Proyecto 1: Aprendizaje supervisado

A00517244 Camila Navarro
A01705747 Enrique García
A01197399 Diana Cadena
Tecnológico de Monterrey
Ingeniería en Ciencia de Datos y Matemáticas
Monterrey, Nuevo León, México

RESUMEN

En base al «Drug Review» dataset buscamos aplicar modelos de clasificación con aprendizaje supervisado para encontrar un modelo predictivo que pueda recomendar una medicina en base a la condición del paciente y las reseñas del medicamento.

Palabras clave *– modelos de predicción, área de salud, aprendizaje supervisado, knn, redes neuronales, svm*

ACM Reference Format:
A00517244 Camila Navarro, A01705747 Enrique García, A01197399 Diana Cadena. 2022. Proyecto 1: Aprendizaje supervisado. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1. INTRODUCCIÓN

El *Machine Learning* es el estudio de métodos y herramientas de utilidad para identificar patrones en los datos. [11] El identificar estos patrones sirve ya sea para comprender mejor lo que describen o para hacer predicciones. Esta área de estudios tiene una gran variedad de aplicaciones y entre ellas se encuentra el área de salud, el enfoque general del proyecto. En su mayoría, las aplicaciones del ML entran dentro del aprendizaje supervisado, no supervisado y de refuerzo. En este caso, se enfocará el reporte en el aprendizaje supervisado ya que se busca clasificar los datos de acuerdo a lo buscado. [11]

El uso de *Machine Learning* dentro del área de salud es amplia y puede llegar a considerarse compleja, sobre todo si es que se están analizando una gran cantidad de datos, lo cual sería lo ideal debido a la complejidad del cuerpo humano. Generalmente hablando, la aplicación del ML es de gran utilidad para hacer más eficientes los procesos y consultas dentro de esta área, sobre todo tomando en consideración que la oferta debe de cumplir con la demanda. [3] El diagnóstico pronto y asertivo de ciertas enfermedades o condiciones es de suma importancia para un buen tratamiento y posible recuperación.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

No obstante, el dataset en específico a utilizarse no es específicamente sobre alguna enfermedad en particular sino sobre los medicamentos recetados con base a ciertos síntomas y las experiencias de lxs pacientes. Este dataset tiene como nombre *UCI ML Drug Review dataset*, el cual ha sido analizado múltiples veces con enfoques distintos. Como se menciona anteriormente, el dataset provee información acerca la satisfacción general de lx paciente con el medicamento así como posibles efectos secundarios y su experiencia. Como complementario a esto, se incluye una calificación dentro del rango [1, 10]. Además, claramente, se incluye el nombre del medicamento recetado así como la condición que se presentaba para su misma receta. [4]

Los datos que proporciona dicho dataset fueron recopilados de distintos sitios farmacéuticos web, con la intención de estudiar y analizar distintos aspectos. Hablando específicamente sobre las reseñas contenidas en el archivo, estas hacen referencia tanto a la eficacia del medicamento así como sus efectos secundarios. Al contar con información en lenguaje natural, el análisis de los datos resulta ser más interesante pero complejo a la vez. Sin embargo, el análisis de los mismos resulta en información valiosa además de ayudar en la toma de decisiones y la mejora del seguimiento de la salud pública con base a la experiencia colectiva. [4]

Algunos de los enfoques de análisis que se le han dado a estos datos son los siguientes:

1. **Clasificación.** Predecir la condición de lx paciente con base en la reseña.
2. **Regresión.** Predecir la calificación otorgada al medicamento con base a la reseña.
3. **Análisis de sentimiento.** Identificar los elementos de la reseña que hacen que sea de utilidad para otras personas, así como lxs pacientes que tienden a dejar reseñas negativas. Además, determinar si la reseña es positiva, neutral o negativa.
4. **Visualización de los datos.** Identificar tanto los medicamentos como padecimientos que se encuentran en el dataset.

Un aspecto importante a tener en consideración para que los modelos de predicción funcionen correctamente es que las bases de datos deben de ser lo más robustas posibles. [3] Entre más datos haya para entrenar y evaluar el modelo, mejor serán los resultados arrojados por el mismo. Asimismo, el modelo no es capaz de predecir o identificar correctamente la relación entre distintas condiciones o variables si no se encuentra presente en los datos. Cabe mencionar que el uso de ML dentro del área de salud no sirve como un reemplazo de los diagnósticos y procedimientos llevados a cabo

en la actualidad sino como una herramienta que aumenta y mejora su desempeño. [3]

2. CONCEPTOS PREVIOS

Se habló del *Machine Learning* de manera general previamente, pero dentro de este existen diversos algoritmos que cumplen con la función del ML: relacionar o predecir. A continuación, se explicarán los distintos algoritmos a utilizarse para realizar las predicciones correspondientes del proyecto.

2.1. KNearestNeighbors

El algoritmo de K vecinos más cercanos, o K-NN por sus siglas en inglés, es un algoritmo de clasificación supervisada. Este busca clasificar un dato utilizando un conjunto de datos de los cuales ya se conoce su clase, esto con base a distancias y otros parámetros. Básicamente, se calcula la distancia entre el dato a clasificar y los datos del conjunto, tomando en consideración únicamente los k datos más cercanos. [10] Se clasifica el dato en cuestión de acuerdo a la clase con mayor frecuencia dentro de los k datos más cercanos.

El funcionamiento en sí de este algoritmo se puede enumerar de la siguiente manera. [10]

1. El algoritmo recibe un conjunto de entrenamiento, la instancia a clasificar y un vector de las distintas clases presentes en el conjunto.
2. Se calcula la distancia, normalmente Euclidiana o de Manhattan, entre cada instancia del conjunto de entrenamiento y la instancia a clasificar.
3. Se seleccionan las k instancias con menor distancia.
4. Se obtiene la predicción de la clase.

