UNIVERSIDAD DE COSTA RICA
SISTEMA DE ESTUDIOS DE POSGRADO

AGRUPAMIENTO Y PREDICCIÓN AUTOMÁTICA DE RETROALIMENTACIÓN DE CLIENTES: UNA APLICACIÓN PARA EL SERVICIO WEB HULIPRACTICE

Trabajo final de investigación aplicada sometido a la consideración de la Comisión del Programa de Estudios de Posgrado en Estadística para optar al grado y título de Maestría Profesional en Estadística

FABIÁN HERNÁNDEZ ARRIETA

Ciudad Universitaria Rodrigo Facio, Costa Rica
2021

DEDICATORIA

A mi madre y padre, por haberme apoyado a lo largo de los años a alcanzar mis metas. Sin su esfuerzo y dedicación me hubiera sido imposible seguir adelante. A mi padre por enseñarme el valor del trabajo y a mi madre por enseñarme el valor de la humildad.
A mi esposa, por haberme apoyado a largo de los cinco años que tomó el proceso de esta maestría. Su paciencia y comprensión fueron invaluables durante estos años y me permitieron llegar al final de este proceso con éxito.

AGRADECIMIENTOS

Quiero agradecer a mi profesor tutor, Óscar Centeno Mora, por ayudarme tan diligentemente en las revisiones de este proyecto y por guiarme con recomendaciones a través de este proceso.
También les agradezco a los lectores de este proyecto, Juan Felipe Gonzáles Évora y Marcela Alfaro Córdoba, por sus valiosos consejos y por tomar el tiempo para las revisiones del documento final.

"Este trabajo final de investigación aplicada fue aceptado por la Comisión del Programa de Estudios de Posgrado en Estadística de la Universidad de Costa Rica, como requisito parcial para optar al grado y título de Maestría Profesional en Estadística"

OSCAR JAVIER CENTENO MORA (FIRMA)

Firmado digitalmente por OSCAR JAVIER CENTENO MORA (FIRMA)
Fecha: 2021.07.02 17:08:43 -06'00'

M. Sc. Óscar Centeno Mora
Profesor guía

MARCELA ALFARO CORDOBA (FIRMA)

Digitally signed by MARCELA ALFARO CORDOBA (FIRMA)
Date: 2021.07.05 13:19:47 -07'00'

Dra. Marcela Alfaro Córdoba
Lectora

JUAN FELIPE GONZALEZ EVORA (FIRMA)

Firmado digitalmente por JUAN FELIPE GONZALEZ EVORA (FIRMA)
Fecha: 2021.07.04 19:14:08 -05'00'

M. Sc. Juan Felipe Gonzáles Évora
Lector

FABIAN GERARDO HERNANDEZ ARRIETA (FIRMA)

Firmado digitalmente por FABIAN GERARDO HERNANDEZ ARRIETA (FIRMA)
Fecha: 2021.06.30 20:45:31 -06'00'

Fabián Hernández Arrieta
Sustentante

TABLA DE CONTENIDO

Resumen

Para toda compañía, el cliente siempre debe estar en el centro de su accionar. Para productos de software como HuliPractice, es de suma importancia incorporar la retroalimentación de sus clientes como acciones de mejora. Esta retroalimentación responde a necesidades variadas de muchos clientes, por lo que su procesamiento se puede convertir en una tarea tediosa y demandante de tiempo. La minería de texto ofrece una alternativa para procesar automáticamente esta retroalimentación y, con ello, lograr reducir el tiempo de procesamiento. El presente trabajo aplica y valida técnicas de minería de texto para la agrupación y predicción automática de 1519 comentarios de clientes sobre el producto HuliPractice. Esta retroalimentación viene dada en forma de comentarios, los cuales fueron captados por los equipos internos de la compañía Huli, entre octubre de 2016 y setiembre de 2020.

Primeramente, se realiza un análisis descriptivo de la retroalimentación existente utilizando las técnicas de nube de palabras y análisis de redes. Como resultado, se encuentra que las necesidades de los clientes apuntan a funcionalidades como, el soporte a *Google Calendar* dentro de la aplicación y la implementación de la *facturación electrónica* desde el celular.

En segunda instancia, se comparan dos técnicas no supervisadas para agrupar automáticamente esta retroalimentación: el modelo de k-medias y el modelo DBScan. Con el modelo de k-medias, se obtienen los resultados de agrupación más homogéneos y completos. Las características de homogeneidad y completitud de los grupos obtenidos son sintetizados a través del valor v. Entre más cercano este valor a 1, más completos y homogéneos son los grupos de comentarios generados. El modelo de k-medias obtiene el valor v más alto con 0,367. El modelo de DBScan obtiene un valor v de 0,30.

Finalmente, se comparan cuatro técnicas de análisis supervisado para clasificar automáticamente la retroalimentación de clientes: el modelo ingenuo de Bayes(NB), bosques aleatorios(RF), máquinas de soporte vectorial (SVM) y el modelo de potenciación de gradiente extrema(XGBoost).

El modelo de SVM tiene los mejores resultados con una precisión media de 81,0 %. El modelo XGBoost produjo segundos mejores resultados de precisión media con un 79,7 %. Los modelos de NB y RF ocupan el tercer y cuarto lugar en cuanto a resultados obtenidos, con una precisión media de 79,0 % y 77, 5 %, respectivamente.

La precisión media obtenida por el modelo SVM (81.0 %) concuerda con los resultados reportados en proyectos similares de ingeniería de requerimientos basados en grupos. También, la implementación de este proceso representaría hasta 20 horas de ahorro por mes en tareas de clasificación de nuevos comentarios. Además, el uso del algoritmo k-medias para agrupar comentarios no clasificados por el modelo supervisado, ahorraría hasta 4 horas al mes a la compañía. Por lo anterior, se concluye como satisfactorios los resultados de este trabajo según los objetivos planteados.

Palabras Claves — CrowdRE, análisis de redes, nube de palabras, bolsa de palabras, TF-IDF, reducción de dimensión, chi-cuadrado, minería de texto, análisis supervisado, NB, árboles aleatorios, SVM, XGBoost, análisis no supervisado, k-medias, DBScan .

Abstract

The client must be at the center of all company product decisions. For a software product as HuliPractice, it is vital to incorporate client feedback insights as product improvements. This feedback comes from a lot of clients with different necessities, so its processing could become a very demanding and tedious task. Text mining provides an alternative solution by automatically processing this information, and reducing the time invested in such tasks. This project applies text mining techniques to classify and cluster 1519 HuliPractice clients' feedback. This feedback was gathered by Huli internal departments between October 2016 and September 2020.

Firstly, a descriptive analysis using word clouds and network analysis is applied to the feedback. This analysis showed that clients are inclined towards the implementation of a Google Calendar integration with the HuliPractice agenda, and the necessity of adapting the billing module for mobile access.

Secondly, 2 techniques from the unsupervised learning area are compared to automatically cluster this client feedback. These techniques are k-means and DBScan. K-means yields more complete and homogeneous clusters. Homogeneity and completeness are synthesized through v value. This value varies from 0 to 1. The closer to 1 are the results, the more homogeneous and complete are the generated groups. K-means gets a v measure of 0.367 whereas DBScan yields a v measure of 0.30.

Finally, 4 techniques from the supervised analysis area are compared to automatically classify this information. These techniques are naive Bayes (NB), random forest RF, support vector machines (SVM) and extrema gradient boosting (XGBoost). Support vector machines model gives the best mean precision results with 81.0%. XGBoost model had the second-best precision results with a mean of 79.7%. NB and RF got the third and fourth place with a precision mean of 79.0% and 77.5% respectively.

The mean precision yield by the SVM model (81.0%) is similar to the reported values in other crowd-based requirement engineering projects. Furthermore, the implementation of this automatic process to classify new client feedback could represent a monthly 20 hour saving for the company. On top of that, using the k-mean model to cluster non-classified comments brings another monthly 4 hours saving for the organization. Therefore, it is concluded as satisfactory the results according to the project's objectives.

keywords — CrowdRE, network analysis, words cloud, bag of words, TF-IDF, dimension reduction, chi-square, text mining, supervise analysis, NB, random forest, SVM, XGBoost, unsupervised analysis, k-means, DBScan .

Lista de cuadros

Lista de figuras

Lista de abreviaturas

AWS Del ingles, Amazon Web Servirces (Servicios Web de Amazon.). 10, 21, 31

BOW Del ingles, Bag of Words (Bolsa de palabras.). v, 8, 14, 18, 38, 40–48

CRM Del ingles, Customer Relationship Management(Software para gestión de clientes). ix, 1, 3, 6, 10, 32, 33

CrowdRE Del ingles, Crowd-base Requirement Engineering (Ingeniería de requerimientos basado en grupo). vii, viii, 1, 5, 7, 8, 21, 30

DBScan Del ingles, Density-based spatial clustering of applications with noise (Agrupación espacial basada en densidad de aplicaciones con ruido.). v–viii, 17, 18, 38, 39, 47

EC2 Del ingles, Elastic Computing Cloud (Computación Elástica en la Nube.). 10

ETL Del ingles, Extraction, Transformation, and Load (Extracción, transformación y Carga.). 10

IEEE Del ingles, Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos). 5

NB Del ingles, Naive Bayes (Bayes Ingenuo.). v–ix, 21, 40, 41, 46, 48

RF Del ingles, Random Forest (Bosques aleatorios.). v–ix, 24, 26, 41–43, 46, 48

SVM Del ingles, Support Vector Machines (Maquinas de Soporte Vectorial.). v–ix, 22, 23, 43, 44, 46–48

TF-IDF Del ingles, term frequency-inverse document frequency (Frecuencia del término - inversa de la frecuencia del documento.). v, vii, viii, 8, 14, 15, 18, 38, 40, 41, 43, 44, 48

XGBoost Del ingles, Extreme Gradient Boosting (Potenciación de gradiente extrema.). v–ix, 26, 44–46

UNIVERSIDAD DE COSTA RICA

SEP Sistema de Estudios de Posgrado

Autorización para digitalización y comunicación pública de Trabajos Finales de Graduación del Sistema de Estudios de Posgrado en el Repositorio Institucional de la Universidad de Costa Rica.

Yo, Fabián Gerardo Hernández Arrieta, con cédula de identidad 114640816, en mi condición de autor del TFG titulado Agrupamiento y predicción automática de retroalimentación de clientes: una aplicación para el servicio web HuliPractice

Autorizo a la Universidad de Costa Rica para digitalizar y hacer divulgación pública de forma gratuita de dicho TFG a través del Repositorio Institucional u otro medio electrónico, para ser puesto a disposición del público según lo que establezca el Sistema de Estudios de Posgrado. **SI** [X] **NO** *[ ]

***En caso de la negativa favor indicar el tiempo de restricción: _____ año (s).**

Este Trabajo Final de Graduación será publicado en formato PDF, o en el formato que en el momento se establezca, de tal forma que el acceso al mismo sea libre, con el fin de permitir la consulta e impresión, pero no su modificación.

Manifiesto que mi Trabajo Final de Graduación fue debidamente subido al sistema digital Kerwá y su contenido corresponde al documento original que sirvió para la obtención de mi título, y que su información no infringe ni violenta ningún derecho a terceros. El TFG además cuenta con el visto bueno de mi Director (a) de Tesis o Tutor (a) y cumplió con lo establecido en la revisión del Formato por parte del Sistema de Estudios de Posgrado.

INFORMACIÓN DEL ESTUDIANTE:

Nombre Completo: Fabián Gerardo Hernández Arrieta.

Número de Carné: B58415 Número de cédula: 114640816.

Correo Electrónico: FABIAN.HERNANDEZARRIETA@ucr.ac.cr.

Fecha: 2021-06-29. Número de teléfono: 83097721.

Nombre del Director (a) de Tesis o Tutor (a): M. Sc. Óscar Centeno Mora.

FABIAN GERARDO HERNANDEZ ARRIETA (FIRMA)

Firmado digitalmente por FABIAN GERARDO HERNANDEZ ARRIETA (FIRMA)
Fecha: 2021.06.30 20:49:58 -06'00'

FIRMA ESTUDIANTE

Nota: El presente documento constituye una declaración jurada, cuyos alcances aseguran a la Universidad, que su contenido sea tomado como cierto. Su importancia radica en que permite abreviar procedimientos administrativos, y al mismo tiempo genera una responsabilidad legal para que quien declare contrario a la verdad de lo que manifiesta, puede como consecuencia, enfrentar un proceso penal por delito de perjurio, tipificado en el artículo 318 de nuestro Código Penal. Lo anterior implica que el estudiante se vea forzado a realizar su mayor esfuerzo para que no sólo incluya información veraz en la Licencia de Publicación, sino que también realice diligentemente la gestión de subir el documento correcto en la plataforma digital Kerwá.

1. CAPÍTULO I: INTRODUCCIÓN

1.1. Contexto

Los productos de software son consumidos por personas que tienen gustos y necesidades distintas, y que viven en diferentes zonas geográficas. Esto ocasiona que los requerimientos entre los distintos grupos de consumidores (aquellos que presentan características en común, como edades o gustos similares) puedan resultar bastante heterogéneos[Eduard C et al. (2017)]. El éxito de cualquier producto de software va a depender en gran medida de la capacidad que tenga el desarrollador del producto a nivel interno para identificar, procesar y transformar las necesidades de los consumidores en mejoras. A esta metodología que busca transformar requerimientos de clientes en acciones dentro de la compañía se le conoce como *requerimientos de ingeniería basada en multitudes(CrowdRE)*[Eduard C et al. (2017)].

En el contexto de productos de software consumidos vía web, muchas veces los requerimientos tienen que ser extraídos de la retroalimentación que los mismos clientes proveen. Esta retroalimentación suele ser compartida en forma de comentarios relativos hacia alguna funcionalidad de un producto en particular[UserVoice. (2015)].

UserVoice[1], una compañía especializada en retroalimentación de clientes, realizó una encuesta entre 300 administradores de producto en 2015, dentro de Estados Unidos. En esta encuesta encontró que, a pesar de contar con la información suministrada como comentario de clientes, solo un 20 % de las compañías indicó que usa esos datos en la toma de decisiones; el 80 % solo los almacena.[UserVoice. (2015)].

También, debido a que esta retroalimentación es proporcionada por los clientes de una manera constante, la cantidad de información por procesar puede crecer rápidamente, volviendo inviable extraer requerimientos de forma manual.[Jacqueline S. (2018)].

La minería de texto surge ante la necesidad de procesar y generar conocimiento dentro de estos grandes volúmenes de información no estructurada[2]. Se busca aplicar técnicas de análisis de información para generar conocimiento que pueda ser de utilidad en la toma de decisiones[Gurusamy et al. (2014)].

Huli[3] es un emprendimiento costarricense fundando en Costa Rica en el 2012. Cuenta con más de 6000 clientes en América Central y México. Su visión es comunicar a los actores del entorno de la salud a través de la tecnología y la innovación. Uno de los principales productos de software de la compañía es la página web HuliPractice[4]. Esta consiste en una aplicación web donde médicos y asistentes de médicos pueden interactuar; además, pueden manejar labores diarias dentro de sus centros de salud, tales como calendario de citas e información médica de pacientes.

La compañía cuenta con equipos de trabajo encargados de la constante comunicación con los clientes. Esto genera una cantidad importante de retroalimentación, que es canalizada a través del CRM oficial de la compañía.

Hasta la fecha, los comentarios han sido procesados manualmente por los equipos de priorización de tareas y diseño de experiencias. Pero, debido al crecimiento de la compañía en los últimos años, este procesamiento manual ha dejado de ser viable.

[1] Accedido en la web desde https://uservoice.com/

[2] El concepto de *información no estructurada* se refiere a datos contenidos sin estructura de almacenamiento

[3] Del hawaiano 'dar vuelta, cambiar de posición'.

[4] Accedido en la web desde https://hulipractice.com

El presente trabajo utiliza técnicas de minería de texto aplicadas a la retroalimentación de clientes para el producto HuliPractice. Se busca facilitar el procesamiento de esta información dentro de la compañía, con el fin de reducir la carga de trabajo asociada a esta labor.

1.2. Objetivos de la investigación

1.2.1. Objetivo general

- Implementar métodos de minería de texto en la agrupación y predicción automática de retroalimentación de clientes para el servicio web HuliPractice.

1.2.2. Objetivos específicos

- Realizar un análisis descriptivo de la retroalimentación de clientes disponible.
- Utilizar modelos no supervisados para agrupar de manera automática la retroalimentación de clientes existente.
- Utilizar modelos supervisados multi-clase para clasificar de manera automática nuevos comentarios de clientes.
- Seleccionar entre los modelos supervisados y no supervisados los que generen los mejores resultados para el desarrollo de una aplicación web que procese automáticamente esta retroalimentación.

1.3. Justificación

Para toda compañía, el cliente siempre debe estar en el centro de toda decisión. Sam Walton, el multimillonario que fundó la cadena de supermercados Wallmart, dijo una vez:

> *¨Existe solo un jefe. El cliente. El cliente puede despedir a cada persona dentro de la compañía, inclusive el presidente, solo necesita escoger gastar su dinero en algún otro lugar¨. Sam Walton, (1982)*

Walton sabía que el cliente tiene el poder de dictar si un producto va a ser exitoso o no. Por lo tanto, la retroalimentación de un cliente se convierte en la fuerza principal detrás del desarrollo de cualquier producto. Esta información debería ser usada para validar ideas, priorizar iniciativas y como fuente de oportunidades para innovar.[Lowdermilk et al. (2017)]

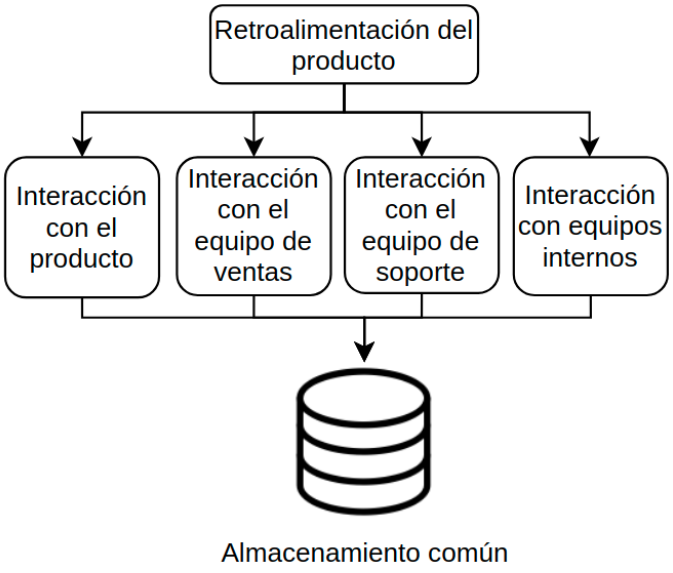
El problema al que nos enfrentamos en esta investigación está en cómo generar un proceso repetible y escalable que permita usar esta retroalimentación de una manera constante.

Huli es una compañía que se enfoca en mejorar la comunicación entre doctores y pacientes en el sector salud. Por su parte, HuliPractice es un producto web de la compañía, donde doctores pueden administrar su clínica. Para Huli, los doctores que usan el producto son los clientes. La retroalimentación que estos brindan viene dada en forma de comentarios sobre las diferentes funcionalidades del producto. Esta retroalimentación es capturada por diferentes fuentes, a saber:

- Durante la interacción del cliente con el producto. Por ejemplo, cuando el usuario está utilizando el producto, le aparece un mensaje que invita a ofrecer retroalimentación.
- Mediante la interacción del cliente con el equipo de ventas. Normalmente, en los comentarios que surgen en este tipo de intercambio se emplean frases como *los clientes están pidiendo X o si tuviéramos X funcionalidad, podría firmar a este cliente.*
- Durante la interacción del cliente con el equipo de soporte técnico del producto. En esta retroalimentación se suelen emplear frases como *el cliente tiene problemas a la hora de X o si tuviéramos X funcionalidad, el cliente podría X.*
- Mediante la interacción del cliente con equipos internos de la compañía, como el equipo de desarrollo del producto o el equipo de manejo de operaciones de la compañía.

Históricamente, los diferentes equipos de la compañía que recibían esta retroalimentación la almacenaban de forma separada. Por ejemplo, el equipo de ventas almacenaba la información en un documento de Microsoft Excel[5], mientras que el equipo de soporte al cliente la almacenaba en un documento de Google Sheet[6]. La retroalimentación por interacción directa con el cliente era almacenada en la base de datos de la aplicación. En 2016, se hizo un esfuerzo por centralizar esta información a través del CRM oficial de la compañía. En la figura 1 muestra cómo, luego del cambio realizado en 2016, la retroalimentación pasó a almacenarse en una fuente común. En lugar de que cada equipo almacenara la retroalimentación recolectada en fuentes separadas, se centralizó el proceso para recolectarlo dentro del CRM de la compañía, y así, almacenarlo en una fuente común.

Figura 1: Fuentes de retroalimentación de productos



Fuente: Elaboración propia.

Esta nueva metodología presentaba las siguientes mejoras:

[5]Producto de la compañía Microsoft usado para facilitar labores administrativas en compañías.
[6]Producto de la compañía Google utilizado para facilitar labores administrativas en compañías.

- Se contaba con más información sobre el cliente: país, funciones del producto que tenía habilitado, etc.
- Se podía registrar para un mismo cliente toda la retroalimentación que había dejado en diferentes ocasiones, lo que permitía identificar al cliente más fácilmente.

Sin embargo, el mismo proceso presentaba ahora nuevos retos:

- Al estar almacenándose esta información de una manera centralizada, el volumen de comentarios por ser analizados aumentó, al igual que el tiempo para poder procesar la información. Entonces, nace la pregunta sobre ¿cómo procesar esta información de una manera eficiente?
- Esta retroalimentación se recibía, y aún se recibe, como texto, y cada comentario es emitido como retroalimentación para una parte particular del producto. Esto plantea el siguiente problema: *¿cómo se podrían clasificar automáticamente estos comentarios para ser analizados por separado?*

El presente trabajo aborda estos problemas desde el área de la minería de texto. Se aplican técnicas para la descripción, agrupación y predicción automática de la retroalimentación para HuliPractice. Con ello se busca facilitar el procesamiento de esta información, a fin de aportar un valor agregado, según lo comentado por el cliente.

