	Sonidos	Corpus de Aprendizaje Modelos Acústicos Diccionario de Unidades de Síntesis	Fonemas
Léxico	Formas	Lematizador Etiquetador Pos Lexicón Computacional	Palabras
Morfológico	Formas	Lematizador Etiquetado POS Lexicón Computacional	Palabras
Sintáctico	Estructuras	Base de Datos Sintácticas Treebank	Frases
Semántico	Significados	Base de Datos Semánticas Lexicón Computacional	Relaciones
Discurso	Comunicación	Bases de Datos Semánticas Ontologías	Textos
Pragmático	Comunicación	Bases de Datos Semánticas Ontologías	Textos

Fuente: Martinez, I. (2015) Minería de Opiniones Basada en Características Guiada por Ontologías.

2.3. Minería de Opiniones y Análisis de sentimiento

En relación al tema expuesto anterior mente, donde se afirma que el Procesamiento de Lenguaje Natural (PLN) es una técnica que busca utilizar las computadoras y las nuevas tecnologías, para la comprensión y la manipulación del texto y el lenguaje natural, con el fin de buscar información relevante, en los últimos años surgió una temática de estudio que ha mostrado un gran crecimiento y se denomina Análisis de Sentimiento (AS).

Así pues en la literatura el análisis de sentimiento tiene varias denominaciones como, minería de opiniones (MO), análisis de opinión, análisis subjetivo, análisis de la emoción, la computación efectiva y la extracción de la evaluación, entre otros, Abbasi, Chen, & Salem,

(2008); C. Miranda (2017). Sin embargo, la más usada en las investigaciones es Minería de Opiniones. Según Liu (2012) el análisis (AS) de sentimiento y minería de opiniones (MO) son un campo indistinto de estudio donde se analiza las opiniones de las personas, los sentimientos, las evaluaciones, las actitudes y las emociones del lenguaje escrito. Del mismo modo para Prameswari, Surjandari, & Laoh (2017) el análisis de sentimiento conocido también como "Minería de Opiniones" es una disciplina que combina el proceso de recuperación de información, extracción de texto y lingüística computacional para detectar las opiniones expresadas en el texto.

En consecuencia, de la amplia utilización del término minería de opiniones en las investigaciones que se realiza, en este trabajo utilizaremos la definición de minería de opiniones (MO) de forma indistinta con el termino de análisis de sentimiento.

2.3.1. Minería de Opiniones.

Minería de Opiniones también llamada análisis de sentimiento es una sub disciplina del procesamiento del lenguaje natural que busca aprovechar la gran cantidad de textos o comentarios disponibles en la web para obtener información relevante, diferentes autores lo definen de la siguiente manera.

Según Sangeetha (2017) la minería de opiniones examina las formas de categorizar todas y cada una de las polaridades de los comentarios como neutrales, negativas o positivas.

Por otro lado Ding et al. (2017) mencionan que la minería de opiniones o análisis de sentimiento utiliza las tecnologías de procesamiento de lenguaje natural para tratar el estado de animo de las personas sobre un producto tema o servicio en particular.

De la misma manera Molinar et al. (2017) definen al análisis de sentimientos o minería de opiniones, como una técnica que permite el análisis de expresiones de las personas tales como opiniones, sentimientos, evaluaciones, actitudes, emociones y valoraciones sobre entidades como productos, servicios, organizaciones, individuos, asuntos, eventos, tópicos y sus atributos.

En suma, la minería de opiniones buscara analizar la inmensa cantidad de comentarios emitidos en las diversas páginas web, sobre diferentes productos, servicios u organizaciones, con el fin de determinar si la parte del contenido es positiva, negativa o neutra, Afzaal &

Usman (2016). Brindando la posibilidad de tener un amplio grado de aplicaciones, Por ejemplo, esta técnica puede monitorear por medio de las opiniones las tendencias de consumo de los consumidores de algunos productos, dejando de lado la tradicional encuesta a los clientes.

Teniendo en cuenta las nuevas tendencias en la utilización de la internet como por ejemplo el comercio electrónico, las redes sociales y la penetración de dispositivos móviles las técnicas de minería de opiniones resulta cada vez mas de mucha utilidad para para exprimir al máximo toda esta información como en el turismo, películas, deportes, política, educación, salud, finanzas, transporte, etc, (Miranda & Guzmán, 2016). Debido a que, las organizaciones necesitan brindar y satisfacer todas las necesidades de sus clientes. Por ejemplo, una empresa en cuya tienda online se permita que los clientes realicen comentarios u opiniones de los productos que oferta, podrá analizar de forma activa lo que sus clientes tienen que decir de los productos que les venden.

2.3.2. Objetivos de la Minería de Opiniones

Como se mencionó en el apartado anterior el objetivo fundamental de la minería de opiniones es que, por medio de las técnicas se pueda analizar los comentarios existentes en la web, con el fin de obtener conocimiento relevante respecto a la razón de estudio. Para ello Como se apunta en Balahur & Montoyo (2010) citado por Peñalver, Valencia, & García (2011) un proyecto de minería de opinión requiere, en primer lugar, que se determinen las páginas Web de las que se van a extraer las opiniones de los usuarios, luego se establece la periodicidad con la que extraer dichas opiniones o comentarios y por último, se procesan las opiniones o comentarios obtenidos para aplicar las métricas que permiten visualizar y gestionar de forma adecuada los resultados obtenidos.

Por tanto, para cumplir estos objetivos que la minería de opiniones busca se tiene una serie de paso que la mayoría de la literatura adopta Figura 3.

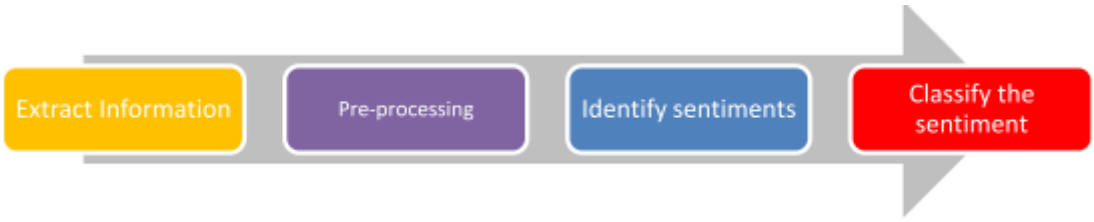


Figura 3. Pasos para la minería de opiniones

Fuente: Miranda, (2017), Un modelo integrado de técnicas de aprendizaje de máquinas no supervisadas y ontologías para la detección automática de sentimientos desde una estructura gramatical simple en español.

2.3.2.1. Extracción información.

El propósito de este paso es que antes de los pasos sub siguientes, se tenga una cantidad considerable de datos en donde se pueda aplicar el estudio, estos datos deberán ser extraídos de las principales fuentes de datos dependiendo del domino de estudio, como por ejemplo podrían porvenir de blogs, páginas de comercio electrónico y turismo.

Para la extracción de estos datos, algunas investigaciones utilizan APIS de acceso como el trabajo de Afzaal & Usman (2016) que utilizan las APIS de twitter para la recolección. Otras recurren a una técnica de recolección conocida como web scraping o web crawling como los trabajos realizados por Prameswari et al. (2017); Ding et al. (2017); Sangeetha, 2017). Así también existen algunos conjuntos de datos o corpus etiquetados, que se brindan gratuitamente en las dos más grandes competiciones de lenguaje de procesamiento natural realizadas por SEMEVAL O SEPLN.

2.3.2.2. Pre-procesamiento.

Las etapas de pre-procesamiento buscan una limpieza y adecuación de los datos que son incompletos e inconsistentes , es decir se buscan corregir o eliminar aquellos datos que no son los idóneos para su análisis, porque estos datos no se pueden usar directamente para la minería de datos (Ding et al., 2017).

según Prameswari et al. (2017) el pre-procesamiento de texto consta de varias etapas, que se aplicaron de acuerdo con las necesidades de la investigación, estas etapas consisten básicamente, en la identificación de errores ortográficos, la eliminación de secuencias arbitrarias de espacios, detener palabras, la detección de límites de frase, la eliminación del uso arbitrario de los signos de puntuación y las mayúsculas, entre otros.

2.3.2.3. Identificación de Sentimiento.

Esta etapa tiene el objetivo de encontrar estrictamente la selección de características o ubicación en el texto de palabras o frases que indiquen un posible sentimiento, siendo que esta etapa es una de los más importantes en la minería de opiniones, existen varias técnicas abordadas en las investigaciones, como la utilización de técnicas de POS Tagger, utilización de términos de presencia y frecuencia, conceptos de negación, entre otros.

2.3.2.4. Clasificación de sentimiento.

En consecuencia, de los pasos anteriores, en esta etapa se determina el sentimiento expresado por el comentario como positiva, negativa o neutra, respecto al objeto de estudio, Otras aplicaciones de AS asignan a una revisión valores numéricos como 4 o 5 estrellas considerada positiva y una revisión con 1 a 2 estrellas considera una opinión negativa. La mayoría de los trabajos de investigación no utilizan la clase neutra, lo que hace el problema de clasificación considerablemente más fácil (C. Miranda, 2017).

2.3.3. Niveles de Minería de Opiniones.

En las diversas investigaciones realizadas en el enfoque de minería de opiniones se ha visto que el análisis se puede realizar en diversos niveles de granularidad, a nivel de documento, a nivel de sentencia y a nivel de características (Prameswari et al., 2017). Figura 4.

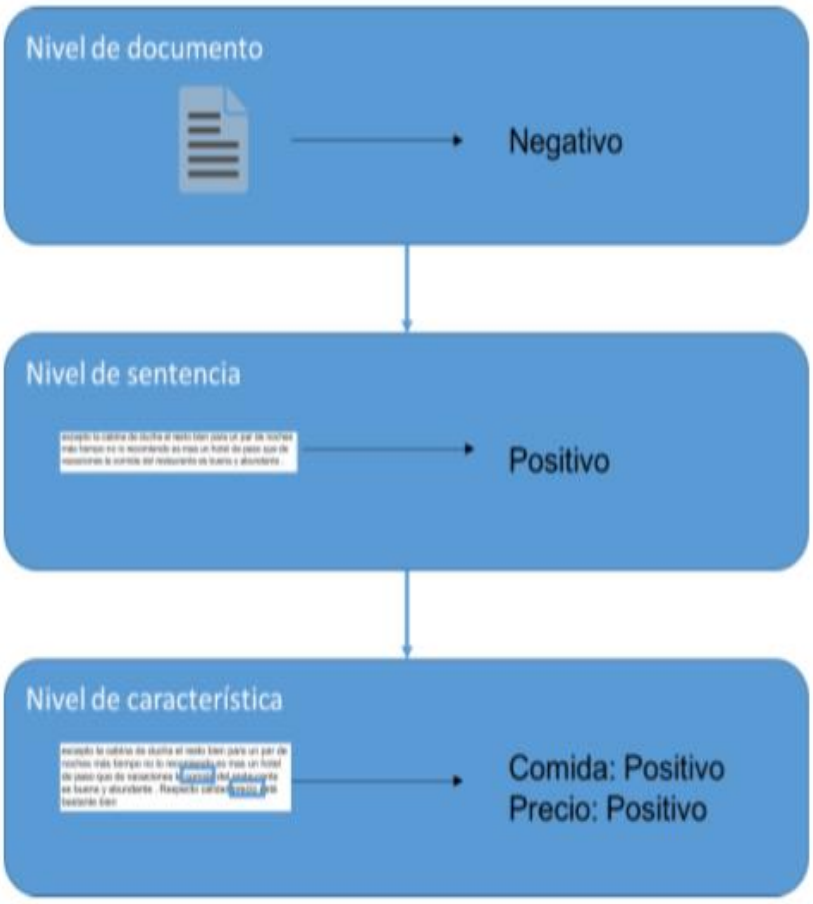


Figura 4. Niveles de Minería de Opiniones.
Fuente: Zárate,(2017). Detección de Patrones Psicolingüísticos para el Análisis de Lenguaje Subjetivo en español.

2.3.3.1. Nivel de Documento.

Este nivel de estudio busca principalmente determinar la polaridad del sentimiento como positivo negativo o neutro, dentro del documento completo Pang & Lee (2008). Algunos autores lo mencionan como el nivel de estudio más tradicional.

2.3.3.2. Nivel de Sentencia.

Este nivel de análisis considera grupos de evaluación en vez de palabras como unidades mínimas de expresión Olivares (2016). Por esto resulta ideal para los análisis en las plataformas de mensajería o microblogging como twitter, donde las personas comparten sus opiniones en mensajes de texto que no superan los 140 caracteres, además de que los comentarios deberán estar basados en temas muy específicos y expresados en frases cortas.

2.3.3.3. Nivel de Características.

También llamada Aspecto, y tiene como objetivo identificar las propiedades o características de una entidad y determinar la polaridad expresada de cada aspecto de esa entidad. Zárate (2017) menciona que la diferencia entre este nivel y los anteriores es que con este nivel se puede determinar qué es exactamente lo que a las personas les gusta o no les gusta. Por lo que el objetivo de este nivel de análisis es descubrir sentimientos sobre entidades y/o sus aspectos.

2.4. Turismo y minería de Opiniones

La industria del turismo es de vital importancia para el desarrollo de muchas comunidades en el que se practica, ya que es responsable del crecimiento económico y social de las zonas, de modo que, con el inicio de la Web 2.0 y el crecimiento explosivo de las redes sociales ,la forma de abordar el tema turístico ha cambiado drásticamente, ya que , las empresas y las personas utilizan cada vez más el contenido de los medios digitales para tomar mejores decisiones, porque pueden compartir y debatir libremente sus opiniones sobre un determinado tema, ya que facilita que la encuestas, grupos focales y algunas técnicas de obtención de información sean innecesarias.

Como consecuencia, de este desarrollo tecnológico de medios de comunicación, medios sociales y sitios web de diferentes tipos, en sector turístico surge la aparición de foros y sitios web que funcionan como fuente líder de información en el campo del turismo y la hospitalidad(Kavitha, 2017). Figura 5. Existiendo en la actualidad una gran cantidad de

páginas web que funcionan como un foro de discusión, así como un lugar para que los turistas vean y escriban reseñas en línea, como tripadvisor, yelp, Citysearch , etc , ganándose la popularidad entre los viajeros de todo el mundo, (Prameswari et al, 2017). En definitiva, el turista de la nueva era, utiliza las nuevas bondades del internet para todas las fases de visita al destino turístico, como antes del viaje, durante el viaje y después del viaje, ver Figura 5.

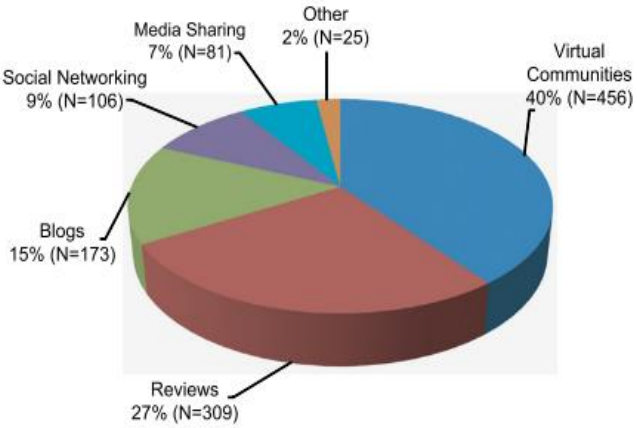


Figura 5. Medios sociales utilizados por los viajeros
Fuente: Xiang, Z., & Gretzel, U. (2010). Role of social media in online travel information search. *Tourism Management*, *31*(2), 179–188. Recuperado de: https://doi.org/10.1016/j.tourman.2009.02.016.

La gran cantidad de viajeros buscan calificar y dar su punto de vista de los lugares que visitaron, y por este motivo comparten sus experiencias través de los sitios web y redes sociales, generando críticas, comentarios y opiniones sobre sus experiencias de viaje, (Afzaal & Usman, 2016). Por esta razón los cometarios en línea que se comparten en estos foros de viajes están en aumento día a día debido al creciente número de viajeros que están dispuestos a compartir sus experiencias de viaje Prameswari et al. (2017).

Figura 6. Momentos del uso de internet por los turistas.
Fuente: Molinar et al., (2017), evaluación de destinos turísticos mediante la tecnología de la ciencia de datos.

Las opiniones y experiencias son fundamentales para casi todas las actividades humanas y son factores clave para influir en nuestro comportamiento, además nuestras creencias y percepciones de la realidad, y las decisiones que tomamos, están en gran medida condicionadas por la forma en que otros ven y evalúan el mundo, por esta razón cuando tomamos una decisión a menudo nos guiamos por las decisiones de los demás (Liu, 2012).

Según Pan et al. (2008) citado por Serna, Marchiori, Gerrikagoitia, Alzua, & Cantoni, (2015), estudios anteriores subrayaron que los visitantes tienden a considerar más los cometarios en línea que las fuentes oficiales, debido a que las conversaciones en los sitios web se basan más en experiencias y están menos relacionadas con los intereses comerciales potenciales, de esta manera las personas pueden obtener una imagen más completa del destino y sus servicios.

Por tanto, cuando un turista busca realizar un viaje de ocio a menudo necesitan consejos sobre dónde ir, cuándo ir y cómo ir, y la forma tradicional de realizarlo es que, el turista sea ayudado por una agencia de viajes, donde se dispone de asesores cuyas recomendaciones a veces pueden estar restringidas por factores humanos como desconocimiento del lugar, por lo cual, los turistas de hoy en día buscan las opiniones de los visitantes anteriores, antes de visitar cualquier lugar turístico y la fuente principal para buscar dicha información son los sitios web de redes sociales y portales de turismo.

Los comentarios de las atracciones turísticas juegan un papel importante en la elección de las atracciones turísticas de las personas, porque hoy en día, la industria del turismo en cualquier lugar depende en gran medida de las opiniones de los visitantes anteriores y sus percepciones, la retroalimentación de los visitantes anteriores es informativa para los nuevos

turistas y sus recomendaciones alientan a los nuevos turistas a visitar el mejor lugar (Afzaal & Usman, 2016; Kavitha, 2017).

Finalmente, desde la perspectiva de los turistas, los turistas se enfrentan al problema de la elección de la atracción turística, desde la perspectiva de las atracciones turísticas, también se enfrentan a cómo entender correctamente la suya, desarrollar la suya propia y cómo atraer mejor a los turistas en el actual entorno feroz de competencia, por ello el análisis adecuando de las comentarios con las técnicas de minería de opinión implica la construcción de un sistema para recopilar y categorizar opiniones sobre un lugar turístico. con el fin de mejorar la experiencia de visita al destino.

2.4.1. Experiencia de vista y destino Turístico.

Según Molinar (2012) Un destino turístico es un conglomerado de localidades, atractivos y servicios ubicado en un área geográfica delimitada y con una cultura propia, en la cual una red social de operadores turísticos públicos y privados colabora con la comunidad y con agentes externos para producir una experiencia turística que resulta satisfactoria para los visitantes, generando una imagen que motiva a los residentes externos a visitarlo.

Además SECO (2014) (Cooperacion Suiza) menciona en su informe, que un destino turístico es un espacio geográfico con características y rasgos particulares y cuenta con atractivos y servicios, accesibilidad, respaldo de la población y marca que pueda ser comercializada.

Así pues, el destino turístico constituye una unidad de negocio cuya gestión debe enfocarse en ciertos criterios de competitividad, a fin de generar beneficios económicos y sociales. Esta visión exige cierta capacidad administrativa, a fin de desarrollar instrumentos comunes que permitan planificar, medir y monitorear los resultados de la gestión.

En resumen, un destino turístico deberá ser capaz de proveer una experiencia de visita adecuada, con el fin de tener una competitividad sobre los demás destinos y así mantener su éxito económico, sin embargo, satisfacer la experiencia de visita conlleva una enorme complejidad, debido a que la experiencia es el resultado de una colección de servicios comerciales y no comerciales que ofrece el destino producidas por empresas públicas, privadas, gobiernos locales, organizaciones, etc. Ver Figura 7.

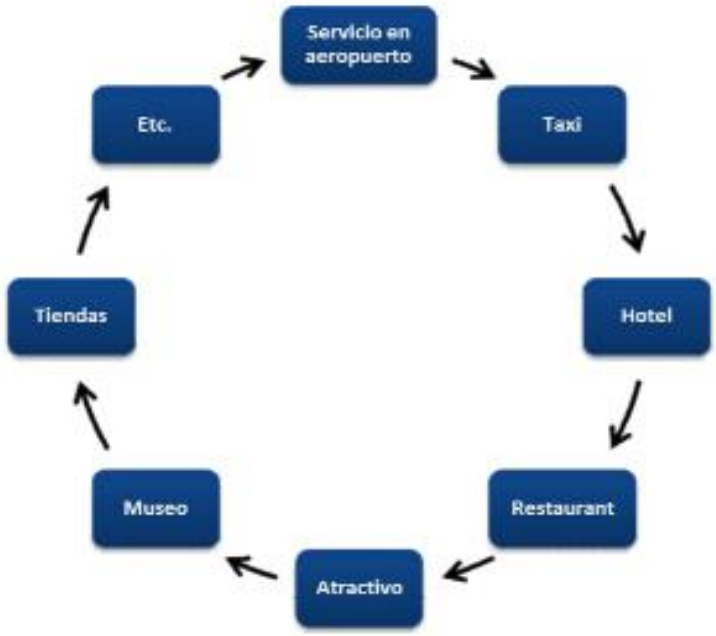


Figura 7. Cadena de eslabones de experiencia de visita al destino.
Fuente: Ritchie, J. B. & Crouch, G. (2005) "The competitive destination: A sustainable tourism perspective".

2.5. Pre Procesamiento de lenguaje Natural.

Para que los clasificadores realicen de forma eficiente su labor y obtengan resultados mejores es necesarios realizar un tratamiento al corpus o los datos que se tengan, esto dependerá en gran medida de la tarea de procesamiento que se realice y el idioma en el que se esté tratando; la principal tarea es que se limpien y normalicen la información, para que se evite principalmente su influencia en los resultados finales.

Esta cuestión es crucial cuando hablamos de mensajes extraídos de redes sociales ya que es muy habitual encontrar mensajes con faltas de ortografía, repeticiones de caracteres, mezcla de letras mayúsculas y minúsculas (Sande, 2018).

Estos tratamientos de información se realizan mediante algunas técnicas que describiremos a continuación.

2.5.1. Normalización.

Es la parte del proceso que consiste en ajuar las palabras de diferentes maneras, como por ejemplo eliminar los signos y símbolos que no son necesarios para el procesamiento, corrección de errores ortográficos, además de convertir las palabras a minúsculas o mayúsculas.

No es lo mismo para los algoritmos la expresión “TURISMO” y “turismo”, a pesar de que tiene el mismo significado para las personas, por esto es necesario normalizar las palabras a su equivalente en minúsculas.

2.5.2. Eliminación de stopwords.

Común mente llamados (stopwords), palabras sin sentido o palabras vacías, según Henriquez & Guzman (2016); Solis (2018) los stopwords son aquellas palabras que no agregan mucha información en lo que respecta al sentimiento, pero son necesarias para la construcción de oraciones con sentido, por esta razón son tan comunes y frecuentes, que aparecen en mayor cantidad en casi todos los comentarios. Por ello generalmente en el español los stopwords son las preposiciones, los pronombres, las conjunciones y las distintas formas del verbo haber, entre otras (Sande, 2018). Por ejemplo, las palabras: de, la, el, que, a, lo, su, etc.

2.5.3. Tokenización.

El proceso de tokenización, busca principalmente dividir el texto en una secuencia de tokens los cuales corresponden a “palabras”, es decir que la oración completa será convertida a unidades más pequeñas llamadas tokens y normalmente cada palabra se corresponde a un token, algunos autores lo definen de la siguiente manera.

Según Martínez (2015) es el proceso que se aplica a un texto en lenguaje natural y devuelve una lista de tokens separados en palabras individuales.

Por otro lado Alfrjani, Osman, & Cosma, (2016) mencionan que Cada comentario en el conjunto de datos se convierte en tokens y cada token tiene un número único, una posición (inicio y final) y otras características, como la longitud del token.

En suma, el resultado será como se muestra en la Figura 8 donde cada casilla corresponde a un token.

No	tengo	palabras	para	describir	esta	película	,	simplemente	es	genial	.

Figura 8. Proceso de tokenización.
Fuente: Zárate, M. del P. S. (2017). Detección de Patrones Psicolingüísticos para el Análisis de Lenguaje Subjetivo en español, https://doi.org/10.13140/RG.2.1.2171.2482.

2.5.4. Lematización.

Este es un proceso de normalización morfológica que transforma cada palabra en su lema mediante el uso de diccionarios y de un proceso de análisis morfológico (Sande, 2018). En toras palabras este proceso buscara unificar todos los términos que expresan la misma información reemplazándola con su lema, entendiendo al lema como un término valido del vocabulario.

Por ejemplo:

lema(“sea”) = ser
lema(“empieza”) = empezar
lema(“estas”) = seste

2.5.5. Stemming.

El proceso de stemming tiene el objetivo de realizar un proceso más rudo que la Lematización, y permite obtener la raíz de la palabra eliminando las terminaciones de sus sufijos e inflexiones. Con el objetivo de al igual de la Lematización de unificar las expresiones que tienen la misma información, pero este stem de la palabra no necesariamente será un término valido del vocabulario.

Por ejemplo:

stem (“recomendable”, “recomendamos”, “recomendar”, “recomendación”) = recomend

2.5.6. Part Of Speech (POS) Tagging.

General mente conocido como Part Of Speech Tagging (o POS Tagging) y su rol fundamental en el PLN es asignar a las palabras una etiqueta que determine cuál es su categoría gramatical, basado en su definición y su contexto, es decir esta etiqueta gramatical asignada nos dirá si la palabra actúa como determinante, sustantivo o nombre, pronombre, verbo, adjetivo, adverbio, preposición, conjunción e interjección, etc.

2.5.7. Otros Preprocesamientos.

Otros preprocesadores también aplicados en PLN son, corregir errores ortográficos, palabras de menos de N letras, Tratamiento de la duplicidad de caracteres, Comienzo y finalización de Sentencias, Eliminación de tildes, Eliminación de números, Normalización de risas, Menciones, enlaces y hashtags, Normalización de jerga, etc. Y son aplicados con el fin de mejorar los resultados y dependiendo al objeto de estudio.

2.5.8. Herramientas de pre-procesamiento de Lenguaje.

Debido al auge en los estudios del procesamiento de lenguaje natural, hoy en día existen una gran variedad de herramientas destinadas a esta tarea, en su mayoría para el tratamiento en el lenguaje inglés, y todos ellos diferentes entre sí respecto a algunas características, a continuación, mencionaremos algunas herramientas disponibles hoy en día.

2.5.8.1. NLTK.

NLTK o Natural Language Toolkit por sus siglas en Ingles, es la biblioteca de PNL de Python más famosa, y ha dado lugar a avances increíbles en el campo, en su propio sitio web, NLTK dice ser una "una biblioteca increíble para jugar con lenguaje natural", así que la herramienta es adecuada para una base educativa de experimentación, exploración. Su estructura modular ayuda a comprender las dependencias entre los componentes y obtener la experiencia de primera mano con la composición de modelos apropiados para resolver ciertas tareas. Posee más de 50 corpus y recursos léxicos, siendo una de las mayores basededatos léxicas para el inglés, aunque también posee procesamiento multilingüe pero no en gran medida.

2.5.8.2. Spacy

SpaCy se anuncia a sí mismo como “la mejor manera de preparar el texto para el aprendizaje profundo”, esta biblioteca relativamente joven fue diseñada para uso de producción, por eso es mucho más accesible que NLTK, además, ofrece el analizador sintáctico más rápido disponible en el mercado hoy en día, dado que el kit de herramientas está escrito en Cython. Sin embargo, SpaCy no admite tantos idiomas como NLTK, de hecho, funciona actualmente solo con textos en inglés, y es probablemente el principal competidor de NLTK. dicen algunos especialistas, y quién sabe, tal vez el nuevo sheriff esté en la ciudad y NLTK tendrá que abandonar el trono algún día.