Como todo algoritmo de *Machine Learning*, para implementar un algoritmo de aprendizaje supervisado correctamente, se deben de dividir los datos en dos subconjuntos: uno de entrenamiento y otro de aprendizaje. El algoritmo de K vecinos más cercanos no es la excepción en este aspecto pero sí difiere en el proceso de aprendizaje. A diferencia de otros algoritmos, K-NN clasifica la nueva instancia sin la necesidad de aprender algún parámetro y únicamente utilizando el subconjunto de datos de entrenamiento. [10] Otros algoritmos requieren aprender algunos parámetros, por medio del proceso de entrenamiento, para clasificar la nueva instancia.

Se habla acerca de un cálculo de distancia, el cual se puede comprender cómo un sólo cálculo pero en realidad hay tres medidas de distancia distintas: Euclidiana, Manhattan y Minkowski. La distancia Euclidiana mide la distancia más corta entre dos puntos y la distancia de Manhattan es la suma de las diferencias absolutas entre los puntos en todas sus dimensiones. [9] En sí, tanto la distancia Euclidiana como de Manhattan se derivan de la distancia Minkowski. Esta última está definida por la siguiente ecuación. [9]

$$D = \left(\Sigma_{i=1}^{n} |p_i - q_i|^p\right)^{\frac{1}{p}} \tag{1}$$

donde,

$$p = \begin{cases} 1, \ distancia\ Manhattan \\ 2, \ distancia\ Euclidiana \\ otro, \ distancia\ Minkowski \end{cases} \tag{2}$$

Cabe mencionar que en la clasificación de las instancias, se puede realizar de manera ponderada o no. Es decir, dependiendo de la distancia entre los k datos más cercanos, se le otorga un peso que ayuda a determinar la clasificación final de la instancia nueva. [10] Asimismo, es importante encontrar la k óptima, pues si se toma en consideración un valor pequeño se tiene el riesgo de estar tomando valores atípicos.

Por otro lado, si se toma en consideración un valor alto se tiene el riesgo de abarcar varias clases. De igual manera, si se tiene un empate en la frecuencia de las clases presentes, se le otorga una clasificación aleatoria a la instancia. [10] Por lo general, se le otorga un número impar a k para evitar esto mismo. Un último aspecto a tener en consideración es que, para evitar malas predicciones debido a la escala en la que se encuentran los predictores, se deben estandarizar estas mismas.

Para la programación del algoritmo de clasificación en sí, se utiliza la librería de *scikitlearn*. Este se establece de la siguiente manera y contiene los hiperparámetros mostrados.

> **KNeighborsClassifier**(*n_neighbors, weights, algorithm, leaf_size, p, metric, metric_params, n_jobs*)
>
> **n_neighbors** → k vecinos a tomar en cuenta; default = 5
> **weights** → pesos otorgados a las distancias; default = uniform
> **algorithm** → cálculo de los vecinos más cercanos; default = auto
> **leaf_size** → depende de la definición del hiperparámetro previo, afecta en la velocidad y la memoria para el almacenamiento; default = 30
> **p** → parámetro para la métrica de Minkowski; default = 2
> **metric** → medida de distancia utilizada; default = minkowski
> **metric_params** → argumentos adicionales; default = None
> **n_jobs** → cantidad de trabajos a realizar en paralelo con la búsqueda de vecinos; default = None

Los hiperparámetros pueden especificarse con la finalidad de mejorar las métricas de desempeño del modelo. En caso de no ser especificados, se toman los valores *default*.

2.2. SVM

El algoritmo de máquinas de soporte vectorial está basado en la minimización de riesgo estructural y ha sido de gran utilidad para resolver problemas de clasificación. [1] Este algoritmo se apoya en vectores de soporte, los cuales son capaces de formar una frontera de decisión con poco o ningún conocimiento de los datos fuera

de dicha frontera. Los datos son mapeados y se busca la máxima separación entre clases, esto se logra por medio de una función de frontera. [1]

Se pueden establecer ciertos pasos sobre el funcionamiento de este algoritmo, como se muestra a continuación. [1]

1. Se mapean los puntos, valores, de entrada en una dimensión superior a la que corresponden.
2. Se identifica si es un problema linealmente separable o no; en caso de que no lo sea, se debe transformar el espacio para que se logre dicha separación lineal.
3. Se encuentra un hiperplano que separe los datos por segmentos.
4. Se maximiza el margen m entre las clases.
5. La solución puede ser escrita como la combinación de algunos valores de entrada, conocidos como vectores de soporte.

Las máquinas de soporte vectorial maximizan el margen m ya sea por medio de multiplicadores de Lagrange o por medio del producto punto de funciones en el espacio de características, las cuales se les conoce como *kernels*. Este algoritmo suele ser de clasificación para dos clases, pero por medio de la implementación de un algoritmo de programación cuadrática, se puede utilizar para multi-clase. [1]

Algunas fortalezas de este algoritmo es que tiene un proceso de entrenamiento sencillo, además de que no cuenta con óptimos locales. Los datos se escalan correctamente a distintos espacios dimensionales, sobre todo si estos son altos. Asimismo, se pueden ingresar diversos formatos como valores de entrada (cadenas de caracteres, árboles) en vez de vectores; la SVM suele ser aplicada para el procesamiento de imágenes. [1] No obstante, cabe mencionar que para que se tenga un buen desempeño del algoritmo, se requiere contar con metodologías lo suficientemente eficientes como para encontrar los parámetros de inicialización óptimos.

En cuanto a las ecuaciones matemáticas que describen, en parte, el funcionamiento del algoritmo, estas se pueden catalogar en los casos mencionados: lineal y no lineal. Para un caso linealmente separable, se establece que $x_i \in \Re^N$ son los puntos de entrenamiento pertenecientes a una clase $y_i \in \{-1, 1\}$. Además, se establece que el vector $z = \phi(x)$ se transforma de un espacio de características ϕ en $\Re$ a un espacio en Z. Teniendo estas definiciones en consideración, se tiene lo siguiente.[1]

$$w \cdot z + b = 0 \tag{3}$$

La cual representa el hiperplano a encontrarse, definido por un par *(w, b)* buscando separar x_i de acuerdo a la siguiente ecuación.[1]

$$f(x_i) = sign(w \cdot z_i + b) = \begin{cases} 1, & y_i = 1 \\ -1, & y_i = -1 \end{cases} \tag{4}$$

Para establecer la linealidad de la separación, se deben cumplir las siguientes inecuaciones para todos los elementos del conjunto.[1]

$$\begin{cases} (w \cdot z_i + b) \geq 0, & y_i = 1 \\ (w \cdot z_i + b) < 0, & y_i = -1 \end{cases} \tag{5}$$

El procedimiento como tal que se sigue para encontrar la función de decisión se puede definir como se muestra a continuación.