2. CAPÍTULO II: ESTADO DE LA CUESTIÓN

Este capítulo aborda el estado de la cuestión, en primer lugar, sobre el tema de ingeniería de requerimientos basada en multitudes, y, en segundo lugar, sobre la de minería de texto, desde el enfoque supervisado y no supervisado de la información.

2.1. Ingeniería de requerimientos basada en multitudes

En 2015, Grown, Dorr y Adam adoptaron por primera vez el términó de *ingeniería de requerimientos basada en multitudes*, proceso conocido como CrowdRE.[M. Glinz. (2019)] Este concepto es usado para referirse al proceso de obtener requerimientos para mejorar productos, el cual se basa en la retroalimentación de clientes.[Groen et al. (2015)].

Los productos comercializados vía web son consumidos por una multitud de clientes, localizados en distintas zonas geográficas. Estos clientes suelen tener diferentes necesidades relacionadas con el producto. La capacidad que demuestre la compañía para transformar estas necesidades en accionables de la compañía juega un papel muy importante en el éxito de la misma.[Koch. (2018)]

Según [Mahmood H et at 2017], se pueden identificar cuatro pilares necesarios para generar un proceso de requerimientos basados en multitudes de clientes:

- La multitud. La identificación de las personas cuyas necesidades se quieren conocer e incorporar como requerimientos a la compañía.
- El recolector. El ente que quiere conocer las necesidades de la multitud.
- La tarea de recolección. La labor ejecutada por el recolector para lograr obtener la información sobre las necesidades de la multitud.
- La plataforma de recolección. El lugar usado para almacenar las necesidades de la multitud obtenidas en la tarea de recolección.

Partiendo de un enfoque similar, desde el 2015 se realiza anualmente una conferencia enfocada en CrowdRE, la cual se organiza de forma colaborativa entre la IEEE y el Instituto Fraunhofer para la Experimentación en Ingeniería de Software. En esta conferencia se presentan los últimos avances en el área de CrowdRE, así como los últimos trabajos en esta línea de investigación.

En la conferencia del 2019, [Van Vliet. M et al (2019)] abordan el tema de requerimientos de productos de manera histórica. Estos expositores describen diferentes plataformas e iniciativas creadas con la intención de facilitar la participación de usuarios y equipos de trabajo en el desarrollo de productos basados en requerimientos. Algunas de estas iniciativas a nivel mundial han sido:

- OPCI: la entidad organizadora y promotora de ideas colaborativas. Es un forum[7] para que las personas puedan compartir sus ideas y recibir retroalimentación. [Castro-Herrera, C. et al(2008)]
- StakeRare: una plataforma en línea donde se fomenta la colaboración de desarrolladores y usuarios para priorizar iniciativas basadas en sugerencias.[Lim, S.L et al. (2012)]

[7]Forum: lugar, reunión o medio donde se comparten ideas y puntos de vista sobre algún tema.

- REfine: una plataforma para la generación de ideas y un sistema donde las personas pueden votar a favor o en contra de iniciativas. [Snijders, R. et al (2015)]

Finalmente, en esta misma conferencia del 2019, se presentó una revisión bibliográfica de artículos enfocados en analizar requerimientos de productos desde la perspectiva de la minería de texto. [Santos, R. et al (2019)]. Se mencionó que, si bien no existe un consenso sobre las mejores metodologías, se observa una prevalencia en el uso de técnicas de análisis supervisado como máquinas de soporte vectorial, el modelo ingenuo de Bayes y bosques aleatorios.

2.2. Ingeniería de requerimientos en Huli

Huli inició su proceso de ingeniería basada en requerimientos en el año 2016. Se creó una metodología en la cual cada persona que recibía retroalimentación de un cliente entraba al CRM y asociaba esta retroalimentación al cliente quien expreso el comentario. En cuatro años de captar esta información, se lograron recolectar 1519 comentarios de clientes, enfocados en diferentes funcionalidades del producto. Mayoritariamente, esta retroalimentación corresponde a los años 2018, 2019 y 2020, dado que el proceso tomó algún tiempo en ser adoptado por la compañía.

En el presente contexto de Huli, los cuatro pilares para la generación de un proceso de requerimientos de productos, según [Mahmood H et at 2017], serían los siguientes:

- La multitud: los doctores que adquieren el producto HuliPractice. En su mayoría, estos pagan una subscripción mensual de renovación automática.
- El recolector: el departamento de ingeniería encargado de la priorización de tareas a nivel de mejoras de producto.
- La tarea de recolección: el proceso que sigue cada departamento para obtener la retroalimentación de los clientes.
- La plataforma de recolección: el CRM donde se registra la retroalimentación asociada a cada cliente.

En 2018 y 2019, se realizaron dos tareas por parte del equipo de ingeniería de producto para incorporar requerimientos basados en la retroalimentación obtenida, pero se encontraron dos problemas principales:

- El procesamiento de los comentarios tomaba alrededor de dos días, ya que no había manera sencilla de agrupar esta información y se ocupa alguna técnica consensuada para interpretar cada comentario.
- Se carecía de un mecanismo para analizar esta información de manera descriptiva: algunos comentarios suelen ser ambiguos y difíciles de interpretar.

Debido a estos problemas, se decidió optar por metodologías automáticas para el análisis de esta información textual. En específico, se eligieron metodologías del área de ingeniería de requerimientos basada en multitudes.

2.3. Minería de texto

La minería de texto se refiere a la extracción de información, previamente desconocida, que se encuentra en fuentes no estructuradas de información, principalmente en textos. [Ananiadou, S (2006)]. El análisis de minería de texto requiere una mezcla de conocimientos entre estadística, aprendizaje de máquina y teoría de la información. [Tufféry, S (2011)]

Las principales aplicaciones de la minería de texto nacen en el campo de la recuperación de la información. Esta se centra en representar, guardar y garantizar el acceso a información textual. La metodología utilizada provee herramientas a los usuarios para el rápido y fácil acceso a la información. [Manning, C. (2008)]

Con la web convirtiéndose en un repositorio universal del conocimiento humano, y con la gran cantidad de información que es generada cada día, se ha vuelto una necesidad la implementación de técnicas que permitan su procesamiento de una manera más ágil y automática. [Manning, C. (2008)]

Empezando con una colección de data textual no estructurada, una aplicación de minería de texto ordena y extrae información útil a partir de los datos disponibles. El interés central de tal procedimiento radica en el uso de la información para la toma de decisiones. El presente trabajo formula la utilización de la minería de texto desde los enfoques de análisis no supervisado y supervisado de información textual. [Grupta, V. (2009)]

2.3.1. Análisis no supervisado

En el análisis no supervisado de información, no se cuenta con una variable Y que guíe el proceso de aprendizaje. En este caso, se tiene un grupo de observaciones N $(x_1, x_2, ..., x_n)$ de un vector aleatorio X con una distribución de densidad $Pr(X)$. El objetivo es inferir los parámetros de $Pr(X)$ sin la ayuda de alguna variable respuesta o de algún error observado.[Hastie T. et al (2008)]

Los algoritmos de agrupamiento de información son utilizados dentro del área de análisis no supervisado. El objetivo de esos algoritmos es encontrar agrupaciones de las observaciones $v = (v_1, v_2, ..., v_n)$ para las variables $X = (X_1, X_2, ..., X_n)$ cuya densidad de probabilidad $Pr(v)$ sea relativamente grande. [Hastie T. et al (2008)]

Sea S_j la representación de todos los conjuntos posibles para los valores de una variable j, y $s_j \subseteq S_j$ un subconjunto de estos valores. Los algoritmos de agrupamiento buscan el subgrupo de variables $s_1, ..., s_p$ cuya probabilidad de densidad sea relativamente grande. [Hastie T. et al (2008)]

También, en el contexto de CrowdRE, dado que se están clasificando comentarios de usuarios y que muchos de estos suelen caracterizarse por su ambigüedad, históricamente los resultados de predicción han oscilado entre 70 % y 85 %.[Santos, R. et al (2019)]. Lo anterior indica que siempre va a existir entre un 25 % y un 15 % de comentarios que no se van a poder clasificar.

Con el análisis no supervisado, en este contexto, se busca agrupar los resultados no predichos por los modelos supervisados y, de esa forma, se pretende reducir el tiempo requerido en las tareas de reclasificación de comentarios. Si, por ejemplo, se tuvieran 1000 comentarios de usuarios en un período y el algoritmo tuviera una capacidad predictiva del 85 %, se esperaría una clasificación correcta de 850 comentarios, aproximadamente. Para los otros 150 comentarios no clasificados, se aplicarían modelos de agrupamiento, a fin de procesar la información de una manera más segmentada y fácil de interpretar.

2.3.2. Análisis supervisado

En el análisis supervisado de información, se usa una matriz de observaciones X (denotada como *predictor*), con la cual se busca predecir el valor de otra variable Y (denotada como *respuesta*). En esta matriz X, cada fila representa el conjunto de mediciones para un mismo individuo i, y cada columna representa el conjunto de mediciones para una misma variable j. [Hastie T. et al (2008)]

El objetivo del análisis supervisado es encontrar la función $\hat{f}(x)$ como aproximación de la función $f(x)$. La función $f(x)$ captura la verdadera relación entre las variables predictoras X y la variable por predecir Y. $\hat{f}(x)$ aproxima esta relación, dado que, para la mayoría de problemas, la relación entre X y Y no va a ser determinística; en cambio, va existir un error ϵ, que es independiente de X y de $E(\epsilon) = 0$. [Hastie T. et al (2008)]

Por lo tanto, la relación entre las variables predictoras y la variable por predecir se puede expresar como $Y = F(X) + \epsilon$.

Esta variable Y, que se quiere predecir, *guía* o *supervisa* el proceso de aprendizaje, de ahí el nombre de análisis supervisado. El termino *aprendizaje de máquina* es una descripción más moderna para referirnos a este concepto.

Como se mencionó anteriormente, en la conferencia del 2019 de CrowdRE, se presentó una revisión bibliográfica de las técnicas más usadas de análisis supervisado en el área análisis de requerimientos de productos basado en multitudes. En esta revisión bibliográfica, se encontró que la mayoría de los investigadores en el área de CrowdRE usan algoritmos de aprendizaje de máquina enfocados en minería textual. Estos algoritmos son usados para clasificar automáticamente la retroalimentación de clientes, con el objetivo de segmentar la carga de trabajo asociada a la interpretación de esa información. [Santos, R. et al (2019)].

Este mismo artículo menciona que, como estos algoritmos no fueron originalmente diseñados para clasificar texto, tienen que ser utilizados en combinación con técnicas que convierten el texto en vectores de donde los algoritmos pueden inferir patrones.

La revisión bibliográfica efectuada arrojó, entre otros, los siguientes resultados:

- La combinación del algoritmo ingenuo de Bayes, la técnica de vectorización de TF-IDF y BOW fue la más usada entre investigadores de CrowdRE.
- El segundo lugar lo ocupó el algoritmo de máquinas de soporte vectorial, también junto con TF-IDF y BOW.
- Se señaló que los algoritmos de bosques aleatorios y el algoritmo de potenciación de gradiente extrema, si bien no se referenciaron tanto como los anteriores, han presentado resultados notables en este tipo de problemas.

Dados estos resultados, se decide resolver el problema de clasificación automática de la retroalimentación de los clientes de Huli usando las técnicas de clasificación y vectorización textual citadas. En la sección de metodología, se describe en detalle el mecanismo para realizar la combinación de modelos y técnicas de vectorización. También, se explica cómo se comparan los resultados para obtener la combinación ideal, según los datos observados.

3. CAPÍTULO III: METODOLOGÍA

En este capítulo, se describen los materiales y métodos utilizados en el procesamiento de la retroalimentación de clientes para HuliPractice.

3.1. Materiales

A continuación, se describe, en primer lugar, la población cuya retroalimentación será analizada; en segundo lugar, la metodología empleada para la recolectar la retroalimentación; y, en tercer lugar, el proceso de etiquetado seguido.

3.1.1. Población

Los principales clientes para el producto web HuliPractice son profesionales de la salud; sin embargo, no todos proporcionan retroalimentación para el producto. Para este trabajo se cuenta con la retroalimentación de doctores clientes de Costa Rica, Panamá y México. Esta retroalimentación se comunica en forma de comentarios hacia alguna funcionalidad o servicio del producto.

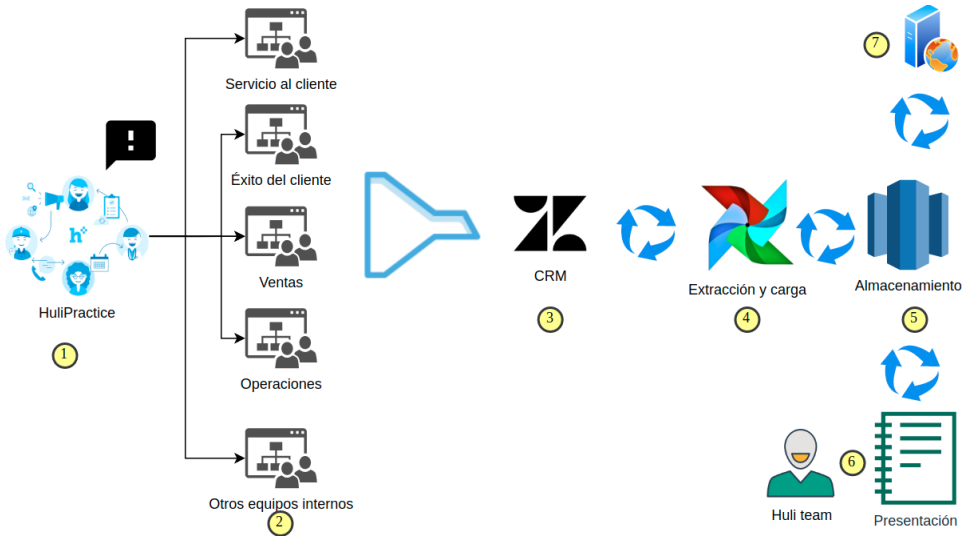
El primer comentario se registró el 25 de octubre del 2016; el último, el 4 de setiembre del 2020. El 4 de setiembre del 2020, se realizó un corte en el proceso, para clasificar los comentarios existentes.

La edad de los doctores que dejaron estos comentarios varía entre los 24 y los 75 años de edad.

3.1.2. Recolección de la retroalimentación

El proceso mediante el cual se recolectó y procesó la retroalimentación de clientes siguió varias etapas. En la figura 2 se muestran estas etapas.

Figura 2: Etapas del proceso de recolección de retroalimentación en Huli



Fuente: Figura de elaboración propia.

A continuación, se describe, paso a paso, el proceso representado en la figura 2.

1. La primera parte del proceso la constituye la acción por parte del cliente de querer brindar retroalimentación. Puesto que es por vía telefónica que se lleva a cabo la mayoría de la comunicación con los clientes, todos los comentarios recibidos fueron recolectados por este medio.

2. El cliente que desea proporcionar retroalimentación se comunica con alguno de los equipos de la compañía. Hay cinco equipos con los que los clientes interactúan: el de servicio al cliente, el de éxito del cliente, el de ventas, el de operaciones y, algunas veces, alguien del equipo de producto recibe algún tipo de retroalimentación. Este tipo de comunicaciones se clasifican como interacción con equipos internos.

3. Una vez que el comentario es recibido por parte de algún miembro de la compañía, este se registra en el CRM. De esta manera, sin importar quién reciba el comentario, este es almacenado en un lugar común asociado al cliente.

4. Usando la herramienta de ETL llamada Airflow[8], implementada en Huli, se cargan estos comentarios, una vez al día, a la base de datos para análisis.

5. Estos comentarios fueron categorizados según la parte del producto a la cual iban dirigidos, y fueron almacenados en la misma base de datos para análisis.

6. Usando la herramienta Jupyter Notebook[9], se realizó el análisis descriptivo, así como la comparación entre modelos no supervisados y supervisados.

7. Finalmente, usando el servicio de AWS EC2, se creó una aplicación web, que basada en los resultados del análisis supervisado de estos comentarios, permite categorizar de manera automática nuevos comentarios.

3.1.3. Proceso de etiquetado de la retroalimentación

La retroalimentación es recibida como comentarios. Estos comentarios vienen sin ninguna categorización previa. Para categorizar los comentarios, se siguió el siguiente procedimiento:

1. Se definieron seis categorías del producto HuliPractice a las cuales los comentarios podrían ir dirigidas:

    - Información médica: sección de HuliPractice donde los doctores registran información médica de sus pacientes.
    - Agenda: sección donde los doctores manejan la agenda de su práctica profesional.
    - Facturación: módulo contable donde los clientes pueden llevar registro administrativo de información contable.
    - Consulta: sección donde los doctores llevan registro de las consultas médicas con sus pacientes.

[8] Airflow es una herramienta escrita en Python que permite realizar tareas de extracción, transformación y carga de datos.

[9] Jupyter Notebook es un documento en línea que permite realizar y compartir análisis en lenguages como Python o R.

- Perfil: parte del producto donde los doctores describen su perfil profesional.
- Servicio: sección que engloba comentarios no necesariamente dirigidos al producto, sino al servicio brindado en torno al producto.

2. Dos personas categorizaron cada uno de los 1519 comentarios por separado. Esta tarea se realizó de manera voluntaria dentro de la compañía. Fue ejecutada por una persona del equipo de producto y otra persona del equipo de servicio al cliente.

3. Una tercera persona revisó los resultados de la primera categorización. Si, para un comentario, las dos primeras personas anotaban categorías distintas, esta persona se encargaba de decidir cuál de clasificaciones tenía más sentido. Por consiguiente, esta persona cumplió un rol de juez en caso de ambigüedad sobre un comentario.

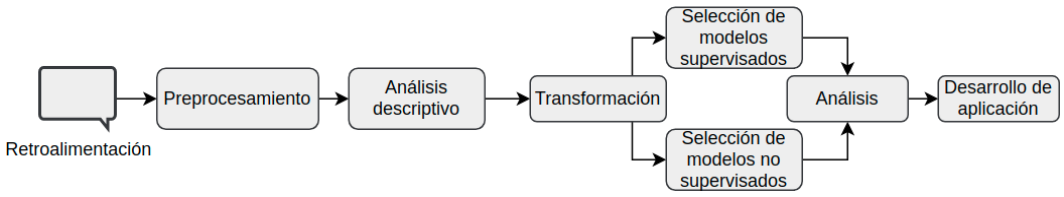
4. Finalmente, los resultados de esta categorización se cargaron a la base de datos de análisis de la compañía.

3.2. Métodos

A continuación, se describe el proceso para convertir la retroalimentación de clientes en información que pueda ser utilizada por los modelos supervisados y no supervisados empleados en este proyecto. Se explica, primero, lo relativo a técnicas exploratorias y al preprocesamiento requerido para este tipo de información. Luego, se describen los modelos usados para el aprendizaje supervisado y no supervisado de la información. Por último se explican las técnicas para entrenar los modelos, el proceso de selección y la validación de los modelos.

La figura 3 muestra un esquema del proceso de datos seguido en este proyecto. En primer lugar, se efectúa un preprocesamiento de la información, el cual incluye la limpieza de la retroalimentación de clientes existente. Luego, se lleva a cabo un análisis descriptivo, con la información ya limpia. Seguidamente, se realizan transformaciones a esta información, para convertir los comentarios en datos que puedan alimentar a los modelos no supervisados y supervisados. Con lo anterior es posible, finalmente, realizar un proceso de validación y escogencia entre estos modelos. El modelo supervisado que produzca los mejores resultados será implementado en una aplicación web para clasificar la retroalimentación.

Figura 3: Etapas del proceso de procesamiento de retroalimentación.



Fuente: Figura de elaboración propia.

3.2.1. Preprocesamiento

La retroalimentación proporcionada por los doctores hacia el producto es compartida en forma de comentarios. Dentro de estos comentarios, algunas palabras cargan más significado que otras.

Si bien existen técnicas de procesamiento de texto en las que la relación semántica y contextual de las palabras es relevante [Qiaozhu, M (2010)], para las técnicas usadas en este trabajo tal relación no lo es. Las técnicas usadas aquí se basan más en la frecuencia de las palabras dentro del texto, por lo que el orden y relaciones internas son irrelevantes en este contexto. Por lo tanto, en el presente trabajo, antes de realizar cualquier análisis, es necesario eliminar y transformar el contenido del texto, para conservar solo los términos más relevantes.

Este preprocesamiento del texto es común entre aplicaciones que procesan información textual no estructurada.[Baeza, R et al (1999)]. Yates y Neto proponen las siguientes etapas de preprocesamiento:

- Eliminar del texto dígitos, tildes, caracteres de puntuación y mayúsculas.
- Eliminar palabras vacías[10] que no aportan nada en la discriminación del contenido textual (todos los comentarios las contienen). Algunos ejemplos de estas palabras son *la*, *a*, *ella* y *el*.
- Remover del texto prefijos y sufijos, conservando solo la raíz de las palabras. Este proceso se llama en inglés *stemming*. Este proceso reduce la variabilidad de las palabras. Por ejemplo, si se tuvieran las palabras *conectar, conectando y conectado*, después de esta transformación, todas estas palabras pasarían a la forma *conect* (la raíz de la palabra) y se agruparían como la misma entidad.
- Corrección ortográfica. Esta etapa consiste en corregir palabras mal escritas, con el propósito de asegurar que su escritura siga las normas ortográficas.
- Dividir el texto en *tokens*[11]. Cada token debe corresponder a una palabra dentro del contenido ya transformado.

El analista decide cuáles etapas implementar y qué modificación efectuar a cada etapa. Para este trabajo, se implementaron las siguientes:

1. Primero, se eliminaron del texto caracteres como dígitos, tildes, caracteres de puntuación y mayúsculas. Además, se removieron espacios en blanco y caracteres conformados con una sola letra.
2. Se eliminaron palabras vacías y otras palabras que no aportaban a la discriminación del contenido, como nombres propios y correos del colaborador de la empresa que dejó el comentario.
3. Se removieron prefijos y sufijos de las palabras.
4. Se realizó una corrección ortográfica en dos etapas. Primero, se corriegieron las palabras usando una librería de Python llamada Hunspell. Luego, se corrigió manualmente un conjunto de 500 palabras que no estaban dentro del diccionario de Huspell en español.
5. Se dividió el texto en tokens, después de haber aplicado las etapas anteriores.

