
Universidad Autónoma Metropolitana

**Unidad Azcapotzalco**

**División de Ciencias Básicas e Ingeniería**

# Modelos de aprendizaje automático para el apoyo en la clasificación de tipos de cáncer a partir de datos estructurados y no estructurados de expedientes clínicos

**Idónea comunicación de resultados**

**que presenta el:**

**Ing. Erick Esteven Montelongo González**

**para obtener el grado de:**

**Maestro en Ciencias de la Computación**

**Directores**

**Dra. Beatriz Adriana González Beltrán**

**Dr. José Alejandro Reyes Ortíz**

**Ciudad de México** **Julio 2020**

**Para mis padres.** *Por brindarme amor incondicional durante toda mi vida. Sus enseñanzas, consejos y valores me han formado como persona. Gracias por siempre darme todo su apoyo, son un orgullo y modelo a seguir.*

**Para Ale.** *Por siempre apoyarme y tener las palabras necesarias en el momento adecuado, e impulsarme a mejorar. Comparto este logro contigo, pues siempre has estado cuando más te necesito.*

# Reconocimientos

Estoy profundamente agradecido con mis asesores de tesis, la Dra. Beatriz Adriana González Beltrán y el Dr. José Alejandro Reyes Ortíz. Por guiarme en todo momento en mi trabajo de tesis, así como brindarme consejos y orientación invaluable, mas allá de lo académico. En cada duda que tuve durante mi trayectoria, siempre tuvieron las palabras correctas para guiarme. Sin su apoyo el completar este trabajo no hubiese sido posible.

Me gustaría agradecer a mis revisores, la Dra. Mireya Tovar Vidal, el M. en C. Josué Figueroa González y el Dr. Juan Villegas Cortez. A través de sus observaciones y comentarios hicieron me impulsaron para enriquecer este trabajo. Gracias por estar disponibles a mis preguntas y orientarme con cada una de sus respuestas.

Agradezco al Dr. Luis Fernando Hoyos Reyes, coordinador de la Maestría en Ciencias de la Computación. Por apoyarme durante todo mi tiempo de estudio, además de darme la posibilidad de estudiar este posgrado.

Al Consejo Mexiquense de Ciencia y Tecnología (COMECYT) por el apoyo económico brindado.

Finalmente, me gustaría reconocer a la Universidad Autónoma Metropolitana, por la formación educativa brindada.

## Resumen

La existencia de grandes volúmenes de datos generados por el área de la salud presenta una oportunidad importante para su análisis. Este puede obtener información que ayude a los médicos en el proceso de toma de decisiones para el diagnóstico o tratamiento de enfermedades, tales como el cáncer.

El presente trabajo presenta una metodología para la clasificación de pacientes con cáncer de hígado, pulmón y pecho, a través de modelos de aprendizaje automático, para obtener el modelo que mejor se desempeña en la clasificación. La metodología considera tres modelos de clasificación: Máquinas de Soporte Vectorial (SVM), Perceptrón Multi-Capa (MLP) y AdaBoost utilizando tanto la información estructurada como no estructurada de los expedientes clínicos de los pacientes.

Los resultados obtenidos muestran que el mejor modelo de clasificación fue el MLP utilizando solamente datos no estructurados, obteniendo un 89 % de precisión, mostrando la utilidad de este tipo de datos en la clasificación de pacientes con cáncer.



**Palabras clave.** Aprendizaje automático, Procesamiento del Lenguaje Natural, clasificación de cáncer, datos no estructurados.

## Abstract

The existence of large volumes of data generated by the health area presents an important opportunity for analysis. This can obtain information to support physicians in the decision-making process for the diagnosis or treatment of diseases, such as cancer.

The present work shows a methodology for the classification of patients with liver, lung and breast cancer, through machine learning models, to obtain the model that performs best in the classification. The methodology considers three classification models: Support Vector Machines (SVM) , Multi-Layer Perceptron (MLP) and AdaBoost usign both structured and unstructured information from the patient's clinical records.

Results show that the best classification model is MLP using only unstructured data, obtaining 89 % of precision, showing the usefulness of this type of data in the classification of cancer patients.



**Keywords.** Machine learning, Natural Language Processing, cancer classification, unstructured data.

# Índice general

# Índice de figuras

# Índice de tablas

# Simbología y Acrónimos

$2 \times \frac{Precision \times Recall}{Precision + Recall}$ Métrica F1 para el clasificador.

$FN$ Falso negativo. Indica que el modelo de clasificación predice incorrectamente la clase negativa.

$FP$ Falso positivo. Indica que el modelo de clasificación predice incorrectamente la clase positiva.

$TP$ Verdadero positivo. Indica que el modelo de clasificación predice correctamente la clase positiva.

$\frac{TP}{TP+FN}$ Métrica de exhaustividad (recall) para el clasificador.

$\frac{TP}{TP+FP}$ Métrica de precisión para el clasificador.

$\gamma$ Desviación estándar.

$\sigma$ Parámetro que define el alcance de las muestras para definir la región de decisión en las SVM.

**AD** AdaBoost.

**AUC** Área bajo la curva, del inglés *Area Under the Curve*.

**BI-RADS** Sistema de reporte de imágenes de pecho y datos, del inglés *Breast Imaging Reporting and Data System*.

**BOW** Bolsa de palabras, del inglés *Bag of Words*.

**BPM** Máquina de puntos de Bayes, del inglés *Bayes Point Machine*.

**CBOW** Bolsa de palabras contínua, del inglés *Continuous Bag-of-Words*.

**CNN** Red neuronal convolucional, del inglés *Convolutional Neural Network*.

**CRF** Campos aleatorios condicionales, del inglés *Conditional Random Fields*.

**CSSVM** Máquina de soporte vectorial sensible al costo, del inglés *Cost-Sensitive Support Vector Machine*.

**CT** Tomografía computarizada, del inglés *Computed Tomography*.

**CUI** Identificador de conceptos únicos, del inglés *Concept Unique Identifier*.

**DL** Aprendizaje profundo, del inglés *Deep Learning*.

**DT** Árboles de decisión, del inglés *Decision Trees*.

**EHR** Historia clínica electrónica, del inglés *Electronic Health Record*.

**ELLR** Regresión logística de red elástica, del inglés *Elastic Net Logistic Regression*.

**EM** Expectación-Maximización, del inglés *Expectation-Maximization*.

**FFRNN** Red neuronal recurrente prealimentada, del inglés *Feed-Forward Recurrent Neural Network*.

**GBM** Máquinas de aumento de gradiente, del inglés *Gradient Boosting Machines*.

**GBT** Árboles de aumento de gradiente, del inglés *Gradient Boosted Trees*.

**GL** Regresión linear general, del inglés *General Linear Regression*.

**ICD** Clasificación internacional de enfermedades, del inglés *International Classification of Diseases*.

**ICD-O** Clasificación internacional de enfermedades para oncología, del inglés *International Classification of Diseases for Oncology*.

**KB** Base de conocimiento, del inglés *Knowdlege Base*.

**KNN** k-Vecinos cercanos, del inglés *k-Nearest Neighbor*.

**LDA** Análisis de discriminante linear, del inglés *Linear Discriminant Analysis*.

**LR** Regresión logística, del inglés *Logistic Regression*.

**LSSVM** Máquina de soporte vectorial de mínimos cuadrados, del inglés *Least Square Support Vector Machine*.

**LSTM** Gran memoria a corto plazo, del inglés *Long Short-Term Memory*.

**LU** Unidades léxicas, del inglés *Lexical Units*.

**MIMIC** Monitoreo inteligente multiparámetro en cuidados intensivos, del inglés *Multiparameter Intelligent Monitoring in Intensive Care*.

**ML** Aprendizaje automático, del inglés *Machine Learning*.

**MLP** Perceptron multi-capa, del inglés *Multilayer Perceptron*.

**NB** Bayes ingenuo, del inglés *Naive Bayes*.

**NLP** Procesamiento del lenguaje natural, del inglés *Natural Language Processing*.

**NMEDW** Almacen de datos empresariales de la medicina del noroeste, del inglés *Northwestern Medicine Enterprise Data Warehouse*.

**NN** Red neuronal, del inglés *Neural Network*.

**NNLM** Modelado del lenguage mediante redes neuronales, del inglés *Neural Network Language Modeling*.

**OHDSI** Ciencia de datos de salud observacional, del inglés *Observational Health Data Sciences*.

**PLN** Procesamiento del Lenguaje Natural.

**PV** Vector de párrafos, del inglés *Paragraph Vector*.

**QDA** Análisis de discriminante cuadrático, del inglés *Quadratic Discriminant Analysis*.

**RBF** Función de base radial, del inglés *Radial Basis Function*.

**RBFN** Red de función de base radial, del inglés *Radial Basis Function Network*.

**RME** Raíz del error medio, del inglés *Root Mean Error*.

**RMSE** Raíz del error cuadrático medio, del inglés *Root Mean Square Error*.

**SEER** Programa de monitoreo, epidemiología y resultados finales, del inglés *Surveillance, Epidemiology, and End Results Program*.

**SGD** Decenso de gradiente estocástico, del inglés *Stochastic Gradient Decent*.

**SNOMED-CT** Nomenclatura sistematizada de medicina - términos clínicos, del inglés *Systematized Nomenclature of Medicine – Clinical Terms*.

**SVM** Máquina de soporte Vectorial, del inglés *Support Vector Machines*.

**TF-IDF** Frecuencia de términos - Frecuencia inversa del documento, del inglés *Term Frequency – Inverse Document Frequency*.

**TNM** Tumor-Nodo-Metástasis.

**TRF** Bosques aleatorios, del inglés *Trees Random Forest*.

# Capítulo 1

# Introducción

Una de las disciplinas de las ciencias de la computación con mayor auge es el aprendizaje automático, del inglés *Machine Learning* (ML), esta disciplina se encarga de la extracción de información de grandes conjuntos de datos. El objetivo de extraer la información es principalmente, la predicción sobre nuevos conjuntos de datos, identificación de patrones y clasificación de datos en distintas clases [1].

Esta disciplina ha sido aplicada con éxito en distintas áreas. Se ha aplicado en negocios para modelar procesos, analizarlos y mejorarlos [2]. También se ha aplicado en genética para encontrar patrones y descubrir procesos biológicos en el genoma humano [3]. Además, en el área de la salud se ha aplicado en distintas tareas, entre ellas, clasificación de pacientes, interacciones de medicamentos y predicción de enfermedades [4].

Las ciencias de la salud [1] juegan un papel muy importante en la vida de todo ser humano, pues gracias a los avances en esta área se mejora la calidad de vida de las personas. Parte de la investigación realizada en el área clínica toma información recabada de los pacientes en las instituciones de salud [6]. Dicha información es obtenida por los profesionales de la salud en forma de reportes médicos, estudios de laboratorio, estudios de imagenología, etc. En los últimos años se ha dado un aumento en el uso de sistemas de cómputo para permitir que la información recabada en las instituciones de salud esté disponible de manera electrónica; por ejemplo, a través de los expedientes clínicos electrónicos. Estos expedientes tienen mucha información para ser analizada; por ejemplo, en la búsqueda de candidatos para pruebas clínicas [7] o para encontrar patrones comunes en enfermedades, como es el caso del cáncer.

El cáncer se define como la división anormal de células que puede invadir al tejido cercano [8]. Esta enfermedad es la segunda causa de muerte en el mundo, y se estima que para 2020 habrá mas de 18 millones de incidencias [9]. Actualmente existe un gran esfuerzo por hacer pública la información anónima de pacientes con cáncer, para que investigadores de distintas áreas puedan utilizarla. Las investigaciones relacionadas con el cáncer hacen posible, entre otras cosas, el diagnóstico temprano, medicamentos a la medida y la categorización de pacientes para tratamientos específicos. De esta manera, los algoritmos de ML permiten realizar análisis de la información de los pacientes con cáncer. Además, los modelos generados por los algoritmos pueden realizar predicciones sobre nuevos conjuntos de datos, disminuyendo así el tiempo que un ser humano necesita para analizar grandes volúmenes de

[1] Las ciencias de la salud estudian todos los aspectos de la salud, enfermedad y cuidado de la salud. Este campo de estudio tiene como objetivo desarrollar conocimiento, intervenciones y tecnología para el uso en el cuidado de la salud para el tratamiento de pacientes [5].

información.

La presente tesis presenta una metodología de ML para la clasificación de diferentes tipos de cáncer, en específico, cáncer de pecho, de pulmón e hígado; basada en datos estructurados como son estudios de laboratorio, así como en datos no estructurados, a partir de la información contenida en las notas clínicas escritas por los profesionales de la salud. Se considera como un aporte que estos modelos permitirán a dichos profesionales reducir el tiempo necesario para analizar grandes volúmenes de información contenida en los expedientes clínicos de los pacientes, dando información de apoyo en el diagnóstico final del tipo de cáncer.

## 1.1. Planteamiento del problema

El problema principal a atacar de la presente tesis es la clasificación de tres tipos de cáncer (mama, hígado y pulmón), utilizando tres modelos de ML, siendo estos Máquinas de Soporte Vectorial, del inglés *Support Vector Machines (SVM)*, Perceptrón Multi-capa, del inglés *Multi-Layer Perceptron (MLP)* y *AdaBoost*. Además, se considera que parte de la información que se utilizará para llevar a cabo la clasificación, consiste en información no estructurada, correspondiente a notas clínicas escritas en texto libre. Así, las principales problemáticas que se resolverán para llevar a cabo la clasificación, son las siguientes:

- **Manejo de distintos tipos de información**. Las instituciones de salud generan distintos tipos de información de los pacientes, según los sistemas utilizados. Estos sistemas generan continuamente información que tiene la oportunidad de ser explotada con la finalidad de proveer un apoyo en el diagnóstico de enfermedades. Dicha información generada puede ser de naturaleza estructurada (como valores de estudios de laboratorio, de microbiología, etc.). También, es posible que esta información se encuentre de forma no estructurada, como notas clínicas que contienen información de, por ejemplo, evolución y diagnóstico de los pacientes.
- **Extracción de conocimiento a partir de datos no estructurados**. Los diagnósticos e interpretaciones de algunos estudios pueden ser dados en forma textual por parte de médicos. Esta información es de gran importancia, pues dentro del proceso de diagnóstico y evolución del paciente, otros médicos utilizan dicha información para continuar con el tratamiento de los pacientes, o bien, dar un diagnóstico final. Mediante la aplicación de técnicas de Procesamiento del Lenguaje Natural (PLN), del inglés *Natural Language Processing (NLP)*, es posible obtener características cuantitativas que posteriormente puedan ser utilizadas por algoritmos de ML para ser clasificados.
- **Manejo de datos estructurados**. Parte del diagnóstico en pacientes es considerar la información obtenida de distintos análisis practicados en el paciente, así como información adicional recabada para el diagnóstico. Estos estudios y cierta información puede ser presentada en forma estructurada, es decir, consistir en valores numéricos o categóricos. Por ejemplo, valores obtenidos de estudios de laboratorio (química sanguínea, microbiología, etc.) o categóricos de información personal del paciente (sexo, estatus marital, religión, etc.). Este tipo de datos debe ser pre-procesado para que algoritmos de ML puedan utilizarlos en la clasificación.
- **Algoritmos de ML para datos médicos**. Con el objetivo de aprovechar la información médica generada por las instituciones de salud para brindar apoyo a los médicos

para el diagnóstico de enfermedades, es posible aplicar algoritmos de ML para dicha tarea. Los algoritmos pueden ser entrenados bajo distintos tipos de datos para generar modelos de clasificación. De esta manera, al consultar nuevas ocurrencias sobre estos modelos, estos deben ser capaces de predecir con un cierto grado de precisión la clase a la que pertenece dicha ocurrencia, considerando la tarea de clasificación.

Los puntos incluidos anteriormente constituyen los principales problemas que serán atacados por esta tesis. Principalmente el resolver cada uno de los puntos anteriores contribuirá a que la información generada por las instituciones de salud sea aprovechada para dar un apoyo en el diagnóstico de los tipos de cáncer considerados en el presente trabajo. De esta manera, se busca que la mayor cantidad posible de información, considerando las distintas naturalezas de la misma, sean integradas en algoritmos clasificación. Así, contribuyendo con una metodología de pre-procesamiento y clasificación de datos médicos para el apoyo en el diagnóstico de cáncer de pecho, hígado y pulmón.

## 1.2. Justificación

Gracias al crecimiento en la adopción de sistemas de cómputo en distintas áreas de la ciencia, ha sido posible generar grandes cantidades de información. La información generada en muchas ocasiones es utilizada solo como un medio de registro de las actividades realizadas en un área, desperdiciando el posible potencial que esta tiene para la toma de decisiones.

El área de la salud es una de las áreas que ha presentado un crecimiento en el uso de sistemas de computo y presenta un crecimiento en la cantidad de datos que genera. Incluso, hoy en día se presenta una adopción cada vez más generalizada de los Expedientes Médicos Electrónicos, del inglés *Electronic Health Record (EHR)*, que se presenta como una forma de tener de una manera más centralizada información relevante de pacientes. Distintos beneficios son posibles gracias a la inclusión de los EHRs [10]. Sin embargo, la posibilidad de tener información de pacientes de manera masiva y centralizada permite que investigadores aprovechen dicha información para realizar investigación en distintos aspectos clínicos (evolución de enfermedades, recurrencia de estas, etc.), mejorando así aspectos esenciales de la vida humana, como es la salud.

No obstante, la inclusión de grandes cantidades de información centralizada como son los EHR también trae consigo ciertos desafíos [11]. Dada la gran diversidad de los datos que se integran en los expedientes, estos pueden ser de naturaleza estructurada o no estructurada. Dependiendo de la naturaleza de los datos, es necesario llevar a cabo procesos distintos para poder utilizarla, según la tarea que se requiera.

Los modelos de ML son una herramienta que puede ser utilizada para aprovechar la información generada y así, aplicarla en distintas tareas en el área clínica, por ejemplo, modelos de clasificación y regresión [12]. Según las necesidades de la tarea en mano, se pueden utilizar distintos algoritmos de ML para cumplirla. La naturaleza de cada uno de los algoritmos es distinta entre sí (aunque puedan ser utilizados para una misma tarea), por lo cual en cada caso es necesario que los datos que se utilicen presenten ciertas características, esto para que los algoritmos puedan ser utilizados.

Considerando la característica de la información clínica y el gran potencial presentado por esta, es posible considerar su uso, en conjunto con algoritmos de ML para así obtener modelos que sean capaces de brindar información útil con el objetivo de apoyar a profesionales de la

salud en distintas tareas. Considerando la tarea de clasificación, esta puede ser utilizada para discriminar entre distintas enfermedades, además de aprovechar la capacidad de predicción de los modelos, y ser aplicado a nuevos casos.

El desarrollo de modelos de ML de clasificación que aprovechen la información clínica generada de pacientes, considerando su distinta naturaleza (estructurada y no estructurada) puede apoyar a los profesionales de la salud en la toma de decisiones. Además, los modelos pueden ser evaluados, mediante métricas específicas, esto permite que, además de ser evaluados, puedan ser mejorados, para así encontrar el más adecuado en la clasificación de ciertas enfermedades, particularmente en los tipos de cáncer. El poder proveer a un médico con una herramienta de clasificación para este tipo de enfermedades, puede apoyarlo en la toma de decisiones sobre el diagnóstico y tratamiento de la enfermedad. El modelo generado puede apoyar al médico en las siguientes áreas:

- Disminuir el tiempo requerido por los médicos en la revisión total de la información.
- Apoyar al médico para brindar una decisión más temprana sobre el diagnóstico de la enfermedad, ayudando así al tratamiento oportuno de esta.
- Brindar una clasificación basada en información de expedientes clínicos. Donde dicha información puede estar en constante crecimiento, mejorando la generalización del modelo.

## 1.3. Objetivos

Objetivo general:

- Evaluar tres modelos de *machine learning* para el apoyo en la clasificación de cáncer de pecho, cáncer de pulmón y cáncer de hígado, basados en datos estructurados y no estructurados de expedientes clínicos.

Objetivos específicos:

1. Desarrollar un método de pre-procesamiento de las notas clínicas de los pacientes.
2. Implementar un algoritmo de procesamiento de lenguaje natural en las notas clínicas para extraer información en forma de características numéricas.
3. Desarrollar un módulo para el pre-procesamiento de los datos estructurados de estudios de laboratorio, estudios de microbiología y datos personales del paciente.
4. Implementar el modelo de máquinas de soporte vectorial, perceptrón multi-capa y *AdaBoost* utilizando los datos obtenidos de notas clínicas, estudios de laboratorio, información personal del paciente y estudios de microbiología.
5. Evaluar los resultados de los modelos de machine learning para mejorar su desempeño.

## 1.4. Principales contribuciones

El trabajo principal presentado en esta tesis se basa en la clasificación de tres tipos de cáncer, utilizando la información contenida en expedientes clínicos electrónicos de los pacientes. Para poder llevar a cabo esta tarea, se presentan principalmente dos contribuciones: pre-procesamiento de datos de entrada para los algoritmos y evaluación y mejora de estos.

La presente tesis da una contribución al proponer una metodología de pre-procesamiento de datos provenientes de los EHR. Esta metodología considera la inclusión tanto de datos estructurados como no estructurados, para así poder aprovechar la mayor cantidad de información posible. Cada uno de los datos presenta ciertas características que deben ser consideradas para su pre-procesamiento. Por esto, es necesario que para cada uno se realice un proceso distinto. Una vez aplicado el proceso propuesto, los datos obtenidos pueden ser utilizados para la siguiente etapa, que es la evaluación y mejora de los algoritmos.

Otra contribución en la metodología es el proceso de evaluación del desempeño de los algoritmos, así como la mejora de estos. Este proceso considera ciertas métricas conocidas, con el fin de poder medir el desempeño de la clasificación. Utilizando estas métricas es posible definir si los modelos mejoran o empeoran ajustando sus *Hiper-parámetros*. De esta manera, es posible llevar a cabo un proceso de mejora del modelo, el cual considera el ajuste de *Hiper-parámetros*, así como su métrica de desempeño asociada, para determinar el mejor modelo de clasificación.

Así, considerando las dos principales contribuciones, se puede obtener el modelo que mejor se desempeña en la clasificación de cáncer de pecho, hígado y pulmón. Tomando en cuenta datos estructurados y no estructurados de los EHR.

## 1.5. Organización de la tesis

En la Sección 2 se presenta el marco teórico de la presente tesis. Presenta de manera general el concepto de cáncer, así como la definición de los tipos de cáncer tratados. También se presentan las bases teóricas del procesamiento de lenguaje natural, tanto su preprocesamiento como su representación vectorial. Finalmente se presenta el sustento teórico de los algoritmos de ML utilizados.

En la Sección 3 se presentan trabajos realizados por otros autores que exploran la clasificación de tipo de cáncer, ya sea utilizando datos estructurados como no estructurados. Se presentan principalmente los algoritmos utilizados por los autores, así como los resultados obtenidos. Para el caso de datos no estructurados, además de lo anterior, también se presenta un resumen de la metodología empleada para manejar estos tipos de datos.

La Sección 4 presenta la metodología propuesta en la presente tesis. Esta sección presenta el proceso propuesto para la extracción, pre-procesamiento, evaluación y mejora de los algoritmos de ML considerados.

La Sección 5 presenta la metodología propuesta aplicada a un conjunto de datos obtenidos de una base de datos clínica. Esta sección presenta la información utilizada, así como los resultados obtenidos en el conjunto de datos utilizado.