1. Definir vectores de soporte y su clase y_i.
2. Aumentar los vectores en una dimensión.
3. Realizar producto punto entre todos los vectores de soporte.
4. Definir un sistema de ecuaciones con los resultados del paso anterior e igualados a la clasificación y_i.
5. Solucionar el sistema de ecuaciones y encontrar los coeficientes.
6. Sustituir los valores en una función $\tilde{w}$, multiplicar los coeficientes y vectores de soporte.
7. Establecer el hiperplano, con el vector $\vec{w}$ y la constante b.

Cuando se trata de un espacio no linealmente separable, se introducen variables no negativas $\varsigma_i \geq 0$.

$$y_i (w \cdot z_i + b) \geq 1 - \varsigma_i \tag{6}$$

$$\begin{gathered} \min\{\tfrac{1}{2} w \cdot w + C \Sigma_i \varsigma_i\} \\ s.a. \\ y_i (w \cdot z_i + b) \geq 1 - \varsigma_i \\ \varsigma_i \geq 0 \end{gathered} \tag{7}$$

donde C es una constante y $\Sigma_i \varsigma_i$ es la medida del error en la clasificación.

Por medio de un Langrangiano y el teorema de Khun-Tucker, se llega a la siguiente ecuación de decisión. [1]

$$f(x_i) = sign(w \cdot z_i + b) = sign(\Sigma_i \alpha_i y_i z_i \cdot z_i + b) \tag{8}$$

Debido a que no se conoce la función ϕ, se tiene una propiedad de la función *kernel*, la cual consta de calcular el producto punto de los valores de entrada. [1]

$$z_i \cdot z_j = \phi_i(x_i) \cdot \phi_j(x_j) = K(x_i, x_j) \tag{9}$$

De esta manera, se puede encontrar el hiperplano como la solución a

$$\begin{gathered} \max W(\alpha) = \Sigma_i \alpha_i - \tfrac{1}{2} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ s.a. \\ \Sigma_i y_i \alpha_i = 0 \\ 0 \leq \alpha_i \leq C \end{gathered} \tag{10}$$

En cuanto a la función de decisión, esta quedaría como se muestra a continuación.[1]

$$f(x_i) = sign(w \cdot z_i + b) = sign\left(\Sigma_i \alpha_i y_i K(x_i, x_j) + b\right) \tag{11}$$

Para la programación del algoritmo en sí, utilizando la librería de

scikitlearn, este se establece de la siguiente manera y contiene los hiperparámetros mostrados.

> **LinearSVC**(*penalty, loss, dual, tol, C, multi_class, fit_intercept, intercept_scaling, class_weight, verbose, random_state, max_iter*)
>
> **penalty** → especifica la norma de penalización a utilizarse; default = l2
> **loss** → especifica la función de pérdida; default = squared_hinge
> **dual** → selección de algoritmo para solucionar un problema de optimización primal o dual; default = True
> **tol** → tolerancia para el criterio de detención; default = 1e-4
> **C** → parámetro de regularización; default = 1.0
> **multi_class** → determina la estrategia multi-clase a utilizarse; default = ovr
> **fit_intercept** → determina si se calcula el intercepto para el modelo; default = True
> **intercept_scaling** → regularización del intercepto, depende de la configuración de *fit_intercept*; default = 1
> **class_weight** → determina el peso para cada clase presente; default = None
> **verbose** → habilita la salida detallada; default = 0
> **random_state** → controla la cantidad de números pseudoaleatorios generados para mezclar los datos, depende de la configuración de *dual*; default = None
> **max_iter** → número máximo de iteraciones; default = 1000

2.3. Redes neuronales

El algoritmo de redes neuronales, como su nombre lo dice, se basa en la estructura y funcionamiento del sistema nervioso. Este algoritmo consta de una cantidad de unidades de procesamiento, las cuales se encuentran organizadas en capas y conectadas por medio de ponderaciones. Esta organización se suele dar de la siguiente manera: capa de entrada, capa(s) oculta(s) y capa de salida. [2]

El algoritmo se caracteriza, básicamente, por: consistir en unidades de procesamiento que intercambian información, ser capaces de aprender y mejorar su funcionamiento y son de utilidad para reconocer patrones. Esto último puede ser tanto en imágenes como textos y secuencias temporales. [7]

El funcionamiento per se del algoritmo de redes neuronales se puede enumerar de la siguiente manera. [7]

1. El algoritmo recibe un vector con los valores de entrada y los procesa por medio de una función de entrada, tomando en consideración los pesos otorgados.
2. Se calcula el estado de actividad de la neurona (activa o inactiva) por medio de una función de activación, la cual transforma el vector de entrada a un estado de activación.
3. Por último, con base a una función de salida, se determina el valor asignado a las neuronas vinculadas. Si dicho valor se encuentra por debajo del umbral determinado, ninguna salida es otorgada a las siguientes neuronas.
4. Se realiza el mismo procedimiento de manera iterativa, ajustando los pesos con la finalidad de obtener mejores predicciones.
5. El proceso de aprendizaje termina una vez se cumpla un criterio de detención establecido. Este criterio suele ser referente al error cuadrado sobre todos los valores y/o referente a que el error observado se encuentra por debajo de un umbral definido.

Las redes neuronales aprender por medio del entrenamiento, se analizan los registros individuales y se van sacando predicciones de manera iterativa hasta tener ciertos criterios de parada por medio del ajuste de las ponderaciones. En un inicio, las ponderaciones otorgadas se dan de manera aleatoria y se van ajustando de manera gradual, conforme se van prediciendo los resultados de datos conocidos. [2] Debido a este proceso de entrenamiento, es que este algoritmo es de gran utilidad al buscar realizar predicciones.