Finalizando el proceso, se pasa de tener un comentario a un arreglo de tokens.

[10] El término *palabras vacías* se conoce en inglés como *stopwords*

[11] Un token es un objeto que sirve de representación para algo, en este caso, palabras

3.2.1.1. Valores extremos

Antes de ejecutar el preprocesamiento de la información, se decidió eliminar valores extremos[12]. Los valores extremos corresponden a comentarios con demasiadas palabras. La técnica usada para el cálculo de valores extremos fue *z-score*. El z-score se define como la cantidad de desviaciones estándar a la que está una observación de la media, asumiendo una distribución normal, y viene dado por la fórmula 1.[Iglewicz, B et al (1993)]

$$z = \frac{x - \upsilon}{\sigma} \tag{1}$$

Se dividió cada comentario en las palabras que lo conformaban, se contó la cantidad de palabras y se calculó la media y la desviación estándar asumiendo una distribución normal. Para cada uno de los comentarios, se calculó el z-score con base en la cantidad de palabras. Se consideraron z-scores mayores a 3.5 como valores extremos.[Iglewicz, B et al (1993)]. Los comentarios cuyo z-score fuera mayor a 3.5 fueron considerados valores extremos y, en consecuencia, fueron excluidos del análisis.

3.2.2. Análisis descriptivo

Para el análisis descriptivo de la retroalimentación, se utilizan las técnicas de nube de palabras y análisis de redes. También, se lleva a cabo un análisis descriptivo de las palabras más prominentes dentro de esos comentarios, usando N-grama[13](en este caso unigramas y bigramas).

3.2.2.1. Nube de palabras

Una nube de palabras es una técnica que se usa para representar visualmente las palabras que aparecen con más frecuencia en un texto. En este tipo de técnica, entre mayor sea la frecuencia de la palabra en el texto, mayor es el tamaño de esta. La nube se puede elaborar partiendo el texto en una palabra(es decir, en unigramas), dos palabras(es decir, en bigramas), tres palabras(es decir, en trigramas), etc. [Bengfort, B et al (2018)]

Para efectos de este trabajo, se optó por mostrar la nube de palabras en unigramas y bigramas. Todo el texto de retroalimentación se considera como una sola entidad, y se calcula la frecuencia de cada palabra dentro de esa gran entidad.

3.2.2.2. Análisis de redes

El modelo de red de palabras es representado como un grafo direccionado $G = (V, E)$, con un conjunto de nodos V que representan $N-gramas$ únicos, y un conjunto de conexiones E que conectan cada nodo. La dirección de la conexión es consistente con el orden en que los $N - gramas$ aparecen en el texto. El peso de cada nodo representa su frecuencia dentro del texto. [Leonard T. (2018)]

El graficar esta red permite apreciar las palabras con más centralidad en un texto. La centralidad indica importancia e influencia de ciertas entidades dentro del contenido. [Bengfort, B et al (2018)]

[12]Los valores extremos pueden aumentar el error en el análisis, por eso se removieron.
[13]Un *N-grama* es una subsecuencia de n elementos de texto

Existen diferentes maneras de calcular la centralidad, pero para este proyecto se escogió el grado de centralidad.

El grado de centralidad d_v de un nodo v en una red de palabras G cuenta la cantidad de conexiones en E que apunta directamente a v. [Leonard T. (2018)]

El método más común para apreciar esta centralidad es considerar las palabras como nodos de un grafo y usar las medidas de centralidad para representar la fuerza de las conexiones entre los nodos. En este trabajo, se van a mostrar las relaciones entre $bi-gramas$ dentro de los comentarios usando un análisis de redes. Cada nodo es un $bi-grama$ dentro del texto y la fuerza de las relaciones es el grado de centralidad del $bi-grama$.

3.2.3. Transformaciones

Los algoritmos de aprendizaje supervisados y no supervisados, en el área de minería de texto, no pueden ser ajustados directamente con texto. Estos modelos esperan vectores de tamaño fijo en lugar de texto de tamaño variable, por lo que, antes de la aplicación de estos modelos, el texto tiene que ser trasformado en una representación numérica de este. Al proceso de transformar una colección de documentos a vectores numéricos se le conoce como vectorización. [Silge, J et al (2017)]

Hay diferentes técnicas para realizar el proceso de vectorización; sin embargo, para fines prácticos, este trabajo se va a centrar en dos técnicas identificadas en la revisión bibliográfica referente a minería de texto para requerimientos de productos: las bolsas de palabras y la técnica TF-IDF.

3.2.3.1. Bolsa de palabras (BOW)

La técnica de vectorización textual conocida como bolsa de palabras consiste en transformar el texto en una matriz donde cada fila corresponde a una oración y cada columna a una palabra [Weinberger, K et al (2009)]. Cada celda de esta matriz puede corresponder a:

- Un 1 o un 0, lo que indica si la palabra está dentro de la oración o no. Ya que se están usando solo unos y ceros, a esta variación particular se le conoce como bolsa de palabras binarizada[14].
- La frecuencia de la palabra dentro de la oración. Dado que este es el caso contrario del anterior, se conoce como bolsa de palabras no binarizada.

Por ejemplo, considérese un caso en el que se tuvieran las dos oraciones siguientes:

"1. Hoy es un día hermoso"
"2. El día es es hermoso."

.

Su representación vectorial como bolsas de palabras se podría apreciar en el cuadro 1.

[14]Binarizar se refiere a representar una cantidad en base numérica dos.

Cuadro 1: Ejemplificación de bolsa de palabras binarizada y no binarizada

Oración	Binarización	hoy	es	un	día	hermoso	el
1	Sí	1	1	1	1	1	0
2	Sí	0	1	0	1	1	1
1	No	1	1	1	1	1	0
2	No	0	2	0	1	1	1

Fuente: Cuadro de elaboración propia.

Las dos oraciones, en su representación binarizada, corresponden a vectores de unos y ceros, mientras que la versión no binarizada corresponde a vectores donde cada celda representa la frecuencia de la palabra dentro de la oración.

3.2.3.2. Frecuencia de término - inversa frecuencia del documento (TF-IDF)

Cuando se trabaja con una gran cantidad de contenido textual, es posible que algunas palabras con poco significado pero que aparecen con mucha frecuencia (por ejemplo, artículos) parezcan más relevantes que palabras menos frecuentes pero con más significado dentro del texto[Weinberger, K et al (2009)]. La técnica de vectorización TF-IDF contrarresta este efecto dividiendo la frecuencia de la palabra entre la frecuencia de la palabra en todo el texto. Se puede expresar como:

$$tfidf(t,d) = tf(t,d) \times idf(t) \tag{2}$$

Donde t se refiere al término y d al documento(oraciones, comentarios, etc.). Mientras que el término $idf(t)$ se define como

$$\log \frac{1+n}{1+df(t)} + 1 \tag{3}$$

Donde n es el total de documentos y $df(t)$ es la cantidad de documentos que contienen el término t. Al final, los vectores resultantes son normalizados usando la norma Euclidea:

$$v_{norm} = \frac{v}{||v||_2} = \frac{v}{\sqrt{v_1^2 + v_2^2 + ... + v_n^2}} \tag{4}$$

Esta técnica fue desarrollada originalmente en el área de recuperación de la información. Ha sido usada en gran medida para clasificación y agrupamiento de documentos[Baeza, R et al (1999)].

3.2.3.3. Reducción de dimensión

Cada comentario contiene una cantidad muy pequeña de palabras, por lo que el vector resultante con cualquiera de las técnicas de vectorización anteriormente descritas va a contener muchos ceros. Antes de aplicar los modelos supervisados y no supervisados sobre la retroalimentación transformada, se tiene que reducir la dimensión de estos vectores.

La técnica escogida para la reducción de dimensión fue la prueba Chi-Square de Pearson($\tilde{\chi}^2$).

La prueba $\tilde{\chi}^2$ evalúa si los parámetros de una distribución multinomial son iguales a algún valor particular. Esta prueba es aplicada a un conjunto de variables categóricas para evaluar qué tan probable es que las diferencias observadas entre las observaciones sea producto de la casualidad. Con esta prueba se pueden realizar tres diferentes comparaciones entre variables categóricas, a saber: bondad de ajuste, homogeneidad e independencia.[Agresti A, 2013]

Considere $H_0 : \pi_j = \pi_{j0}, j = 1, ..., c$, donde $\sum_j \pi_{j0} = 1$. Cuando H_0 es verdadero, los valores esperados de $\{n_j\}$(llamados frecuencias esperadas) son $\mu_j = n\pi_{j0}$, $j = 1, ..., c$, por lo que Pearson propuso el siguiente test:

$$\tilde{\chi}^2 = \sum_{j=1}^{n} \frac{(n_j - \mu_j)^2}{\mu_j} \tag{5}$$

Grandes diferencias de $\|n_j - \mu_j\|$ producen grandes valores de $\tilde{\chi}^2$. En pruebas de independencia, H_0 se define estableciendo que las variables categóricas son independientes, mientras que H_1 establece que existe algún grado de asociación entre las variables. Asumiendo una significancia del 5 %, $p - values < 0{,}05$ muestra evidencia de que rechaza H_0, por lo que surge algún grado de relación entre las variables.[Agresti A, 2013]

En este trabajo, se va a usar la prueba $\tilde{\chi}^2$ para verificar la dependencia entre variables categóricas. Se compara el grado de discriminación de cada término dentro del texto con respecto a las clases primarias, lo que permite eliminar los términos menos relevantes y reducir la dimensión del vector inicial.

Al no conocerse de entrada la dimensión del vector ideal(el tamaño del vector que conserve suficiente información para generar los mejores resultados en los modelos supervisados y no supervisados), se va a usar la siguiente heurística:

1. Calcular el estadístico $\tilde{\chi}^2$ para cada uno de los términos, en relación con las categorías primarias.
2. Conservar los n términos que guarden más relación con las categorías, según la prueba $\tilde{\chi}^2$. Donde $n = \{20, 80, 200, 400, 800, 1400, 1600\}$.
3. Ajustar el modelo con estos vectores y guardar los resultados.
4. Repetir el paso 2 con el siguiente valor de n.
5. Comparar los resultados de todos los modelos para ver qué tamaños de n dangeneran los mejores resultados.

También, se va a utilizar esta prueba para evaluar la dependencia de las categorías primarias con respecto a dos variables categóricas que serán incluidas en el análisis: en primer lugar, el departamento dentro de la compañía al que pertenece el colaborador de Huli que anotó el comentario y, en segundo lugar, una clasificación previa del cliente dentro del proceso de ventas.

3.2.4. Modelos no supervisados

A nivel de modelos no supervisados, se quiere agrupar de manera automática la retroalimentación no clasificada por el modelo supervisado. Con ello, se busca facilitar el proceso de etiquetado manual. Para esto, se van a probar dos modelos de agrupamiento automático de información: el modelo k-medias y el modelo de agrupación espacial basada en densidad de aplicaciones con ruido (DBScan). Asimismo, en este apartado se presenta la metodología para la validación de los modelos y la metodología para la escogencia del modelo ideal.

3.2.4.1. K-medias

El algoritmo de k-medias es usado para particionar un grupo de observaciones en una cantidad predefinida de k clústeres. El algoritmo descrito por [MacQueen, J.B. (1967)] comienza con un grupo aleatorio de k centroides (μ). En cada paso de actualización, todas las observaciones x son asignadas al centroide más cercano (ver ecuación 6). En la versión estándar del algoritmo, solo es posible asignar cada observación a un único conglomerado. Si una observación presenta la misma distancia a dos o más grupos, se asignará a cualquiera de estos de forma aleatoria.

$$S_i^{(t)} = \left\{ x_p : \left\| x_p - \mu_i^{(t)} \right\|^2 \leq \left\| x_p - \mu_j^{(t)} \right\|^2 \ \forall j, 1 \leq j \leq k \right\} \tag{6}$$

Posteriormente, los centroides son reposicionados mediante el cálculo de la media de las observaciones asignadas a cada uno de estos (ver ecuación 7).

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{7}$$

El proceso de actualización se repite hasta que todas las observaciones permanezcan asignadas al mismo centroide; por lo tanto, estos centroides no serán actualizados nuevamente. Esto significa que el algoritmo de k-medias trata de optimizar la función objetivo 8. Como solo existe un número finito de posibles asignaciones para la cantidad de centroides y de observaciones , y como cada iteración tiene que resultar en una mejor solución, el algoritmo siempre termina en un mínimo local.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{8}$$

$$\text{Con } r_{nk} = \begin{cases} 1 & x_n \in S_k \\ 0 & \text{de otra manera} \end{cases}$$

El principal problema de k-medias es su dependencia de los centroides escogidos inicialmente. Estos podrían terminar separando datos similares, mientras que otras observaciones separadas podrían ser agrupadas si algunos de los centroides son más influenciados por valores atípicos.

El enfoque más común, el cual es usado en este trabajo, consiste en desarrollar múltiples agrupaciones utilizando diferentes centroides de inicio e iterar n veces. Se puede considerar como correcta la agrupación que ocurrió en un mayor número de ocasiones en las n iteraciones.

3.2.4.2. Agrupación espacial basada en densidad de aplicaciones con ruido (DBScan)

El algoritmo de DBScan pertenece a la familia de algoritmos de clasificación basados en densidad de grupos. Usando este algoritmo, las observaciones son categorizadas en: puntos centrales que guían el proceso de agrupamiento, puntos limite que delimitan los bordes de los grupos y puntos de ruido que no pueden ser agrupados porque están muy lejos de los puntos centrales y de limite,. [Braune, C et al, (2015)] Antes de describir el algoritmo, se necesita aclarar algunas definiciones:

1. El vencidario ϵ de un punto x_i es el grupo de puntos que tienen una distancia de máximo ϵ al punto x_i.
$$N_\epsilon(x_i) = \{x \in D | d(x_i, x) \leq \epsilon\} \tag{9}$$

2. Al conjunto de grupos centrales pertenecen todos aquellos puntos cuyo vecindario ϵ contiene al menos $minPts$ otros puntos.
$$P = \{x \in D | \|N_\epsilon(x)\| \geq minPts\} \tag{10}$$

3. Una observación p es llamada *alcanzable directamente por densidad* desde un punto central x si $p \in N_\epsilon(x)$

4. Una observación p es llamada *alcanzable por densidad* desde un punto central x si existen una serie de puntos $p_1, p_2, ..., p_n$ con $p = p_n$ y $x = p_1$. Además, si para cada par (p_i, p_{i+1}) se cumple que p_{i+1} es *alcanzable directamente por densidad* desde p_i.

5. Dos observaciones p y q están *conectadas por densidad* si existe un punto o tal que p y q son *alcanzables por densidad* desde o.

Con estos conceptos de densidad, se puede afirmar que el algoritmo DBScan encuentra grupos C, tal que para cada p y q que pertenecen a C se cumple que p y q están *conectados por densidad* y no hay un subgrupo de C que también se pueda considerar un grupo según la definición anterior.[Braune, C et al, (2015)]

Cada punto que no esté *conectado por densidad* a otro punto es considerado como un punto de ruido. Por lo anterior, a este algoritmo se le conoce como *agrupamiento espacial basado en densidad de aplicaciones con ruido.*

La escogencia de $minPts$ y ϵ influencia los resultados del algoritmo, por lo que estos se consideran hiperparámetros por optimizar.

3.2.4.3. Métricas para el calculo de distancias

Los dos algoritmos de agrupamiento descritos anteriormente necesitan calcular distancias entre diferentes puntos. En este trabajo, se quiere agrupar la retroalimentación de los clientes, motivo por el cual, antes de aplicar estos modelos no supervisados, los comentarios tienen que transformarse también a una representación vectorial. Para ello, se usarán las mismas cuatro técnicas de vectorización empleadas para el análisis supervisado: BOW binarizada/no-binarizada y TF-IDF binarizada/no-binarizada.

La escogencia de métricas para el calculo de distancia juega también un papel importante en los resultados. En este trabajo, se van a usar dos métricas diferentes para calcular distancia: la distancia Euclidea y la distancia de cosenos.

La distancia Euclidea se define como:

$$d_e(p,q) = \sqrt{\sum_{i=1}^{i}(q_i - p_i)^2} \tag{11}$$

donde p y q son dos vectores de dimensión n.[Manning, P et al. (2008)]

Por su parte, la distancia de cosenos se define como:

$$d_c(p,q) = \frac{pq}{\sqrt{pp}\sqrt{qq}} \tag{12}$$

Donde p y q son también dos vectores.[Manning, P et al. (2008)]

3.2.4.4. Validación de modelos

Dado que se cuenta con 1492 comentarios, ya categorizados, de clientes, se va a aplicar una aproximación supervisada para validar los modelos y técnicas de agrupamiento. Se van a ajustar los modelos no supervisados con base en diferentes hiperparámetros y se van a comparar los resultados según las categorías reales de los comentarios. Esta comparación se va a llevar a cabo mediante la *medida V.*

Según [Rosenberg, A et al (2007)], la *medida V* provee una solución elegante para problemas de agrupamiento donde se trabaja con grupos previamente definidos. Esta medida resume dos aspectos deseables de los agrupamientos: homogeneidad y completitud.

Una agrupación es homogénea si todos todos los grupos contienen observaciones de una misma categoría[Bonaccorso B. (2018)]. El cálculo de la homogeneidad está definido como:

$$h = 1 - \frac{H(Y_{true}|Y_{pred})}{H(Y_{true})} \tag{13}$$

Por otro lado, una agrupación se considera completa si todas las observaciones de una misma clase están contenidas en el mismo grupo[Bonaccorso B. (2018)]. El cálculo de la completitud de una agrupación está definido como:

$$c = 1 - \frac{H(Y_{pred}|Y_{true})}{H(Y_{pred})} \tag{14}$$

La *medida V* es una medida harmónica entre la homogeneidad y la completitud de una agrupación. Esta medida está definida como:

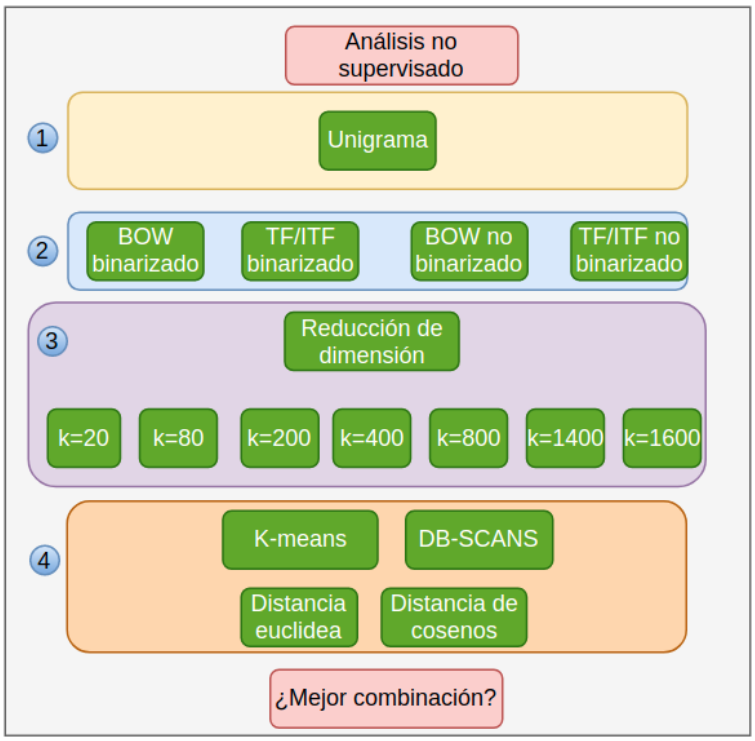
$$v = \frac{(1+\beta)\cdot h \cdot c}{\beta \cdot h + c} \tag{15}$$

Donde β otorga peso a la relación entre la homogeneidad y la completitud y $0 \leq v \leq 1$. Valores de v cercanos a 1 dan lugar, por consiguiente, a grupos muy homogéneos y completos. Si β es mayor a 1, se le asigna más importancia a la completitud que a la homogeneidad; mientras que, si β es menor a 1, se le asigna más peso a la homogeneidad que a la completitud. Para este trabajo, se va a mantener $\beta = 1$, lo que atribuye igual peso a la homogeneidad y la completitud de la agrupación.

3.2.4.5. Selección del modelo según criterios de análisis

En la figura 4, se muestran las técnicas escogidas en cada etapa del análisis no supervisado. Se busca seleccionar el mejor conjunto de hiperparámetros y técnicas para agrupar, de manera automática, la retroalimentación de clientes.

Figura 4: Etapas en el proceso de selección del modelo no supervisado ideal



Fuente: Figura de elaboración propia.

A continuación, se describen las etapas representadas en la figura 4.

1. Etapa 1: selección de unigramas o bigramas. En este caso, dados los resultados de los modelos de clasificación, se optó por usar solo unigramas.

2. Etapa 2: selección de técnicas de vectorización.

3. Etapa 3: selección de k ($k = 20, 80, 200, 400, 800, 1400, 1800$) términos por conservar, según la relación con las categorías primarias, para la reducción de la dimensión del vector original.

4. Etapa 4: selección de hiperparámetros para cada uno de los modelos de agrupamiento y métricas de distancia.

Diferentes combinaciones de estas técnicas e hiperparámetros pueden mejorar o decrecer el resultado del agrupamiento. Por lo tanto, puesto que de antemano no se cuenta con información de la combinación ideal y, además, el espacio de búsqueda es amplio, se procede de la siguiente manera:

1. Para cada uno de los modelos no supervisados:

   a) Se seleccionan los hiperparámetros por ajustar y el conjunto de valores para cada uno de los hiperparámetros.

   b) Se selecciona un valor para cada uno de los hiperparámetros del modelo.

   c) Se selecciona una métrica de distancia.

   d) Se selecciona una técnica en cada una de las etapas 1-3 del proceso de análisis supervisado.

   e) Se ajustan los modelos y se guardan los resultados de la *medida V* para los resultados de las agrupaciones.

   f) Se repiten los pasos b-e hasta que se haya explorado todo el conjunto de valores de los hiperparámetros y cada una de las técnicas de las etapas 1-3.