Finalmente, la Sección 6 presenta las conclusiones derivadas del presente trabajo, así como el posible trabajo a futuro que puede ser aplicado.

# Capítulo 2

# Marco teórico

La presente tesis aborda distintos temas para llevar a cabo la clasificación de pacientes con cáncer a través de datos estructurados y no estructurados. En este capítulo se presentará el sustento teórico de la investigación para llevar a cabo la clasificación de pacientes con cáncer. Los temas principales que se abordarán son el cáncer y los subtipos que se consideraron para el estudio, así como las técnicas utilizadas para la clasificación de la información, siendo estos, algoritmos de ML y técnicas de PLN. Finalmente, se mostrará la estructura de la base de datos clínica utilizada para llevar a cabo la investigación.

## 2.1. Cáncer

El cáncer es un concepto que permite describir diversas clases de enfermedades, en las cuales las células tienen un crecimiento descontrolado con el potencial de ser malignas como resultado de factores genéticos y ambientales [13]. Dado el proceso anterior, se forman tumores que pueden ser cancerígenos o no. Los tumores benignos, o no cancerígenos, no se extienden a otras partes del cuerpo. En cambio, los tumores malignos se extienden a células sanas. En la Figura 2.1 se muestran algunas estadísticas de cáncer en los Estados Unidos, donde se muestra la alta tasa de incidencia y mortalidad que presenta la enfermedad.

El cáncer mantiene su crecimiento mediante un proceso llamado metástasis, mediante el cual las células cancerígenas viajan a través del sistema linfático y eventualmente forman tumores en otras partes del cuerpo. Existen más de cien tipos de cáncer, y estos usualmente son nombrados por los órganos o tejidos donde se forman los tumores.

**Cáncer de pecho** El cáncer de écho es el cáncer mas diagnosticado en mujeres en el mundo [15]. Se cree que este tipo de cáncer es dependiente de las hormonas, con un incremento en el riego mientras mayor sea la exposición y los niveles de estrógenos. Su diagnóstico usualmente se da a través de una biopsia de una masa palpable del pecho o por una biopsia de una anomalía mamográfica [16]. En la Figura 2.2 se muestra una imagen ilustrativa sobre el cáncer de pecho.

**Cáncer de pulmón** El cáncer de pulmón es el más diagnosticado en hombres en el mundo [15]. La principal causa de este tipo de cáncer se da por fumar, tanto en fumadores activos como pasivos. Incluso hay casos en los cuales este tipo de cáncer se da en personas que no han sido expuestas a factores de riesgo (tabaco, asbesto o radiación) presentan

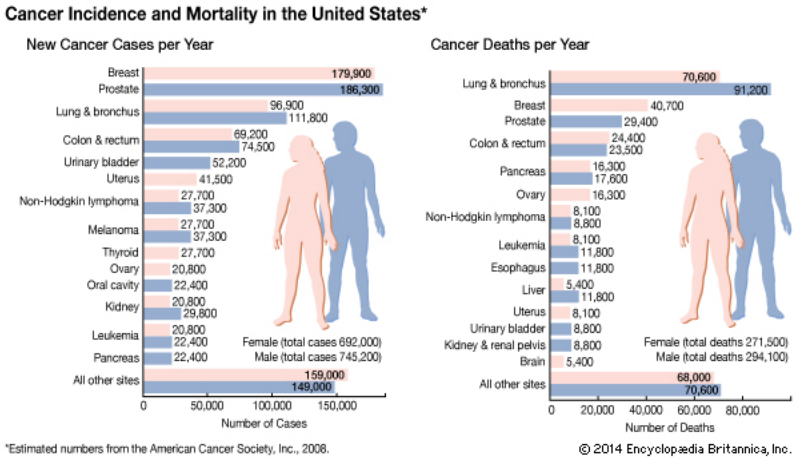



Figura 2.1: Incidencia de cáncer y tasa de mortalidad en los Estados Unidos, tomada de [14].

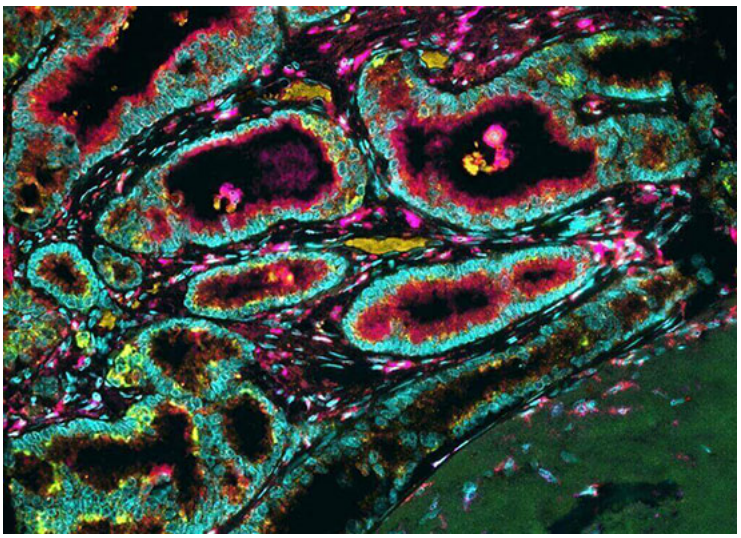


Figura 2.2: Sección del tejido del pecho de una paciente mujer con cáncer invasivo de pecho, tomada de [17].

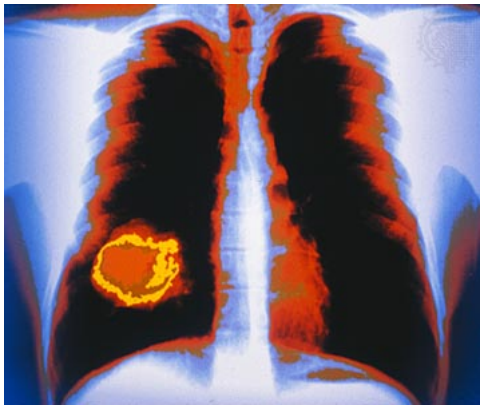


Figura 2.3: Radiografía con mejora en el color mostrando (en color amarillo) un tumor del pulmón derecho, tomada de [18].

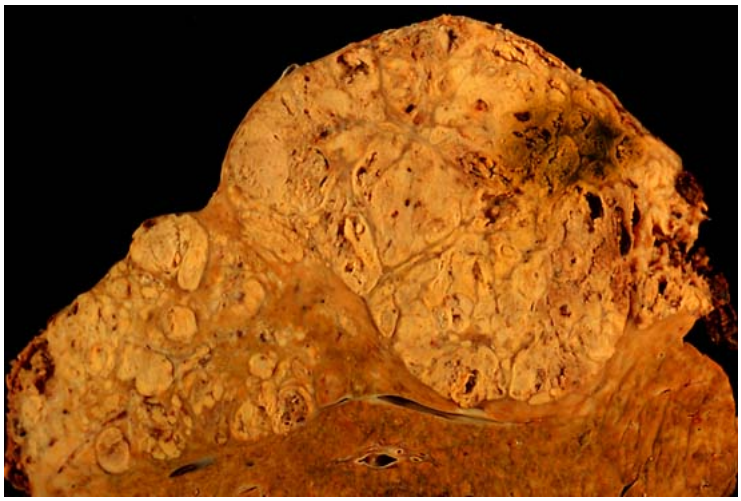


Figura 2.4: Corte longitudinal del hígado con carcinoma hepatocelular.

la enfermedad, a lo cual se continua con la investigación orientada a factores genéticos y dietéticos [16]. La Figura 2.3 muestra una imagen ilustrativa sobre el cáncer de pulmón.

**Cáncer de hígado** El cáncer de hígado es el segundo tipo de cáncer con mayor mortalidad en hombres en el mundo [15]. El cáncer que se esparce al hígado es más común al que comienza con células del hígado. Su causa más común proviene de mutaciones genéticas, aunque también es común que comience por lesiones en el órgano, como por ejemplo daño por infecciones de hepatitis [16]. De igual manera otros factores de riego comunes son la cirrosis y la diabetes. En la Figura 2.4 se muestra una imagen ilustrativa del cáncer de hígado.

## 2.2. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural investiga el uso de las computadoras para procesar o entender lenguajes humanos con el propósito de realizar tareas útiles [19].

El lenguaje natural puede ser visto como un sistema construido para transmitir semántica, donde dicho sistema tiene una naturaleza simbólica. Dicha naturaleza puede ser vista como señales. La primera señal física u observable se trata del texto y la segunda una señal producida por el habla.

En ambos casos, el análisis del lenguaje requiere tareas que pueden ser específicas, según el procesamiento de lenguaje hablado o textual. Además, en ambos casos es posible utilizar un análisis en común como el *conocimiento del lenguaje* para llevar a cabo una tarea en específica. Un ejemplo de análisis específico para el lenguaje hablado puede considerarse bajo las tareas de *reconocimiento del lenguaje* o *síntesis del lenguaje*, donde es necesario que los sistemas tengan conocimiento sobre la fonética y fonología del lenguaje.

En el caso de lenguaje en forma textual, por ejemplo, es necesario realizar procesamiento adicional al texto para eliminar ciertos caracteres o palabras que no sean de utilidad para la tarea, o bien, en el caso de las palabras escritas de manera errónea, puede ser necesario identificarlas y sustituirlas por su versión correcta.

Continuando con las particularidades del análisis de la representación textual del lenguaje, las siguientes secciones describen algunos procesos comunes a realizar en el PLN de este tipo.

### 2.2.1. Pre-procesamiento

El pre-procesamiento del texto es requerido para convertir un formato no estructurado en uno estructurado con una representación multi-dimensional [20]. En el texto, usualmente puede ocurrir que los datos contenidos (caracteres o palabras) puedan contener información irrelevante. Además, el problema de la ambigüedad en el lenguaje puede provocar que los significados de las palabras puedan variar dependiendo del contexto en el cual se usa, o bien, que en el contexto bajo el cual se realiza el análisis, existan palabras que no generen información relevante sobre el texto.

Basándose en lo anterior se puede considerar que es necesario realizar, en primer instancia un análisis a nivel de palabras para poder manejar casos como los que se mencionan. El proceso de dividir una secuencia de caracteres en una secuencia de palabras es conocido como *segmentación*, donde cada palabra en el texto es representada como un *token*.

El proceso mediante el cual se realiza la segmentación requiere un conocimiento del lenguaje que se requiere analizar, e.g., en el idioma español podemos considerar que un *token* es representado por la palabra que se encuentra entre dos espacios vacíos, pero en cambio, al analizar el lenguaje chino, puede que este no sea el caso. Algunas de las tareas comunes en el pre-procesamiento se presentan en [20], ilustrado en la Figura 2.5 y son descritos a continuación:

1. Extracción del texto: Dependiendo del formato desde el cual se extrae el texto, su procesamiento puede cambiar. Considerando la extracción de un documento web en formato HTML, se puede aprovechar la estructura de este tipo de documentos para realizar búsquedas en etiquetas específicas para obtener la información. Sin embargo,

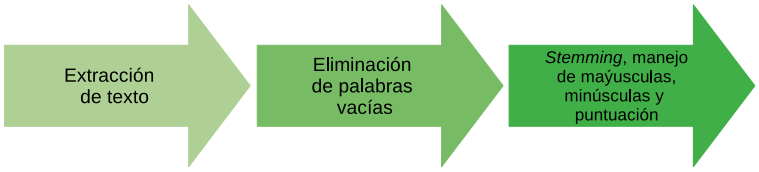



Figura 2.5: Flujo de pre-procesamiento para datos textuales.

las notas escritas en texto libre pueden tener muy poca o nula estructura para tomar como referencia, por lo cual el proceso de extracción textual en ambos casos difiere.

2. Eliminación de palabras vacías: Las palabras vacías son consideradas como aquellas que no aportan información al texto que se analiza. Usualmente estas palabras son preposiciones (e.g., a, sobre, ante, por, etc.). También este tipo de palabras puede ser definida con base en el dominio del texto. Por ejemplo, en un dominio médico, las palabras que son de interés son aquellas inherentes a este dominio, las palabras comunes pueden ser desechadas para facilitar las tareas posteriores.

3. *Stemming*, manejo de mayúsculas, minúsculas y puntuación: El proceso de *stemming* se refiere a llevar a una palabra a su base léxica [21]. Por ejemplo, en conjugaciones de verbos como procedieron, procederán, etc., se pretende representar esta palabra como su base léxica, que es proceder. Esto con el fin de disminuir la dimensionalidad de las palabras en el texto, es decir, disminuir el número de palabras en el texto. Además, con este proceso se lleva cada una de estas variaciones a una representación común. Por otra parte, cuando existen palabras que tienen una combinación de mayúsculas y minúsculas, se debe definir si esta combinación es de importancia para la tarea a realizar. Es decir, en algunos casos, y dependiendo del lenguaje, la representación de una palabra que inicia con mayúscula tiene un significado semántico, por lo cual el análisis requiere que esta palabra se mantenga con esta representación. Por último, para el manejo de la puntuación, nuevamente se debe considerar el lenguaje del texto, pues en algunos casos es necesario que, por ejemplo, los puntos se mantengan para realizar el proceso de segmentación.

## 2.3. Modelos de representación de documentos

Una vez que el texto se ha pre-procesado para eliminar la mayor cantidad de información innecesaria para la tarea de clasificación en la cual se trabaja, y además de obtener la extracción de *tokens* del texto, es necesario obtener una representación numérica del texto a clasificar.

Dicha representación es necesaria para que los algoritmos de clasificación puedan procesar dicha representación. A continuación se presentan algunos modelos de representación de textos.

### 2.3.1. Modelo de bolsa de palabras

Uno de los algoritmos clásicos para la representación de documentos en un espacio de vectores es la representación mediante el modelo de bolsa de palabras, del inglés *Bag-of-*

| Término | Símbolo | Significado |
|---|---|---|
| Frecuencia de términos | $tf_{i,j}$ | Número de ocurrencias de la palabra $w_i$ en el documento $d_j$ |
| Frecuencia en documento | $df_i$ | Número de documentos en la colección donde la aparece palabra $w_i$ |
| Frecuencia en la colección | $cf_i$ | Total de ocurrencias de la palabra $w_i$ en la colección |

Tabla 2.1: Términos utilizados para la normalización TF-IDF.

*Words (BOW)*. Este término se utiliza para denominar las representaciones de texto que no toman en consideración el ordenamiento de los términos (*tokens*), pero sí la ocurrencia de estos en cada documento del corpus. El proceso considera el pesado de términos por documento, para que dichos pesos representen a cada documento como un vector de longitud fija, compuestos por los pesos asignados a cada término dentro del documento. El peso asignado a cada palabra puede ser asignado considerando distintos enfoques. Por ejemplo, el peso por frecuencia puede ser considerado como la frecuencia con la que un término aparece en el documento, y dicha frecuencia es asignada como el peso de dicho término en el documento. O bien, puede considerarse el peso simplemente indicando con un '1' si el término existe dentro de un documento en específico, o con un '0' en caso contrario, que se conoce como pesado lógico (*boolean*). Sin embargo, estos dos tipos de cálculo para el peso de las palabras pueden no ser suficientemente representativos del documento, es por ello que la asignación de pesos más utilizada en el enfoque BoW para la clasificación de textos es el TF-IDF. Este tipo de normalización parte de la premisa de que un término que aparece con gran frecuencia en un documento, difícilmente puede ayudar a diferenciarlo de los demás.

La normalización TF-IDF se basa en *frecuencia de términos*, *frecuencia en documentos* y *frecuencia en la colección* que se definen en la Tabla 2.1 [22].

La frecuencia de términos nos brinda información acerca de los términos que mas se mencionan en un documento, que puede significar que dicho término pueda ser útil para caracterizar el documento. Una medida común para la frecuencia de términos está dada por: $tf = \sqrt{tf}$ o $tf = 1 + log(tf), tf > 0$. Este proceso es realizado dado que la frecuencia es simple; es decir, el número de ocurrencia de las palabras, puede representar información errónea sobre la importancia del término en el documento. Por ejemplo, un documento puede tener una frecuencia de 3 para un término, mientras que otro tenga solo 1, pero esto no quiere decir que para el primer documento dicho término es 3 veces más representativo.

También es importante recalcar que el hecho de que un documento cuente con una frecuencia alta para algunos términos, estos sean representativos del texto. Esto se puede dar en textos que hablen sobre temas similares, es común que en la colección de documentos los términos se repitan con alta frecuencia dentro del documento, así como en otros documentos. Esto indica que este término ya no es representativo de un documento en particular. Para combinar la frecuencia de términos y la frecuencia en documento para indicar el peso de un término en TF-IDF se utiliza la Ecuación 2.1.

$$Peso(i,j) = \begin{cases} (1 + log(tf_{i,j}))log\dfrac{N}{df_i} & \text{si } tf_{i,j} \geq 1 \\ 0 & \text{si } tf_{i,j} = 0 \end{cases} \tag{2.1}$$

En la Ecuación 2.1, el término $N$ representa el número total de documentos. Como podemos observar en la Ecuación 2.1, a los términos que no ocurren en el documento se les asigna un peso de 0 y el uso del término $log(\frac{N}{df_i})$ hace que los términos que aparecen de manera

recurrente a través de los documentos, tengan un peso de cero, mientras que términos poco comunes tengan un mayor peso que ayude a caracterizarlos. Si bien el proceso BoW permite una representación numérica de documentos, uno de los problemas que presenta es que dicha representación se da solo como una colección de palabras, sin relación alguna entre ellas.

### 2.3.2. *Modelado del lenguaje mediante redes neuronales*

En años recientes con el aumento en el poder de cómputo y la disponibilidad de grandes conjuntos de datos, se ha hecho más común el utilizar modelos basados en redes neuronales para obtener la representación de texto. Esto con el objetivo de captar la mayor información posible al codificar un texto en una representación vectorial.

Una de las primeras propuestas de utilizar redes neuronales para llevar a cabo la codificación de los vectores como es presentada en [23], conocida como modelado del lenguaje mediante redes neuronales, del inglés *Neural Network Language Modeling (NNLM)*. Se propone el *aprendizaje distribuido* de las palabras en un documento. Donde este aprendizaje se basa en que la representación de una palabra como vectores, depende de las palabras en el contexto que se usa. Esto quiere decir que la representación de una palabra se da según la probabilidad que el vector de la palabra se de dependiendo de los vectores del contexto. Esto permite que al aprender la representación basada en el contexto, los vectores encontrados tengan relación entre sí, y puede generalizarse dado que las mismas palabras se pueden usar en contextos similares.

El modelo propuesto considera una secuencia de palabras $w_1, \cdots, w_t$, donde $w_t \in V$ y donde $V$ es el vocabulario. Se tiene como objetivo encontrar un modelo que obtenga la probabilidad condicional de la palabra siguiente en relación con las anteriores como se muestra en la Ecuación 2.2.

$$f(w_t, \cdots, w_{t-n+1}) = \widehat{P}(w_t | w_1^{t-1}) \tag{2.2}$$

Se propone la descomposición de la Ecuación 2.2 en dos partes:

1. Una transformación $C$ de cualquier elemento $i \in V$ a un vector real $C(i) \in R^m$, que representa los vectores distribuidos de características. $C$ es representado por una matriz $|V| \times m$.

2. Una función de probabilidad sobre las palabras expresadas con $C$: una función $g$ que transforma los vectores de entrada para el contexto de una palabra $(C(w_{t-n+1}), \cdots, C(w_{t-1}))$ a una distribución de probabilidad de las palabras en $V$ para la siguiente palabra $w_t$. La salida de g es un vector donde la $i$-esima entrada estima la probabilidad $\widehat{P}(w_t = i | w_1^{t-1})$.

La función compuesta por las dos partes anteriores se representa como se muestra en la Ecuación 2.3.

$$f(i, w_{t-1}, ..., w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1})) \tag{2.3}$$

De esta manera se indica que la representación de la palabra $i$ es denotada por la entrada $C(i)$ de la matriz de vectores. Además, proponen que la función $g$ puede ser implementada por una *Feed-Forward Recurrent Neural Network (FFRNN)* y proponen la arquitectura mostrada en la Figura 2.6. En este figura se muestran las palabras codificadas como entradas en la matriz C. Se considera el uso de $w_{t-n+1}$ palabras de contexto, a las cuales se le aplica una

función $g$ de transformación representada por la función de activación *tanh* de una FFRNN. Finalmente, se obtienen las probabilidades de las palabras que pueden aparecer dado el contexto de entrada mediante una función *softmax*.

### 2.3.3. Word2Vec

Considerando las bases de NNLM presentado en [23], se propone un modelo basado en redes neuronales para obtener la representación vectorial de palabras, conocido como *word2vec* [24]. El objetivo principal es realizar una representación vectorial de palabras, basándose en una arquitectura de redes neuronales. El modelo propuesto se basa en dos arquitecturas: *Continuous Bag-of-Words (CBOW)* y *Continuous Skip-gram Model*. La arquitectura CBOW se basa en el concepto tradicional de BoW en el sentido que el orden de las palabras no influye en la proyección. Además, usa una representación continua del contexto de palabras, similar a la que se presenta en [23]. La diferencia es que en este caso, se considera que la representación en la capa de proyección de la red neuronal, se toma en cuenta el promedio de la representación vectorial del contexto. Adicionalmente, se considera que las palabras que se utilizan para predecir una palabra en particular, se toman tanto del contexto como del futuro. Se puede observar en la Figura 2.7a que el modelo CBOW se utiliza para predecir la palabra dado el contexto utilizado. Además, se introduce la arquitectura *Skip-gram* (ver Figura 2.7b) que busca el proceso contrario a CBOW. En esta arquitectura se busca predecir el contexto basándose en una palabra. Así, se predicen las palabras en una distancia $C$ de la palabra de entrada. En ambos casos la representación vectorial se toma como la entrada correspondiente para la matriz de proyección, como se propuso en el modelo NNLM.

### 2.3.4. *Vectores de párrafo*

Extendiendo la idea presentada en [24], el modelo de vectores de párrafo, del inglés *Paragraph Vector (PV)* [25] utiliza una representación vectorial de las palabras a través de redes neuronales, que puede considerar documentos enteros. El acercamiento de PV está basado en el aprendizaje de la representación vectorial de las palabras y una matriz adicional en la cual cada columna es la representación de cada párrafo, donde se define como párrafo a un conjunto de oraciones, mismos que pueden considerarse como documentos completos.