A esto último se le conoce como un aprendizaje adaptativo, pues el algoritmo aprende a realizar tareas basadas en un entrenamiento o experiencia inicial. Entre otras de las ventajas de las redes neuronales, se encuentra su auto-organización; esta hace referencia a que el algoritmo es capaz de establecer la forma en la que se representa la información obtenida en los procesos previos. [7]

Anteriormente, se mencionaron tres funciones clave para el funcionamiento de una red neuronal: función de entrada, de activación y de salida. La función de entrada, también conocida como *input function*, describe cómo se pueden combinar los distintos valores de entrada y estos a su vez son multiplicados por los pesos. La función, general, de entrada, se puede establecer de la siguiente forma. [7]

$$input_i = (in_{i1} \cdot w_{i1}) * (in_{i2} \cdot w_{i2}) * ... * (in_{in} \cdot w_{in}) \tag{12}$$

donde,

* representa al operador (sumatoria, productoria, máximo, etc.),
$\boldsymbol{n}$ representa la cantidad de entradas a la neurona,
$\boldsymbol{w_i}$ representa el peso, e
$\boldsymbol{in_{in}}$ representa el valor de entrada.

A continuación, se muestran las funciones de entrada más comunes en el siguiente orden: máximo de las entradas pesadas, sumatoria de las entradas pesadas y productoria de las entradas pesadas. [7] El máximo toma en consideración únicamente el valor de entrada más fuerte, multiplicado por su peso correspondiente. [7] Por otra parte, tanto la sumatoria como productoria toman en consideración todos los valores y hacen la operación correspondiente (suma o multiplicación) entre el valor de entrada y su peso.

$$\max_j \left(n_{ij} w_{ij}\right), j = 1, 2, ..., n \tag{13}$$

$$\Sigma_j \left(n_{ij} w_{ij}\right), j = 1, 2, ..., n \tag{14}$$

$$\Pi_j \left(n_{ij} w_{ij}\right), j = 1, 2, ..., n \tag{15}$$

Por otra parte, se encuentra la función de activación, la cual es la entrada global de los valores menos el umbral. Debido a su naturaleza binaria, activa o inactiva, los valores arrojados por esta función suelen entrar dentro de un rango de valores específicos; ya sea (0, 1) o (-1, 1). Las funciones de activación utilizadas con más frecuencia son la lineal, sigmoidea y tangente hiperbólica; detalladas a continuación en este mismo orden. [7]

$$p = \begin{cases} -1, \ x \leq \frac{-1}{a} \\ a * x, \ \frac{-1}{a} < x < \frac{1}{a} \\ 1, \ x \geq \frac{1}{a} \end{cases} \tag{16}$$

donde, a > 0 y x = $gin_i - \Theta_i$.

$$f(x) = \frac{1}{1 + e^{-gx}} \tag{17}$$

$$f(x) = \frac{e^{gx} - e^{-gx}}{e^{gx} + e^{-gx}} \tag{18}$$

Por último, se tiene la función de salida, la cual establece qué valor se le otorga a las neuronas consiguientes. Estos pueden encontrarse dentro de un rango, [-1, 1] o [0, 1], o ser meramente binarios, 0, 1 o -1, 1. Cabe mencionar que si el valor de salida es igual al valor de entrada, esto se le conoce como función identidad y no hay función de salida como tal para este caso. Otra función de salida común sería la binaria, la cual se muestra a continuación. [7]

$$binaria : \begin{cases} 1, act_i \geq \varsigma_i \\ 0, \ de\ lo\ contrario \end{cases} \tag{19}$$

donde act_i es el valor de activación y ς_i es el umbral.

Para la programación del algoritmo de clasificación en sí, utilizando la librería de *scikitlearn*, este se establece de la siguiente manera y contiene los hiperparámetros mostrados.

MLPClassifier(*hidden_layer_sizes, activation, solver, alpha, batch_size, learning_rate, learning_rate_init, power_t, max_iter, shuffle, random_state, tol, verbose, warm_start, momentum, nesterovs_momentum, early_stopping, validation_fraction, beta_1, beta_2, epsilon, n_iter_no_change, max_fun*)

hidden_layer_sizes → el *i*-ésimo elemento representa la cantidad de neuronas en las *i*-ésima capa oculta; default = (100,)
activation → función de activación para la(s) capa(s) oculta(s); default = relu
solver → optimización de pesos; default = adam
alpha → término de regularización L2; default = 0.0001
batch_size → tamaño de subconjuntos para optimización estocástica, depende de la configuración de *solver*; default = auto
learning_rate → tasa de aprendizaje para la actualización de los pesos, depende de la configuración de *solver*; default = constant
learning_rate_init → tasa de aprendizaje inicial, depende de la configuración de *solver*; default = 0.001
power_t → exponente de la tasa de aprendizaje de escala inversa, depende de la configuración de *solver* y *learning_rate*; default = 0.5
max_iter → cantidad de iteraciones máximas; default = 200
shuffle → determina si se re-acomodan aleatoriamente los datos en cada iteración, depende de la configuración de *solver*; default = True
random_state → determina un número aleatorio inicial para los pesos y sesgos; default = None
tol → tolerancia para la optimización; default = 1e-4
verbose → imprime mensajes de progreso; default = False
warm_start → elimina los datos guardados en corridas previas; default = False
momentum → momentum para el gradiente descendiente actualizado, depende de la configuración de *solver*; default = 0.9
nesterovs_momentum → determina si se usa el momentum de Nesterov, depende de la configuración de *solver* y *momentum*; default = True
early_stopping → determina si se detiene la etapa de entrenamiento cuando no mejora la puntuación de validación; default = False
validation_fraction → proporción de datos de entrenamiento a utilizarse en caso de que se detenga dicha etapa antes, depende de la configuración de *early_stopping*; default = 0.1
beta_1 → tasa de decrecimiento exponencial para los primeros estimados, depende de la configuración de *solver*; default = 0.9
beta_2 → tasa de decrecimiento exponencial para los segundos estimados, depende de la configuración de *solver*; default = 0.999
epsilon → valor la estabilidad numérica, depende de la configuración de *solver*; default = 1e-8
n_iter_no_change → número máximo para no cumplir con *tol*; default = 10
max_fun → número máximo de llamadas a una función de pérdida, depende de la configuración de *solver*; default = 15000

