

Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Proyecto Tecnológico

**Algoritmos de aprendizaje automático para el análisis de
opiniones a partir de textos en español**

José Fabián Paniagua Reyes

2112002241

Asesor:

José Alejandro Reyes Ortiz

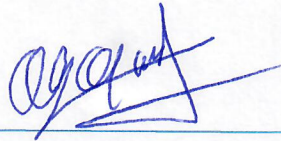
Departamento de Sistemas

Trimestre 17-Invierno

25 de Abril de 2017

Yo, José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma.



Yo, José Fabián Paniagua Reyes, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el repositorio Institucional de UAM Azcapotzalco.



Firma.

Resumen

La minería de datos es cada vez más importante para las empresas ya que la información que se puede obtener puede ser de gran interés debido a que permite adentrarse más en comportamientos de clientes o aspectos generales que sobre producción en una fábrica, o pueden encontrar valores de interés que nos permitan tomar decisiones que mejoren las prestaciones de algún dispositivo.

Este proyecto tiene como base el realizar técnicas de Procesamiento de Lenguaje Natural (PLN) e implementar cuatro diferentes procesos de Minería de Textos. La existencia de trabajos de minería de textos se centraliza en textos en inglés, la cantidad de documentos y herramientas existentes para textos en español no es tan grande por lo que implementar estas técnicas son de gran valor debido a la gran cantidad de información en el idioma español. Los resultados obtenidos al aplicar las técnicas de PLN y minería de textos entre diferentes idiomas deben ser diferente por lo que es de interés realizar una comparativa de estos resultados.

Una técnica utilizada para el procesamiento del lenguaje es la eliminación de palabras que no aportan valor al contenido del texto (*stopwords*) y para ello se utiliza una herramienta la cual obtiene la raíz de la palabra al remover las derivaciones en esta (*stemming*). Este tipo de procesos mejoran los resultados de los algoritmos encargados del aprendizaje en los textos provenientes de la red social Twitter. Además, se utilizan diferentes algoritmos como máquinas de soporte vectorial (SVM por sus siglas en inglés), Naïve Bayes, J48 y K-vecinos más cercanos para realizar la minería de texto y se comparan los resultados generados.

Tabla de contenido

Resumen	
1. Introducción.....	1
2. Justificación.....	1
3. Antecedentes.....	2
4. Objetivos.....	4
5. Marco Teórico.....	5
5.1 Sistema de información.....	5
5.2 Procesamiento de lenguaje natural.....	5
5.3 Algoritmos de Aprendizaje Automático.....	6
5.3.1 Algoritmo Bayes Naïve	7
5.3.2 Algoritmo K vecinos más cercanos	7
5.3.3 Algoritmo J48	7
5.3.4 Algoritmo Máquinas de Soporte Vectorial	7
6. Desarrollo	8
6.1 Diseño del sistema (diagrama de cases)	8
6.1.1 Clase eliminaUTF.....	8
6.1.2 Clase tokenStemming.	9
6.1.3 Clase Conteo	9
6.1.4 Clase pesado	10
6.2 Metodología de solución	10
6.2.1 Lectura de archivo TXT de tweets, eliminacion de encodificado.....	11
6.2.2 Fragmentación de cada tweet en un vector de palabras (Tokenización).....	12
6.2.3 Normalización de risa, dígitos y monosílabas	13
6.2.4 Eliminación de palabras sin valor (Stopwords).....	14
6.2.5 Obtención raíces de las palabras resultantes.....	15
6.2.6 Conteo de frecuencia de palabras.....	17
6.2.7 Evaluación de la frecuencia de palabras (TF-IDF).	17
6.2.8 Transformación a dato tipo matriz	18
6.2.9 Almacenamiento de resultados.....	20
6.3 Minería de opiniones.....	22
7. Resultados.....	29

8. Análisis y discusión de resultados.....	33
9. Conclusiones.....	34
10. Bibliografía.....	35
ANEXO 1	37
Código fuente del proyecto.....	37
ANEXO 2	42
Resultados del algoritmo Bayes Nãive	42
ANEXO 3	44
Resultados del algoritmo J48	44
ANEXO 4	46
Resultados del algoritmo Maquinas de Soporte Vectorial (SVM).....	46
ANEXO 5	48
Resultados del algoritmo K-Vecinos más cercanos.	48

1. Introducción

Las redes sociales forman parte de la vida cotidiana de un gran número de personas las cuales al tener libertad de publicar o documentar su vida en estas plataformas generan diariamente grandes cantidades de información. Twitter al ser un medio de comunicación libre o con menores restricciones para su uso, los usuarios pueden plasmar sus pensamientos reflejando sus opiniones.

La minería de opiniones o análisis de sentimientos es una línea de investigación moderna que se ha vuelto cada vez más importante gracias a la interacción de usuarios de redes sociales con productos, servicios, figuras públicas, instituciones, etcétera.

La minería de opiniones o análisis de sentimientos es una línea de investigación moderna que se ha vuelto cada vez más importante debido a la interacción de los usuarios de redes sociales con relación a productos, servicios, figuras públicas, instituciones, etcétera, emitiendo comentarios que reflejan el sentimiento de los usuarios que se transforma en información útil para diferentes entidades como empresarios, gobiernos, investigadores después de ser procesada por diferentes tecnologías de clasificación.

Las tecnologías de clasificación utilizan algoritmos que se caracterizan por métricas de eficiencia. En el presente proyecto se pretende realizar un estudio de utilización de los diferentes algoritmos y aplicaciones existentes analizando textos de opinión en español para realizar una comparativa con el uso de técnicas de minería de opiniones realizada con la herramienta de código abierto Weka [14].

2. Justificación

Todo desarrollo tecnológico es susceptible de tener fallas, Bautin et al 2016 [10], inclusive los programas estado del arte en traducción de lenguajes fallan al traducir grandes cantidades de texto, hacen errores serios en lo que traducen, y reducen textos bien formados en fragmentos.

La globalización ha facilitado el acceso a la información de diversas fuentes para cualquier persona. Sin embargo, desarrollar productos o metodologías de análisis de calidad requiere de investigaciones fundamentadas en las alternativas disponibles analizando las ventajas y desventajas que cada caso representa. Al enfrentarse a la problemática de realizar análisis de textos y generar métodos de calificación de un conjunto de datos de gran magnitud como Big Data se ve altamente beneficiado con la eficiencia y eficacia de los algoritmos y tecnologías utilizadas.

El microblogging, la actividad de generar comentarios cortos, informales y frecuentes en sitios web, permite que su uso sea accesible a cualquier usuario de la internet. En el sitio web, Alexa [13] se califica a la red Social Twitter en el 9° lugar de los sitios web con más popularidad con base al tráfico generado. El puesto 9° se puede traducir en una gran cantidad de usuarios los cuales alimentan a la red social renovando la información en diferentes tópicos. El uso de esta información puede ser de diversas maneras.

El uso de algoritmos de aprendizaje automático sobre textos de opinión en español puede encontrar aplicaciones comerciales, investigación científica, aplicación de leyes en diferentes estratos de organizaciones, corporaciones, gobierno e inclusive individuos, ya que representa el sentimiento desde diferentes perspectivas de una gran variedad de personas. Es así que el valor de la información generada puede ser de valor incalculable al convertirse en herramientas para corregir problemas diversos, crear nuevos productos, mejorar servicios satisfaciendo necesidades específicas, mejorar la imagen de una institución o complementar otras tecnologías.

La utilización de metodologías para generar conocimiento en el mundo actual está en auge y representa grandes oportunidades competitivas para diversas instituciones. Sin embargo, es un área de investigación y especialidad extensa ya que existen más algoritmos empleados en minería de datos y la realización de este proyecto permite tener un acercamiento al gran panorama de minería de datos.

Aplicaciones futuras de estas tecnologías podrían complementar sistemas de inteligencia artificial al realizarse más rápido utilizando algoritmos optimizados ya que las bases de datos que se pueden utilizar pueden tener proporciones muy grandes en comparación con la capacidad de procesamiento de un solo dispositivo. El dimensionamiento de la problemática amerita el uso de sistemas distribuidos con computo paralelo para tareas más extensas.

3. Antecedentes

A continuación, se enuncian ejemplos de documentos y proyectos relacionados con el presente que tienen relación con proyecto.

Proyecto terminal

1. Aplicación de Distintas Técnicas de Minería de Datos para el Tratamiento de Información [1].

Este proyecto terminal utiliza técnicas de minería de datos con el objetivo de realizar limpieza de datos. Una similitud es el tratamiento previo a la base de datos,

utiliza el algoritmo Bayes Naïve y la utilización de la aplicación Weka. Difiere en la aplicación de menos algoritmos que la presente propuesta de proyecto terminal.

2. Sistema Configurable de Minería Web [2].

El texto del proyecto terminal realiza una indexación de sitios web que resulte en búsquedas más eficientes. Una similitud de este proyecto es la utilización del proceso de clustering para la clasificación de datos. Las diferencias es que utiliza menos algoritmos de minería de datos y no utiliza la aplicación Weka.

3. Implementación de una red neuronal booleana multicapa en un FPGA [3].

El autor del proyecto terminal implementa una red neuronal la cual se ejecuta en un hardware programable (FPGA) para aprender funciones booleanas en menores tiempos. La similitud con este proyecto terminal es la implementación de una red neuronal. Difiere en el uso de la red neuronal, la implementación de otros algoritmos y el uso de la aplicación Weka.

Tesis

4. Minería de Opiniones Basada en Características Guiada por Ontologías [4].

La tesis doctoral tiene como objetivo descubrir una técnica para realizar minería de opiniones mediante el uso de ontologías con relación al texto procesado. Similitudes con el presente proyecto terminal son la base un procedimiento de análisis vectorial en un espacio de tres dimensiones y el análisis de información con relaciones léxicas. Diferente en la falta de uso de otros algoritmos y falta de uso de Weka.

5. Programación Matemática para las Máquinas de Vector de Apoyo [5].

El documento analiza el uso de Maquinas de Soporte Vectorial (SVM por sus siglas en inglés) el cual se utiliza con fines de minería de datos. Diferentes casos del uso son revisados por el autor como clasificación binaria, clasificación multi grupo y problemas lineales de gran tamaño entre otros. Esta tesis doctoral contempla varios algoritmos como SVM, Redes Neuronales. Similar a este proyecto terminal es el uso de SVM y difiere en la implementación de los otros algoritmos y comparación con Weka.

6. Advanced Models for Nearest Neighbor Classification Based on Soft Computing Techniques [6].

En la tesis se describe la clasificación de los tipos de algoritmos “*machine learning*”. Aspectos relevantes al momento de realizar comparativas en los resultados obtenidos en el momento de la implementación, mismos que son poco o no mencionados en otros trabajos son las cinco métricas de desempeño de los algoritmos de clasificación. Es similar el trabajo de investigación del autor ya que involucra diferentes implementaciones de K-vecinos más cercanos (KNN por sus siglas en inglés), mismo que se investiga en este proyecto terminal. Difiere por la falta de implementación de los demás algoritmos y comparativa con Weka.

Artículos

7. Event based sentence level interpretation of sentiment variation in twitter data [7].

En este documento, los autores realizan un procesamiento previo de la base de datos de tweets para mejorar los resultados. Es similar a este documento en el pre procesamiento de los datos. Diferente en el uso de los algoritmos planteados en este proyecto terminal.

4. Objetivos

General

Implementar, evaluar y comparar algoritmos de aprendizaje automático para el análisis de opiniones a partir de textos en español extraídos de la red social Twitter.

Específicos

- Implementar los algoritmos K-vecinos más cercanos (KNN), Bayes Naïve, J48, Maquinas de Soporte Vectorial (SVM).
- Desarrollar un módulo para realizar una evaluación de los algoritmos K-vecinos más cercanos, Bayes Naïve, Redes Neuronales, Maquinas de Soporte Vectorial, utilizando métricas precisión, recuerdo (exhaustividad) y F-measure.
- Desarrollar un módulo de análisis de resultados estructurando (tablas y gráficos) con base en los resultados obtenidos en las métricas.