En la fase de aprendizaje de los vectores, el objetivo es predecir una palabra dada otras palabras en el contexto, de manera similar al modelo CBOW. Formalmente, dado un conjunto de palabras de aprendizaje $w_1, w_2, w_T$, se busca maximizar la probabilidad logarítmica promedio, como se muestra en la Ecuación (2.4).

$$\frac{1}{T}\sum_{t=k}^{T-k} log\ p(w_t \mid w_{t-k}, \ldots, w_{t+k}). \tag{2.4}$$

La segunda fase corresponde a la tarea de predicción, misma que es realizada por un clasificador multiclase. Se propone el uso de *softmax* como se muestra en la Ecuación (2.5).

$$p(w_t \mid w_{t-k}, \ldots, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}. \tag{2.5}$$

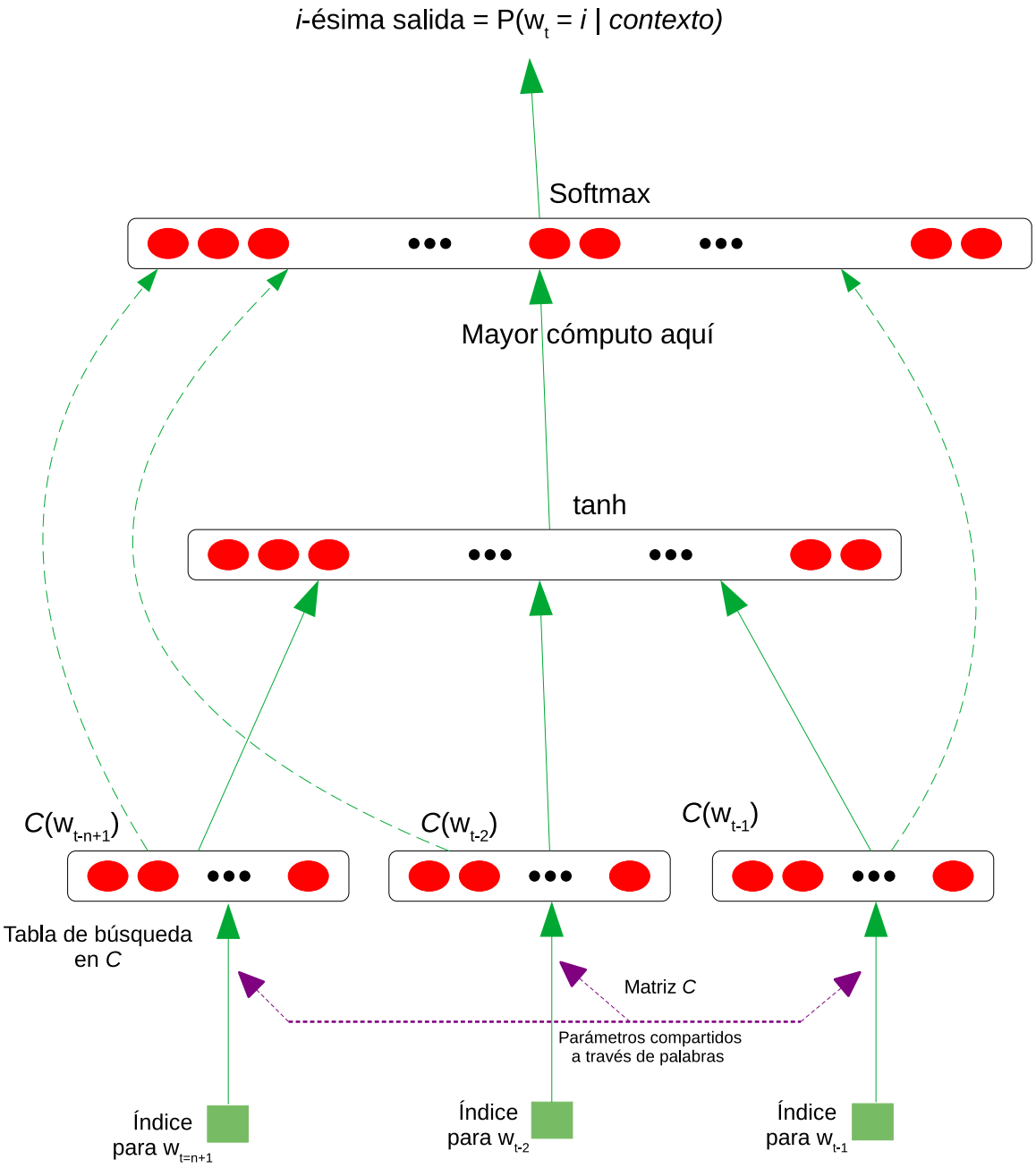
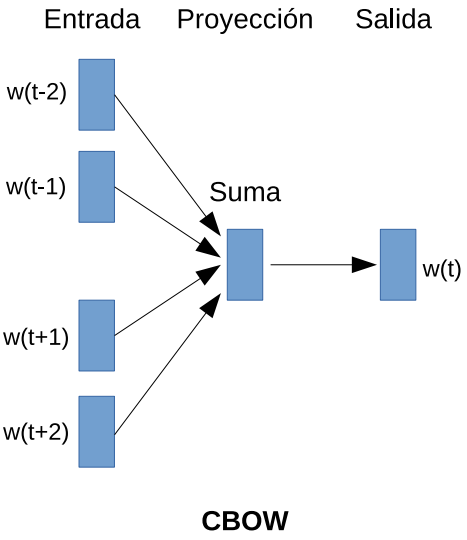
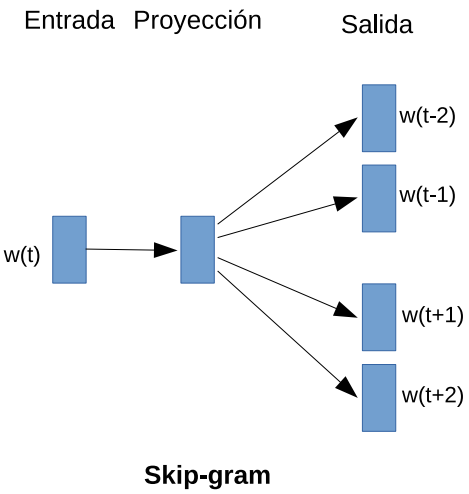



Figura 2.6: Modelo de representación NNLM, tomado de [23]




(a) Arquitectura de represetación vectorial de palabras CBOW, tomada de [24]




(b) Arquitectura de represetación vectorial de palabras mediante Skip-gram, tomada de [24]

Figura 2.7: Modelos de representación de textos

En la Ecuación 2.5, el término $y_i$ corresponde a la probabilidad logarítmica sin normalizar para cada palabra $i$ obtenida con la Ecuación (2.6).

$$y = b + Uh(w_{t-k}, \ldots, w_{t+k}W). \tag{2.6}$$

En la Ecuación 2.6, $U, b$ son parámetros de *softmax*, $h$ es construido como el promedio o concatenación de la matriz de párrafos $D$ y la matriz de vectores de palabras $W$.

El entrenamiento de las palabras y vectores de los párrafos es realizado a través de *Stochastic Gradient Decent (SGD)*. En cada paso de SGD, un ejemplo de contexto de longitud fija es extraído de un párrafo aleatorio. El error del gradiente es calculado y los pesos de la red son actualizados. Una vez que el entrenamiento es completado, los vectores obtenidos pueden ser utilizados como la representación de cada párrafo. Eventualmente, estos vectores pueden ser utilizados como entradas para algoritmos de clasificación de ML.

## 2.4. Aprendizaje Automático

El término aprendizaje automático se define como un programa de computadora que aprende a partir de la experiencia, con respecto a una clase de tareas y medida de desempeño [26]. Estos algoritmos presentan parámetros que definen el comportamiento particular de cada uno, y finalmente generan un modelo que puede ser utilizado para una tarea en específico. El hecho de que los algoritmos puedan realizar un aprendizaje a partir de la experiencia es dado gracias a que estos algoritmos reciben datos de entrada para realizar el proceso de aprendizaje. Una clasificación usual de los algoritmos de ML se basa en el tipo de aprendizaje que llevan a cabo, esta clasificación es la siguiente [27]:

- **Aprendizaje supervisado**: Como entrada al algoritmo se provee un conjunto de datos, etiquetados con su valor de salida esperado. De esta manera el algoritmo es

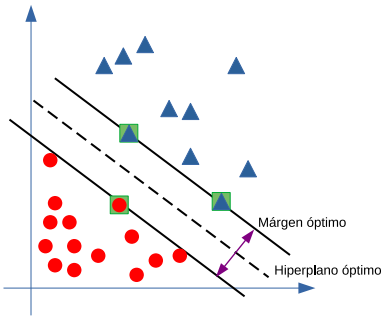



Figura 2.8: Concepto de SVM para separar dos clases, tomada de [29].

entrenado bajo estos datos para generalizar y responder correctamente a todas las posibles entradas.

- **Aprendizaje no supervisado**: En este caso como valores entrada no son etiquetados y, en su lugar, el algoritmo busca similitudes entre los valores de entrada para que puedan ser categorizados juntos.

- **Aprendizaje reforzado**: Puede ser considerado como un punto medio entre las dos categorías anteriores. En este caso se le indica al algoritmo cuando su respuesta es errónea, sin embargo, no se le indica como corregirla. De esta manera el algoritmo explora diferentes posibilidades para obtener la respuesta correcta.

- **Aprendizaje evolutivo**: Esta categoría se inspira en el proceso biológico de adaptación, supervivencia y reproducción [28]. Se utilizan estos conceptos en conjunto con el de una medida de desempeño *fitness* para indicar qué tan buena es una solución.

Dentro de los algoritmos supervisados, se encuentra la tarea de clasificación, definida como el problema de tomar vectores de entrada y decidir a qué clase pertenece, tomando en cuenta el conjunto de datos de entrenamiento. A continuación se presentan los algoritmos considerados en esta tesis para llevar a cabo la clasificación de tipos de cáncer. Estos algoritmos son las máquinas de soporte vectorial, perceptrón multi-capa y AdaBoost.

### 2.4.1. Máquinas de Soporte Vectorial

Los algoritmos denominados *Support Vector Machines (SVM)* fueron descritos en [29] como un algoritmo de ML cuya idea fue transformar vectores de entrada en un espacio de características con mayores dimensiones, a través de una función de transformación no lineal. En dicho espacio se construye un área de decisión con propiedades especiales para asegurar la habilidad de generalización del algoritmo.

El procesa funciona tomando en cuenta vectores de prueba y con ellos generar *Vectores de Soporte*, con los cuales se busca separar los conjuntos de datos con el mayor margen posible entre los vectores de soporte, a través de un hiper-plano. Esto con la finalidad que dichos vectores puedan generalizar la separación considerando ahora datos de prueba. En la Figura 2.8 se muestra un ejemplo de separación de dos clases con los vectores de soporte.

A continuación se presenta la formulación del problema de dividir un conjunto de datos en dos clases, basándose en los hiperplanos de las SVM. La formulación es descrita en [30].

Para definir la separación de datos con un ejemplo de dos clases consideramos las etiquetas +1/-1 para cada clase. Además, consideramos $X = \{x^t, r^t\}$ donde $r^t = +1$ si $x^t \in C_1$ y $r^t = -1$ si $x^t \in C_2$. El objetivo es encontrar $w$ y $w_0$ tal que:

$$\begin{aligned} W^T x^t + w_0 \geq +1 \text{ para } r^t = +1 \\ W^T x^t + w_0 \leq +1 \text{ para } r^t = -1 \end{aligned} \tag{2.7}$$

que puede ser escrito como:

$$r^t(w^T x^t + w_0) \geq +1 \tag{2.8}$$

Como se busca la maximización del margen, se usa la hipótesis de que el hiperplano de separación óptima es aquel que maximiza el margen.

Si consideramos que la distancia de un punto $x^t$ al hiperplano discriminante es:

$$\frac{|w^T x^t + w_0|}{\| w \|} \tag{2.9}$$

donde cuando $r^t \in \{-1, +1\}$ puede ser escrito como:

$$\frac{r^t w^T x^t + w_0}{\| w \|} \tag{2.10}$$

Además, se busca que al menos para un valor $\rho$ :

$$\frac{r^t w^T x^t + w_0}{\| w \|} \geq \rho, \forall t \tag{2.11}$$

En este caso $\rho$ representa el margen a maximizar. Como se tiene una infinidad de soluciones, se considera $\| w \| = 1$, en donde ahora para maximizar el margen se busca minimizar $w$. De esta manera, el objetivo de las máquinas vectoriales es minimizar:

$$\min \frac{1}{2} \| w \|^2 \text{ sujeto a } r^t(w^T x^t + w_0) \geq +1, \forall t \tag{2.12}$$

La formulación anterior considera que el problema es linearmente separable, en caso contrario, una de las técnicas utilizadas es transformar el conjunto de entradas a una dimensionalidad mayor. Por esto, es necesario reformular el problema para que la complejidad no dependa de la dimensionalidad del mismo, sino que dependa del número de instancias de entrenamiento. Esta reformulación utilizando *Multiplicadores de Lagrange* $\alpha^t$ es:

$$L_p = \frac{1}{2} \| w \|^2 - \sum_t \alpha^t r^t (w^T x^t + w_0) + \sum_t \alpha^t \tag{2.13}$$

la cual debe ser minimizada con respecto a $w, w_0$ y maximizada con respecto a $\alpha^t \geq 0$. Una vez resuelto para $\alpha^t$ podemos observar que la mayoría de las instancias de entrenamiento desaparecen con $\alpha^t = 0$ y solamente un pequeño porcentaje se mantiene para $\alpha > 0$. Por esto, dichos puntos de entrenamiento $x^t$ son los vectores de soporte. Estos puntos son los que satisfacen :

$$r^t(w^T x^t + w_0) = 1 \tag{2.14}$$

y se encuentran en el margen. Con esto podemos calcular $w_0$ para cualquier vector como:

$$w_0 = r^t - w^T x^t \tag{2.15}$$

Para las SVM los puntos que cumplen con la condición $r^t(w^T x^t + w_0) > 1$ son los puntos de entrenamiento que son clasificados correctamente por el hiperplano. Esto quiere decir que dichos puntos no generan información para cambiar la dirección del hiperplano que divide las clases. Con esto consideramos que si removemos dichos puntos, no se afectaría la solución obtenida. Por esto, las SVM toman como información relevante los puntos que se encuentran en la periferia de la región de decisión.

Para extender la idea a casos donde no es posible la separación linear de las clases, se integra una nueva variable, llamada *Variable de Holgura* $\xi \geq 0$ que representa la desviación de un punto de entrenamiento del margen. Se consideran dos tipos de desviación: aquellas instancias que se encuentren en el lado incorrecto del hiperplano y son erróneamente clasificadas y aquellas que son correctamente clasificadas, pero que se encuentran dentro del margen. La integración de la variable de holgura en la restricción es:

$$r^t(w^T x^t + w_0) \geq 1 - \xi^t \tag{2.16}$$

donde considera que $\xi^t = 0$ indica que $x^t$ no tiene error. En el caso de $0 < \xi^t < 1$ indica que $\xi^t$ se encuentra clasificado correctamente, pero dentro del margen y $\xi \geq 1$ indica que $x^t$ esta erróneamente clasificado. Considerando el número de instancias mal clasificadas se considera el error como:

$$\sum_t \xi^t \tag{2.17}$$

Y se integra dentro de la función objetivo de las SVM como un término de penalización como:

$$_p = \frac{1}{2} \parallel w \parallel^2 + C \sum_t \xi^t \tag{2.18}$$

donde se busca penalizar a la función por una constante y la suma de los errores que han sido clasificados de manera errónea, además de aquellos que se encuentren dentro del margen. Considerando la integración del parámetro $C$ en la formulación mediante multiplicadores de Lagrange, ahora la restricción se convierte en:

$$\sum_t \alpha^t r^t = 0 \text{ y } 0 \leq \alpha^t \leq C, \forall t \tag{2.19}$$

Por esta restricción, podemos observar al parámetro $C$ como un *Hiper-parámetro* que indica la penalización que se integra a la clasificación errónea de las instancias de prueba.

Como se mencionó anteriormente, el problema considera que las instancias son linearmente separables. Para el caso en el que esto no sea válido, se utiliza un acercamiento conocido como *kernel trick*. Este proceso transforma las instancias de entrada a través de funciones base de la forma:

$$z = \phi(x) \text{ donde } z_j = \phi_j(x), j = 1, \cdots, k \tag{2.20}$$

para transformar de un espacio d-dimensional al espacio $z$ k-dimensional. La función objetivo se mantiene como se muestra en la Ecuación 2.18, pero ahora la restricción cambia a:

$$r^t w^T \phi(x^t) \geq 1 - \xi^t \tag{2.21}$$

El objetivo es aplicar la *función kernel* $K$, representada como una matriz, directamente en el espacio original, en lugar de calcular la función base $\phi(x^t)$.

Finalmente, para extender el modelo para tomar en cuenta mas de dos clases, en [31] se propone el uso de una función sigmoide a la salida del clasificador usual. Esto para obtener las probabilidades de la clasificación, pudiendo ser adaptada para decisiones mas globales, como por ejemplo, la extensión a un modelo de más de dos clases. Sin embargo, el acercamiento de *One-vs-All*(Uno contra los demás) es el método preferido para extender este modelo a más de dos clases. La idea a seguir para este acercamiento, es generar un modelo de clasificación por cada una de las clases del conjunto de datos.

### 2.4.2. Perceptrón Multi-Capa

Los modelos basados en redes neuronales buscan un comportamiento similar al del cerebro humano. El comportamiento del cerebro y sus mecanismos han sido estudiados por distintos neurocientíficos, que buscan su entendimiento para plasmar o simular dicho comportamiento en distintos ámbitos.

En específico, las redes neuronales aplicadas al cómputo, buscan que la simulación del comportamiento cognitivo del cerebro ayude a la solución de problemas computacionales de manera más óptima.

Haciendo similitud con la unidad básica en el cerebro, la neurona, en [32] se propone al *Perceptrón* como una máquina que simula un sistema nervioso hipotético e "ilustra las propiedades fundamentales de un sistema inteligente". Las principales características que describe para el perceptrón son las siguientes:

1. Conexiones físicas del sistema, que son involucradas en el aprendizaje. Considera que las primeras conexiones se realizan de manera altamente aleatoria.

2. Conforme se realiza el aprendizaje es probable que el estímulo en cierto grupo de células no cambie, dado que las interacciones entre ciertos grupos de células se mantienen.

3. Los efectos a largo plazo de un estímulo en particular tienden a formar rutas en los mismos grupos de células similares entre sí.

4. La aplicación de estímulos positivos o negativos pueden facilitar o dificultar conexiones.

5. La similitud en el sistema tiende a activar el mismo conjunto de células ante estímulos similares.

Posterior al modelo propuesto por en [32], un modelo computacional del perceptrón de una capa y sus limitantes es propuesto [33].

De esta manera consideramos al perceptrón como la unidad básica de procesamiento para las redes neuronales, que cuenta con conexiones que pueden venir del ambiente o de otros perceptrones. Así, tenemos que para cada entrada $x_j \in \Re, j = 1, \ldots, d$, se tiene una *conexión de peso* y una salida $y$. Esta salida puede ser descrita de la manera más simple como la suma de las conexiones de entrada. Esto es representado por:

$$y = \sum_{j=1}^{d} w_j x_j + w_0 \tag{2.22}$$

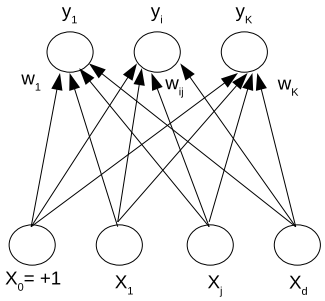



Figura 2.9: Modelo de perceptrón con $K > 2$ salidas.

donde se considera a $w_0$ como el peso proveniente de una unidad de *bias*. Al reformular la Ecuación 2.22 como producto punto de dos vectores tenemos:

$$y = w^T x \tag{2.23}$$

donde consideramos que $w = [w_0, w_1, \ldots, w_d]^T$ y $x = [1, x_1, \ldots, x_d]^T$ son los vectores de pesos y *bias* respectivamente para una entrada en específico. El proceso de aprendizaje consiste en "aprender" los pesos $w$ para una entrada $x$, para obtener la salida deseada $y$. En el caso de considerar dos clases linearmente separables, podemos formular la salida esperada del perceptrón como una recta de la forma:

$$y = wx + w_0 \tag{2.24}$$

De esta manera, es posible encontrar un hiperplano, que divida al conjunto de datos. Así, es posible definir una función de límite (*threshold*) para la salida del perceptrón. Este tipo de función indica con 1 que los valores de entrada en el perceptrón excedieron cierto límite y 0 en otro caso. Para una clasificación de dos clases esta representación indica:

$$\begin{cases} C_1 \text{ si } s(w^T x) > 0 \\ C_2 \text{ en otro caso} \end{cases} \tag{2.25}$$

donde $C$ representa la pertenencia a la clase 1 o 2, respectivamente, y $s$ representa a la función de límite. También cabe destacar, que la función puede tomarse como propia de cada perceptrón, para calcular una salida basándonos en sus entradas, esta función también es conocida como *función de activación*.

Para el caso de más de dos clases, $K > 2$ se considera que cada perceptón de salida representa a una de las clases $k$, con un vector de pesos asociados de la forma:

$$y_i = \sum_{j=1}^{d} w_{ij} x_j + w_{i0} = w_i^T xy = Wx \tag{2.26}$$

donde $w_{ij}$ es el peso asignado para la entrada $x_j$ a la salida $y_i$. $W$ la matriz de pesos con filas $w_{ij}$ para los $K$ perceptrones. De esta manera, para obtener la clasificación, asignamos la clase $C_i$ si $y_i$ cumple con ser la salida de valor máximo. La representación de este modelo se presenta en la Figura 2.9.

Además, es usual que se requieran las probabilidades de pertenencia a la clase, posterior al procesamiento de la función aplicada en cada perceptrón. Usualmente en una implementación

de redes neuronales, esta función aplicada a la salida del perceptrón es *softmax* como se muestra a continuación:

$$y_i = \frac{e^{w_i^T}}{\sum_k e^{w_k^T}} \tag{2.27}$$

Durante el proceso de aprendizaje para los perceptrones, este se puede dar de dos maneras, aprendizaje en línea o fuera de línea (*online learning* y *offline learning*, respectivamente). El aprendizaje en línea considera que el conjunto total de datos de entrenamiento para el perceptrón no se conoce. En este caso, el aprendizaje se realiza con cada instancia de datos de entrenamiento, en cada instancia se calcula el error y se actualizan los parámetros del perceptrón. En el aprendizaje fuera de línea, el conjunto de datos de entrenamiento se conoce en su totalidad, por lo cual el proceso de aprendizaje se realiza iterando sobre este conjunto, considerando el error en el conjunto completo de datos.

Es usual que las implementaciones de redes neuronales utilicen la implementación de aprendizaje en línea. En el escenario para una implementación de $K > 2$ clases, con una función de error diferenciable, se considera la técnica de descenso de gradiente (*gradient descent)* para indicar la actualización del peso, como se muestra en:

$$\Delta x_{ij}^t = \eta(r_i^t - y_i^t)x_j^t \tag{2.28}$$

donde $r_i^t$ es el valor de salida obtenido, $y_i^t$ es el valor esperado de salida y $x_j^t$ el valor de entrada, y $\eta$ es la magnitud de la actualización. De esta manera podemos observar que la actualización se da mediante valores positivos o negativos, dependiendo de los valores obtenidos y esperados en la salida. Además, se considera la magnitud de la actualización $\eta$, también conocida como tasa de aprendizaje (*learning rate*).

Para las formulaciones anteriores, se considera el caso en el cual las clases son linearmente separables, en donde se busca el hiperplano a separar con una función de recta como se muestra en la Ecuación 2.24. Sin embargo, en la mayoría de los problemas de regresión o clasificación de conjuntos de datos reales, el conjunto no es linearmente separable. Para solventar este problema, se introduce el modelo de **Perceptrón multi-capa** (*Multi-Layer Perceptron).*

El modelo del MLP, consiste en introducir "capas" de perceptrones intermedias, entre la entrada y salida de este modelo, también conocidas como *capas escondidas*. De esta manera, la capa escondida permite la implementación de discriminantes no lineares para la clasificación. Con base en los perceptrones de entrada $x$, sus salidas son propagadas hacia delante, hacia la capa escondida, donde la *función de activación* es calculada para cada uno de los perceptrones en la capa escondida $z_h$. Para calcular la función de salida de cada uno de los perceptrones de la capa escondida se aplica una función no linear, que sea diferenciable para utilizar un método de gradiente para actualizar sus pesos. Es usual que se utilice una función sigmoide como se muestra en la Ecuación 2.29, donde $d$ representa la dimensionalidad de la capa de entrada y $H$ la dimensionalidad de la capa escondida. Si bien, es usual que se utilice la función sigmoide como activación de los perceptrones, también existen otras funciones comunes como la *tangente hiperbólica (tanh)* o *Radial Basis Functtion (RBF).*

$$z_h = sigmoid(w_h^T x = \frac{1}{1 + e^{-(\sum_{j=1}^{d} w_{hj} x_j + w_{h0})}}, h = 1, \ldots, H \tag{2.29}$$

Ahora los valores de salida del perceptrón, obtenidos mediante la Ecuación 2.29, se convertirán en los pesos de entrada para la capa de salida. La salida de esta capa se representa

como:

$$y_i = v_i^T z = \sum_{h=1}^{H} v_{ih} z_h + v_{i0} \tag{2.30}$$

donde el valor $v_{i0}$ denota el término *bias* de la capa escondida.