2. Se comparan los modelos de agrupamiento, según la *medida V*, seleccionando la combinación que produzca los mejores resultados (cuanto más alta la *medida V*, más homogéneos y completos son los resultados de agrupamiento).

Cabe resaltar que, ya que se tuvieron que calcular alrededor de 972 modelos de agrupamiento, se usó también un servidor de AWS ml.m4.16xlarge optimizado específicamente para tareas de análisis. Este computador cuenta con 96 núcleos y 256 GB de memoria RAM, por lo que el ajuste de estos modelos tardó alrededor de dos horas y representó un costo aproximado de $8.

3.2.5. Modelos supervisados

Para la clasificación automática de los comentarios dentro de las categorías primarias del producto, se van a usar cuatro modelos diferentes que la revisión bibliográfica en CrowdRE sugiere: el modelo ingenuo de Bayes, máquinas de soporte vectorial, bosques aleatorios y el modelo XGBoost. [Santos, R. et al (2019)]

En esta sección, se describen dichos modelos. Asimismo, se describe la metodología empleada para el ajuste de los hiperparámetros de los modelos, la metodología aplicada para escoger el mejor modelo y la metodología usada para evaluar la complejidad de estas técnicas.

3.2.5.1. Modelo ingenuo de Bayes(NB)

El modelo ingenuo de Bayes[15] para múltiples clases es una técnica probabilística, la cual asume que la distribución de las diferentes palabras dentro del texto es independiente entre sí.[Manning, P et al. (2008)] La probabilidad de que un documento d pertenezca a una clase c es calculada como:

$$P(c|d) \approx P(c) \prod_{1 \leq k \geq n_d}^{k} P(t_k|c) \tag{16}$$

Donde $P(t_k|c)$ es la probabilidad condicional del término t_k que ocurre en un documento de la clase c. $P(t_k|c)$ se puede interpretar como: tanta evidencia t_k contribuye a que c sea la clase correcta para

[15]En inglés, *Multiclass Naive Bayes classifier*

d. $P(c)$ es la distribución previa de un documento que ocurre en la clase c. La fórmula anterior se deriva del modelo original propuesto por Thomas Bayes alrededor de 1700, conocido como el teorema de Bayes.[Manning, P et al. (2008)]

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{17}$$

Donde A y B son eventos y $P(B) \neq 0$.

En clasificación de textos, el objetivo es encontrar la mejor clase para el documento.[Platt, J. (1998)] La mejor clase para el documento, usando este modelo, se define como:

$$c_{map} = argmax_{c \in C} P(c|d) = argmax_{c \in C} P(c) \prod_{1 \leq k \geq n_d}^{k} P(t_k|c) \tag{18}$$

Para problemas multiclase, la distribución está parametrizada por el vector $\theta_y = (\theta_{y1}, ..., \theta_{yn})$ para cada clase de y, donde n es el número de palabras en el vocabulario y θ_{yi} es la probabilidad de la palabra i que aparece en la clase y.[Platt, J. (1998)]
El parámetro θ_y es estimado por:

$$\hat{\theta_{yi}} = \frac{N_{yi} + \alpha}{N_y + n\alpha} \tag{19}$$

donde $N_{yi} = \sum_{x \in T} x_i$ es la cantidad de veces que la palabra i aparece en la clase y para la muestra T, y $N_y = \sum_{i=1}^{n} N_{yi}$ es la cantidad total de veces que aparece la palabra i en la clase y.

El hiperparámetro por optimizar para este modelo es el valor de α. Este hiperparámetro normalmente varía de 1 a 0, y se usa para suavizar la estimación de $\hat{\theta_{yi}}$ por términos no presentes en el grupo de entrenamiento.

3.2.5.2. Modelo máquinas de soporte vectorial(SVM)

El modelo original de máquinas de soporte vectorial(SVM) fue inventado por Vladimir Vapnik y Alexey Chervonenkis en 1993. En 1992, un modelo mucho más robusto fue propuesto por Bernhard Boser, Isabelle Guyon y Vladimir Vapnik[Platt, J. (1998)]. La idea principal del modelo es seleccionar un hiperplano de separación que equidista de los ejemplos más cercanos de cada clase, a fin de obtener el margen máximo de cada lado del hiperplano y, de esa manera, separar las observaciones.

A modo de ejemplo, considere un grupo de entrenamiento de n observaciones $\{\mathbf{x}_i, y_i\}$ $i = 1, \ldots, n$ donde $y_i \in \{-1, 1\}$ es la clase de observación x_i. Tomando dos clases como $y_i = 1$ y $y_i = -1$, queremos encontrar una regla de clasificación (hiperplano) que pueda mapear x_i a una dimensión más alta, de tal forma que las observaciones de las dos clases puedan ser divididas. También, se quiere que el hiperplano posea el margen máximo que permita maximizar la distancia entre el hiperplano y los puntos más cercanos de ambas clases.

Para casos lineales, el hiperplano puede ser escrito como:

$$\mathbf{x}_i \mathbf{w} + b = 0$$

Queremos encontrar dos hiperplanos paralelos que puedan separar la información a la vez y que, al mismo tiempo, mantengan la distancia más grande posible entre sí mismos. Estos hiperplanos se

definen como:

$$\mathbf{x}_i\mathbf{w} + b = +1$$

y

$$\mathbf{x}_i\mathbf{w} + b = -1$$

Además, la distancia entre los dos hiperplanos se define como $\frac{2}{||w||}$, donde:

$$d_+ + d_- = \frac{|1-b|}{\|w\|} + \frac{|-1-b|}{\|w\|} = \frac{2}{\|w\|} \tag{20}$$

Así que lo que se busca es minimizar $\|w\|$ y, además, que cada $i \in (1, n)$, x_i y y_i siga las siguientes reglas, para garantizar la separación de la mayor cantidad posible de observaciones:

$$\mathbf{x}_i\mathbf{w} + b \geq +1, \quad y_i = +1 \tag{21}$$

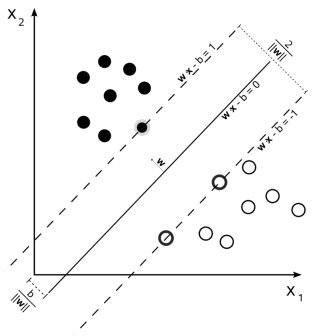
$$\mathbf{x}_i\mathbf{w} + b \leq -1, \quad y_i = -1 \tag{22}$$

$$\equiv \tag{23}$$

$$y_i(\mathbf{x}_i\mathbf{w} + b) - 1 \geq 0, \qquad \forall i \tag{24}$$

En la figura 16, se muestra una representación gráfica del procedimiento de separación con dos hiperplanos.

Figura 5: Separación de observaciones con hiperplanos en SVM



Fuente: Cristianini et al, (2000)

También, este algoritmo cuenta con extensiones para ser utilizado en problemas de clasificación no lineales. Si los ejemplos no son separables linealmente en el espacio original, la búsqueda del hiperplano de separación – normalmente de muy alta dimensión– se lleva a cabo de forma implícita, utilizando distintos Kernels[16]. Estos kernels son funciones que reciben los datos de entrenamiento como entrada y realizan transformaciones para proyectarlos en un espacio dimensional superior que sea linealmente separable[Cristianini, N, et al. (2000)]. Las funciones de Kernel más usadas son la lineal, la polinomial, la sigmoidea y la base radial. Las 4 funciones se probarán para optimizar los resultados del modelo.

Por otra parte, en la clasificación con múltiples clases, lo que se hace es descomponer el problema en pares de clasificación binaria y usar un método de consenso para llegar a la respuesta correcta.

[16]Del íngles, núcleos

Por lo tanto, si se tienen c clases, se tienen que entrenar $\frac{c(c-1)}{c}$ modelos. Para este caso es $c = 6$, por lo que, para cada combinación de hiperparámetros, se tienen que entrenar 15 modelos. Al final, se utiliza un sistema de votos entre los modelos, para decidir a cuál clase pertenece una nueva observación.[Cristianini, N, et al. (2000)]

Este modelo cuenta con un hiperparámetro de regularización llamado C. La fuerza de la regularización es inversamente proporcional a este hiperparámetro, el cual tiene que ser estrictamente positivo. Se van a probar diferentes valores de C para optimizar los resultados; además, se va a verificar el cambio en la complejidad de este.

3.2.5.3. Modelo de bosques aleatorios(RF)

Para adentrarse en el modelo de bosques aleatorios, es necesario iniciar detallando los conceptos de *árboles de decisión y bagging*[17].

Los árboles de decisión consisten en un modelo estadístico donde se busca estratificar el espacio de variables predictoras en un número de regiones simples que clasifiquen las observaciones en categorías determinadas. Esta estratificación adquiere forma de árbol, de ahí el nombre del modelo.[Gareth J, et al. (2013)]

Para generar este modelo, se utiliza el algoritmo de Hunt, el cual se usa para dividir los datos de manera recursiva. Si s_t es el conjunto de entrenamiento asociado con el nodo t, y si $c_j = \{c_1, ..., c_n\}$ es el conjunto de etiquetas de cada clase, el algoritmo de Hunt recursivo se define según los siguientes criterios:

1. Si todas las observaciones en s_t pertenecen a la clase c_j, entonces t es un nodo hoja que se etiqueta como y_j y la partición acaba por esa rama.

2. Si s_t contiene registros que pertenecen a más de una clase, se escoge la variable que mejor divida el conjunto original en subconjuntos más pequeños(se crean nuevos nodos) y se dividen las observaciones. El algoritmo se vuelve a aplicar hasta que solo se tengan nodos hojas.

Las divisiones de los nodos se realizan de tal manera que la impureza del nodo raíz sea menor que la de los nodos hijos. Para medir la impureza de los nodos, se empleó el índice de Gini, definido como:

$$Gini(t) = 1 - \sum_j P(j|t)^2 \tag{25}$$

Donde $P(j|t)$ es la probabilidad de pertenecer a la clase j, si está en el nodo t.
Para cada variable se calcula la ganancia de información(GI), definida como:

$$GI(t) = Gini(nodopadre) - \sum_j \frac{n}{n_j} Gini(nodohijo) \tag{26}$$

Al final, para realizar las divisiones sucesivas en el algoritmo de Hunt, se selecciona la variable que genere la mayor ganancia de información.

[17]Este término se traduce al español como *harpillera*, material usado en zapatos y sacos.

Por su parte, la técnica de *bagging* busca reducir la varianza de métodos de aprendizaje. Gareth et al. (2001), proponen que una forma de reducir la varianza y, por ende, de incrementar la precisión de un modelo es tomar muchos grupos de entrenamiento de la población, construir un modelo para cada uno de estos y promediar sus predicciones. Entonces, el procedimiento consiste en calcular $\hat{f}^1(x), \hat{f}^2(x), ..., \hat{f}^B(x)$, donde cada f es un árbol de decisión ajustado para B grupo de entrenamiento. Estos datos se promedian para obtener un único modelo de varianza baja, dado por:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

Este procedimiento presenta la limitación de que, usualmente, no se tiene acceso a múltiples grupos de entrenamiento. En su lugar, es posible remuestrear diferentes grupos a partir de un único grupo de entrenamiento (*bootstrap*). En este enfoque, se generan B diferentes grupos de entrenamiento remuestreados a partir del conjunto de datos original. Posteriormente, se entrena el modelo en cada b-ésimo grupo de entrenamiento, para calcular $\hat{f}^{*b}(x)$ y, finalmente, promediar todas las predicciones, a fin de obtener:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

Esto es conocido como *bagging*.

El modelo de bosques aleatorios usa los dos conceptos arriba descritos para llevar a cabo tareas de clasificación. Se construyen k árboles de decisión y se emplea el método de bagging para generar el promedio de las predicciones; luego de esto, es posible clasificar una nueva observación.

Así, el algoritmo de árboles aleatorios se puede describir como:

1. Para $b = 1$ a B. Donde B es la cantidad de árboles.
    - *a*) Generé un muestra usando el método de remuestreo *bootstrap* Z^* de tamaño N del grupo de entrenamiento.
    - *b*) Genere un árbol de decisión T_b de esta muestra. Esto aplicando el siguiente proceso de manera recursiva para cada nodo terminal del árbol. Continué hasta que se alcance el valor mínimo del nodo n_min.
        1) Seleccione m variables aleatorios de las p iniciales.
        2) De las m variables disponibles, escoja la que mejor divide el árbol.
        3) Divida el nodo en dos nodos hijos.
2. Guarde el conjunto de árboles ${T_b}^B$ generados del proceso anterior y úselos para hacer una nueva predicción.
3. La clasificación de una nueva observación se realiza por mayoría de votos entre todos los árboles generados.

La definición de las m variables a escoger en el paso b,1 del algoritmo anterior es de suma importancia. Normalmente este valor se define por defecto como $\sqrt{p}$. Para este trabajo se va a probar con dos valores: $\sqrt{p}$ y $\log p$.

También, este modelo cuenta con un hiperparámetro llamado *poda mínima para reducir el costo y la complejidad*. Este es un hiperparámetro por optimizar, para evitar sobreajuste.[Breiman, L. et al (1984)]. El algoritmo de poda es parametrizado por $\alpha \geq 0$, conocido como el hiperparámetro de complejidad. Este hiperparámetro es usado para definir el costo de la complejidad de un árbol T, y está dado por:

$$R_{alpha}(T) = R(T) + \alpha|\hat{T}| \tag{27}$$

Donde $\hat{T}$ es el número de nodos terminales en T. $R(T)$ es definido como el total de clasificaciones incorrectas en los nodos terminales de T. Conforme α crece, es mayor la poda de los árboles y, por consiguiente, es menor el sobreajuste (pero decrece la precisión).

Los hiperparámetros de α, la cantidad de árboles para generar el promedio de las predicciones y m, son los hiperparámetros por optimizar para este modelo.

3.2.5.4. Modelo de potenciación de gradiente extrema(XGBoost)

Este modelo, al igual que el de RF, está conformado de árboles de decisión. La diferencia entre ambos radica en el hecho que esta técnica, en lugar de implementar *bagging* como método de agregación, utiliza el método de potenciación.

La técnica de potenciación funciona de forma similar al *bagging*, con la excepción de que los modelos se construyen de forma secuencial. Potenciación involucra combinar la respuesta de un número de modelos $\hat{f}^1(x), ..., \hat{f}^B(x)$, con el propósito de obtener una repuesta compuesta. A diferencia del *bagging*, cada modelo es ajustado usando una versión modificada del grupo de datos original y no en un grupo obtenido mediante remuestreo (*bootstrap*). En vez de ajustar cada modelo con una partición de los datos, en este caso cada modelo se construye mediante un sistema de pesos, en el cual se les otorga cada vez más importancia a las observaciones clasificadas de manera incorrecta. Nótese que, a diferencia del *bagging*, en potenciación la construcción del modelo depende fuertemente de los modelos que fueron previamente ajustados.[Gareth J, et al. (2013)]

Las predicciones de todos los clasificadores $\hat{f}(x)$ se combinan según la cantidad mayoritaria de votos ponderados, para producir la predicción final, determinada con base en la siguiente fórmula.

$$f(x) = \sum_{k=1}^{k} a_k f^k(x) \tag{28}$$

Donde $a_1, a_2, ..., a_k$ se calculan mediante el algoritmo de potenciación.[Hastie T. et al (2008)]

Las modificaciones que se efectúan a cada modelo durante la potenciación consisten en aplicar pesos $w_1, ..., w_n$ a cada observación en el grupo de entrenamiento. Al inicio, todos los pesos se fijan en $1/N$, para asignar un peso igual a cada observación. Para cada modelo sucesivo, m (donde $m = 1, 2, ..., M$), los pesos de las observaciones se modifican, lo que les confiere más peso a las observaciones que fueron clasificadas erróneamente, mientras que los pesos se disminuyen en las observaciones que fueron clasificadas de manera correcta. De esta manera, cada clasificador es forzado a enfocarse en las observaciones que fueron clasificados previamente de manera incorrecta en la secuencia.[Hastie T. et al (2008)]

Dados los árboles de decisión, se define la función objetivo por optimizar para un conjunto de

parámetros θ como:

$$Obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{k} \omega \hat{f}_k \tag{29}$$

Donde $l(y_i, \hat{y}_i)$ representa la función de pérdida que compara el valor verdadero con el predicho. Por su parte, $\omega \hat{f}_k$ es el término de regularización que penaliza el modelo utilizado para evitar el sobreajuste.
Un problema importante con este modelo es que tiene a sobre ajustar muy rápido los datos de entrenamiento. Una estrategia para combatir este problema, es mediante la incorporación de un hiperparámetro que *retrase* el proceso de entrenamiento. A este hiperparámetro se le llama *tasa de aprendizaje*. Al igual que en el modelo de bosques aleatorios, los hiperparámetros seleccionados para el análisis de complejidad de este modelo fueron la cantidad de árboles usados en el proceso de potenciación, el hiperparámetro de complejidad α descrito en la sección 3.2.5.3 y la tasa de aprendizaje.

3.2.5.5. Validación de modelos

Antes de seleccionar el modelo *ideal* para abordar el problema de clasificar retroalimentación de clientes, primero se debe tener claro qué se considera como *ideal* dentro del contexto del problema.

Puesto que el objetivo general para Huli, al implementar un modelo de análisis supervisado, es reducir el tiempo requerido durante la tarea de etiquetado de nueva retroalimentación de clientes, en este trabajo se quiere implementar el modelo que etiquete correctamente la mayor cantidad de comentarios.

Por lo anterior, se seleccionó como métrica para comparar los resultados de los modelos la *precisión global*. La precisión global de un modelo se puede definir como:

$$Precisión_i = \frac{ClasificadosCorrectamente_i}{ClasificadosCorrectamente_i + ClasificadosIncorrectamente_i} \tag{30}$$

Además, con el propósito de validar los modelos, se utilizó *validación cruzada* con k grupos. La validación cruzada simple es el método más sencillo y más usado para estimar la predicción del error. Este método utiliza una parte de los datos para entrenar el modelo y otra parte para probarlo. [Hastie T. et al (2008)].

Una variación de validación cruzada simple es la validación cruzada con k grupos. Esta técnica parte los datos en k grupos del mismo tamaño. Usa el $k - esimo$ grupo para validar los datos del modelo y los otros $k - 1$ grupos para entrenarlo. Esto se hace para $k = 1, 2, ..., k$, y se combinan los resultados para estimar los resultados de la predicción.[Hastie T. et al (2008)]

En este trabajo, se usó $k = 10$. El proceso de validación cruzada puede repetirse más de una vez, para tener una mayor certeza de la estimación del error. En este caso, durante el ajuste de los hiperparámetros de los modelos, se repitió la validación cruzada con diez grupos cinco veces, mientras que, para la comparación y selección del modelo final, se repitió la validación cruzada con diez grupos diez veces. Ya que se está usando validación cruzada con k grupos y h repeticiones, la precisión del modelo i se calcula como:

$$Precisión_i = \frac{\sum_h \frac{\sum_k Precisión_{kh}}{k}}{h} \tag{31}$$

Al final, para el modelo con la mayor precisión, esta se calcula por clase y por la matriz de confusión. La matriz de confusión es una tabla que muestra, para cada categoría por predecir, los resultados predichos correctamente. Esta tabla permite calcular la precisión en la clase j, en el modelo i, mediante:

$$Precisión_{ji} = \frac{ClasificadosCorrectamente_{ji}}{ClasificadosCorrectamente_{ji} + ClasificadosIncorrectamente_{ji}} \tag{32}$$

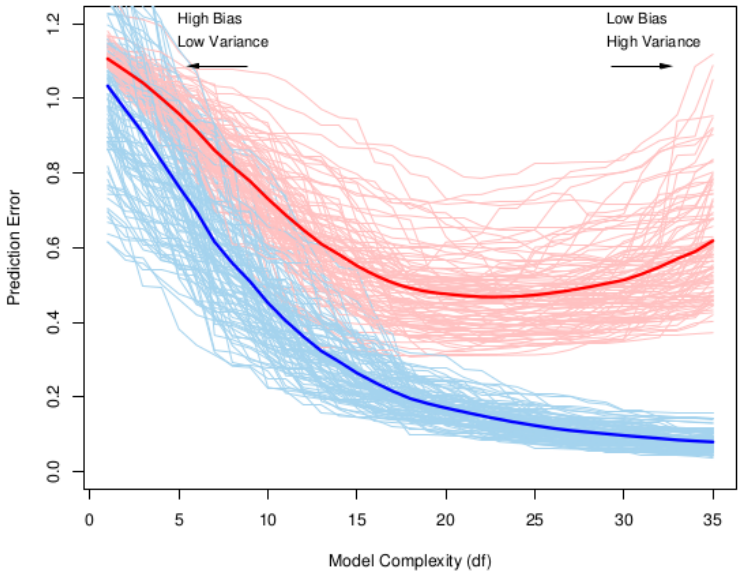
Para estabilizar la precisión por clase, se usó validación cruzada, para lo cual se realizaron diez repeticiones. Además, se calculó la matriz de confusión para cada repetición y, finalmente, se calculó la moda para cada elemento de la matriz.

3.2.5.6. Análisis de complejidad

Si bien la precisión global es una métrica que permite comparar modelos, no se debe olvidar que la selección del mejor modelo también tiene que considerar la complejidad de este.

[Hastie T. et al (2008)] menciona que, conforme un modelo se vuelve más complejo, se usan cada vez más las relaciones intrínsecas de los datos de entrenamiento; en consecuencia, el modelo es capaz de predecir muy bien los datos de entrenamiento(sobreajuste), pero predice muy mal los datos en el grupo de pruebas (el error de las predicciones en diferentes grupos de pruebas va a ser muy variable). Por otro lado, si el modelo es poco complejo y no representa suficientemente bien las relaciones de los datos, los resultados van a presentar también un error alto: se predicen muchas observaciones de manera errónea.

Figura 6: Problema de complejidad en los modelos



Fuente: Hastie et al (2008), pag 220

Por ejemplo, en la figura 6 se puede apreciar el resultado de ajustar un modelo variando su complejidad (a partir de muchas repeticiones). Las líneas azules representan los resultados del modelo

en el grupo de entrenamiento; las líneas rojas, los resultados en el grupo de pruebas. Los modelos poco complejos muestran un error muy alto, mientras que los modelos muy complejos producen errores bajos pero muchos ajustes en los datos.