3. METODOLOGÍA

Para la resolución del problema en cuestión, se llevo a cabo la metodología CRISP-DM (Cross Industry Standard Process for Data Mining). Esta consta de 6 fases: comprensión del negocio, estudio y comprensión de los datos, análisis de los datos y selección de características, modelado, evaluación y despliegue. A continuación,

se detallará el uso de cada fase dentro del reporte. Cabe destacar que la solución para este reto tuvo sustento en el trabajo realizado por Huster, K. [6]

3.1. Comprensión del negocio.

En la industria farmacéutica un área de oportunidad muy grande es la de satisfacción al cliente. Si bien se sabe que este ha sido un problema por un largo tiempo, la introducción de medios digitales, foros de internet y redes sociales, ha creado una nueva dificultad en la regulación de las reseñas de medicamentos. Los últimos años han sido testigos del rápido crecimiento de un servicio de atención médica digital: la consulta médica en línea (OMC por sus siglas en inglés). A pesar de su popularidad, muchas plataformas de OMC han encontrado problemas en la adopción inicial y el uso continuado entre los pacientes, principalmente por una falta de confianza hacia estas plataformas [8]. Este es un fenómeno complejo que involucra consideraciones tanto interpersonales como tecnológicas, sin embargo, filtros tecnológicos que puedan facilitar la interacción del cliente con la herramienta son muy útiles para esta industria. El análisis de opiniones sobre los diversos aspectos de las reseñas de medicamentos puede proporcionar información valiosa, ayudar en la toma de decisiones y mejorar el seguimiento de la salud pública al revelar la experiencia colectiva de los usuarios. Por ejemplo, una recomendación automatizada de medicamentos a los clientes basada en su condición actual o filtros para controlar la desinformación en la sección de comentarios. Esta información compartida en OMCs tiene también gran utilidad para estudios de mercado.

Nuestra propuesta es tomar provecho de estos datos para predecir cuáles son los medicamentos más recomendables para el usuario, basado en su condición y la calificación del medicamento. En el caso de aprendizaje supervisado, esto entraría dentro de un modelo de clasificación, ya que trabajaríamos con variables categóricas como son la condición del paciente y su satisfacción con el respectivo medicamento.

3.2. Estudio y comprensión de los datos.

El dataset escogido fue creado como parte de una investigación de análisis de sentimiento. Se utilizaron datos de dos páginas web independientes para recuperar las reseñas de los usuarios y las calificaciones sobre la experiencia con los medicamentos. *Drugs.com* es, según el proveedor, el sitio web de información farmacéutica más grande y más visitado que proporciona información tanto para consumidores como para profesionales de la salud. Proporciona reseñas de usuarios sobre medicamentos específicos junto con la condición relacionada y una calificación del usuario de 10 estrellas que refleja la satisfacción general del usuario [5].

Del mismo modo, *Druglib.com* es un recurso de información sobre medicamentos para consumidores y profesionales de la salud. Contiene un número considerablemente menor de reseñas, pero se proporcionan de una manera más estructurada, pues tienen una sección de Efectividad y Efectos secundarios. Se recopilaron los comentarios y las calificaciones de los usuarios de ambas páginas mediante un *web scraper* automático. El rastreo de estos dominios dio como resultado dos conjuntos de datos que comprenden 215063 reseñas de Drugs.com y 3551 reseñas de Druglib.com. Ambos conjuntos de datos se dividieron en particiones de entrenamiento y prueba de acuerdo con un esquema de muestreo aleatorio estratificado con la proporción de 75 % y 25 %.

3.3. Análisis de los datos y selección de características.

Para realizar la parte exploratoria, se unieron ambos datasets. Además, se realizaron distintas gráficas con la finalidad de comprender el comportamiento de los datos así como la correlación entre los mismos. A continuación, se muestran distintas descripciones de los datos presentes así como las gráficas y apoyos visuales correspondientes.

Descripción general del dataset.

- Número de instancias: 215,063 registros
- Valores faltantes: 1,194, valores
- No. de atributos cuantitativos: 2, siendo estos *rating* y *usefulCount*.
- No. de atributos categóricos: 5, siendo estos *uniqueID*, *drugName*, *condition*, *review* y *date*.

Descripción de atributos

- uniqueID: Identificador de paciente; único para cada entrada por lo que no es realmente de utilidad y se puede reemplazar con el índice.
- drugName: Nombre del medicamento recetado; cuenta con 3,671 registros únicos. Es nuestra variable a predecir.
- condition: Nombre del padecimiento reportado; cuenta con 916 registros únicos. Es de gran utilidad para el proceso de aprendizaje y correlación.
- review: Reseña del medicamento; comentarios acerca de cómo fue su experiencia con el medicamento, por qué fue recetado y sobre el cambio que notaron al haberlo tomado. Es de utilidad para el proceso de aprendizaje no supervisado.
- rating: Calificación otorgada 1-10 al medicamento; esta es dada por los pacientes complementando así el review de lo que observaron al haberlo tomado. Es de gran utilidad para el proceso de aprendizaje.
- date: Fecha en la que se escribió la reseña. No agrega mucha utilidad al modelo, a menos que se decida descartar las entradas de mayor antigüedad.
- usefulCount: Cantidad de usuarios que encontraron útil la reseña, siendo un número basado en sí la reseña les fue útil o no y la frecuencia de los resultados afirmativos. Nos es de utilidad para saber qué reseñas tienen un mayor peso.

En cuanto a la limpieza de los datos, se eliminaron todas las instacias de datos nulos. Estos se encontraban únicamente en la columna de *condition*. Esta columna presentaba también elementos que no aportaban información al dataset por aparecer como «NOT LISTED» o presentar entradas erróneas (conteniendo en específico el mensaje «Xusers found this comment helpful.»). Estas entradas fueron eliminadas.