Ahora usando el modelo MLP, la discriminación de $K > 2$ clases se formula de manera similar a la Ecuación 2.27, aplicando una función *softmax* a la salida. Se define $o_i^t$ como:

$$o_i^t = \sum_{h=1}^{H} v_{ih} z_h^t + v_{i0} \tag{2.31}$$

y se aplica *softmax* a la capa de salida de MLP de la siguiente forma:

$$y_i^t = \frac{e^{o_i^t}}{\sum_k e^{o_i^t}} \tag{2.32}$$

De igual manera, la actualización por gradiente de los pesos de la red neuronal es similar a la presentada en la Ecuación 2.28, pero ahora se integran la actualización de la capa escondida como:

$$\Delta v_h = \eta \sum_t (r^t - y^t) z_h^t \tag{2.33}$$

y la actualización de la primera capa como:

$$\Delta w_{hj} = \eta \sum_t \left[ \sum_i (r_i^t - y_i^t) v_{ih} \right] z_h^t (1 - z_h^t) x_j^t \tag{2.34}$$

Mediante este acercamiento, es posible que el perceptrón encuentre discriminantes de funciones linearmente no separables, utilizando la *propagación hacia atrás* del error para actualizar los pesos del MLP.

### 2.4.3. AdaBoost

El concepto de *boosting* consiste en generar modelos de aprendizaje "básicos" para realizar el entrenamiento de modelos posteriores utilizando los errores de los modelos anteriores. Esto con la finalidad de obtener un modelo robusto de aprendizaje, combinando modelos básicos.

El algoritmo original de **boost**[1] es presentado en [35]. Se considera que una clase o "concepto" es fuertemente capaz de aprender (*strongly learnable*) si existe un algoritmo que en tiempo polinomial pueda encontrar los conceptos de la clase con un bajo grado de error. En cambio, una clase débilmente capaz de aprender (*weak learnability*) es aquella que tiene una confidencia mayor al 50 % sobre los conceptos de la clase, es decir, es solo un poco mejor que la elección aleatoria. Bajo estos conceptos, se propone un algoritmo que produce la clasificación de un conjunto de datos de entrenamiento, produciendo un modelo de aprendizaje fuerte, a partir de tres modelos débiles.

El proceso consiste en tomar un conjunto de datos de entrenamiento $X$ y dividirlo aleatoriamente en tres subconjuntos $\{X_1, X_2, X_3\} \subseteq X$. Posteriormente, se realiza el proceso de entrenamiento con el modelo $d_1$ utilizando el conjunto de aprendizaje $X_1$ y $X_2$. Se toman

[1] Incrementar, aumentar[34]

los ejemplos de entrenamiento en el cual $d_1$ fue correcto en la clasificación, además de los clasificados incorrectamente de $X_2$, que en conjunto formarán el conjunto de entrenamiento para $d_2$. Posteriormente, se toma el conjunto $X_3$, que se utiliza como entrenamiento para $d_1$ y $d_2$, donde los casos en los cuales la salida de $d_1$ y $d_2$ difieran, serán utilizados como casos de entrenamiento para $d_3$. Finalmente, al considerar el proceso de verificación, el caso a clasificar se ingresa a $d_1$ y $d_2$, y en caso de que ambos difieran en la salida, se considera que la salida correcta está dada por $d_3$. De esta manera, se muestra que este algoritmo se enfoca en mejorar los clasificadores basándonos en el error de otros clasificadores débiles. En [35], se demuestra que el modelo fuerte tiene una tasa de error menor, que puede ser disminuida aplicando recursivamente el proceso.

Sin embargo, una de las desventajas de este modelo, es particionar un conjunto grande de datos de entrenamiento. Para solventar este problema se proponen el algoritmo *AdaBoost* [36] que toma conceptos del algoritmo de *boosting* original, pero es extendido para aceptar una cantidad arbitraria de modelos débiles.

Se proponen dos versiones del algoritmo **AdaBoost.M.1** y **AdaBoost.M.2**. Para AdaBoost.M.1 se considera un conjunto $m$ ejemplos $S = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$, donde $x_i$ representa una instancia del conjunto de entrenamiento $X$, y $y_i$ la clase a la cual pertenece $x_i$, con $y_i \in Y$. Además, se define un algoritmo **WeakLearn** que representa el algoritmo de aprendizaje débil que será utilizado. Posteriormente **WeakLearn**, toma una distribución $D_t$ sobre $S$ y calculará las hipótesis de este conjunto como $h_t : X \to Y$ teniendo un número no trivial de ejemplos mal clasificados. Así, el objetivo de **WeakLearn** es encontrar la hipótesis $h_t$ que minimiza el error $\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$. Se aplica este proceso durante $T$ iteraciones, donde finalmente se combinarán las hipótesis $h_1, \ldots, h_T$ en una hipótesis final $h_{fin}$.

La idea es que para tomar los ejemplos de $S$, se considere una distribución que tome en cuenta el error obtenido. Dado esto, se inicia $D_t$ para la primera iteración como $D_1(i) = 1/m \, \forall i$. Posteriormente, para distribuciones $D_{t+1}$ a partir de $D_t$ y la hipótesis anterior $D_t$, se multiplica el peso del ejemplo $i$ por un número $\beta_t \in [0, 1)$, si la hipótesis clasifica correctamente $x_i$, de lo contrario no se modifica el peso. El valor de $\beta_t$ se calcula en función del error como: $\beta_t = \dfrac{\varepsilon_t}{1 - \varepsilon_t}$. Además los pesos se re-normalizan utilizando una constante de normalización $Z_t$. De esta manera, los ejemplos mal clasificados, son los que contarán con mayor peso. Finalmente, la hipótesis final $h_{fin}$ está dada por la votación según la suma de pesos de cada hipótesis débil. Es importante considerar que los clasificadores débiles para esta versión, deben tener un error ligeramente menor a 1/2, de lo contrario la hipótesis final tiende a decrementar rápidamente a cero.

La versión AdaBoost.M.2 pretende superar la limitación anterior, permitiendo que la hipótesis de **WeakLearn** sea mas descriptiva, aceptando un vector con valores $[0, 1]^k$, donde este valor se interpreta un "grado de confianza", donde los valores cercanos a 0 o 1, corresponden a dichas etiquetas. En este caso se integra una distribución sobre los errores clasificados incorrectamente $(i, y)$, donde $i$ representa el índice del ejemplo, y $y$ la etiqueta incorrecta. La distribución de ejemplos mal clasificados se define como $B = \{(i, y) : i \in \{1, \ldots, m\}, y \neq y_i\}$. En cada iteración, **WeakLearn** toma la distribución de los ejemplos mal clasificados $D_t$ y produce una hipótesis de la forma $h_t : X \times Y \to [0, 1]$. Además se redefine el proceso para

el cálculo del error mediante la "pseudo-pérdida" como se muestra en:

$$\varepsilon_t = \frac{1}{2} \sum_{(i,y)\in B} D_t(i,y)(1 + h_t((x_i, y) - (x_i, y_i)) \tag{2.35}$$

De esta manera, el error disminuye cuando los valores cercanos a la clasificación correcta $y_i$ son 1, mientras que valores incorrectos son cercanos a 0. El teorema presentado en [36] indica que en este caso los modelos de aprendizaje débiles, solamente deben tener un "pseudo-error" menor a 1/2, sin importar el número de clases.

La formulación anterior constituye la base para el algoritmo AdaBoost, propuesto también para clasificación multi-clase. Sin embargo, su efectividad en este tipo de clasificación disminuye. Una versión multi-clase para el algoritmo de AdaBoost, llamado SAMME [37] es propuesta. Esto se lleva a cabo siguiendo la estructura clásica de AdaBoost y realizando un cambio en el cálculo de la distribución.

## 2.5. Bases de datos clínicas

Hoy en día existe una alta demanda por el análisis de información clínica dado que existe gran cantidad de información generada a través de los sistemas de información automática para generación de expedientes clínicos electrónicos o de dispositivos IoT para el monitoreo de pacientes. El objetivo de explotar la información generada es que sirva de apoyo para a la labor diaria del personal médico, así como a la mejora de la calidad de vida en los pacientes [38].

Existen esfuerzos por generar bases de datos clínicas en distintos ámbitos médicos para contribuir a la explotación de estos datos en distintas áreas médicas, como es la genómica [39, 40], la detección oportuna del cáncer [41, 42] o de la información generada por un área específica del hospital [43], entre otras.

La base de datos MIMIC-II [43] contiene información clínica recopilada de una variedad de pacientes admitidos en cuidados intensivos, en el hospital *Beth Israel Deaconess Medical Center* en Boston. Esta base de datos cuenta con dos componentes importantes, datos generados como señales de monitoreo de los sistemas de cuidado intensivo e información clínica del paciente durante el tiempo que estuvo en el hospital. La estructura general de la base de datos se presenta en la Figura 2.10 [44].

En esta figura se muestra la información principal de la cual se compone la base de datos MIMIC-II. El primer módulo denominado ICU, es la información de monitoreo de pacientes en unidades de cuidados intensivos, con un énfasis en la información de formas de onda obtenidas a través de sensores.

El segundo componente de la base de datos es la información general de los pacientes ingresados y procesos administrativos del hospital. Como información útil para la clasificación de pacientes se tienen los módulos de análisis de laboratorio, información demográfica, notas y reportes. Estos últimos siendo notas de texto libre sobre la evolución del paciente a través de su tratamiento, así como el diagnóstico e interpretación de estudios dados por profesionales de la salud.

En ambos casos la información de los pacientes pasó por un proceso para volver anónima la información, procedimiento común en la base de datos clínicas, para proteger la información personal de los pacientes.

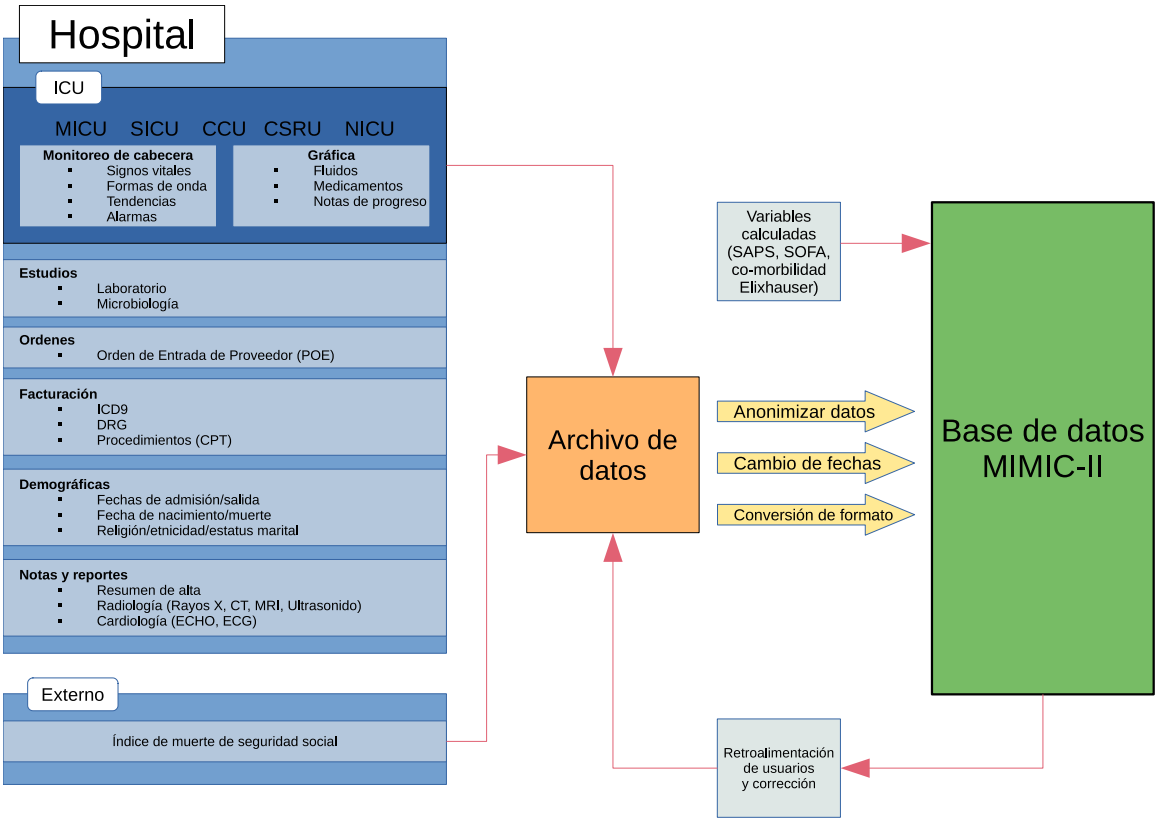



Figura 2.10: Estructura general de la base de datos MIMIC-II, tomada de [44]

Existen distintas bases de datos con información referente a pacientes con cáncer, con distintos enfoques. En la Sección 3 se presentan distintos trabajos donde utilizan datos estructurados y no estructurados para la clasificación de pacientes con cáncer.

# Capítulo 3

# Estado del arte

La clasificación automática de enfermedades, utilizando algoritmos de *Machine Learning (ML)*, se ha abordado en extenso en la literatura. Existen trabajos no solamente con casos de cáncer, sino también con otro tipo de enfermedades crónicas como la diabetes o la hipertensión que se benefician del análisis computacional para encontrar patrones dentro de los datos clínicos. Las estrategias de clasificación se pueden llevar a cabo según el tipo de dato (estructurado o no) que se encuentra en los expedientes clínicos. Por esto, a continuación se presentan diversos trabajos de investigación que han abordado la clasificación de enfermedades, ya sea utilizando datos estructurados o no estructurados.

## 3.1. Datos Estructurados

Los algoritmos de *Machine Learning (ML)* para la clasificación automática de información, orientados al cáncer se han aplicado con distintos fines como lo son la predicción de la supervivencia, la detección temprana y diagnóstico de cáncer, o la clasificación de tipos de cáncer. En esta sección se presentan algunos trabajos relacionados con dichas actividades, en particular para el cáncer de mama, de pulmón y de hígado, que son los tipos de cáncer que constituyen el objeto de estudio de esta tesis.

La predicción de supervivencia de cáncer de mama es presentado en [45]. En este trabajo se buscó el desarrollo de modelos predictivos para encontrar las relaciones entre las variables independientes y la supervivencia a este tipo de cáncer. Los datos utilizados fueron 72 variables socio-demográficas así como información específica al cáncer de mama. Esta información fue pre-procesada y la experimentación final se realizó con base en 16 variables independientes y una variable dependiente que corresponde a la supervivencia del cáncer. Esta información se utilizó para entrenar tres modelos predictivos: MLP, *Decision Trees (DT)* y *Logistic Regression (LR)*. Se encontró que el mejor modelo para realizar esta tarea fueron los árboles de decisión con una precisión de 93.62 %, siguiéndole el MLP con 91.21 %, y finalmente LR con 89.20 %.

El trabajo presentado en [46] aborda también la predicción de la supervivencia en el cáncer de mama. Este trabajo utilizó siete modelos de ML para predecir la supervivencia de un paciente, basándose en datos estructurados. Los datos que fueron utilizados en este estudio corresponden a registros del *Cancer Registry Organization of Kerman Province* en Irán, y corresponden a la demografía del paciente, la morfología y los estudios realizados. Los modelos utilizados para llevar a cabo la clasificación fueron *Naive Bayes (NB), Trees Ran-*

*dom Forest (TRF), 1-Nearest Neighbor (1-NN), AdaBoost (AD), Support Vector Machines (SVM), RBF Network (RBFN) y MLP*. Se obtuvo como resultado que los mejores modelos para clasificación fueron TRF y MLP con una precisión del 96 % y 95 %, respectivamente.

El trabajo presentado en [47] consistió en comparar algoritmos de ML para la predicción de supervivencia al cáncer de mama. Además, en este estudio se toma en cuenta la predicción de metástasis en el paciente. Considerando esto como un signo importante de la progresión, y eventualmente el resultado para estudios de cáncer. Los autores proponen la comparación de algoritmos de ML bajo las métricas de sensibilidad, especificidad, valor predictivo positivo, valor predictivo negativo, radio de probabilidad positiva y negativa y la precisión total. Los algoritmos que consideraron para llevar a cabo la comparación fueron NB, LR, *Linear Discriminant Analysis (LDA)*, RF, SVM, *Least Square Support Vector Machine (LS-SVM)*, AD y Adabag. El conjunto de datos consistió en 83.4 % de pacientes vivos y 16.6 % de pacientes muertos. Además, del conjunto total de datos (pacientes vivos y muertos) el 85 % no presentó metástasis. Las clases consideradas para cada una de las predicciones fueron: vivo o muerto, para la supervivencia; y experiencia o no, para la predicción de metástasis. Además cada uno de los modelos de ML fue entrenado con dos configuraciones distintas de entrenamiento y prueba, la primera utilizando un 70 % de entrenamiento y 30 % para prueba, la segunda utilizando 50 % tanto para entrenamiento como para prueba. El estudio encontró que para la predicción de supervivencia los modelos con el mejor desempeño fueron SVM y LDA con una precisión del 93 %. En el caso de la predicción de metástasis se considera el mejor modelo fue LDA con una precisión del 86 %, sin embargo se hace énfasis que la sensibilidad para esta clasificación bajó en todos los modelos, que en el caso de LDA fue de 0.27.

La comparación de algoritmos de ML orientados a la detección temprana y diagnóstico de cáncer de mama se presenta en [48]. Los autores proponen el uso de técnicas de minería de datos utilizando ML, con el propósito de desarrollar modelos predictivos y disminuir el esfuerzo humano y errores técnicos al diagnosticar cáncer de mama. La metodología propuesta fue la inclusión de una base de conocimientos (*Knowdlege Base (KB))* donde se guarda la información de entrenamiento y prueba de los algoritmos de ML. A partir de esta información se entrenan tres algoritmos de ML propuestos (DT, RF y SVM) para construir el modelo de clasificación, y es utilizado con datos de prueba para, finalmente, ser incluidos en la KB. Los datos utilizados por los autores provienen de la base de datos SEER, a los cuales se les aplicó pre-procesamiento y el algoritmo *Expectation-Maximization (EM)* para lidiar con los datos faltantes. El conjunto de datos final que se utilizó consideró 16 variables estructuradas. Como resultado se obtuvo que el algoritmo de RF fue el que mejor desempeño tuvo en la tarea de clasificación, obteniendo una precisión del 73 %.

También la comparación de algoritmos de ML para la detección de cáncer de mama es presentada en [49]. El objetivo del trabajo fue la comparación de algoritmos de ML para la clasificación del cáncer de mama con base en si el tumor fuese benigno o maligno. Los datos considerados fueron obtenidos del repositorio *UCI ML*, considerando un conjunto de datos con 32 atributos sin valores faltantes y un atributo de salida que considera si el tumor es maligno o benigno. Los modelos de ML que consideró el estudio fueron RF, k-NN, y NB, que fueron entrenados con la técnica de validación cruzada con 10 pliegues, considerando cada uno de ellos con porciones equitativas del conjunto de datos, y dejando uno de los pliegues como prueba. El estudio encontró que el algoritmo de k-NN fue el que mejor desempeño mostró, obteniendo una precisión del 95 %.

El diagnóstico de cáncer de mama también es abordado en [50]. En este estudio se tiene como objetivo la integración de algoritmos genéticos para la selección de características para aumentar el desempeño de la clasificación, además de integrar los costos de la clasificación errónea (falsos positivos) en un modelo inteligente de clasificación. El proceso propuesto por los autores considera la ganancia de información de un caso del conjunto de datos, posteriormente se calcula el *fitness* de este caso considerando una función propuesta por los autores que integra pesos asignados a clasificaciones erróneas. Posteriormente al proceso de un algoritmo evolutivo, los autores integran el algoritmo *Cost-Sensitive Support Vector Machine (CSSVM)* como método de cálculo del *fitness* de los individuos, para posteriormente continuar con el flujo de un algoritmo evolutivo, integrando validaciones de recocido simulado. Una vez que el proceso del algoritmo evolutivo concluye, se integran los individuos (conjuntos de características) al conjunto de entrenamiento y prueba al algoritmo CSSVM. Los autores validaron su propuesta utilizando el conjunto de datos de *Wisconsin Breast Cancer* y *Wisconsin Breast Cancer Diagnosis*. El estudio demostró que el acercamiento propuesto por los autores mejoró la clasificación en comparación con los clasificadores antes de aplicar el método de selección de características. Se observó un aumento en el desempeño de la clasificación, con un 95.7 % de precisión al implementar la metodología propuesta, además de una disminución en la complejidad del cálculo.

La supervivencia del cáncer de pulmón es presentada en [51]. Este trabajo tiene como objetivo la predicción del tiempo de supervivencia de un paciente diagnosticado con cáncer de pulmón, utilizando cuatro modelos de ML y un modelo personalizado. La información obtenida fueron datos estructurados de demografía, estudios de laboratorio e indicadores específicos, obtenidos de la base de datos SEER. Los modelos de ML utilizados para llevar a cabo la clasificación fueron LR, DT, *Gradient Boosting Machines (GBM)*, SVM y un método personalizado que utilizaba los modelos anteriores en conjunto para dar un veredicto en la clasificación basándose en pesos para cada modelo. Los resultados se obtuvieron comparando los valores *Root Mean Square Error (RMSE)* para cada uno de los modelos. El trabajo obtuvo que el modelo con el menor RMSE fue el método personalizado (15.30), teniendo al modelo con mayor peso en la clasificación al GBM, que de manera individual obtuvo un RMSE de (15.32).

La comparación del desempeño de algoritmos de ML para el apoyo en el diagnóstico temprano del cáncer de pulmón es presentado en [52]. En este estudio los autores compararon seis algoritmos de ML: NB, RF, SVM, MLP, *Gradient Boosted Trees (GBT)* y *Neural Networks (NN)*. Además de un sistema de voto considerando los modelos MLP, GBT y SVM. Para cada uno de los modelos de ML se llevó a cabo un pre-procesamiento de datos, además de un proceso de validación cruzada de 10 iteraciones. Los datos utilizados para este estudio fueron obtenidos del repositorio UCI para cáncer de pulmón, que cuenta con 56 atributos de predicción y un atributo de clase. El estudio encontró que el sistema de voto fue el que presentó el mejor desempeño con una precisión del 88.57 %.