2.5.8.3. Core NLP

Se trata de una herramienta muy poderosa de procesamiento del lenguaje natural desarrollada y mantenida por una institución de investigación de élite la Universidad de Stanford, pero puede que no sea lo mejor para las cargas de trabajo de producción. Esta herramienta tiene una licencia doble con una licencia especial para fines comerciales, Está escrito en Java, aunque hay wrappers de Python hechos por la comunidad. Es confiable, robusto, más rápido que NLTK (pero spacy es mucho más rápido) y también es compatible

con múltiples idiomas, además, algunos de los componentes CoreNLP pueden integrarse con NLTK, lo que seguramente aumentará la eficiencia de este último.

2.5.8.4. Freeling

FreeLing es una librería de código abierto para el procesamiento multilingüe automático, que proporciona una amplia gama de servicios de análisis lingüístico para diversos idiomas Padró (2011). Esta librería esta implementada en el lenguaje de programación C++, aunque posee Apis para Python, provee una serie de módulos para el análisis de lenguaje y procesamiento de textos. El proyecto Freeling se inició desde el centro TALP de la Universidad Politécnica de Catalunya (UPC) y permite trabajar sobre una variedad considerable de idiomas, aunque el idioma fundamental en el que fue construida es el español teniendo una amplia gama de reputación en esta.

Como podemos observas las herramientas presentan una gama de características en cuanto a su funcionalidad, y los resultados de este son prometedores dependiendo a la herramienta y el lenguaje en el que se esté haciendo el estudio, en el siguiente cuadro comparativo mostramos algunas funcionalidades elementales que nos permiten las herramientas de PLN descritas en este apartado.

Cabe mencionar que la herramienta escogida en este trabajo es freeling por su robustez al trabajar con el lenguaje español ya que es el idioma centra en la que esta implementada esta librería.

Tabla 2.

Comparación de funcionalidades de herramientas PLN.

Nombre	**Spacy**	**NLTK**	**Core NLP**	**Freeling**
Instalación rápida	si	si	si	No
Python API	si	si	no	si
Soporte Multi Lenguaje	no	si	si	si
Tokenizacion	si	si	si	si
Stemming	si	si	si	si
Lematización	si	si	si	si
NER	si	si	si	si
Reconocimiento de tiempo	no	no	si	si
Part-of-speech tagging	si	si	si	si
Segmentación de oraciones	si	si	si	si
Análisis de dependencia	si	no	si	si
Reconocimiento de la entidad	si	si	si	si
Word Vectores integrados	no	no	si	no

Fuente: Elaboración propia.

2.6. Clasificadores de Textos.

Las técnicas para la clasificación de textos consisten en principalmente poder encontrar patrones y características del lenguaje que permitan asignar una categoría, etiqueta o clase a un documento, en otras palabas se busca predecir la clase de un documento por ejemplo spam o no spam, documento relevante o no relevante, estas técnicas son diversas, pero una de las más utilizadas hoy en día son las que se basan en técnicas de machine learning.

En sistemas de AS se buscan características y patrones asociados a opiniones, emociones y sentimientos y el objetivo consiste en predecir la polaridad (positiva o negativa) de un documento, sentencia o aspectos determinados de un texto. (Dubiau, 2013).

En los siguientes apartados presentaremos las técnicas más utilizadas de machine learning en tareas de análisis de sentimiento.

2.6.1. Técnicas de Machine Learning.

Se define Machine Learning, o Aprendizaje Automático en español, como el campo de estudio que le da a las computadoras la habilidad de aprender sin haber sido explícitamente programadas (Arthur Samuel, 1959). En otras palabras, estas técnicas son un conjunto de métodos que son capaces de detectar patrones de manera automática en base a la experiencia obtenida de los procesos que realizo anteriormente.

Este concepto es aplicable a sistemas de MO donde el algoritmo aprenderá por sí mismo por medio de un entrenamiento realizado a partir de un gran nuero de documentos (experiencias), atributos de esos documentos (features) y resultados conocidos (clases) para posterior mente clasificarlos en positivos o negativos.

2.6.1.1. Aprendizaje Supervisado.

El aprendizaje supervisado es aquella en la que el entrenamiento está dado por una gran cantidad de datos correctamente etiquetados, experiencias, atributos de esas experiencias y resultados, estos datos actúan como una guía para que el algoritmo aprenda las conclusiones a las que tiene que llegar.

Según González (2019), requiere que los posibles resultados del algoritmo ya sean conocidos y que los datos utilizados para entrenar el algoritmo ya estén etiquetados con las respuestas correctas. Por ejemplo, un algoritmo de clasificación aprenderá a identificar animales después de haber sido entrenados en un conjunto de datos de imágenes que están apropiadamente etiquetados con las especies del animal y algunas características de identificación.

En sistemas de AS, el entrenamiento del algoritmo de clasificación se realiza a partir de un gran número de documentos previamente clasificados correctamente (como positivos o negativos) y se busca, a partir de determinados atributos de los documentos, predecir la clase de otros documentos desconocidos con alta probabilidad de acierto.

2.6.1.2. Aprendizaje No Supervisado.

En algoritmos de aprendizaje no supervisado el entrenamiento se realiza a partir de datos que representan las experiencias y sus atributos de esas experiencias, pero a diferencia del aprendizaje supervisado aquí no se tienen datos sobre los resultados correctos o labels, es decir, no se conoce la clase a la que pertenece cada experiencia. En este tipo de algoritmos

el objetivo es encontrar similitudes, patrones o estructuras en los datos que permitan agruparlos sin conocer previamente los grupos existentes.

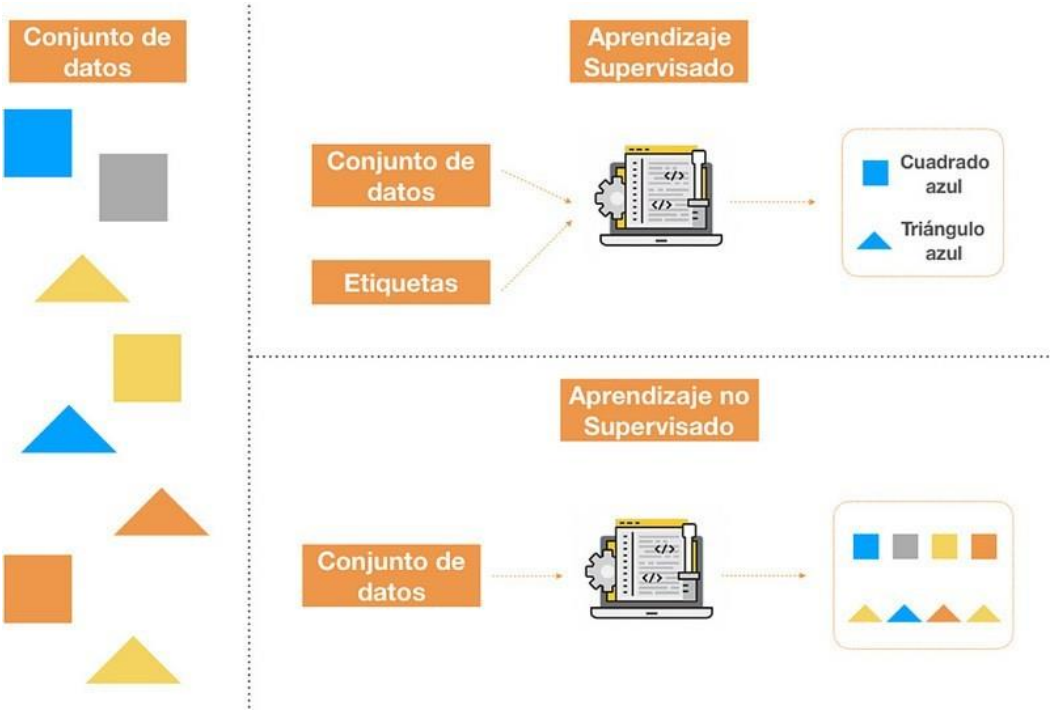


Figura 9. Aprendizaje supervisado y no supervisado.
Fuente: González, L. (2019). Diferencia entre aprendizaje supervisado y no supervisado, recuperado de: http://ligdigonzalez.com/diferencia-entre-aprendizaje-supervisado-y-no-supervisado/.

2.6.2. Modelos de Clasificación.

En este apartado mencionaremos los clasificadores texto más utilizados y mejore resultados obtenidos en tareas de procesamiento de texto. Según Lilleberg, Zhu, & Zhang (2015) la mayoría de investigaciones obtuvieron buenos resultados con las técnicas de Support vector machine y Naive Bayes.

Es por esto que el estudio de Sande (2018) muestra esta diferencia de resultados situando al algoritmo de clasificación Maquina de Vectores de soporte como la mejor en tareas de Mineria de opiniones, sobre los algoritmos de Naive Bayes, Arboles de decisión, y K vecinos mas cercanos.

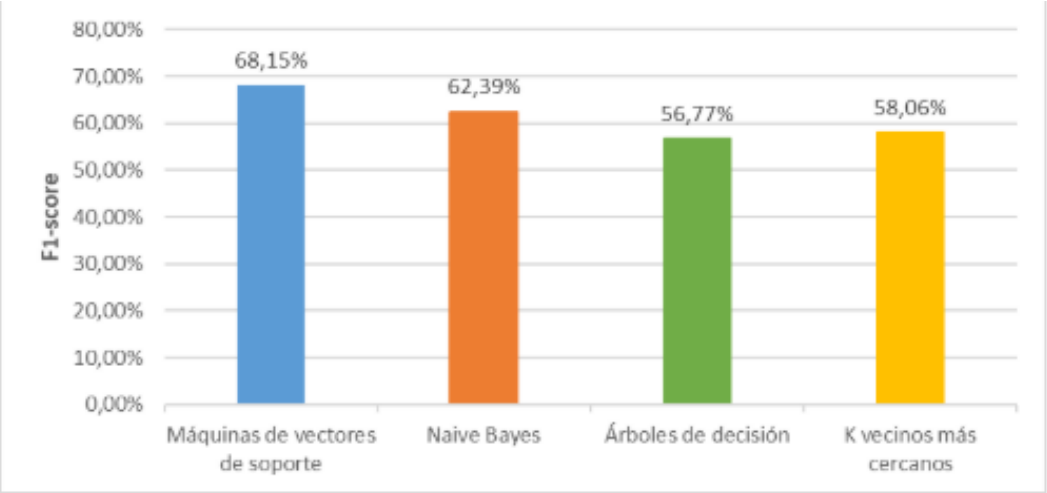


Figura 10. Mejor clasificador en tareas de minería de Opniones
Fuente: Sande, J. C. S. (2018). Análisis de sentimientos en Twitter.

2.6.2.1. Clasificador de Naive Bayes.

Naive Bayes es un método de clasificación supervisado y generativo, que se basa en el teorema de Bayes y en la premisa de independencia de los atributos dada una clase. Esta premisa es conocida como "Naıve Assumption" y se le llama "naıve" (o ingenua) considerando que en la práctica los atributos raramente son independientes.

Considerando el teorema de Bayes,

$$P(C_i|D) = \frac{P(C_i)P(D|C_i)}{P(D)}$$

Donde D es un documento del conjunto de datos de entrenamiento, Ci es cada una de las posibles clases y P(Ci|D) es la probabilidad de que el documento D pertenezca a la clase Ci. El clasificador seleccionará la clase que maximice esta probabilidad.

P(D) puede ser ignorada ya que es la misma para todas las clases y no afecta los valores relativos de probabilidad.

Además, basados en la premisa de independencia de los atributos dada una clase podemos descomponer P(D|Ci) como se ve en la ecuación que sigue:

$$P(C_i|D) \propto P(C_i) \prod_{k=1}^{n} P(f_k|C_i)$$

Donde fk son los features del documento y P(fk|Ci) es la probabilidad de ocurrencia del feature en la clase dada.

Por lo tanto, la clase seleccionada por el clasificador será la que maximice la probabilidad anterior

$$\boxed{C_{NB} = \arg\max_i P(C_i) \prod_{k=1}^{n} P(f_k|C_i)}$$

Las distintas implementaciones del algoritmo de Naıve Bayes difieren principalmente en la aproximación de P(fk|Ci).

2.6.2.2. Clasificador Support Vector Machines (SVM).

La técnica de clasificación basada en vectores de soporte o Support Vector Machines, es la técnica basada en aprendizaje supervisado, se introdujeron inicialmente en la década de 1960 y luego se refinaron en la década de 1990. Sin embargo, es solo ahora que se están volviendo extremadamente populares, debido a su capacidad para lograr resultados brillantes en diversas tareas como los de procesamiento de textos.

El objetivo del algoritmo de la máquina de vectores de soporte es encontrar un hiperplano en un espacio N-dimensional (N, el número de entidades) que clasifique claramente los puntos de datos. Para separar las dos clases de puntos de datos, hay muchos posibles hiperplanos que podrían elegirse. Nuestro objetivo es encontrar un plano que tenga el margen máximo, es decir, la distancia máxima entre los puntos de datos de ambas clases.

Veamos en más detalle el funcionamiento de la técnica de clasificación.

Imaginemos que tenemos datos en dos dimensiones cuyas clases de clasificación también son dos, esto hace imaginar que son lineal mente separables Figura 11

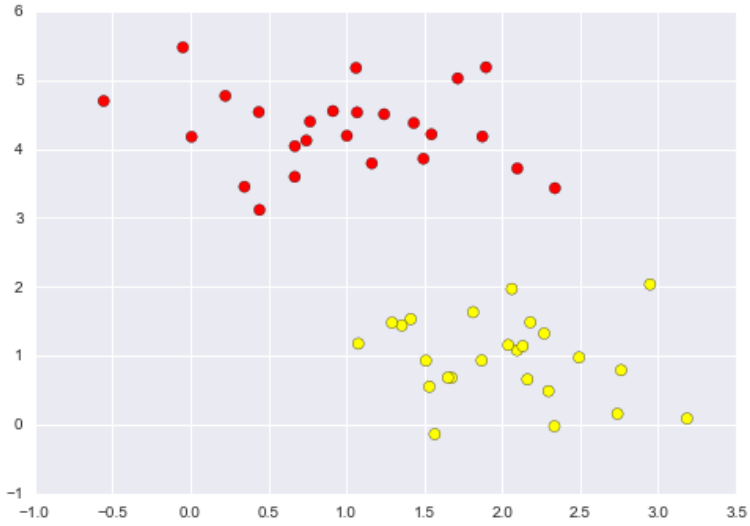


Figura 11. Ejemplo de visualización de datos.
Fuente: VanderPlas, J. (2019). In-Depth: Support Vector Machines | Python Data Science Handbook, recuperado de: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html.

Así pues, se intentará encontrar un límite que divide ambos datos de tal manera que minimice el error, sin embargo, en los problemas reales se encontrará que hay más de una posible línea divisora que puede discriminar perfectamente entre las dos clases.

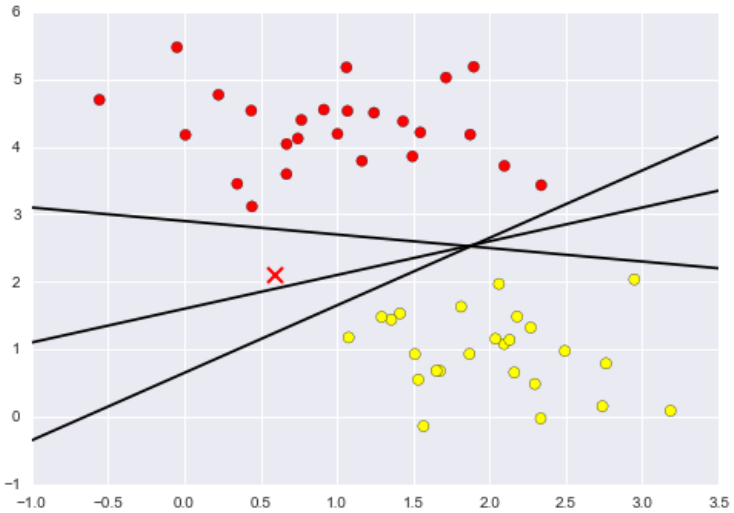


Figura 12. Ejemplo límite de división.
Fuente: VanderPlas, J. (2019). In-Depth: Support Vector Machines | Python Data Science Handbook, recuperado de: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

La máquina de Vectores de soporte busca como dijimos anterior mente, encontrar un plano que tenga el margen máximo, es decir en lugar de simplemente dibujar una línea de

ancho cero entre las clases, podemos dibujar alrededor de cada línea un margen de cierta anchura, hasta el punto más cercano, de tal manera que estos puntos se denominaran vectores de soporte, así pues la línea que maximiza este margen es la que elegiremos como modelo óptimo.

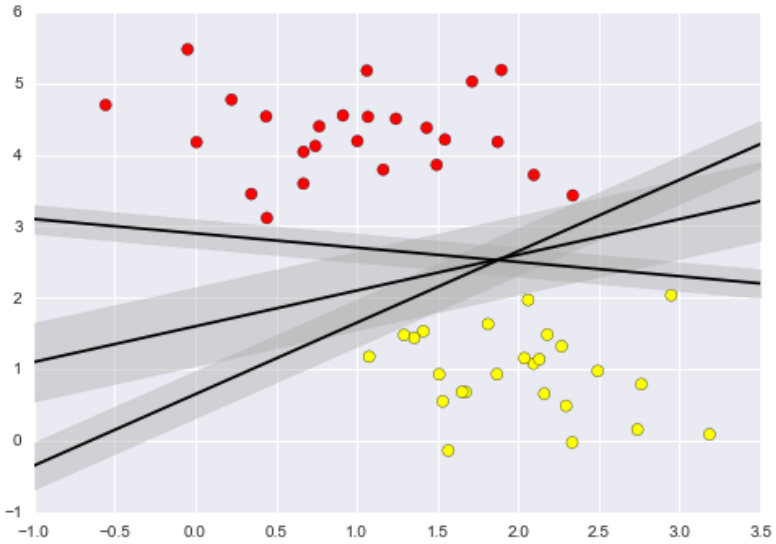


Figura 13. Ejemplo margen máximo SVM.
Fuente: VanderPlas, J. (2019). In-Depth: Support Vector Machines | Python Data Science Handbook, recuperado de: https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html

Los vectores de soporte son puntos de datos que están más cerca del hiperplano e influyen en la posición y orientación del hiperplano. Usando estos vectores de soporte, maximizamos el margen del clasificador. Eliminar los vectores de soporte cambiará la posición del hiperplano. Estos son los puntos que nos ayudan a construir nuestro SVM.

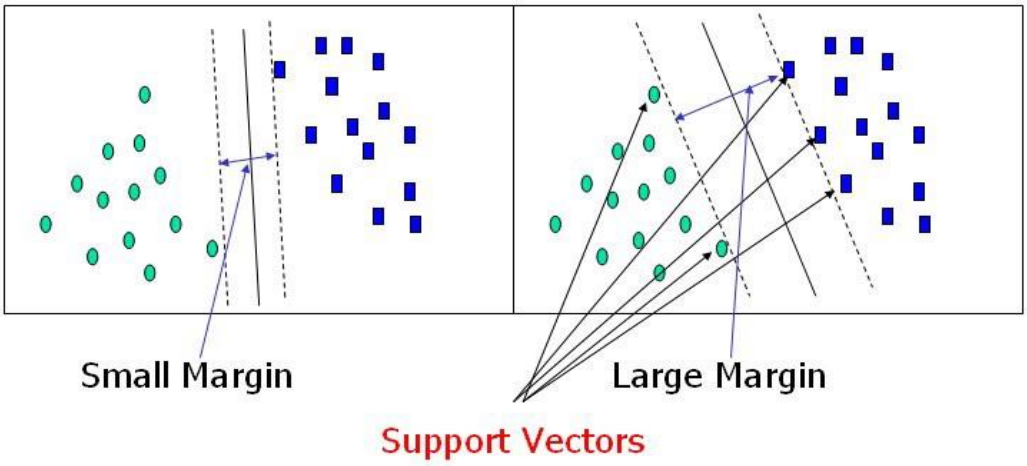


Figura 14. Ejemplo márgenes y vectores de soporte.
Gandhi, R. (2019). Support Vector Machine — Introduction to Machine Learning Algorithms. Retrieved from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Hay matemáticas complejas involucradas detrás de la búsqueda de los vectores de soporte, calculando el margen entre el límite de decisión y los vectores de soporte y maximizando este margen. En este trabajo no detallaremos dicha formulación matemática, dado que en nuestra experimentación utilizaremos la implementación provista por el framework Scikit-learn.

2.7. Evaluación de Clasificadores.

Para la aceptación del clasificador existen métricas que se utilizan para evaluar y aceptar los resultados del clasificador y obtener una medida de su eficiencia, en esta sección describiremos las distintas métricas disponibles para métodos supervisados en la evaluación de los clasificadores en función de los datos de entrenamiento.

2.7.1. Métricas.

La matriz de confusión es una técnica que nos da una idea de cómo están clasificándose nuestros modelos, dándonos un conteo de los aciertos y errores de cada una de las clases por las que estemos clasificando. Así, podremos comprobar si nuestro modelo está confundiéndose entre clases, y en qué medida.

Una matriz de confusión de dos clases, positivo y negativo es de la siguiente, manera Figura 15, donde cada una de las columnas, es el número de predicciones para cada una de

las clases realizadas por el modelo, y cada fila son los valores actuales por cada una de las clases, por lo que quedan divididos en 4 clases TP, FN, FP, TN. Gracias a estas cuatro categorías podemos calcular métricas más elaboradas. Como accuracy, Precision, Recall, F-measure que veremos más adelante.

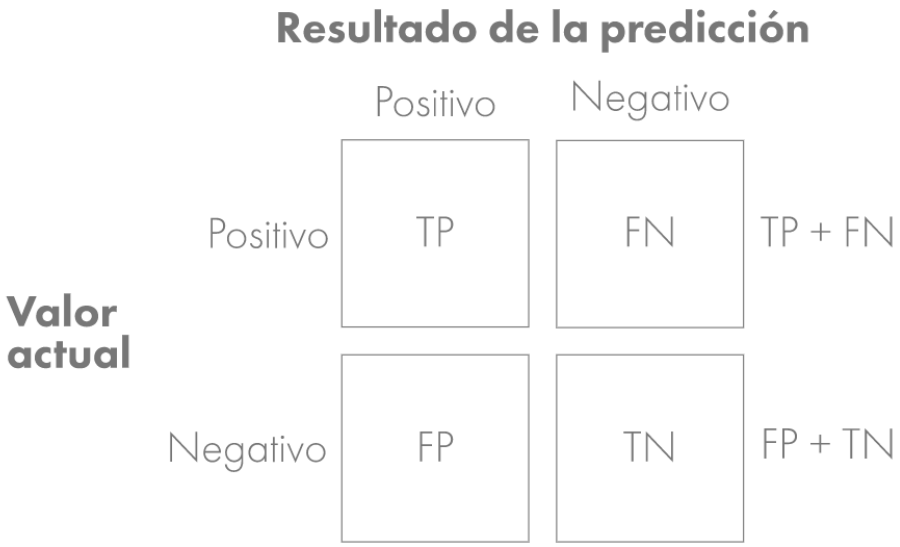


Figura 15. Matriz de confusión de dos clases.
Fuente: Selección de métricas para los modelos de aprendizaje automático | Fayrix. (2019), recuperado de https://fayrix.com/machine-learning-metrics_es.

- **True Positives o Verdaderos Positivos *(TP):*** Cuando los elementos de la clase real son positivos y los elementos de la predicción también es positivo, la clasificación es correcta.

- **False Positives o Falsos Positivos *(FP):*** Cuando los elementos de la clase real son negativos y el pronóstico es positivo, a estos casos también se les denomina errores de tipo I.

- **False Negatives o Falsos Negativos *(FN)*:** Cuando los elementos de la clase real son Positivos y el valor predicho es negativo, a estos casos también se les denomina errores de tipo II.

- **Verdaderos Negativos (TN):** Cuando los elementos de la clase real son negativos y el valor predicho es negativo, la clasificación es correcta.

“En tareas de AS las clases involucradas serán: “Documentos Positivos” y “Documentos Negativos” por lo que el significado que le daremos a la medición de resultados ser ‘a: Documentos clasificados como positivos que eran efectivamente positivos (tp); documentos clasificados como positivos que eran negativos (fp); documentos clasificados como negativos que eran positivos (fn); y documentos clasificados como negativos que eran efectivamente negativos (tn)” (Dubiau, 2013, p. 42).

2.7.2. Sensibilidad o Recall.

También se la llama tasa de verdaderos positivos. Nos da la probabilidad de que, dada una observación realmente positiva, el modelo la clasifique así.

$$\text{Sensibilidad} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

2.7.3. Especifidad:

También llamado ratio de verdaderos negativos. Nos da la probabilidad de que, dada una observación realmente negativa, el modelo la clasifique así.

$$\text{Especifidad} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

2.7.4. Accuracy.

Porcentaje total de los aciertos de nuestro modelo, es decir representa la porción de documentos que son clasificados correctamente sobre el total de casos.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

2.7.5. Precisión.

También llamado valor de predicción positiva. Nos da la probabilidad de que, dada una predicción positiva, la realidad sea positiva también.

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

2.7.6. F-Measure.

Combina las medidas de precisión y recall a partir de la media armónica ponderada.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

2.7.7. k-fold Cross-Validation.

Esta técnica de validación del modelo consiste en que un determinado conjunto de datos se divide en un numero de k secciones, cada una denominada **fold**, donde cada fold se utilizará como un conjunto de prueba en algún momento, y el resto como un conjunto de entrenamiento (k - 1).

Tomemos el escenario de la validación cruzada de 5 veces (K = 5). Aquí, el conjunto de datos se divide en 5 folds. En la primera iteración, el primer fold se usa para probar el modelo y el resto se usa para entrenar el modelo. En la segunda iteración, el segundo fold se utiliza como conjunto de pruebas, mientras que el resto sirve como conjunto de entrenamiento. Este proceso se repite hasta que cada fold de los 5 folds se haya utilizado como conjunto de prueba. (Hewa, 2019).

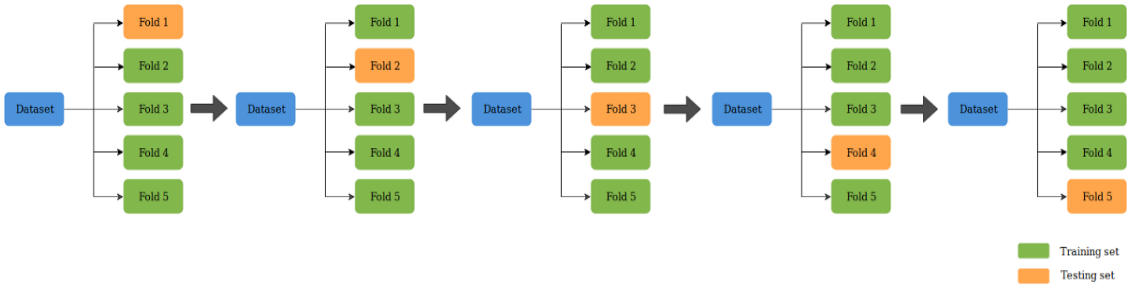


Figura 16. Fold Cross Validation.
Fuente: Hewa, K. (2019). K-Fold Cross Validation, recuperado de: https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833

El valor de k más utilizado es 10, pero puede seleccionarse k tantos valores como sea posible, siempre en cuando los datos lo permitan, la desventaja de este método es que ejecutar la clasificación k veces puede resultar muy costoso computacionalmente.