Además de eliminar aquellas filas con errores decidimos no tomar en cuenta aquellas con medicamentos que aparezcan poco en

el dataset. Nuestro objetivo es poder predecir el medicamento más recomendable en base a la condición y su calificació, pero medicamentos que solo tengan un par de instancias no tendrán mucha utilidad y harán ruido al realizar nuestras predicciones.

3.4. Modelado

Previo a la minería de los datos, se realizó una transformación de los categóricos que se estarían ocupando. Siendo estos, el nombre del medicamento y la condición médica que estaba presentando el paciente.

Además, se separó el dataset en dos secciones para poder contar con los de entrenamiento y los de prueba. Para ello nos quedamos con la variable target de *drugName* y las variables de independientes de *rating* y la de *condition*. Para el algoritmo de KNN, se requiere la normalización de los datos para que los resultados sean lo mejor posible. Por esto mismo, en este algoritmo se trabajó con la separación del dataset normalizada. Para los demás algoritmos, se dejó tal cual está.

El siguiente paso fue la creación de los modelos de predicción de clasificación. Cabe destacar que para ellos, se ocuparon las herramientas derivadas de la librería de sklearn, para los cuales se trabajo el de k-Nearest Neighbors, Redes Neuronales y Support Vector Machines. En cada uno de estos algoritmos, se planea encontrar los hiperparámetros óptimos por medio de la herramienta de GridSearchCV.

Una vez obtenidos estos hiperparámetros, se realizan las predicciones correspondientes de cada modelo así como sus evaluaciones para definir la utilidad de dichas predicciones. Cabe mencionar que esta búsqueda de hiperparámetros óptimos puede requerir de cierto tiempo, por lo que es importante tenerlo en consideración al momento de implementar la herramienta.

3.5. Evaluación

Al realizar un modelo de predicción multi-clase, se requieren de métricas de evaluación específicas. Al realizar un modelo con tantas clases a predecir, debemos ser cuidadosos al interpretar los resultados que nos puedan dar las mediciones. En específico, medimos los siguientes aspectos de la predicción:

1. **Precisión** Es la capacidad del clasificador de no etiquetar una muestra negativa como positiva. Esta dado por la relación

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

donde TP representa verdaderos positivos y FP falsos positivos.
2. **Exhaustividad (Recall)** Es la proporción de positivos reales que se clasifican correctamente. Esta dado por la relación

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

donde TP representa verdaderos positivos y FN falsos negativos.
3. **Medida F1** Es la media armónica entre precisión y exhaustividad. Esta dado por la relación

$$\frac{2(precision)(recall)}{precision + recall}$$

.
4. **Support** Es el número de ocurrencias de cada clase en y.

Estas medidas binarias pueden ser aplicadas a problemas multiclase mediante distintas técnicas. Primero, un problema multiclase se divide en una serie de problemas binarios utilizando enfoques Uno contra uno (OVO) o Uno contra el resto (OVR, también llamado uno contra todos). OVO presenta inconvenientes computacionales, por lo que el estándar es el enfoque OVR.

Esencialmente, la estrategia One-vs-Rest convierte un problema multiclase en una serie de tareas binarias para cada clase en el objetivo. Para cada tarea, se crea un clasificador binario y su rendimiento se mide mediante una métrica de clasificación binaria como la precisión (o cualquiera de las métricas mencionadas anteriormente).

Tarea n: Clase n vs [Todas las demás clases]

El resultado será n puntajes de la precisión o métrica medida. Para poder comparar un clasificador con otro, se necesita una única puntuación de precisión, no n, por lo que se representa la precisión en todas las clases a través de varias técnicas de promedio.

1. **Micro** Es la fracción de predicciones que el modelo acertó. El micropromedio se encuentra dividiendo la suma de las celdas diagonales de la matriz entre la suma de todas las celdas. Al ser una métrica tan engañosa, esta técnica de promediación rara vez se toma en cuenta.
2. **Macro** Calcula la media aritmética simple de todas las métricas en todas las clases. Esta técnica otorga el mismo peso a todas las clases.
3. **Weighted** Toma en cuenta el desequilibrio de clases calculando el promedio de las métricas binarias ponderadas por el número de muestras de cada clase en el objetivo. El promedio ponderado se calcula multiplicando cada puntaje por el número de ocurrencias de cada clase y dividiendo entre el número total de muestras.

3.6. Despliegue

Al realizar los modelos y encontrar aquel con las mejores métricas, tenemos ya un predictor entrenado para el uso de deseamos: recomendar la mejor opción de medicina para el paciente.

Si bien el objetivo de este proyecto no es realizar un prototipo ejecutable de esta función, proponemos su uso como una app web dentro de los sitios donde se recogieron los datos para así dar una experiencia más personalizada al usuario. Puede incluso no ser parte de uno de los sitios mencionados, sino ser aplicado a cualquier sitio web de consultas médicas online.

A00517244 Camila Navarro
A01705747 Enrique García
A01197399 Diana Cadena

4. RESULTADOS

4.1. Análisis descriptivo

En total el dataset contiene 7 columnas y 21,097 filas. De estas, la única columna con valores nulos es *condition* con 1209 entradas nulas.

Las variables categóricas son:

1. uniqueID
2. drugName
3. condition
4. review
5. date

Las cuantitativas son:

1. rating
2. usefulCount

Estas fueron descritas anteriormente.

Ya que nuestro modelo utiliza solamente *condition* y *rating* como atributos y *drugName* como objetivo, enfocamos nuestro análisis en estas variables.

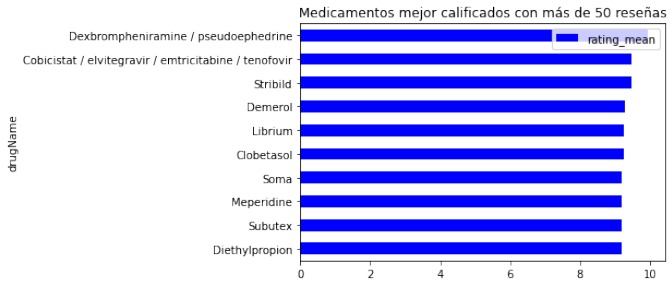

Figura 1: Medicamentos mejor calificados y con más de 50 reseñas.