La predicción de cáncer de pulmón no microcítico es presentado en [53]. Los autores comparan el desempeño de algoritmos de ML para la predicción de este tipo de cáncer, utilizando información sobre marcadores genéticos de los pacientes para conocer si es posible que un modelo de ML tenga una precisión aceptable para la predicción de este tipo de cáncer. Los algoritmos considerados para el estudio son SVM, K-NN, RF, NN y DT. El conjunto de datos utilizados consistió en 14 atributos sobre mutaciones genéticas, así como 203 observaciones. Finalmente el estudio mostró como resultado que el modelo NN mostró

el mejor desempeño, obteniendo una precisión del 93.4 %.

El estudio sobre el desempeño de algoritmos de ML para la predicción del resultado de tratamiento por radioterapia para cáncer de pulmón no microcítico, cáncer de cabeza y cuello y meningioma es presentado en [54]. El objetivo de este estudio es la comparación del desempeño de algoritmos de ML, en un conjunto de datos sobre los resultados de tratamiento de pacientes con los tipos de cáncer antes mencionados. Los autores pretenden descubrir si existe un clasificador superior para el modelado predictivo de los resultados de tratamiento por radioterapia, la dependencia del conjunto de datos para la selección del clasificador y los beneficios de seleccionar un clasificador basándose en conjuntos de datos similares. El proceso seguido por los autores fue el pre-procesamiento de los datos, aplicando imputación de características, escalamiento, eliminación de características con varianza cero, aplicado a cinco subconjuntos de *cross-validation* "exteriores". Posteriormente, cada subconjunto es tomado por el algoritmo de ML para ser entrenado, en donde, durante el proceso, se crean otros cinco subconjuntos "interiores" para la optimización de los hiper parámetros. Finalmente, el modelo es probado con los datos de prueba del subconjunto "exterior" y se obtienen sus métricas para ser analizadas. Los datos utilizados para el estudio fueron 12 conjuntos de resultados de tratamiento obtenidos por los autores. Los algoritmos de ML considerados fueron *Elastic Net Logistic Regression(EL-LR)*, RF, NN, SVM, *LogitBoost* y DT. El estudio encontró que los algoritmos con mejor desempeño fueron RF y ELLR; sin embargo, se menciona que la selección de clasificadores basándose en las características del conjunto de datos, muestra beneficios en la capacidad de discriminación del modelo.

La predicción sobre tiempos de supervivencia al cáncer de pulmón es presentado en [55]. Los autores proponen un modelo de clasificación de tiempo de supervivencia, además de un modelo de regresión aplicada a cada uno de los conjuntos clasificados para obtener el tiempo de supervivencia del paciente. El modelo propuesto se basó en la clasificación de pacientes en tres categorías, correspondientes a los meses de supervivencia, siendo estos: menor o igual a 6 meses, de 7 a 24 meses, y mayores a 24 meses. El modelo de clasificación propuesto fue RF, siendo entrenado con el 75 % de los datos. Posteriormente, tres modelos de regresión fueron entrenados con las categorías anteriormente mencionadas. Los datos de entrenamiento para cada conjunto traslapaban ligeramente con los demás, para permitir que el modelo pudiese predecir tiempos ligeramente fuera de su conjunto. Los modelos de regresión utilizados fueron *General Linear Regression (GL)*, GBM, RF, y un sistema que combina los modelos anteriores con un peso distinto, determinado por una regresión linear del tiempo predicho y del tiempo real de supervivencia. Finalmente, RMSE es utilizado para calcular la precisión del modelo obtenido. El estudio encontró que la predicción se desempeñaba mejor en el rango de tiempo de 3-7 meses, mientras que sufría una caída en desempeño para tiempos de supervivencia mayores.

La predicción de supervivencia al cáncer de pulmón aplicando técnicas de ML es presentado en [56]. En su trabajo, los autores buscan la predicción de la supervivencia a este tipo de cáncer en etapas tempranas, con el objetivo apoyar en el mejor tratamiento para el paciente. El modelo propuesto por los autores se basa en la adquisición de datos de pacientes con este tipo de cáncer, que consiste en datos seleccionados por patólogos y doctores en distintos hospitales, además de unir esta información con los datos *NCCTG Lung Cancer Data*. Posteriormente, se pasa a una etapa de selección de características, donde se determina el uso de 14 características para el entrenamiento predicción de los algoritmos, con una de ellas indicando si un paciente sobrevivirá por mas de un año o no. Después, los datos de

las características seleccionadas son utilizados para el entrenamiento de los algoritmos de ML, siendo estos SVM, C4.5 y NB. Una vez entrenados los modelos de ML, el desempeño de estos es comparado con base en la precisión para obtener el mejor modelo de clasificación para esta tarea. El estudio encontró que el algoritmo C.45 mostró el mejor desempeño en la clasificación, incrementando el tamaño del conjunto de datos, obteniendo como su mejor valor de precisión 82.6 %.

La comparación de algoritmos de ML para la predicción de carcinoma hepatocelular es presentado en [57]. Los autores comparan diez algoritmos de ML para la predicción de este tipo de carcinoma, proponiendo un pre-procesado de datos, además de la inclusión de un optimizador de parámetros y características, basado en algoritmos genéticos. El pre-procesamiento de los datos es presentado con un proceso de tres pasos: uso del valor de la moda para llenar los valores categóricos faltantes, uso del promedio para llenar los valores numéricos faltantes, y la normalización de los valores anteriores mediante un escalamiento entre un rango entre 0 y 1. Posteriormente, cada uno de los algoritmos de ML pasan por un algoritmo genético para la selección de parámetros óptimos y características de los datos, para llevar a cabo el entrenamiento y validación de cada algoritmo de ML. Los algoritmos de ML considerados por el estudio fueron SVM linear, SVM, MLP, KNN, NB, LDA, *Quadratic Discriminant Analysis (QDA)*, LR y RF. El conjunto de datos utilizado incluyó la información de 165 pacientes diagnosticados con carcinoma hepatocelular, en el hospital *Coimbra's Hospital and Universitary Center (CHUC)* en Portugal. Estos datos contenían 25 características cuantitativas y 26 características cualitativas sobre aspectos relevantes de la enfermedad. Como resultados de sus experimentos se obtuvo que el modelo con el mejor desempeño fue SVM, obteniendo una precisión del 88.49 %.

El desarrollo de un modelo para la predicción de carcinoma hepatocelular es presentado en [58]. Los autores diseñaron un *framework* para la selección óptima de un modelo basándose en los datos que se aplicarán, así como la aplicación de este *framework* a datos actuales de carcinoma hepatocelular para la selección de un modelo óptimo. Dicho *framework* consistió en una interfaz gráfica para la entrada de conjuntos de datos en formato csv. Posteriormente, aplicaron una búsqueda de parámetros *grid search* para cada uno de los algoritmos de ML considerados. Finalmente, obtuvieron el modelo de ML junto con sus hiper parámetros óptimos del modelo que mejor desempeño tuvo en la clasificación del conjunto de datos de entrada. Los algoritmos de ML considerados para este estudio fueron LR, LR con penalizado L1, L2 y elástico, SVM, GB, RF, NN y DL. El sistema fue probado con una base de datos de pacientes que visitaron la clínica de hígado en *University of Tokyo Hospital* entre 1997 y mayo del 2016, considerando un total de 4242 pacientes. Para el conjunto de datos utilizados en el estudio, se encontró que el modelo GB obtuvo el mejor desempeño, obteniendo una precisión del 87.34 %.

En el trabajo presentado en [59] se explora el desempeño de algoritmos de ML, aplicando optimización basada en enjambre de partículas (PSO) en la predicción de una enfermedad de hígado. Los autores proponen un modelo basado en tres fases para la clasificación de enfermedades de hígado (hígado graso, hepatitis, cirrosis y cáncer). La primer etapa consiste en la normalización entre mínimo y máximo de las características del conjunto de datos. La segunda etapa consiste en la selección de características de entrenamiento para los algoritmos de ML, basada en PSO. En la tercer etapa, se entrenan los algoritmos de ML con las características obtenidas de la etapa anterior. Los algoritmos utilizados por los autores son RF, SVM, J-48, MLP y NB. Finalmente, para obtener el cálculo de la precisión de los

algoritmos se toma en consideración el cálculo de *Root Mean Error (RME)* y RMSE. El estudio encontró al modelo J-48 como el que mostró el mejor desempeño con un 95.04 % al aplicar PSO en la selección de características.

La predicción de enfermedades de hígado mediante modelos de ML también es abordado en [60]. Los autores compararon el desempeño de algoritmos de ML en la predicción de enfermedades del hígado, con el objetivo de disminuir el costo para su diagnóstico. Se propone la comparación de seis algoritmos de ML, siendo estos LR, KNN, DT, SVM, NB y RF. El proceso consideró un análisis de las características de la clasificación, en donde se identificó correlación entre algunas de ellas, por lo cual fueron omitidas para mejorar la predicción. El conjunto de datos utilizado consistió en información de 583 pacientes con alguna enfermedad hepática, dicho conjunto contenía 11 parámetros, de los cuales 10 fueron considerados como características de análisis y uno como la enfermedad a clasificar. Los autores encontraron que el modelo que mejor llevó a cabo la clasificación de la enfermedad fue LR, obteniendo un 75 % de precisión.

La comparación de algoritmos de ML para la predicción de la mortalidad de pacientes con carcinoma hepatocelular es presentada en [61]. El objetivo en este trabajo fue la comparación de modelos de ML con la finalidad de encontrar el modelo óptimo para el pronóstico de la mortalidad de pacientes con carcinoma hepatocelular. El proceso consistió en un pre-procesamiento de los datos, en su mayoría numéricos, aplicando un método de imputación para lidiar con valores faltantes, así como un análisis de correlación de los datos para eliminar características redundantes en los datos. Los algoritmos de ML considerados para el estudio fueron RF, *Bagging*, *AdaBoost*, LR y DT. El conjunto de datos utilizado fue un conjunto de 165 pacientes diagnosticados con carcinoma hepatocelular, obtenidos de una base de datos del Hospital Universitario en Portugal. Dicho conjunto contiene 49 características numéricas y categóricas. El estudio encontró que el error de los algoritmos comparados se mantiene similar entre sí; aunque el modelo que obtuvo mejor desempeño en precisión fue RF, obteniendo un 74 % de precisión.

La predicción de la supervivencia al cáncer de hígado, combinando datos clínicos con marcadores genéticos es presentada en [62]. En el estudio los autores presentan la clasificación de pacientes que presentan cáncer de hígado, con base en el tiempo de supervivencia que han tenido. El proceso considerado se divide en cuatro etapas. La primera consiste en el pre-procesamiento de los datos, discretizando las características. La segunda etapa consistió en la selección de las características que se utilizarían como entradas para los algoritmos de ML, esta selección se realizó con base en la evaluación de correlación, de ganancia de información y de un método de *Wrapper* para subconjuntos. La tercera etapa consistió en el entrenamiento de los algoritmos de ML para la clasificación de pacientes, siendo estos algoritmos: NN, SVM, DT, *RepTree*, RF, *JRip* y *PART*. Finalmente, los algoritmos fueron evaluados para obtener la precisión de cada uno y determinar el mejor para la tarea de clasificación. El estudio encontró que el modelo que presentó mejor precisión fue DT, con las características discretizadas y sin aplicar método de selección de características, obteniendo un 87.3459 % de precisión.

## 3.2. Datos no estructurados

La clasificación automática de información utilizando datos no estructurados en los modelos de ML se lleva a cabo con técnicas de Procesamiento de Lenguaje Natural (PLN).

Estos procesos son aplicados usualmente a reportes médicos que se encuentran en el expediente clínico de pacientes. En esta sección se presentan trabajos que han servido de apoyo a la toma de decisiones para el diagnóstico de cáncer de mama, de pulmón y de hígado.

En el trabajo [63], se propone un sistema de apoyo a la toma de decisiones analizando reportes de mamografía. Este apoyo se da en la forma de una predicción sobre la probabilidad de que un tumor sea maligno, así como una clasificación en base al sistema *Breast Imaging Reporting and Data System (BI-RADS)*. El proceso consiste en el uso de técnicas de PLN para extracción de la terminología basada en BI-RADS que contienen "descriptores". Estos descriptores son utilizados como valores estructurados de entrada para un modelo de redes bayesianas para llevar a cabo la clasificación. Se llevó a cabo la comparación de salidas obtenidas de la red bayesiana teniendo como entrada el conjunto de datos estructurados indicados por un médico, con los descriptores obtenidos a través del flujo de PLN. La correlación de las probabilidades obtenidas por ambos métodos fue del 95 % y la precisión obtenida del modelo cuyas entradas provinieron del proceso de PLN fue del 97.58 %.

En el estudio presentado en [64] se busca una correlación entre los hallazgos descritos en reportes de mamografía y patología con el subtipo de cáncer de mama. El proceso fue realizado con una implementación de técnicas PLN para minería de información clínica, llamado MOTTE. Este proceso consideró las fases de segmentación, *stemming*, eliminación de palabras vacías, modelado en espacio de vectores y cálculo de similitud. El proceso MOTTE fue aplicado a la base de datos METEOR para buscar pacientes con cáncer invasivo, utilizando una serie de palabras claves que describen este padecimiento. La población total de cáncer invasivo fue de 543 pacientes, de los cuales nuevamente se aplicó MOTTE para obtener información demográfica, clínica y de mamografía, para encontrar correlación de estas variables con el subtipo de cáncer. Los resultados obtenidos fueron validados con un 10 % de notas que fueron revisadas manualmente para indicar el subtipo de cáncer que presentaba el paciente. El estudio encontró que con base en los marcadores encontrados por MOTTE se obtuvo una correlación de estos marcadores con el subtipo de cáncer y demostraron que aplicando técnicas de PLN se pueden distinguir entre subtipos de cáncer.

La extracción de menciones sobre la etapa del cáncer usando técnicas de PLN es presentado en [65]. En este trabajo, los autores extrajeron información acerca del TNM (Tumor, Nodo, Metástasis) de tres tipos de cáncer: colon, pulmón y próstata. La importancia de la extracción de información TNM se da, ya que este tipo de información es utilizada como referencia para iniciar tratamientos de cualquier tipo de cáncer. Para este estudio se utilizaron dos tipos de registros que contienen información no estructurada y que fueron manualmente anotados para menciones de TNM. El primero fue el registro de pacientes del NAACCR (*North American Association of Central Cancer Registries*) que contiene campos con texto no estructurado describiendo la historia clínica, resultados de exámenes clínicos, descripciones de estudios de imagenología e información potencial para la etapa del cáncer. El segundo registro fue E-path, que consiste mayoritariamente en campos de texto no estructurado acerca de la patología del tumor, histología y diagnósticos finales. Se consideró un conjunto aleatorio de 100 casos de cáncer para cada uno de los tipos mencionados anteriormente. Se dividió el conjunto de datos en entrenamiento (50 %), desarrollo (17 %) y prueba (33 %). A dichos conjuntos se aplicaron dos técnicas para la extracción de entidades a partir de un flujo PLN, basado en reglas (REGEX) y *Conditional Random Fields (CRF)* que es basado en ML. Esta extracción se realizó para obtener menciones TNM clasificándolas por la etapa y temporalidad (clínica o patológica). El estudio encontró que ambos modelos fueron

aceptables al detectar menciones de TNM, con un desempeño similar entre ellos (94.0 % - 97.1 %).

La extracción de síntomas de cáncer de mama a partir de notas en texto libre es presentado en [66]. En este trabajo, los autores extraen información de los síntomas que presentan los pacientes con este tipo de cáncer, ya sea por la enfermedad misma, por el tratamiento que lleva a cabo, o por una combinación de ambas causas. Esto debido a que muy pocos estudios incluyen la adición de síntomas a las variables a considerar, ya que la revisión manual de cada una de las notas generadas en busca de los síntomas puede considerarse una tarea demasiado costosa e ineficiente. El modelo propuesto por los autores consideró un proceso de PLN para las notas obtenidas de EHRs. Dicho proceso consistió en la segmentación de las notas, etiquetado individual de palabras en tres categorías (positivo, neutral y negativo), donde positivo indica la presencia de un síntoma, neutral considerado como una palabra no relacionada y negativo como la ausencia de un síntoma. Un médico llevó a cabo el etiquetado manual de las notas de 39 pacientes elegidos aleatoriamente, para ser tomadas como estándar de comparación para el algoritmo de ML. Posteriormente, las notas son consideradas como entradas para el algoritmo CRF para obtener el resultado a nivel palabra, para evaluar si representa un síntoma. Los datos utilizados para este estudio fueron obtenidos de los registros en el repositorio *Research Patient Data Registry (RPDR)*, donde se filtraron los pacientes que recibieron al menos una dosis de quimioterapia con paclitaxel. Como resultado se obtuvo que el algoritmo CRF identificó 56 % de síntomas positivos y 69 % de síntomas negativos, con una precisión del 82 % y 86 %, respectivamente, comparados con los síntomas etiquetados manualmente.

El análisis de reportes radiológicos en forma de texto libre es presentado en [67]. En este trabajo los autores llevan a cabo la clasificación de reportes radiológicos para obtener reportes de clasificación en "tiempo real". Esto con el objetivo de disminuir el trabajo requerido por la validación manual, y en su caso, reducir la carga de trabajo requerida por el médico. El modelo se basa en un esquema de clasificación propuesto por radiólogos del hospital *Spedali Civili di Brescia*. El esquema consiste en cinco clases de alto nivel que pueden asumir dos o mas valores: tipo de examinación, resultado de la examinación, naturaleza de la lesión, sitio de la lesión y tipo de la lesión. Los autores proponen tres modelos para el análisis de esta información. El primer modelo combina técnicas de extracción de información y técnicas de clasificación de textos. Este comienza con un pre-procesado con técnicas de PLN a nivel sintáctico y semántico; después, utiliza anotación automática de frases para incluir frases que presentan evidencia de pertenecer a una clase en específico. Posteriormente, cada oración es asociada con un nivel en caso de que contenga evidencia de pertenencia, utiliza las frases anotadas automáticamente y además se realiza una clase específica para los valores de cada nivel, utilizando el modelo BOW. Finalmente, el reporte de clasificación se da al unir las oraciones clasificadas de acuerdo a reglas heurísticas propuestas por los autores. El segundo modelo es similar al primero, solo cambia el proceso mediante el cual se clasifican las oraciones en clases. El texto se anota usando una etiqueta de clase específica para cada nivel, sin utilizar una segunda clasificación. El tercer modelo está basado en BOW, donde el reporte completo es utilizado para obtener la clasificación. Este último modelo no utiliza la anotación manual ni automática. Para los métodos que requirieron el uso del modelo BOW, se utilizaron cuatro modelos de ML para llevar a cabo la clasificación: NB, SVM, RF y MLP. Los datos utilizados por los autores fueron obtenidos de tomografías computarizadas del hospital *Spedali Civil di Brescia)*, dividiendo el 80 % de los reportes como datos de en-

trenamiento y un 20 % de prueba. Para el primer método se obtuvo que RF tiene el mejor desempeño para los niveles más altos de la clasificación, obteniendo un 95.6 % de precisión en la clasificación de "tipo de examen". Finalmente, los autores también reportan que el tercer método tiene un desempeño mayor en los niveles más altos de clasificación utilizando RF (98.3 % en el nivel más alto); sin embargo, para niveles mas específicos, su desempeño disminuye considerablemente. En contraste, el desempeño de los métodos que utilizan una clasificación a nivel de oración obtienen mejores resultados, considerando como evidencia que a niveles más específicos es necesario llevar a cabo el análisis a nivel oración.

La extracción de información a partir de certificados de muerte por cáncer es presentado en [68]. En este trabajo, los autores extraen información de los certificados de muerte para obtener estadísticas de mortalidad por cáncer. Esto con el objetivo de tener información de mortalidad para monitoreo, planificación y evaluación de la administración del cáncer, asunto de gran importancia para la salud pública. El modelo propuesto por los autores para llevar a cabo la extracción de información considera cuatro etapas. La primera considera la extracción de características basándose en dos categorías: características basadas en términos y en conceptos que son derivados de términos que pertenecen a las terminologías médicas *Systematized Nomenclature of Medicine – Clinical Terms (SNOMED-CT)* y *International Classification of Diseases for Oncology (ICD-O)*. Una vez que se lleva a cabo la extracción de características, se procede a generar un vector de características para los términos. Una primera clasificación puede darse con un proceso basado en reglas. En el caso de los conceptos SNOMED-CT, se llevó a cabo una conversión a ICD-O en un proceso basado en reglas y una transformación final a un código del *International Classification of Diseases (ICD)* versión 10. Los autores utilizan SVM para obtener una clasificación sobre la causa de muerte, utilizando como datos de entrada los vectores de características obtenidos previamente. Para este proceso se consideran múltiples modelos de SVM, uno para cada tipo de cáncer a considerar. Posteriormente, se propone un método de fusión para la clasificación obtenida mediante SVM y el proceso basado en reglas, considerando las puntuaciones normalizadas de cada uno de los clasificadores, además de una estrategia para seleccionar un único código para el certificado. Los datos utilizados para el estudio fueron obtenidos de certificados de muerte provistos por *Cancer Institute NSW*, donde cada certificado contenía un único código ICD-10 para indicar la causa de muerte. Los autores encontraron que la fusión tuvo un incremento del 5 %, en la clasificación de cáncer común y del 1 % para cáncer raro, ambos bajo la medida F.

El procesamiento de notas de evolución extraídas de EHR para identificar ocurrencias locales de cáncer de mama es presentado en [69]. En su estudio, los autores proponen una metodología para obtener información de notas de progreso y patológicas de los pacientes, con la finalidad de predecir la probabilidad de que el paciente presente una lesión local de cáncer de mama. La metodología propone el anotado manual de oraciones, anotadas como positivas, para la presencia de una lesión local de cáncer de mama, anotada con el uso de *MetaMap*. Posteriormente, se realiza un pre-procesamiento del conjunto de notas, para eliminar duplicados, segmentación y eliminación de símbolos no alfanuméricos. Luego de este pre-procesado, el conjunto de notas son anotadas utilizando MetaMap, para obtener identificadores únicos de cada concepto, del inglés *Concept Unique Identifier (CUIs)*. Una vez obtenidos los CUIs, se generan conjuntos de CUIs con la combinación de los conceptos individuales, que ayudan a inferir la recurrencia local de cáncer de mama. Además, utilizan información como el número de reportes o la presencia de biopsias como características

adicionales a los CUIs y a los conjuntos de CUIs para ayudar a la inferencia. Finalmente, las características filtradas, junto con el número de reportes patológicos son ingresados a un modelo SVM para indicar la posibilidad de presentar lesiones locales. El modelo propuesto por los autores es comparado con tres clasificadores entrenados en tres conjuntos distintos: un conjunto completo de los conceptos anotados con MetaMap, solo las características filtradas sin el número de reportes patológicos y un modelo de BoW. La comparación se basó en valores de precisión, *recall*, *F1-score* y *Area Under the Curve (AUC)*. Como resultado se encontró que el modelo propuesto por los autores (combinación de características filtradas con la clasificación SVM obtiene el mejor desempeño con un valor AUC de 0.93 en cross-validation, en comparación con los otros tres modelos.