5. Marco Teórico

Los textos al ser escritos por personas reflejan su forma de pensar y sentir, sin embargo, para un sistema de información no le es sencillo procesar información y saber qué significado específico tiene una palabra ya que puede tener múltiples significados que se ajustan según el contexto en el que se encuentra. Un sistema de información requiere de un proceso previo que le indique de qué forma clasificar un texto.

5.1 Sistema de información

Un sistema de información es un software con el objetivo de procesar información mediante el tratamiento de datos organizados y facilitar su uso posterior. Los datos se muestran a través de un modelo gráfico que permita su representación de manera estructurada. Los componentes de un sistema de información son:

- Datos
- Usuarios
- Recursos
- Actividades de procesamiento
- Información

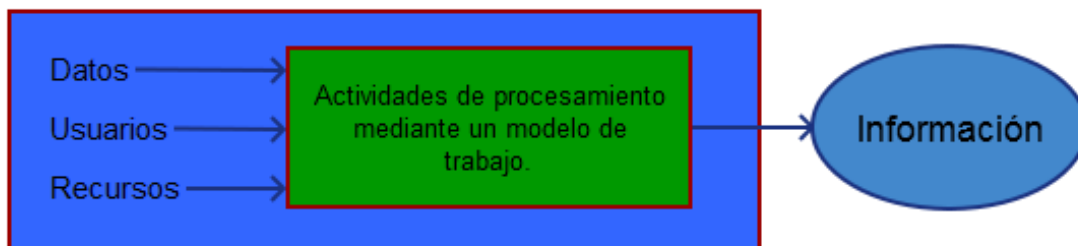


Figura 1. Componentes de un Sistema de Información.

5.2 Procesamiento de lenguaje natural

El procesamiento de lenguaje natural es un conjunto de técnicas para facilitar la comunicación con la computadora por lo que es fundamental en el área de Inteligencia Artificial. Facilita realizar desarrollos computacionales que estén relacionados con procesamiento de textos como traducción automática, recuperación de información, extracción de información y reconocimiento de voz.

Una arquitectura general de un sistema de Procesamiento de Lenguaje Natural está constituida de los siguientes bloques descritos en la figura X.

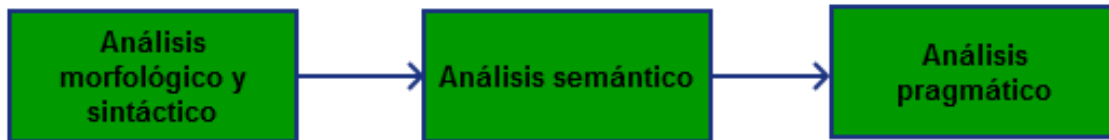


Figura 2. Boques de tareas principales en PLN.

5.3 Algoritmos de Aprendizaje Automático

Es conjunto de procesos con los cuales un sistema es capaz de mejorar y adaptarse para la resolución de tareas y aplica métodos de inferencia inductiva a las tareas procesadas. Para un conjunto de datos puede utilizarse una partición para realizar el aprendizaje de los datos de entrenamiento un subconjunto de validación y un conjunto de prueba.

Existen diferentes formas de clasificar los algoritmos de aprendizaje según los métodos utilizados para llevar acabo la tarea.

- Descripción: se realiza un análisis previo de los datos (resumen, propiedades de los datos, casos extremos). De esta manera el usuario tiene buen conocimiento de los datos. El objetivo es encontrar medias, estándares, desviaciones, etc.
- Clasificación: Ya que los datos tienen valores que pueden ser segmentados, se pueden definir clases para agruparlos con etiquetas.
- Estimación: en este caso el objetivo es definir un modelo que permita predecir los valores de una clase continua.
- Segmentación: Para lograr el objetivo se procede a realizar grupos a partir de las clases. Se subdividen en métodos exhaustivos, con traslapes, así como mutuamente exclusivas o jerárquicas, posteriormente se utilizan algoritmos de clasificación.
- Optimización y búsqueda: Podemos encontrar varios algoritmos de este tipo algunos son individual o poblacional, local o global, algoritmos genéticos, deterministas o aleatorios.

5.3.1 Algoritmo Bayes Naïve

Este algoritmo de clasificación tiene como base el teorema de Bayes con independencia de suposiciones entre predictores. El algoritmo evalúa una variable llamada variable de clase y un conjunto de variables de atributos. La tarea de clasificación se hace mapeando una instancia de una variable X con el valor de la variable Y . Se obtiene la probabilidad posterior de la clase objetivo dando el atributo.

5.3.2 Algoritmo K vecinos más cercanos

El fundamento del algoritmo es que cada nuevo caso se clasifica en la clase con mayor frecuencia a la que sus k vecinos son más cercanos. En comparación con los demás algoritmos de clasificación para un nuevo caso, se realizan dos procesos uno es inducción de la clasificación y el otro la deducción que es cuando se efectúa la clasificación de un nuevo modelo. Para el caso de K vecinos más cercanos no existe un modelo explícito, las tareas antes mencionadas se conjuntan en una llamada *transducción*.

5.3.3 Algoritmo J48

Es también conocido C4.5, un algoritmo de inducción el cual implementa un árbol derivado de reglas las cuales agrupan casos en los que se puede clasificar una clase. Se evalúan dos aspectos, primero el valor de la información que se obtiene de una rama del árbol y el segundo, al generar una regla se obtiene la ganancia por mejora en el desempeño global, en otras palabras, se evalúa si hay beneficio.

5.3.4 Algoritmo Máquinas de Soporte Vectorial

El proceso de este algoritmo es analizar las propiedades de una clase con las que realizará la clasificación del modelo. Los elementos del conjunto original son comparados y clasificados con cada uno de los modelos y así destaca a que modelo pertenece. En términos matemáticos, se utiliza una función para mapear el conjunto original y las clases. SVM mapea de un espacio de características en R^2 a un espacio en R^3 , posteriormente encuentra un hiperplano que separe y maximice la distancia entre las clases. Esta característica permite simplificar el problema de clasificación

6. Desarrollo

6.1 Diseño del sistema (diagrama de cases)

El código del proyecto tiene la estructura mostrada en el diagrama de clases UML. Este diagrama no presenta una clase principal que no existe como tal, el script principal es el que llama a las clases. Las clases se muestran en la Figura 3.

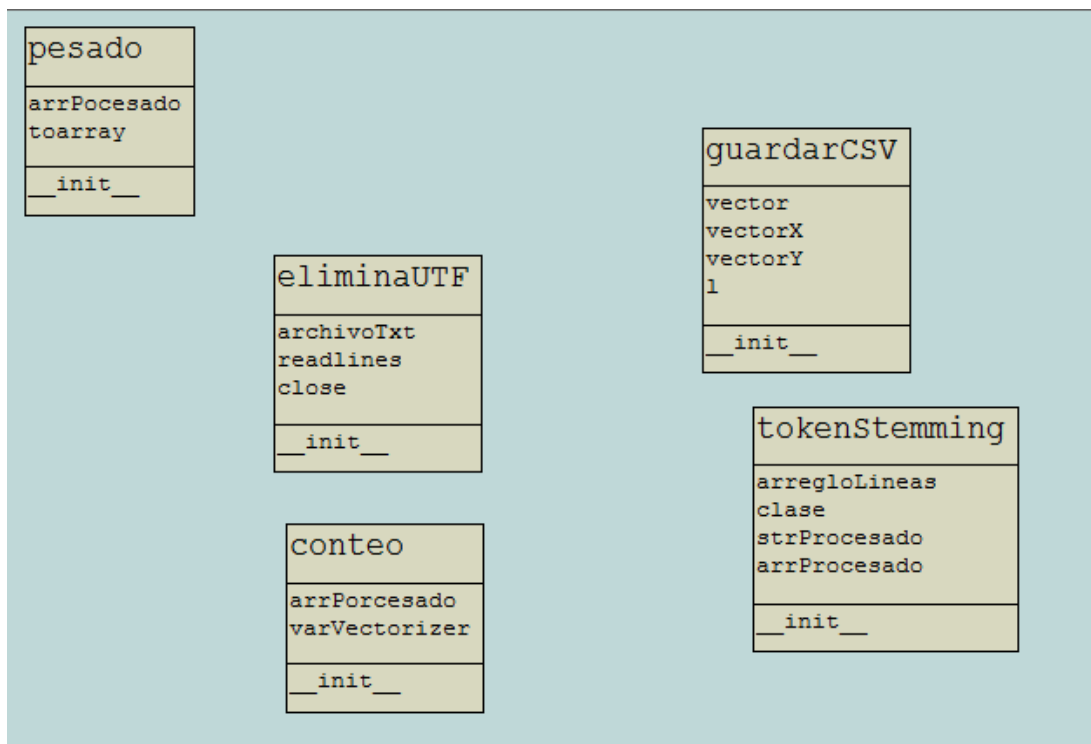


Figura 3. Diagrama de clases de este proyecto.

6.1.1 Clase eliminaUTF

Esta clase tiene un único método que tienen las tareas siguientes:

Primero: es leer los primeros bytes del archivo a procesar y eliminarlos. El objetivo de eliminar los caracteres que se encuentran al principio del archivo es permitirle a el método de lectura correcta del siguiente método. El resultado se devuelve en la variable global archivoTxt.

Segundo: se ocupa de leer las líneas de texto que son cada uno de los tweets junto con un espacio generado por un carácter tabulador seguido de una etiqueta de clasificación.

Posteriormente se almacena las líneas de código y devuelve el resultado en una variable tipo vector. El resultado se devuelve con la variable `readlines`.

Tercero, `close`: su única función es la de cerrar el archivo de origen de los tweets.

La clase `pesado` se puede consultar en el ANEXO 1.

6.1.2 Clase `tokenStemming`.

La clase cuenta un método encargado de diferentes procesos, estos son:

- Primero: toma cada una de las líneas del arreglo `Lineas` y las fragmenta en tokens utilizando la clase `word_tokenize()` de `nltk.tokenize`.
- Segundo: para cada uno de los tokens dentro de un ciclo el cual se apoya del uso de expresiones regulares. Se buscan coincidencias de las palabras onomatopéyica que representan risa como son “jajaja”, “jejejej” o “jiji” con cualquier combinación de longitud y se toma en cuenta la posibilidad de error de escritura a la falta la última vocal como se muestra en el ejemplo. También se eliminan las coincidencias de números y palabras de longitud de dos caracteres.

El proceso siguiente es eliminar la última localización del vector de `arregloLineas` que es el que contiene la etiqueta de clasificación del tweet.

A continuación, se realiza la reducción de las palabras eliminando los sufijos de estas. Este proceso se realiza con la clase `SnowballStemmer` de `nltk.stem`

El paso final del método es buscar coincidencias de las palabras procesadas en una lista de palabras caracterizadas como palabras que no aportan al corpus, `stopwords`, para realizar este procedimiento se utiliza la clase `stopwords.words('spanish')` la utiliza la versión de palabras en español.

Las palabras procesadas se guardan en la variable de tipo arreglo de vectores `arrProcesado`.

La clase `pesado` se puede consultar en el ANEXO 1.

6.1.3 Clase `Conteo`

Para realizar el procesamiento en el único método de esta clase se usa el atributo `CountVectorizer(min_df=1)` que se encarga de realizar una matriz de conteo de los tokens procesados por la clase `tokenStemming`. La clase `pesado` se puede consultar en el ANEXO 1.

6.1.4 Clase pesado

Con el método por defecto de esta clase se realiza un cálculo de peso de las palabras en relación con el documento analizado, en este caso el tweet, utilizando TfidfVectorizer de la librería sklearn.feature_extraction.text y que se almacena haciendo una transformación a una matriz de los resultados obtenidos para así devolver el valor. La clase pesado se puede consultar en el ANEXO 1.

6.2 Metodología de solución

Los pasos realizados para la elaboración del proyecto se encuentran en el diagrama de bloques mostrado en la figura 4.



Figura 4. Diagrama de bloques del proyecto

El punto de partida para este de este proyecto es un corpus de textos extraídos de la red social Twitter. Este corpus se encuentra procesado anteriormente al planteamiento de este proyecto para el cual se ha eliminando acentos y caracteres especiales como a “ñ”, menciones a otros usuarios con el uso de la arroba “@” así como el uso de “hashtag”.

