Universidad Carlos III de Madrid - Grado en Ingeniería Informática

INTELIGENCIA ARTIFICIAL EN LAS ORGANIZACIONES

Práctica 2 - Minería de Datos

Curso 2021/2022

Jorge Rodríguez Fraile, Grupo 83

Franco Exequiel Schüler Allub, Grupo 83

Xu Chen, Grupo 83

Enrique Ángel Arrabal Ruiz, Grupo 83

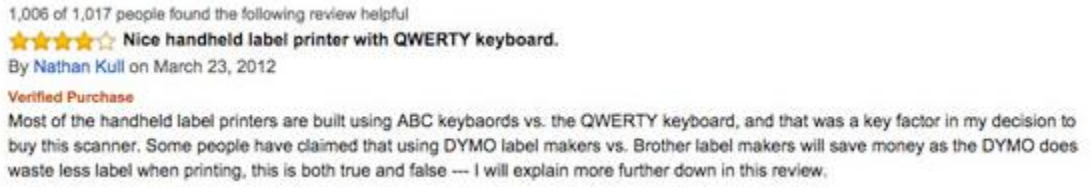
Índice

Índice de tablas

Índice de ilustraciones

1. Introducción

El objetivo principal de este proyecto es poner en práctica los conocimientos adquiridos sobre la Minería de Datos, en el que se tratará de predecir, según el comentario dejado por un usuario en Amazon [3], cuántas estrellas le pondrá al producto. Se puede ver en la *Ilustración 1.* Ejemplo de reseña de un producto de Amazon un ejemplo de cómo son los datos que recibimos:


1,006 of 1,017 people found the following review helpful

Nice handheld label printer with QWERTY keyboard.

By Nathan Kull on March 23, 2012

Verified Purchase

Most of the handheld label printers are built using ABC keybaords vs. the QWERTY keyboard, and that was a key factor in my decision to buy this scanner. Some people have claimed that using DYMO label makers vs. Brother label makers will save money as the DYMO does waste less label when printing, this is both true and false --- I will explain more further down in this review.

Ilustración 1. Ejemplo de reseña de un producto de Amazon

La información extraída de las reseñas se ha proporcionado en formato CSV, en el que se recogen 34661 instancias referentes principalmente a productos de la propia marca Amazon. Para cada reseña se muestran muchos datos además de los que se consideran para este proyecto tales como, nombre del producto, categoría del producto, marca, fecha, hora, nombre del usuario, etc.

> id,name,asins,brand,categories,keys,manufacturer,reviews.date,reviews.dateAdded,reviews.dateSeen,reviews.didPurchase,reviews.doRecommend,reviews.id,reviews.numHelpful,reviews.rating,reviews.sourceURLs,reviews.text,reviews.title,reviews.userCity,reviews.userProvince,reviews.username

Se comenzará este trabajo presentando algunos casos de uso de minería de datos en diferentes ámbitos para poder ver y entender otros enfoques que se le pueden dar a esta clase de técnicas y lo útiles que pueden llegar a ser. Lo siguiente será explicar el tratamiento de datos y filtros que se han empleado sobre los datos dados para poder aplicar esta técnica eficientemente y dejar fuera de nuestro corpus lo irrelevante.

Después se presentarán los diferentes clasificadores que se han considerado para predecir el número de estrellas (poor, fair, good, verygood y excellent), así como todas las pruebas que se han realizado y los mejores resultados obtenidos.

Los ficheros que se han empleado para la realización de este proyecto y su descripción se pueden descargar desde el siguiente enlace: IAO-P2

2. Contexto

Sentiment Analysis and Text Mining for Social Media Microblogs using Open Source Tools: An Empirical Study [1]

Actualmente, las redes sociales e Internet generan miles de millones de datos diariamente en forma de opiniones de usuario y *reviews*, lo que abre todo un abanico de posibilidades y una gran oportunidad para el Text Mining (en concreto, para el Opinion Mining). Esto es lo que trataron de aprovechar los autores de este artículo extraído de Google Académico [2].