2.8. Vectorización.

Imaginemos una simple búsqueda en google sobre “desayunos”, el resultado será una gran colección de páginas web, artículos, revistas, etc. sobre la palabra que pusimos, en este caso “desayunos”, otro escenario es cuando realizamos una traducción en google traductor, el resultado será el texto traducido en el idioma que elijamos, entonces, ¿Qué tienen en común estos escenarios mencionados?, probablemente lo hayas adivinado y es el “procesamiento de texto”, los ejemplos mencionados trabajan con una enorme cantidad de textos para poder mostrar sus resultados. Es fácil imaginar que los seres humanos estamos muy familiarizados de forma intuitiva con los textos, pero ¿cómo hacemos que las computadoras de hoy realicen agrupamientos, clasificación, etc. en datos de texto, ya que sabemos que generalmente son ineficientes en el manejo y procesamiento de cadenas o textos para obtener resultados fructíferos?

Una computadora claramente te puede decir si dos términos son iguales además de darte la longitud de estas, sin embargo ¿Cómo las computadoras puedan encontrar coincidencias exactas sobre futbol, Argentina y mundial cuando la palabra que buscaste es Messi? ¿Cómo hacen las computadoras para saber que la palabra en ingles “Apple” utilizada en la oración “Apple is a tasty fruit” es una fruta que se puede comer y no una compañía?

La respuesta a todas estas cuestiones es que, hay técnicas asombrosas en la que se crean representaciones de las palabras que capturan sus significados, relaciones semánticas y los diferentes tipos de contextos en los que se utilizan.

En suma, están implementadas mediante el uso de lo que en ingles se conoce como “Word Embeddings” o representaciones numéricas, para que las computadoras puedan entender los textos.

En este apartado veremos que son los Word Embeddings y sus diferentes tipos aplicados hoy en día.

2.8.1. Word Embeddings.

Los word embeddings en palabras simples son una manera de convertir los textos en números, y estos números son llamados vectores de palabra, esto debido a que la mayoría de los algoritmos como los que vimos en el apartado anterior, son incapaces de procesar cadenas

de caracteres en su forma normal, estos necesitan como entrada números para realizar su trabajo.

Por ejemplo, la oración "Ten un buen día" para ser convertida a un word embeddings debería pasar por un proceso de conversión, construyendo un exhaustivo vocabulario que en este caso lo llamaremos "V".

V = ["Ten", "un", "buen", "día"]

Ahora deberíamos crear un "one-hot encoded vector" o vectores de palabras representadas en una numeración de 0 -1, 1 para el lugar donde existe y 0 en donde no lo está, así se tiene un vector de palabras de dimensión 5 como vemos a continuación.

Ten = [1,0,0,0]; un = [0,1,0,0]; buen = [0,0,1,0]; día = [0,0,0,1]

2.8.2. Tipos de word embeddings

Según Singh Sarwan (2019), los diferentes tipos word embeddings se pueden clasificar ampliamente en dos categorías claras:

- word embeddings basada en frecuencia
- word embeddings basada en predicción

En los word embeddings basadas en frecuencia tenemos:

- Count Vector
- TF-IDF Vector
- Co-Occurrence Vector

En los word embeddings basada en predicción tenemos.

- Word2Vec

En este trabajo de investigación utilizamos wor2vec para los casos de experimentación por lo que nos enfocaremos en mostrar las bondades de esta forma de representación de word enbeddings.

2.8.3. Word2Vec

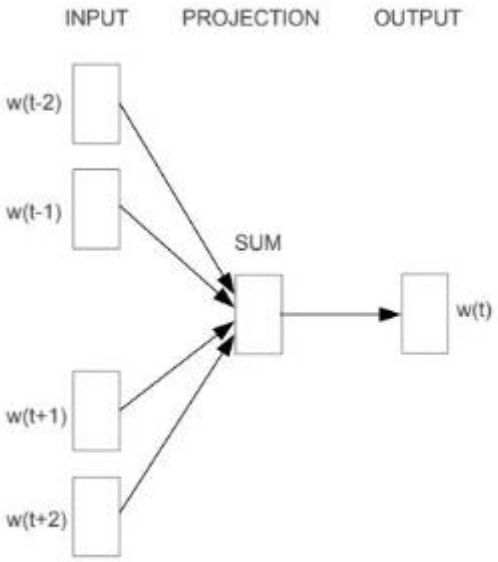
En primera instancia wor2vec parte de la suposición que dos palabras que comparten contextos similares, también compartirán un significado similar, en otras palabras, el significado de la palabra puede ser inferida por las palabras que las preceden y anteceden llamadas "contextos".

Word2vec es una red neuronal de dos capas que procesa texto. Su entrada es un corpus de texto y su salida es un conjunto de vectores, vectores de características para palabras en ese corpus. (A Beginner's Guide to Word2Vec and Neural Word Embeddings, 2019). Es aplicable a multitud de tareas más allá del procesamiento de texto, como por ejemplo el procesamiento de imágenes, listas de reproducción entre otros, lo que busca es detectar similitudes matemáticamente.

Word2vec crea vectores que son distribuciones numéricas y representan a las características de las palabras, así como también el como el contexto. Word2vec es una combinación de 2 diferentes modelos CBOW (Continuous bag of words) y el modelo Skip-gram, Ambas son redes neuronales poco profundas que asignan palabra(s) a la variable objetivo que también es una palabra(s). Ambas técnicas aprenden pesos que actúan como representaciones vectoriales de palabras.

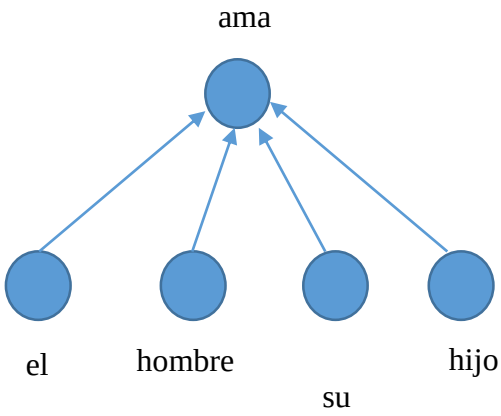
2.8.3.1. Modelo CBOW (Continuous Bag of words).

Es el menos popular de los modelos utilizados en wor2vec debido a que o produce resultados menos aceptables, su objetivo es predecir la probabilidad de una palabra dado un contexto, por ejemplo, intentara predecir la palabra "python" cuando el contexto que le doy sea "quiero aprender". Un contexto puede ser una sola palabra o un grupo de palabras dependiendo del "context window" o "Window size" que es parámetro que determina el número de palabras de cada contexto.



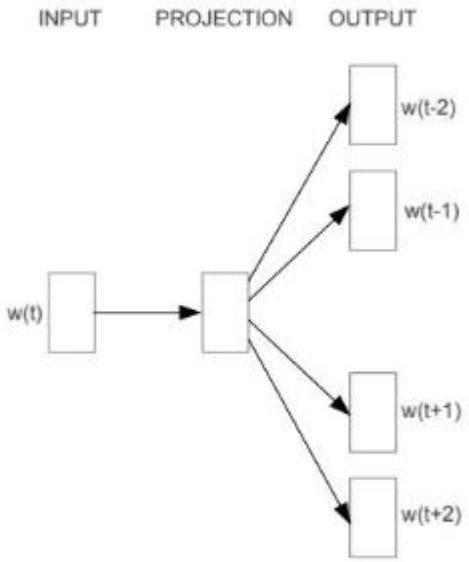
CBOW

Por ejemplo, si tenemos la siguiente oración "el hombre ama su hijo" donde "ama" es la palabra objetivo que queremos predecir dada una ventana de contexto de 2, el modelo CBOW recibirá como entradas el contexto ["el" "hombre"] o el contexto ["su", "hijo"] generando la palabra "ama".



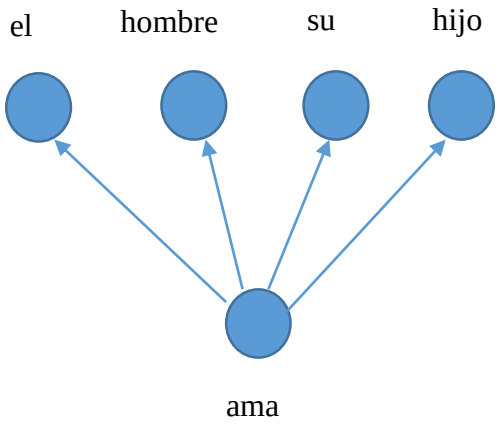
2.8.3.2. Modelo Skip -Gram

El modelo skip – gram funciona a de la misma manera que CBOW, pero con la diferencia que voltea su arquitectura, el objetivo de este modelo es predecir el contexto dado una palabra, es decir si quiero predecir el contexto "quiero aprender" la palabra que le pase será "python".



Skip-gram

Imaginemos que tenemos la oración anterior “el hombre ama su hijo”, el texto de secuencia será [“el”, “hombre”, “ama”, “su”, “hijo”] usaremos la palabra central “ama” para predecir el contexto con un Window size de 2.



CAPÍTULO III. Materiales y métodos

3.1. Descripción del lugar de ejecución

El presente proyecto será ejecutado de las instalaciones de las oficinas de CONCYTEC – UpeU Filial Juliaca como parte del proyecto Plataforma digital inteligente y Big Data para el turismo rural comunitario en la Región Puno", proyecto que fue financiado por el CONCYTEC-FONDECYT en el marco de la convocatoria "Proyecto Investigación Básica 2015" [número de contrato 184-2015].

3.2. Arquitectura de Solución

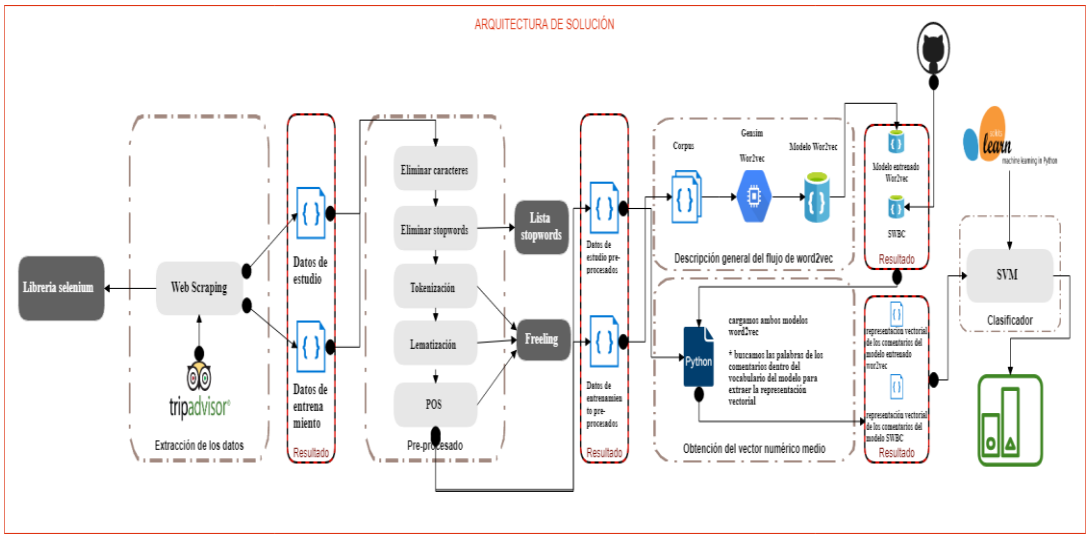


Figura 17. Arquitectura de Solución.
Fuente: Elaboración Propia.

Ver anexo B para ver la imagen ampliada.

3.3. Materiales Físicos

El trabajo se desarrolló en un computador Toshiba, con procesador i5 2.6 GHz, disco duro de 500 Gb y Memoria RAM de 4 Gb. cuyo sistema operativo fue Linux en la versión de Ubuntu Bionic beaver, 18.04 LTS.

3.4. Materiales de software.

3.4.1. Python.

Python es uno de los lenguajes de programación más utilizados en los trabajos relacionados a ciencia de datos, porque posee una gran cantidad de librerías en este campo, y hacen sencilla las tareas que involucran, como por ejemplo análisis de información, la obtención de datos, la limpieza, refinamiento, generación de modelos (machine learning), visualización de datos, etc. Es un lenguaje de programación de alto nivel, tipado dinámico, interpretado y multipropósito, además hace mucho hincapié en la sintaxis, lo que hace que este sea muy productivo.

Todas estas características hacen que python sea una herramienta muy poderosa, por este motivo en el trabajo se decidió utilizar este lenguaje de programación en su versión Python 3.6, y fue implementada en un entorno virtual de manera local con ayuda de virtualenvwrapper en un ambiente Ubuntu Linux.

3.4.2. Pip.

Utilizamos pip en una versión para Python 3, la herramienta nos permitió la gestión de paquetes necesarios para nuestra investigación, es decir, fue útil para poder instalar y hacer uso de las librerías y módulos que necesitamos, de las cuales hablaremos más adelante.

3.4.3. Web scraping.

Web scraping es una técnica muy utilizada para la extracción de información de las páginas web de una manera automatizada, su objetivo es transformar los datos sin estructura que están disponibles en la web en datos estructurados que pueden ser almacenados en base de datos, ficheros de texto, hojas de cálculo, etc. Para que de esta manera puedan ser analizados y aprovechados, en diferentes tipos de estudios; se tiene muchas herramientas que realizan esta tarea, desde aquellas que necesitan una configuración previa y conocimientos de programación, hasta aquellas que funcionan en un entorno web donde se proporcionan la url de los sitios y los campos a extraer. para la realización de esta tarea, nosotros utilizamos la herramienta Selenium.

Selenium es una herramienta de automatización del navegador web, Principalmente, es para automatizar aplicaciones web con fines de prueba, pero ciertamente no se limita a eso. Te permite abrir un navegador de tu elección, hacer clics automatizados, Ingresar

información en formularios Buscar información específica en las páginas web, extraer información, etc. En otras palabras, selenium te permite controlar de manera automática el navegador como si fuera un humano, por medio de un webdriver.

3.4.4. Mongodb.

Mongodb es una base de datos NoSQL orientada a documentos y de código abierto, esto significa que no guarda los datos como generalmente lo haría una base de datos SQL, sino que, en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

En mongodb las tablas pasarían a ser denominadas "colecciones" y cada fila de esas tablas se llamarían "documentos".

3.4.5. PyMongo.

Pymongo es una librería de Python para poder conectarnos a una base de datos MongoDB, gracias a ella podemos interactuar con la base de datos directamente desde el código, sin necesidad de recurrir directamente al cliente de mongodb desde la terminal.

3.4.6. Robo 3T.

Robo 3T es una herramienta disponible para varias plataformas como windows, linux y Mac, te permite interactuar de manera gráfica con la base de datos mongodb de manera fácil y rápida, además de ser muy intuitiva.

3.4.7. Freeling

Freeling como mencionamos en apartados anteriores es la herramienta que escogimos para las tareas de PLN, debido a que es bastante robusta en trabajos del idioma español, su instalación es medianamente tediosa porque requiere de configuraciones previas para ello, sin embargo estas instrucciones están aclaradas en su documentación oficial disponible en inglés, que seguramente con una lectura detallada se podrá realizar fácilmente , en nuestro caso tuvimos que realizar configuraciones adicionales de librerías debido a la versión de nuestro sistema operativo, puede ver el apartado de anexos para más información acerca del proceso.

3.4.8. Gensim.

Gensim es una librería para python que sirve para extraer automáticamente los temas semánticos de los documentos de la forma más eficiente y con menos complicaciones posibles, para que de esta manera pueda crear modelos de lenguaje luego de un entrenamiento con un corpus de texto extenso. Es un gran paquete para procesar textos, trabajar con modelos de vectores de palabras (como Word2Vec, FastText, etc.) otra ventaja importante con gensim es que te permite manejar archivos de texto grandes sin tener que cargar todo el archivo en la memoria.

en este trabajo haremos uso de esta herramienta para poder entrenar un modelo word2vec, con datos relacionados al turismo y extraídos gracias a las técnicas de web scraping.

3.4.9. Sci-Kit Learn

Se trata de una librería de python estrictamente para tareas de machine learning o aprendizaje automático, que pone a disposición del usuario varios algoritmos de clasificación, regresión, análisis de resultados, pruebas, etc.

En este proyecto hicimos uso de el algoritmo SVM support vector machines, para las tareas de clasificación, así como también se usó las herramientas de separación de datos y métricas de evaluación.

3.5. Tipo de Investigación.

La presente investigación es de tipo aplicada y tecnológica ya que se realizaron experimentos con datos reales proporcionados por personas que visitaron los destinos turísticos, atravez de fases de investigación, establecimos nuevas estrategias de solución a partir de los procedimientos iniciales hasta lograr soluciones finales aceptables. Además, realizamos la aplicación practica de las técnicas Porcesamiento de lenguaje Natural para el análisis computacional de textos.

3.6. CRISP –DM

CRISP-DM es una metodología de minería de datos integral y un modelo de proceso que proporciona a cualquier persona, desde principiantes hasta expertos en minería de datos, un plan completo para llevar a cabo un proyecto de minería de datos, esta metodología consta de seis faces claramente definidos Figura 18, donde las flechas indican las dependencias que son necesarias , este ciclo de faces no es necesariamente estricta ya que permite que se

avance o se retroceda en el desarrollo de las faces, y el resultado determina que se debe de realizar luego de haberlo terminado, el círculo externo en la figura simboliza la naturaleza cíclica de los proyectos de análisis de datos. El proyecto no se termina una vez que la solución se despliega. La información descubierta durante el proceso y la solución desplegada pueden producir nuevas iteraciones del modelo. Los procesos de análisis subsecuentes se beneficiarán de las experiencias previa.

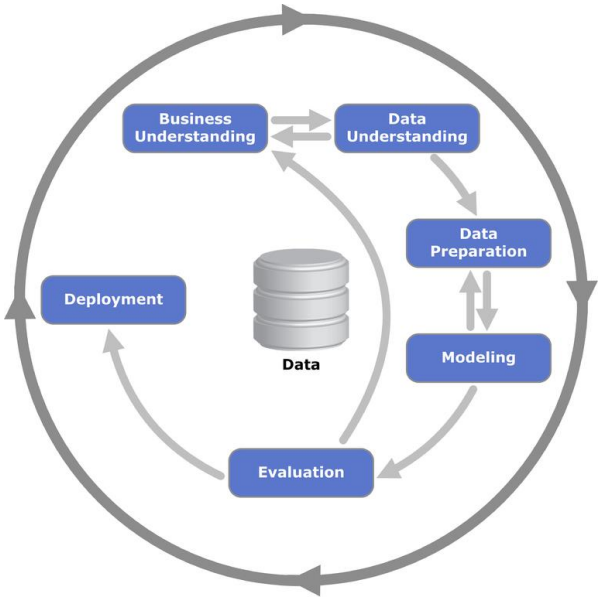


Figura 18. Metodología Crisp-DM.
Fuente: https://data.sngular.com/es/art/25/crisp-dm-la-metodologia-para-poner-orden-en-los-proyectos-de-data-science.

A continuación, describiremos las fases de la metodología.

3.6.1. Business Understanding (Comprensión del negocio).

Esta fase busca principalmente hacer la comprensión o el entendimiento de los principales objetivos del negocio y los requerimientos, para que de esta manera se tenga una definición específica del problema de minería de datos.

3.6.2. Data Understanding (Estudio y comprensión de los datos).

Esta fase, busca la recolección de datos inicial y la familiarización con estos, además de identificar cuáles son los problemas de calidad, para, hacer una mirada previa a los resultados que se obtendrían.

3.6.3. Data Preparation (análisis de los datos y selección de características).

Esta fase está encargada de la preparación de los datos para que se obtenga el conjunto final de los datos que se usaran en el modelado, esto se realiza por medio de actividades como la limpieza de datos, selección de atributos o transformación de estos.

3.6.4. Modeling (modelado).

Para el modelado es necesario aplicar las herramientas de data mining existentes, siempre en cuando estén en armonía con el problema de aplicación.

3.6.5. Evaluation (evaluación, obtención de resultados).

Una vez que se haya construido y aplicado el modelado en los datos, es necesario que sean evaluados, con el fin de comparar los resultados obtenidos con los objetivos del negocio, al final de esta fase, se debería obtener una decisión sobre la aplicación de los resultados del proceso de análisis de datos.

3.6.6. Deployment (puesta en producción).

Poner en práctica los modelos resultantes, dependiendo de los requisitos, la fase de desarrollo puede ser tan simple como la generación de un informe o tan compleja como la realización periódica y quizás automatizada de un proceso de análisis de datos en la organización.

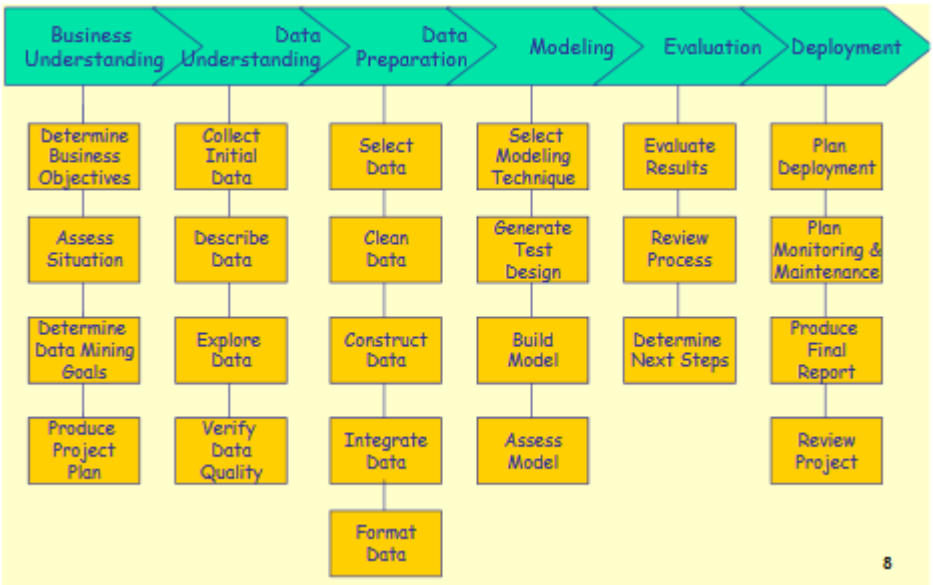


Figura 19. Fases y tareas CRIP-DM.
Fuente: https://paginas.fe.up.pt/~ec/files_0405/slides/02%20CRISP.pdf.

3.7. Desarrollo de la investigación. Según la metodología CRISP-DM

Para que llevemos a cabo de manera eficiente el proyecto de MO, se seguirá la metodología Crisp-DM, que detallamos en el apartado anterior, este nos permitió que el proyecto tenga trazabilidad y escalabilidad alrededor de la perspectiva de negocio.

En los siguientes apartados describiremos cada una de las fases aplicada a nuestro proyecto.

3.7.1. Comprensión del negocio.

3.7.1.1. Objetivos del negocio.

Los objetivos que fueron descritos en el apartado 1.2 representan nuestros objetivos de negocio.

3.7.1.2. Evaluación de la situación.

Con la amplia variedad de información disponible en la red, especialmente de los comentarios en línea en diversas páginas web dedicadas exclusivamente al turismo, buscamos evaluar cuáles de esas son las que se adecuaron mejor a nuestro caso de estudio. Tuvimos que considerar cuales de ellas son las más populares, ya que mientras más populares mayor cantidad de comentarios se tendrá en la página, además una tarea de suma importancia fue considerar aquellas páginas en donde los comentarios disponibles también este en el idioma español y tengan una valoración de los lugares a los que se refiere, ya que este fue la clase de separación para nuestro clasificador.

Además, también se tuvo que tener en cuenta que el estudio está exclusivamente enfocado a comentarios de los destinos turísticos de la región de Puno, que está ubicada en Perú, así que la página seleccionada para poder extraer los comentarios debía tener comentarios referidos a los destinos turísticos de la región.

Como dijimos, existen gran cantidad de páginas dedicadas a viajes, turismo, hospedaje, renta de carros etc. Nosotros nos concentramos exclusivamente a páginas, dedicadas a viajes y turismo, según la página similarweb https://www.similarweb.com/top-websites/category/travel-and-tourism para el día en el que se hizo la consulta el 19 de febrero del 2019 a las 18:00 horas, las 3 mejores webs de viajes y turismo son.

Tabla 3.

Webs dedicadas al rubro de viajes.

Nombre	Rubro
booking.com	Viajes y turismo Alojamiento y hoteles.
tripadvisor.com	Viajes y turismo
airbnb.com	Viajes y turismo Alojamiento y hoteles.

Fuente: Elaboración propia.

De esta manera revisando la disponibilidad de comentarios y otras características ya mencionadas se eligió a la página tripadvisor.com como fuente de nuestros datos.

Otra de las cuestiones de comprensión del negocio es, sobre qué lugares se debería extraer los datos, así pues, se tomaron como referencia a las estadísticas proporcionadas por el Mincetur en su portal web Turismo in, sobre los lugares más visitados por los turistas en el año 2017 ver Figura 20, así que se tomaron a esos destinos más visitados, para posteriormente extraer sus comentarios.

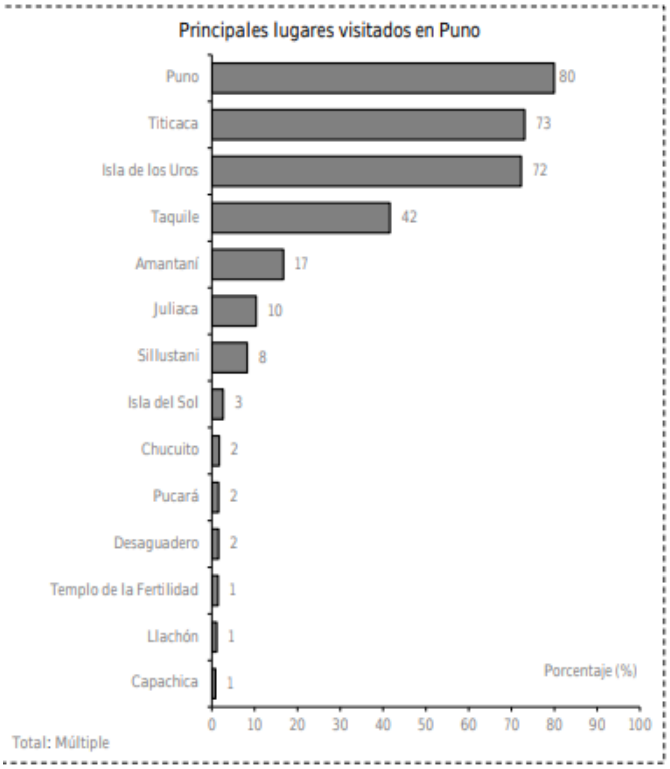


Figura 20. Principales lugares visitados en Puno en el 2017.
fuente: Turismo in, recuperado de https://www.promperu.gob.pe/TurismoIN/sitio/VisorDocumentos.

De manera adicional necesitábamos otro set de datos exclusivamente para poder entrenar nuestro modelo World2Vec, que sirvió para extraer la representación vectorial de las

palabras presentes en los comentarios de los destinos de la región de Puno, esto con el objetivo de realizar una comparación de resultados entre un modelo que entrenamos exclusivamente con comentarios en el dominio de turismo , y otro con comentarios en español de diferentes dominios, cuyo procedimiento detallaremos más adelante; en este caso lo que necesitábamos era solo la mayor cantidad de comentarios posibles relacionados al turismo, así que se buscó la mayor cantidad de destinos turísticos posibles, en este caso del Perú.

3.7.2. Comprensión de los datos.

En esta fase lo que buscamos es tener un primer contacto con los datos, haciendo una exploración de los mismos, y comprobamos su calidad.

Como dijimos los datos se encuentran publicadas en internet, en forma de comentarios que se les dan a los destinos turísticos, por ejemplo, como se muestra en la figura 21, es un comentario del destino lago Titicaca, donde se tienen los siguientes datos.