Realizar este proceso es indispensable ya que no solo mejora los resultados de los algoritmos si no que también los siguientes procesos no soportan el uso de estos caracteres.

6.2.1 Lectura de archivo TXT de tweets, eliminacion de encodificado

Se abre el archivo txt para poder acceder a los tweets. Sin embargo el archivo contiene caracteres los cuales indican el tipo de encodificación y que el metodo de lectura toma encuentra. La clase **eliminarUTF** se encarga de esta tarea, se logra descartar los caracteres durante la lectura al avanzar en la primera línea del contenido del archivo.

La clase contiene la variable `extensionBOM` la cual esta formada de 3 tuplas. Las tuplas se conforman de 3 tipos de encodificación UTF, UTF-8, UTF-16 y UTF-32 asi como la longitud de los caracteres que ocupa en un archivo que son 3, 2 y 4. Se leen los primeros valores del archivo para que se comparen estos valores con los del arreglo de tuplas. En el caso que se encuentren los valores en archivo se almaceana el valor para que el apuntador en el archivo, se recorra a la posicion posterior a los caracteres UTF.

Una vez descartados los valores de los caracteres UTF se procede a leer todas las lineas del archivo y devolver la variable que las contiene con el atriburo `archivoTxt` y el método `readlines`. Ejemplificación del proceso se muestra en la figura 5.


```

Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (
AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\CodeLines\Dropbox\Proyecto Terminal\Python Scripts\parte6.0 c
lases final.py
Leyendo tweets
['yurenitta928 cono eske lo e visto y era un bmw si llega a ser un peugeot jajaj
aja\tneutral\r\n', 'automocion jajaj y recambios el que mas baja damiler 402 bmw
388 \tnegative\r\n', 'ni ferrari ni lamborghini ni bmw yo quiero es un popular
vw de estos \tnegative\r\n', 'nohasestadoenelsevero si no has pensado como se ha
pagado el juani el bmw\tneutral\r\n', 'dondatos se vende bmw 315 ano 82 estado
inmejorable y original 29mill conversable \tpositive\r\n', 'fitornello casioedif
ice corrio en 3 equiposbmw saubertoro rosso y red bull\tpositive\r\n', 'a ver si
nos intervienen ya que tengo que rehipotecar la casa para cambiar el bmw\tnegat
ive\r\n', 'bmw 2011 bmw 5 series sedan 4dr sdn 550i rwd sedan roswell erxjyspjdz
ojm ngpzy qezpiydzz btosob \tpositive\r\n', 'no se como puede haber gente asi es
como tener un bmw cambiarlo x un andino y luego reclamarle al bmw x dejarse cam
biar\tneutral\r\n', 'anuncio gratis bmw r 1200 gs valencia espana 14900 \tpositi
ve\r\n', 'fabbrojony17 bmw color blanco\tneutral\r\n', 'bmw abraira una nueva fil
ial en barcelona \tneutral\r\n', 'un bmw por antiguo que sea siempre sera un bmw
truefacts\tpositive\r\n', 'bmw zagato coupe un deportivo unico en el mundo \tpo
sitive\r\n', 'bolivene el problema no es tener un bmw el problema es para que lo
usas\tneutral\r\n', 'scientek jejeje es cierto con tony se puede esperar todo p
or cierto haz visto el corto de bmw beatthedevil de tony\tneutral\r\n', 'navegan
do en la x6 bmw en orlando ready para esta noche en caridad restaurant oyo \tpos

```

Figura 5. Resultado de la lectura de las líneas del corpus

6.2.2 Fragmentación de cada tweet en un vector de palabras (Tokenización)

Para el segundo bloque la clase en funcionamiento es **tokenSteming**, las líneas de tweets se requieren fragmentadas en unidades mínimas que es una palabra. A este proceso se le conoce como **tokenizacion**, este proceso facilita la manipulación del texto. Se realiza con la clase *word_tokenize* de la librería *nltk.tokenize*.

Dentro de un ciclo que se repite para cada línea, se analiza la cantidad de los tokens, se almacenan los tokens en un arreglo llamado tokens, y se almacena el ultimo token que clasifica al tweet. Ejemplificación del proceso se muestra en la figura 6.

```

Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (
AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\CodeLines\Dropbox\Proyecto Terminal\Python Scripts\UsingJ48.p
y
Leyendo tweets
18
yurenitta928 como eske lo e visto y era un bmw si llega a ser un peugeot jajajaj
aj          neutral

[u'yurenitta928', u'como', u'eske', u'lo', u'e', u'visto', u'y', u'era', u'un',
u'bmw', u'si', u'llega', u'a', u'ser', u'un', u'peugeot', u'jajajajajaj', u'neutra
l']
12
automocion y recambios el que mas baja damiler 402 bmw 388          negative

[u'automocion', u'y', u'recambios', u'el', u'que', u'mas', u'baja', u'damiler',
u'402', u'bmw', u'388', u'negative']
15
ni ferrari ni lamborghini ni bmw yo quiero es un popular vw de estos          negative

[u'ni', u'ferrari', u'ni', u'lamborghini', u'ni', u'bmw', u'yo', u'quiero', u'es
', u'un', u'popular', u'vw', u'de', u'estos', u'negative']
14
nohasestadoenelsevero si no has pensado como se ha pagado el juani el bmw
neutral

[u'nohasestadoenelsevero', u'si', u'no', u'has', u'pensado', u'como', u'se', u'h
a', u'pagado', u'el', u'juani', u'el', u'bmw', u'neutral']
14
dondatos se vende bmw 315 ano 82 estado inmejorable y original 29mill conversabl
e          positive

[u'dondatos', u'se', u'vende', u'bmw', u'315', u'ano', u'82', u'estado', u'inmej
orable', u'y', u'original', u'29mill', u'conversable', u'positive']

```

Figura 6. Resultado del proceso de tokenizacion de las líneas del corpus.

Al arreglo de vectores resultante del proceso de fragmentación se le elimina el último elemento el cual es una etiqueta de clasificación del tweet, y se almacena en un vector por separado para posteriormente reincorporar al resultado obtenido y permitir que el texto del tweet sea procesado sin la etiqueta que no forma parte del texto analizar.

6.2.3 Normalización de risa, dígitos y monosílabas

Los textos al ser informales pueden tener expresiones de sentimiento en palabra que no existen en un diccionario como son las onomatopeyas utilizadas para expresar la risa. En un ciclo anidado con el primero se realiza lo siguiente. En el caso de que existan múltiples variaciones de su escritura se emplean expresiones regulares las cuales

buscan las coincidencias de la subcadena “ja” acompañada de una o más coincidencias de “ja”. Los resultados positivos de estas búsquedas son remplazados por la palabra risa con el objeto de no afectar el posible significado que fue pretendido en el corpus. Ejemplificación del proceso se muestra en la figura 7.

```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\CodeLines\Dropbox\Proyecto Terminal\Python Scripts\UsingJ48.p
y
Leyendo tweets
18
[u'yurenitt']
[u'yurenitt']
[u'yurenitt', u'eske']
[u'yurenitt', u'eske', u'']
[u'yurenitt', u'eske', u'', u'']
[u'yurenitt', u'eske', u'', u'', u'vist']
[u'yurenitt', u'eske', u'', u'', u'vist', u'']
[u'yurenitt', u'eske', u'', u'', u'vist', u'']
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'']
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw']
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'']
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'', u'lleg']
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'', u'lleg', u'']
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'', u'lleg', u'', u
'ser']
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'', u'lleg', u'', u
'ser', u'']
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'', u'lleg', u'', u
'ser', u'', u'peugeot']
jajajajaj
risa
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'', u'lleg', u'', u
'ser', u'', u'peugeot', u'ris']
12
[u'automocion']
[u'automocion', u'']
[u'automocion', u'', u'recambi']
[u'automocion', u'', u'recambi', u'']
[u'automocion', u'', u'recambi', u'']
```

Figura 7. Resultados de la normalización de la risa.

6.2.4 Eliminación de palabras sin valor (Stopwords)

Los tweets al provenir de un usuario promedio es probable que contengan palabras que no aportan valor a el contenido del tweet por lo que estas son eliminadas. Se utiliza la clase stopwords de la librería nltk.corpus para buscar coincidencias con una bolsa de palabras que no aportan información a un corpus, en caso de existir una coincidencia la

palabra examinada se descarta y pasa la siguiente. Ejemplificación del proceso se muestra en la figura 7.

```
>>>
RESTART: C:\Users\CodeLines\Dropbox\Proyecto Terminal\Python Scripts\UsingJ48.p
y
Leyendo tweets
18
yurenitta928
<_sre.SRE_Match object at 0x0000000018467BF8>
yurenitt
[u'yurenitt']
cono
con
[u'yurenitt']
eske
eske
[u'yurenitt', u'eske']
lo

[u'yurenitt', u'eske', u'']
e

[u'yurenitt', u'eske', u'', u'']
visto
vist
[u'yurenitt', u'eske', u'', u'', u'vist']
y

[u'yurenitt', u'eske', u'', u'', u'vist', u'']
era
era
[u'yurenitt', u'eske', u'', u'', u'vist', u'']
un

[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'']
bmw
bmw
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw']
si

[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'']
```

Figura 7. Resultado del proceso de Stemming y eliminación de Stopwords.

6.2.5 Obtención raíces de las palabras resultantes.

Al existir variaciones de una palabra, la idea tiene el mismo origen que es la raíz de la palabra. La obtención de las raíces de cada palabra se realiza con la clase

SnowballStemmer de la librería *nltk.stem*. Este procesamiento logra obtener una sola raíz que puede asignarse de igual manera para un grupo de palabras y así obtener un diccionario (lexicón) de palabras normalizado. Ejemplificación del proceso se muestra en la figura 8.

```
[u'yurenitt', u'eske', u'', u'', u'vist', u'', u'', u'bmw', u'', u'lleg', u'', u
'ser', u'', u'peugeot', u'ris']
[u'yurenitt eske vist bmw lleg ser peugeot ris']
12
[u'automocion']
[u'automocion', u'']
[u'automocion', u'', u'recambi']
[u'automocion', u'', u'recambi', u'']
[u'automocion', u'', u'recambi', u'']
[u'automocion', u'', u'recambi', u'', u'mas']
[u'automocion', u'', u'recambi', u'', u'mas', u'baj']
[u'automocion', u'', u'recambi', u'', u'mas', u'baj', u'damil']
[u'automocion', u'', u'recambi', u'', u'mas', u'baj', u'damil', u'']
[u'automocion', u'', u'recambi', u'', u'mas', u'baj', u'damil', u'', u'bmw']
[u'automocion', u'', u'recambi', u'', u'mas', u'baj', u'damil', u'', u'bmw', u'']
]
[u'yurenitt eske vist bmw lleg ser peugeot ris', u'automocion recambi m
as baj damil bmw ']
15
[u'']
[u'', u'ferrari']
[u'', u'ferrari', u'']
[u'', u'ferrari', u'', u'lamborghini']
[u'', u'ferrari', u'', u'lamborghini', u'']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw', u'']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw', u'', u'quier']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw', u'', u'quier', u'']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw', u'', u'quier', u'', u'']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw', u'', u'quier', u'', u'', u'p
opul']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw', u'', u'quier', u'', u'', u'p
opul', u'']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw', u'', u'quier', u'', u'', u'p
opul', u'', u'']
[u'', u'ferrari', u'', u'lamborghini', u'', u'bmw', u'', u'quier', u'', u'', u'p
opul', u'', u'']
[u'yurenitt eske vist bmw lleg ser peugeot ris', u'automocion recambi m
as baj damil bmw ', u' ferrari lamborghini bmw quier popul ']
```

Figura 8. Obtención de raíces del vocabulario (Stemming).