En concreto, los autores realizaron una aproximación por la cual realizaron una extracción de datos referentes a *microblogs* de la red social Twitter, elegida por su exponencial crecimiento en los últimos años. Los datos fueron recogidos, preprocesados, analizados y visualizados haciendo uso de herramientas de text mining de código abierto. La recolección de los datos fue, según los autores, la fase más lenta y trabajosa de todo el proyecto, puesto que disponían de una cantidad de datos abrumadora que debía ser procesada y analizada. Este tipo de análisis de sentimientos o minería de opinión permite a las empresas conocer su valor competitivo en el mercado y tratar de predecir lo que sus clientes quieren a partir de sus opiniones y reseñas.

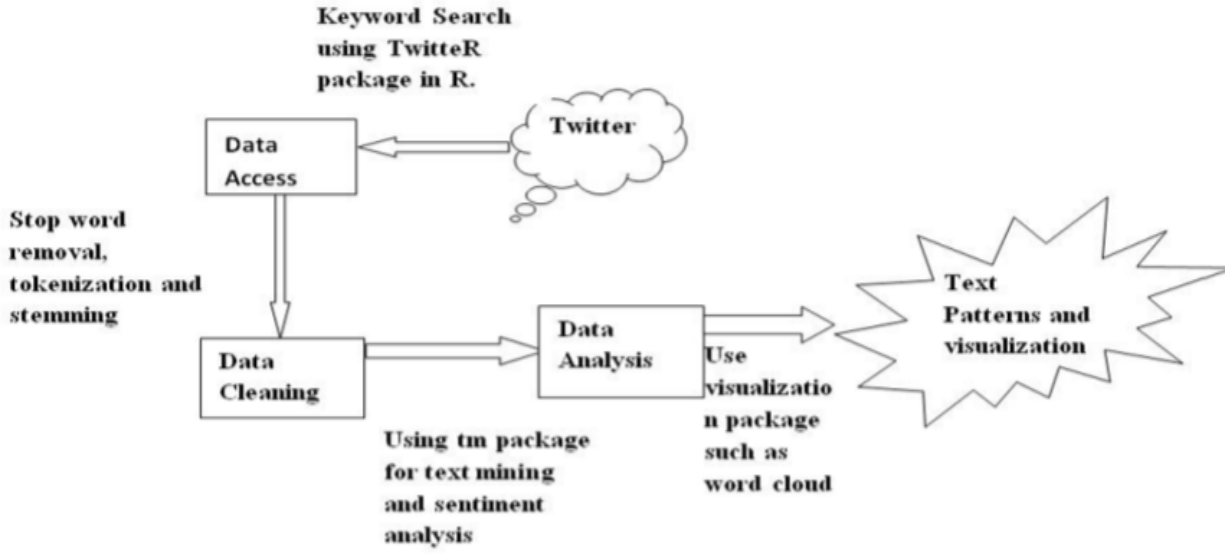


Ilustración 2. Proceso de Data Mining realizado en el artículo

Como podemos ver en la *Ilustración 2* obtenida de dicho artículo, los autores siguieron un método estandarizado de Text Mining. Primero, obtuvieron la información de Twitter utilizando lenguaje R, extrayendo las palabras clave de los Microblogs. A continuación, eliminaron las stop words o palabras que resultan irrelevantes para el proceso de análisis (preposiciones, verbos, etc.), aplicaron tokenización (obtener las palabras relevantes de los blogs) y stemming (reducir las palabras más relevantes a su raíz con el fin de generalizar a nuevas frases o palabras). Como paso previo al proceso de análisis, se llevó a cabo una limpieza de los datos, eliminando instancias incompletas, ambiguas, incoherentes, duplicadas, etc. Finalmente, hicieron uso de software de código abierto para el proceso de análisis, obteniendo así los patrones de texto y la visualización de estos.