- una valoración de 5 estrellas que significan “Excelente”, “Muy Bueno”, “regular”, “malo”, “Pésimo”
- el día en el que se escribió
- mediante qué medio se escribió el comentario
- El usuario que da la opinión
- el título del comentario
- el comentario
- Fecha de experiencia

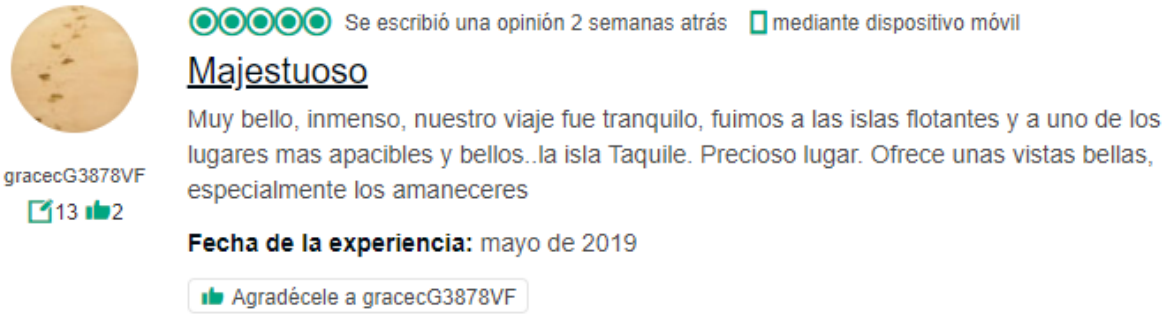


Figura 21. Ejemplo de comentario en tripadvisor.
Fuente: Tripadvisor, recuperado de. tripadvisor.com.

De estos datos disponibles tuvimos que escoger cuales son los necesarios para nuestro objeto de estudio, por lo que a continuación, detallamos cómo fue la extracción de los dos sets de datos, como dijimos un set de datos que contiene comentarios de la región de Puno para la realización del estudio y otra para el entrenamiento del modelo wor2vec conteniendo comentarios de varios destinos del Perú.

3.7.2.1. Extracción de datos para el estudio.

Los datos que consideramos relevantes para la extracción fueron (ver tabla 4), de los cuales los campos más importantes para el estudio son “rating” y “coment”, asimismo cabe aclarar que la clasificación que se realizó con SVM está bajo dos clases, comentarios “positivos” y comentarios “negativos”, motivo por el cual en esta parte del proceso de extracción tuvimos que asignar numeraciones a los comentarios que pertenezcan a alguna de estas clases.

Así pues, en el apartado anterior analizamos los datos a extraer y vimos que el campo “rating” estaba bajo 5 tipos de valoraciones “Excelente”, “Muy Bueno”, “Regular”, “Malo” y “Pésimo”, por lo cual tuvimos que asignar estos 5 tipos de valoraciones a solo 2, “Positivos” o “Negativos”.

Dubiau (2013) en su estudio realizó la clasificación de los textos extraídos de Google Play de la siguiente manera “asignar la etiqueta ”POSITIVO” a aquellos comentarios calificados por el usuario con 5 estrellas (excelente) y la etiqueta ”NEGATIVO” a los comentarios cuya calificación fue de 1 estrella (mala)”.

Por otra parte, la plataforma de descargas de aplicaciones móviles Google Play, clasifica como comentarios positivos a aquellos cuya valoración fue de “4” o “5” estrellas, mientras que asigna como negativos a aquellos cuya valoración fue de “1”, “2” o “3”, ver Figura 22.

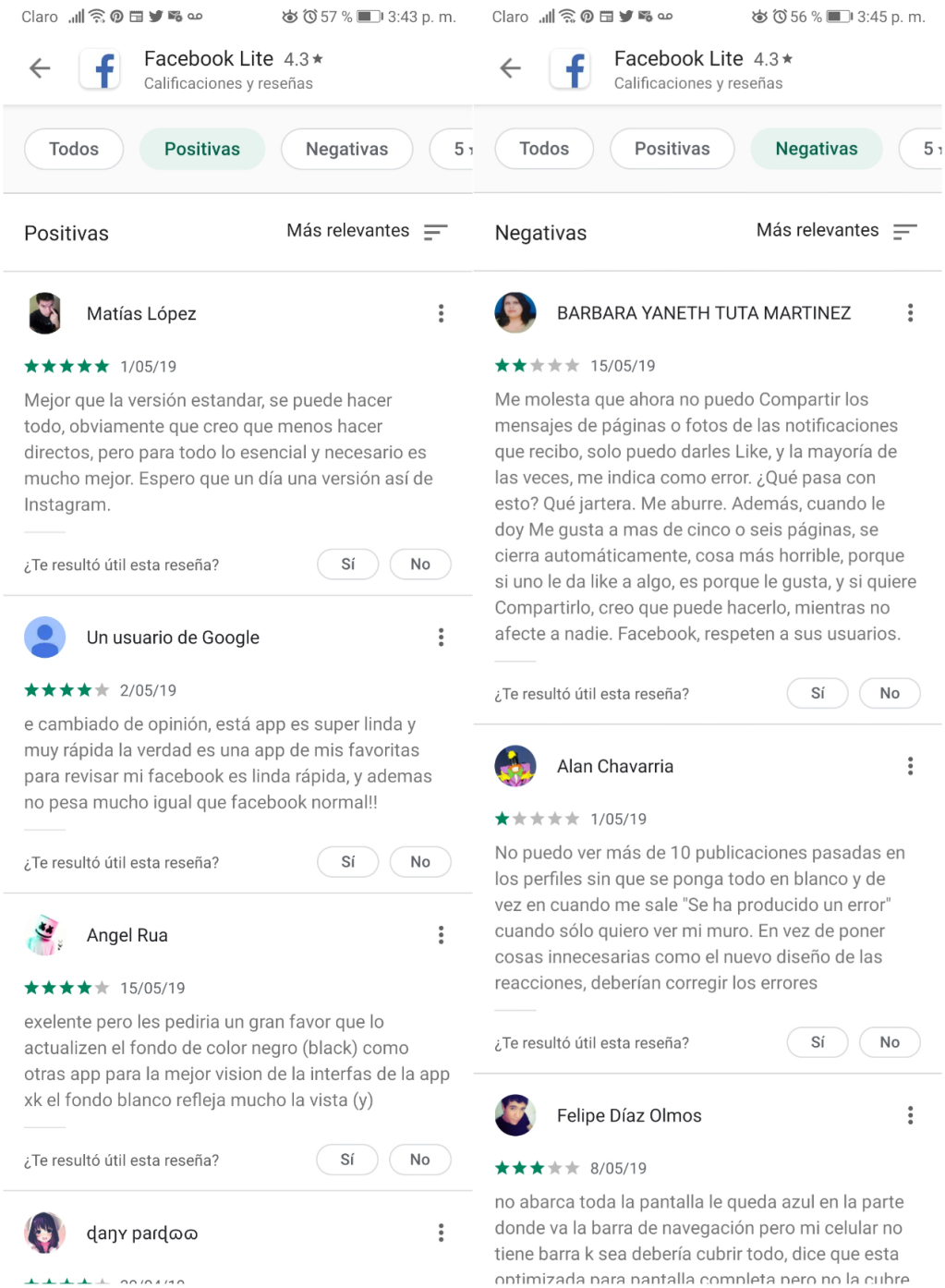


Figura 22. Asignación de clases positiva o negativa de Google Play.
Fuente: play.google.com.

De manera que, para este estudio realizamos la clasificación de los comentarios, de la siguiente manera. Aquellos comentarios cuyas valoraciones sea “Excelente” y “Muy bueno”, se consideró que pertenecen a la categoría de comentarios “positivos” asignándoles una numeración de 6, Figura 23.

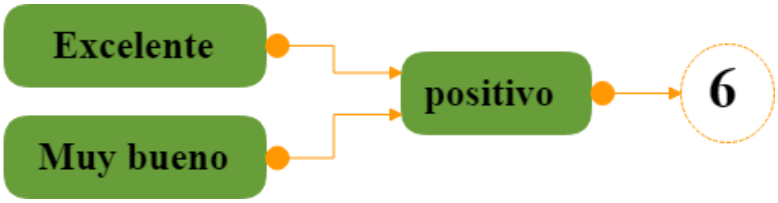


Figura 23. Asignación de datos a la clase Positiva.
Fuente: Elaboración propia.

Por otro lado, aquellos comentarios que tengan valoraciones de "Regular", "Malo" y "Pésimo", se consideró que pertenece a la clase de comentarios "negativos" asignándole una numeración de 1, Figura 24.

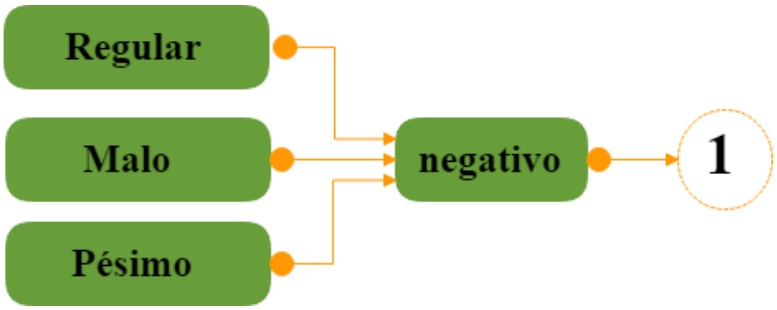


Figura 24. Asignación de datos a la clase Negativa.
Fuente: Elaboración propia.

Tabla 4.

Datos extraídos.

campos	**descripción**
reviewid	El id de la review establecida en trip advisor
zone	la zona a la que pertenece el comentario
rating	La valoración que el usuario le dio al destino turístico
date	Fecha en que se creó el comentario
coment_title	El título del comentario
coment	El comentario que escribió el usuario respecto al destino turístico
exp_date	Fecha en la que el usuario tuvo la experiencia en el destino turístico

Fuente: Elaboración Propia.

Para la extracción de esta información desde el sitio web tripadvisor tuvo que realizarse un script en Python, con la utilización de selenium.

```
45
46      def get_coments(self, url,name):
47          # obtener la url para la extraccion
48          self.driver.get(url)
49          #variable data
50          self.data = []
51          # Obtenemos la cantidad de comentarios del lugar
52          all_coments = re.findall(
53              r'\d+,\d+|\d+', self.driver.find_element_by_class_name('pagination-details').text)[-1]
54          print('N° de comentarios: ', all_coments)
55          # invocamos el metodo qeu extrae los comentarios
56
57          self.__get_page_coments(name)
58
59      def __get_page_coments(self, *args):
60          # agregamos un script en java script para que los comentarios que permanecen ocultos apareccan
61          self.driver.execute_script(
62              'Array.from(document.getElementsByClassName("ulBlueLinks")).forEach(function(item){item.click()});')
63          time.sleep(1)  # Necesario porque se demora actualizar el DOM
64          reviews = self.driver.find_elements_by_class_name('review-container')
65          #print (reviews[:5])
66
67          for review in reviews:
68
69              try:
70                      #agregamos a un diccionario todos los elemeto extraidos
71                      self.data.append(dict(
72                          reviewid = review.get_attribute('data-reviewid'),
73                          zone= ''.join(map(str, args)),
74                          rating =((review.find_element_by_class_name('ui_bubble_rating').get_attribute('class')).split("_")[-1])[0],
75
76                          date = review.find_element_by_class_name('ratingDate').get_attribute('title'),
77                          coment_title=self.driver.execute_script(
78                              'return arguments[0].querySelector(".noQuotes").textContent', review),
79                          coment=self.driver.execute_script(
80                              'return arguments[0].querySelector(".partial_entry").textContent', review),
81                          exp_date = (review.find_element_by_class_name('prw_reviews_stay_date_hsx').text)[25:]
82                      ))
83
84              except Exception as e:
85                  print(e)
86          #acciones para dar click next y pasar de casilla de comentarios
87          has_next = self.driver.execute_script(
88              'return !document.querySelector(".ui_pagination>.next").classList.contains("disabled")')
89          print("Count: ", len(self.data))
90          if len(self.data)<9999:
91              if has_next:
92                  self.driver.execute_script(
93                      'document.querySelector(".ui_pagination>.next").click()')
94                  time.sleep(2)
95                  self.__get_page_coments(name)
96
97          #print(self.data)
98          #print(type(self.data))
99
```

Figura 25. Script para la extracción de comentarios.
Fuente: Elaboración propia.

Luego de la extracción, los datos fueron insertados a una colección llamada “review”, siempre en cuando este aun no haya sido insertado y no exista en la colección. Como resultado de la ejecución del script desarrollado, se obtuvo una cantidad de “8955” comentarios de los siguientes sitios ver Tabla 5, y guardados en la colección bajo el siguiente formato, ver Figura 26.

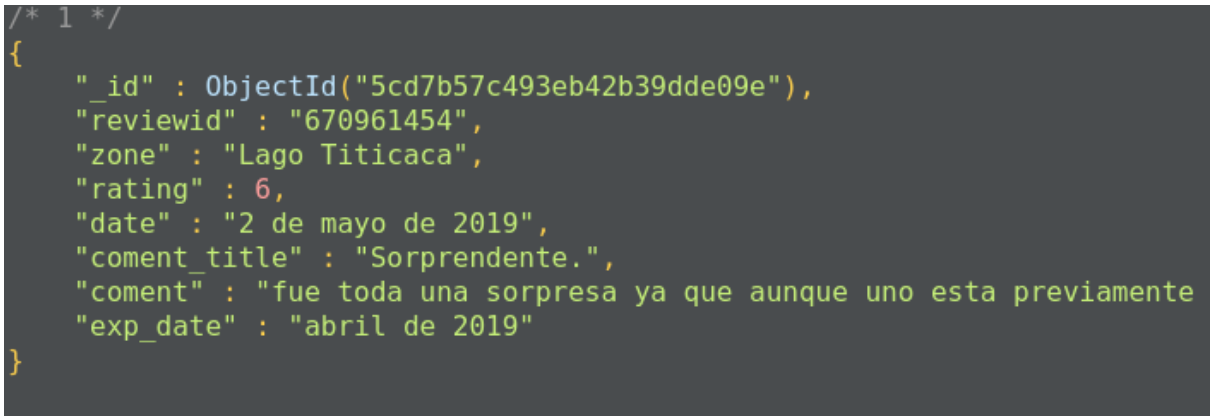


Figura 26. Colección insertada a mongodb.
Fuente: Elaboración propia.

Tabla 5.

Número de comentarios extraídos por destino.

Destino Turístico	**Número de comentarios del destino**
Lago Titicaca	1101
Isla Flotante los Uros	3797
Isla Taquile	2685
Isla Amantani	206
Sillustani	346
Portal de Aramu Muru	101
Catedral de Puno	541
Chucuito	178
total	8955

Fuente: Elaboración propia.

*3.7.2.2. E*xtracción de datos para el entrenamiento del modelo.

En relación a la obtención de un modelo Word2vec, exclusivamente con comentarios turísticos que utilizaremos para obtener las representaciones vectoriales de las palabras, tuvimos que extraer la mayor cantidad posible de comentarios existentes.

Por tanto, para esta tarea extrajimos los comentarios de diversos destinos turísticos del Perú ver Figura 27, y así generar un vocabulario muy amplio, estos comentarios fueron sacados como en el caso anterior de tripadvisor, y la extracción fue muy similar a la manera como se obtuvieron los datos para el estudio, es importante precisar que estos datos obtenidos tienen la misma estructura o campos, sin embargo en la tarea de generar el vocabulario y entrenamiento del modelo, solo se utilizará el campo “coment”, puesto que

ahí se encuentran las sentencias que deberán ser representadas bajo un espacio vectorial; la cantidad de comentarios extraídos se puede ver en la Tabla 6.

```
#url de cada uno de los destinos a extraer comentarios
atractions_urls = [('Huayna Picchu','https://www.tripadvisor.com.pe/Attraction_Review-g294318-d548890'),
                   ('Salinas de Maras','https://www.tripadvisor.com.pe/Attraction_Review-g304042-d553972'),
                   ('Miraflores','https://www.tripadvisor.com.pe/Attraction_Review-g294316-d311718'),
                   ('Intipuncu','https://www.tripadvisor.com.pe/Attraction_Review-g294318-d548895'),
                   ('Pisac','https://www.tripadvisor.com.pe/Attraction_Review-g294320-d549817'),
                   ('Chan Chan','https://www.tripadvisor.com.pe/Attraction_Review-g298444-d311730'),
                   ('Lineas De Nazca','https://www.tripadvisor.com.pe/Attraction_Review-g304044-d311724'),
                   ('Huaca del Sol y la Luna','https://www.tripadvisor.com.pe/Attraction_Review-g298444-d547645'),
                   ('Nadando con Tortugas en El Nuro','https://www.tripadvisor.com.pe/Attraction_Review-g667820-d6486862'),
                   ('Huaca Pucllana','https://www.tripadvisor.com.pe/Attraction_Review-g294316-d311722'),
                   ('El Parque del Amor','https://www.tripadvisor.com.pe/Attraction_Review-g294316-d550142'),
                   ('Fortaleza de Kuelap','https://www.tripadvisor.com.pe/Attraction_Review-g799618-d2536313'),
                   ('Sarcófagos de Karajia','https://www.tripadvisor.com.pe/Attraction_Review-g799618-d3748693'),
                   ('Complejo Arqueologico Wari','https://www.tripadvisor.com.pe/Attraction_Review-g316041-d5483571'),
                   ('Laguna 69','https://www.tripadvisor.com.pe/Attraction_Review-g318878-d6628275'),
                   ('Laguna Parón','https://www.tripadvisor.com.pe/Attraction_Review-g304037-d3633267'),
                   ('Cordillera Huayhuash','https://www.tripadvisor.com.pe/Attraction_Review-g2656665-d1034375'),
                   ('Monasterio de Santa Catalina','https://www.tripadvisor.com.pe/Attraction_Review-g294313-d312030'),
                   ('Volcan Misti','https://www.tripadvisor.com.pe/Attraction_Review-g294313-d776566'),
                   ('Plaza de Armas','https://www.tripadvisor.com.pe/Attraction_Review-g294313-d313683'),
                   ('Mirador Cruz del Cóndor','https://www.tripadvisor.com.pe/Attraction_Review-g798819-d9870207'),
                   ('Santuario Histórico de Machu Picchu', 'https://www.tripadvisor.com.pe/Attraction_Review-g294318-d668949'),
                   ('San Blas', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d311932'),
                   ('Montaña de Siete Colores', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d10311540'),
                   ('Tipón', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d2691856'),
                   ('Tambomachay', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d2318800'),
                   ('Ausangate', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d3385305'),
                   ('Sacsayhuaman', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d311726'),
                   ('Camino Inca', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d311715'),
                   ('Centro Historico De Cusco', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d2422362'),
                   ('Plaza de Armas (Huacaypata)', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d311933'),
                   ('Iglesia De Santo Domingo', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d316238'),
                   ('Catedral de Cusco', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d311935'),
                   ('Laguna Humantay', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d12158251'),
                   ('San Blas', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d311932'),
                   ('Montaña de Siete Colores', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d10311540'),
                   ('Tipón', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d2691856'),
                   ('Tambomachay', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d2318800'),
                   ('Ausangate', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d3385305'),
                   ('Piedra de los doce angulos', 'https://www.tripadvisor.com.pe/Attraction_Review-g294314-d553923')
                   ]
```

Figura 27. Destinos de las que se obtuvieron los comentarios.
Fuente: Elaboración propia.

Tabla 6.

Cantidad de comentarios extraídos para el entrenamiento del modelo.

Número de destinos	**Número de comentarios**
40	84596

Fuente: Elaboración propia.

3.7.3. Preparación de los datos.

En esta fase se buscará preprocesar el conjunto de datos que se obtuvo, y consta de fases en las que aplicaremos algunas de las técnicas de preprocesamiento que describimos en el apartado 2.4, con la finalidad de obtener buenos resultados en nuestra fase de análisis que describiremos más adelante.

Luego de esta transformación de datos, el resultado será un corpus de datos listos para iniciar los procedimientos de análisis.

3.7.3.1. Eliminación de caracteres y paseo a minúsculas.

En primer lugar, es importante decir que bajar en minúscula todos nuestros datos, es una de las formas más simples y efectivas de preprocesamiento de texto. Es aplicable a la mayoría de los problemas de minería de texto y PNL y puede ayudar en los casos en que su conjunto de datos no es muy grande y ayuda significativamente con la consistencia del resultado esperado.

En segundo lugar, la eliminación de ruido (Noise removal) consiste en eliminar los dígitos de los caracteres y las piezas de texto que pueden interferir con el análisis de texto, además es altamente dependiente del dominio, incluye eliminación de puntuación, eliminación de caracteres especiales, eliminación de números, eliminación de formato html, eliminación de palabras clave específicas del dominio (por ejemplo, 'RT' para retweet), eliminación de código fuente, eliminación de encabezados y más.

De ahí que estos procedimientos esenciales son los que se aplicaron en nuestro trabajo de investigación. La primera tarea fue pasar todos los textos a minúsculas, luego los caracteres que se extrajeron del corpus fueron como por ejemplo puntos ".", comas ",", signos de slash "/", signos de admiración "!", porque estos no añaden ni un tipo e significado en nuestros textos, ejemplo.

Tabla 7.

Ejemplo eliminación de caracteres y parseo a minúsculas.

opiniones originales	**opiniones con eliminación de caracteres y paseo a minúsculas**
"La vista del lago es fabulosa, me encanto el amanecer... Y el atardecer es para quedarse CONTEMPLÁNDOLO horas!!! :)"	"la vista del lago es fabulosa me encanto el amanecer y el atardecer es para quedarse contemplándolo horas"
"Muy bello!! lugar muy bien cuidado. seguro pero mucho cuidado con la altura eso dificulta un poco el viaje pero un lugar de (sueños) :)."	"muy bello lugar muy bien cuidado seguro pero mucho cuidado con la altura eso dificulta un poco el viaje pero un lugar de sueños"
"Excelente lugar lleno de energía.. con hermosos paisajes / & más aún los pobladores que viven en esta zona conservando su "cultura". "	"excelente lugar lleno de energía con hermosos paisajes y más aún los pobladores que viven en esta zona conservando su cultura"

Fuente: Elaboración propia.

3.7.3.2. Eliminación de stopwords

Para que los el estudio se centre en las palabras que aportan información al texto, tuvimos que en primer lugar segmentar cada documento en palabras individuales, y facilitar su

búsqueda en una lista de stopwords en español obtenidas del siguiente repositorio https://github.com/stopwords-iso/stopwords-es/blob/master/stopwords-es.txt, para que de esta manera se eliminen de los documentos, por ejemplo.

Tabla 8.

Ejemplo Segmentación de comentario.

opiniones con eliminación de caracteres y parseo a minúsculas	opiniones segmentadas
“la vista del lago es fabulosa me encanto el amanecer y el atardecer es para quedarse contemplándolo horas”	“la”, “vista”, “del”, “lago”, “es”, “fabulosa”, “me”, “encanto”, “el” “amanecer”, “y”, “el”, “atardecer”, “es”, “para”, “quedarse”, “contemplándolo”, “horas”
“muy bello lugar muy bien cuidado seguro pero mucho cuidado con la altura eso dificulta un poco el viaje pero un lugar de sueños”	“muy”, “bello”, “lugar”, “muy”, “bien”, “cuidado”, “seguro”, “pero”, “mucho”, “cuidado”, “con”, “la”, “altura”, “eso”, “dificulta”, “un”, “poco”, “el”, “viaje”, “pero”, “un”, “lugar”, “de”, “sueños”
“excelente lugar lleno de energía con hermosos paisajes y más aún los pobladores que viven en esta zona conservando su cultura”	“excelente”, “lugar”, “lleno”, “de”, “energía”, “con”, “hermosos”, “paisajes”, “y”, “más”, “aún”, “los”, “pobladores”, “que”, “viven”, “en”, “esta”, “zona”, “conservando”, “su”, “cultura”

Fuente: Elaboración propia.

El resultado será como la tabla siguiente, para posteriormente ser ingresada a la base de datos.

Tabla 9.

Ejemplo eliminación de Stopwords.

opiniones segmentadas	opiniones con eliminación de stopwords
“la”, “vista”, “del”, “lago”, “es”, “fabulosa”, “me”, “encanto”, “el” “amanecer”, “y”, “el”, “atardecer”, “es”, “para”, “quedarse”, “contemplándolo”, “horas”	“vista lago fabulosa encanto amanecer atardecer quedarse contemplándolo”
“muy”, “bello”, “lugar”, “muy”, “bien”, “cuidado”, “seguro”, “pero”, “mucho”, “cuidado”, “con”, “la”, “altura”, “eso”, “dificulta”, “un”, “poco”, “el”, “viaje”, “pero”, “un”, “lugar”, “de”, “sueños”	"bello lugar cuidado seguro cuidado altura dificulta viaje lugar sueños"
“excelente”, “lugar”, “lleno”, “de”, “energía”, “con”, “hermosos”, “paisajes”, “y”, “más”, “aún”, “los”, “pobladores”, “que”, “viven”, “en”, “esta”, “zona”, “conservando”, “su”, “cultura”	"excelente lugar lleno energía hermosos paisajes pobladores viven zona conservando cultura"

Fuente: Elaboración propia.

3.7.3.3. Tokenización, Lematización y POS.

Para finalizar las tareas de preprocesamiento aplicadas en este trabajo de investigación, utilizamos freeling como herramienta fundamental para las acciones de tokenización, Lematización y etiquetado POS.

Para que freeling funcione correctamente necesitamos cargar a nuestro proyecto sus módulos disponibles para la utilización en Python, estos módulos se obtienen una vez que se haya instalado el Api para Python, disponibles para la versión 3 y 2 ver sección de anexos para más información.

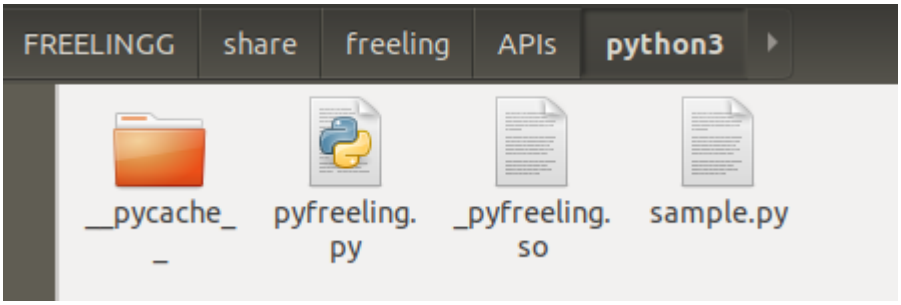


Figura 28. Módulos de carga api freeling para python3.
Fuente: Elaboración propia.

En la imagen de implementación del script Figura 28, lo que realizamos en primer lugar es hacer la conexión a la base de datos "removestopwords" que es donde se ubica nuestro corpus con los stopwords eliminados, seguidamente añadimos la configuración de freeling para python, estas configuraciones incluyen la carpeta donde se instaló freeling, cargar los elementos referentes al lenguaje en el que se va a trabajar, en este caso el español "LANG = es " y cargar los módulos disponibles para el procesamiento de texto, nosotros activamos los módulos de análisis morfológico que son necesarias para las acciones que realizaremos, la descripción de estos módulo se encuentran en la documentación oficial.

Seguidamente en la línea 98 realizamos una búsqueda de los destinos turísticos que poseen comentarios, para seguidamente agregarlos a una variable "rev" e iniciar el proceso de análisis.

```
97
98  def query_names(): # Método que devuelve los nombres de los sitios que tienen reviews
99      query = reviews.distinct('zone')
100
101     return query
102
103 def get_data(name): #Obtiene las reviews de un sitio determinado
104
105     rev = reviews.find({'zone': name}, {'_id': 0})
106
107     return rev
108
109 def freeling_analysis(lin, p_a_f_e, p_a_s): #Metodo que recibe la review, y la analiza con freeling para obtener la FC y la etiqueta.
110
111     while(lin): #Mientras no se acabe el texto introducido
112         l = tk.tokenize(lin) # Tokenizamos
113         ls = sp.split(sid, l, True) # Dividimos la cadena
114         ls = mf.analyze(ls) # Analizador morfológico
115         ls = tg.analyze(ls) #Permite obtener entidades (lugares, nombres propios, etc)
116         global parsed_string
117
118         for s in ls:
119             ws = s.get_words()
120             for w in ws:
121
122                 print("_____")
123                 parsed_string += w.get_lemma() + " "
124                 print(parsed_string)
125                 print("\n")
126                 p_a_s.append(w.get_lemma())
127                 print(p_a_s)
128                 print("\n")
129
130                 p_a_f_e.append({'FC': w.get_lemma(), 'Etiqueta': w.get_tag()})
131                 print(p_a_f_e)
132                 print("\n")
133                 print("_____")
134
135
136         break
137
```

Figura 29. Script para la lematización, tokenización y POS con freeling.
Fuente: Elaboración propia.