6.2.6 Conteo de frecuencia de palabras

Una vez que las palabras están procesadas de manera óptima, es necesario analizar la frecuencia de aparición en cada tweet. Para realizar esta tarea se utiliza la clase `CountVectorizer` de la librería `sklearn.feature_extraction.text`. La cual genera un vector de longitud optimizada en el cual se incrementa en 1 en caso de que un elemento del lexicon esté contenido para cada uno de las líneas (tweets) del corpus. Ejemplificación del proceso se muestra en la figura 9.

```
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\CodeLines\Dropbox\Proyecto Terminal\Python Scripts\parte6.0 c
lases SUCIO.py
Leyendo tweets
[u'yurenitt eske vist bmw lleg ser peugeot ris']
[u'yurenitt eske vist bmw lleg ser peugeot ris', u'automocion ris recamb
i mas baj damil bmw ']
Conteo de palabras
[[0 0 1 0 1 1 0 1 0 1 1 1]
 [1 1 1 1 0 0 1 0 1 1 0 0]]
```

Figura 9. Vector de Conteo del corpus de los 2 primeros tweets.

6.2.7 Evaluación de la frecuencia de palabras (TF-IDF).

Posteriormente se utiliza la clase `TfidfVectorizer` de la librería `sklearn.feature_extraction.text` para realizar la tarea de pesado. Se utiliza el conjunto de vectores generados por `CountVectorizer` para obtener los valores requeridos de frecuencia de términos – frecuencia inversa en el documento (TF-IDF). Se utiliza el proceso matemático siguiente:

$$TF - IDF = Tf \cdot IDf$$

$$Tf = n_t$$

$$IDf = \log\left(\frac{1 + n_d}{1 + df}\right) + 1$$

$$V_{norm} = \frac{v}{|v_i|^2}$$

Nd = Número total de documentos.

Df = Número de documentos que contienen el término.

En la figura 16 se muestra el contenido del vector 0 del proceso TF-IDF de un corpus mínimo de 2 tweets donde el número de ocurrencias está dado por el vector que realizo el conteo por CountVectorizer – [0,0,1,0,1,1,0,1,0,1,1,1,1] = TF

Se calcula cada valor idf, para el primer 1 de vector, para la palabra “bmw”

$$n_d = 2 \text{ y } df = 2$$

$$IDf = \log\left(\frac{1+2}{1+2}\right) + 1 = 1$$

Calculamos la norma

$$norma = \sqrt{0^2 + 0^2 + 1^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 0^2 + 1^2 + 1^2 + 1^2 + 1^2} = 3.7218$$

Realizando $\frac{Tf*IDf}{norma}$ se obtiene el vector

$$[0,0,0.2686,0,0.3776,0.3776,0,0.3776,0,0.2686,0.3776,0.3776,0.3776]$$

Como se puede apreciar es el mismo vector para para el vector cero de la figura 10.

```
Leyendo tweets
Conteo de palabras
Conteo de palabras
Pesado - tf-idf
(2, 13)
[[ 0.          0.          0.26868528  0.          0.37762778  0.37762778
  0.          0.37762778  0.          0.26868528  0.37762778  0.37762778
  0.37762778]
 [ 0.4078241  0.4078241  0.29017021  0.4078241  0.          0.
  0.4078241  0.          0.4078241  0.29017021  0.          0.          0.
  ]
Tamaño de vocabulario: 13
[u'automocion', u'baj', u'bmw', u'damil', u'eske', u'lleg', u'mas', u'peugeot',
u'recambi', u'ris', u'ser', u'vist', u'yurenitt', 'clase']
```

Figura 10. Resultados del cálculo de TF-IDF para las 2 primeras líneas del corpus.

6.2.8 Transformación a dato tipo matriz

EL tipo de dato resultante del proceso anterior de es una matriz densa, es decir una matriz donde solo los valores diferentes de cero se almacenan, sin embargo, se requiere una matriz completa para almacenarse en archivo. Con el método toarray() se transorma completamente el dato a un vector de vectores por lo que representa una matriz completa. La matriz densa se ejemplifica en la figura 11 y la matriz completa se muestra en la figura 12.

```

Python 2.7.13 (v2.7.13:a06454blafa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (
AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\CodeLines\Dropbox\Proyecto Terminal\Python Scripts\UsingJ48.p
y
Leyendo tweets
Conteo de palabras
(0, 152)      1
(0, 133)      1
(0, 165)      1
(0, 97)       1
(0, 22)       1
(0, 191)      1
(0, 66)       1
(0, 199)      1
(1, 50)       1
(1, 16)       1
(1, 105)      1
(1, 146)      1
(1, 15)       1
(1, 152)      1
(1, 22)       1
(2, 136)      1
(2, 143)      1
(2, 94)       1
(2, 74)       1
(2, 22)       1
(3, 90)       1
(3, 127)      1
(3, 38)       1
(3, 131)      1
(3, 117)      1
:           :
(30, 36)      1
(30, 125)     1
(30, 83)      1
(30, 197)     1
(30, 135)     1
(30, 177)     1

```

Figura 11. Muestra de la primera parte del proceso de pesado del corpus. Impresión en forma de matriz densa.


```

RESTART: C:\Users\CodeLines\Dropbox\Proyecto Terminal\Python Scripts\UsingJ48.p
y
Leyendo tweets
Conteo de palabras
Pesado - tf-idf
Tamaño de vocabulario: 202
Escribiendo matriz en archivo CSV
fila: 0
[ 0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.11143697 0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.41118221 0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.36734728 0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.41118221 0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.3121
218
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.3362459 0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.36734728 0.          0.          0.
  0.          0.          0.          0.          0.41118221 0.          0.
]

```

Figura 12. Impresión de vector resultante del proceso de pesado TF-IDF en forma completa.

6.2.9 Almacenamiento de resultados.

En el paso final de pre-procesamiento del corpus es el almacenamiento del lexicón, Matriz de pesado y el vector que contiene la clase de polaridad. En el caso del lexicón se le agrega la etiqueta clase al final del vector, la función de este vector será la de crear una cabecera en el archivo de resultados. La utilidad de la cabecera de resultados es permitir al programa Weka identificar los atributos que contiene cada instancia (tweet) y así realizar la clasificación de estas.

Para el proceso de escritura en archivo se utiliza la clase writer de la librería CSV y se implementa una instrucción para escribir el lexicón en forma de cabecera. A continuación, dos ciclos anidados para recorrer cada elemento de la matriz y mediante el método writerow de la clase DictWriter en la librería CSV se realiza la escritura. Un ejemplo de su funcionamiento de muestra en la figura 13.

```
RESTART: C:\Users\CodeLines\Dropbox\Proyecto Terminal\Python Scripts\UsingJ48.p
y
Leyendo tweets
Conteo de palabras
Pesado - tf-idf
Tamaño de vocabulario: 13
Escribiendo matriz en archivo CSV
[u'automocion', u'baj', u'bmw', u'damil', u'eske', u'lleg', u'mas', u'peugeot',
u'recambi', u'ris', u'ser', u'vist', u'yurenitt', 'clase']
fila: 0
0.0
0.0
0.268685276185
0.0
0.377627780741
0.377627780741
0.0
0.377627780741
0.0
0.268685276185
0.377627780741
0.377627780741
0.377627780741
neutral
fila: 1
0.40782410415
0.40782410415
0.290170208991
0.40782410415
0.0
0.0
0.40782410415
0.0
0.40782410415
0.290170208991
0.0
0.0
0.0
negative
>>> |
```

Figura 13. Muestra de los datos que se escriben en el archivo en formato separado por comas, csv.

6.3 Minería de opiniones

Con el explorador de Weka se carga el archivo resultante del pre-procesado en el botón explorer como se muestra en la figura 14.



Figura 14. Proceso para abrir el explorador de Weka.

Se selecciona el archivo en con el botón open, se busca el archivo en la carpeta que contiene al archivo en formato CSV. Se muestra la aplicación para cargar el archivo en la figura 15.

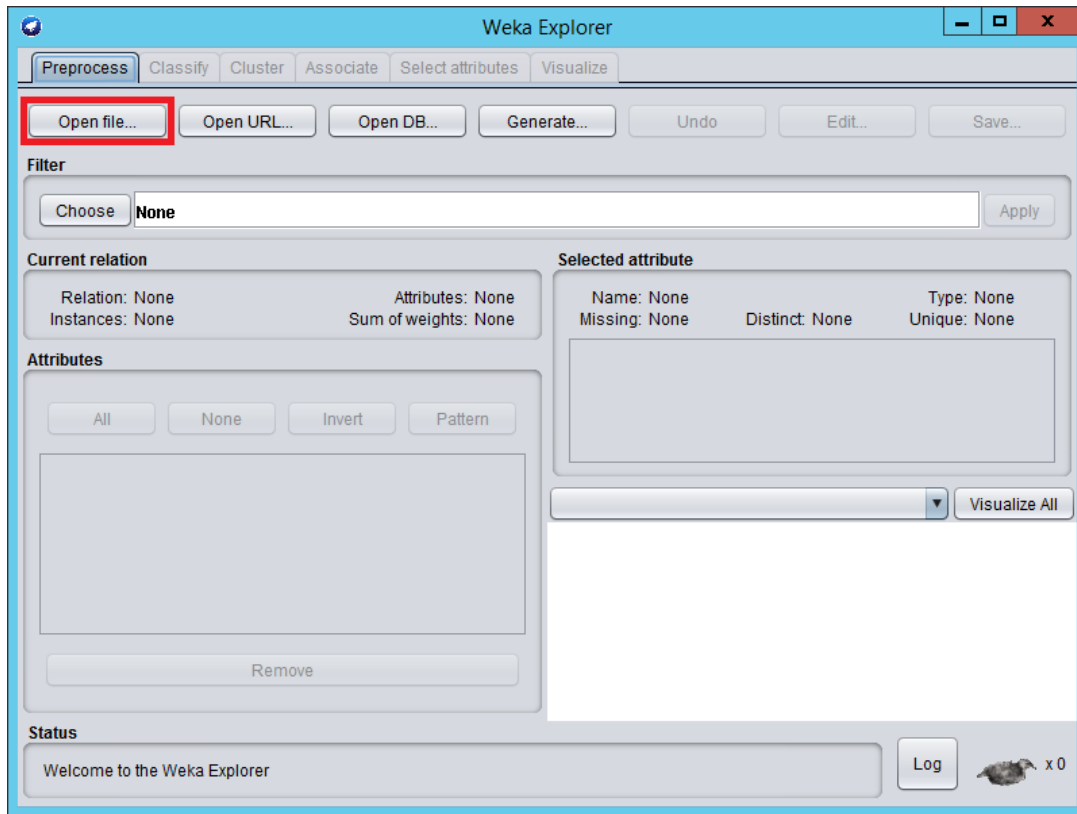


Figura 15. Proceso para abrir un archivo CSV.

Posteriormente en la pestaña de clasificación se selecciona el algoritmo que se ejecutará como se muestra en la figura 16.

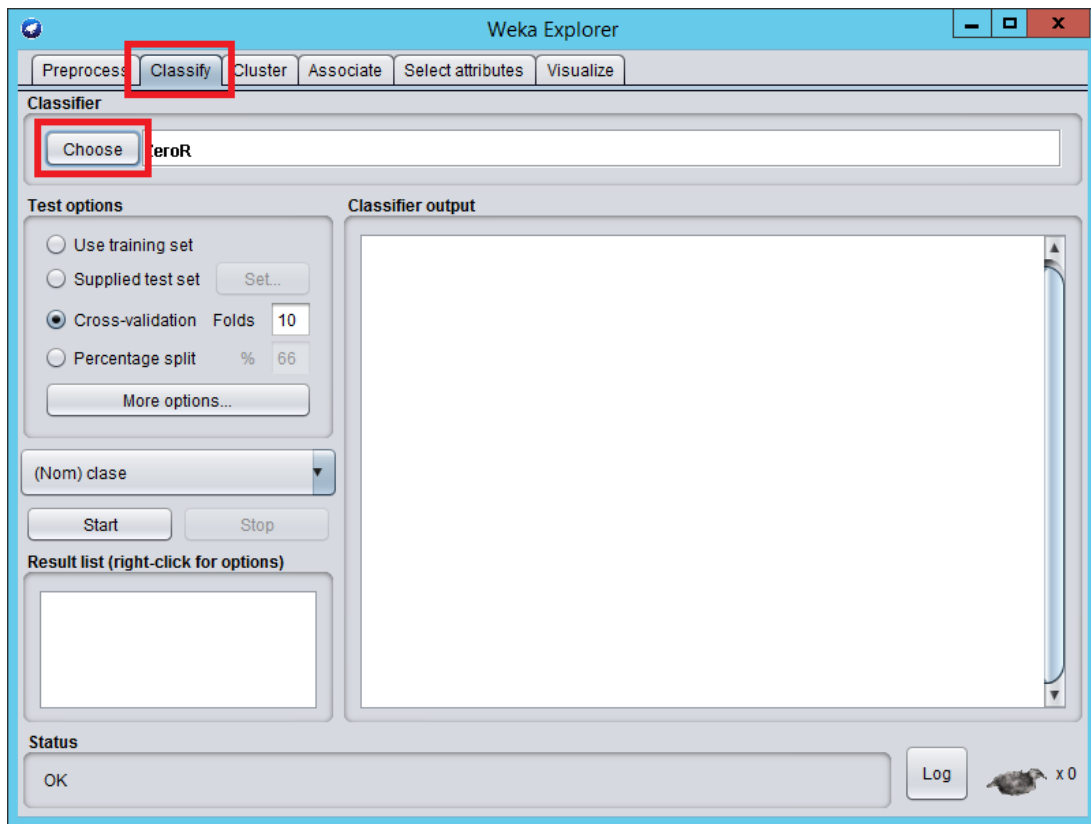


Figura 16. Proceso para seleccionar un algoritmo de clasificación.