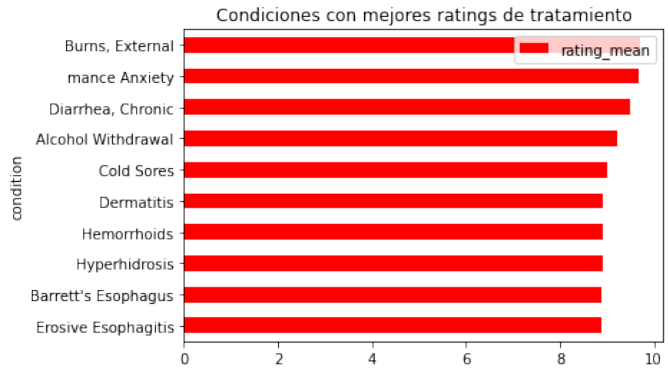

Figura 2: Condiciones con tratamientos mejor calificados.

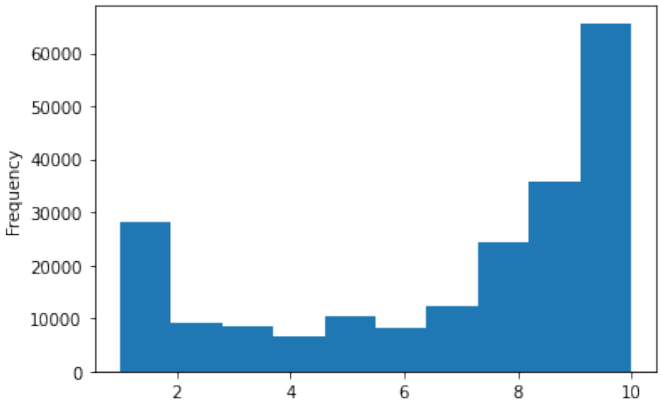

Figura 3: Histograma de la variable *rating*.

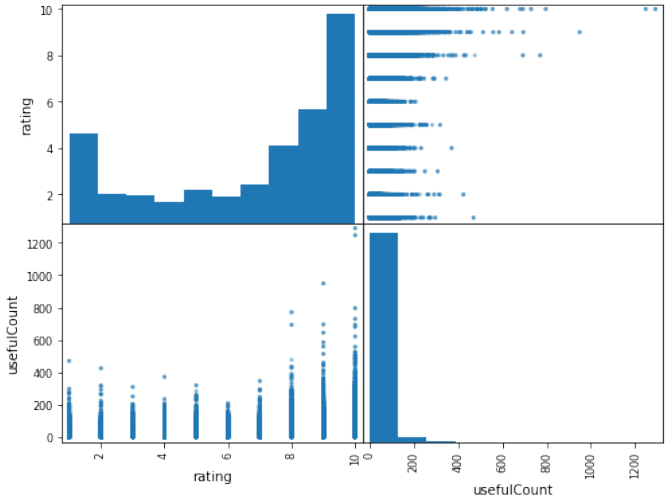

Figura 4: Matriz de dispersión de las varibles *rating* y *usefulCount*.

Además, pudimos observar como del dataset contaba con el 21.9 % del rating con una calificación baja siendo esta menor cuatro. Para la calificación media se tenía un 17.9 % el siendo este un rango entre los que calificaron entre el cuatro al ocho. Por la parte de la calificación alta se tuvo al 60.1 % de los registros siendo estos igual o mayores que ocho.

4.2. Configuración de los algoritmos

Del algoritmo de **K vecinos más cercanos**, se buscaron optimizar los siguientes hiperparámetros: *n_neighbors*, *weights* y *p*. Por medio de la herramienta previamente mencionada, GridSearchCV, y con una validación cruzada de 5 *folds*, se obtuvo que los *k vecinos* a tomarse en consideración deben de ser 9. Además, se toma un peso con base a la distancia y el valor de p es igual a 1, lo que significa que se utiliza la distancia Manhattan.

En cuanto a los siguientes dos algoritmos, se inició la búsqueda de los hiperparámetros óptimos pero no se logró el objetivo inicial. En el algoritmo de **redes neuronales**, los hiperparámetros a optimizar fueron: *activation*, *solver*, *learning_rate* y *max_iter*. Se optó por establecer los siguientes hiperparámetros, correspondientes al orden establecido con anterioridad: relu, adam, adaptive, 100.

Por último, los hiperparámetros del algoritmo de **máquinas de soporte vectoriales** que se pretendían optimizar eran: *penalty*, *loss*, *multi_class* y *max_iter*. En este algoritmo, la penalización seleccionada fue l2, loss = squared_hinge, multi_class = ovr y max_iter = 2000.

4.3. Resultados de evaluación

K-NN La precisión del algoritmo, utilizando validación cruzada da un valor de 14.81 %. En la siguiente figura, se muestran las distintas métricas, explicadas con anterioridad, que son de utilidad para la interpretación del desempeño del modelo.

	precision	recall	f1-score	support
accuracy	0.145819	0.145819	0.145819	0.145819
macro avg	0.651655	0.089733	0.065315	62612.000000
weighted avg	0.387307	0.145819	0.116281	62612.000000

Figura 5: K-NN

Redes neuronales La precisión del algoritmo, utilizando validación cruzada da un valor de 36.99 %. En la siguiente figura, se muestran las distintas métricas, explicadas con anterioridad, que son de utilidad para la interpretación del desempeño del modelo.

	precision	recall	f1-score	support
accuracy	0.369999	0.369999	0.369999	0.369999
macro avg	0.626168	0.345135	0.286994	6873.000000
weighted avg	0.567054	0.369999	0.291861	6873.000000

Figura 6: Redes neuronales

SVM La precisión del algoritmo, utilizando validación cruzada da un valor de 34.51 %. En la siguiente figura, se muestran las distintas métricas, explicadas con anterioridad, que son de utilidad para la interpretación del desempeño del modelo.

	precision	recall	f1-score	support
accuracy	0.345119	0.345119	0.345119	0.345119
macro avg	0.724840	0.317949	0.246353	6873.000000
weighted avg	0.634915	0.345119	0.247569	6873.000000

Figura 7: SVM

5. CONCLUSIONES Y REFLEXIONES INDIVIDUALES

De los tres modelos empleados para el reporte, el que mejor porcentaje de precisión tuvo fue el de redes neuronales. No obstante, todos los porcentajes pueden ser considerados como bajos, ya que son menores al 40 %. Esta sería una gran área de oportunidad para futuras mejoras del proyecto, cómo entablar las relaciones entre los datos para hacer que estas métricas mejoren. De esta manera, también se estaría realizando un mejor modelo predictivo y sería de mayor utilidad.