Para los resultados, se utilizó una especie de Score, fruto de la diferencia entre la suma de palabras positivas y palabras negativas, por lo que el Score podrá ser positivo o negativo. Se utilizaron también dos empresas dedicadas a las ventas para predecir. En ambas, el Score fue cercano a cero.

How to predict explicit recommendations in online reviews using text mining and sentiment analysis [4]

En un mundo cada vez más globalizado, las opiniones de viajeros de todo el mundo resultan de gran ayuda para turistas con el objetivo de obtener información, recomendaciones, etc. y, de esta manera, reducir los riesgos de tomar malas decisiones a la hora de viajar. Este artículo obtenido de Google Académico [2] tratará de realizar recomendaciones a los usuarios basándose en opiniones previas de otros usuarios.

Las reseñas fueron analizadas haciendo uso de Text Mining basado en Procesamiento de Lenguaje Natural (NLP). Esto lo que hará será estructurar automáticamente los textos en grupos de palabras basado en el contexto y la información semántica. Como es habitual, los datos tuvieron que ser preprocesados, eliminando stop words, palabras no relevantes, números, porcentajes, etc. De esta manera, las reseñas pudieron ser clasificadas en varias categorías, las cuales pueden ser vistas en la *Ilustración 3*:

Intercept
Customer Support
Negative Attitude
Negative Budget
Negative Competence
Negative Feeling
Negative Functioning
Positive Attitude
Positive Budget
Positive Competence
Positive Feeling
Positive Functioning
WaitTime

Ilustración 3. Categorías utilizadas para clasificar las reseñas

Los resultados obtenidos por los autores utilizando distintos algoritmos y modelos fueron los siguientes:

	Probit	Binary Logistic	CHAID	C&RT	Random Forest
Accuracy	66.86%	66.86%	66.05%	66.05%	66.83%

Ilustración 4. Resultados obtenidos por los distintos clasificadores

Como se puede ver, los resultados obtenidos en la *Ilustración 4* para cada algoritmo son muy similares a la par que prometedores. Si bien un 66 % puede no parecer gran cosa, teniendo en cuenta la gran cantidad de datos utilizados y la dificultad de la tarea en cuestión, los resultados resultan bastante aceptables.

3. Parte 1

3.1. Procedimiento

3.1.1. Descripción de los datos

Antes de iniciarse en la tarea de predicción, se debe obtener la información necesaria para poder llevar a cabo dicha tarea. Los datos nos han sido proporcionados en Aula Global, pero son procedentes de la página web oficial de Amazon [3], compañía internacional de comercio electrónico.

Para este caso práctico es necesario principalmente la valoración que los usuarios proporcionan para los productos que ofrece la compañía, a los cuales serán asignadas 5 categorías diferentes dependiendo de la valoración ofrecida, siendo estas las siguientes: *Poor* (1 estrella), *Fair* (2 estrellas), *Good* (3 estrellas), *VeryGood* (4 estrellas) y *Excellent* (5 estrellas). Y, aparte de la valoración comentada anteriormente, se encontrará también, con los datos que se nos han proporcionado, con una opinión en forma de texto.

No obstante, estos datos no se podrán usar directamente en la herramienta Weka, puesto que no cumplen con el estándar de archivos aceptados por dicho programa. Para solventar ese problema, se ha llevado a cabo una transformación de los datos mediante filtrados de comentarios, el uso de Macros que se podrá encontrar en la opción de *Programador* situado en la paleta de opciones de Excel. A partir de las Macros, se obtendrá unos archivos para cada categoría de valoraciones, los cuales serán utilizados para generar en Weka mediante la opción ***Simple CLI*** el archivo de entrenamiento que se podrán usar para el caso práctico.

3.1.2. Filtros empleados

Para la realización de este trabajo es fundamental un buen preprocesado de los datos de entrada, que nos permita eliminar aquellos términos que no sea relevante o que obstaculice el proceso de clasificación que se trata de conseguir, para esto se han empleado 2 filtros que nos proporciona Weka.