El algoritmo Bayes Naïve se encuentra en la sección bayes como se ejemplifica en la figura 17.

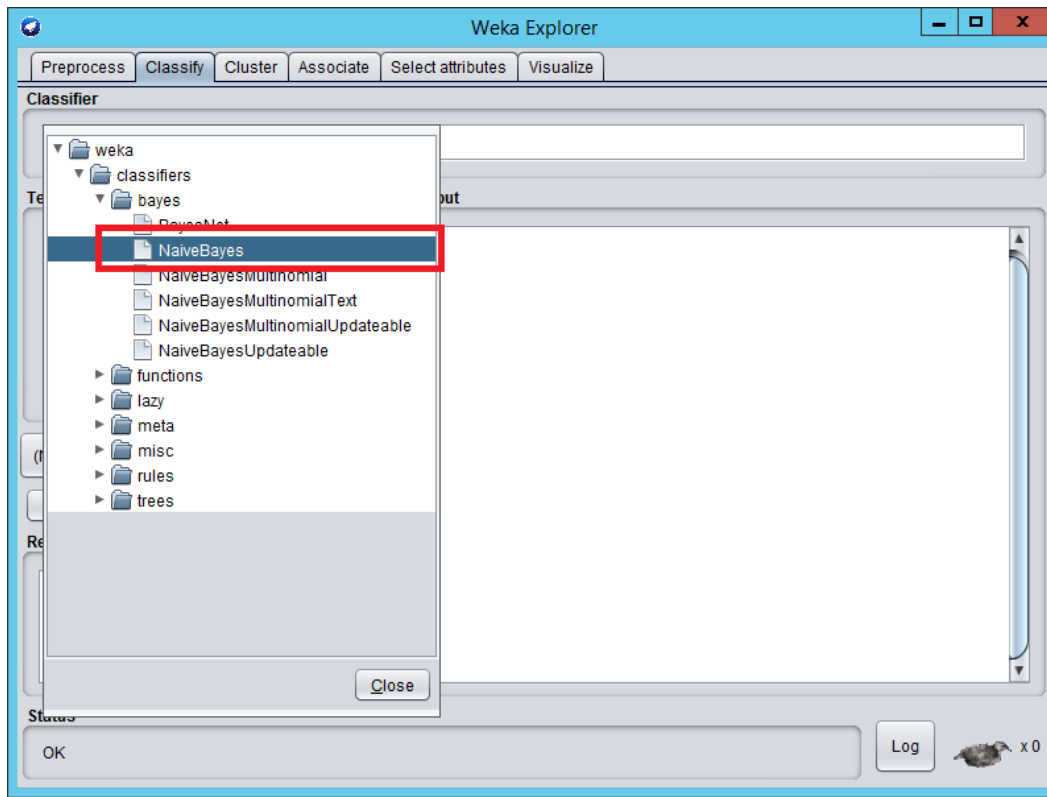


Figura 17. Selección del algoritmo Bayes Naïve.

Se procede a verificar que los parámetros de configuración sean por defecto para los cuatro algoritmos, se accede a estos al dar clic en el nombre del algoritmo seleccionado. La ventana de configuraciones se muestra en la figura 18.

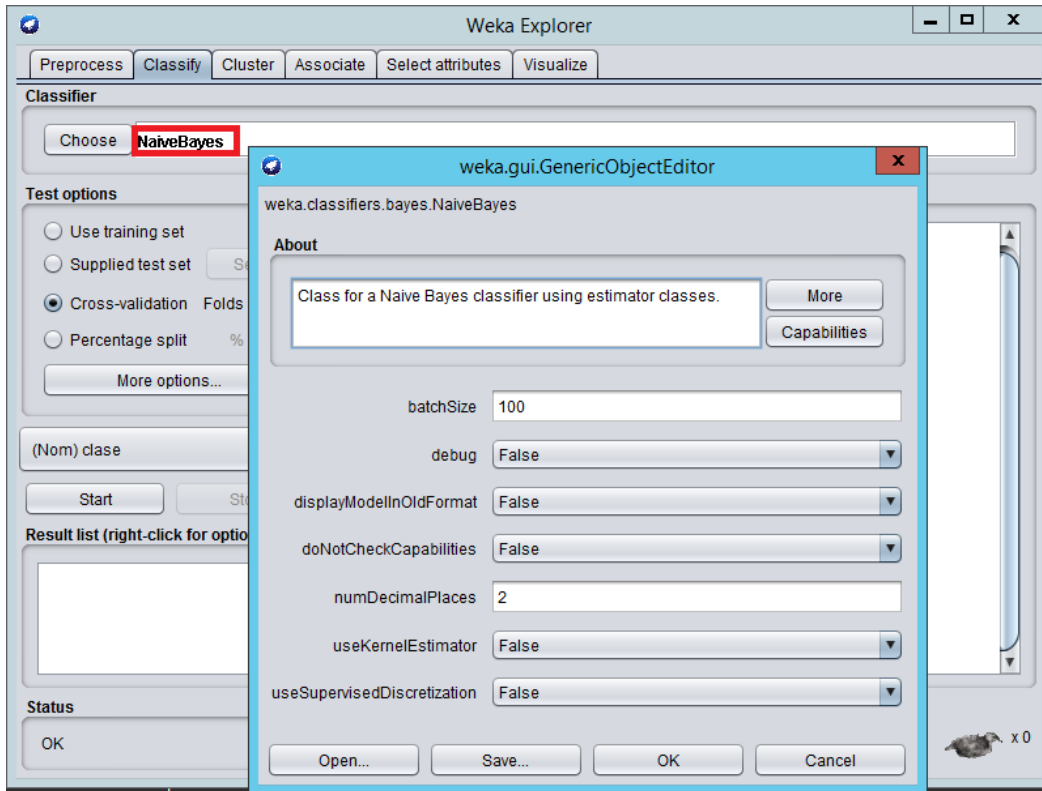


Figura 18. Verificación de valores por defecto en Weka.

Por último, se inicia la ejecución de los algoritmos al presionar *start* como se ejemplifica en la figura 19.

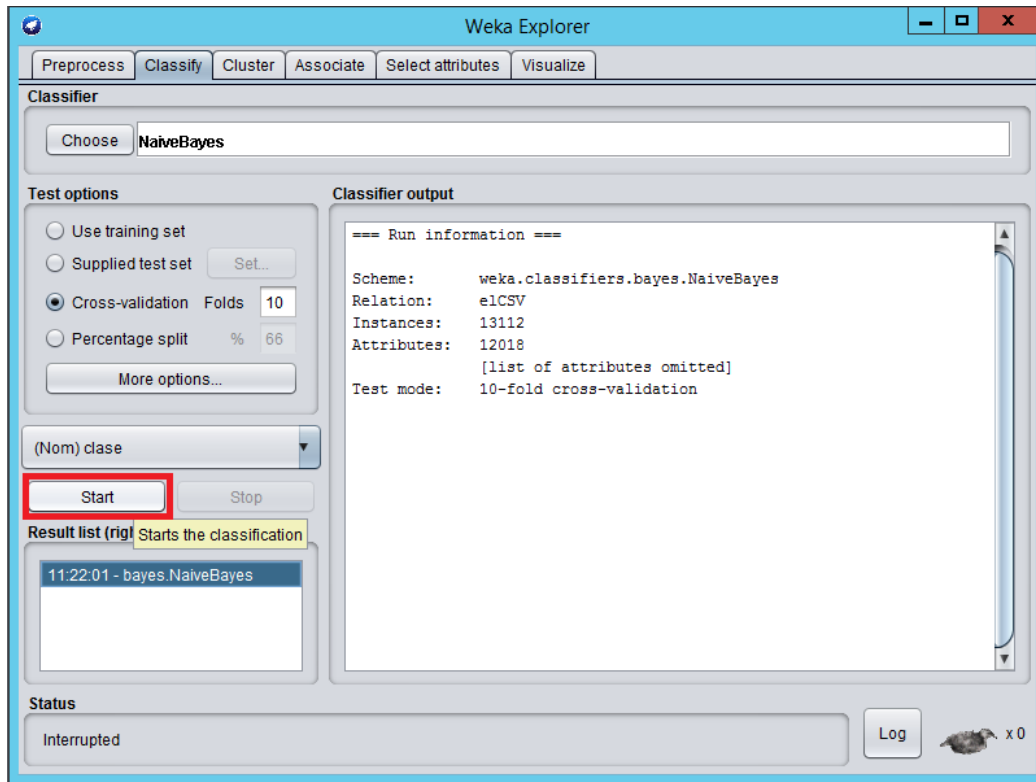


Figura 20. Inicio del proceso de minería de opiniones.

Al finalizar el proceso de clasificación se despliegan los resultados como se puede apreciar en la figura 20.

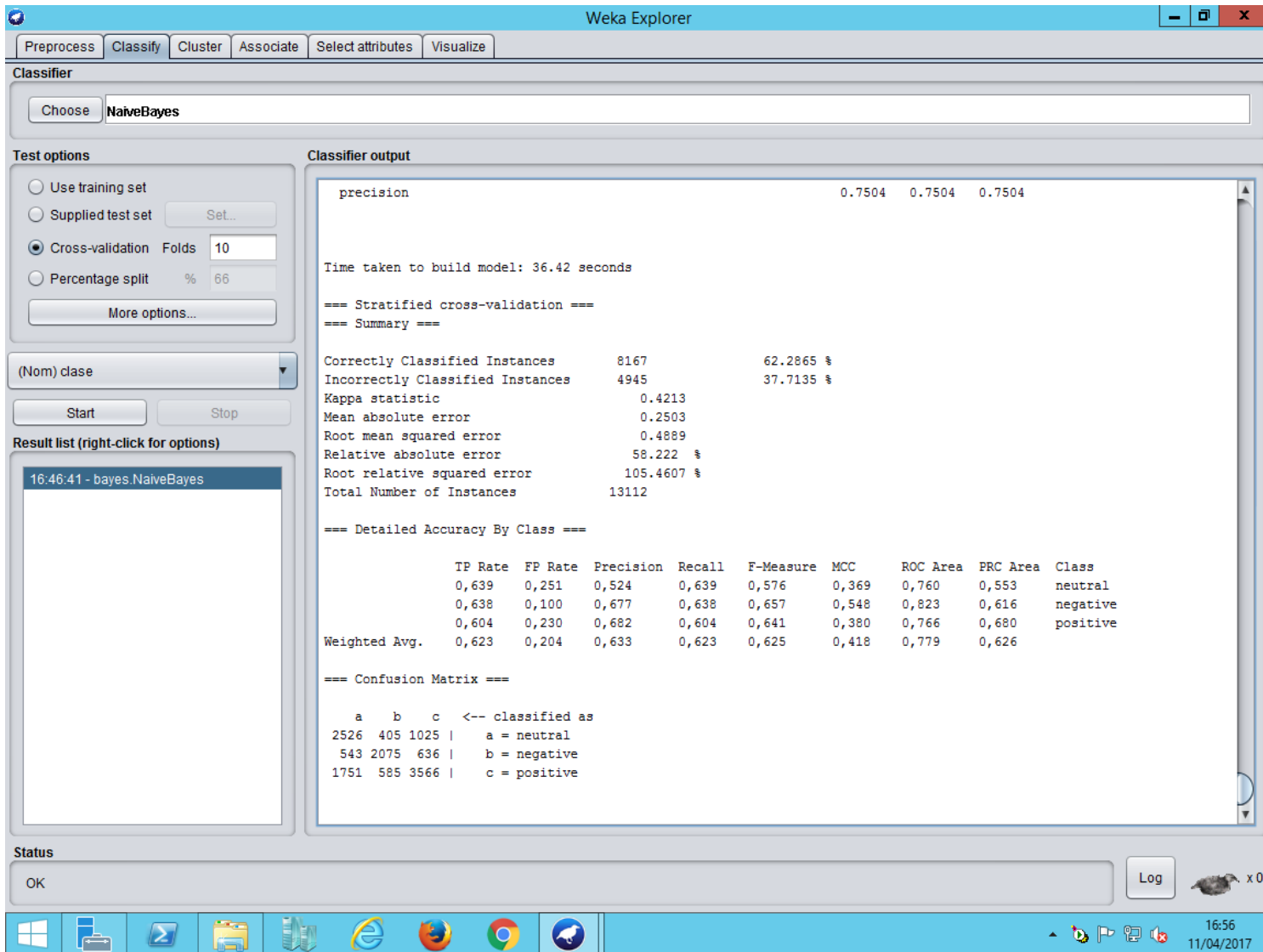


Figura 20. Resultados que se obtienen de la aplicación Weka al ejecutar el algoritmo Bayes Naïve.