Freeling divide sus procesamientos dentro de tres tipos ubicadas en diferentes clases, la clase" párrafo", "oración" y "palabra", es una especie de subdivisión lógica. Los párrafos son el elemento principal de los textos, las oraciones son los elementos principales de los párrafos y las palabras son los elementos principales de las oraciones, así que cuando se trabaja con freeling uno decidirá bajo que niveles desea realizar el procesamiento de texto, en nuestro caso lo realizamos a nivel de palabra por lo que utilizamos la clase "word".

Clase word: Esta clase se encarga principalmente de almacenar la forma de una palabra e información relativa, como por ejemplo su codificación fonética y gramatical entre otros.

El módulo tokenizer: Convierte texto sin formato, es decir texto plano en un vector de objetos" word", de acuerdo con un conjunto de reglas de tokenización, basadas en expresiones regulares.

El módulo Splitter: Recibe listas de objetos de palabras (ya sea producidos por el tokenizador o por cualquier otro medio a) y los almacena temporalmente en el buffer hasta que se detecte un límite de oración.

Analizador morfológico: Se trata de un meta-módulo que no realiza un procesamiento por sí mismo, sino que su declaración sirve para inicializar todos los sub módulos que ofrece,

estos sub módulos son los encargados de analizar las palabras y pueden ser recuperadas con funciones.

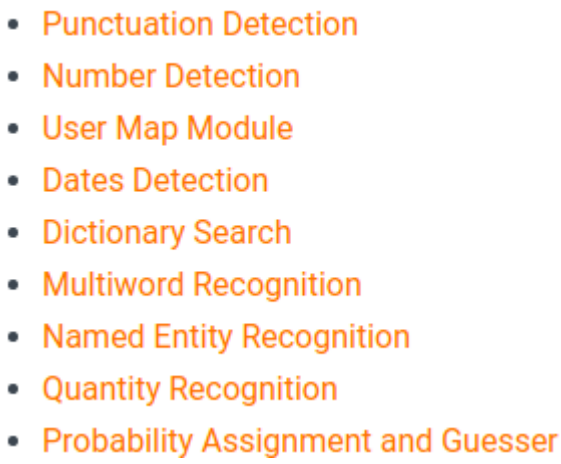


Figura 30. Módulos del analizador morfológico freeling.
Fuente: Talp-upc, recuperado de: https://talp-upc.gitbook.io/freeling-4-1-user-manual/processing-classes/maco.

Finalmente, tras aplicar todos los procesos anteriormente descritos con la herramienta freeling, tuvimos que guardar los datos en una nueva colección "freelingprocesor" de la base de datos "turistsentiment", donde los datos procesados se mostraran de la siguiente manera.

Tabla 10.

Ejemplo Tokenización, Lematizacion y POS.

opiniones con eliminación de stopwords	**opiniones lematizadas**	**Tokenización**	**POS**
"vista lago fabulosa encanto amanecer atardecer quedarse contemplándolo"	"ver lago fabuloso encanto amanecer atardecer quedar contemplar "	"ver","lago", "fabuloso", "encanto","amanecer", "atardecer","quedar", "se", contemplar", "lo"	"FC" : "ver", "Etiqueta" : "VMP00SF, "FC" : "lago","Etiqueta" : "NCMS000", "FC" : "fabuloso",Etiqueta" : "AQ0FS00", "FC" : "encanto",Etiqueta" : "NCMS000", "FC" : "amanecer",Etiqueta" : "VMN0000" "FC" : "atardecer",Etiqueta" : "VMN0000", "FC" : "quedar",Etiqueta" : "VMN0000", "FC" : "se","Etiqueta" : "PP3CN00", "FC":"contemplar","Etiqueta":"VMG0000 "FC" : "lo","Etiqueta" : "PP3MSA0"

Fuente: Elaboración propia.

3.7.4. Modelado.

En esta fase del desarrollo del proyecto, utilizamos dos enfoques, el primero encargado de entrenar un modelo wor2vec con el fin de aprovechar los vectores que genera y el segundo entrenar el clasificador SVM con los vectores que representan los comentarios y las clases como "positivo" = 6 y "negativos" = 1, a continuación, explicaremos cada una de las acciones realizadas para implementarlos.

3.7.4.1. Modelado Word2vec

Existen modelos Word2vec que fueron desarrollados por diversas empresas como google, investigadores independientes o universidades, y se hacen llamar modelos pre entrenados (pre-trained models) en inglés, disponibles para distintos idiomas, y dominios de estudio, sin embargo en el domino del turismo no encontramos un modelo para poder utilizarlo, pero existe un modelo denominado Spanish Billion Words Corpus and Embeddings, desarrollado por Cristian Cardellino (2016), https://crscardellino.github.io/SBWCE/ ,que está exclusivamente entrenado por una gran cantidad de palabras en español.

Por lo cual, nosotros realizaremos un entrenamiento propio de un modelo con un corpus construido por comentarios en español y que tienen como dominio el turismo, el proceso de obtención del corpus lo describimos en el apartado 3.5.2.2.

De manera que, esto nos permitió realizar una comparación de los resultados de clasificación que se obtienen con el modelo propio y el modelo pre-entrenado, y lo describiremos en la etapa de evaluación.

Para la realización de estas tareas se ha llevado a cabo dos etapas entrenamiento del modelo y la obtención del vector medio, cuya implementación describiremos a continuación.

3.7.4.2. Entrenamiento del modelo

Word2vec es un algoritmo que aprende una palabra a partir de Word embeddings para generar una representación vectorial de esas palabras, proporcionando una proyección en la que palabras con significados similares se agrupan localmente dentro del espacio.

El objetivo en esta fase fue crear un modelo en el que se tendrá un vocabulario de palabras con su respectiva representación vectorial, y para ellos se hizo uso de la herramienta gensim ver figura 31.

```python
from pymongo import MongoClient
from bson.objectid import ObjectId
import logging
import multiprocessing
from gensim.models import Word2Vec
import time

#recueprando comentarios y construyendo arrary de arrays
def mongoDocumentsSplitted( db ):
    #Lista
    splitted_records = []
    #Iterando sobre los datos de los documentos
    for record in db["reviewdatasetfreelingprocesor"].find():
        #agregando comentarios al diccionarios
        splitted_records.append(record["parsed_array_string"])
        #mostrando los comentarios que se estan agregando
        logging.info("estamos agregando {0}\n".format(record["ariginal_coment"]))

    return splitted_records

def main():
    #medir el tiempo del porcesamiento
    t = time.time()
    #Nos porporciona la imformacion de la jecucion
    logging.basicConfig(format="%(levelname)s - %(asctime)s: %(message)s", datefmt= '%H:%M:%S', level=logging.INFO)
    #Conectando a la base de datos
    con = MongoClient('localhost', 27017)
    db = con.turistsentiment # Conectado a la BD "tfg"
    #Guardando los datos devueltos por el metodo de recuperacion de comentarios
    documents= mongoDocumentsSplitted(db)
    #NUmero de nucleos que se utilizaran para el entrenamiento
    cores = multiprocessing.cpu_count()

    print("agregando parametros\n\n")
    w2v_model = Word2Vec(min_count=5,
                         window=4,
                         size=300,
                         workers=cores)

    print("construyendo vocbulario\n\n")
    w2v_model.build_vocab(documents, progress_per=10000)
    print("entrenando el modelo\n\n")
    w2v_model.train(documents, total_examples=w2v_model.corpus_count, epochs=30, report_delay=1)
    print("guardando modelo\n\n")
    w2v_model.save('mimodelo.txt')
    print('tiempo del entrenamiento: {} mins'.format(round((time.time() - t) / 60, 2)))

if __name__ == "__main__":
    main()
```

Figura 31. Script de entrenamiento del modelo.
Fuente: Elaboración propia.

Los datos que servirán para el entrenamiento del modelo deberán ser puestos a un formato en particular, específicamente cada sentencia deberá ser tokenizada, y en algunos casos preprocesada según el estudio, veamos como realizamos este proceso.

en la línea 9 tenemos un método llamado “mongoDocumentsSplitted”, este método recibe como argumento la base de datos, y se encarga de recuperar todos los datos de nuestra colección “reviewdatasetfreelingprocesor”, sin embargo, como dijimos el único campo que será importante para el entrenamiento son los comentarios, estos comentarios deberán ser antes preprocesados y tokenizados, tarea que realizamos de forma similar a lo descrito en los apartados 3.5.2.4, 3.5.2. y 3.5.2.6.

De ahí que este método recuperara los comentarios tokenizados y preprocesados, que almacenamos anteriormente en el campo “parsed_array_string” de la colección “reviewdatasetfreelingprocesor”. Para posterior mente ser insertados en un array llamado “splitted_records”. Por lo tanto, esencialmente estamos pasando al entrenamiento una lista

de listas, donde cada lista dentro de la lista principal, contiene un conjunto de tokens de un comentario.

En la línea 32 realizamos un conteo de los núcleos disponibles en nuestro equipo, en nuestro caso poseemos 4 núcleos y los utilizamos todos con el fin de acelerar el proceso de entrenamiento. Luego procedemos al entrenamiento del modelo en 3 pasos.

*3.7.4.2.1. Word2Vec**():***

En primer lugar, fue asignar los parámetros de modelo.

mincount: Es el número mínimo de veces que tiene que aparecer una palabra en el corpus para no ser descartada y añadirla al vocabulario. El tiempo de entrenamiento se incrementa exponencialmente cuanto más pequeño es este número, ya que se considerarán más casos.

size: Dimensión del vector numérico resultante. Cuanto más grande, más exactitud, pero también más tiempo de entrenamiento.

window: Este valor denota el contexto que se usará, es decir, el número de palabras vecinas

workers: Es el número de núcleos del computador que se utilizaran, en este caso = 4

3.7.4.2.2. Build_vocab():

Word2Vec nos obliga a construir la tabla de vocabulario (simplemente digiriendo todas las palabras y filtrando las palabras únicas, y haciendo algunas cuentas básicas sobre ellas), aquí ingresamos nuestros datos lo cual inicializa el modelo e inicia la construcción del vocabulario.

3.7.4.2.3. Train():

Finalmente, entrena el modelo. añadiendo los registros del proceso que nos servirán para el monitoreo, asegurándose de que no se ejecuten subprocesos de forma instantánea.

3.7.4.3. Obtención del vector numérico medio

El objetivo es encontrar un vector numérico medio para cada opinión del corpus de datos del caso de estudio, para ello realizamos los scripts necesarios de recuperación de información, comparación de palabras con el modelo, extracción de vectores que representan las palabras y cálculo del vector medio de cada opinión, y serán ejecutadas para cada modelo,

el que se creó con nuestro corpus de datos para el entrenamiento y para el modelo Spanish Billion Words Corpus. A continuación, detallaremos como fue dicha implementación

En primer lugar, tuvimos que cargar nuestro corpus de datos y también los modelos necesarios para el procedimiento ver Figura 32.

```
29
30  # CARGAR EL MODELO ENTRENADIO PROPIO
31  '''print("Cargando modelo...")
32  model = gensim.models.Word2Vec.load("modelo_entrenado.txt")
33  print("Modelo cargado.")
34  word_vectors = model.wv'''
35
36
37  #CARGAR EL MODELO PRE ENTRENADO..... Over 1 million 300-dimensional word vectors for Spanish
38
39  '''print("Cargando modelo...")
40  model = KeyedVectors.load_word2vec_format("SBW-vectors-300-min5.txt")
41  print("Modelo cargado.")
42  word_vectors = model.wv'''
43
```

Figura 32. Cargando modelos wor2vec.
Fuente: Elaboración propia.

En segundo lugar, recuperaremos todas las opiniones lematizadas y preprocesadas de cada destino turístico y luego iterar sobre ellas. Si el comentario aún no se ha convertido a vector numérico y no esta insertada a la base de datos se procede a realizar las siguientes acciones.

- Ingresar al diccionario “row” los campos de “reviewid”, “rating”, “date”, “exp_date”, “ariginal_coment”, “original_coment_stopwords”, “parsed_string”.
- Creamos una lista de strings llamada “ar” de los comentarios almacenados anteriormente en el campo “parsed_string”.
- Iteramos en cada palabra de la lista “ar”.
    - Comprobamos que la palabra del comentario esté en el vocabulario del modelo, y si no, la añadimos a una lista de palabras no encontradas.
    - Si la palabra del comentario está en el vocabulario, añadimos el vector numérico correspondiente a la palabra a una lista llamada “array_word2vec”, y se obtendrá un vector numérico de num_palabras x 300 (que es la dimensión de vectores de nuestros 2 modelos).
- Seguidamente comprobamos que lista tenga elementos guardados para calcular la matriz media. de la siguiente forma.

```
array_word2vec = [[1,2,3], [2,1,3], [3,2,1]]

media = [[2, 1.6, 2.3]]
```

- Convertimos los resultados de la media en un formato binario, que es el que acepta mogo db.
- Finalmente ingresamos los datos a una colección de la base de datos.
- Volvemos a repetir el proceso para otro comentario.

```
#Obtenemos los nombres de todos los sitios
nombres = query_names()
rejected_words = []
t = time.time()
#iteramos en los destinos turísticos
for i in nombres:
    row ={}
    row['zone'] = i #Nombre del sitio
    #Obtenemos todos los datos asociados a un destino turístico
    data = get_data(i)
    #iteramos sobre los datos
    for j in data:
        #comprobamos que el comentario no este convertida a un vector numérico
        resultados = mean_word2vec.find({'reviewid': j['reviewid'], 'zone': i})
        #si no esta convertida y no existe en la base de datos
        if (resultados.count()==0):
            array_word2vec = []
            res = []
            media = []
            #ingresamos los datos a un diccionario
            row['_id'] = ObjectId()
            row['reviewid'] = j['reviewid']
            row['rating'] = j['rating']
            row['date'] = j['date']
            row['exp_date'] = j['exp_date']
            row['ariginal_coment'] = j['ariginal_coment']
            row['original_coment_stopwords'] = j['original_coment_stopwords']
            row['parsed_string'] = j['parsed_string']
            s= j['parsed_string']
            # Creamos una lista de strings separando cada palabra de la review
            ar = s.split()
            #Para cada palabra en la lista
            for i in range(len(ar)):
                #si la palabra de la lista esta en el vocabulario del modelo
                if ar[i] in word_vectors.vocab:
                    #agregamos el vector asociado a la palabra a una lista
                    palabra = list(model.wv[ar[i]])
                    #Tenemos ua lista con cada palabra convertida a vector de 300 dimensiones
                    array_word2vec.append(palabra)
                else:
                    #Recolectamos las palabras que no se encuentren en el vocabulario.
                    rejected_words.append(ar[i])
            print("Word2Vec:\n")
            print(array_word2vec)

            if not array_word2vec:
                print("No se ha obtenido media de esta review.")
            else:
                #realizamos el calculo de la media
                media = numpy.mean(array_word2vec, axis=0)
                 #Convertimos array a binario para poder insertarlo.
                binary_media = Binary(pickle.dumps(media, protocol=2), subtype=128)
                #finalmente ingresamos al diccionario
                row['mean_vector'] = binary_media
                # Insertar lso datos del diccionario a una coleccion de la base de datos
                mean_word2vec.insert(row)
                print("Inserted!\n")

        else:
            print("Already inserted!")
        print("_____________________________________________________________________________________")
    print('tiempo del entrenamiento: {} mins'.format(round((time.time() - t) / 60, 2)))

print("Palabras no aceptadas:")
print(rejected_words)
#guardamos en un fichero txt las palabras que no se encontraron
f = open('palabras _rechazadas/reviewpalabras_rechazadas3two.txt', 'w')
for item in rejected_words:
    f.write("%s\n" % item)
con.close()
```

Figura 33. Script para la obtención del vector numérico medio.
Fuente: Elaboración propia.

Como resultado de la aplicación de los scripts anteriores se obtuvo dos colecciones de datos, el primero denominado “mean_word2vecone” que tiene las representaciones vectoriales de los comentarios, que fueron obtenidas aplicando el modelo entrenado por nosotros y el segundo denominado “mean_word2vectwo” que posee las representaciones vectoriales de los comentarios obtenidas de la aplicación del modelo Spanish Billion Words Corpus. Estas colecciones será las que utilizaremos para realizar nuestra clasificación de clases, para realizar la comparación de resultados.

3.7.4.4. Modelado SVM

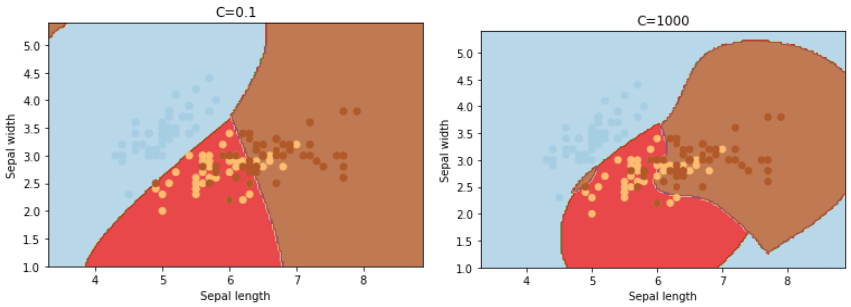
Luego de que se almacenaran las representaciones vectoriales obtenidas por los modelos y los datos relacionados a sus clases como “Positivo” y “Negativo”, utilizamos el algoritmo de clasificación SVM, para predecir la clase del comentario. Este entrenamiento del algoritmo se realizó en la herramienta júpiter notebook porque nos permitía interactuar de manera sencilla con las pruebas y visualizar los resultados, en suma, nos permitió separar los bloques de código en diferentes celdas para ejecutarlos independientemente.

El algoritmo SVM se implementa usando kernels, un kernel toma un espacio de entrada de baja dimensión y lo transforma en un espacio de dimensión superior, en otras palabras, se puede decir que convierte un problema no separable en problemas separables agregándole más dimensión, en este trabajo de investigación buscamos obtener el mejor kernel para nuestra clasificación de datos, por lo que probamos 3 kernels, así buscar obtener el mejor rendimiento del modelo, estos kernels fueron.

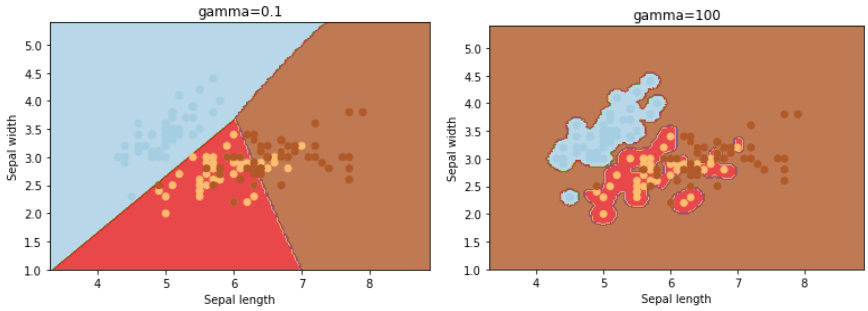
- **Linear Kernel:** Se utiliza cuando los datos que se tienen son linealmente separables.
- **Radial Basis Function Kernel (rbf):** Es uno de los kernels más usados debido a que permite hacer la separación de datos que no son linealmente separables, además permite realizar la separación en datos que tengan muchos espacios dimensionales.
- **Polynomial Kernel:** Un kernel polinomial es una forma más generalizada del kernel lineal, permite identificar la curva de separación de los datos.

Otro de los elementos que son necesarios para el rendimiento del modelo son los hyper parámetros, que permiten seleccionar el mejor tipo de hyperplano que se ajusta a los datos y además que muestre el mejor resultado, estos híper parámetros utilizados son:

- **C:** Es el parámetro de penalización del término de error, que representa una clasificación errónea o un término de error. El término de error de clasificación o error le dice a la optimización SVM cuánto error es soportable. Así es como puede controlar la compensación entre el límite de decisión y el término de clasificación errónea. Un valor menor de C crea un hiperplano de margen pequeño y un valor mayor de C crea un hiperplano de margen mayor. El aumento de los valores de C puede llevar a un ajuste excesivo (overfitting) de los datos de entrenamiento.

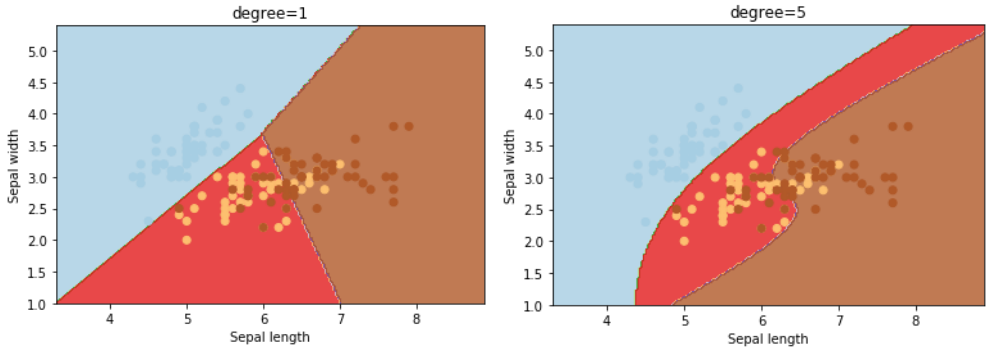


- **Gamma:** Gamma es un parámetro para hiperplanos no lineales, Un valor más bajo de gamma se ajustará ligeramente al conjunto de datos de entrenamiento, mientras que un valor más alto de gamma se ajustará exactamente al conjunto de datos de entrenamiento, lo que causa un ajuste excesivo.



Podemos ver que el aumento de gamma conduce a un ajuste excesivo (overfitting) a medida que el clasificador intenta ajustar perfectamente los datos de entrenamiento.

- **Degree:** Es un parámetro usado cuando el kernel está configurado como 'poli'. Es básicamente el grado del polinomio utilizado para encontrar el hiperplano para dividir los datos.



Por lo cual, para obtener un resultado aceptable tuvimos que probar diferentes valores para cada parámetro así analizar los resultados que se tiene.

3.7.5. Evaluación.

Como dijimos en el apartado anterior, para poder encontrar los mejores resultados tuvimos que realizar varias pruebas de Híper parámetros para ambas colecciones obtenidas a partir de los dos modelos wor2vec (ver apartado 3.5.3.3), a continuación, detallaremos cuales fueron los procedimientos aplicados de ambos modelos el modelo “entrenado y el modelo SWBC”.

3.7.5.1. Evaluación de híper parámetros para el modelo entreado.

Kernel Linear: El parámetro necesario para este kernel es C, y buscaremos la mejor numeración que devuelta buenos resultados, además aplicaremos la técnica de cross validation.

Tabla 11.

Evaluación hiper parámetros C 1-25 para el modelo entrenado, kernel linear.

Valor de C	**Accuracy**
1	0.8309134024550666
2	0.8304661007005706
3	0.8299074414826935
4	0.8308007974284237
5	0.8310241364148563
6	0.8310242608375351
7	0.831136242082547
8	0.8310246346616503
9	0.8308011709744996
10	0.8312480983488012
11	0.8312478492254041
12	0.8311359926811104
13	0.8313595810689798
14	0.8311361173818288
15	0.8312477245246859
16	0.8313597054916585
17	0.8313597054916585
18	0.8311363665052258
19	0.8311362418045075
20	0.8311363665052258
21	0.8306895638316423
22	0.831247973648083
23	0.8304661001444915
24	0.8308012956752178
25	0.8308012956752178

Fuente: Elaboración propia.

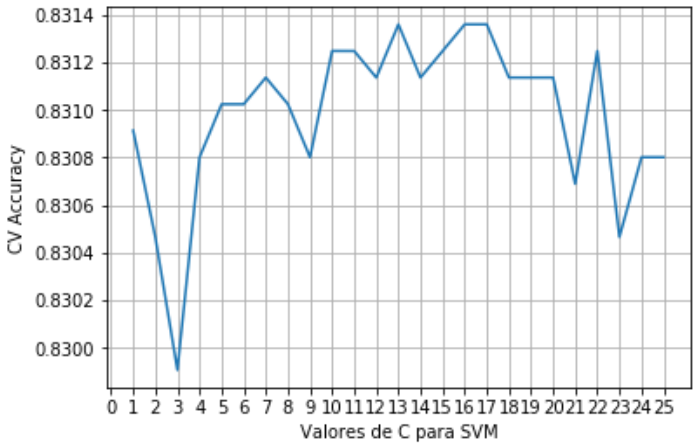


Figura 34. Accurancy hiper parametros C para kernel linear en el modelo entrenado. Fuente: Elaboración propia.

Así pues, profundizando en los valores óptimos de C, ver Anexo C obtuvimos que los valores óptimos son 0.1,0.5,0.9,7.07,7.09,7.10,7.11.

Kernel RBF: el parámetro necesario para este kernel es gamma, y buscaremos la mejor numeración que devuelta buenos resultados, aplicaremos la técnica de cross validation.

Tabla 12.

Evaluación híper parámetros gamma 0.001- 100 para el modelo entrenado, kernel rbf.

Valores gamma	Accuracy
0.0001	0.8365161951617122
0.001	0.8365161951617122
0.01	0.8219911801976261
0.1	0.8506853253974185
1	0.8370743558547558
10	0.8365161951617122
100	0.8365161951617122

Fuente: Elaboración propia.

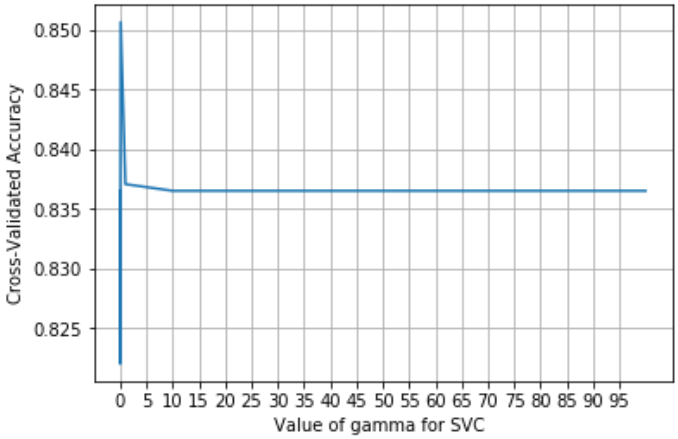


Figura 35. Accurancy hiperparametros gamma para kernel rbf en el modelo entrenado.
Fuente: Elaboración propia.

Así pues, profundizando en los valores óptimos Ver anexo C de gamma obtuvimos que los valores óptimos son 0.01,0.03,0.05,0.15,0.16,0.18.

Kernel poly: el parámetro necesario para este kernel es degree, y buscaremos la mejor numeración que devuelta buenos resultados, aplicaremos la técnica de cross validation.

Tabla 13.

Evaluación hiper parámetros degree 2- 6 para el modelo entrenado, kernel poly.

Degree	Accuracy
2	0.8365161951617122
3	0.8365161951617122
4	0.8365161951617122
5	0.8365161951617122
6	0.8365161951617122

Fuente: Elaboración propia.

Así pues, los valores óptimos degree son 2,3,4 figura 36.