StringToWordVector es el primer filtro que se aplicará a los datos, este filtro permite pasar de tener el comentario como una cadena de caracteres a tener una lista de atributos para cada reseña según los términos (palabras o partes de palabras) que aparecen en la misma con su frecuencia. Esta lista de términos/atributos compondrán el corpus de nuestro problema. Los parámetros que se deberán ajustar son:

- **IDFTransform**: Determina si la frecuencia de los términos del documento debe ser transformados.
- **TFTransform**: Determina si la frecuencia de los términos en la colección debe ser transformados.
- **lowerCaseTokens**: Pasa los tokens a minúsculas de esta manera se evita que una misma palabra aparezca varias veces.
- **minTermFreq**: Establecer cuál es el número mínimo de veces que debe aparecer un término para tenerlo en cuenta.
- **normalizeDocLength**: Normaliza la frecuencia de las palabras, de esta manera todas las frecuencias estarán a escala.
- **outputWordCounts**: Cuenta el número de palabras de salida, no solo si la palabra está presente o ausente.
- **Stemmer**: Seleccionar el algoritmo para extraer la raíz de las palabras que emplearemos.
- **stopwordsHandler**: Elimina del corpus de manera rápida una serie de palabras que son conocidas que no aportan conocimiento al problema, tales como artículos, preposiciones y ciertos verbos auxiliares.
- **Tokenizer**: El algoritmo de generación de token que utilizará.

AttributeSelection permitirá obtener una valoración de los diferentes atributos según la tarea de clasificación que se pretenderá resolver, por lo que facilita limpiar el corpus de aquellos términos menos relevantes y poderse centrar en los importantes. Para ajustar este filtro se elegirá:

- **Evaluator**: Selecciona el criterio de evaluación de los atributos.
- **Search**: Determina el método de búsqueda de los atributos siguiendo el criterio del evaluator.

3.1.3. Clasificadores probados

Random Forest: Es una combinación de un largo número de árboles de decisión individuales que operan de manera conjunta. Cada árbol predice una clase y la clase mayoritaria será la que predice el modelo. En este algoritmo se pueden definir distintos parámetros cómo la profundidad máxima o el número de ciclos, así como la semilla.

Naïve Bayes: Es un algoritmo de clasificación basado en el Teorema de Bayes, donde se utiliza una suposición de independencia entre los predictores. Es un clasificador muy utilizado para conjuntos de datos muy grandes. En este caso puede ser particularmente útil debido a que existen muchas palabras y por tanto muchas instancias.

Bagging: Es un meta-algoritmo diseñado con el objetivo de realizar combinaciones de modelos a partir de un conjunto inicial para reducir la varianza y evitar el sobreajuste. Es muy común aplicarlo con métodos basados en árboles de decisión.

3.2. Modelos

En esta sección se mostrará algunos de los modelos que hemos obtenido a lo largo de la práctica, no se incluyen todos dada la alta cantidad de pruebas que se han realizado y que han dado malos resultados, pero si los resultados de aquellos que sean relevantes para este estudio.

Se han ido probando diferentes valores para los parámetros de los filtros, pero de manera general los que mejores resultados se han dado para:

- **IDFTransform**: True.
- **TFTransform**: True.
- **lowerCaseTokens**: True.
- **normalizeDocLength**: True.
- **outputWordCounts**: True.
- **Stemmer**: Se indicará para cada modelo.
- **stopwordsHandler**: El fichero que se nos proporcionó con palabras en inglés.
- **Tokenizer**: En cada modelo se indica el usado.

El estudio inicialmente se comenzó utilizando solo el StringToWordVector y eliminando unas 20-30 palabras de las menos relevantes de AttributeSelection, pero los modelos que se obtenían con aproximadamente 2600 atributos no permitían subir la tasa de aciertos del 50 %.