Los resultados que se obtienen son un conjunto de datos que muestran el procesamiento que se le ha hecho a los atributos de las clases, estos incluyen varias métricas que son precisión, recall, f-measure, verdaderos positivos, falsos positivos, correlación del coeficiente de Matthews(MCC por sus siglas en ingles) y característica de funcionamiento del receptor(Curva de ROC por sus siglas en ingles).

7. Resultados

A continuación, se describe las métricas que utiliza la herramienta Weka para evaluar las ejecuciones de los algoritmos.

Precisión (Precision): esta es una de las métricas más importantes ya que mide los cuantos elementos fueron reconocidos correctamente respecto a los predichos sin importar si son verdaderos o falsos.

$$PPV = \frac{TP}{TP + FP}$$

Cobertura (Recall, TPR): es la métrica que de cuantifica la proporción de elementos reconocidos como correctos respecto a la cantidad total de elementos reales.

$$TPR = \frac{TP}{TP + FN}$$

Valor-F (F-measure, F_1): el valor de F-measure opera con la precisión y la exhaustividad, esta métrica es una prueba de precisión ya que su valor es alto solo cuando ambos parámetros son altos.

$$F_1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

Verdaderos positivos (TP): son los tweets que fueron clasificados como positivos negativos o neutros.

$$TPR = \frac{TPR}{TP + FN}$$

Falsos positivos (FP): son los elementos que el algoritmo clasifican como pertenecientes a una clase, pero en realidad pertenecen a otra.

$$FPR = \frac{FN}{FN + TP} = 1 - TPR$$

La matriz de confusión: contiene datos que permite evaluar visualmente los resultados de la ejecución de los algoritmos. Cada columna representa las instancias de las clases predichas y las filas representan las instancias de las clases reales. Los valores en la diagonal de la matriz son los elementos clasificados correctamente, es decir son valores de interés ya que cuando estos resultados suben y los demás bajan el algoritmo es más eficiente en términos generales.

En la figura 20 se puede apreciar el software Weka desplegando los resultados de la ejecución del algoritmo Bayes Naïve, se puede apreciar las métricas de TP rate, FP rate, Precision, Recall, F-measure entre otras y también se puede observar la matriz de confusión.

Los resultados reportados por la aplicación Weka se resumen a continuación en las siguientes tablas.

Tabla 1. Resultados de Weka del algoritmo Bayes Naïve.

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,639	0,251	0,524	0,639	0,576	0,369	0,760	0,553	neutral
	0,638	0,100	0,677	0,638	0,657	0,548	0,823	0,616	negative
	0,604	0,230	0,682	0,604	0,641	0,380	0,766	0,680	positive
Weighted Avg.	0,623	0,204	0,633	0,623	0,625	0,418	0,779	0,626	

Tabla 2. Matriz de confusión del algoritmo Bayes Naïve.

Matriz de confusión			
a	b	c	
2526	405	1025	a = neutral
543	2075	636	b = negativo
1751	585	3566	c = positivo

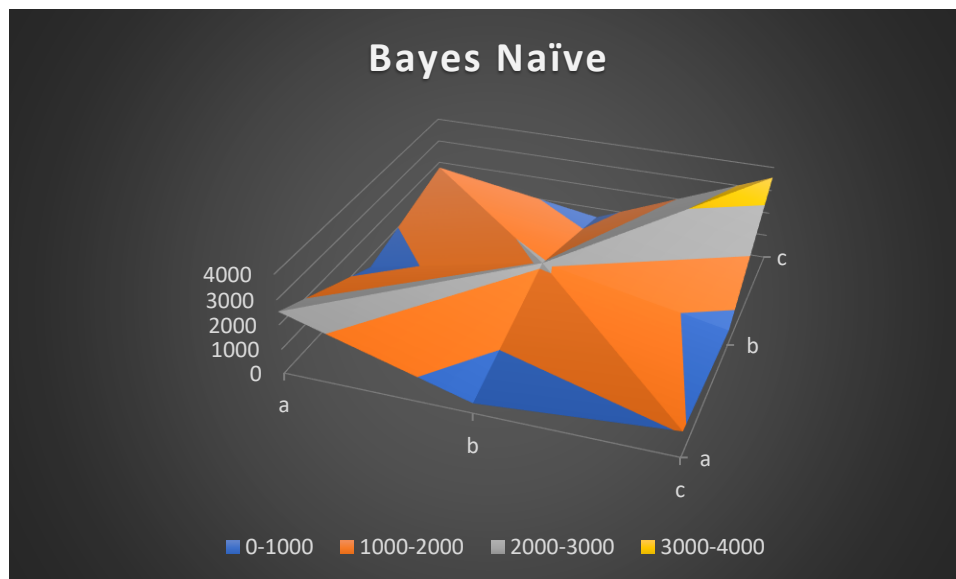


Figura 21. Gráfico demostrativo de la matriz de confusión, Algoritmo Bayes Naïve.

Tabla 3. Resultados de Weka del algoritmo J48.

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,436	0,106	0,641	0,436	0,519	0,376	0,802	0,655	neutral
	0,577	0,016	0,922	0,577	0,710	0,669	0,916	0,833	negative
	0,879	0,443	0,619	0,879	0,726	0,452	0,830	0,806	positive
Weighted Avg.	0,671	0,235	0,701	0,671	0,660	0,483	0,843	0,767	

Tabla 4. Matriz de confusión del algoritmo J48.

Matriz de confusión			
a	b	c	
2693	209	1054	a = neutral
262	2507	485	b = negativo
713	313	4876	c = positivo

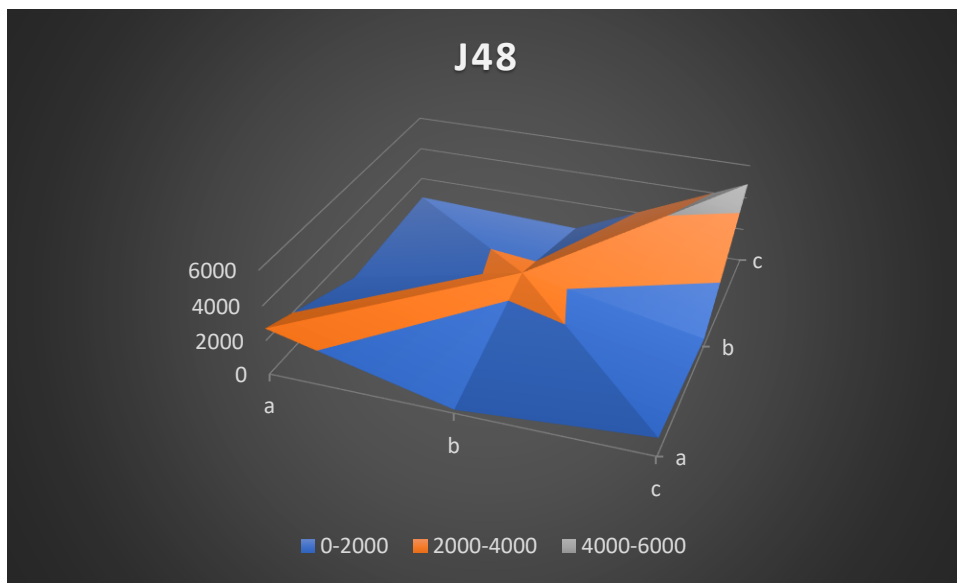


Figura 22. Gráfico demostrativo de la matriz de confusión, Algoritmo J48.

Tabla 5. Resultados de Weka del algoritmo SVM.

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,933	0,024	0,944	0,933	0,939	0,913	0,964	0,911	neutral
	0,955	0,009	0,973	0,955	0,964	0,952	0,981	0,949	negative
	0,965	0,043	0,948	0,965	0,956	0,920	0,964	0,933	positive
Weighted Avg.	0,953	0,029	0,953	0,953	0,953	0,926	0,968	0,931	

Tabla 6. Matriz de confusión del algoritmo SVM.

Matriz de confusión			
a	b	c	
3692	35	229	a = neutral
63	3107	84	b = negativo
154	52	5696	c = positivo

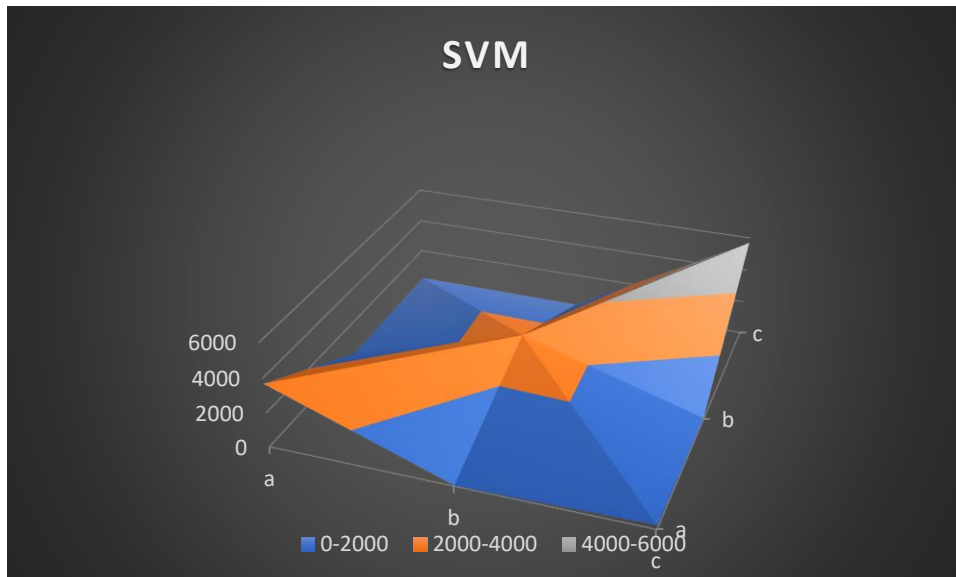


Figura 23. Gráfico demostrativo de la matriz de confusión, Algoritmo SVM.

Tabla 7. resultados de Weka del algoritmo KNN.

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,436	0,106	0,641	0,436	0,519	0,376	0,802	0,655	neutral
	0,577	0,016	0,922	0,577	0,710	0,669	0,916	0,833	negative
	0,879	0,443	0,619	0,879	0,726	0,452	0,830	0,806	positive
Weighted Avg.	0,671	0,235	0,701	0,671	0,660	0,483	0,843	0,767	

Tabla 8. Matriz de confusión del algoritmo KNN.