La búsqueda de recurrencias distantes del cáncer de mama es presentado en [70]. Las recurrencias distantes se definen como la metástasis de un tumor primario hacia nodos linfáticos o locaciones mas allá del campo patológico, en este caso, locaciones distintas al tumor primario de pecho. Los autores combinaron datos extraídos de EHR, tanto de las narraciones clínicas, como de los datos estructurados con el objeto de determinar si el paciente presentaría una recurrencia distante del cáncer de mama. Los datos estructurados fueron extraídos de la base *Northwestern Medicine Enterprise Data Warehouse (NMEDW)*, correspondientes a variables demográficas, historia familiar, características de tumor y tratamiento. En el caso de los textos clínicos, se llevó a cabo un pre-procesamiento. Posteriormente, se realizó un anotado de los conceptos médicos utilizando *MetaMap*, obteniendo los CUIs mas relevantes. Una vez llevado a cabo el proceso anterior, los datos de CUIs se unieron con 18 variables estructuradas para crear las características para llevar a cabo la clasificación. Finalmente, estas características fueron utilizadas entrenar modelos de ML y *Deep Learning (DL)*. El modelo SVM fue entrenado con cuatro tipos de datos: conceptos médicos etiquetados por *MetaMap*, conjunto filtrado de *MetaMap*, solo datos estructurados y modelo BOW con TF-IDF. En el caso de los modelos de DL se consideró MLP entrenado con CUIs y datos estructurados, *Convolutional Neural Network (CNN)* entrenado con textos clínicos, CUIs y datos estructurados, y CNN utilizando la codificación de palabras con *word2vec* entrenado con la base de datos MIMIC-III, datos estructurados y CUIs. El estudio encontró que el modelo MLP entrenado con datos estructurados y valores filtrados de CUIs fue el que demostró el mejor desempeño predictivo, obteniendo una precisión del 81 %.

La comparación de técnicas de vectorización de textos oncológicos y su efecto en el desempeño de algoritmos de ML es presentado en [71]. Su estudio se basa en comparar modelos de ML con distintas técnicas de PLN para vectorizar texto, categorizando los reportes radiológicos con base en su seguimiento. Las categorías de seguimiento propuestas en el estudio son: progresión, mejora, enfermedad estable y resolución/no cáncer. El modelo propuesto realizó un pre-procesamiento de las notas (eliminación de puntuación, segmentación y *stemming*). Tres modelos de vectorización fueron utilizados para los textos: pesado TF-IDF, TF y *feature hashing*. Cada uno de los modelos fueron optimizados con base en sus parámetros. Para TF y TF-IDF se consideraron los parámetros de tamaño *K-skip*, frecuencia de N-grama mínima y proporción máxima de N-gramas. Los algoritmos de ML considerados para la clasificación fueron LR, *Random Decision Forest (RDF)*, SVM, *Bayes Point Machine (BPM)* y NN. Se encontró que el mejor conjunto de pre-procesamiento consistió en segmentación por unigramas y bigramas con TF-IDF, eliminación de palabras vacías, *stemming* y filtrado de características. El modelo encontrado con el mejor desempeño, bajo el pre-procesamiento óptimo mencionado anteriormente fue SVM, obteniendo una precisión del 90.6 %.

Un método de extracción de información relacionada con el cáncer, a partir de notas clínicas es presentado en [72], En este estudio los autores buscan la extracción de información sobre cáncer, basándose en la teoría de *frame semantics*, misma que se basa en eventos, estados y relaciones. Esta teoría indica que las palabras o frases que desencadenan un *frame* son unidades léxicas (*Lexical Units (LU)*), y los elementos descritos tienen un rol o son participantes del *frame* que define las características o atributos asociadas a las LU. En este estudio se consideran las LU como los distintos tipos de cáncer que pueden ser encontrados en las narraciones médicas, y los elementos del *frame* son la información que describe a un cáncer en particular, por ejemplo, estatus, características histológicas, etc. La metodología propuesta comienza por un conjunto de datos anotados con las LU y los elementos correspondientes a *frames* específicos. Los *frames* considerados fueron: diagnóstico del cáncer, procedimiento terapéutico y descripción del tumor. Posteriormente, se propone un flujo de anotación que extrae las LU en dos pasos. Primero, un sistema de reconocimiento de conceptos identifica todas las LU que desencadenan uno de los *frames* indicados anteriormente. Este sistema se basa en una combinación de redes *Long short-term memory (LSTM)* bidireccionales para capturar información tanto anterior como posterior a la oración, con CRF en la capa de de-codificación para considerar correlación, usado para predecir el *frame* de la palabra. Además, se considera la codificación por carácter en cada palabra. En el segundo paso, se asigna la LU interesada y la sentencia es dividida basándose en los LU asignados, donde cada una se centrará en un LU en específico. Finalmente, los resultados de cada análisis individual de las oraciones se unen en una oración final. Se realizó la experimentación considerando la concatenación de la representación por carácter de 100 dimensiones, con la representación de la palabra. Para la representación de palabras se utilizó GloVe con 100 dimensiones, codificación del dominio con 100 dimensiones basados en la base de datos MIMIC-III, y una unión de ambos. El estudio encontró que las anotaciones son realizadas con un F1-Score del 93.7 %, 96.33 % y 87.18 % para diagnóstico de cáncer, procedimiento terapéutico y descripción, respectivamente. Además, encontraron que la mejor metodología de codificación, por un pequeño margen, fue la combinación de GloVe con la codificación de dominio basada en MIMIC-III.

El proceso de identificación de enfermedades es presentado en [73]. Los autores buscan identificar pacientes con diagnóstico quirúrgico de neumoperitoneo, así como el diagnóstico médico de metástasis del leptomeníngeo, que es secundaria en la etapa cuatro del cáncer de mama, con el objetivo de mejorar el cuidado paliativo de los pacientes. La metodología consistió en la identificación de pacientes con algunas de estas enfermedades, utilizando códigos ICD-9. Posteriormente, con los registros encontrados anteriormente, se utilizó el software *ClinicalRegex* para identificación de oraciones que contuvieran palabras clave para las enfermedades. Para ambos padecimientos, se analizaron las notas basadas en palabras clave específicas para cada uno. En ambos casos, las notas analizadas fueron revisadas por un profesional del área correspondiente, donde se realzaban las palabras o frases clave encontradas. Mediante este estudio, los autores indican una disminución de alrededor del 95 % de notas clínicas utilizando la anotación mediante PLN, posterior al filtrado inicial con códigos ICD-9. De esta manera el tiempo que se lleva a cabo para la revisión de notas clínicas de pacientes con enfermedades críticas disminuye gracias al segundo filtrado.

La identificación de pacientes que requieren seguimiento adicional ante la sospecha de algún tipo de cáncer es presentado en [74]. En su estudio, los autores indican la necesidad de identificar pacientes que requerirán estudios adicionales ante la sospecha de tumores ma-

lignos de hígado, páncreas, riñón y glándulas suprarrenales. Durante el proceso, se toman notas radiológicas de pacientes que presentan algunos de estos tumores, así como se etiquetan para identificarlos como benigno, indeterminado, sospechoso o malignos. De esta manera, aquellos etiquetados como indeterminado o sospechoso son aquellos que necesitarán estudios posteriores. El flujo de pre-procesamiento de las notas consistió en la transformación de las palabras a minúsculas, eliminación de palabras vacías y caracteres especiales y *stemming*. Posteriormente, para generar las características de los documentos, se consideró el modelo BoW con unigramas, bigramas y trigramas. Dada la alta dimensionalidad de la representación se llevó a cabo el proceso *joint mutual information* para conservar las características relevantes. Una vez que se tuvieron las características, tres modelos de ML fueron utilizados, siendo estos: NB, DT y Máxima Entropía. Los resultados mostraron que el algoritmo DT obtuvo la mejor precisión en la clasificación, siendo esta del 86.2 %.

La extracción de la etapa del cáncer a partir de notas clínicas es presentado en [75]. En este estudio, los autores buscan extraer la etapa del cáncer basándose en el sistema TNM con el objetivo de generar anotaciones automáticas de esta información. La metodología consistió en la comparación de dos métodos para obtener la etapa del cáncer. El primero consistió en un flujo PLN para el tratamiento de las notas CT de los pacientes. El proceso PLN de pre-procesamiento consistió en la segmentación, eliminación de palabras vacías, aplicación de reglas para la extracción de información y aplicación de reglas para TNM para obtener la etiqueta de la etapa. El segundo método consistió en el uso de una red LSTM, utilizando GloVe para la codificación de las palabras. El estudio encontró que el método que mejor llevó a cabo el etiquetado fue el basado en LSTM obteniendo una precisión de hasta el 85 %.

La clasificación de pacientes con base en la etapa de cáncer de próstata es presentado en [76]. Los autores buscaron la clasificación de pacientes con este tipo de cáncer a partir de registros de EHR, que fueron mapeados a un modelo de datos conocido como *Observational Health Data Sciences (OHDSI)*, para posteriormente utilizar algoritmos de ML para llevar a cabo la clasificación. La metodología consistió en la comparación de pacientes encontrados con el tipo de cáncer, basándose en una búsqueda mediante código ICD y algoritmos de ML. Para la clasificación mediante algoritmos de ML se consideró la construcción de vectores de características basado en el paquete APHRODITE que considera la frecuencia de los siguientes elementos: condición, procedimientos, observación, exposición a medicamentos y valores de laboratorio. Los modelos de ML considerados para el estudio fueron RF, GBM, XGBoost y LR-LASSO. Los resultados mostraron que el modelo que presentó la mayor precisión fue RF obteniendo un 90 %, considerando una ventana de tiempo de 12 meses.

La predicción sobre uso adicional de recursos radiológicos por pacientes en vigilancia por carcinoma hepatocelular es presentado en [77]. En este estudio, los autores buscan predecir si es que un paciente utilizará recursos de imagen radiológica al estar en vigilancia por lesiones del hígado en caso de presentar un carcinoma hepatocelular. Esto con el objetivo de mejorar la demanda de dichos recursos en los pacientes. El proceso consistió en el uso de reportes radiológicos para predecir el uso de recursos, anotando cada uno de los reportes para indicar el uso o no de estos. El proceso de PLN para las notas consistió en un pre-procesamiento clásico, así como el uso de dos representaciones: BOW y TF-IDF. Posteriormente, tres algoritmos de ML fueron utilizados con cada una de las representaciones para predecir el uso de los recursos. Los modelos considerados para el estudio fueron LR, SVM y RF. El estudio encontró al modelo de TF-IDF con el modelo SVM como el que mejor se desempeñó, obteniendo una precisión del 92.2 %.

## 3.3. Comparativa de los trabajos relacionados

Con base en los trabajos discutidos en las secciones 3.1 y 3.2, se muestra la tabla 3.1, donde se realiza una comparación de dichos trabajos con el propuesto en esta tesis. Para la comparación se consideran los algoritmos utilizados, los tipos de datos y el tipo de cáncer analizado.

| Trabajo | Tipo de cáncer | Algoritmo(s) | Tipo de dato |
|---|---|---|---|
| Delen et al. [45] | Pecho | MLP, DT, LR | Estructurados |
| Montazeri et al. [46] | Pecho | NB, TRF, 1NN, AD, SVM, RBFN y MLP | Estructurados |
| Tapak et al. [47] | Pecho | NB, LR, LDA, RF, SVM, LS-SVM, AD y Adabag | Estructurados |
| Farooqui and Ritika [48] | Pecho | DT, RF y SVM | Estructurados |
| Sharma et al. [49] | Pecho | RF, k-NN y NB | Estructurados |
| Liu et al. [50] | Pecho | CSSVM | Estructurados |
| Lynch et al. [51] | Pulmón | LR, DT, GBM, SVM y Personalizado por votación | Estructurados |
| Faisal et al. [52] | Pulmón | NB, RF, SVM, MLP, GBT, NN y Sistema de voto | Estructurados |
| Kumar et al. [53] | Pulmón | SVM, k-NN, RF, ANN y DT | Estructurados |
| Deist et al. [54] | Pulmón, cabeza y cuello | EL-LR, RF, NN, SVM, LogitBoost y DT | Estructurados |
| Bartholomai and Frieboes [55] | Pulmón | GL, GBM, RF y Sistema de pesos | Estructurados |
| Pradeep and Naveen [56] | Pulmón | SVM, C4.5 y NB | Estructurados |
| Ksiazek et al. [57] | Hígado | SVM linear, SVM, MLP, k-NN, NB, LDA, QDA, LR y RF | Estructurados |
| Sato et al. [58] | Hígado | LR, LR con penalizado L1, L2 y elástico, SVM, GB, RF, NN y DL | Estructurados |
| Priya et al. [59] | Hígado | RF, SVM, J-48, MLP y NB | Estructurados |
| Rahman et al. [60] | Hígado | LR, k-NN, DT, SVM, NB y RF | Estructurados |
| AlicjaSala [61] | Hígado | RF, Bagging, AdaBoost, LR y DT | Estructurados |
| Abdelaziz et al. [62] | Hígado | ANN, SVM, DT, RepTree, RF, JRip y PART | Estructurados |
| Bozkurt et al. [63] | Pecho | BN | No Estructurados |
| Patel et al. [64] | Pecho | MOTTE (Basado en PLN) | No Estructurados |
| AAlAbdulsalam et al. [65] | Pulmón, Colon y Próstata | REGEX y CFR | No Estructurados |
| Forsyth et al. [66] | Pecho | CRF | No Estructurados |
| Gerevini et al. [67] | Orientado a lesiones | BoW, Heurísticas | No Estructurados |
| Koopman et al. [68] | Todos los presentes en conjunto de datos | SNOMED, ICDO, n-Gramas, Reglas y SVM | No Estructurados |
| Zeng et al. [69] | Pecho | MetaMap y BoW | No Estructurados |
| Zeng et al. [70] | Pecho | MetaMap, SVM, BoW, word2Vec, MLP y CNN | No Estructurados |
| Chen et al. [71] | Todos los presentes en conjunto de datos | BoW, RDF, SVM, BPM y NN | No Estructurados |
| Si and Roberts [72] | Todos los presentes en conjunto de datos | Frame Semantics, LSTM, CRF, GloVe | No Estructurados |
| Udelsman et al. [73] | Pecho | ClinicalRegex | No Estructurados |
| Lou et al. [74] | Hígado, páncreas, riñón y glándulas suprarrenales | BoW, Joint Mutual Information, NB, DT y Máxima Entropía | No Estructurados |
| Gupta et al. [75] | Todos los presentes en conjunto de datos | GloVe y LSTM | No Estructurados |
| Seneviratne et al. [76] | Próstata | Modelo OHDSI, APHRODITE, RF, GBM, XGBoost y LR-LASSO | No Estructurados |
| Brown and Kachura [77] | Hígado | BoW, LR, SVM y RF | No Estructurados |

Tabla 3.1: Tabla comparativa de trabajos de clasificación de cáncer e información clínica

A diferencia de los trabajos presentados en la tabla 3.1, la presente tesis aborda la integración de ambos tipos de datos, tanto estructurados como no estructurados, para verificar el desempeño de tres algoritmos de ML. Los algoritmos considerados para el presente trabajo toman en consideración algunos de los utilizados en los trabajos relacionados. Como es el caso de SVM, que demuestra un desempeño estable y en la mayoría de los casos uno los

mejores algoritmos para llevar a cabo la clasificación; de igual manera se considera la integración de MLP, ya que recientemente se han obtenido buenos resultados en la clasificación considerando métodos basados en redes neuronales. El caso de AdaBoost se considera dado que es un algoritmo que ha presentado buenos resultados y además es considerado como una combinación de clasificadores para dar un resultado final.

Finalmente, también podemos considerar que los trabajos presentados consideran información altamente especializada sobre el tipo de cáncer que se busca clasificar. En el caso del presente trabajo, se considera información más general que puede encontrarse en los expedientes clínicos de los pacientes con cáncer. Esto puede ser de gran ayuda, pues la información de pacientes puede ser ingresada con la metodología propuesta, sin requerir esfuerzo ni tiempo en filtrar la información.

# Capítulo 4

# Metodología de solución

En este capítulo se presenta la metodología propuesta para la clasificación de pacientes con cáncer de mama, pulmón e hígado, utilizando información de notas clínicas así como información estructurada de bases de datos clínicas.

El flujo completo se muestra en la Figura 4.1. La primera etapa consiste en la extracción tanto de notas clínicas como de datos estructurados de la base de datos clínica descrita en la sección 2.5. Posteriormente, cada conjunto de datos pasa por un pre-procesamiento para convertirlos en una representación numérica. Esta representación se utiliza como datos de entrada para los algoritmos de ML. Finalmente, los hiper-parámetros de los algoritmos son ajustados para obtener el modelo con el mejor desempeño.

Las siguientes secciones describen a detalle cada uno de los procesos llevados a cabo para obtener el modelo de ML con el mejor desempeño en la clasificación. La Sección 4.1 describee el proceso de extracción de los datos estructurados y no estructurados desde la base de datos clínica. La sección 4.2 describe el pre-procesamiento llevado a cabo tanto para datos estructurados como para no estructurados. La Sección 4.3 detalla el proceso de representación numérica para ambos tipos de datos. Finalmente, la Sección 4.4 muestra tanto los algoritmos de ML considerados, así como los hiper-parámetros considerados para cada uno.

## 4.1. Extracción de datos

La metodología parte de la extracción de los datos que serán utilizados por procesos posteriores. La base de datos considerada para esta metodología fue MIMIC-II[43], que contiene la información de pacientes admitidos al área de cuidados intensivos del hospital *Beth Israel Deaconess Medical Center* en Boston, EE. UU.. Esta base de datos contiene tanto información demográfica como estudios de laboratorio realizados a los pacientes. Su estructura general y los datos de pacientes contenidos en esta base de datos se describen en la Sección 2.5. El proceso de extracción fue llevado a cabo tanto para los datos estructurados, como para las notas clínicas de los pacientes.

Se lleva a cabo la extracción del identificador único (ID) de los pacientes. Esta extracción se realizó mediante identificación de pacientes con cáncer de hígado, pulmón o pecho, a través de su código ICD-9 [78]. Una vez obtenida la lista de pacientes a considerar, es posible proceder la información estructurada de cada uno de ellos, que consiste en resultados de estudios de laboratorio e información personal y demográfica.

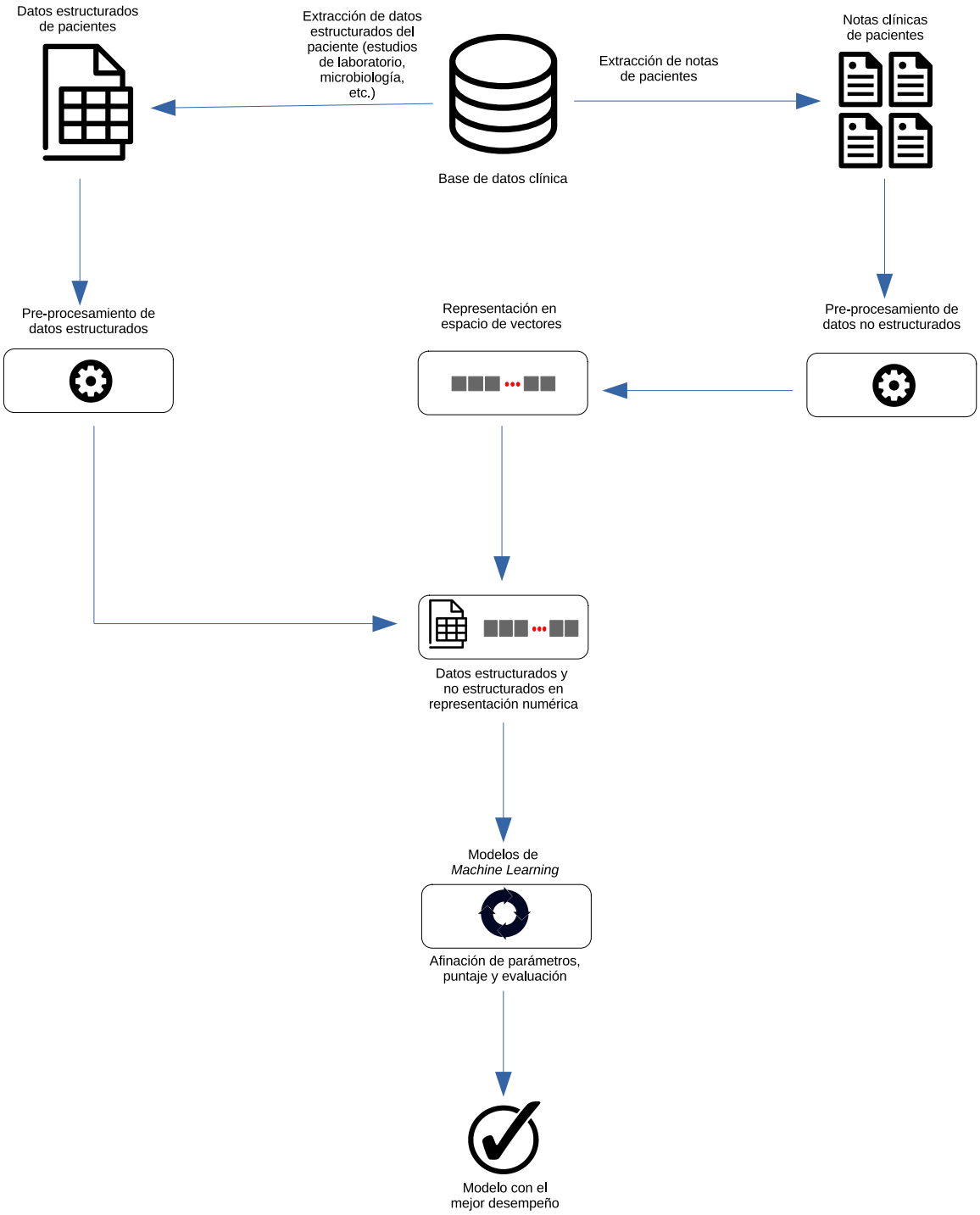



Figura 4.1: Metodología propuesta para la clasificación de pacientes con cáncer

La información estructurada a utilizar para cada uno de los pacientes consiste en 19 variables que se muestran en la Tabla 4.1.

| Variable | Descripción | Tipo |
|---|---|---|
| Sexo biológico (Binario) | Sexo del paciente | Alfanumérica (A) |
| Estatus Marital | Estatus marital del paciente | A |
| Etnicidad | Etnicidad del paciente | A |
| Religión | Religión del paciente | A |
| Tipo de admisión | Razón por la que el paciente ingresó al hospital | A |
| Fuente de admisión | Lugar desde el cuál fue transferido el paciente | A |
| Altura | Altura del paciente | Numérica (N) |
| Peso | Peso del paciente | N |
| UREA N | Valores del exámen de nitrógeno ureico en sangre | N |
| PLT CNT | Valores del estudio del conteo plaquetario | N |
| HCT | Valores del estudio de hematocrito | N |
| HGB | Valores del estudio de hemoglobina en sangre | N |
| MCHC | Valores del estudio de concentración corpuscular media de hemoglobina | N |
| MCH | Valores del estudio para hemoglobina corpuscular media | N |
| MCV | Valores del estudio para volumen corpuscular medio | N |
| RBC | Valores del estudio de conteo de células rojas | N |
| CREAT | Valores del estudio de nivel de creatinina en sangre | N |
| RDW | Valores del estudio de distribución de células rojas | N |
| WBC | Valores del estudio para conteo de células blancas | N |

Tabla 4.1: Variables estructuradas obtenidas de la base de datos MIMIC-II

Una vez obtenida la información estructurada de cada uno de los pacientes, esta se almacena en un archivo para ser utilizada por los proceso posteriores.

La información no estructurada consiste en el conjunto de notas clínicas de los pacientes. Estos archivos contienen narrativas sobre el diagnóstico y tratamiento de cada paciente. Para esto es necesario obtener el archivo de nota para cada uno de los pacientes considerados y renombrarlo con su ID, además de almacenarlo en un directorio para su manejo posterior.