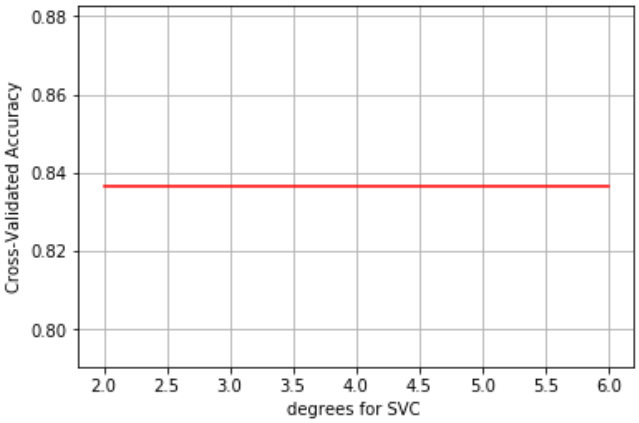


Figura 36. Accurancy hiperparametros degree para kernel poly en el modelo entrenado.
Fuente: Elaboración propia.

3.7.5.1. Evaluación de híper parámetros para el modelo SBWC.

Kernel Linear: investigando valores óptimos de C

Tabla 14.

Evaluación hiperparámetros C 1-25 para el modelo SBWC, kernel Linear.

Valor de C	Accuracy
1	0.8365161951617122
2	0.8365161951617122
3	0.8366279270052877
4	0.8357340722566843
5	0.8342815582902038
6	0.8321586532622709
7	0.8295888208600362
8	0.8281364315942741
9	0.8280245753280198
10	0.8266836685043966
11	0.825119547395059
12	0.8244489072102092
13	0.8251184259227131
14	0.8264588342215028
15	0.8255651041736176
16	0.8267932821039985
17	0.8275747815054348
18	0.8283562814629505
19	0.8286908537681248
20	0.8294724781483194
21	0.8305894227599582
22	0.8304775662156644
23	0.8308123879222753
24	0.8314821557581759
25	0.8311469602274496

Fuente: Elaboración propia.

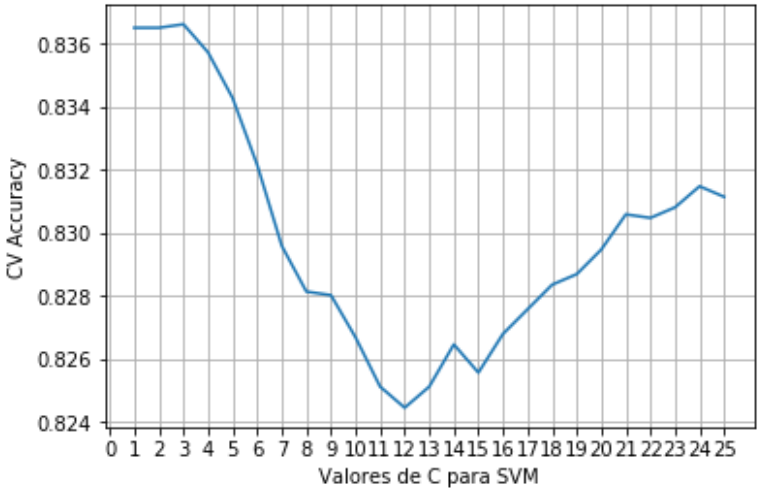


Figura 37. Accurancy hiperparametros C para kernel linear en el modelo SBWC.
Fuente: Elaboración propia.

Así pues, profundizando en los valores óptimo de C , ver Anexo D obtuvimos que los valores óptimos son 0.1,0.5,0.9,2.7,2.8,2.9.

Kernel RBF: Buscando los mejores parámetros.

Tabla 15.

Evaluación hiperparámetros gamma 0.001- 100 para el modelo SBWC, kernel rbf.

Valores gamma	**Accuracy**
0.0001	0.8365161951617122
0.001	0.8365161951617122
0.01	0.8365161951617122
0.1	0.8365161951617122
1	0.8365161951617122
10	0.8339348978004628
100	0.8477898921907379

Fuente: Elaboración propia.

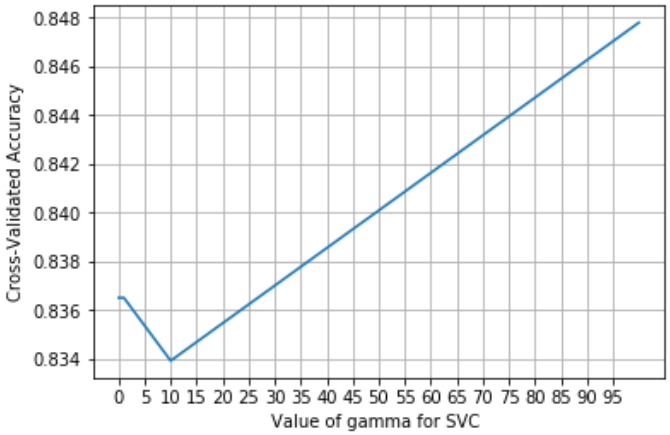


Figura 38. Accurancy hiperparametros gamma para kernel rbf en el modelo SBWC.
Fuente: Elaboración propia.

Así pues, profundizando en los valores óptimos de gamma estos son 0.01,0.03,0.05,1.5,1.6,1.8, ver Anexo D

Kernel poly: buscando mejores parámetros degree

Tabla 16.

Evaluación hiperparámetros degree 2- 6 para el modelo SBWC, kernel poly.

Degree	Accuracy
2	0.8365161951617122
3	0.8365161951617122
4	0.8365161951617122
5	0.8365161951617122
6	0.8365161951617122

Fuente: Elaboración propia.

Así pues, los valores óptimos degree son 2,3,4, Figura 39

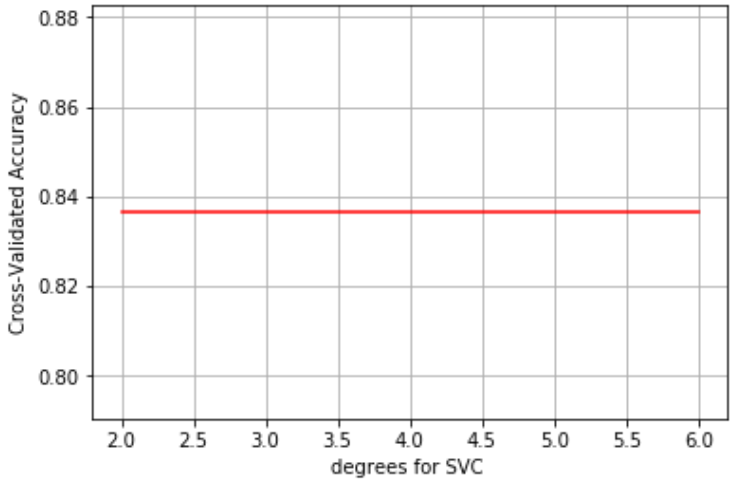


Figura 39. Accurancy hiperparametros degree para kernel poly en el modelo SWBC. Fuente: Elaboración propia.

3.7.5.1. Buscando los mejores parámetros con GridSearchCV.

En trabajos de ML la tarea que produce más pesadillas es estipular valores para los hiper parámetros. Por esto se han implementado bibliotecas, como GridSearchCV de la biblioteca sklearn, para automatizar este proceso y hacer la vida un poco más fácil.

Grid search es el proceso de realizar el ajuste de hiperparámetros para determinar los valores óptimos para un modelo dado. Esto es significativo, ya que el rendimiento de todo el modelo se basa en los valores de hiper parámetros especificados.

Por ello, una vez obtenida los valores óptimos para cada hiper parámetro utilizamos GridSearchCV para combinarlos y encontrar cuales son los mejores parámetros que nos proporcionan mejores resultados. a continuación, detallamos los resultados de la búsqueda.

GridSearchCV Para el modelo entrenado.

En primer lugar, definimos los parámetros óptimos para cada kernel que identificamos anteriormente

```
from sklearn.model_selection import GridSearchCV

modelo_svm= svm.SVC()
# definiendo parametros para cada kernel
parametros = [{'kernel':['linear'],'C':[0.1,0.5,0.9,7.07,7.09,7.10,7.11]},
              {'kernel':['rbf'],'gamma':[0.01,0.03,0.05,0.15,0.16,0.18],'C':[0.1,0.5,0.9,7.07,7.09,7.10,7.11]},
              {'kernel':['poly'],'gamma':[0.01,0.03,0.05,0.15,0.16,0.18],'C':[0.1,0.5,0.9,7.07,7.09,7.10,7.11],
               'degree': [2,3,4] }
                  ]
```

Figura 40. Definiendo hiperparametros con GridSearchCv para el modelo entrenando.
Fuente: Elaboración propia.

Seguidamente ingresamos los parámetros al modelo e iniciamos el entrenamiento. Además, el tiempo de entrenamiento dependerá del número de parámetros y el valor de división de la validación cruzada. La visualización una vez realizada la tarea lo puede ver en el apartado Anexo E.

```
import time

#importando time para medir el tiempo de procesamiento
t = time.time()
#defniendo los parametros que medira gridsearch aplicando validacion cruzada k = 10
svm_modelo= GridSearchCV(modelo_svm, parametros,cv=10,scoring='accuracy')
#entrenamos el modelo
svm_modelo.fit(x_train_sample_vector,y_train_sample_rating)

print("Mejor parametro")
print()
print(svm_modelo.best_params_)
print()
print("valores de accuracy  de cada parametro")
print()
me = svm_modelo.cv_results_['mean_test_score']
st = svm_modelo.cv_results_['std_test_score']
for me, st, param in zip(me, st, svm_modelo.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"% (me, st * 2, param))
print()
print('Tiempo de procesamiennto: {} mins'.format(round((time.time() - t) / 60, 2)))
```

Figura 41. Búsqueda de resultados con GridSearchCV para el modelo entrenado.
Fuente: Elaboración propia.

Finalmente obtendremos los valores óptimos para el modelo.

```
print('Mejores parametros: ',svm_modelo.best_params_)
print('Mejor score acurrancy:',svm_modelo.best_score_)
print('Mejor parametro C:',svm_modelo.best_estimator_.C)
print('Mejor  Kernel:',svm_modelo.best_estimator_.kernel)
print('Mejor parametro  Gamma:',svm_modelo.best_estimator_.gamma)
print('Mejor parametro  degree:',svm_modelo.best_estimator_.degree)
```

```
Mejores parametros:  {'C': 0.9, 'gamma': 0.05, 'kernel': 'rbf'}
Mejor score acurrancy: 0.8639028475711893
Mejor parametro C: 0.9
Mejor  Kernel: rbf
Mejor parametro  Gamma: 0.05
Mejor parametro  degree: 3
```

Figura 42. Resultados de la búsqueda GridSearchCV para el modelo entrenado.
Fuente: Elaboración propia.

GridSearchCV Para el modelo SBWC.

Definiendo parámetros para cada Kernel

```
from sklearn.model_selection import GridSearchCV

modelo_svm= svm.SVC()
# definiendo parametros para cada kernel
parametros = [{'kernel':['linear'],'C':[0.1,0.5,0.9,2.7,2.8,2.9,7.07,7.09,7.10,7.11]},
              {'kernel':['rbf'],'gamma':[0.01,0.03,0.05,0.15,0.16,0.18,1.5,1.6,1.8],
               'C':[0.1,0.5,0.9,2.7,2.8,2.9,7.07,7.09,7.10,7.11]},
              {'kernel':['poly'],'gamma':[0.01,0.03,0.05,0.15,0.16,0.18,1.5,1.6,1.8],
               'C':[0.1,0.5,0.9,2.7,2.8,2.9,7.07,7.09,7.10,7.11],'degree': [2,3,4] }
                  ]
```

Figura 43. Definiendo hiperparámetros para el modelo SBWC.
Fuente: Elaboración propia.

Entrenamiento y búsqueda de parámetros

```
import time

#importando time para medir el tiempo de procesamiento
t = time.time()
#defniendo los parametros que medira gridsearch aplicando validacion cruzada k = 10
svm_modelo= GridSearchCV(modelo_svm, parametros,cv=10,scoring='accuracy')
#entrenamos el modelo
svm_modelo.fit(x_train_sample_vector,y_train_sample_rating)

print("Mejor parametro")
print()
print(svm_modelo.best_params_)
print()
print("valores de accuracy  de cada parametro")
print()
me = svm_modelo.cv_results_['mean_test_score']
st = svm_modelo.cv_results_['std_test_score']
for me, st, param in zip(me, st, svm_modelo.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"% (me, st * 2, param))
print()
print('Tiempo de procesamiennto: {} mins'.format(round((time.time() - t) / 60, 2)))
```

Figura 44. Búsqueda de resultados con Grid search para el modelo SBWC.
Fuente: Elaboración propia.

valores óptimos para el modelo.

```
print('Mejores parametros: ',svm_modelo.best_params_)
print('Mejor score acurrancy:',svm_modelo.best_score_)
print('Mejor parametro C:',svm_modelo.best_estimator_.C)
print('Mejor  Kernel:',svm_modelo.best_estimator_.kernel)
print('Mejor parametro  Gamma:',svm_modelo.best_estimator_.gamma)
print('Mejor parametro  degree:',svm_modelo.best_estimator_.degree)
```

```
Mejores parametros:  {'C': 7.09, 'gamma': 1.8, 'kernel': 'rbf'}
Mejor score acurrancy: 0.8591568955890564
Mejor parametro C: 7.09
Mejor  Kernel: rbf
Mejor parametro  Gamma: 1.8
Mejor parametro  degree: 3
```

Figura 45. Resultados de la búsqueda Grid search para el modelo SWBC.
Fuente: Elaboración propia.

CAPÍTULO IV. Resultados y Discusión

En este capítulo describiremos cuales fueron los resultados obtenidos y comprobaremos el grado en que los modelos definitivos responden a nuestros objetivos.

Así que se procederá evaluar los resultados de clasificación para los mejores parámetros encontrados anteriormente en cada Modelo word2vec.

4.1. Resultados de Performance.

Las métricas de clasificación se muestran a continuación

Tabla 17.

Performance del clasificador en ambos modelos.

	modelo entrenado			modelo SBWC			
	Precisión	**Recall**	**F1-score**	**Precisión**	**Recall**	**F1-score**	**support**
Negativos	0.73	0.08	0.14	0.54	0.08	0.14	338
Positivos	0.82	0.99	0.90	0.82	0.98	0.90	1453
Accuracy	0.8207705192629816			0.8135120044667783			
			Total				1791

Fuente: Elaboración propia.

Podemos verificar que la exactitud (Accuracy) que representa el porcentaje de predicciones que fueron correctas en la clasificación para ambos modelos word2vec, es de 82% y 81% respectivamente, esto muestra que nuestro modelo entrenado posee mejor resultado que el modelo SBWC, sin embargo, es necesario precisar que, a diferencia de modelo SBWC, nuestro modelo se entrenó con datos pre procesados como la lematización, normalización, etc. Esto hace que sea más adaptable al conjunto de clasificación y obtenga mejores resultados.

Por otra parte, en cuanto a las medidas de Precisión (Precicion) y Exhaustividad (Recall) podemos ver que, para la clase negativa en el modelo entrenado representan un 73% y 0.08% respectivamente, lo que quiere decir “que el 73% de predicciones fueron correctas” y “el 0.08% de casos negativos fueron detectados”, para la clase positiva se obtiene que 82% de las predicciones fueron correctas y 99% de los casos fueron detectados. En cuanto al modelo

SBWC podemos ver que obtenemos un 54% de predicciones correctas como negativas frente a 82% de predicciones correctas como positivas, además se tiene que solo el 0.08% de los casos negativos fueron detectados, mientras que el 98% de los casos positivos fueron detectados.

Algo que llama la atención en los resultados de exhaustividad de ambas comparaciones es el bajo porcentaje que representan, específicamente en la clase "Negativos", esto se debe a la poca cantidad de comentarios que se tienen en esta clase frente a la clase "Positivos" que es alta, y como consecuencia se genera un "desequilibrio de clases".

Para evitar el desequilibro de clases y ver la verdadera exactitud de los modelos, aplicamos las técnicas de cross validation y los resultados son los siguientes. En la tabla podemos ver que para el modelo entrenado la exactitud de la clasificación es de 84% mientas que en el modelo SBWC la exactitud se aproxima al 83%.

Tabla 18.

Exactitud (accuracy) real del clasificador en ambos modelos

	modelo entrenado	**modelo SBWC**
K-folds = 10	exactitud de cada fold	exactitud de cada fold
1	0.87	0.86
2	0.86	0.85
3	0.86	0.86
4	0.87	0.83
5	0.86	0.85
6	0.86	0.85
7	0.85	0.85
8	0.85	0.84
9	0.84	0.84
10	0.69	0.68
Exactitud real	0.84	0.83

Fuente: Elaboración propia.

Para finalizar podemos decir que nuestro modelo presenta resultados medianamente superiores, sin embargo, insisto, este resultado se debe al pre procesamiento del corpus de entrenamiento word2vec.

4.2. Resultados de clasificación de Polaridad.

Acerca de los resultados de clasificación de polaridad, tenemos que en el modelo entrenado del set de “ratings esperados” que son 1791 la cantidad de 1470 coinciden con la predicción y en el modelo SWBC del set de “ratings esperados” que son 1791 la cantidad de 1457 coinciden con la predicción. Ver figuras 46, 47

```
#realizamos un recuento de los valores reales que fueron clasificadoscorrectamente

#print("Valoraciones esperadas:")
#print(y_test_sample_expected_ratings)
#print("Valoraciones obtenidas con Scikit:")
#print(predicted_ratings)
equal = 0
for x,y in zip(y_test_sample_expected_ratings, predicted_ratings):
    if x == y:
        equal += 1
print("Equal ratings: ", equal,"de" ,len(y_test_sample_expected_ratings))
```

```
Equal ratings:  1470 de 1791
```

*Figura 46. C*antidad de resultados correctos modelo entrenado.
Fuente: Elaboración propia.

```
#realizamos un recuento de los valores reales que fueron clasificadoscorrectamente

#print("Valoraciones esperadas:")
#print(y_test_sample_expected_ratings)
#print("Valoraciones obtenidas con Scikit:")
#print(predicted_ratings)
equal = 0
for x,y in zip(y_test_sample_expected_ratings, predicted_ratings):
    if x == y:
        equal += 1
print("Equal ratings: ", equal,"de" ,len(y_test_sample_expected_ratings))
```

```
Equal ratings:  1457 de 1791
```

*Figura 47. C*antidad de resultados correctos modelo SWBC.
Fuente: Elaboración propia.

Para ver de más cerca cuales son estos resultados que se clasificaron como correctas y resultados incorrectos haremos uso de la matriz de confusión ver figura 48, 49.

Matriz de confusión modelo wor2vec entrenado: En la siguiente matriz podemos detallar cual fue el comportamiento de la clasificación de polaridad.

- La cantidad de documentos clasificados como positivos que eran efectivamente positivos fue de 1443 (TP)
- La cantidad de documentos clasificados como positivos que eran negativos fue de 311 (FP)
- La cantidad de documentos clasificados como negativos que eran positivos fue de 10 (FN)
- La cantidad de documentos clasificados como negativos que eran efectivamente negativos fue de 27 (TN)

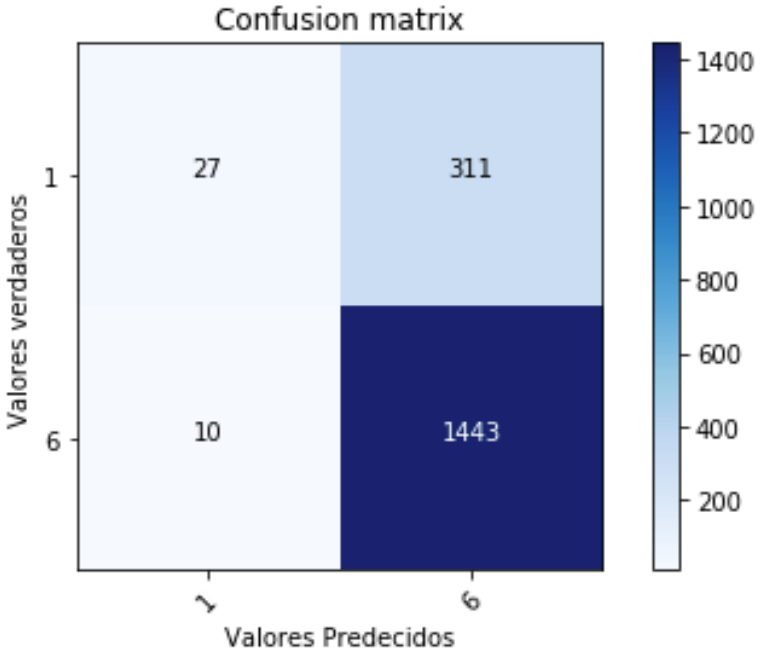


Figura 48. Matriz de confusión modelo entrenado.
Fuente: Elaboración propia.

Matriz de confusión modelo SWBC: En la siguiente matriz podemos detallar cual fue el comportamiento de la clasificación de polaridad.

- La cantidad de documentos clasificados como positivos que eran efectivamente positivos fue de 1429 (TP)
- La cantidad de documentos clasificados como positivos que eran negativos fue de 310 (FP)
- La cantidad de documentos clasificados como negativos que eran positivos fue de 24 (FN)
- La cantidad de documentos clasificados como negativos que eran efectivamente negativos fue de 28 (TN)

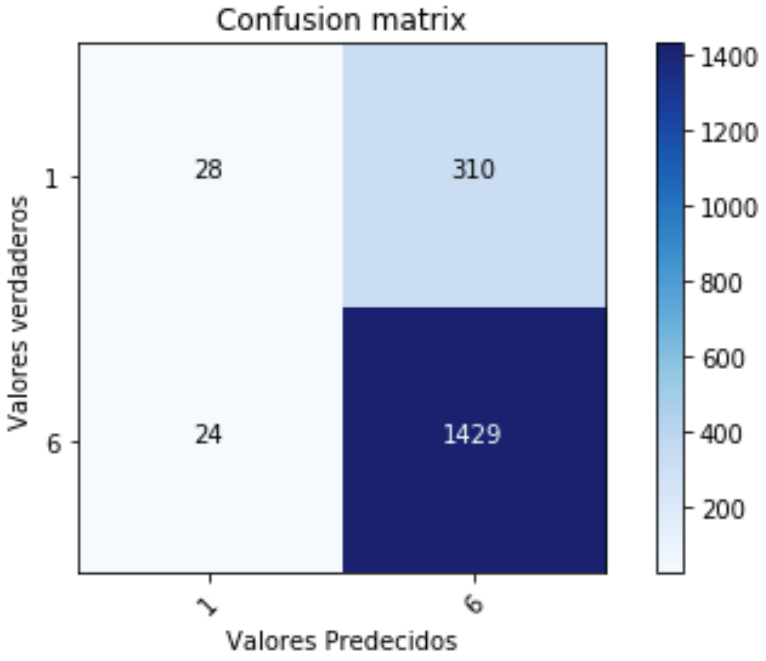


Figura 49. Matriz de confusión modelo SWBC.
Fuente: Elaboración propia.

Finalmente mostramos los gráficos de clasificación de opiniones.

En ambos modelos la clasificación de polaridad tiende a ser mayor en opiniones positivas incurriendo en un menor nuero de errores, sin embargo, en la clase de opiniones negativas por la cantidad de datos disponibles la cantidad de aciertos y errores depende del modelo.

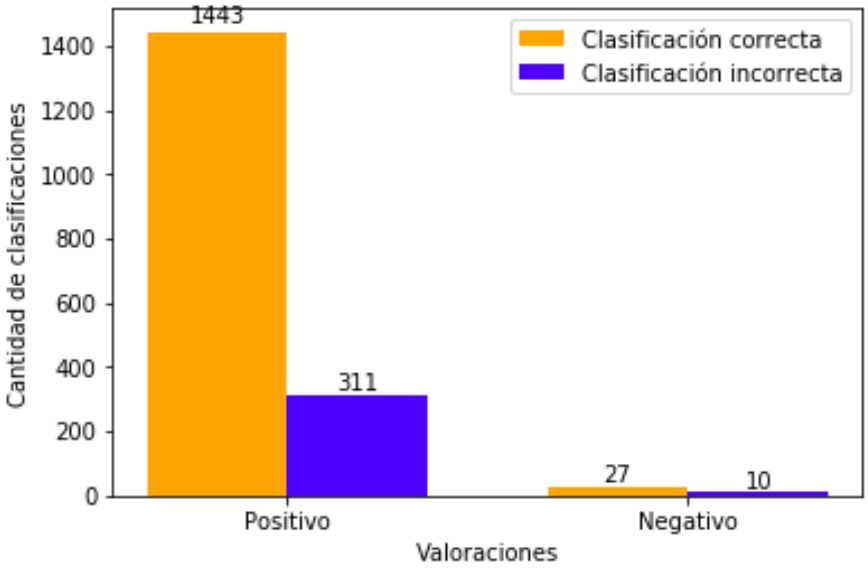


Figura 50. Cantidad de opiniones positivas y negativas clasificadas por el modelo entrenado.
Fuente: Elaboración propia.

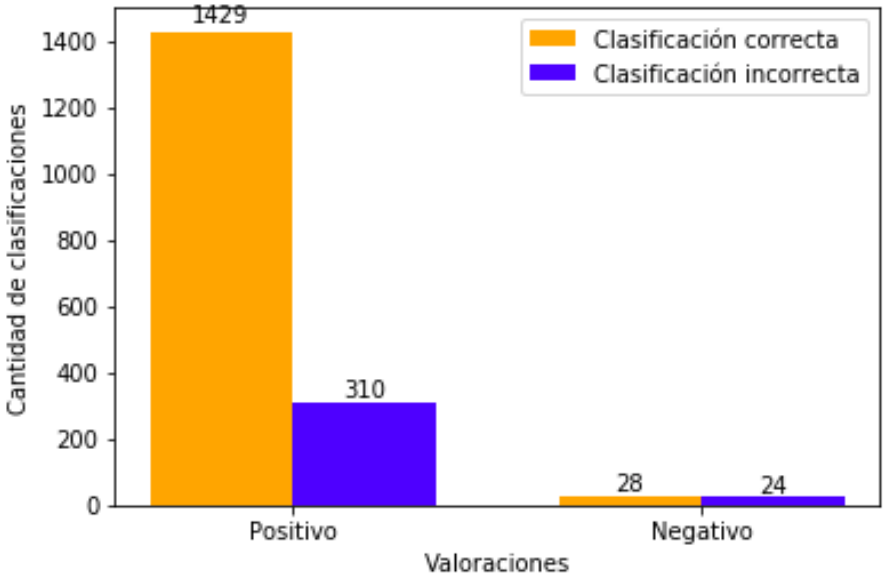


Figura 51. Cantidad de opiniones positivas y negativas clasificadas modelo SWBC.
Fuente: Elaboración propia.

4.3. Discusión

Una de las contribuciones de importancia en este trabajo de investigación es la utilización de minería de opiniones bajo un enfoque supervisado para el análisis de comentarios, en este caso nos ayudamos de comentarios de destinos turísticos de la región de Puno Perú. Aplicamos todos los procedimientos habituales en los distintos procesos de el procesamiento de lenguaje natural, y ayudados de una metodología en este CRISP-DM pudimos realizar cada una de las tareas que se involucran en estas investigaciones, partiendo desde la comprensión del negocio, obtención de los datos que trabajamos, el procesamiento de estos datos, el modelado, la evaluación de cada uno de los modelos involucrados y finalmente realizar el análisis de resultados.

Bajo los resultados de performance obtenidos, se puede apreciar que el modelo wor2vec que entrenamos con comentarios turísticos y que fue utilizado para obtener el vector medio o representación vectorial del comentario, tiene mejores resultados de clasificación con el algoritmo SVM, en comparación del modelo SWBC. Sin embargo, debemos tener en cuenta que estos resultados son más altos y aceptables, debido a que el modelo se adecua a nuestros comentarios que son analizados, es decir, al igual que los comentarios de análisis, el corpus de entrenamiento del modelo también fue pre procesado, mientras que el corpus con el que fue implementado SWBC no tiene el mismo pre procesamiento.