Los clasificadores que se han empleado son los que se describen en la sección de clasificadores, pero tras múltiples pruebas se puede observar que los mejores resultados los proporcionaba random forest y bagging con random forest. Por este motivo la gran mayoría de modelos se han realizado con estas técnicas, aunque para cada uno de estos también se probó con Naïve Bayes y J48, pero los resultados estaban lejos de ser aceptables.

Los resultados de esta primera fase pueden verse en la *Tabla 1*:

Stemmers	Tokenizer	Clasificador	% Aciertos
Lovins	CharacterNGram	Bagging 10 de Random Forest 100	47,85%
Lovins	CharacterNGram	Random Forest 100	46,40%
Lovins	CharacterNGram	Bagging 10 de Random Forest 10	45,60%
IteratedLovins	Alphabetic	Bagging 10 de Random Forest 10	40,45%
Lovins	CharacterNGram	J48	39,95%
Lovins	CharacterNGram	PART	39,20%
Snowball	Alphabetic	Bagging de 10 de Random Forest 10 ciclos	37,70%

Tabla 1. Resultados dados por los primeros modelos de clasificación

Estos resultados todavía eran mejorables por lo que en esta nueva fase se probó a reducir significativamente el número de atributos utilizando AttributeSelection con:

- **Evaluator**: Se ha utilizado el ClassifierAttributeEval y el CorrelationAttributeEval.
- **Search**: Ranker, para obtener la lista ordenada de atributos según el evaluator.

Se redujeron el número de atributos a 1000, 500 y 60. Con esta nueva modificación se obtuvieron nuevos mejores resultados, esto se debe a que ahora sí solo se tienen en cuenta aquellos atributos que realmente son importantes para determinar su clase. Para la selección de dichos atributos se utilizaron diversas técnicas basadas en correlación y ranking.
Algunos de los resultados obtenidos son:

N. Atributos	Sel. Atributos	Stemmers	Tokenizer	Clasificador	% Aciertos
500	Correlación	Snowball	Alphabetic	Random Forest 500	55,65%
500	Correlación	Snowball	Alphabetic	Random Forest 100	54,15%
1000	Correlación	Snowball	Alphabetic	Random Forest 500	53,75%
60	Clasificación	Lovins	CharacterNGram	Bagging 10 de Random Forest 100	47,95%

Tabla 2. Resultados obtenidos por los mejores modelos de clasificación

Como se puede ver en la *Tabla 2*, hay un salto considerable con aquellos que utilizaban todos los atributos.

3.3. Resultado

El mejor modelo que se ha conseguido consiste en seleccionar los mejores 500 atributos según su correlación, obtenidos mediante AttributeSelection, utilizar el stemmer Snowball y el tokenizer Alphabetic, que coge las palabras completas. Estas instancias se pasaron por el clasificador random forest durante 500 ciclos y mediante cross validation de 10 particiones se obtuvo un porcentaje de aciertos del 55,65 %. A continuación, en la *Tabla 3* se puede ver la matriz de confusión que ha generado este modelo:

excellent	verygood	good	fair	poor	
217	109	28	25	21	**excellent**
141	**160**	43	34	22	**verygood**
4	33	**358**	4	1	**good**
32	42	29	**159**	138	**fair**
31	15	18	117	**219**	**poor**

Tabla 3. Matriz de confusión del mejor modelo de la Parte 1

Como era de esperar es mucho mejor clasificando para la clase good, dado que es el valor intermedio y es la que mejor ha aprendido a clasificar. Las otras clases que mejor ha aprendido a clasificar son los extremos excellent y poor, sin embargo los valores entre el centro y los extremos son más difíciles de distinguir para el modelo. Aparte se puede observar la tendencia de fair y verygood hacia sus extremos correspondientes, por lo que, aunque puede que el porcentaje no sea muy alto para estos, el modelo es capaz de distinguir bastante bien si la reseña será buena o mala.