Matriz de confusión			
a	b	c	
1725	105	2126	a = neutral
307	1878	1069	b = negativo
659	54	5189	c = positivo

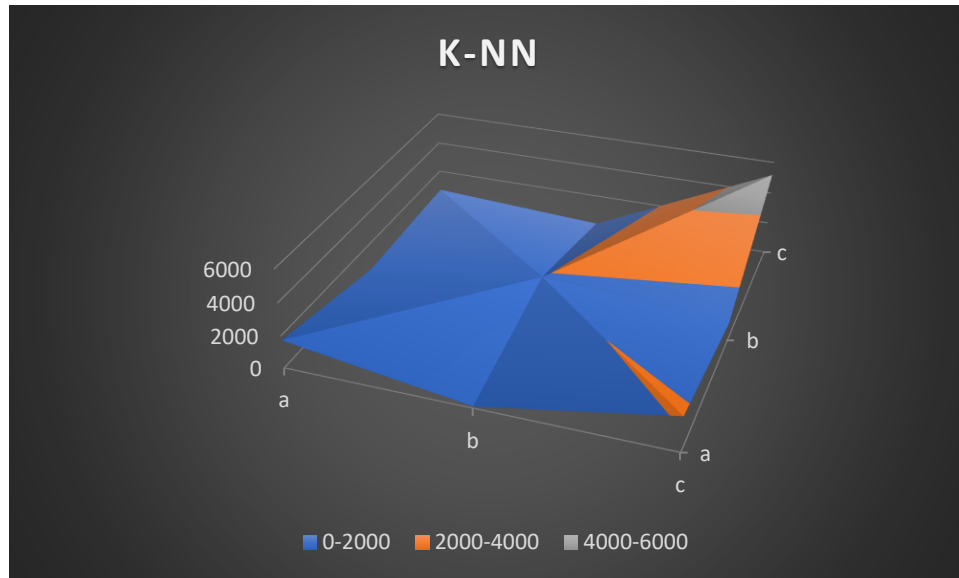


Figura 24. Gráfico demostrativo de la matriz de confusión, Algoritmo SVM.

En las gráficas de cada uno de los resultados de ejecución se puede encontrar que los casos de verdaderos positivos (VP), la diagonal con valores más altos en el espacio, denota que entre más agudo es el pico y relación los valles más bajos hacen notar mejores resultados.

8. Análisis y discusión de resultados

Los resultados obtenidos son satisfactorios y ya que los reportes obtenidos por la aplicación Weka son similares entre los diferentes algoritmos y también son similares a los resultados que se reportan en los trabajos relacionados. El proceso de definir el algoritmo con la mejor ejecución se hace evidente al observar las métricas del algoritmo SVM ya que sus resultados son por mucho mejores. SVM reporta cifras de precisión y *recall* del 95% y *f-measure* de 96%. El algoritmo con las métricas más bajas fue Bayes Naïve con precisión y *recall* del 63% y *f-measure* del 62%, sin embargo, es el algoritmo más rápido, por lo que compensa su calificación y representa una opción viable cuando se clasifica grandes volúmenes de datos. Los algoritmos J48 y K-vecinos más cercanos son intermedios en las posiciones y los tiempos de ejecución más bajos que los resultados fueron J48 precisión, *recall* y *f-measure* de 76%, K-vecinos más cercanos con precisión 71% *recall* 67% y *f-measure* 66%.

En las figuras 21 a 24 podemos observar grosso modo el comportamiento del algoritmo ya que permiten ver la relación entre la más métricas de Verdaderos Positivos (TP), Falsos Positivos (FP), Verdaderos Negativos (TF) y Falsos Negativos (FN). Como se explica anteriormente, los picos agudos en las gráficas muestran (en general) que los algoritmos son más eficientes. Tal es el caso de SVM en la figura 23, la diagonal de TP conformada por aa, bb, cc tiene valores más altos y los demás valores (FP, TF, FN) son muy bajos. Para los demás algoritmos es el caso

contrario, la diagonal se aprecia menos aguda, por lo tanto, valores menores para la diagonal de TP y valores superiores en FP, TF, FN como se puede apreciar en la figura 24.

En la figura 25 se muestra una comparativa entre los valores de las métricas de evaluación de los diferentes algoritmos que ayuda a apreciar directamente las diferencias entre estos.

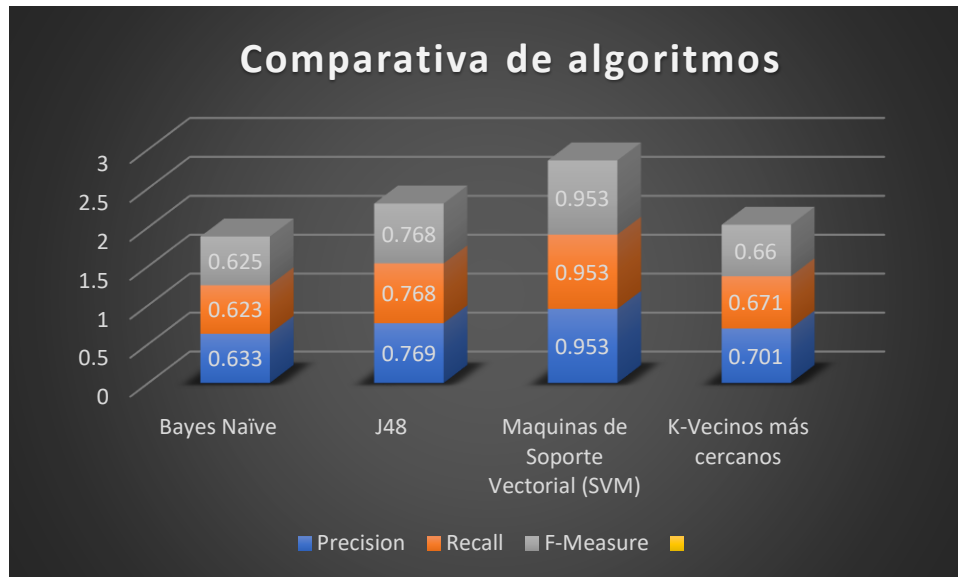


Figura 25. Grafica comparativa de los diferentes resultados de los algoritmos.

Es de interés investigar si el algoritmo Maquinas de Soporte Vectorial replica la diferencia con respecto a los demás algoritmos en las métricas al utilizar diferentes corpus y así hacer un comparativo que también permita evaluar que tan benéfico es el pre procesado realizado en este proyecto.

9. Conclusiones

Este proyecto logra realizar un esquema de procesamiento de textos en español provenientes de la red social Twitter. Se obtiene un lexicón de términos relevantes provenientes de los tweets los cuales son evaluados para obtener la relevancia de estas en relación a las etiquetas de sentimiento mediante el cálculo de frecuencias relativas TF-IDF dando información de que tanto aparecen los términos en corpus. Posteriormente se realiza minería de opiniones a partir de estos valores con cuatro algoritmos de clasificación con diferentes métodos de funcionamiento.

Durante la ejecución de los algoritmos se puede destacar las métricas de la ejecución de estos y se encuentra que el algoritmo Maquinas de Soporte Vectorial (SVM) tiene los mejores resultados reportando un promedio de instancias correctamente clasificadas de 95% con una precisión del 95% y recuerdo del 95%. Además de estas métricas está el factor tiempo que es relevante para realizar minería de datos en tiempo real, o en caso necesario, para optimizar

recursos. En estos términos el Algoritmo Bayes Naïve tiene el menor tiempo de entrega de resultados (no evaluado en el proyecto, pero si identificado al realizar las pruebas), por lo que dentro de los algoritmos estudiados este es el que representa una opción de valor para procesar grandes cantidades de datos.

Las aportaciones que tiene el proyecto son a) un esquema de para el procesamiento de textos y minería de textos, con relevancia para la minería de opiniones b) Análisis del funcionamiento de cuatro algoritmos de clasificación en un escenario de funcionamiento idéntico y configuración con valores por defecto.

Como trabajo futuro se puede mejorar el pre-procesado de los textos como incluir los emoticons que representan un estado de ánimo y debería afectar los resultados positivamente ya que son expresiones que reflejan la opinión en textos informales. El uso de palabras altisonantes también puede influir la polaridad de textos y puede ser útil considerar se al ser twitter una red social informal. Otro aspecto relevante a considerar para trabajos futuros es incluir el reconocimiento de negaciones y derivado de esto se encuentra el uso de n-gramas para generar el lexicón. Se puede incluir o cambiar los diferentes algoritmos de clasificación, por ejemplo, una red neuronal o sus variantes y comparar los resultados. Es posible utilizar partes del código para implementar un escenario de procesamiento en tiempo real.

10. Bibliografía

- [1]N. Guzmán González, "Aplicación de Distintas Técnicas de Minería de Datos para el Tratamiento de Información", Licenciatura, Universidad Autónoma Metropolitana, 2011.
- [2]A. Urquiza Pérez, "Sistema Configurable de Minería Web", Licenciatura, Universidad Autónoma Metropolitana, 2012.
- [3]J. Hernández Castillo, "Implementación de una Red Neuronal Multicapa en un FPGA", Licenciatura, Universidad Autónoma Metropolitana, 2015.
- [4]I. Peñalver Martínez, "Minería de Opiniones Basada en Características Guiada por Ontologías", Doctorado, Universidad de Murcia, 2015.
- [5]B. Barragán, "Programación Matemática para las Máquinas de Vector de Apoyo. Mathematical Programming for Support Vector Machines. - Fondos Digitalizados de la Universidad de Sevilla", *Fondosdigitales.us.es*, 2006. [Online]. Available: <http://fondosdigitales.us.es/tesis/tesis/1830/programacion-matematica-para-las-maquinas-de-vector-de-apoyo-mathematical-programming-support-vector-machines/#description>. [Accessed: 01- Nov- 2016].
- [6]Derrac Rus, J. (2013). *3. Advanced Models for Nearest Neighbor Classification Based on Soft Computing Techniques* (Doctorado). Universidad de Granada. Available: http://150.214.191.180/Documentos/tesis_dpto/171.pdf. [Accessed: 01- Nov- 2016].
- [7]T. Thejas Mol and B. Pretty, "IEEE Xplore Document - Event based sentence level interpretation of sentiment variation in twitter data", *ieeexplore.ieee.org*, 2016. [Online]. Available: <http://ieeexplore.ieee.org/document/7395176/>. [Accessed: 01- Nov- 2016].