Por otro lado, podemos apreciar que los resultados obtenidos en ambos casos, no difieren en gran manera, sino que se asemejan mucho, en cuanto a la exactitud se refiere, esto hace pensar que, mientras más grande sea el corpus con el que se entrene el modelo Word2vec más efectivo será, sin importar mucho el dominio de estudio. Ya que nuestro modelo word2vec que entrenamos solo tiene no más de 700000 palabras en su vocabulario, mientras que el modelo SWCB pose Billones de palabras en su vocabulario.

Así mismo en cuanto a los resultados de clasificación de comentarios, podemos mencionar que los diferentes destinos de los que se recolectaron los comentarios, tienden a tener una valoración ampliamente positiva frente a la negativa, como se pudo observar en las figuras 50 y 51, esto hace pensar que los turistas que arriban a la región de Puno y específicamente a los destinos de los cuales se tiene los comentarios, están satisfechos con los servicios brindados, aumentando el flujo de turistas a la región , esto se refleja a las estadísticas del arribo de turistas a lo largo de los años a alguno de los destinos estudiados.

Tabla 19.

Arribo de turistas a destinos turísticos año 2016-2017.

Destino Turístico	**Año 2015**	**Año 2016**	**Año 2017**
Lago Titicaca	150 639	149 689	155 483
Isla Flotante los Uros	20 342	29 144	37 815
Isla Taquile	69 194	84 483	105 733
Isla Amantani	33 189	27 643	37 336
Sillustani	89 245	95 008	80 434

Fuente: Adaptado de http://datosturismo.mincetur.gob.pe/appdatosTurismo/Content2.html.

CAPÍTULO V. Conclusiones y recomendaciones

5.1. Conclusiones

Con forme a los procedimientos, los datos y los resultados del trabajo de investigación denominado minería de opiniones basado en aprendizaje supervisado para la evaluación de destinos turísticos de la región de Puno, se llegó a las siguientes conclusiones.

Respecto al objetivo general de Aplicar las técnicas de Minería de Opiniones y aprendizaje supervisado para la evaluación de destinos turísticos de la región Puno, se siguió los procedimientos necesarios en los cuales desarrollamos diferentes procesos de minería de Opiniones, para finalmente obtener cuadros que reflejen la clasificación de opiniones brindadas por los turistas, que en este caso resulto que la mayoría de los comentarios de los destinos turísticos de la región de Puno tienden a ser positivos.

Con respecto a al primer objetivo específico se realizó la extracción de los datos, previa evaluación de las paginas disponibles para realizar las tareas de web scraping y evaluación de los datos necesarios.

Con respecto al segundo objetivo específico se realizó las técnicas de pre procesamiento de información, en las que desarrollamos scripts que permitan limpiar nuestros datos

Con respecto al tercer objetivo específico, realizamos el entrenamiento de modelos word2vec, así como también hacer la representación vectorial de los comentarios extraídos.

Con respecto al cuarto objetivo específico, se realizó la preparación de los scripts necesarios para el entrenamiento y pruebas del algoritmo de clasificación supervisada SVM.

Finalmente se realizó la evaluación de resultado de la clasificación de cada uno de los modelos word2vec comparados.

5.2. Recomendaciones

- Se recomienda realizar investigaciones futuras con un nivel más avanzado, se propone realizar un estudio a nivel de aspecto, con la utilización de wordembeddings.

- También se propone realizar un estudio aplicando diferentes tipos de clasificadores diferentes a SVM además de aplicar diferentes procesos de pre procesamiento de texto.

- Se recomienda hacer uso del modelo word2vec SWBC ya que produce buenos resultados además de poseer una gran cantidad de vocabulario.

- Se recomienda realizar los entrenamientos de los modelos wor2vec con la mayor cantidad posible de palabras, además de realizar varias pruebas de rendimiento del modelo.

- SVM es un buen algoritmo para tareas de clasificación de texto.

REFERENCIAS

Abbasi, A., Chen, H., & Salem, A. (2008). Sentiment Analysis in Multiple Languages: Feature Selection for Opinion Classification in Web Forums AHMED. *ACM Transactions on Information Systems*, *26*(3), 1–34. https://doi.org/10.1145/1361684.1361685

Afzaal, M., & Usman, M. (2016). A novel framework for aspect-based opinion classification for tourist places. *The 10th International Conference on Digital Information Management, ICDIM 2015*, (Icdim), 1–9. https://doi.org/10.1109/ICDIM.2015.7381850

Alfrjani, R., Osman, T., & Cosma, G. (2016). A new approach to ontology-based semantic modelling for opinion mining. *Proceedings - 2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation, UKSim 2016*, 267–272. https://doi.org/10.1109/UKSim.2016.15

Cortez, A., Vega, H., & Pariona, J. (2009). Procesamiento de lenguaje natural. *Revista de Investigación de Sistemas e Informática*, *6*(2), 45–54.

Ding, Y., Li, B., Zhao, Y., & Cheng, C. (2017). Scoring tourist attractions based on sentiment lexicon. *Proceedings of 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference, IAEAC 2017*, 1990–1993. https://doi.org/10.1109/IAEAC.2017.8054363

Dubiau, L. (2013). *Procesamiento de Lenguaje Natural en Sistemas de Análisis de Sentimientos*.

García, S. R. (2016). *Minería de textos y análisis de sentimientos en sanidadysalud.com*. Universidad Complutense de Madrid.

Gobierno Regional de Puno. (2011). *Plan Estratégico Regional De Turismo Puno - Pertur 2021* (p. 220). p. 220.

Henriquez, C., & Guzman, J. (2016). *Modelo ontológico para la detección de características de una entidad aplicado al análisis de sentimientos a nivel de aspectos en español* . (Cicic), 215–220.

Kavitha, S. (2017). Opinion Mining on Tourism. *IJIRST –International Journal for Innovative Research in Science & Technology*, *3*(08), 128–131.

Liddy, E. D. (2001). Natural Language Processing. *Encyclopedia of Library and Information Science*, *2nd Ed*, 220.

Lilleberg, J., Zhu, Y., & Zhang, Y. (2015). Support vector machines and Word2vec for text classification with semantic features. *Proceedings of 2015 IEEE 14th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*CC 2015*, 136–140. https://doi.org/10.1109/ICCI-CC.2015.7259377

Liu, B. (2012). Sentiment Analysis and Opinion Mining. *Encyclopedia of Machine Learning and Data Mining*, (May), 1–10. https://doi.org/10.1007/978-1-4899-7502-7_907-1

Machado, F. N. (2018). *Análisis de sentimientos basado en opiniones turísticas*.

Marrese-Taylor, E., Velásquez, J. D., Bravo-Marquez, F., & Matsuo, Y. (2013). Identifying customer preferences about tourism products using an aspect-based opinion mining approach. *Procedia Computer Science*, *22*, 182–191. https://doi.org/10.1016/j.procs.2013.09.094

Martínez, I. P. (2015). *Minería de Opiniones Basada en Características Guiada por Ontologías.*

Miranda, C. (2017). *Un modelo integrado de técnicas de aprendizaje de máquinas no supervisadas y ontologías para la detección automática de sentimientos desde una estructura gramatical simple en español*.

Miranda, C. H., & Guzmán, J. (2016). A review of Sentiment Analysis in Spanish. *Tecciencia*, *12*(22), 35–48. https://doi.org/10.18180/tecciencia.2017.22.5

Molinar, C. (2012). *EL PROCESO DE LEALTAD DEL CONSUMIDOR HACIA LOS DESTINOS DE SOL Y PLAYA. UN ANÁLISIS EMPÍRICO DE LOS TURISTAS QUE VISITAN LLORET DE MAR Y CANCÚN*.

Molinar, C., Espinoza, P., & Llamas, I. (2017). EVALUACIÓN DE DESTINOS TURÍSTICOS MEDIANTE LA TECNOLOGÍA DE LA CIENCIA DE DATOS. *Centro de Investigaciones y Estudios Turísticos (Buenos Aires, Argentina)*, *26*(2), 286–305.

Olivares, C. A. (2016). *Revisión Sistemática Sobre La Aplicación De De Ontologías De Dominio En El Análisis De Sentimientos*.

Padró, L. (2011). *Analizadores Multilingues en FreeLing*. *3*, 13–20.

Pang, B., & Lee, and L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, *2006*, 1–135. https://doi.org/10.1561/1500000001

Peñalver, I., Valencia, R., & García, F. (2011). Minería de opiniones basada en características guiada por ontologías. *Procesamiento de Lenguaje Natural*, *46*, 91–98.

Piryani, R., Madhavi, D., & Singh, V. K. (2017). Analytical mapping of opinion mining and sentiment analysis research during 2000–2015. *Information Processing and Management*, *53*(1), 122–150. https://doi.org/10.1016/j.ipm.2016.07.001

Prameswari, P., Surjandari, I., & Laoh, E. (2017). Opinion mining from online reviews in Bali tourist area. *Proceeding - 2017 3rd International Conference on Science in Information Technology: Theory and Application of IT for Education, Industry and Society in Big Data Era, ICSITech 2017*, *2018-Janua*, 226–230. https://doi.org/10.1109/ICSITech.2017.8257115

PROMPERU. (2017). *Vacasionista extranjero que visito al Peru*.

Rosso, P., Locoro, A., Mascardi, V., & Balaguer, E. V. (2010). *Análisis de opiniones con ontologías*. *41*, 29–37.

Sande, J. C. S. (2018). *Análisis de sentimientos en Twitter*.

Sangeetha. (2017). Aspects based opinion mining from online reviews for product recommendation. *ICCIDS 2017 - International Conference on Computational Intelligence in Data Science, Proceedings*, *2018-Janua*, 1–6. https://doi.org/10.1109/ICCIDS.2017.8272657

SECO. (2014). *Conceptos básicos para la gestión de destinos turísticos*. 31.

Serna, A., Marchiori, E., Gerrikagoitia, J. K., Alzua, A., & Cantoni, L. (2015). An Auto-Coding Process for Testing the Cognitive-Affective and Conative Model of Destination Image. *Information and Communication Technologies in Tourism 2015: Proceedings of the International Conference in Lugano*, (FEBRUARY), 111–123. https://doi.org/10.1007/978-3-319-14343-9_9

Shein, K. P. P., & Nyunt, T. T. S. (2010). Sentiment Classification Based on Ontology and SVM Classifier. *2010 Second International Conference on Communication Software and Networks*, 169–172. https://doi.org/10.1109/ICCSN.2010.35

Solis, S. R. Q. (2018). *ANÁLISIS DE INTERESES TURÍSTICOS UTILIZANDO REDES SOCIALES EN UN ENTORNO DE BIG DATA*. ESCUELA POLITÉCNICA NACIONAL.

Sosa, E. (1997). Procesamiento del lenguaje natural: revisión del estado actual, bases teóricas y aplicaciones (Parte I). Retrieved from El profecional de la información

website: http://www.elprofesionaldelainformacion.com/contenidos/1997/enero/procesamiento_del_lenguaje_natural_revisin_del_estado_actual_bases_tericas_y_aplicaciones_parte_i.html

Villanueva, J. D. B. (2007). Competitividad Y Percepción Del Servicio Turístico Peruano Estudio De Investigación 2005. *Gestión En El Tercer Milenio, 10*(19), 45–52.

Zárate, M. del P. S. (2017). *Detección de Patrones Psicolingüísticos para el Análisis de Lenguaje Subjetivo en Español*. https://doi.org/10.13140/RG.2.1.2171.2482

ANEXOS

Anexo A. Instalación de freeling

La instalación en un principio es bastante sencilla, pues los desarrolladores distribuyen paquetes DEB para diferentes distribuciones de las recientes versiones para Windows y MacOS. Sin embargo, la versión del sistema operativo que se utilizó en esta tesis es Ubuntu 18.04 (bionic), por lo que no pudimos instalar el paquete deb, asi que optamos por la instalación desde un paquete zip, disponible en la página oficial. Pero al momento de la instalación resultaba en el problema de que esta herramienta tiene como dependencias algunas de las librerías libboost en las versiones 1.58, que son sólo proporcionadas en los repositorios de xenial. La solución fue añadir el repositorio de xenial a las listas de APT para que aparecieran dichas versiones para los paquetes. A continuación, detallamos los pasos de instalación.

PASO 1. Instalación de requerimientos del sistema.

- Lo primero a realizar es agregar el repositorio xenial a Ubuntu 18.

  cd etc/apt

- Abrir el archivo de repositorios.

  vim sources.list

- Añadir los repositorios xenial.

  deb http://es.archive.ubuntu.com/ubuntu xenial main
  #deb-src http:/es.archive.ubuntu.com/ubuntu xenial main

- Instalar los repositorios.

  sudo apt install libboost-filesystem1.58.0

- Ejecutar el siguiente comando**.**

  sudo apt-get install build-essential

- Instalación de Swig (a compiler that makes it easy to integrate C and C++ code)

  sudo apt-get install swig

- instalar cmake versiones iguales o superiores a 3.8, el comando sudo apt-get install cmake intalara cmake v3.5, para obtener la ultima versión cmake debemos descargar la ultima versión en el siguiente enlace (https://cmake.org/download/)

  mkdir /opt/cmake
  sudo sh cmake-3.13.4-Linux-x86_64.sh --prefix=/opt/cmake --skip-license
  ln -s /opt/cmake/bin/cmake /usr/local/bin/cmake
  cmake –versión

- Instalar boost 1.58 en ubuntu 18 download and extract with

  tar -xf boost_1_62_0.tar.gz
  cd boost_1_62_0

- Obtener librerias:

  sudo apt-get update && sudo apt-get install build-essential g++ python-dev autotools-dev libicu-dev build-essential libbz2-dev libboost-all-dev
  build with ./bootstrap.sh --prefix=/usr/local
  install with sudo ./b2 install

- Installation of Boost (C++ Libraries development files)

  sudo apt-get install libboost-all-dev

- Installation of Boost (C++ Libraries development files)

  sudo apt-get install zlib1g-dev

- Instalar python 3 dev

  sudo apt-get install python3-dev

PASO 2. Instalación de requerimientos del sistema.

- Descargar freeling tar.gz y descomprimir, seguidamente ingresar a la carpeta de freeling y ejecutar como administrador los siguientes comandos.

  mkdir build
  cd build
  cmake ..
  make install

- Como la documentación lo indica, se puede cambiar la ruta donde se instalara freeling, utilizando el comando , cmake , pero si no desea cambiar la ruta de instalación, los comando de arriba lo realizara por usted, instalándolo en la ruta por defecto /usr/local. Si tiene problemas de instalación revise este enlace, además

para utilizar una api “ES NECESARIO INDICARLO EN ESTE PUNTO”, nosotros instalamos freeling en una carpeta personalizada y optamos por utilizar API DE PYTHON3 lo realizamos de la siguiente manera.

```
# !!!nosotros instalamos freeling en una carpeta personalizada y obtamos por  utilizar API DE PYTHON3,
#lo realizamos de la siguiente manera¡¡¡¡¡

# 1. Realizamos los mismos primeros pasos ingresando a la carpeta donde se descomprimio

  mkdir build
  cd build

  #ruta resultante
  roger@r0g3r:~/Downloads/FreeLing-4.1/build

# 2. Seguidamente ejecutamos cmake añadiendo instrucciones de la ruta personalizada donde se instalara
# ademas de activar la API para python3

  #para asegurar el buen funcionamiento dimos a la carpeta donde se instalara dimos permisos especiales a la carpeta
  # carpeta "FREELINGG" con el comando chmod 777 FREELINGG

  roger@r0g3r:~/Downloads/FreeLing-4.1/build$ cmake .. -DCMAKE_INSTALL_PREFIX=/home/roger/FREELINGG

  #SEGUIDAMIENTE ACTIVAMOS LA API

  roger@r0g3r:~/Downloads/FreeLing-4.1/build$ cmake .. -DPYTHON3_API=ON

# 3. instalamos freeling

  make install
```

PASO 3. Probando Freeling.

Para testear freeling, usted debe hacer uso de la ruta donde ese instalo freeling, como en este caso lo instalamos una ruta personalizada será /home/roger/FREELINGG sin embargo si usted lo realiza en una ruta por defecto esta será `/usr/local` , la documentación menciona * *FreeLing provides a main program that allows to execute most of its capabilities. However, remember that FreeLing is a library and that many more functionalities are accessible writting your own main program.* **

Basicamente menciona que freling es una librería y podemos ejecutar el programa, realizando el siguiente comando

```
FREELINGDIR/bin/analyze -f en.cfg < myfile.txt
```

¡PERO! ¿dónde se encuentras la carpeta FREELINGDIR ?, en nuestro caso FREELINGDIR será la carpeta FREELINGG sin embargo si usted utiliza la ruta por defecto, no piense que esta carpeta FREELINGDIR se encuentra en **/usr/local**, sino que esta misma

ruta se comporta como la carpeta FREELINGDIR, es decir usted solo tendrá que ingresar a usr/local para ejecutar el test.

Para ejecutar el test creamos un archivo myfile.txt con el texto "El gato come pescado. Pero a Don Jaime no le gustan los gatos." en la carpeta de nuestra preferencia

Con nuestra ruta personalizada ejecutaremos el siguiente comando para testear:

roger@r0g3r:~/FREELINGG$ bin/analyze -f en.cfg < /home/roger/archivesfreeling/myfile.txt

Si usted instalo en una ruta por defecto ejecute ejecutamos el comando.

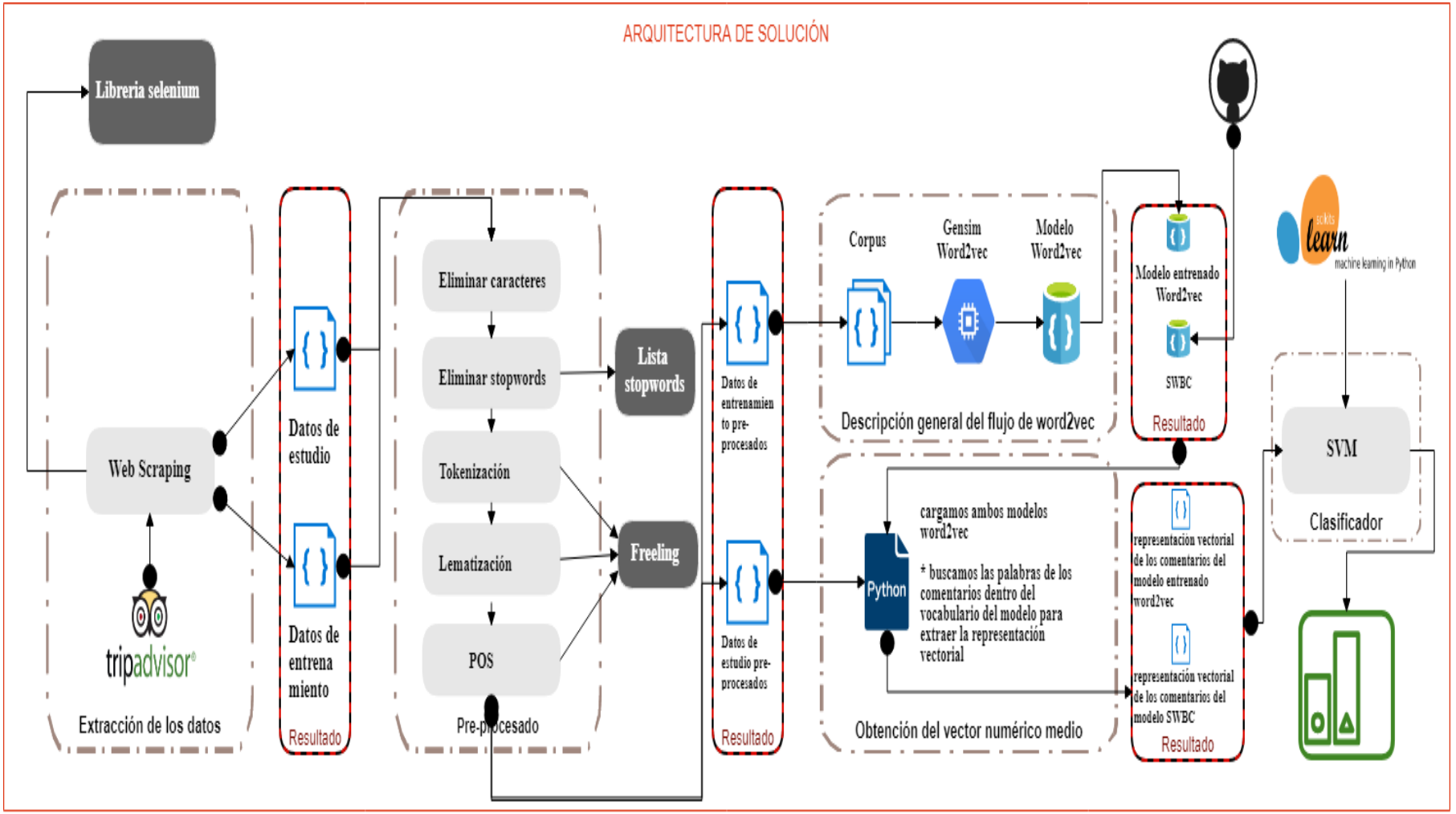
usr/local/bin/analyze -f en.cfg < myfile.txt,

```
roger@r0g3r:/usr/local$ bin/analyze -f es.cfg < /home/roger/analysisarch/mytext.txt
El el DA0MS0 1
gato gato NCMS000 1
come comer VMIP3S0 0.978902
pescado pescado NCMS000 0.822581
. . Fp 1

Pero pero CC 0.999902
a a SP 0.998775
Don_Jaime don_jaime NP00000 1
no no RN 0.999297
le le PP3CSD0 1
gustan gustar VMIP3P0 1
los el DA0MP0 0.992728
gatos gato NCMP000 1
. . Fp 1
```

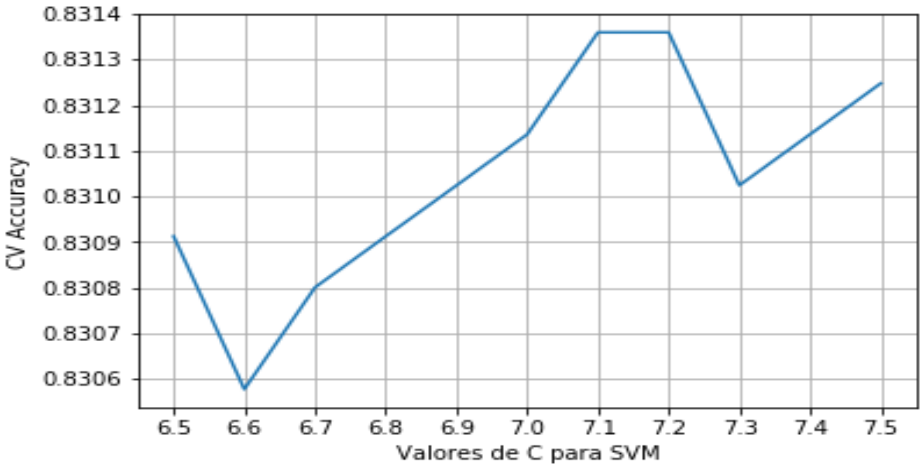
¡EN HORA BUENA YA TIENE INSTALADO FREELING!

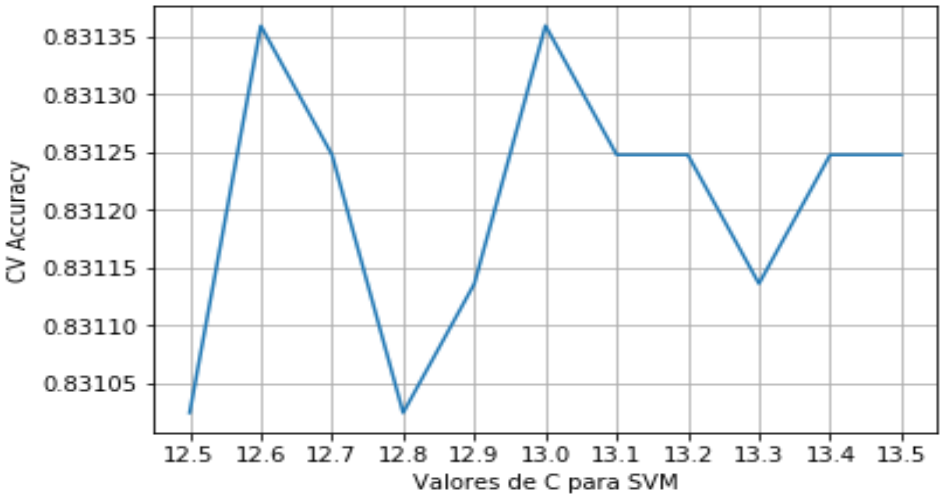
Anexo B. Arquitectura de Solución.