En la *Tabla 4*, se puede ver la matriz de confusión como porcentajes para que se pueda entender mejor la distribución de los resultados respecto a los reales:

excellent	verygood	good	fair	poor	
54,25%	27,25%	7,00%	6,25%	5,25%	**excellent**
35,25%	**40,00%**	10,75%	8,50%	5,50%	**verygood**
1,00%	8,25%	**89,50%**	1,00%	0,25%	**good**
8,00%	10,50%	7,25%	**39,75%**	34,50%	**fair**
7,75%	3,75%	4,50%	29,25%	**54,75%**	**poor**

Tabla 4. Matriz de confusión que muestra el porcentaje de instancias predichas para cada categoría.

En general, se podrá apreciar que se obtienen porcentajes mayores en las clases que se tratan de predecir (porcentajes en negrita), aunque se sabe que podrían ser mejorables.

4. Parte 2

En esta segunda parte se pasará de un problema de clasificación a un problema de clustering. Como su nombre indica, se trata de una tarea que consiste en agrupar un conjunto de objetos (no etiquetados) en subconjuntos de objetos llamados **clusters.**

Para ello se hará uso del algoritmo K-means, con el cual, variando sus distintos parámetros, se obtendrán diez o más modelos que serán analizados.

4.1. Modelos

Los modelos más relevantes que se han generado utilizando Weka para esta segunda parte de clustering, se describen en la siguiente tabla. La codificación de los nombres de los modelos es la siguiente XXX_atributos_clustering_ZZZ:

- XXX: Número de atributos de los datos empleados para generar el modelo.
- ZZZ: Número de clústeres que se consideran para generar el modelo, se ha variado este parámetro porque en determinados modelos hay clústeres que quedan vacíos.

Modelo	Distancia	Semilla	Stemmers	Tokenizer	Algoritmo	% Aciertos
500_atributos_clustering_3	Euclídea	15	Snowball	Alphabetic	Random Forest	55,10%
500_atributos_clustering_5	Manhattan	5	Snowball	Alphabetic	Random Forest	55%
500_atributos_clustering_3	Euclídea	12	Snowball	Alphabetic	Random Forest	54,75%
500_atributos_clustering_5	Manhattan	13	Snowball	Alphabetic	Random Forest	54,65%
500_atributos_clustering_4	Euclídea	12	Snowball	Alphabetic	Random Forest	54,55%
500_atributos_clustering_5	Manhattan	5	Snowball	Alphabetic	Random Forest	54,30%
500_atributos_clustering_4	Manhattan	2	Snowball	Alphabetic	Random Forest	54,1%
500_atributos_clustering_4	Euclídea	10	Snowball	Alphabetic	Random Forest	53,70%
500_atributos_clustering_5	Manhattan	512	Snowball	Alphabetic	Random Forest	52,40%
500_atributos_clustering_3	Euclídea	10	Snowball	Alphabetic	Random Forest	52,35%
60_atributos_clustering_3	Euclídea	10	Lovins	CharacterNGram	Random Forest	46,90%
60_atributos_clustering_2	Euclídea	10	Lovins	CharacterNGram	Random Forest	46,85%