Al finalizar el proceso de extracción, obtenemos un archivo que contiene el conjunto de datos estructurados por paciente en un archivo, que será utilizado por el pre-procesamiento de datos estructurados. Además se contará con el conjunto o *corpus* de notas clínicas de los pacientes que se utilizarán como datos de entrada al pre-procesamiento de datos no estructurados.

En el Algoritmo 1 se muestra el proceso de extracción descrito en esta sección.

**Algoritmo 1:** Algoritmo para extracción de datos estructurados del paciente.

**Datos:** $I$= Lista de pacientes con cáncer de pecho, pulmón o hígado.
**Resultado:** $A - Estructurados$= Archivo de pacientes con sus datos estructurados,
$D - Notas$= Directorio con el conjunto de notas clínicas
**para** *Paciente* $p$ *en* $I$ **hacer**
  $d$= Obtener datos demográficos de $p$
  $l$= Obtener resultados de estudios de laboratorio de $p$
  $n$= Obtener nota médica del paciente $p$
  Almacenar los datos de $d$, $l$ en el archivo $A - Estructurados$
  Renombrar el archivo $n$ con el ID del paciente $p$.
  Almacenar archivo $n$ en el directorio $D - Notas$
**fin**

## 4.2. Pre-procesado de notas clínicas y datos estructurados

Como se mostró en la sección 2.2.1, el pre-procesamiento en textos es necesario para dar un formato mas estructurado al texto y después convertirlo en una representación vectorial. A continuación se describe el proceso realizado tanto para las notas clínicas como para los datos estructurados.

### Pre-procesado de notas clínicas (textos)

El proceso de pre-procesamiento para los datos textuales propuesto para las notas clínicas consiste se muestra en la Figura 4.2 y se describe a continuación.

1. **Segmentación de archivo**: Este proceso permite obtener los "segmentos" relevantes dentro del archivo para la clasificación, en este caso se realiza la extracción de cada uno de los textos de narración médica dentro del archivo. En la Figura 4.3 se muestra un ejemplo de la segmentación de archivo realizada. La nota clínica por paciente se encuentra en formato CSV, separando valores dentro de la nota del paciente. Para nuestro interés, solo es necesario la narrativa de la nota, por lo cual es necesario remover toda la información adicional. Cabe mencionar que por archivo, existen varias secciones de narrativa, por lo cual es necesario extraer cada una de estas secciones de interés. En la figura se resalta en color rojo los distintos valores separados por coma que es necesario eliminar, así como en color verde, la sección del archivo que corresponde a la narrativa que se requiere extraer.

2. **Segmentación de texto**: Este proceso considera la división del texto en unidades atómicas, en este caso se considera la división por palabra. Los procesos posteriores analizan el texto por *token* para llevar a cabo el pre-procesamiento. En la Figura 4.4 se muestra una oración de ejemplo encontrada en una de las notas. En este proceso cada palabra es almacenada en una posición de un arreglo.

3. **Eliminación de caracteres especiales**: Debido a que los caracteres especiales dentro del texto no brindan información relevante, se analiza cada uno de los *tokens* obtenidos, y si se encuentra un carácter especial, este es eliminado. En la Figura 4.5 se muestra un ejemplo de palabras contenidas en el arreglo. Como se puede observar la posición resaltada con color rojo contiene una cadena compuesta solamente de caracteres especiales y numéricos. Por esta razón, como parte del pre-procesamiento se eliminan todos estos caracteres de dicha posición del arreglo, como al eliminarlos ya no queda más información, esta posición es eliminada del arreglo.

4. **Lematización de palabras**: Con la finalidad de reducir la dimensionalidad del texto, las palabras se llevan a su representación raíz. De esta manera distintas conjugaciones verbales son representadas de manera única. En la Figura 4.6 se muestra un ejemplo del proceso de lematización. Las palabras resaltadas en color azul son las que a través del proceso de lematización resultaran en aquellas resaltadas en color verde.

5. **Eliminación de palabras vacías**: Algunas de las palabras contenidas en el texto no tienen suficiente poder de discriminación. Estas palabras tienen una ocurrencia alta en todas las notas analizadas, por lo que se decide eliminarlas. Este proceso se llevó a cabo considerando una lista definida de palabras vacías en el idioma inglés [79]. En la

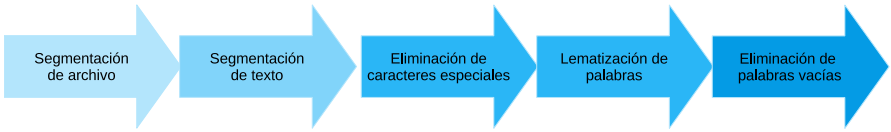



Figura 4.2: Flujo de pre-procesado aplicado a las notas clínicas.

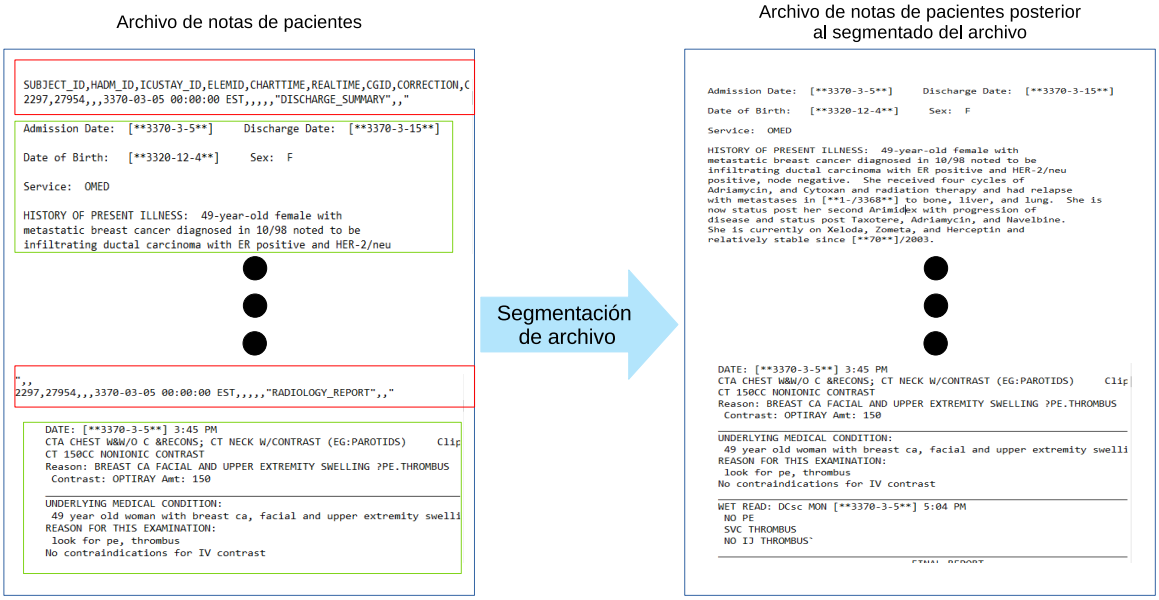



Figura 4.3: Ejemplo de segmentación de archivo aplicado a las notas clínicas

Figura 4.7 se muestra un ejemplo de la eliminación de palabras vacías. En la oración de la izquierda se resalta en color rojo las palabras que son consideradas como vacías en el ejemplo, posterior al proceso se eliminan de la oración, como se muestra en la parte derecha de la figura.

El proceso descrito anteriormente es aplicado al conjunto de notas obtenidas del proceso de extracción de datos. En la Figura 4.8 se muestra uno de los archivos del *corpus* que contienen notas clínicas del paciente, al que se le aplicó el pre-procesamiento.

Las distintas notas de un paciente se encuentran en un archivo dividido por comas, como se muestra resaltado en color rojo en la figura mencionada. Por esta razón es necesario obtener cada una de las notas, como segmentos del archivo original.

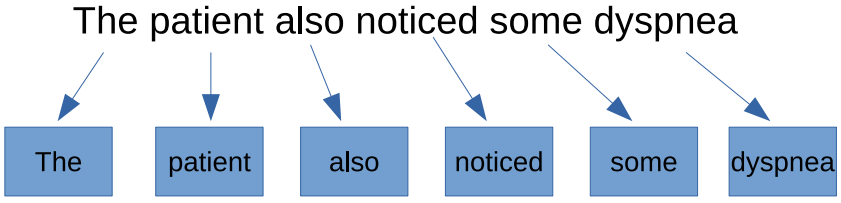



Figura 4.4: Ejemplo de segmentación de texto aplicado a las notas clínicas.

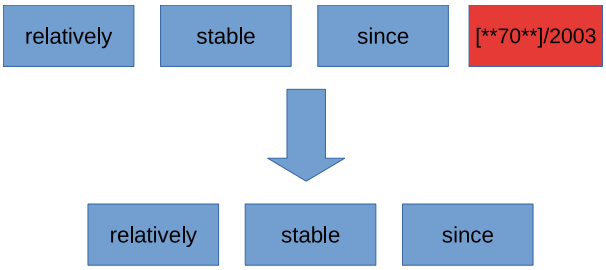



Figura 4.5: Ejemplo de eliminación de caracteres especiales del texto, aplicado a las notas clínicas.

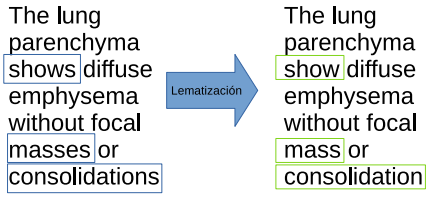



Figura 4.6: Ejemplo de lematización aplicado a una oración.

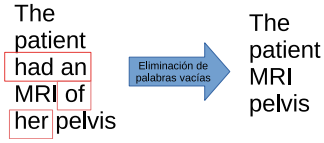



Figura 4.7: Ejemplo de eliminación de palabras vacías aplicado a una oración.

Posteriormente, cada uno de los segmentos obtenidos es considerado una nota médica, a la que se le aplica el proceso de segmentación. Mediante este proceso, se divide por palabras el texto obtenido.

La eliminación de caracteres especiales se realiza, ya que dentro de las notas se encuentra información extra, como la fecha de admisión y nacimiento, resaltado en color azul en la Figura 4.8. Los datos numéricos y los caracteres especiales como *,[,:,etc. no son relevantes para la clasificación, por lo cual son eliminados de la nota médica.

El proceso de lematización se llevó a cabo en palabras tales como la que se encuentra resaltada en color café. En este caso palabras conjugadas se llevaron a su representación base, en este caso *received*, se representó como *receive*.

Finalmente, con base en la lista de palabras vacías considerada para el proceso, las palabras como las resaltadas en color verde fueron eliminadas, ya que al ser muy comunes, no poseen gran capacidad de discriminación entre textos. La Figura 4.9 muestra un segmento de una nota clínica posterior al pre-procesamiento.

Como salida de este proceso obtenemos un nuevo *corpus* que contendrá el conjunto de notas clínicas pre-procesadas. Este conjunto de notas se utilizará como entrada al proceso de representación en espacio de vectores. El Algoritmo 2 muestra el procedimiento realizado para el pre-procesado de cada nota clínica.

**Algoritmo 2:** Algoritmo de pre-procesado de notas clínicas

```
Datos: C= Corpus de notas clínicas de los pacientes, V= Conjunto de palabras
       vacías, CV= Conjunto de caracteres a eliminar.
Resultado: C2= Corpus de notas clínicas pre-procesadas.
para cada Nota clínica c en C hacer
    S = Aplicar el proceso de segmentación a c
    para cada Segmento s en S hacer
        P = Aplicar segmentación de palabras a s
        para cada Palabra p en P hacer
            si La palabra p contiene algún caracter de CV entonces
                Elimina los caracteres
            fin
            Aplicar proceso de lematización a p
            si La palabra p está contenida en V entonces
                Elimina la palabra p
            fin
        fin
        Concatenar el segmento s a una variable auxiliar aux
    fin
    Almacenar aux en el corpus C2
    Limpiar los contenidos de aux
fin
```

Cabe mencionar que a través de este proceso, es posible que cualquier nota médica que respete la estructura descrita pueda ser pre-procesada. Esto con la finalidad de obtener información relevante en textos y ser preparada para su representación vectorial, usada para algoritmos de ML.

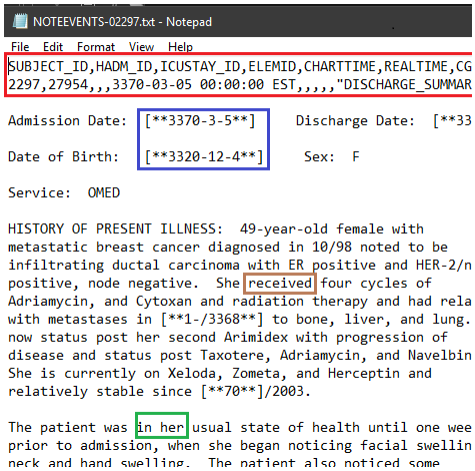



Figura 4.8: Ejemplo de una nota clínica sin pre-procesar.

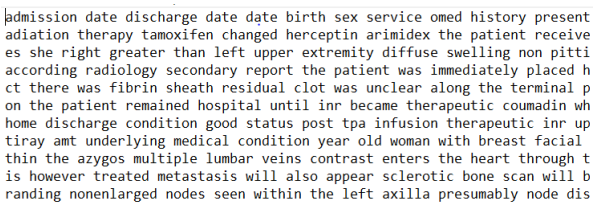



Figura 4.9: Ejemplo de una nota clínica posterior al pre-procesamiento.

### Pre-procesado de datos estructurados

Este proceso utiliza como datos de entrar el archivo de datos estructurados obtenidos de la extracción de datos.Para el conjunto de datos estructurados, el pre-procesamiento se basa en el tipo de variable, ya sea numérica o alfanumérica. La Figura 4.10 muestra un fragmento de ejemplo sobre los datos estructurados que se utilizaron. El proceso que se llevó a cabo fue el siguiente:

1. **Manejo de valores faltantes**: En los registros de los pacientes es común que los valores de algunas variables no estén especificados. En el caso de las variables categóricas se consideró la estrategia de completar estos valores, utilizando el valor más utilizado en la variable. Para el caso de las variables continuas se consideró utilizar el promedio de la variable para completar los datos faltantes.

2. **Codificación de valores categóricos**: Por la naturaleza de los algoritmos de ML a utilizar, es necesario que la entrada a los algoritmos considerados estén en una representación numérica. Por esto, los valores categóricos presentes en los registros tuvieron que ser codificados para ser representados de forma numérica. El método que se llevó a cabo para esta codificación corresponde al proceso de *One-Hot encoding*.

3. **Escalamiento**: Por la naturaleza de los datos, es común que las magnitudes de cada una de las categorías difieran entre sí. El proceso de escalamiento se llevó a cabo para evitar que exista un gran margen en la magnitud de las características, y de

| SubjectID | sex | marital_status | ethnicity | religion | admission_type | admission_source | height | weight_first | UREA N |
|---|---|---|---|---|---|---|---|---|---|
| 56 | F | | WHITE | NOT SPECIFIED | EMERGENCY | EMERGENCY ROOM ADMIT | | 49.2 | 21 |
| 103 | F | MARRIED | UNKNOWN/NOT SPECIFIED | CATHOLIC | EMERGENCY | TRANSFER FROM HOSP/EXTRAM | 165.1 | 61.5 | 7 |
| 143 | M | SINGLE | WHITE | NOT SPECIFIED | EMERGENCY | PHYS REFERRAL/NORMAL DELI | 180.34 | 113 | 9 |
| 150 | F | MARRIED | OTHER | UNOBTAINABLE | EMERGENCY | TRANSFER FROM HOSP/EXTRAM | 160.02 | 36.5 | 7 |
| 188 | M | MARRIED | WHITE | CATHOLIC | EMERGENCY | EMERGENCY ROOM ADMIT | | | 19 |
| 236 | M | MARRIED | PATIENT DECLINED TO ANSWER | CATHOLIC | EMERGENCY | PHYS REFERRAL/NORMAL DELI | 162.56 | 74 | 16 |
| 246 | M | MARRIED | WHITE | CATHOLIC | ELECTIVE | PHYS REFERRAL/NORMAL DELI | 182.88 | 95.4 | 23 |
| 281 | F | SINGLE | BLACK/AFRICAN AMERICAN | PROTESTANT QUAKER | EMERGENCY | EMERGENCY ROOM ADMIT | | 84.3 | 28 |
| 329 | M | MARRIED | ASIAN - VIETNAMESE | BUDDHIST | EMERGENCY | EMERGENCY ROOM ADMIT | | | 14 |
| 393 | M | SINGLE | WHITE | CATHOLIC | EMERGENCY | TRANSFER FROM HOSP/EXTRAM | 182.88 | 100.3 | 16 |

Figura 4.10: Ejemplo de los datos estructurados extraídos del expediente clínico del paciente, obtenida de la base de datos MIMIC-II[43].

esta manera, evitar que algunas de ellas cobren mayor importancia solamente por la diferencia en el margen.

Este proceso toma el archivo obtenido de la extracción y lo aplica a cada una de las variables consideradas. Como resultado se obtiene un nuevo archivo, con cada una de las variables pre-procesadas. El Algoritmo 3 muestra el procedimiento llevado a cabo para realizar el pre-procesado de datos estructurados, y obtener el nuevo archivo que será utilizado por procesos posteriores en la metodología.

**Algoritmo 3:** Algoritmo de pre-procesamiento de datos no estructurados

**Datos:** $A$= Archivo con datos estructurados de los pacientes
**Resultado:** $A_2$= Archivo con datos estructurados pre-procesados
**para cada** *Variable* $v$ *en* $A$ **hacer**
  **si** $v$ *es una variable alfanumérica* **entonces**
    Completa los valores faltantes de $v$ con el valor más utilizado en $v$
    Codifica de forma numérica los valores de $v$ usando *One-Hot encoding*
  **fin**
  **si no, si** $v$ *es una variable numérica* **entonces**
    Completa los valores faltantes de $v$ con el promedio de los valores de $v$
    Aplica escalamiento de valores a $v$
  **fin**
  Almacena los valores de $v$ en $A_2$
**fin**

La metodología de pre-procesado de datos estructurados permite transformar este tipo de datos, obtenidos a partir de bases de datos clínicas, y prepararlos para su uso por algoritmos de clasificación.

## 4.3. Representación de notas clínicas y datos estructurados

La representación de datos depende de los algoritmos de ML que se utilizarán para el proceso de clasificación. Por la selección de los algoritmos en este estudio, los valores de entrada deben ser vectores numéricos.

Para el caso de las notas clínicas, se utiliza como entrada el *corpus* compuesto por las notas clínicas de los pacientes. Cada una de las notas del conjunto debe ser representada como un vector de longitud fija, para ser utilizado como entrada de los algoritmos de ML.

En este caso el proceso de codificación a vector se basó en el proceso de *Paragraph Vectors (PV)* [25].Como parte del proceso descrito por PV se llevó a cabo el proceso de aprendizaje para la representación vectorial de cada palabra presente en el corpus.

Posteriormente, en conjunto con los vectores de palabras obtenidos, se lleva a cabo el proceso de aprendizaje del vector que representa el párrafo. En este caso el párrafo se define como cada uno de los archivos que contienen el conjunto de notas clínicas por paciente.

Lo anterior genera un modelo que puede ser utilizado para codificar un texto en un vector de longitud fija. Así se utiliza el modelo generado para obtener la representación vectorial de todo el conjunto de notas del *corpus*. Los vectores obtenidos son almacenados en un archivo para su manipulación posterior. El proceso de representación vectorial es presentado en el Algoritmo 4.

**Algoritmo 4:** Algoritmo para la representación vectorial de las notas clínicas

**Datos:** $C_2$= Corpus de notas clínicas pre-procesadas
**Resultado:** $AP$= Archivo con los vectores de párrafo de todos los pacientes
$MP$= Llevar a cabo el proceso de aprendizaje de representación de palabras del *corpus* $C_2$
$PV$= Llevar a cabo el proceso de aprendizaje de representación de documentos usando $MP$ y $C_2$
**para cada** *Nota* $n$ *en* $C_2$ **hacer**
    v = Obtener la representación vectorial de $n$ usando $PV$
    Agregar en $AP$ el vector $v$ junto con el $ID$ de la nota.
**fin**

Utilizando el archivo pre-procesado de datos estructurados, así como el archivo de vectores obtenido, se procede a realizar la unión de ambos tipos de datos.

Dado que los datos estructurados ya cuentan con una representación numérica, no es necesario realizar procesamiento adicional a estos datos para ser utilizados como datos de entrada a los modelos de ML. La unión propuesta se muestra en la Figura 4.11. Se toma como ejemplo las características estructuradas de un paciente, resaltado en recuadros de color azul el conjunto de características estructuradas obtenidas. En color rojo se muestran las características del vector obtenido a través de la representación en espacio vectorial de el conjunto de notas del paciente. En color verde se resalta la característica que indica a la clase que pertenece el conjunto anterior mencionado. En este caso indica si el paciente fue diagnosticado con cáncer de pecho, pulmón o hígado. Así, la unión de ambos tipos de características, junto con su etiqueta de clase, resultaran en un solo vector que representará la información del paciente.

Finalmente, a través de la unión los vectores provenientes de los datos no estructurados, así como las variables obtenidas a través de datos estructurados se unen en una representación en común (numérica) para que ambos tipos de datos sean utilizados en el proceso de aprendizaje y clasificación de los algoritmos de ML. Cada uno de los vectores de los pacientes, como son mostrados en la Figura 4.11, se unen para formar solo un archivo de valores separados por comas, del inglés *Comma-Separated Values (CSV)* que contendrá el conjunto de vectores de todos los pacientes, como se muestra en la Figura 4.12.

Gracias a la transformación de archivos de texto, pre-procesados en representación vectorial, es posible utilizar algoritmos de ML para llevar a cabo la clasificación. Dentro de la

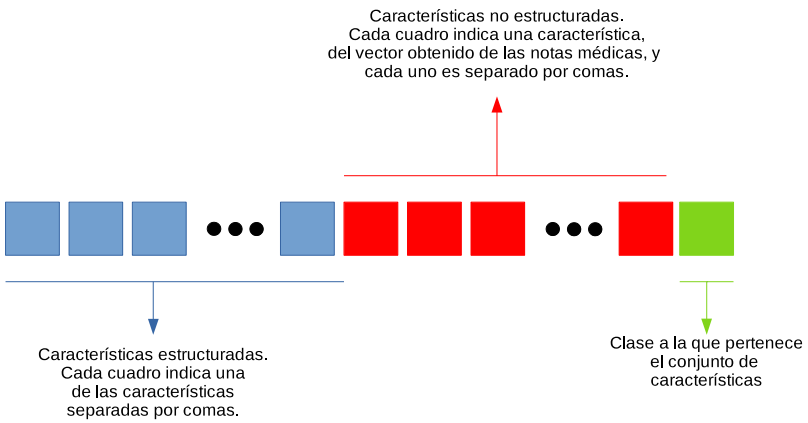



Figura 4.11: Forma de unión propuesta para las características estructuradas y no estructuradas.