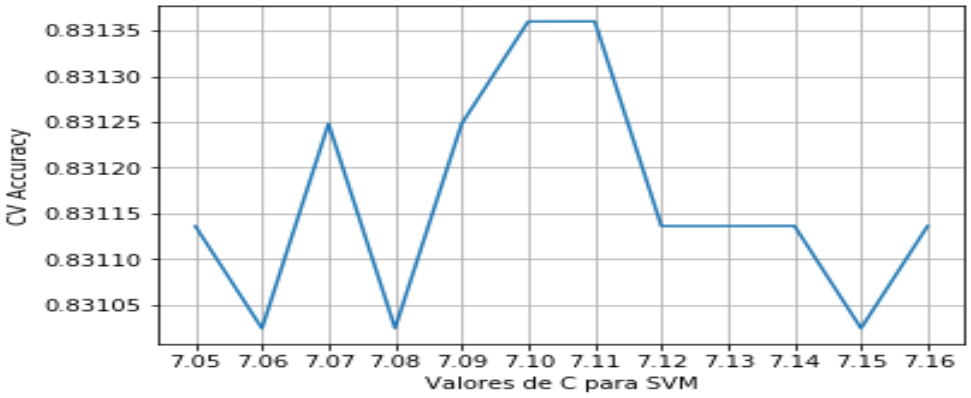


Anexo C. Profundizando en valores de hiperparámetros del modelo entrenado

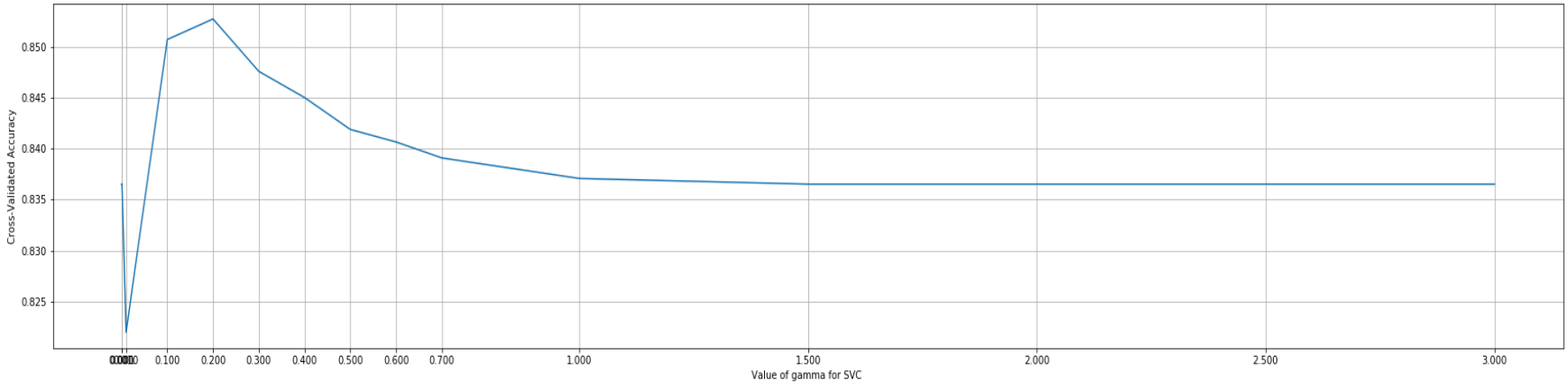
- **Linear**

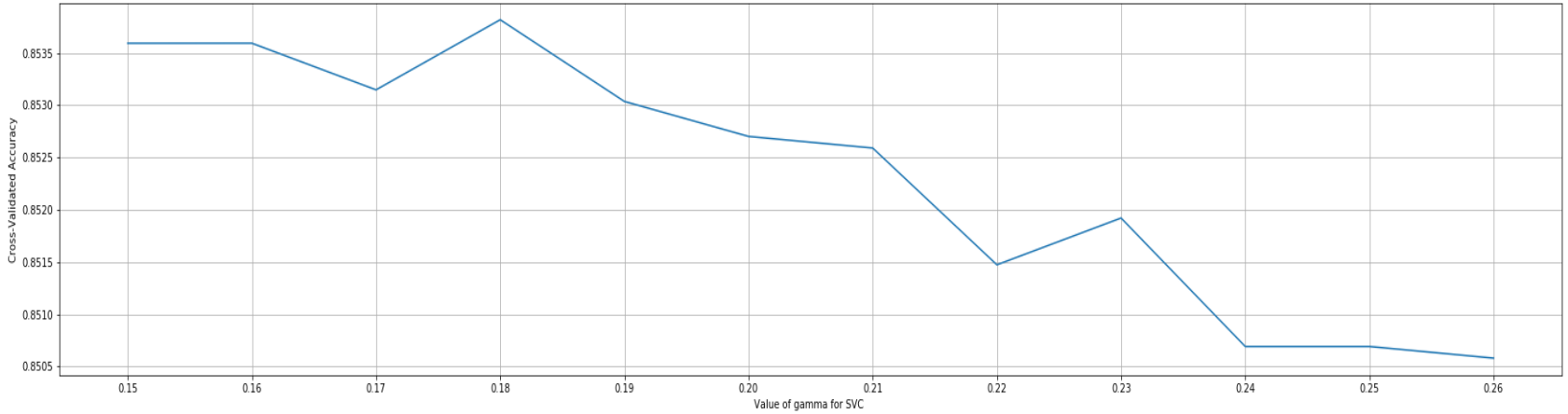






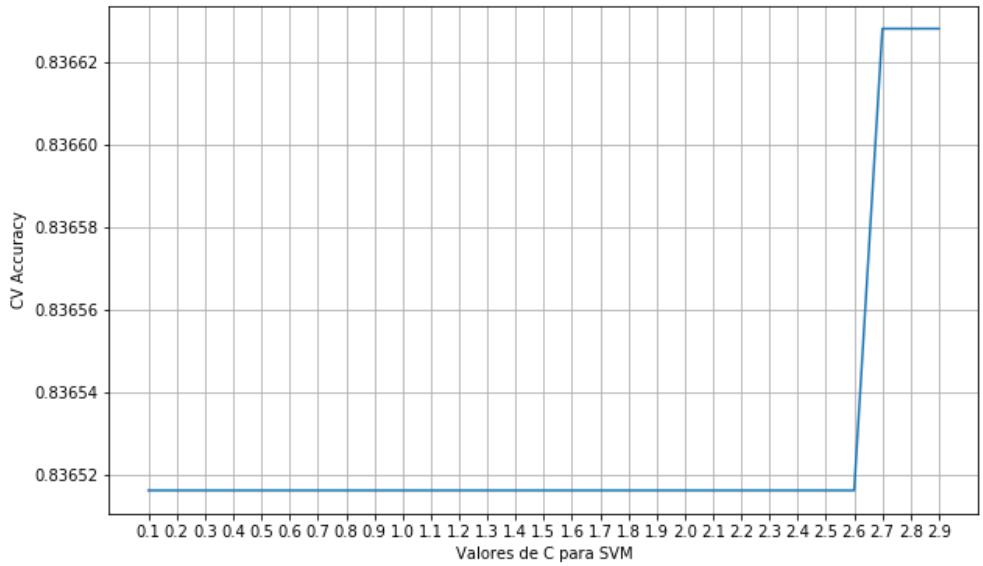
- **rbf.**


Cross-Validated Accuracy
0.850
0.845
0.840
0.835
0.830
0.825
0.100
0.200
0.300
0.400
0.500
0.600
0.700
1.000
1.500
2.000
2.500
3.000
Value of gamma for SVC

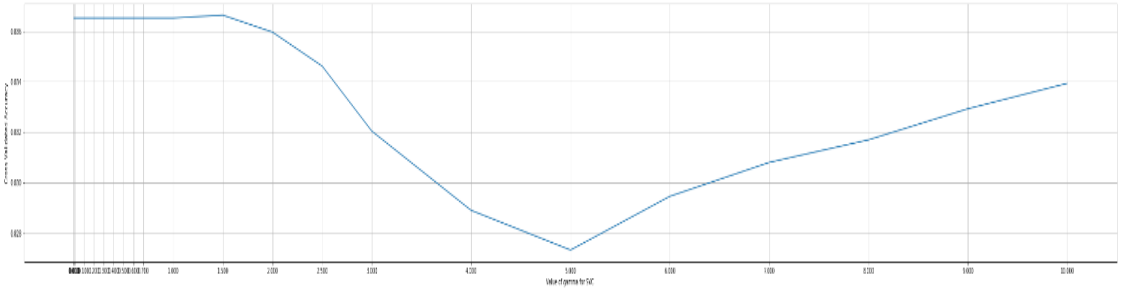

Cross-Validated Accuracy
0.8535
0.8530
0.8525
0.8520
0.8515
0.8510
0.8505
0.15
0.16
0.17
0.18
0.19
0.20
0.21
0.22
0.23
0.24
0.25
0.26
Value of gamma for SVC

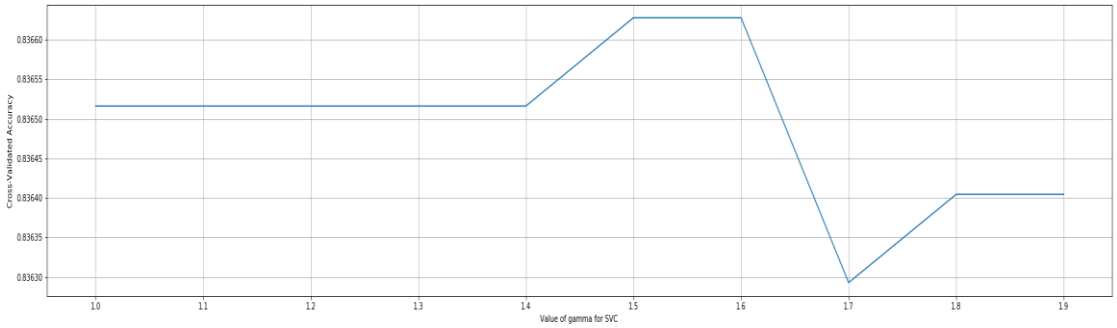
Anexo D. Profundizando en valores de hiperparámetros del modelo SWBC.

- **Linear**



- **bf.**





Anexo E. Resultado GridSearchCv

- **Modelo Entrenado**

Mejor parametro

{'C': 0.9, 'gamma': 0.05, 'kernel': 'rbf'}

valores de accuracy de cada parametro

0.860 (+/-0.013) for {'C': 0.1, 'kernel': 'linear'}
0.859 (+/-0.022) for {'C': 0.5, 'kernel': 'linear'}
0.858 (+/-0.025) for {'C': 0.9, 'kernel': 'linear'}
0.856 (+/-0.024) for {'C': 7.07, 'kernel': 'linear'}
0.856 (+/-0.022) for {'C': 7.09, 'kernel': 'linear'}
0.856 (+/-0.023) for {'C': 7.1, 'kernel': 'linear'}
0.856 (+/-0.023) for {'C': 7.11, 'kernel': 'linear'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.18, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 0.01, 'kernel': 'rbf'}
0.849 (+/-0.014) for {'C': 0.5, 'gamma': 0.03, 'kernel': 'rbf'}
0.857 (+/-0.017) for {'C': 0.5, 'gamma': 0.05, 'kernel': 'rbf'}
0.859 (+/-0.018) for {'C': 0.5, 'gamma': 0.15, 'kernel': 'rbf'}
0.859 (+/-0.015) for {'C': 0.5, 'gamma': 0.16, 'kernel': 'rbf'}
0.856 (+/-0.013) for {'C': 0.5, 'gamma': 0.18, 'kernel': 'rbf'}
0.843 (+/-0.003) for {'C': 0.9, 'gamma': 0.01, 'kernel': 'rbf'}
0.860 (+/-0.016) for {'C': 0.9, 'gamma': 0.03, 'kernel': 'rbf'}
0.864 (+/-0.018) for {'C': 0.9, 'gamma': 0.05, 'kernel': 'rbf'}
0.863 (+/-0.019) for {'C': 0.9, 'gamma': 0.15, 'kernel': 'rbf'}
0.861 (+/-0.017) for {'C': 0.9, 'gamma': 0.16, 'kernel': 'rbf'}
0.861 (+/-0.016) for {'C': 0.9, 'gamma': 0.18, 'kernel': 'rbf'}
0.863 (+/-0.021) for {'C': 7.07, 'gamma': 0.01, 'kernel': 'rbf'}
0.858 (+/-0.025) for {'C': 7.07, 'gamma': 0.03, 'kernel': 'rbf'}
0.858 (+/-0.026) for {'C': 7.07, 'gamma': 0.05, 'kernel': 'rbf'}
0.860 (+/-0.018) for {'C': 7.07, 'gamma': 0.15, 'kernel': 'rbf'}
0.859 (+/-0.019) for {'C': 7.07, 'gamma': 0.16, 'kernel': 'rbf'}
0.859 (+/-0.016) for {'C': 7.07, 'gamma': 0.18, 'kernel': 'rbf'}
0.863 (+/-0.020) for {'C': 7.09, 'gamma': 0.01, 'kernel': 'rbf'}
0.858 (+/-0.025) for {'C': 7.09, 'gamma': 0.03, 'kernel': 'rbf'}
0.858 (+/-0.026) for {'C': 7.09, 'gamma': 0.05, 'kernel': 'rbf'}
0.860 (+/-0.018) for {'C': 7.09, 'gamma': 0.15, 'kernel': 'rbf'}
0.859 (+/-0.019) for {'C': 7.09, 'gamma': 0.16, 'kernel': 'rbf'}
0.859 (+/-0.016) for {'C': 7.09, 'gamma': 0.18, 'kernel': 'rbf'}
0.863 (+/-0.020) for {'C': 7.1, 'gamma': 0.01, 'kernel': 'rbf'}
0.858 (+/-0.025) for {'C': 7.1, 'gamma': 0.03, 'kernel': 'rbf'}
0.858 (+/-0.025) for {'C': 7.1, 'gamma': 0.05, 'kernel': 'rbf'}
0.860 (+/-0.018) for {'C': 7.1, 'gamma': 0.15, 'kernel': 'rbf'}
0.859 (+/-0.019) for {'C': 7.1, 'gamma': 0.16, 'kernel': 'rbf'}
0.859 (+/-0.016) for {'C': 7.1, 'gamma': 0.18, 'kernel': 'rbf'}
0.863 (+/-0.020) for {'C': 7.11, 'gamma': 0.01, 'kernel': 'rbf'}
0.858 (+/-0.024) for {'C': 7.11, 'gamma': 0.03, 'kernel': 'rbf'}
0.858 (+/-0.025) for {'C': 7.11, 'gamma': 0.05, 'kernel': 'rbf'}
0.860 (+/-0.018) for {'C': 7.11, 'gamma': 0.15, 'kernel': 'rbf'}
0.859 (+/-0.019) for {'C': 7.11, 'gamma': 0.16, 'kernel': 'rbf'}
0.859 (+/-0.016) for {'C': 7.11, 'gamma': 0.18, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.857 (+/-0.009) for {'C': 0.1, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.856 (+/-0.010) for {'C': 0.1, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.857 (+/-0.013) for {'C': 0.1, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 0.1, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}

0.850 (+/-0.017) for {'C': 0.1, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.849 (+/-0.018) for {'C': 0.1, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.849 (+/-0.019) for {'C': 0.1, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 0.1, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.026) for {'C': 0.1, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.029) for {'C': 0.1, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.840 (+/-0.028) for {'C': 0.1, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 0.5, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.850 (+/-0.010) for {'C': 0.5, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.853 (+/-0.022) for {'C': 0.5, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.852 (+/-0.021) for {'C': 0.5, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.852 (+/-0.020) for {'C': 0.5, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 0.5, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.849 (+/-0.008) for {'C': 0.5, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.846 (+/-0.028) for {'C': 0.5, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.844 (+/-0.030) for {'C': 0.5, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.840 (+/-0.031) for {'C': 0.5, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 0.5, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.846 (+/-0.009) for {'C': 0.5, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.838 (+/-0.032) for {'C': 0.5, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.836 (+/-0.031) for {'C': 0.5, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.835 (+/-0.029) for {'C': 0.5, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.847 (+/-0.006) for {'C': 0.9, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.857 (+/-0.009) for {'C': 0.9, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.852 (+/-0.018) for {'C': 0.9, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.852 (+/-0.019) for {'C': 0.9, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.848 (+/-0.020) for {'C': 0.9, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.845 (+/-0.004) for {'C': 0.9, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 0.9, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.839 (+/-0.030) for {'C': 0.9, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.838 (+/-0.031) for {'C': 0.9, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.837 (+/-0.033) for {'C': 0.9, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 0.9, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.845 (+/-0.009) for {'C': 0.9, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.837 (+/-0.029) for {'C': 0.9, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.834 (+/-0.025) for {'C': 0.9, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.832 (+/-0.029) for {'C': 0.9, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.846 (+/-0.007) for {'C': 7.07, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.857 (+/-0.021) for {'C': 7.07, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.852 (+/-0.019) for {'C': 7.07, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.826 (+/-0.023) for {'C': 7.07, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.825 (+/-0.023) for {'C': 7.07, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.826 (+/-0.022) for {'C': 7.07, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.852 (+/-0.007) for {'C': 7.07, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.849 (+/-0.023) for {'C': 7.07, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.832 (+/-0.028) for {'C': 7.07, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.832 (+/-0.028) for {'C': 7.07, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.831 (+/-0.030) for {'C': 7.07, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.845 (+/-0.009) for {'C': 7.07, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.025) for {'C': 7.07, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.830 (+/-0.025) for {'C': 7.07, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.830 (+/-0.026) for {'C': 7.07, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}

0.831 (+/-0.026) for {'C': 7.07, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.846 (+/-0.007) for {'C': 7.09, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.857 (+/-0.021) for {'C': 7.09, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.852 (+/-0.019) for {'C': 7.09, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.826 (+/-0.023) for {'C': 7.09, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.825 (+/-0.022) for {'C': 7.09, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.826 (+/-0.023) for {'C': 7.09, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.852 (+/-0.007) for {'C': 7.09, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.849 (+/-0.024) for {'C': 7.09, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.832 (+/-0.028) for {'C': 7.09, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.832 (+/-0.028) for {'C': 7.09, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.831 (+/-0.030) for {'C': 7.09, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.845 (+/-0.009) for {'C': 7.09, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.026) for {'C': 7.09, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.830 (+/-0.025) for {'C': 7.09, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.830 (+/-0.026) for {'C': 7.09, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.831 (+/-0.026) for {'C': 7.09, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.846 (+/-0.007) for {'C': 7.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.857 (+/-0.021) for {'C': 7.1, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.852 (+/-0.018) for {'C': 7.1, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.826 (+/-0.023) for {'C': 7.1, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.825 (+/-0.022) for {'C': 7.1, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.826 (+/-0.022) for {'C': 7.1, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.852 (+/-0.007) for {'C': 7.1, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.849 (+/-0.024) for {'C': 7.1, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.832 (+/-0.028) for {'C': 7.1, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.832 (+/-0.028) for {'C': 7.1, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.831 (+/-0.030) for {'C': 7.1, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.845 (+/-0.009) for {'C': 7.1, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.025) for {'C': 7.1, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.830 (+/-0.025) for {'C': 7.1, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.830 (+/-0.026) for {'C': 7.1, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.831 (+/-0.026) for {'C': 7.1, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.846 (+/-0.008) for {'C': 7.11, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.857 (+/-0.021) for {'C': 7.11, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.852 (+/-0.018) for {'C': 7.11, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.826 (+/-0.023) for {'C': 7.11, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.825 (+/-0.023) for {'C': 7.11, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.826 (+/-0.022) for {'C': 7.11, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.852 (+/-0.007) for {'C': 7.11, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.849 (+/-0.024) for {'C': 7.11, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.832 (+/-0.028) for {'C': 7.11, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.832 (+/-0.028) for {'C': 7.11, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.831 (+/-0.030) for {'C': 7.11, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.845 (+/-0.009) for {'C': 7.11, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.025) for {'C': 7.11, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.830 (+/-0.025) for {'C': 7.11, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.830 (+/-0.026) for {'C': 7.11, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.831 (+/-0.026) for {'C': 7.11, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}

Tiempo de procesamiennto: 425.93 mins

- **Modelo SWBC**

Mejor parametro

{'C': 7.09, 'gamma': 1.8, 'kernel': 'rbf'}

valores de accuracy de cada parametro

0.843 (+/-0.001) for {'C': 0.1, 'kernel': 'linear'}
0.843 (+/-0.001) for {'C': 0.5, 'kernel': 'linear'}
0.843 (+/-0.001) for {'C': 0.9, 'kernel': 'linear'}
0.843 (+/-0.001) for {'C': 2.7, 'kernel': 'linear'}
0.843 (+/-0.001) for {'C': 2.8, 'kernel': 'linear'}
0.843 (+/-0.001) for {'C': 2.9, 'kernel': 'linear'}
0.845 (+/-0.005) for {'C': 7.07, 'kernel': 'linear'}
0.845 (+/-0.005) for {'C': 7.09, 'kernel': 'linear'}
0.845 (+/-0.005) for {'C': 7.1, 'kernel': 'linear'}
0.845 (+/-0.005) for {'C': 7.11, 'kernel': 'linear'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 0.18, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 1.5, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 1.6, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 0.18, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 1.5, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 1.6, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.5, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 0.18, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 1.5, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 1.6, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.9, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.7, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.7, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.7, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.7, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.7, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.7, 'gamma': 0.18, 'kernel': 'rbf'}
0.846 (+/-0.006) for {'C': 2.7, 'gamma': 1.5, 'kernel': 'rbf'}
0.846 (+/-0.006) for {'C': 2.7, 'gamma': 1.6, 'kernel': 'rbf'}
0.848 (+/-0.007) for {'C': 2.7, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.8, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.8, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.8, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.8, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.8, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.8, 'gamma': 0.18, 'kernel': 'rbf'}
0.846 (+/-0.006) for {'C': 2.8, 'gamma': 1.5, 'kernel': 'rbf'}
0.847 (+/-0.007) for {'C': 2.8, 'gamma': 1.6, 'kernel': 'rbf'}
0.849 (+/-0.009) for {'C': 2.8, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.9, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.9, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.9, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.9, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.9, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 2.9, 'gamma': 0.18, 'kernel': 'rbf'}

0.846 (+/-0.006) for {'C': 2.9, 'gamma': 1.5, 'kernel': 'rbf'}
0.847 (+/-0.007) for {'C': 2.9, 'gamma': 1.6, 'kernel': 'rbf'}
0.849 (+/-0.010) for {'C': 2.9, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.07, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.07, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.07, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.07, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.07, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.07, 'gamma': 0.18, 'kernel': 'rbf'}
0.857 (+/-0.018) for {'C': 7.07, 'gamma': 1.5, 'kernel': 'rbf'}
0.857 (+/-0.021) for {'C': 7.07, 'gamma': 1.6, 'kernel': 'rbf'}
0.859 (+/-0.019) for {'C': 7.07, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.09, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.09, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.09, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.09, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.09, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.09, 'gamma': 0.18, 'kernel': 'rbf'}
0.857 (+/-0.018) for {'C': 7.09, 'gamma': 1.5, 'kernel': 'rbf'}
0.857 (+/-0.022) for {'C': 7.09, 'gamma': 1.6, 'kernel': 'rbf'}
0.859 (+/-0.019) for {'C': 7.09, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.1, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.1, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.1, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.1, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.1, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.1, 'gamma': 0.18, 'kernel': 'rbf'}
0.857 (+/-0.018) for {'C': 7.1, 'gamma': 1.5, 'kernel': 'rbf'}
0.857 (+/-0.022) for {'C': 7.1, 'gamma': 1.6, 'kernel': 'rbf'}
0.859 (+/-0.019) for {'C': 7.1, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.11, 'gamma': 0.01, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.11, 'gamma': 0.03, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.11, 'gamma': 0.05, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.11, 'gamma': 0.15, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.11, 'gamma': 0.16, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 7.11, 'gamma': 0.18, 'kernel': 'rbf'}
0.857 (+/-0.018) for {'C': 7.11, 'gamma': 1.5, 'kernel': 'rbf'}
0.857 (+/-0.022) for {'C': 7.11, 'gamma': 1.6, 'kernel': 'rbf'}
0.859 (+/-0.019) for {'C': 7.11, 'gamma': 1.8, 'kernel': 'rbf'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}

0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.1, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.5, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 0.9, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}

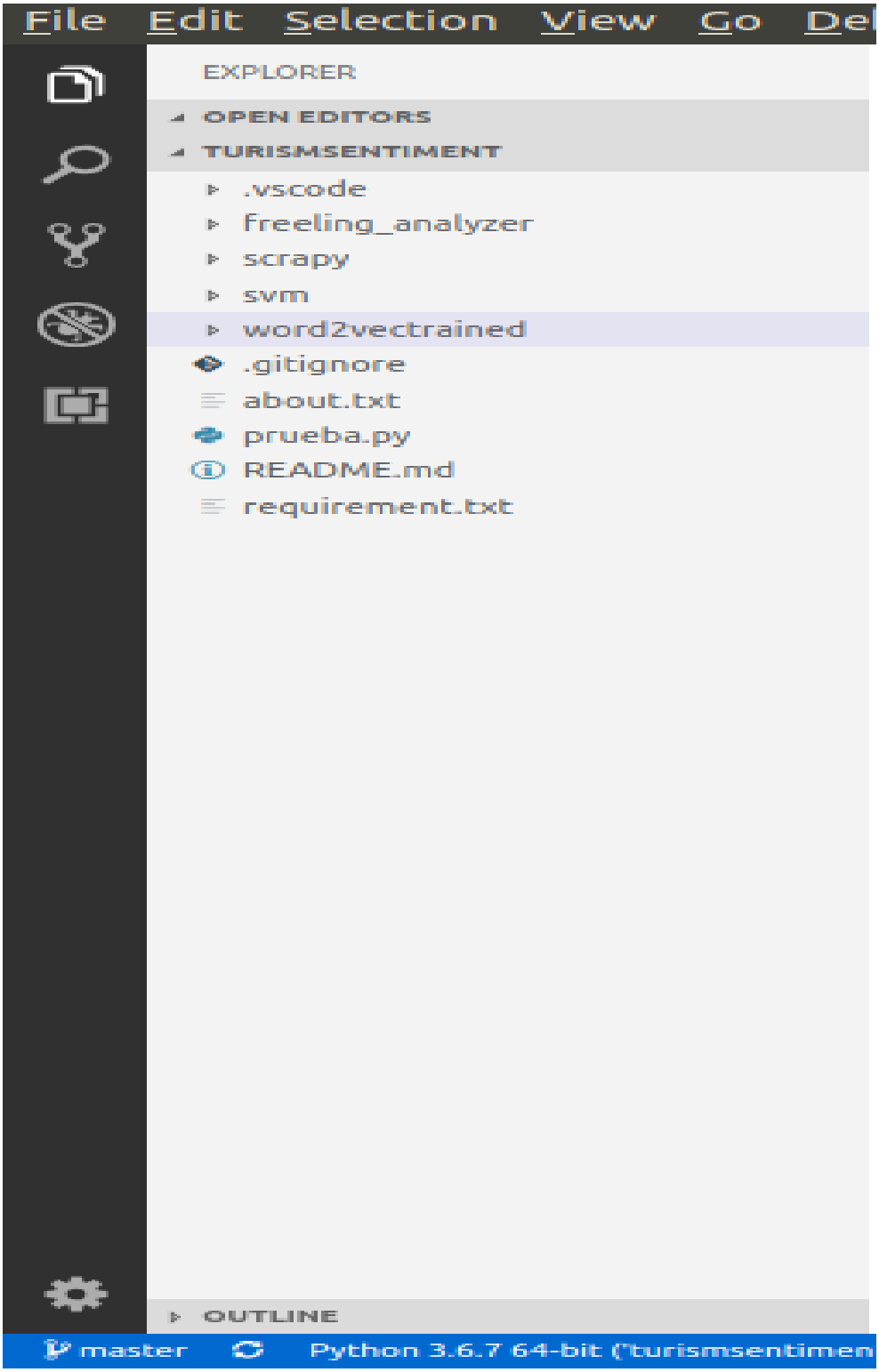
0.843 (+/-0.001) for {'C': 2.7, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 2.7, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 2.7, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 2.7, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.7, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.002) for {'C': 2.7, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 2.8, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 2.8, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 2.8, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 2.8, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.8, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.002) for {'C': 2.8, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.002) for {'C': 2.9, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.844 (+/-0.002) for {'C': 2.9, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 2.9, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}

0.843 (+/-0.001) for {'C': 2.9, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 2.9, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 2.9, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.844 (+/-0.002) for {'C': 2.9, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.849 (+/-0.010) for {'C': 7.07, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 7.07, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.853 (+/-0.015) for {'C': 7.07, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.845 (+/-0.005) for {'C': 7.07, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.847 (+/-0.007) for {'C': 7.07, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 7.07, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.07, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 7.07, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 7.07, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.848 (+/-0.010) for {'C': 7.07, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.849 (+/-0.010) for {'C': 7.09, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 7.09, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.853 (+/-0.015) for {'C': 7.09, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.845 (+/-0.005) for {'C': 7.09, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.847 (+/-0.007) for {'C': 7.09, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 7.09, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}

0.843 (+/-0.001) for {'C': 7.09, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.09, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 7.09, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 7.09, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.848 (+/-0.010) for {'C': 7.09, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.849 (+/-0.010) for {'C': 7.1, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 7.1, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.853 (+/-0.015) for {'C': 7.1, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.845 (+/-0.005) for {'C': 7.1, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.847 (+/-0.007) for {'C': 7.1, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 7.1, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.1, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 7.1, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 7.1, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.848 (+/-0.010) for {'C': 7.1, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 2, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 2, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 2, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 2, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 2, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 2, 'gamma': 0.18, 'kernel': 'poly'}
0.849 (+/-0.010) for {'C': 7.11, 'degree': 2, 'gamma': 1.5, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 7.11, 'degree': 2, 'gamma': 1.6, 'kernel': 'poly'}
0.853 (+/-0.015) for {'C': 7.11, 'degree': 2, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 3, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 3, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 3, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 3, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 3, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 3, 'gamma': 0.18, 'kernel': 'poly'}
0.845 (+/-0.005) for {'C': 7.11, 'degree': 3, 'gamma': 1.5, 'kernel': 'poly'}
0.847 (+/-0.007) for {'C': 7.11, 'degree': 3, 'gamma': 1.6, 'kernel': 'poly'}
0.850 (+/-0.012) for {'C': 7.11, 'degree': 3, 'gamma': 1.8, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 4, 'gamma': 0.01, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 4, 'gamma': 0.03, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 4, 'gamma': 0.05, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 4, 'gamma': 0.15, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 4, 'gamma': 0.16, 'kernel': 'poly'}
0.843 (+/-0.001) for {'C': 7.11, 'degree': 4, 'gamma': 0.18, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 7.11, 'degree': 4, 'gamma': 1.5, 'kernel': 'poly'}
0.844 (+/-0.004) for {'C': 7.11, 'degree': 4, 'gamma': 1.6, 'kernel': 'poly'}
0.848 (+/-0.010) for {'C': 7.11, 'degree': 4, 'gamma': 1.8, 'kernel': 'poly'}

Tiempo de procesamiennto: 671.27 mins

Anexo F. Entorno de trabajo VS Code



Anexo G. Información de la base de datos