500_atributos_clustering_4	Euclídea	10	Snowball	Alphabetic	PART	43,45%
60_atributos_clustering_2	Euclídea	10	Lovins	CharacterNGram	PART	43,45%
500_atributos_clustering_3	Euclídea	10	Snowball	Alphabetic	PART	43,10%
60_atributos_clustering_3	Euclídea	10	Lovins	CharacterNGram	PART	42,75%
500_atributos_clustering_4	Euclídea	56	Snowball	Alphabetic	J48	42,70%
60_atributos_clustering_4	Euclídea	10	Lovins	CharacterNGram	PART	42,25%
500_atributos_clustering_4	Euclídea	56	Snowball	Alphabetic	PART	42,25%
500_atributos_clustering_5	Manhattan	1	Snowball	Alphabetic	J48	42,00%
500_atributos_clustering_5	Euclídea	10	Snowball	Alphabetic	PART	41,95%
500_atributos_clustering_2	Euclídea	10	Snowball	Alphabetic	J48	41,90%
60_atributos_clustering_5	Euclídea	10	Lovins	CharacterNGram	PART	41,60%
500_atributos_clustering_4	Euclídea	10	Snowball	Alphabetic	J48	41,40%
500_atributos_clustering_3	Euclídea	10	Snowball	Alphabetic	J48	41,40%
60_atributos_clustering_3	Euclídea	10	Lovins	CharacterNGram	J48	41,35%
500_atributos_clustering_5	Euclídea	10	Snowball	Alphabetic	J48	41,10%
60_atributos_clustering_5	Euclídea	10	Lovins	CharacterNGram	J48	41,10%
60_atributos_clustering_4	Euclídea	10	Lovins	CharacterNGram	J48	40,90%
60_atributos_clustering_2	Euclídea	10	Lovins	CharacterNGram	J48	40,60%
500_atributos_clustering_2	Euclídea	10	Snowball	Alphabetic	PART	40,30%

Tabla 5. Resultados dados por todos los modelos de clustering probados.

En la *Tabla 5*, se puede observar cada una de las configuraciones empleadas para tratar de conseguir el mejor modelo posible. Los resultados obtenidos no son para nada buenos, y además no son mejores que los obtenidos en la Parte 1. Por otra parte, se podrá apreciar que, en general, se obtienen mejores resultados utilizando el algoritmo PART que haciendo uso del conocido J48. Además, los mejores resultados se obtienen con el clasificador RandomForest, pues el porcentaje de instancias clasificadas correctamente es mayor que para el resto de los algoritmos.

Para poder visualizar cómo se crean los árboles de clasificación de J48 se representará el mejor modelo para este algoritmo, 500_atributos_clustering_4, que puede verse en la *Ilustración 5*:

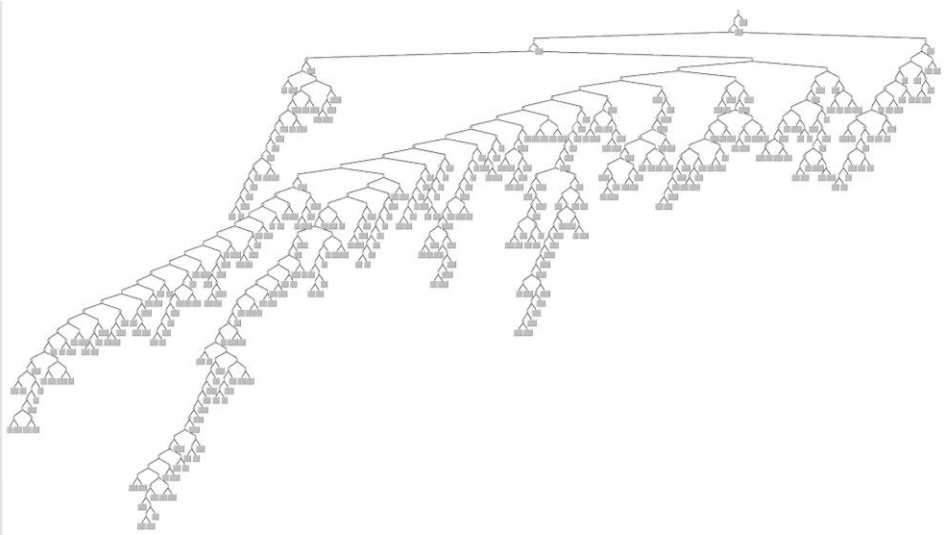


Ilustración 5. Árbol de decisión J48 para el mejor modelo de la Parte 2

Se ha representado tan pequeño debido a la gran cantidad de atributos considerados, pero se puede ver cómo según determinadas palabras va por una rama u otras hasta llegar a una hoja determinando la clase.