[Hastie T. et al (2008)] menciona que existe un punto medio en la complejidad del modelo. En este punto medio, se representan lo suficientemente bien las relaciones intrínsecas de los datos y se pueden generalizar los resultados.

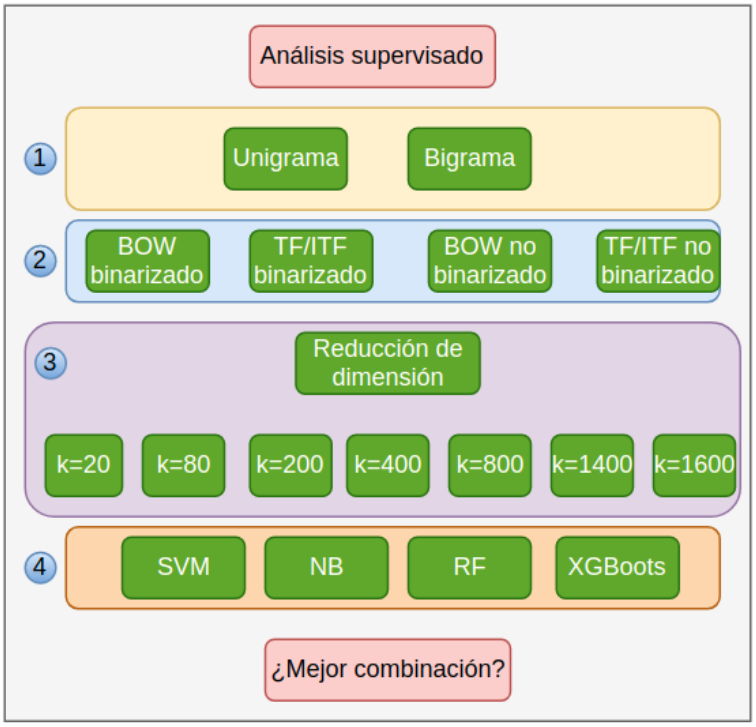
Complejidad no es un término general, sino que depende del modelo que se esté ajustando. Por ejemplo, en el modelo de XGBoost, se observan diferentes variables que aportan a la complejidad de este, como el hiperparámetro de complejidad α descrito en la sección 3.2.5.3 y la tasa de aprendizaje.

Para este trabajo, los hiperparámetros de algunos modelos se van a seleccionar tomando en cuenta la relación de precisión global en los grupos de entrenamiento y pruebas. Se va a seleccionar el hiperparámetro del modelo que genere una precisión alta, pero sin realizar un sobreajuste al grupo de entrenamiento. Esto se va a llevar a cabo de manera visual, graficando los resultados del proceso del ajuste de hiperparámetros.

3.2.5.7. Selección del modelo según criterios de análisis

La revisión bibliográfica expuesta por [Santos, R. et al (2019)] en CrowdRE acota el espacio de búsqueda a nivel metodológico para el análisis supervisado de la retroalimentación de clientes. Sin embargo, este no ofrece un resultado evidente sobre la combinación ideal de técnicas y hiperparámetros. Por ende, se realizó un procedimiento en el que, para cada etapa del análisis, se prueban diferentes técnicas. En la figura 7 se muestran las técnicas escogidas en cada etapa del análisis supervisado.

Figura 7: Etapas en el proceso supervisado para retroalimentación de clientes



Fuente: Figura de elaboración propia.

A continuación, se detallan las técnicas para cada etapa.

1. Etapa 1: selección de unigramas o bigramas.
2. Etapa 2: selección de técnica de vectorización.
3. Etapa 3: selección de k ($k = 20, 80, 200, 400, 800, 1400, 1800$) términos por conservar, según la relación con las categorías primarias, para la reducción de la dimensión del vector original.
4. Etapa 4: selección de hiperparámetros para cada uno de los modelos supervisados y selección del modelo ideal.

Diferentes combinaciones de estas técnicas e hiperparámetros pueden mejorar o decrementar el resultado de clasificación. Por lo tanto, dado que de antemano no se cuenta con información de la combinación ideal y el espacio de búsqueda es amplio, se procederá de la siguiente manera:

1. Para cada uno de los modelos supervisados:

    a) Se seleccionan los hiperparámetros por ajustar, así como el conjunto de valores para cada uno de los hiperparámetros.

    b) Se selecciona un valor para cada uno de los hiperparámetros del modelo.

    c) Se selecciona una técnica en cada una de las etapas 1-3 del proceso de análisis supervisado.

    d) Se ajusta el modelo, para lo cual se emplea validación cruzada con diez grupos e inicios aleatorios.

    e) Se guardan los resultados de precisión en cada uno de los grupos de la validación cruzada, así como la precisión media.

    f) Se repiten los pasos b-e hasta que se haya explorado todo el conjunto de valores de los hiperparámetros y cada una de las técnicas de las etapa 1-3.

    g) Se repiten cinco veces los pasos b-f, para tener más repeticiones.

    h) Se analiza la precisión de cada una de las combinaciones, por repetición. Para ello, se seleccionan los hiperparámetros y las técnicas de las etapas 1-3 que produzcan los mejores resultados de precisión y el modelo menos complejo.

2. Una vez que se cuenta con la combinación ideal de cada uno de los modelos, según el paso 1, se realiza el proceso de validación cruzada usando los mismos grupos con cada uno de los modelos. Se repite el experimento diez veces. Las diez repeticiones se completaron con objetivo de seleccionar grupos diferentes durante la validación cruzada.

3. Se analizan los resultados de precisión media para cada uno de los modelos en cada uno de los experimentos. Se selecciona el que genera los mejores resultados, según la métrica de precisión.

El modelo que produzca los mejores resultados en el paso 3 de la metodología anterior se considera como el ideal para este problema y, por consiguiente, se implementa en la aplicación *web* para demostrar los resultados de este trabajo.

Cabe resaltar que, dado que fue necesario calcular 1225400 modelos, se usó un servidor de AWS ml.m4.16xlarge optimizado específicamente para tareas de análisis supervisado. Este computador cuenta con 96 núcleos y 256 GB de memoria RAM, por lo que el ajuste de estos 1225400 modelos tardó alrededor de 6 horas e implicó un costo económico aproximado de $24.

4. CAPÍTULO IV: RESULTADOS

A nivel de resultados, en este apartado se sigue la metodología mostrada en la figura 3 de la sección 3.2. Se presentan primero los resultados del preprocesamiento de la información. Luego, se realiza un análisis descriptivo de la información. Finalmente, se presenta la selección de modelos no supervisados y supervisados.

4.1. Preprocesamiento

En esta sección, se describe el preprocesamiento realizado a la información de retroalimentación de clientes. En primera instancia, se describen los datos y las variables utilizadas en el análisis. En segunda instancia, se exponen las transformaciones hechas al texto de los comentarios.

4.1.1. Descripción de variables

Inicialmente, se tienen 1519 comentarios clasificados en seis categorías del producto. En el cuadro 2, se detalla la cantidad de comentarios por cada una estas categorías. Las categorías de servicio y perfil son las que presentan menos comentarios relacionados, con solo el 15 % combinado. Por su parte, las categorías de información médica, agenda y facturación son los que abarcan la mayor cantidad de comentarios, con un 66 % combinado.

Cuadro 2: Comentarios de clientes por categorías del producto HuliPractice

Categoría	**Cantidad**	**Porcentaje**
Información médica	347	22 %
Agenda	345	22 %
Facturación	343	22 %
Consulta	230	15 %
Servicio	179	11 %
Perfil	75	4 %
Total	**1519**	**100** %

Fuente: Cuadro de elaboración propia.

Además de los comentarios, se consideran dos variables categóricas extras para el análisis: en primer lugar, el tipo de contacto en el CRM y, en segundo lugar, el departamento de la persona que registró el comentario.

4.1.1.1. Tipo de contacto

El tipo de contacto en el CRM corresponde a un proceso de etiquetado de clientes dentro del equipo de ventas. Un *lead*[18] es un posible cliente que mostró interés en el producto. Un *contact*[19] es

[18]Del inglés. En español, *líder*. Se refiere a una oportunidad de venta.
[19]Del inglés. En español, *contacto*.

un cliente que mostró interés en comprar el producto. Un *deal*[20] es un cliente que compró el producto. En el cuadro 3, se puede ver la distribución de comentarios por tipo de contacto.

Cuadro 3: Comentarios por tipo de contacto en el CRM

Categoría	Cantidad	Porcentaje
Contact	1177	77 %
Deal	248	16 %
Lead	94	7 %
Total	**1519**	**100** %

Fuente: Cuadro de elaboración propia.

La mayoría de comentarios corresponden al tipo *contact*, con un 77 %. El tipo *lead* es el menos representado, con solo un 7 %. No se observan valores perdidos para esta variable.

El p-value de la prueba $\tilde{\chi}^2$, utilizado para determinar la relación existente entre la categoría primaria y el tipo de contacto, da un $2{,}36e^{-23}$. Entonces, por un nivel de significancia del 5 %, se rechaza la hipótesis nula de independencia entre las dos variables. Se decide usar esta variable tanto en el análisis supervisado como en el no supervisado de los comentarios.

4.1.1.2. Departamento dentro de la compañía

La retroalimentación de clientes se registra dentro del CRM por un colaborador de la compañía. Como variable para el análisis, se utiliza el identificador de cada una de estas personas. Al ser 37 personas diferentes las que registraron comentarios, se decidió reconvertir esta variable al departamento al que pertenece la persona. Cabe resaltar que el colaborador que registró más comentarios ingresó al CRM 208. Además, los siete colaboradores que reportaron más comentarios ingresaron el 70 % del total.

En el cuadro 4, se especifica la distribución de comentarios por el departamento de la persona que los dejó .

Cuadro 4: Comentarios por departamento dentro de la compañía

Categoría	Cantidad	Porcentaje
Customer Success	1060	69 %
Sales	392	25 %
Other	67	6 %
Total	**1519**	**100** %

Fuente: Cuadro de elaboración propia.

El 69 % de los comentarios corresponden a colaboradores del departamento de *customer success*[21]. Esto se debe a que las personas de este departamento permanecen en constante comunicación con los

[20] Del inglés. En español, *trato*.
[21] Del inglés. En español, *éxito del cliente*

clientes; por ende, son las que reciben la mayor cantidad de comentarios. Además, el departamento de *sales*[22] fue el segundo departamento con mayor cantidad de comentarios, con un 25 %. Al estar las otras personas distribuidas en departamentos sin demasiada participación, se decidió etiquetarlas como *other*[23].

El p-value de la prueba $\tilde{\chi}^2$, para ver la relación entre la categoría primaria y el departamento, dio como resultado un $6{,}60e^{-64}$. Por consiguiente, por un nivel de significancia del 5 %, se rechaza la hipótesis nula de independencia entre las dos variables. Se decide usar también esta variable tanto en el análisis supervisado como en el no supervisado de los comentarios.

4.1.2. Modificaciones al texto original

Como se mencionó en la sección 3.2.1, antes de poder usar la retroalimentación de los clientes, dentro del análisis supervisado o no supervisado, se tiene que transformar y preprocesar el texto. A continuación, se describen los principales resultados de este proceso.

4.1.2.1. Valores extremos

Se dividió cada uno de los 1519 comentarios en las palabras que lo conforman(tokens). Se contó la cantidad de tokens por comentario. La media de tokens por comentario es de 25, con una desviación estándar de 28. El comentario más grande comprende 311 palabras; el más pequeño, una. El 75 % de los comentarios tiene 33 tokens o menos.

Se calculó el *zscore* para cada uno de los comentarios y se dejaron aquellos cuya cantidad de tokens se encontrara a menos de tres desviaciones estándares de la media. Esto redujo la cantidad de comentarios a 1492 (se dejaron por fuera 27 comentarios).

4.1.2.2. Corrección gramatical y preprocesamiento

Se usó el paquete estadístico Huspell[24] para conocer cuáles correcciones gramaticales debían realizarse en el texto de comentarios. Se encontraron 854 correcciones. Muchas de estas correcciones correspondían a nombres propios no registrados en el paquete, como el nombre de colaboradores u otras compañías.

Se llevó a cabo un proceso donde las palabras mal escritas se corrigieron y todas las palabras no identificadas por el paquete, pero bien escritas, se conservaron. Cabe resaltar que, de las 854 correcciones que se realizaron, 504 palabras aparecían más de una vez dentro del texto.

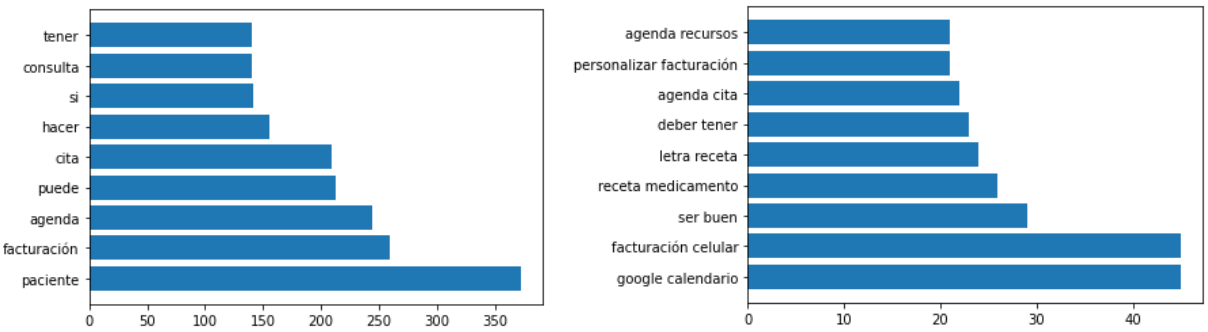
Finalmente, se aplicaron todas las transformaciones textuales descritas en la sección 3.2.1. Estas transformaciones, junto con la corrección ortográfica y la eliminación de valores extremos, produjo que la media de tokens por comentario bajara a 23, con una desviación de 20, lo que significa que se redujo la cantidad de tokens por ser analizados.

[22]Del inglés. En español, *ventas*.
[23]Del inglés. En español, *otro*.
[24]Accedido desde https://pypi.org/project/hunspell/

4.2. Análisis descriptivo

Después del preprocesamiento de la información, se realizó un análisis descriptivo de la retroalimentación. Primero, se calculó la frecuencia de los tokens dentro de los comentarios. En la figura 8 se puede ver la raíz de los diez tokens más frecuentes dentro del texto, en unigramas y bigramas.

Figura 8: Términos más frecuentes en la retroalimentación de clientes



Fuente: Retroalimentación de clientes, Huli(2020)

En cuanto a uni-gramas, la palabra paciente (raíz *pacient*) es la más frecuente, con más de 350 apariciones. Esto tiene mucho sentido, dado que el producto HuliPractice está diseñado para que los doctores administren las interacciones con sus pacientes. Es de esperarse, por lo tanto, que mucha de la retroalimentación vaya dirigida a cómo mejorar esta interacción. El segundo token más frecuente es facturación (raíz *factur*). También esto es de esperarse, ya que un gran porcentaje de los clientes de HuliPractice son doctores costarricenses que realizan su facturación electrónica usando el producto.

En cuanto a los bigramas, las combinaciones de Google calendar (raíz *googl calendar*) y facturación celular (de raíz *factur celular*) son las más frecuentes, con 40 apariciones. Estas, junto a otras como letra receta (raíz *letr recet*), apuntan como tales a funcionalidades que deberían visitarse por los equipos de producto.

4.2.0.1. Nube de palabras

Otra manera de analizar la frecuencia de las palabras dentro del comentario es por medio de una nube de palabras. En la figura 9 se muestra la nube de palabras creada para unigramas y bigramas.

Para unigramas, resaltan funcionalidades particulares del producto, como: facturación, agenda, receta, expedientes, consulta y cita. También, un resultado importante es la aparición de palabras de conexión, como: deber, tener, hacer o necesitar. La aparición de estas palabras brinda mucha información sobre los propósitos de esta retroalimentación, ya que confirma que los comentarios van en la línea de mejoras que debería tener el producto.

A nivel de bigramas es donde se presentan los resultados más útiles. La unión de dos palabras proporciona mucha información sobre funcionalidades del producto de las que se recibe retroalimentación. Específicamente, la receta de medicamentos electrónica adquirió mucha relevancia en el 2020,

Figura 9: Nubes de palabras en la retroalimentación de clientes

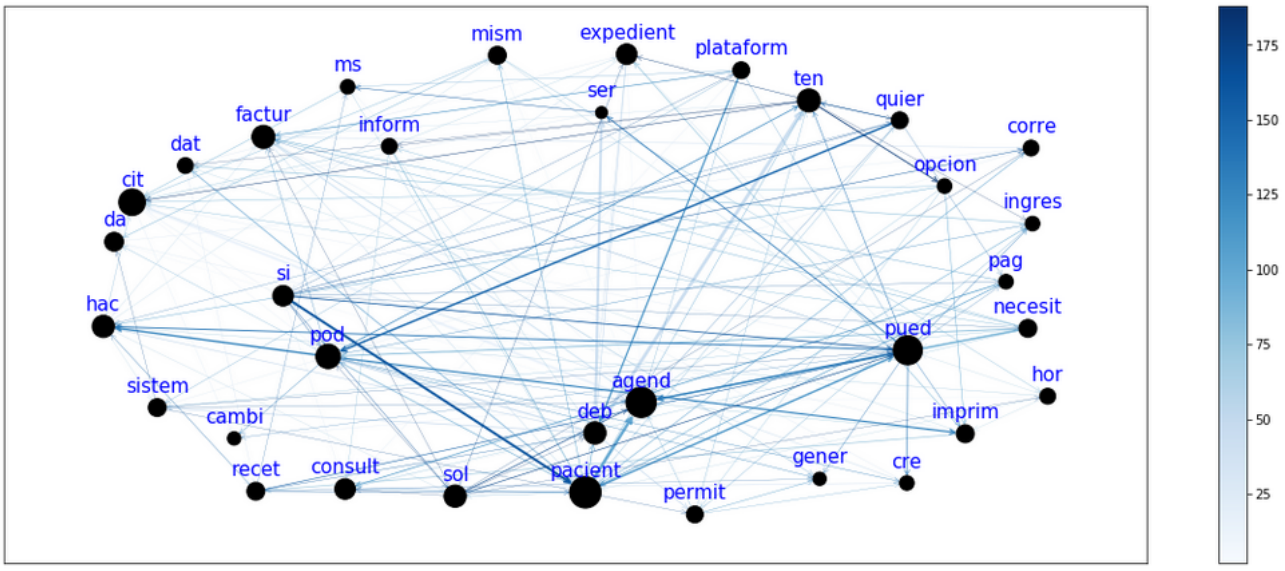
Fuente: Retroalimentación de clientes, Huli(2020)

debido a la pandemia del COVID-19 y a la necesidad de reducir las interacciones presenciales con los doctores.

4.2.0.2. Análisis de redes

Finalmente, para el análisis de redes se emplean únicamente unigramas, ya que a nivel de bigramas las conexiones no brindaban tanta información y eran poco frecuentes. En la figura 10, se presenta el gráfico de las conexiones de los unigramas.

Figura 10: Análisis de redes en la retroalimentación de clientes



Fuente: Retroalimentación de clientes, Huli(2020)

Como se observa en la figura 10, los unigramas con mayor cantidad de conexiones entrantes aparecen en un tamaño mayor. Además, la intensidad de la conexión con otro token varía de intensidad

de azul, según la frecuencia de aparición de las dos palabras juntas.

De acuerdo con lo anterior, se puede señalar que la palabras agendar (raíz *agend*), paciente (raíz *pacient*) y pueden (raíz *pued*) son las más céntricas. De hecho, el token pueden sirve para conectar funcionalidades, por ejemplo, agenda e imprimir, lo que puede interpretarse como la necesidad de poder imprimir la agenda.

El hecho que paciente se presente como un nodo central en la red constituye un buen indicador de que las necesidades de los clientes (doctores) siguen girando en torno a brindarles un mejor servicio a los pacientes.

4.3. Modelos no supervisados

En esta sección, se presentan los resultados de los modelos no supervisados de agrupación. Como se mencionó en la sección 3.2.4.5, para cada uno de los dos modelos no supervisados se va a presentar el ajuste de hiperparámetros y la selección de configuración ideal, según la *medida V*.

4.3.1. K-medias

Para la técnica de agrupación k-medias, se ajustaron 486 modelos, lo que implicó una duración de 40 minutos. Cada modelo se ajusto con diez repeticiones de inicios aleatorios, a fin de estabilizar los resultados. Se realizó el ajuste de los modelos variando los siguientes hiperparámetros:

- Técnica de vectorización: BOW binarizado/no binarizado y TF-IDF binarizado/no binarizado.
- Tokenización: unigramas, bigramas y unigramas/bigramas.
- Reducción de dimensión: se conservaron $k = [20, 80, 200, 400, 800, 1400, 1600]$ tokens, los cuales guardan más relación con las categorías primarias.
- Distancia de cosenos y distancia euclidea.
- La cantidad de grupos g: $g = [2, 3, 4, 5, 6, 7, 8, 9, 10]$.

La combinación de BOW no binarizada, unigramas, $k = 800$, con distancia de cosenos y cinco grupos fue la que produjo mejores resultados, con un $v = 0{,}367$. Conservando seis grupos (la cantidad original de categorías primarias), la medida v presenta un valor de 0,29. Además, al variar una técnica de vectorización a TF-IDF, la medida v genera un valor de $v = 366$.

4.3.2. Agrupación espacial basada en densidad de aplicaciones con ruido (DBScan)

Para la técnica de agrupación DBScan, se ajustaron 486 modelos, lo que representó una duración de 120 minutos. Se realizó un ajuste de los modelos variando los siguientes hiperparámetros:

- Técnica de vectorización: BOW binarizado/no binarizado y TF-IDF binarizado/no binarizado.
- Tokenización: unigramas, bigramas y unigramas/bigramas.
- Reducción de dimensión: se conservaron $k = [20, 80, 200, 400, 800, 1400, 1600]$ tokens que guardan más relación con las categorías primarias.
- Distancia de cosenos y distancia euclidea.
- Distancia ϵ ϵ: $g = [1, 0{,}9, 0{,}7, 0{,}5, 0{,}3, 0{,}1]$.
- La cantidad mínima de puntos $minPts$: $minPts = [5, 10, 20]$

La combinación de BOW no binarizada, unigramas, $k = 800$, con distancia de cosenos, $\epsilon = 0{,}5$ y $minPts = 20$ fue la que produjo los mejores resultados, con un $v = 0{,}30$.