- [8]S. Villalba-Osornio, J. Pérez-Celis, L. Villaseñor-Pineda and M. Montes-y-Gómez, "Sentiment Analysis for Reviews in Spanish: Algorithm for Handling the Negation", *Rcs.cic.ipn.mx*, 2015. [Online]. Available: http://www.rcs.cic.ipn.mx/rcs/2015_97/Sentiment%20Analysis%20for%20Reviews%20in%20Spanish_%20Algorithm%20for%20Handling%20the%20Negation.html. [Accessed: 11-Nov- 2016].
- [9]M. Thelwall, K. Buckley and G. Paltoglou, "Sentiment in Twitter events", *The ACM Digital Library*, 2011. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1952360>. [Accessed: 01-Nov- 2016].
- [10]M. Bautin, L. Vijayarenu and S. Skiena, "International Sentiment Analysis for News and Blogs", *Association for the Advancement of Artificial Intelligence*, 2016.
- [11]E. Martínez Cámara, M. Martín Valdivia, J. Perea Ortega and L. Ureña López, "Técnicas de clasificación de opiniones aplicadas a un corpus en español", *Sociedad Española para el Procesamiento del Lenguaje Natural*, no. 47, pp. 163-170, 2011.
- [12]A. Juárez-González, G. Velázquez-Villar, E. Villatoro-Tello and G. Ramírez-de-la-Rosa, "Plataforma web para la identificación y el análisis de eventos en Twitter", *Ccd.cua.uam.mx*, 2016. [Online]. Available: http://ccd.cua.uam.mx/~gramirez/bibtexbrowser.php?key=Juarez-Gonzalez2015&bib=library_ramirezdelarosa.bib. [Accessed: 09- Nov- 2016].
- [13]"Twitter.com Traffic, Demographics and Competitors - Alexa", *Alexa.com*, 2016. [Online]. Available: <http://www.alexa.com/siteinfo/twitter.com>. [Accessed: 22- Oct- 2016].
- [14] Weka 3 - Data Mining with Open Source Machine Learning Software in Java. (2016). *Cs.waikato.ac.nz*. Retrieved November 2, 2016, from <http://www.cs.waikato.ac.nz/ml/weka/>
- [15]"Naive Bayes classifier", *En.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier. [Accessed: 11- Apr- 2017].
- [16]A. Moujahid and I. Inza y Pedro, *Clasificadores K-NN*, 1st ed. Universidad del País Vasco–Euskal Herriko Unibertsitatea, 2017, pp. 1-2.
- [17]P. Vizcaino, "APLICACIÓN DE TÉCNICAS DE INDUCCIÓN DE ÁRBOLES DE DECISIÓN A PROBLEMAS DE CLASIFICACIÓN MEDIANTE EL USO DE WEKA (WAIKATO ENVIRONMENT FOR KNOWLEDGE ANALYSIS).", *FUNDACIÓN UNIVERSITARIA KONRAD LORENZ CARRERAS UNIVERSITARIAS Y POSGRADOS EN BOGOTÁ*, 2017. [Online]. Available: http://www.konradlorenz.edu.co/images/stories/suma_digital_sistemas/2009_01/final_paula_andrea.pdf. [Accessed: 11- Apr- 2017].
- [19]E. Cuevas Alfaro, "MÁQUINAS DE SOPORTE VECTORIAL CON ALGORITMOS BASADOS EN POBLACIONES PARA EL PRONÓSTICO DEL PRECIO DE ACCIONES LAN CHILE", *Licenciatura, PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO*, 2017.

ANEXO 1

Código fuente del proyecto

```
import time, codecs
import nltk
import re
import csv
import numpy
import itertools
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

rutaDatos = "C:\Users\CodeLines\Documents\UAM\Proyecto Terminal\Base de Datos"
archivoTxt = open(rutaDatos + "\Tweets3.txt", 'rb')
arregloLineas = []
raices = SnowballStemmer("spanish")
varVectorizer = CountVectorizer(min_df=1)
palabrasRelevantes = []
vector = []
strProcesado = ""
arrProcesado = []
clase = []
l = 0

class eliminaUTF:
    def __init__(self, archivo):
```

```

self.archivoTxt = archivo
encabezado = archivoTxt.read(4)
#Bucar la marca de inicio BOM para UTF-8
extensionBOM = 0
encodificado = [ ( codecs.BOM_UTF32, 4 ),
                 ( codecs.BOM_UTF16, 2 ),
                 ( codecs.BOM_UTF8, 3 ) ]

```

```

# Eliminar los Bytes BOM
for cabeza, longitud in encodificado:
    if encabezado.startswith(cabeza):
        extensionBOM = longitud
        break
archivoTxt.seek(0)
archivoTxt.read(extensionBOM)
self.readlines = archivoTxt.readlines
self.close = archivoTxt.close

```

```

class tokenStemming:
    def __init__(self, arregloLineas, clase, strProcesado, arrProcesado):
        self.arregloLineas = arregloLineas
        self.clase = clase
        self.strProcesado = strProcesado
        self.arrProcesado = arrProcesado
        for linea in arregloLineas:
            valor = len(word_tokenize(linea))
            tokens = word_tokenize(linea.decode('utf8'))
            clase.append(tokens[-1])
            del tokens[-1]
            procesado = []

```

```

muestra = ""

for i in tokens:
    temp1 = re.search(r'(ja)([ja])+',i)
    temp2 = re.search(r'(je)([je])+',i)
    temp3 = re.search(r'(ji)([ji])+',i)
    temp4 = len(i)
    if temp1:
        muestra = re.sub(r'(ja)([ja])+', "risa", i)
    elif temp2:
        muestra = re.sub(r'(je)([je])+', "risa", i)
    elif temp3:
        muestra = re.sub(r'(ji)([ji])+', "risa", i)
    elif temp4 <= 2:
        muestra = re.sub(r'\w\w?', "", i)
    else:
        muestra = re.sub(r'\d', "", i)
    temp = re.search(r'\d', i)
    if temp:
        muestra = re.sub(r'\d', "", i)
    muestra = raices.stem(muestra)
    if muestra not in stopwords.words('spanish'):
        procesado.append(muestra)
        print procesado
strProcesado = ' '.join(procesado)
arrProcesado.append(strProcesado)

```

class conteo:

```

def __init__(self, arrProcesado, varVectorizer):
    self.arrPorcesado = arrProcesado

```

```
self.varVectorizer = varVectorizer
print "Conteo de palabras"
vectorX=varVectorizer.fit_transform(arrProcesado)
```

```
class pesado:
```

```
def __init__(self,arrProcesado):
    self.arrPocesado = arrProcesado
    print "Pesado - tf-idf"
    matriz_tfidf = TfidfVectorizer(min_df=1)
    vectorY=matriz_tfidf.fit_transform(arrProcesado)
    self.toarray = vectorY.toarray
```

```
class guardarCSV:
```

```
def __init__(self,vector,vectorX,vectorY,l):
    self.vector = vector
    self.vectorX = vectorX
    self.vectorY = vectorY
    self.l = l
    print "Escribiendo matriz en archivo CSV"
    with open('E:\elCSV.csv','wb') as arch:
        i_arch = 0
        metodoEscritura1 = csv.writer(arch, delimiter=",")
        [caso.encode('utf-8') for caso in vector]
        metodoEscritura = csv.DictWriter(arch, fieldnames=vector)
        metodoEscritura1.writerow(vector)
        for documento in vectorY.toarray():
            print "fila: %d" %(i_arch)
            i_palabra = 0
            vecDiccionario = {}
            for puntuacion in documento.tolist():
```

```
    palabra = vector[i_palabra]
    vecDiccionario[vector[i_palabra]]=puntuacion
    i_palabra += 1
vecDiccionario[llave_clase]= clase[l]
metodoEscritura.writerow(vecDiccionario)
l += 1
i_arch +=1
arch.close
```

```
archivoTxt = eliminaUTF(archivoTxt)
print "Leyendo tweets"
arregloLineas = archivoTxt.readlines()
strProcesado = tokenStemming(arregloLineas,clase,strProcesado,arrProcesado)
vectorX = conteo(arrProcesado,varVectorizer)
vectorY = pesado(arrProcesado)
print "Tamaño de vocabulario: %s" % len(varVectorizer.get_feature_names())
vector=varVectorizer.get_feature_names()
llave_clase = "clase"
vector.append(llave_clase)
cambios = guardarCSV(vector,vectorX,vectorY,l)
```

ANEXO 2

Resultados del algoritmo Bayes N ive

=== Run information ===

Scheme: weka.classifiers.bayes.NaiveBayes -batch-size 1024

Relation: elCSV

Instances: 13112

Attributes: 12018

[list of attributes omitted]

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

Attribute	Class		
	neutral	negative	positive
	(0.3)	(0.25)	(0.45)
=====			
=====			
aaa			
mean	0.0004	0	0.0002
std. dev.	0.027	0.027	0.027
weight sum	3956	3254	5902
precision	0.1618	0.1618	0.1618

Time taken to build model: 32.2 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	8167	62.2865 %
Incorrectly Classified Instances	4945	37.7135 %
Kappa statistic	0.4213	
Mean absolute error	0.2503	
Root mean squared error	0.4889	
Relative absolute error	58.222 %	
Root relative squared error	105.4607 %	
Total Number of Instances	13112	

=== Detailed Accuracy By Class ===

	TP	Rate	FP	Rate	Precision	Recall	F-Measure	MCC	ROC	Area	PRC	Area	Class
	0,639	0,251	0,524	0,639	0,576	0,369	0,760	0,553					neutral
	0,638	0,100	0,677	0,638	0,657	0,548	0,823	0,616					negative
	0,604	0,230	0,682	0,604	0,641	0,380	0,766	0,680					positive
Weighted Avg.	0,623	0,204	0,633	0,623	0,625	0,418	0,779	0,626					

=== Confusion Matrix ===

Matriz de confusión			
a	b	c	
2526	405	1025	a = neutral
543	2075	636	b = negativo
1751	585	3566	c = positivo

ANEXO 3

Resultados del algoritmo J48

=== Run information ===

Scheme: weka.classifiers.lazy.IBk -K 5 -W 0 -A

"weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Relation: eICSV

Instances: 13112

Attributes: 12018

[list of attributes omitted]

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier

using 5 nearest neighbour(s) for classification

Time taken to build model: 0.05 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	8792	67.0531 %
Incorrectly Classified Instances	4320	32.9469 %
Kappa statistic	0.4614	
Mean absolute error	0.2521	
Root mean squared error	0.3789	
Relative absolute error	58.638 %	
Root relative squared error	81.7367 %	

Total Number of Instances 13112

=== Detailed Accuracy By Class ===

	TP	Rate	FP	Rate	Precision	Recall	F-Measure	MCC	ROC	Area	PRC	Area	Class
	0,436	0,106	0,641	0,436	0,519	0,376	0,802	0,655					neutral
	0,577	0,016	0,922	0,577	0,710	0,669	0,916	0,833					negative
	0,879	0,443	0,619	0,879	0,726	0,452	0,830	0,806					positive
Weighted Avg.	0,671	0,235	0,701	0,671	0,660	0,483	0,843	0,767					

=== Confusion Matrix ===

Matriz de confusión			
a	b	c	
2693	209	1054	a = neutral
262	2507	485	b = negativo
713	313	4876	c = positivo

ANEXO 4

Resultados del algoritmo Maquinas de Soporte Vectorial (SVM)

=== Run information ===

Scheme: weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K
"weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator
"weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"

Relation: elCSV

Instances: 13112

Attributes: 12018

[list of attributes omitted]

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

SMO

Kernel used:

Linear Kernel: $K(x,y) = \langle x,y \rangle$

Classifier for classes: neutral, negative

BinarySMO

Machine linear: showing attribute weights, not support vectors.

-0.4431 * (normalized) aaa

=== Detailed Accuracy By Class ===

TP	Rate	FP	Rate	Precision	Recall	F-Measure	MCC	ROC	Area	PRC	Area	Class
0,933	0,024	0,944	0,933	0,939	0,913	0,964	0,911	neutral				

	0,955	0,009	0,973	0,955	0,964	0,952	0,981	0,949	negative
	0,965	0,043	0,948	0,965	0,956	0,920	0,964	0,933	positive
Weighted Avg.	0,953	0,029	0,953	0,953	0,953	0,926	0,968	0,931	

=== Confusion Matrix ===

Matriz de confusión			
a	b	c	
3692	35	229	a = neutral
63	3107	84	b = negativo
154	52	5696	c = positivo

ANEXO 5

Resultados del algoritmo K-Vecinos más cercanos.

=== Run information ===

Scheme: weka.classifiers.lazy.IBk -K 5 -W 0 -A

"weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Relation: eICSV

Instances: 13112

Attributes: 12018

[list of attributes omitted]

Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier

using 5 nearest neighbour(s) for classification

Time taken to build model: 0.05 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	8792	67.0531 %
Incorrectly Classified Instances	4320	32.9469 %
Kappa statistic	0.4614	
Mean absolute error	0.2521	
Root mean squared error	0.3789	
Relative absolute error	58.638 %	
Root relative squared error	81.7367 %	

Total Number of Instances 13112

=== Detailed Accuracy By Class ===

	TP	Rate	FP	Rate	Precision	Recall	F-Measure	MCC	ROC	Area	PRC	Area	Class
	0,436	0,106	0,641	0,436	0,519	0,376	0,802	0,655					neutral
	0,577	0,016	0,922	0,577	0,710	0,669	0,916	0,833					negative
	0,879	0,443	0,619	0,879	0,726	0,452	0,830	0,806					positive
Weighted Avg.	0,671	0,235	0,701	0,671	0,660	0,483	0,843	0,767					

=== Confusion Matrix ===

Matriz de confusión			
a	b	c	
1725	105	2126	a = neutral
307	1878	1069	b = negativo
659	54	5189	c = positivo