5.1. Navarro, C.

Este proyecto fue bastante retador por la naturaleza del dataset. Decidimos como equipo trabajar con una base de datos no convencional para probar nuestros conocimientos en un escenario más aplicable, ya que venía con dificultades como una limpieza más exhaustiva y una comprensión de los algoritmos y las métricas de evaluación diferente a los escenarios vistos en clase. Esta diferencia radicó principalmente en la gran cantidad de clases que contenía nuestro dataset. Si bien es más confuso evaluar este tipo de modelos, la realidad es que en el mundo práctico los datos no se encuentran siempre en clases binarias. Esto significó también llevar a cabo más investigación, ya que en un principio no conocíamos cómo interpretar las métricas. Fue así como descubrimos distintas técnicas como el One vs the Rest en la que el modelo binario de evaluación aprendido en clase pudo ser aplicado a un problema de mayor complejidad. En definitiva, la investigación fue la parte más importante del proyecto. Esta ocurrió desde el principio, en la implementación del método CRISP-DM en que tuvimos que entender a fondo cómo fueron recopilados los datos de la web, hasta el final del proyecto, en que tuvimos que encontrar métodos que funcionaran mejor para la condición multiclase de nuestros datos.

5.2. García, E.

Puedo concluir con el proyecto sobre la importancia que se tiene en la evaluación inicial de los datos, siendo así la parte más retadora del proyecto encontrar sobre lo que íbamos a describir a través del mismo y las predicciones como se iban a obtener. Considero que para iniciar fue muy preciso las el análisis descriptivo, lo que permite un mayor comprensión del dataset y así mismo como poder manipularlo. Me parece que por la parte de los modelos de clasificación aun nos queda mucho por aprender y poder seguir desarrollando, ya que por la parte de hipermarámetros la investigación se ve limitada por el tiempo, así como lo que tardaba en correr el código. Finalmente, me pareció un tema muy interesante, el poder estar contribuyendo con conocimiento básico que tenemos a la parte de salubridad y por como es este tema hay mucho más donde se pueda seguir investigando.

5.3. Cadena, D.

En este proyecto, se buscó ser lo más puntuales posibles sobre la forma de presentar los procesos que siguen los algoritmos de modelación de los datos. Así como la forma de obtener las métricas de evaluación, dando a conocer a lx lectorx sobre las distintas funciones que se tienen y mostrando los resultados del proyecto como ejemplo. Es importante destacar que se tiene que considerar

A00517244 Camila Navarro
A01705747 Enrique García
A01197399 Diana Cadena

mejorar las predicciones a través de un mayor procesamiento de la información para destacar los valores que si serán relevantes únicamente para la predicción.

Considero que este fue un gran ejemplo de cómo los modelos no pueden trabajar por sí solos, sino que debe haber toda una planeación detrás del uso que se dará al dataset. En nuestro caso, esto conllevó limpieza, normalización y un análisis exploratorio más profundo para entender los datos. Este análisis se aplicó también al realizar los modelos. Como integrante, una aportación que hice al proyecto fue el análisis de hiperparámetros para saber cuáles darían mejores resultados a cada modelo así como la descripción de los algoritmos dentro del reporte.

REFERENCIAS

[1] Gustavo A Betancourt. 2005. Las máquinas de soporte vectorial (SVMs). *Scientia et technica* 1, 27 (2005).
[2] IBM Corporation. 2021. El modelo de redes neuronales. (2021). https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model
[3] Arwinder Dhillon and Ashima Singh. 2019. Machine learning in healthcare data analysis: a survey. *Journal of Biology and Today's World* 8, 6 (2019), 1–10.
[4] Felix Gräßer, Surya Kallumadi, Hagen Malberg, and Sebastian Zaunseder. 2018. Aspect-Based Sentiment Analysis of Drug Reviews Applying Cross-Domain and Cross-Data Learning. In *Proceedings of the 2018 International Conference on Digital Health* (Lyon, France) *(DH '18)*. Association for Computing Machinery, New York, NY, USA, 121–125. https://doi.org/10.1145/3194658.3194677
[5] Felix Gräßer, Surya Kallumadi, Hagen Malberg, and Sebastian Zaunseder. 2018. Aspect-Based Sentiment Analysis of Drug Reviews Applying Cross-Domain and Cross-Data Learning. In *Proceedings of the 2018 International Conference on Digital Health* (Lyon, France) *(DH '18)*. Association for Computing Machinery, New York, NY, USA, 121–125. https://doi.org/10.1145/3194658.3194677
[6] K. Huster. 2020. UCI ML Drug Review dataset. (2020). https://github.com/katya/Udacity-_capstone/blob/master/UCI_ML_Drug_Review_dataset.ipynb
[7] Damián Jorge Matich. 2001. Redes Neuronales: Conceptos Básicos y Aplicaciones. (2001). https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf
[8] Melody Kiang Fangyun Yuan Ming Yang, Jinglu Jiang. 2021. Re-Examining the Impact of Multidimensional Trust on Patients' Online Medical Consultation Service Continuance Decision. *Information systems frontiers : a journal of research and innovation* (2021), 1–25. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7932182/#Bib1title
[9] Pulkit Sharma. 2020. 4 Types of Distance Metrics in Machine Learning. (2020).
[10] Mikel Xabier Uriz Martín. 2015. Aprendizaje de distancias basadas en disimilitudes para el algoritmo de clasificación KNN. (2015).
[11] Jenna Wiens and Erica S Shenoy. 2018. Machine learning for healthcare: on the verge of a major shift in healthcare epidemiology. *Clinical Infectious Diseases* 66, 1 (2018), 149–153.