4.3.3. Selección del mejor modelo

En la figura 11, se puede ver la comparación de los dos modelos no supervisados en sus configuraciones ideales. El modelo de k-medias produce mejores resultados en la medidas de homogeneidad y completitud. Por ende, la medida v para el modelo de k-medias (0,36) es superior a esa misma medida para el modelo DBScan (0,30). Lo anterior se puede interpretar de la siguiente manera: para estos datos, las agrupaciones formadas con el modelo de k-medias se parecen más a las agrupaciones verdaderas.

Figura 11: Comparación de modelos no supervisados

Fuente: Retroalimentación de clientes, Huli(2020)

Finalmente, en el cuadro 5, se presenta el resultado de las agrupaciones para los cinco grupos del modelo de k-medias. En el grupo 1, predominan los comentarios de facturación, mientras que en el grupo 4 predominan los resultados de agenda. Por su parte, en el grupo 5 priman los resultados de información médica, pero estos se basan en solo 17 comentarios de consulta. Los comentarios de perfil quedaron distribuidos en los 5 grupos, con mayor presencia en el grupo 2. Por último, los comentarios de servicio muestran más presencia en los grupos 3 y 4.

Cuadro 5: Resultados de agrupación para k-medias con 5 grupos

	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5
Inf. médica	4	27	45	93	166
Agenda	6	94	9	232	1
Facturación	211	30	42	39	16
Consulta	1	1	59	16	149
Servicio	3	38	64	64	27
Perfil	0	28	8	17	22

Fuente: Retroalimentación de clientes, Huli(2020)

4.4. Modelos supervisados

Como se mencionó en la sección 3.2.5.7, para cada uno de los cuatro los modelos supervisados se va a presentar el ajuste de hiperparámetros y el análisis de complejidad. Al final, se comparan los cuatro modelos según su configuración ideal, con el objetivo de seleccionar el mejor para este contexto.

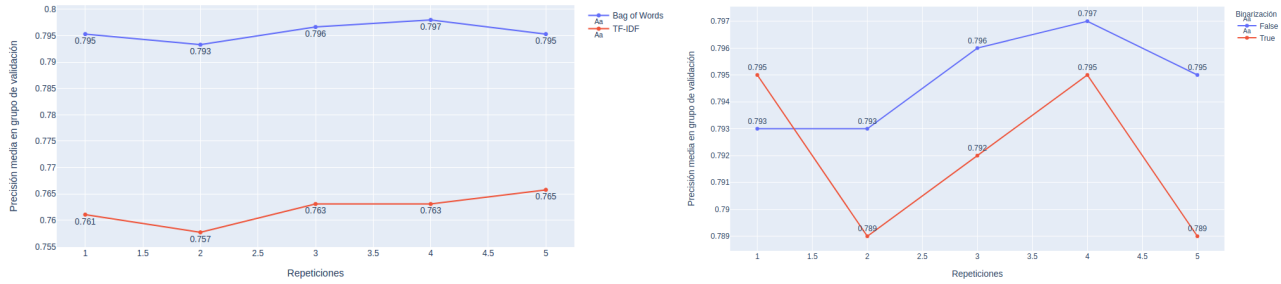
4.4.1. Modelo ingenuo de Bayes(NB)

Para esta técnica, se ajustaron en total 74600 modelos, lo que implicó una duración de 8 minutos y 45 segundos. Se realizaron ajustes con base en las siguientes variaciones de los hiperparámetros:

- Técnica de vectorización: BOW binarizado/no binarizado y TF-IDF binarizado/no binarizado.
- Tokenización: unigramas, bigramas y unigramas/bigramas.
- Reducción de dimensión: se conservaron $k = [20, 80, 200, 400, 800, 1400, 1600]$ tokens, los cuales guardan más relación con las categorías primarias.
- El hiperparámetro α del modelo NB: $\alpha = [0,0{,}1, 0{,}2, 0{,}3, 0{,}4, 0{,}5, 0{,}6, 0{,}7, 0{,}8, 0{,}9, 1]$.

A nivel de técnicas de vectorización, en la figura 12 se puede observar que la técnica de BOW no binarizada fue la que produjo los mejores resultados medios en las cinco repeticiones. Hubo una repetición en la que la versión binarizada de BOW generó mejores resultados; no obstante, por consenso de las cinco repeticiones, se concluye que la versión no binarizada es la mejor.

Figura 12: Comparación de técnicas de vectorización NB



Fuente: Retroalimentación de clientes, Huli(2020)

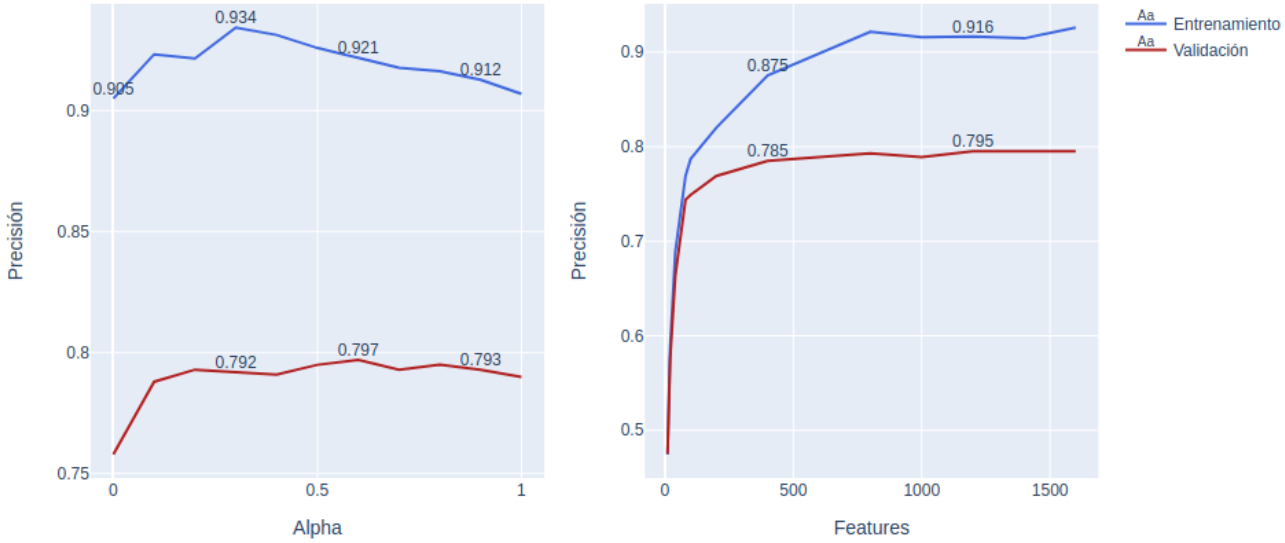
En cuanto a métodos de tokenización, el modelo que emplea unigramas produce mejores resultados en las cinco repeticiones. Este método mostró una precisión media combinada de 79,7 %, contra 78,1 % del modelo con unigramas/bigramas y 61,1 % del modelo que utiliza solo bigramas.

Finalmente, los hiperparámetros por optimizar para este modelo son el hiperparámetro de suavizamiento α y la cantidad de k términos conservados durante el proceso de reducción de dimensión.

Durante el ajuste del modelo, al variar los hiperparámetros de α y k, se obtuvo que $k = 1600$ y $alpha = 0{,}7$ generaban los mejores resultados, por consenso de las cinco repeticiones. Sin embargo, al

observar los resultados del análisis de complejidad en la figura 13, se aprecia que $k = 1200$ y $alpha = 0{,}4$ son más adecuados. Los resultados de precisión media son similares a los que produjeron los mejores resultados para estos hiperparámetros, pero el modelo es más simple (menos suavizamiento del modelo y un vector más reducido), lo que evita el sobreajuste de los datos de entrenamiento.

Figura 13: Análisis de complejidad para α y k, NB



Fuente: Retroalimentación de clientes, Huli(2020)

Se concluye que el modelo de NB con BOW no binarizado, unigramas, que conserva los 1200 términos que guardan más relación con las categorías primarias y un hiperparámetro de suavizamiento de 0,4 es el más adecuado para comparar con los otros modelos de este proyecto.

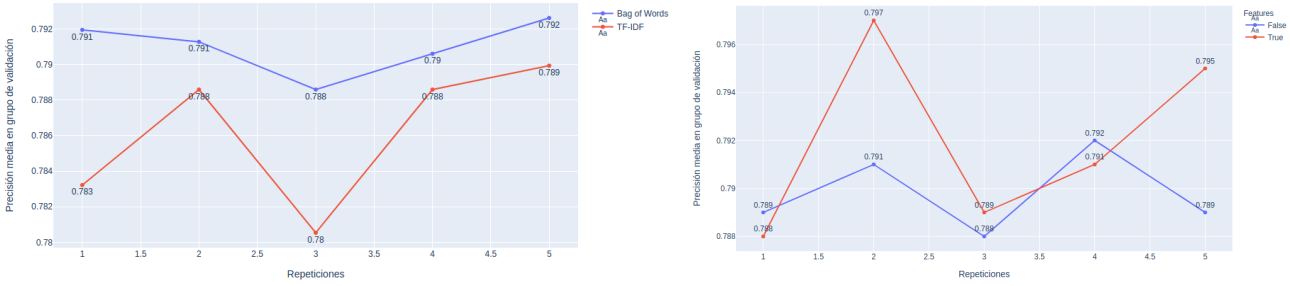
4.4.2. Modelo de bosques aleatorios(RF)

Para la técnica de bosques aleatorios, se ajustaron un total de 352800 bosques, lo que representó una duración de 70 minutos. Se realizaron ajustes con base en las siguientes variaciones de los hiperparámetros:

- Técnica de vectorización: BOW binarizado/no binarizado y TF-IDF binarizado/no binarizado.
- Tokenización: unigramas, bigramas y unigramas/bigramas.
- Reducción de dimensión: se conservaron $k = [20, 80, 200, 400, 800, 1400, 1600]$ tokens que guardan más relación con las categorías primarias.
- La cantidad de árboles: $[20, 40, 80, 100, 200, 300, 500]$.
- El calculo de m: $\sqrt{p}$ y $\log p$
- El hiperparámetro α de complejidad: $[0{,}0, 0{,}001, 0{,}002, 0{,}003, 0{,}004, 0{,}005]$.

A nivel de técnicas de vectorización, en la figura 14 se puede observar que la técnica de BOW no binarizada fue la que demostró los mejores resultados medios en las cinco repeticiones. En este caso, la binarización o no binarización no mostró diferencias muy amplias, ya que en dos de las cinco repeticiones BOW binarizado produjo mejores resultados. Sin embargo, por consenso en las cinco repeticiones, se asume la versión no binarizada como la mejor.

Figura 14: Comparación de técnicas de vectorización RF



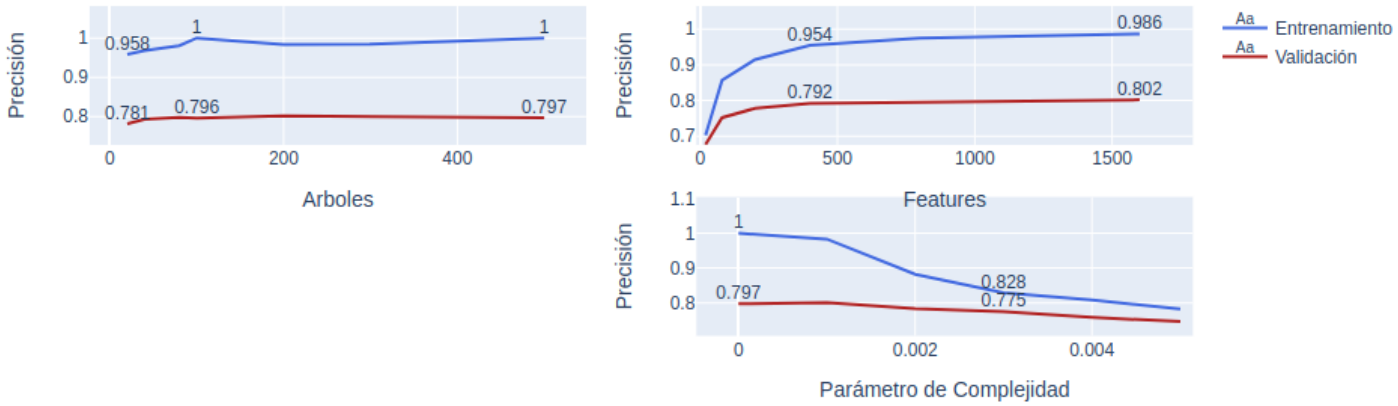
Fuente: Retroalimentación de clientes, Huli(2020)

También para este modelo, el uso de unigramas generó mejores resultados. Esta técnica de tokenización mostró una precisión media de 79,3 % en las cinco repeticiones, frente a 77,5 % obtenido al usar bigramas.

En relación a la técnica para el calculo de m, usar $\log p$ tuvo mejores resultados medios de precisión media con un 79,3 %, mientras que $\sqrt{p}$ tuvo un 78,9 %.

Durante el ajuste del modelo, al variar los hiperparámetros de α, k y la cantidad de árboles, se obtuvo que: $k = 1600$, $alpha = 0{,}0$ y 200 árboles producían los mejores resultados por consenso de las cinco repeticiones. La figura 15 permite apreciar que, si bien estos hiperparámetros aseguran los mejores resultados, también provocan que el modelo sea más complejo.

Figura 15: Análisis de complejidad para α, k y la cantidad de árboles, RF.



Fuente: Retroalimentación de clientes, Huli(2020)

Con respecto a la cantidad de árboles, se puede ver también en la figura 15 que aumentar la cantidad a más de 200, además de aumentar el tiempo de ajuste del modelo, no aumentan significativamente los resultados, por ende, se decidió conservar 200 árboles.

Por lo anterior, se decidió usar los hiperparámetros de $k = 1200$, $alpha = 0{,}001$, el calculo de m con $\log p$ y 200 árboles. Estos hiperparámetros presentan resultados similares de precisión, pero inciden en que el modelo sea más simple.

Se concluye que el modelo de RF con BOW no binarizado, unigramas, que conserva los 1200 términos que guardan más relación con las categorías primarias, un hiperparámetro de complejidad α de 0,001, el calculo de m con $\log p$ y 200 árboles es el más adecuado para comparar con los otros modelos de este proyecto.

4.4.3. Modelo de máquinas de soporte vectorial(SVM)

Para esta técnica, se realizaron ajustes de 420000 modelos, lo que representó una duración total de 66 minutos. Los ajustes del modelo se aplicaron variando los siguientes hiperparámetros:

- Técnica de vectorización: BOW binarizado/no binarizado y TF-IDF binarizado/no binarizado.
- Tokenización: unigramas, bigramas y unigramas/bigramas.
- Reducción de dimensión: se conservaron $k = [20, 80, 200, 400, 800, 1400, 1600]$ tokens que guardan más relación con las categorías primarias.
- El hiperparámetro de regularización C: $[4, 2, 1, 0{,}1, 0{,}01, 0{,}001, 0{,}0001]$
- Función de Kernel: lineal, polinomial, sigmoidea y base radial.

En cuanto a técnicas de vectorización, en la figura 16 se detallan los resultados. En este caso, TF-IDF no binarizado obtuvo los mejores resultados, con una precisión media en las cinco repeticiones de 81,3 %. Al igual que en RF, los resultados en cuanto a la binarización o no quedaron divididos. Sin embargo, en tres de las cinco repeticiones, la versión no binarizada obtuvo mejores resultados, por lo que se seleccionó esta técnica.

Figura 16: Comparación de técnicas de vectorización SVM

Fuente: Retroalimentación de clientes, Huli(2020)

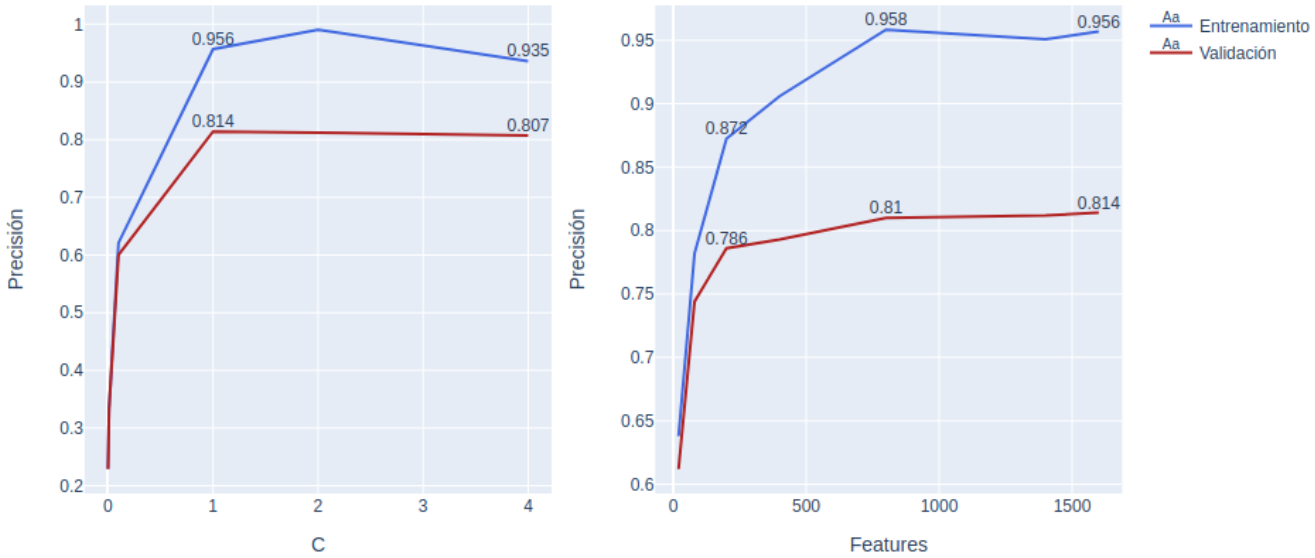
También para esta técnica, el uso de unigramas generó los mejores resultados, con una precisión media de 81,1 %. Sin embargo, la versión donde se usan unigramas y bigramas mostró resultados cercanos, con un 79,9 %.

Además, el kernel lineal fue el que tuvo los mejores resultados con una precisión media en las 5 repeticiones de 81,1 %. Las funciones polinomiales y de base radial tuvieron el segundo y tercer

mejor resultado con una presición media de 77,5 % y 73,5 % respectivamente. Finalmente, la función sigmoidea tuvo los resultados de precisión media más bajos con un 59,5 %.

Los hiperparámetros por optimizar para este modelo fueron el hiperparámetro de regularización C y k. Analizando estos hiperparámetros solo por los valores donde se obtuvieron mejores resultados, se tendrían que conservar $k = 1600$ y $C = 1$ (sin regularización del modelo). No obstante, al observar los resultados de la figura 17, se puede notar que los hiperparámetros de $k = 1200$ y $C = 2$ producen resultados similares, pero con menor ajuste a los datos de entrenamiento.

Figura 17: Análisis de complejidad para C, k, SVM



Fuente: Retroalimentación de clientes, Huli(2020)

Se concluye que el modelo de SVM con TF-IDF no binarizado, unigramas, que conserva los 1200 términos que guardan más relación con las categorías primarias y un hiperparámetro de complejidad de 2 es el más adecuado para comparar con los otros modelos de este proyecto.

4.4.4. Modelo de potenciación de gradiente extrema(XGBoost)

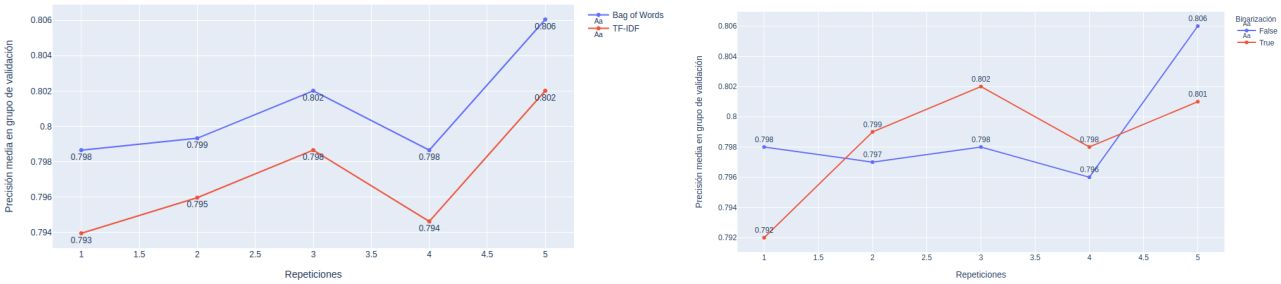
Para esta técnica, se realizaron ajustes de 352800 modelos, lo que representó una duración total de 160 minutos. Los ajustes aplicados se efectuaron variando los siguientes hiperparámetros:

- Técnica de vectorización: BOW binarizado/no binarizado y TF-IDF binarizado/no binarizado.
- Tokenización: unigramas, bigramas y unigramas/bigramas.
- Reducción de dimensión: se conservaron $k = [20, 80, 200, 400, 800, 1400, 1600]$ tokens que guardan más relación con las categorías primarias.
- La cantidad de árboles: $[20, 40, 80, 100, 200, 300, 500]$.
- Hiperparámetro α de complejidad: $[0{,}0, 0{,}001, 0{,}002, 0{,}003, 0{,}004, 0{,}005]$.

- La tasa de aprendizaje: [0,3, 0,2, 0,1, 0,01, 0,001]

Nuevamente, la técnica de vectorización numérica BOW fue la que obtuvo los mejores resultados de precisión, con una media de 79,9 %. Además, la versión binarizada generó mejores resultados en tres de las cinco repeticiones. En la figura 18 se pueden muestran estos resultados.