4.2. Resultado

En esta sección se profundizará en el mejor modelo de la tabla anterior, este modelo se corresponde a emplear el mejor conjunto de atributos de la parte anterior, los 500 mejores atributos por correlación, 3 clústeres, distancia euclídea para determinar la similitud, stemmer Snowball, tokenizer Alphabetic y después clasificar las instancias con Random Forest de 500.

Se ha realizado para 3 clústeres dado que cuando se trataba de realizar para 4 o 5 deja uno sin ninguna instancia o muy pocas, por lo que se ha reducido buscando que todos los clústeres sean relevantes y útiles para la clasificación.

excellent	verygood	good	fair	poor	
221	108	30	19	22	**excellent**
136	**161**	49	33	21	**verygood**
5	42	**349**	4	0	**good**
35	38	35	**144**	148	**fair**
31	20	18	104	**227**	**poor**

Tabla 6. Matriz de confusión para el mejor modelo de la Parte 2

Al igual que en la Parte 1, se puede ver en la *Tabla 6* que el clasificador acierta de manera casi inconfundible la clase good, puesto que es la más intermedia. Por otra parte, en los extremos se obtienen resultados pasables, si bien es cierto que en las clases cercanas a los extremos (very good y fair) el clasificador tiende a confundirse, lo cual era de esperar.

excellent	verygood	good	fair	poor	
55,25%	27,00%	7,50%	4,75%	5,50%	**excellent**
34,00%	**40,25%**	12,25%	8,25%	5,25%	**verygood**
1,25%	10,50%	**87,25%**	1,00%	0,00%	**good**
8,75%	9,50%	8,75%	**36,00%**	37,00%	**fair**
7,75%	5,00%	4,50%	26,00%	**56,75%**	**poor**

Tabla 7. Matriz de confusión que muestra el porcentaje de instancias predichas para cada categoría.

En la *Tabla 7*, se pueden apreciar las distribuciones de los aciertos en proporción. Como se ha comentado, el clasificador no obtiene los mejores resultados en los extremos, pero en la clase good se comporta bastante decente. En las clases cercanas a los extremos el clasificador se equivoca rotundamente.

5. Conclusiones y dificultades encontradas

Por último, realizaremos en este apartado una reflexión sobre los resultados obtenidos y una serie de dificultades encontradas.

Una de las mayores dificultades se ha encontrado en la Parte 1 de la práctica ha sido lograr modelos con mejores resultados. En general, se han obtenido resultados en torno al 55 %, lo cual no es para nada atractivo. Las razones por las que se cree que se debe esto es una mala selección de atributos, la gran cantidad de estos o el ruido en los datos. La selección de atributos que hemos empleado puede que no haya sido lo suficientemente significativa y relevante para las predicciones del modelo. Por otra parte, inicialmente disponíamos de unos 3000 atributos, por lo que lograr un modelo mejor se tradujo en reducir significativamente esa gran cantidad de información.

En la Parte 2 ocurrió más de lo mismo. No solo no se logró encontrar modelos mejores que los obtenidos en la parte uno, sino además empeoraron. Esto puede deberse a las mismas razones que las ya expuestas.

A pesar de las dificultades encontradas, hemos aprendido una gran cantidad de técnicas gracias a este proyecto. Hemos aplicado una serie de técnicas de minería de texto y la utilización de diversos filtros y algoritmos que a menudo se utilizan en este ámbito. En general, ha sido una práctica muy enriquecedora.

6. Referencias

[1] Younis, E. M. (2015). Sentiment analysis and text mining for social media microblogs using open source tools: an empirical study. International Journal of Computer Applications, 112(5).
[2] Google Académico [en línea] Acceso: 07/10/2021. https://scholar.google.es/schhp?hl=es

[3] Amazon.com [en línea] Acceso: 07/10/2021. https://www.amazon.com

[4] Guerreiro, J., & Rita, P. (2020). How to predict explicit recommendations in online reviews using text mining and sentiment analysis. Journal of Hospitality and Tourism Management, 43, 269-272.