Figura 18: Comparación de técnicas de vectorización XGBoost

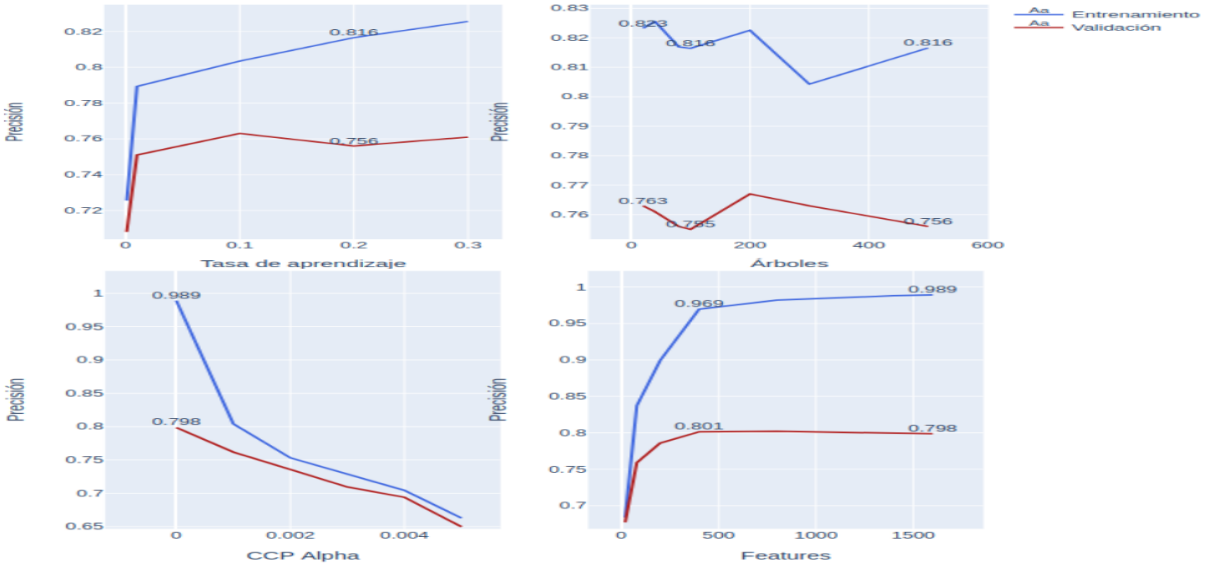


Fuente: Retroalimentación de clientes, Huli(2020)

La tokenización con unigramas también produjo los mejores resultados medios de precisión, con 79,7 %, frente a los 78,1 % de la versión de unigramas/bigramas y los 59,9 % con solo bigramas.

Los hiperparámetros por optimizar para este modelo fueron la cantidad de árboles por usar en el proceso de potenciación, la cantidad de términos por conservar durante la reducción de dimensiones, el hiperparámetro de complejidad α y la tasa de aprendizaje. Si nos guiáramos solo por los valores donde se obtuvieron los mejores resultados de precisión, se tendrían que conservar $k = 1600$, 200 árboles, $\alpha = 0$ (sin hacer poda de nodos) y una tasa de aprendizaje de 0.3. Sin embargo, observando los resultados de la figura 19, con $k = 1200$, 200 árboles, $\alpha = 0{,}001$ y una tasa de aprendizaje de 0.1, se obtiene un modelo más simple y resultados muy similares de precisión media.

Figura 19: Análisis de complejidad para hiperparámetros de XGBoost



Fuente: Retroalimentación de clientes, Huli(2020)

Finalmente, se concluye que el modelo de XGBoost con BOW binarizado, que utiliza unigramas, que conserva los 1200 términos que guardan más relación con las categorías primarias, un hiperparámetro de complejidad α de 0,001, 200 árboles y una tasa de aprendizaje de 0.1, es el más adecuado para comparar con los otros modelos de este proyecto.

4.4.5. Selección del mejor modelo

Con las versiones optimizadas de los cuatro modelos, se realizó una validación cruzada con diez grupos y diez repeticiones. Se ajustó cada modelo con los mismos grupos de entrenamiento y pruebas. En la figura 20 se pueden ver los resultados de las precisiones medias en cada validación cruzada por repetición.

Figura 20: Comparación de modelos supervisados

Fuente: Retroalimentación de clientes, Huli(2020)

El modelo de SVM fue el que produjo los mejores resultados de precisión media en las diez repeticiones, con 81,0 %, seguido del modelo XGBoost, con 79,7 %, y del modelo de NB, con 79,0 %. El modelo con los resultados más deficientes fue el modelo de RF, con 77,5 %.

La moda de la matriz de confusión para este modelo en las diez repeticiones se puede visualizar en el cuadro 6. La precisión por clase se detalla a continuación: información médica 90,0 %, agenda 83,3 %, facturación 90,0 %, consulta 78,1 %, perfil 100 % y servicio 64,0 %. Cabe resaltar que los resultados para clase *perfil* se justifican por el hecho de que solo hubieron dos observaciones en la muestra, las cuales se predijeron correctamente.

Cuadro 6: Moda de matriz de confusión para SVM en diez repeticiones

	Inf. médica	Agenda	Facturación	Consulta	Perfil	Servicio
Inf. médica	30	0	0	0	0	1
Agenda	1	15	1	1	0	1
Facturación	0	0	30	2	0	1
Consulta	1	2	1	25	0	2
Servicio	0	1	0	1	2	0
Perfil	1	0	1	3	0	9

Fuente: Retroalimentación de clientes, Huli(2020)

5. CAPÍTULO V: CONCLUSIONES Y DISCUSIÓN

El presente trabajo abordó el tema de agrupación y predicción automática de retroalimentación de clientes como mecanismo para agregar valor a las decisiones de priorización de tareas en la compañía Huli. El análisis descriptivo de la retroalimentación proporciona evidencia de que hay que aumentar la cobertura de clientes en el proceso de retroalimentación. Los 1519 comentarios captados en el periodo estudiado solo representan el 5 % de los clientes; es decir, existen clientes que no están siendo escuchados. Además, un solo departamento de la compañía registró el 70 % de los comentarios, y 7 colaboradores registraron el 70 % de la retroalimentación. En la compañía laboran cien colaboradores y existen seis departamentos, por lo que se requiere aumentar la cobertura de la iniciativa.

Se registró un 25 % de comentarios de más de 33 palabras y 27 comentarios con más 100 palabras. Examinando estos comentarios, se nota que el texto resume varias dolencias del cliente, las cuales apuntan a diferentes secciones del producto. Se tiene que capacitar a los colaboradores sobre cómo registrar estos comentarios. Para tareas de clasificación, es mejor que cada comentario se refiera a la dolencia del cliente en una sola parte del producto.

Por otra parte, la nube de palabras y el análisis de frecuencia en bigramas apunta a cuatro funcionalidades especificas del producto HuliPractice que deberían revisarse, a saber: la letra de la receta dentro de la sección de consulta médica, la integración de la agenda con Google Calendar, la implementación de facturación electrónica para ser accedida desde el celular y la agenda electrónica por recursos.

A nivel de modelos no supervisados, se escoge el modelo de k-medias con BOW no binarizada, unigramas, $k = 800$, distancia de cosenos y cinco grupos. Este obtuvo los mejores resultados, con un $v = 0{,}367$. Dicho modelo permite agrupar relativamente bien las clases de agenda, facturación e información médica; sin embargo, no agrupa tan bien las clases de consulta, perfil y servicios.

La diferencia en la medida v de este método de agrupamiento, en relación con DBScan, es de apenas 0,07. El método de DBScan tarda en promedio diez veces menos encontrando las agrupaciones que k-medias. De igual manera que para el análisis supervisado, si se quisiera implementar este modelo en un ambiente de alta demanda, se tendrían que equilibrar estos resultados con los tiempos de agrupamiento.

Además, el algoritmo de k-medias, si se mantiene la cantidad de grupos fija en 6 (la cantidad inicial de categorías primarias), produce peores resultados que con cinco grupos (0,29 vs 0,36). Este

hecho, junto a los resultados de clasificación tan bajos de la clase perfil, apuntan a que hay que revisitar la clasificación inicial de los comentarios.

A nivel del análisis supervisado de la información, no se puede asumir una técnica de vectorización numérica como la mejor en la clasificación. En tres de los cuatro modelos supervisados, BOW generó buenos resultados; sin embargo, para la técnica con los mejores resultados(SVM), TF-IDF fue mejor. Lo misma conclusión puede extenderse a la binarización o no de la técnica de vectorización.

El modelo de SVM con TF-IDF no binarizado, con unigramas, que conserva los 1200 términos que guardan más relación con las categorías primarias, un hiperparámetro de complejidad de 2 y con kernel lineal fue el que obtuvo los mejores resultados de precisión, con un 81,0 %. Esto significa que, de los 1492 comentarios finales, se hubieran podido clasificar correctamente 1209. Si revisando cada comentario se durara un minuto, la tarea automática de clasificación de comentarios le estaría ahorrando a la compañía aproximadamente 20 horas. Aun así, si se usara el modelo con los peores resultados(RF), la compañía se hubiera ahorrado aproximadamente 18 horas.

Estos resultados concuerdan con los encontrados por [Santos, R. et al (2019)] en su revisión bibliográfica. El modelo de SVM es de los modelos que más se citan en esta revisión, y los resultados de precisión obtenidos (81,0 %) están entre los reportados en proyectos similares (75,0 % y 85,0 %).

Las diferencias de precisión media de SVM con los otros modelos fue de 4 %. Este 4 % representa 60 de los 1419 comentarios existentes. Las diferencias en tiempo de ajuste de los cuatro modelos varía significativamente. NB duró en promedio 50 veces menos en tareas de predicción que la técnica de SVM. Si se tuviera que implementar un modelo en un ambiente de alta demanda, se tendrían que equilibrar los resultados de predicción con los resultados de tiempo de ejecución de la técnica.

Los resultados de precisión para los comentarios que apuntan a funcionalidades de los productos consulta, información médica, facturación, consulta y perfil fueron en promedio de 90 %. La categoría de servicio fue la que presentó los resultados más deficientes, con 64 %. Esto puede ser un indicador de que los comentarios dentro de esta categoría son muy ambiguos y, por ende, tienen que visitarse, ya sea para dividirlos en más categorías o para no tomarlas en cuenta para el análisis.

Los algoritmos usados en este trabajo dejan de lado la relación contextual de los comentarios. Dentro de esta estructura contextual, existe información que puede resultar muy valiosa para la compañía, como intenciones no directas del cliente o contenido ambiguo difícil de interpretar de manera explícita. Como una etapa posterior a este proyecto, valdría la pena explorar esta misma información desde un punto de vista contextual, para ver que otros resultados aporten más evidencia en el análisis.

Además, algunos de estos comentarios pueden ser usados para conocer el grado de felicidad del cliente con el producto. Un análisis de sentimientos puede arrojar señales que ayuden al equipo encargado de servicio al cliente a identificar cuáles clientes están en riesgo de abandonar la plataforma. Algunas de las técnicas de análisis supervisado expuestas en este mismo trabajo podrían ser usadas para tareas de clasificación de sentimientos.

Analizar los comentarios del 2016 al 2020 asume implícitamente que el producto Hulipractice no ha cambiado en el tiempo. La realidad es que el producto ha mejorado de manera continua. Comentarios del 2016 podrían referirse a problemas que en el 2020 ya están resueltos. Asimismo, eventos coyunturales, como la pandemia ocasionada por el virus COVID-19 en el 2020, podrían ocasionar que durante períodos los comentarios de clientes se enfoquen en una sola línea del producto. Por lo tanto, valdría la pena realizar este mismo análisis por periodos, a fin de conocer si los resultados

son consistentes. Aun más, se podrían identificar los comentarios emitidos durante estos eventos y analizarlos por separado, para conocer más exactamente las necesidades de los clientes.

Finalmente, durante este trabajo se aplicaron y validaron técnicas de minería de texto para la agrupación y predicción automática de la retroalimentación de clientes para el servicio web Huli-Practice, por lo que se cumplió con el objetivo general y con los objetivos específicos planteados inicialmente.

Referencias

[Eduard C et al. (2017)] Eduard C. Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitza Guzman, et al. (2017) *The crowd in requirements engineering: The landscape and challenges.* IEEE Software, 34(2):44–52.

[UserVoice. (2015)] UserVoice. (2015) *Building a Feedback Machine. Inbound Content Marketing Management, UserVoice.*

[Jacqueline S. (2018)] Jacqueline Schowalter, Eduard C. Groen, Sylwia Kocpzy´nska, Svenja Polst, and Sadaf Alvani. (2018) *Is there really a need for using NLP to elicit requirements? A benchmarking study to assess scalability of manual analysis.* In Klaus Schmid and Paolo Spoletini, editors, Requirements Engineering – Foundation for Software Quality (REFSQ) Joint Proceedings of the Co-Located Events, CEUR Workshop Proceedings 2075, 2018.

[Gurusamy et al. (2014)] Gurusamy, V y Kannan, S (2014) *Preprocessing Techniques for Text Mining. Universidad de Madurai Kamaraj.*

[Lowdermilk et al. (2017)] Lowdermilk, T y Rich, J (2017) *Converting Converting Customer Feedback into Successful Products. O'Reilly Media, Inc.*

[M. Glinz. (2019)] M. Glinz (2019) *CrowdRE: Achievements, Opportunities and Pitfalls. IEEE 27th International Requirements Engineering Conference Workshops (REW), Jeju Island, Korea (South), 2019, pp. 172-173, doi: 10.1109/REW.2019.00036.*

[Groen et al. (2015)] E. C. Groen, J. Doerr and S. Adam (2015) *Towards crowd-based requirements engineering – a research preview. REFSQ 2015, pp. 247-253, 2015.*

[Koch. (2018)] Koch, Eduard, Groen and Matthias. (2018). *Crowd-based Requirements Engineering (Whitepaper).*

[Mahmood H et at 2017] Mahmood Hosseini, Keith Phalp, Jacqui Taylor, and Raian Ali. (2017) *The Four Pillars of Crowdsourcing: a Reference Model. The Four Pillars of Crowdsourcing: a Reference Model Mahmood Hosseini, Keith Phalp, Jacqui Taylor, and Raian Ali Faculty of Science and Technology Bournemouth University, UK*

[Van Vliet. M et al (2019)] Martijn van Vliet1, Eduard C. Groen1,2Q, Fabiano Dalpiaz1, andSjaak Brinkkemper (2019). *Identifying and Classifying User Requirements inOnline Feedback via Crowdsourcing. raunhofer IESE, Kaiserslautern, Germany.*

[Castro-Herrera, C. et al(2008)] Castro-Herrera, C., Duan, C., Cleland-Huang, J., Mobasher, B. (2018) *Using data miningand recommender systems to facilitate large-scale, open, and inclusive requirementselicitation processes. In: Proc. of RE, pp. 165–168 (2008)*

[Lim, S.L et al. (2012)] Lim, S.L., Finkelstein, A. (2012) *StakeRare: Using social networks and collaborative fil-tering for large-scale requirements elicitation. IEEE Transactions on Software Engineering 38(3), 707–735 (2012)*

[Snijders, R. et al (2015)] Snijders, R., Dalpiaz, F., Brinkkemper, S., Hosseini, M., Ali, R., Ozum, A. (2015) *REfine: A gamified platform for participatory requirements engineering. In: Proc.of CrowdRE, pp. 1–6 (2015)*

[Santos, R. et al (2019)] Rubens Santos, Eduard C. Groen, Karina Villela. (2019) *An Overview of User Feedback Classification Approaches. Fraunhofer IESE, Kaiserslautern, Germany. (2019)*

[Ananiadou, S (2006)] Ananiadou. S., Keek,D y Tsujii, J (2006). *Text mining and its potential applications in systems biology. Science Direct. doi: http://doi.org/bd8pr8*

[Tufféry, S (2011)] Tufféry, S (2011). *Data mining and statistics for decision making. John Wiley Sons.*

[Manning, C. (2008)] Manning, C. D., Raghavan, P., Schutze, H. (2008). Introduction to Information Retrieval. Cambridge University Press. Doi:0521865719

[Grupta, V. (2009)] Vishal Gupta. (2009). A Survey of Text Mining Techniques and Applications. Lecturer Computer Science Engineering, University Institute of Engineering Technology, Panjab University Chandigarh, India

[Hastie T. et al (2008)] Trevor Hastie, Robert Tibshinari, Jerome Friedman. (2008). The elements of statistical learning. Data mining, Inference and Prediction. Second Edition. Springer

[Qiaozhu, M (2010)] Mei, Qiaozhu. (2010). Contextual text mining. Department of Computer Science. University of Illinois at Urbana-Champaig

[Baeza, R et al (1999)] Ricardo Baeza-Yates, Berthier Ribeiro-Neto. (199). Modern Information retrieval. ACM Press

[Iglewicz, B et al (1993)] Boris Iglewicz and David Hoaglin (1993). The ASQC Basic References in Quality Control: Statistical Techniques, Edward F. Mykytka, Ph.D., Editor.

[Bengfort, B et al (2018)] Benjamin Bengfort, Rebecca Bilbro, Tony Ojeda. Text analysist with Python. O'Reilly.

[Silge, J et al (2017)] Julia Silge and David Robinson. Text Mining with R. A Tidy Approach. O'Reilly.

[Weinberger, K et al (2009)] JKilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola and Josh Attenberg (2009).Feature hashing for large scale multitask learning. Proc. ICML.

[Agresti A, 2013] Allan Agresti, (2013). Categorical Data Analysis. O'Relley. Third edition, pag 18.

[Manning, P et al. (2008)] C.D. Manning, P. Raghavan and H. Schuetze (2008). Introduction to Information Retrieval. Cambridge University Press, pp. 234-265

[Platt, J. (1998)] Platt, John.: Fast Training of Support Vector Machines using Sequential Minimal Optimization, in Advances in Kernel Methods – Support Vector Learning, B. Scholkopf, C. Burges, A. Smola, eds., MIT Press (1998).

[Cristianini, N, et al. (2000)] Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press, Cambridge (2000)

[Gareth J, et al. (2013)] James, Gareth and Witten, Daniela and Hastie, Trevor and Tibshirani, Robert: An Introduction to Statistical Learning. Springer Texts in Statistics, Springer New York (2013)

[MacQueen, J.B. (1967)] MacQueen, J.B. (1967). Some methods for classification and analysis of multivariable observations. In *MSP* (pp. 281-297)

[Braune, C et al, (2015)] Christian Braune, Stephan Besecke, and Rudolf Kruse, (2015). Density Based Clustering: Alternatives to DBSCAN. University Magdeburg, Magdeburg, Germany. Springer International Publishing Switzerland 2015.

[Rosenberg, A et al (2007)] Andrew Rosenberg and Julia Hirschberg, 2007. V-Measure: A conditional entropy-based external cluster evaluation measure.

[Breiman, L. et al (1984)] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, (1984).

[Leonard T. (2018)] Timothy Leonard. Network Data Analysis of Word Graphs With Applications to Authorship Attribution. University of Rhode Island, (2018). https://digitalcommons.uri.edu/theses/1259

[Bonaccorso B. (2018)] Bonaccorso B. Mastering Machine Learning Algorithms. Packt Publising, (2018).

A. Anexos

```
# limpieza inicial
raw_data = raw_data.assign(year=lambda df: df['created_at'].apply(lambda created_at: created_at.year)).assign(month=lambda df: df['created_at'].apply(lambda created_at: created_at.month))

# pruebas de chi-quadrado
chi_res = chi2_contingency(pd.crosstab(raw_data['resource_type'], raw_data['primary']))
print('Chi2 Statistic: {}, p-value: {}'.format(chi_res[0], chi_res[1]))

chi_res = chi2_contingency(pd.crosstab(raw_data['creator_id'], raw_data['primary_category']))
print('Chi2 Statistic: {}, p-value: {}'.format(chi_res[0], chi_res[1]))

# tokenizacion
raw_data['tokens'] = raw_data.apply(lambda row:  word_tokenize(row['comment']), axis=1)
raw_data=raw_data.assign(tokens_len=lambda df: df['tokens'].apply(lambda token: len(token)))

# limpieza gramatical
corrections = pd.DataFrame(columns = ['corrected_token', 'frequency'])
for index, sentence in raw_data.iterrows():
    tokens = sentence['tokens']
    for token in sentence['tokens']:
        token = token.lower()
        token = token.strip()
        if token == "": continue
        if token in string.punctuation: continue

        if not h.spell(token):
            correction = h.suggest(token)
            if len(correction) != 0:
                corrected_token = correction[0]
                if token in corrections.index:
                    corrections.loc[token, 'frequency'] += 1
                else:
                    row = pd.Series({'corrected_token':corrected_token,'frequency':1}, name = token)
                    corrections = corrections.append(row)

# pre-procesamiento
preprocess = ColumnTransformer([
    ('preprocess', Pipeline(steps=[
        ('LowerCaser', LowerCaser()),
        ('PuntuationRemover', PuntuationRemover()),
        ('Striper', Striper()),
        ('TokenCleaner', TokenCleaner()),
        ('WordCorrector', WordCorrector()),
        ('StopWordsRemover', StopWordsRemover()),
        ('SingleLetterRemover', SingleLetterRemover()),
        ('Stemmer', Stemmer()),
    ]), ['tokens']),
```

```
], remainder='passthrough')

proccessed = preprocess.fit_transform(tokenized_comments.reset_index())
```

Listado 1: Pre-procesamiento

```
unigramdist = FreqDist()
bigramfdist = FreqDist()
threeramfdist = FreqDist()

for index, sentence in comments.iterrows():
    unigrams = ngrams(sentence['tokens'], 1)
    bigrams = ngrams(sentence['tokens'], 2)
    trigrams = ngrams(sentence['tokens'], 3)
    unigramdist.update(unigrams)
    bigramfdist.update(bigrams)
    threeramfdist.update(trigrams)

# nube de palabras
words = list(filter_words.keys())
colors = [plotly.colors.DEFAULT_PLOTLY_COLORS[random.randrange(1, 10)] for i in
   range(len(filter_words))]
weights = [x/5 for x in list(filter_words.values())]

data = go.Scatter(x=[random.random() for i in range(len(filter_words))],
                  y=[random.random() for i in range(len(filter_words))],
                  mode='text',
                  text=words,
                  marker={'opacity': 0.3},
                  textfont={'size': weights,
                            'color': colors})
layout = go.Layout({'xaxis': {'showgrid': False, 'showticklabels': False, '
   zeroline': False},
                    'yaxis': {'showgrid': False, 'showticklabels': False, '
   zeroline': False}})
fig = go.Figure(data=[data], layout=layout)

fig.show()

# analisis de redes
G = nx.DiGraph()
G.add_nodes_from(np.unique(np.array([[k[0], k[1]] for k, v in d[0].items()]).
   flatten()))
maxFreq = max([v for k, v in d[0].items()])

for k, v in d[0].items():
    G.add_edge(k[0], k[1], weight=(v/maxFreq)*5)

remove = [node for node,degree in dict(G.degree()).items() if degree < 15]
G.remove_nodes_from(remove)
remove = [node for node,degree in dict(G.degree()).items() if degree == 0]
G.remove_nodes_from(remove)

fig, ax = plt.subplots(figsize=(20, 8))
pos = nx.spring_layout(G, k=2)
weights = [G[u][v]['weight'] for u,v in G.edges()]
```

```
nodeSizes = [v for u,v in G.degree()]
maxSize = max(nodeSizes)
nodeSizes = [(i/maxSize)*600 for i in nodeSizes]
M = G.number_of_edges()
edgeColors = range(2, M + 2)
edgeAlphas = [(5 + i) / (M + 4) for i in range(M)]

nodes = nx.draw_networkx_nodes(G, pos, node_size=nodeSizes, node_color="black")
edges = nx.draw_networkx_edges(
    G,
    pos,
    node_size=nodeSizes,
    arrowstyle="->",
    arrowsize=10,
    edge_color=edgeColors,
    edge_cmap=plt.cm.Blues,
    width=weights,
    ax=ax
)

for key, value in pos.items():
    x, y = value[0], value[1]+0.065
    ax.text(x, y, s=key, horizontalalignment='center', fontsize=15, color='blue')

pc = mpl.collections.PatchCollection(edges, cmap=plt.cm.Blues)
pc.set_array(edgeColors)
plt.colorbar(pc)
```

Listado 2: Análisis descriptivo

```

data = comments
variables = data.drop('primary_category', axis = 1)
response = data['primary_category'].values
tunningRepetitions = 5
experiment=2

# bayes
features = {
    'vec' : {
        'bow': [
            {
                'ngram_range': ((1, 1), (1, 2), (2, 2)),
                'binary': [True, False]
            },
            CountVectorizer()
        ],
        'tf-idf': [
            {
                'ngram_range': ((1, 1), (1, 2), (2, 2)),
                'binary': [True, False]
            },
            TfidfVectorizer()
       ]
    },
    'fs_technique' : {
```

```
        'k-best-chi' : [
            {
                'k': [20, 80, 200, 400, 800, 1400, 1600]
            },
            SelectKBest(chi2)
        ]
    },
    'model' : {
        'bayes': [
            {
                'alpha': [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
            },
            MultinomialNB()
        ],
    }
}

bayes_result=mix_func_repetition(features)
expanded_bayes_results = expand_results(bayes_result, ['alpha'])
expanded_bayes_results.sort_values(by=['rank_test_score'], ascending=True).head()

# arboles aleatorios
features = {
    'vec' : {
        'bow': [
            {
                'ngram_range': ((1, 1), (1, 2), (2, 2)),
                'binary': [True, False]
            },
            CountVectorizer()
        ],
        'tf-idf': [
            {
                'ngram_range': ((1, 1), (1, 2), (2, 2)),
                'binary': [True, False]
            },
            TfidfVectorizer()
       ]
    },
    'fs_technique' : {
        'k-best-chi' : [
            {
                'k': [20, 80, 200, 400, 800, 1400, 1600]
            },
            SelectKBest(chi2)
        ]
    },
    'model' : {
        'forest': [
             {
                 'n_estimators': [20, 40, 80, 100, 200, 300, 500],
                 'criterion': ["gini"],
                 'max_features': ['sqrt','log2'],
                 'ccp_alpha': [0.0, 0.001, 0.002, 0.003, 0.004, 0.005]
             },
            RandomForestClassifier()
```

```
        ],
    }
}

forest_result = mix_func_repetition(features)
expanded_forest_results = expand_results(forest_result, ['n_estimators', '
    ccp_alpha', 'criterion', 'max_features'])
expanded_forest_results.sort_values(by=['rank_test_score'], ascending=True).head
    ()

# maquinas de soporte
features = {
    'vec' : {
        'bow': [
            {
                'ngram_range': [(1, 1),(1, 2),(2, 2)],
                'binary': [True, False]
            },
            CountVectorizer()
        ],
        'tf-idf': [
            {
                'ngram_range': [(1, 1), (1, 2), (2, 2)],
                'binary': [True, False],
            },
            TfidfVectorizer()
       ]
    },
    'fs_technique' : {
        'k-best-chi' : [
            {
                'k': [20, 80, 200, 400, 800, 1400, 1600]
            },
            SelectKBest(chi2)
        ]
    },
    'model' : {
        'svc': [
             {
                 'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
                 'gamma':[1, 0.1, 0.01, 0.001, 0.0001],
                 'C': [1, 0.1, 0.01, 0.001, 0.0001]
             },
            SVC()
        ],
    }
}

svc_result = mix_func_repetition(features)
expanded_svc_results = expand_results(svc_result, ['kernel', 'gamma', 'C'],)
expanded_svc_results.sort_values(by=['rank_test_score'], ascending=True).head()

# xgboost
features = {
    'vec' : {
        'bow': [
```

```
            {
                'ngram_range': [(1, 1), (1, 2), (2, 2)],
                'binary': [True, False]
            },
            CountVectorizer()
        ],
        'tf-idf': [
            {
                'ngram_range': [(1, 1), (1, 2), (2, 2)],
                'binary': [True, False]
            },
            TfidfVectorizer()
       ]
    },
    'fs_technique' : {
        'k-best-chi' : [
            {
                'k': [20, 80, 200, 400, 800, 1400, 1600]
            },
            SelectKBest(chi2)
        ]
    },
    'model' : {
         'xgboost': [
             {
                 'loss':['deviance'],
                 'criterion':['friedman_mse'],
                 'n_estimators': [20, 40, 80, 100, 200, 300, 500],
                 'learning_rate': [0.2, 0.3, 0.1, 0.01, 0.001],
                 'ccp_alpha': [0.0, 0.001, 0.002, 0.003, 0.004, 0.005]
             },
             GradientBoostingClassifier()
         ]
    }
}

xgboost_result = mix_func_repetition(features)
expanded_xgboost_results = expand_results(xgboost_result, ['n_estimators', '
    ccp_alpha', 'criterion', 'loss', 'learning_rate'],)
expanded_xgboost_results.sort_values(by=['rank_test_score'], ascending=True).head
    ()
```

Listado 3: Ajuste de modelos supervisados

```
bayes_optimal_pipeline = Pipeline(steps = [
    ('all',  FeatureUnion(transformer_list = [
        ('cat_feature', Pipeline(steps = [
            ('selector', CategoricalFeatureSelector()),
            ('encoding', OneHotEncoder())
        ])),
       ('txt_feature', Pipeline(steps = [
            ('selector', TxtFeatureSelector()),
            ('vectorizer', CountVectorizer(ngram_range=(1,1), binary=False)),
        ]))
    ])),
    ('fect_selec', SelectKBest(chi2, k=1200)),
```

```
    ('model', MultinomialNB(alpha=0.4))
])

forest_optimal_pipeline = Pipeline(steps = [
    ('all',  FeatureUnion(transformer_list = [
        ('cat_feature', Pipeline(steps = [
            ('selector', CategoricalFeatureSelector()),
            ('encoding', OneHotEncoder())
        ])),
       ('txt_feature', Pipeline(steps = [
            ('selector', TxtFeatureSelector()),
            ('vectorizer', CountVectorizer(ngram_range=(1,1), binary=False)),
        ]))
    ])),
    ('fect_selec', SelectKBest(chi2, k=1200)),
    ('model', RandomForestClassifier(criterion='gini', n_estimators=100,
   ccp_alpha=0.001))
])

svm_optimal_pipeline = Pipeline(steps = [
    ('all',  FeatureUnion(transformer_list = [
        ('cat_feature', Pipeline(steps = [
            ('selector', CategoricalFeatureSelector()),
            ('encoding', OneHotEncoder())
        ])),
       ('txt_feature', Pipeline(steps = [
            ('selector', TxtFeatureSelector()),
            ('vectorizer', TfidfVectorizer(ngram_range=(1,1), binary=False)),
        ]))
    ])),
    ('fect_selec', SelectKBest(chi2, k=1200)),
    ('model', SVC(kernel='linear', gamma="scale", C=2))
])

xgboost_optimal_pipeline = Pipeline(steps = [
    ('all',  FeatureUnion(transformer_list = [
        ('cat_feature', Pipeline(steps = [
            ('selector', CategoricalFeatureSelector()),
            ('encoding', OneHotEncoder())
        ])),
       ('txt_feature', Pipeline(steps = [
            ('selector', TxtFeatureSelector()),
            ('vectorizer', CountVectorizer(ngram_range=(1,1), binary=False)),
        ]))
    ])),
    ('fect_selec', SelectKBest(chi2, k=1200)),
    ('model', GradientBoostingClassifier(loss='deviance', n_estimators=200,
   criterion='friedman_mse', ccp_alpha=0.001))
])

X = data.reset_index().drop('primary_category', axis = 1)
y = data.reset_index()[['primary_category']]
result = pd.DataFrame(columns=['mean_test', 'std_test', 'repetition', 'model'])
for i in range(0, fitRepetitions):
    cv_outer = StratifiedKFold(n_splits=cv, shuffle=True, random_state=i)

```

```
    bayes_res = np.zeros(cv)
    forest_rest = np.zeros(cv)
    svm_rest = np.zeros(cv)
    xgboost_rest = np.zeros(cv)

    count = 0
    for train_ix, test_ix in cv_outer.split(X, y):
        X_train, X_test = X.iloc[train_ix,:], X.iloc[test_ix,:]
        y_train, y_test = y.iloc[train_ix,:], y.iloc[test_ix,:]

        # bayes
        bayes_optimal_pipeline.fit(X_train, y_train)
        bayes_predict = bayes_optimal_pipeline.predict(X_test)
        bayes_res[count] = accuracy_score(y_test, bayes_predict)

        # random forest
        forest_optimal_pipeline.fit(X_train, y_train)
        forest_predict = forest_optimal_pipeline.predict(X_test)
        forest_rest[count] = accuracy_score(y_test, forest_predict)

        # svc
        svm_optimal_pipeline.fit(X_train, y_train)
        svm_predict = svm_optimal_pipeline.predict(X_test)
        svm_rest[count] = accuracy_score(y_test, svm_predict)

        # xgboost
        xgboost_optimal_pipeline.fit(X_train, y_train)
        xgboost_predict = xgboost_optimal_pipeline.predict(X_test)
        xgboost_rest[count] = accuracy_score(y_test, xgboost_predict)

        count+=1

    result = result.append({'mean_test' : mean(bayes_res) , 'std_test' : std(
    bayes_res), 'repetition': i, 'model': "Naive Bayes"} , ignore_index=True)
    result = result.append({'mean_test' : mean(forest_rest) , 'std_test' : std(
    forest_rest), 'repetition': i, 'model': "Random Forest"} , ignore_index=True)
    result = result.append({'mean_test' : mean(svm_rest) , 'std_test' : std(
    svm_rest), 'repetition': i, 'model': "Support Vec Machine"} , ignore_index=
    True)
    result = result.append({'mean_test' : mean(xgboost_rest) , 'std_test' : std(
    xgboost_rest), 'repetition': i, 'model': "XGBoost"} , ignore_index=True)
```

Listado 4: Comparación de modelos supervisados

```
test_predict = pd.DataFrame(data=[
    {
        'creator_department': 'Sales',
        'resource_type': 'contact',
        'comment': 'Conectar la aplicaci n con Google Calendar'
    },
    {
        'creator_department': 'Customer Success',
        'resource_type': 'contact',
        'comment': 'Cambiar el tama o de la letra a la factura'
    }
], index=[1, 2])
```

```

preprocess = ColumnTransformer([
    ('preprocess', Pipeline(steps=[
        ('LowerCaser', LowerCaser()),
        ('PuntuationRemover', PuntuationRemover()),
        ('Striper', Striper()),
        ('TokenCleaner', TokenCleaner()),
        ('WordCorrector', WordCorrector()),
        ('StopWordsRemover', StopWordsRemover()),
        ('SingleLetterRemover', SingleLetterRemover()),
        ('Stemmer', Stemmer()),
    ]), ['tokens']),
], remainder='passthrough')
test_predict['tokens'] = test_predict.apply(lambda row:  word_tokenize(row['
    comment']), axis=1)
proccessed = preprocess.fit_transform(test_predict.reset_index())

svm_optimal_pipeline = Pipeline(steps = [
    ('all',  FeatureUnion(transformer_list = [
        ('cat_feature', Pipeline(steps = [
            ('selector', CategoricalFeatureSelector()),
            ('encoding', OneHotEncoder())
        ])),
       ('txt_feature', Pipeline(steps = [
            ('selector', TxtFeatureSelector()),
            ('vectorizer', TfidfVectorizer(ngram_range=(1,1), binary=False)),
        ]))
    ])),
    ('fect_selec', SelectKBest(chi2, k=1200)),
    ('model', SVC(kernel='linear', gamma="scale", C=2, probability=True))
])

svm_optimal_pipeline.fit(variables[['creator_department','resource_type','comment
    ']], response)

svm_optimal_pipeline.predict_proba(proccessed)
```

Listado 5: Predicción con modelo supervisado

```
features = {
    'vectorizer' :  {
        'bow-binary': CountVectorizer(binary=True, ngram_range=(1, 1)),
        'bow-non_binary': CountVectorizer(binary=False, ngram_range=(1, 1)),
        'tf_idf-binary': TfidfVectorizer(binary=True, ngram_range=(1, 1)),
        'tf_idf-no_binary': TfidfVectorizer(binary=False, ngram_range=(1, 1)),
    },
    'fect_selec': {
        'chi_k-all': 'all',
        'chi_k-1600': 1600,
        'chi_k-1400': 1400,
        'chi_k-1000': 1000,
        'chi_k-800': 800,
        'chi_k-600': 600,
        'chi_k-400': 400,
        'chi_k-200': 200,
        'chi_k-100': 100,
```

```
    },
    'cluster':{
        # K means
        'kmeans-2-cousine': KmeansCustom(2, cosine_distance),
        'kmeans-3-cousine': KmeansCustom(3, cosine_distance),
        'kmeans-4-cousine': KmeansCustom(4, cosine_distance),
        'kmeans-5-cousine': KmeansCustom(5, cosine_distance),
        'kmeans-6-cousine': KmeansCustom(6, cosine_distance),
        'kmeans-7-cousine': KmeansCustom(7, cosine_distance),
        'kmeans-8-cousine': KmeansCustom(8, cosine_distance),
        'kmeans-9-cousine': KmeansCustom(9, cosine_distance),
        'kmeans-10-cousine': KmeansCustom(10, cosine_distance),
        'kmeans-2-euclidean': KmeansCustom(2, euclidean_distance),
        'kmeans-3-euclidean': KmeansCustom(3, euclidean_distance),
        'kmeans-4-euclidean': KmeansCustom(4, euclidean_distance),
        'kmeans-5-euclidean': KmeansCustom(5, euclidean_distance),
        'kmeans-6-euclidean': KmeansCustom(6, euclidean_distance),
        'kmeans-7-euclidean': KmeansCustom(7, euclidean_distance),
        'kmeans-8-euclidean': KmeansCustom(8, euclidean_distance),
        'kmeans-9-euclidean': KmeansCustom(9, euclidean_distance),
        'kmeans-10-euclidean': KmeansCustom(10, euclidean_distance),

        # DBSCAB
        'dbscan-1-5-euclidean': DBScanCustom(1, 5, 'euclidean'),
        'dbscan-09-5-euclidean': DBScanCustom(0.9, 5, 'euclidean'),
        'dbscan-07-5-euclidean': DBScanCustom(0.7, 5, 'euclidean'),
        'dbscan-05-5-euclidean': DBScanCustom(0.5, 5, 'euclidean'),
        'dbscan-03-5-euclidean': DBScanCustom(0.3, 5, 'euclidean'),
        'dbscan-01-5-euclidean': DBScanCustom(0.1, 5, 'euclidean'),
        'dbscan-1-10-euclidean': DBScanCustom(1, 10, 'euclidean'),
        'dbscan-09-10-euclidean': DBScanCustom(0.9, 10, 'euclidean'),
        'dbscan-07-10-euclidean': DBScanCustom(0.7, 10, 'euclidean'),
        'dbscan-05-10-euclidean': DBScanCustom(0.5, 10, 'euclidean'),
        'dbscan-03-10-euclidean': DBScanCustom(0.3, 10, 'euclidean'),
        'dbscan-01-10-euclidean': DBScanCustom(0.1, 10, 'euclidean'),
        'dbscan-1-20-euclidean': DBScanCustom(1, 20, 'euclidean'),
        'dbscan-09-20-euclidean': DBScanCustom(0.9, 20, 'euclidean'),
        'dbscan-07-20-euclidean': DBScanCustom(0.7, 20, 'euclidean'),
        'dbscan-05-20-euclidean': DBScanCustom(0.5, 20, 'euclidean'),
        'dbscan-03-20-euclidean': DBScanCustom(0.3, 20, 'euclidean'),
        'dbscan-01-20-euclidean': DBScanCustom(0.1, 20, 'euclidean'),
        'dbscan-1-5-cosine': DBScanCustom(1, 5, 'cosine'),
        'dbscan-09-5-cosine': DBScanCustom(0.9, 5, 'cosine'),
        'dbscan-07-5-cosine': DBScanCustom(0.7, 5, 'cosine'),
        'dbscan-05-5-cosine': DBScanCustom(0.5, 5, 'cosine'),
        'dbscan-03-5-cosine': DBScanCustom(0.3, 5, 'cosine'),
        'dbscan-01-5-cosine': DBScanCustom(0.1, 5, 'cosine'),
        'dbscan-1-10-cosine': DBScanCustom(1, 10, 'cosine'),
        'dbscan-09-10-cosine': DBScanCustom(0.9, 10, 'cosine'),
        'dbscan-07-10-cosine': DBScanCustom(0.7, 10, 'cosine'),
        'dbscan-05-10-cosine': DBScanCustom(0.5, 10, 'cosine'),
        'dbscan-03-10-cosine': DBScanCustom(0.3, 10, 'cosine'),
        'dbscan-01-10-cosine': DBScanCustom(0.1, 10, 'cosine'),
        'dbscan-1-20-cosine': DBScanCustom(1, 20, 'cosine'),
        'dbscan-09-20-cosine': DBScanCustom(0.9, 20, 'cosine'),
        'dbscan-07-20-cosine': DBScanCustom(0.7, 20, 'cosine'),
```

```
        'dbscan-05-20-cosine': DBScanCustom(0.5, 20, 'cosine'),
        'dbscan-03-20-cosine': DBScanCustom(0.3, 20, 'cosine'),
        'dbscan-01-20-cosine': DBScanCustom(0.1, 20, 'cosine'),
    }
}

result = calculate_pool(mix(features), variables, response)
```

Listado 6: Ajuste de modelos no supervisados

```

results = clustering_results\
.assign(vec_tech = lambda x: x['vec_name'].apply(lambda y: y.split('-')[0]))\
.assign(binary = lambda x: x['vec_name'].apply(lambda y: y.split('-')[1]))\
.assign(k = lambda x: x['fs_name'].apply(lambda y: y.split('-')[1]))\
.assign(clust_tech = lambda x: x['clust_name'].apply(lambda y: y.split('-')[0]))\
[['adjusted_rand','homogeneity','completeness','v_measure','n_clusters','noise','
    vec_tech','binary','k','clust_tech','clust_name']]\
.rename(columns= {
    'adjusted_rand': 'Rand-Ajustado',
    'homogeneity': 'Homogeneidad',
    'completeness': 'Completitud',
    'v_measure': 'Medida-V',
    'n_clusters': 'Grupos',
    'vec_tech': 'Vectorizaci n',
    'noise': 'Ruido',
    'binary': 'Binarizaci n',
    'clust_tech': 'Agrupaci n',
    'clust_name': 'Detalle',
    'k': 'K'
})\
.replace({
    'bow': 'BOW',
    'tf_idf': 'TF-IDF',
    'kmeans': 'K-Means',
    'dbscan':'DBScan',
    'binary': 'Si',
    'non_binary':'No',
    'all': 'Todos'
})
results.head()

clust_tech = pd.melt(results.sort_values(by='Medida-V',ascending=False)\
            .groupby('Agrupaci n')\
            .first().reset_index()\
            [['Agrupaci n', 'Homogeneidad', 'Completitud', 'Medida-V']],
        id_vars=['Agrupaci n'],
        value_vars=['Homogeneidad', 'Completitud', 'Medida-V'])
clust_tech.head()

clust_tech["value"] = clust_tech["value"].apply(lambda w: math.floor(w * 10 ** 3)
    / 10 ** 3)
fig = px.scatter(clust_tech.rename(columns={
    'variable': ' ndice ',
    'value': 'Valor'
}), x=" ndice ", y="Valor", color='Agrupaci n', title="Comparaci n de T cnicas
```

```
    de Agrupaci n", text="Valor")
fig.show("notebook")

vect_tech = pd.melt(results[results['Agrupaci n'] == 'K-Means'].sort_values(by='
    Medida-V',ascending=False)\
            .groupby('Vectorizaci n')\
            .first().reset_index()\
            [['Vectorizaci n', 'Homogeneidad', 'Completitud', 'Medida-V']],
        id_vars=['Vectorizaci n'],
        value_vars=['Homogeneidad', 'Completitud', 'Medida-V'])
vect_tech.head()

fig = px.line(vect_tech.rename(columns={
    'variable': ' ndice ',
    'value': 'Valor'
}), x=" ndice ", y="Valor", color='Vectorizaci n', title="Comparaci n entre
    M todos de Vectorizaci n", text="Valor")
for g in fig.data:
    g.update(mode='lines+markers')
fig.show("notebook")
```

Listado 7: Comparación modelos no supervisados