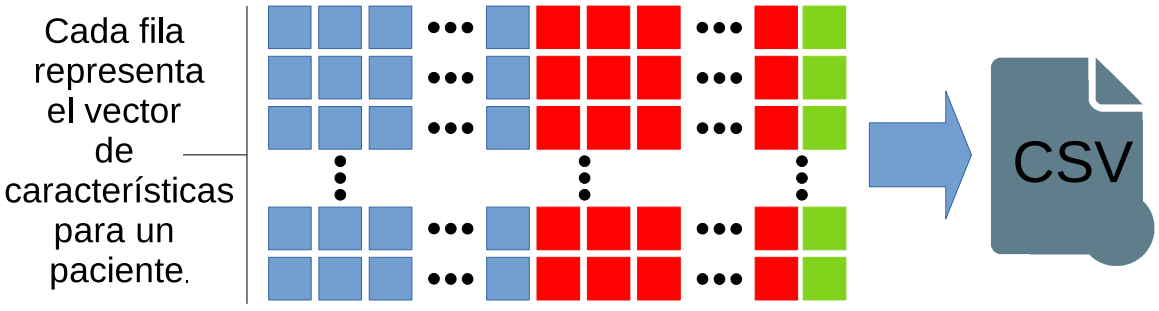



Figura 4.12: Unión de los vectores de cada paciente en un solo archivo CSV.

metodología se propone el uso de un algoritmo de PLN que ha demostrado tener buenos resultados en la codificación vectorial de textos. Así, considerando la metodología propuesta para el pre-procesamiento de datos no estructurados, el aplicar técnicas de representación vectorial para textos se realiza de manera directa.

## 4.4. Clasificación de notas clínicas

Una vez obtenida la representación numérica de los datos estructurados y no estructurados, se procede a realizar el proceso de aprendizaje y mejora de los algoritmos de ML para la clasificación de los pacientes. Donde el archivo de entrada para cada uno de los algoritmos será el archivo de unión de características estructuradas y no estructuradas.

Los algoritmos considerados para la tarea de clasficación fueron: Máquinas de Soporte Vectorial (SVM), Perceptrón Multi-Capa (MLP) y AdaBoost. El funcionamiento de cada uno de estos algoritmos está descrito en la Sección 2.4.

El proceso realizado para la clasificación de notas médicas dentro de la metodología general es presentado en la Figura 4.13. En esta figura se muestra el proceso realizado dentro del bloque de algoritmos de ML presentado en la metodología general. Como se puede observar, los datos de entrada son los archivos csv generados de la unión de datos estructurados y no estructurados (Figura 4.12).

Cada uno de los modelos serán entrenados y evaluados de manera separada, cada uno considerando el mismo conjunto de datos. En las siguientes secciones se discutirán los hiper-parámetros de cada modelo, que serán considerados para su entrenamiento. Además se presentan las métricas bajo las cuales se medirá el desempeño de cada uno de los modelos.

### 4.4.1. Hiper-parámetros de modelos

Cada algoritmo considerado tiene su conjunto de hiper-parámetros cuyo valor es utilizado para controlar el proceso de aprendizaje.

Los hiper-parámetros de SVM afectan principalmente la generación del hiper-plano basándose en las muestras tomadas del conjunto para generarlo, así como la dimensionalidad del conjunto de datos. Los hiper-parámetros considerados para ajustar en este modelo son los siguientes:

- **Kernel**: Este parámetro es utilizado para indicar la función de transformación para un espacio $N - dimesional$ a uno $M - dimensional$, tal que $N < M$. Es utilizado cuando los problemas de clasificación no son separables de forma lineal.

- **C**: Este parámetro es utilizado para la penalización de los errores en la clasificación. Ya que el algoritmo busca el mayor margen entre los vectores de soporte para la clasificación, el parámetro C indica la penalización de que una de las muestras se encuentre mal clasificada, pero que aumente el margen entre los vectores. Es considerado para la compensación entre un margen amplio y errores en la clasificación.

- **Gamma**: Define el alcance de las muestras para definir la región de decisión; es decir, indica la lejanía de las muestras que se considerarán para definir la región de clasificación. Ante altos valores de gamma, las muestras mas lejanas serán consideradas para definir la región.

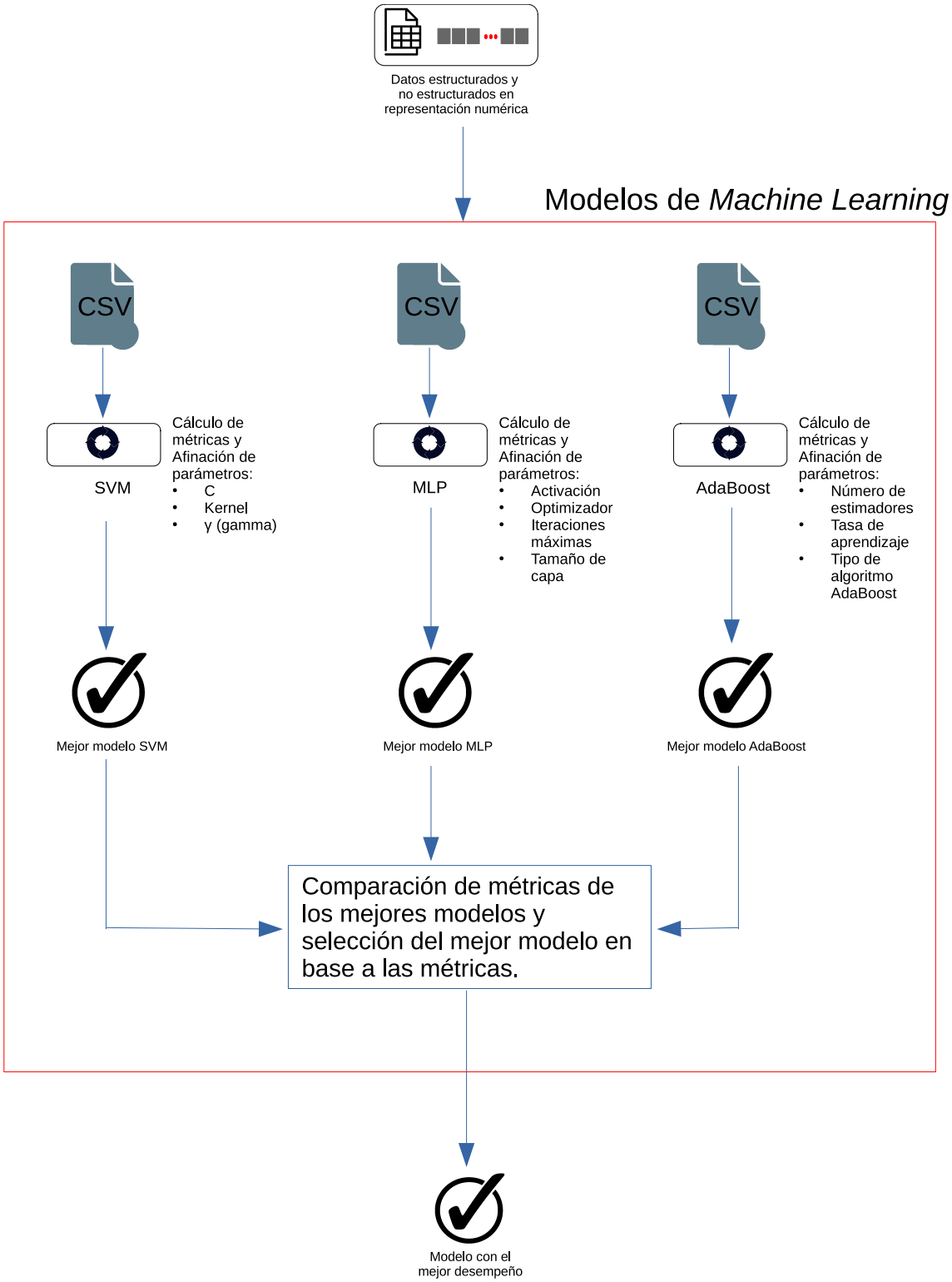



Figura 4.13: Sección de procesamiento de ML dentro de la metodología general de solución.

Los principales parámetros del modelo MLP modifican la manera en que se generan las salidas, así como la actualización de los pesos del modelo. Los hiper-parámetros considerados para este modelo son los siguientes:

- **Activación**: Indica la función de activación para los perceptrones. Esta función tiene características que activarán o no la salida del perceptrón, así como los valores que se presentarán. Por ejemplo, una función sigmoide confina los valores de salida del perceptrón entre 0 y 1 con base en los pesos de entrada, lo cual es útil para indicar probabilidad como salida.

- **Optimizador**: Es el algoritmo utilizado para la actualización de los pesos. Durante el proceso de aprendizaje de la red neuronal, su salida es comparada con los valores esperados. Basándose en esto se calcula la función de error y los pesos de la red son actualizados.

- **Tasa de aprendizaje**: Es el valor utilizado como "paso" para actualizar los pesos en la red. Los valores altos de la tasa de aprendizaje indican que la actualización de los pesos realizados por el optimizador se realizará con una mayor o menor amplitud. Este parámetro afecta la convergencia pues los valores pequeños pueden quedar atrapados en mínimos locales, mientras que valores grandes pueden incluso divergir de la solución. Cabe mencionar que este parámetro es utilizado por los optimizadores de descenso de gradiente.

- **Iteraciones máximas**: Número máximo de iteraciones que realizará el proceso de aprendizaje de la red. Incluye el proceso de optimización de los pesos de la red.

- **Tamaño de la capa**: Número de perceptrones por capa de la red.

Los hiper-parámetros de AdaBoost consideran principalmente a los clasificadores a utilizar. Los hiper-parámetros considerados para este modelo son los siguientes:

- **Número de estimadores**: Número de clasificadores "débiles" a utilizar. Estos son los clasificadores que se entrenarán con el conjunto de datos para contribuir a la decisión de clasificación.

- **Tasa de aprendizaje**: Indica la contribución de cada clasificador en la decisión. Este parámetro incrementa o decrementa los pesos asignados a las muestras del conjunto de pruebas, acelerando o disminuyendo el proceso de aprendizaje.

- **Algoritmo**: Algoritmo usado para llevar a cabo el proceso de *AdaBoost*. Originalmente AdaBoost fue propuesto sin restricción del número de clases; sin embargo se ha mejorado este algoritmo para trabajar en estos escenarios.

Considerando los hiper-parámetros descritos anteriormente para los algoritmos. Cada uno será entrenado realizando una variación de dichos parámetros.

### 4.4.2. Métricas de evaluación

Para poder determinar el desempeño de cada modelo obtenido, es necesario realizar un cálculo de métricas. Las métricas consideradas para la evaluación del desempeño de los

modelos de clasificación fueron precisión (*precision* ), recuperación (*recall* ) y $F_1$*-score*. Estas métricas son mostradas en [80] y se discuten a continuación.

La *precision*, como se muestra en la Ecuación 4.1, es la razón entre los casos clasificados como verdaderos positivos (TP), y la suma de los verdaderos positivos y los falsos positivos (FP). En este contexto los verdaderos positivos son aquellas instancias clasificadas en la clase correcta, y los falsos positivos fueron aquellas instancias que fueron clasificadas en la clase, pero que realmente no pertenecen a ella. Se considera esta métrica como el desempeño que tiene el algoritmo en clasificar correctamente las instancias. Su valor va de 0 a 1, como 0 indicando que todas las instancias fueron clasificadas de manera errónea.

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

El valor de la recuperación, tal como se muestra en la Ecuación 4.2, es la razón entre los casos clasificados como verdaderos positivos, y la suma los verdaderos positivos y los falsos negativos (FN). Los valores considerados son similares a los de la métrica de precisión, pero se introducen los valores de los falsos negativos. En este caso los falsos negativos son aquellas instancias que fueron clasificados erróneamente en otra clase. Se puede considerar esta métrica como el desempeño del clasificador en encontrar las instancias pertenecientes a la clase.

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

Por último, la métrica $F_1$*-score* considera las dos anteriores para indicar el desempeño del modelo. Esta métrica se interpreta como el promedio ponderado de la precisión y la recuperación, con contribuciones equitativas entre ambas métricas. Su cálculo se muestra en la Ecuación 4.3.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.3}$$

Para cada uno de los algoritmos se lleva a cabo un proceso de evaluación, considerando métricas estándar. Estas métricas son utilizadas para ajustar los hiper-parámetros de los algoritmos, para así obtener el modelo con el mejor desempeño así como el conjunto de hiper-parámetros que generan dicho modelo. Este proceso es realizado de manera individual para cada uno de los modelos de ML considerados. Es decir. Una vez obtenido el mejor modelo de cada uno de los algoritmos, se procederá a comparar sus métricas, para de esta manera elegir el modelo que mejor desempeño presenta para la clasificación de los tipos de cáncer.

### 4.4.3. Entrenamiento y evaluación de modelos

Una vez definidos los hiper-parámetros bajo los cuales se trabajarán en cada uno de los modelos, así como las métricas de evaluación de los mismos, se procede a discutir el proceso de entrenamiento y evaluación del conjunto de modelos.

Como se presentó al inicio de la Sección 4.4, cada uno de los modelos se entrenarán de manera individual bajo el mismo conjunto de datos de entrada. Para realizar el proceso de entrenamiento, en cada modelo se definirá el conjunto de posibles valores para los hiper-parámetros propios del algoritmo. Posteriormente se procederá a realizar el proceso de entrenamiento bajo cada uno de los posibles valores considerados para cada hiper-parámetro.

El modelo será entrenando considerando todas las combinaciones posibles entre el conjunto de valores para cada hiper-parámetro. Para cada modelo entrenado se realizará la evaluación del modelo bajo las métricas presentadas en la Sección 4.4.2. Una vez obtenidas las métricas de desempeño para todas las diferentes configuraciones de hiper-parámetros del modelo, se procederá a obtener el que mejor desempeño presentó.

El proceso anteriormente descrito será aplicado para cada uno de los algoritmos de ML, para así, obtener el mejor modelo generado para cada uno de los algoritmos. Es decir, se obtendrá el modelo de SVM, MLP y AdaBoost con el mejor desempeño.

El Algoritmo 5 muestra el proceso seguido para el entrenamiento y evaluación del modelo SVM. De igual manera el Algoritmo 6 y el Algoritmo 7 presentan el mismo proceso para MLP y AdaBoost respectivamente.

**Algoritmo 5:** Algoritmo para el proceso de entrenamiento y evaluación de SVM

**Datos:** $D$= Archivo con los vectores que representan a cada paciente,$C$= Conjunto de posibles valores para el hiper-parámetro C, $G$= Conjunto de posibles valores para el hiper-parámetro gamma, $K$= Conjunto de posibles valores para el hiper-parámetro kernel

**Resultado:** $M$= Mejor modelo de SVM

**para cada** *Valor c en C* **hacer**
  **para cada** *Valor g en G* **hacer**
    **para cada** *Valor k en K* **hacer**
      Entrenar modelo SVM usando los hiper-parámetros $c, g, k$ en un conjunto de entrenamiento $x \subset D$
      Evaluar el modelo generado bajo las métricas consideradas con un conjunto de prueba $y \subset D$
      Registrar en *aux* las métricas obtenidas por el modelo, así como su configuración.
    **fin**
  **fin**
**fin**
Obtener de *aux* el modelo con el mejor desempeño

Finalmente, al obtener el mejor modelo para cada uno de los algoritmos, es posible comparar entre ellos su desempeño, y así obtener el mejor modelo para la clasificación de pacientes con cáncer de pecho, pulmón e hígado.

Introduciendo dentro de la metodología un proceso de evaluación y mejora, permite seleccionar el mejor algoritmo de clasificación para una tarea en específico. Este caso es aplicado al cáncer de pecho, pulmón e hígado, y a través de la evaluación de los modelos encontrados, es posible encontrar aquél modelo que mejor se desempeñe en la clasificación. Esto permite que ante la adición de nuevos datos, el sistema se encuentre en mejora continua y se pueda decidir por un modelo de clasificación u otro basándose en sus métricas de desempeño.

**Algoritmo 6:** Algoritmo para el entrenamiento y evaluación de MLP

**Datos:** $D$= Archivo con los vectores que representan a cada paciente, $A$= Conjunto de posibles valores para la función de activación, $O$= Conjunto de posibles optimizadores, $I$= Conjunto valores para las iteraciones máximas, $C$= Conjunto de posibles valores para el tamaño de capa

**Resultado:** $M$= Mejor modelo de MLP

```
para cada Valor a en A hacer
    para cada Valor o en O hacer
        para cada Valor i en I hacer
            para cada Valor c en C hacer
                Entrenar modelo MLP usando los hiper-parámetros a, o, i, c en un
                 conjunto de entrenamiento x ⊂ D
                Evaluar el modelo generado bajo las métricas consideradas con un
                 conjunto de prueba y ⊂ D
                Registrar en aux las métricas obtenidas por el modelo, así como su
                 configuración.
            fin
        fin
    fin
fin
```

Obtener de *aux* el modelo con el mejor desempeño

**Algoritmo 7:** Algoritmo para el proceso de entrenamiento y evaluación de AdaBoost

**Datos:** $D$= Archivo con los vectores que representan a cada paciente, $E$= Conjunto de posibles valores para el número de estimadores, $T$= Conjunto de posibles valores para la tasa de aprendizaje, $V$= Conjunto de posibles valores para el tipo de algoritmo AdaBoost

**Resultado:** $M$= Mejor modelo de AdaBoost

```
para cada Valor a en A hacer
    para cada Valor t en T hacer
        para cada Valor v en V hacer
            Entrenar modelo AdaBoost usando los hiper-parámetros a, t, v en un
             conjunto de entrenamiento x ⊂ D
            Evaluar el modelo generado bajo las métricas consideradas con un
             conjunto de prueba y ⊂ D
            Registrar en aux las métricas obtenidas por el modelo, así como su
             configuración.
        fin
    fin
fin
```

Obtener de *aux* el modelo con el mejor desempeño

# Capítulo 5

# Experimentación y Resultados

En esta sección se presentan el proceso de experimentación llevado a cabo mediante la metodología propuesta en la Sección 4. Además se presentan las características del *hardware* y *software* mediante los cuales se ejecutó el proceso.

## 5.1. Especificaciones técnicas

El proyecto se realizó utilizando el lenguaje de programación java versión 1.8, actualización 181. Además de utilizar las siguientes librerías:

- Aprendizaje Profundo para Java, del inglés *Deep Learning for Java (dl4j)*[81] en su versión 1.0.0-beta3
- Commons-csv [82]
- OpenNLP versión 1.9.1

El lenguaje java se utilizó principalmente por la flexibilidad que se cuenta en el desarrollo, ya que su código puede ser ejecutado en cualquier dispositivo que cuente con su implementación de la máquina virtual de java, del inglés *Java Virtual Machine (JVM)*. Además se cuenta con la disponibilidad de bibliotecas como dl4j, que se encuentra en constante mejora al ser un proyecto *open-source*. Así mismo, esta biblioteca cuenta con una implementación de Doc2Vec, que permitió llevar a cabo el que cuenta con la implementación del algoritmo Doc2Vec. En el caso de la biblioteca Commons-csv se utilizó para procesar los archivos CSV que correspondían a las notas médicas de los pacientes. En el caso de OpenNLP se utilizó para realizar tareas del pre-procesamiento de los datos no estructurados, más específicamente, la segmentación de texto en *tokens*.

Para el manejo de la información se utilizó el servidor de base de datos PostgreSQL [83]. Esta decisión se tomó ya que la base de datos MIMIC-II [43] indica este servidor como el oficial para cargar dicha base. La base de datos se encuentra en archivos con formato csv, que en total tienen un peso 2.4 Gb. Esto es dado que no se consideró el componente de formas de onda de la base de datos, debido a que esta información no era útil para el alcance del presente trabajo. La decisión sobre el uso de esta base de datos se da debido a que fue debido a que su disponibilidad y acceso. Existen otras bases de datos con información de pacientes, sin embargo, debido al complicado uso de información de pacientes, muchas

de ellas restringen los accesos a la información, haciendo mas complicada la recolección de datos. Además, la decisión sobre utilizar información en el idioma inglés se da, de igual manera, por la disponibilidad de la información. Hay menor cantidad de información pública y disponible, sobre pacientes en el idioma español.

Otro lenguaje de programación utilizado para el desarrollo de este proyecto fue Python [84],en su versión 3.7. Además se utilizaron los siguientes paquetes:

- Pandas versión 0.23
- scikit-learn versión 0.19.2

La razón de utilizar Python se da debido a la gran facilidad que brinda el lenguaje para realizar manipulación de datos, a través de Pandas. Además de el uso de las implementaciones de los algoritmos de SVM, MLP y AdaBoost presentes en scikit-learn.

En cuanto a especificaciones de hardware, se utilizó una PC personal con las siguientes características:

- Procesador Intel Core I5-4690K / 3.5 GHz
- Memoria RAM 8 Gb a 1866 MHz
- Sistema Operativo Microsoft Windows 10 de 64 bits
- Almacenamiento de 1 Tb de Disco Duro

## 5.2. Configuración de experimentos

Dado que la presente tesis aborda el tema de clasificación de tipos de cáncer basado en datos estructurados y no estructurados, los experimentos realizados tuvieron la finalidad de buscar el mejor algoritmo de clasificación para desempeñar esta tarea. Para determinar esto, se realizaron tres tipos de experimentos considerando las distintas fuentes de datos que se tienen, siendo estas estructuradas y no estructuradas. De esta manera, los algoritmos de ML fueron entrenados considerando tres configuraciones de datos de entrada:

1. Entrenamiento utilizando solo los vectores obtenidos de los datos no estructurados.
2. Entrenamiento utilizando los datos estructurados pre-procesados.
3. Entrenamiento considerando la unión de ambos tipos de datos, considerando los vectores de datos no estructurados así como los datos estructurados.

Para la generación de los vectores de párrafo para las notas de los pacientes se consideraron un total de 10,518 notas distribuidas a través de 225 pacientes, como se muestra en la Tabla 5.1. Además se consideraron los siguientes valores para el algoritmo de PV:

- **Tasa de aprendizaje**: 0.020
- **Tasa mínima de aprendizaje**: 0.001
- **Tamaño de capa**: 300 (por recomendación de los autores [24])
- **Epochs**: 45

| Tipo Cáncer | # de pacientes | # de notas | Promedio de notas por paciente |
|---|---|---|---|
| Pulmón | 75 | 3653 | 48 |
| Pecho | 75 | 2157 | 28 |
| Hígado | 75 | 4708 | 62 |
| Total | 225 | 10518 | |

Tabla 5.1: Total de notas médicas utilizadas por tipo de cáncer

| Hiper-parámetro | Conjunto de valores |
|---|---|
| C | Valores del 1 a 1000 en incremetos de 100 |
| $\gamma$ | Valores del 0.1 al 1 en incrementos de 0.1 |
| Kernel | RBF |

Tabla 5.2: Conjunto de valores para hiper-parámetros de SVM usados para llevar a cabo el entrenamiento.

En todos los casos se consideró una partición de 80 % de los datos para fines de entrenamiento del algoritmo, así como un 20 % de datos para validación. Se utilizó la implementación de cada uno de los algoritmos de ML utilizando el API de Python *scikit-learn* [85]. Para el entrenamiento de los modelos, fue importante considerar el problema de *over-fitting* [86], que implica que un algoritmo de ML se desempeña con métricas muy buenas (quizás perfectas), sin embargo, esto solo logra hacerlo bajo sus datos de prueba. En el caso de que se le presenten nuevos conjuntos de datos, que nunca han sido alimentados al modelo, este fallará y sus puntajes en las métricas se verán altamente afectados. Esto debido a que el modelo presenta problemas de generalización fuera de sus datos de entrenamiento. Este problema utilizó un proceso de validación cruzada *(cross-validation)* [87], utilizando el método *k-fold*. Este proceso realiza subdivisiones del conjunto de datos de entrenamiento, posteriormente uno de estos subconjuntos funciona como datos de validación, mientras que los demás funcionan como datos de entrenamiento. En cada una de las iteraciones del algoritmo de validación cruzada, la división del conjunto de prueba se hace de manera distinta y se calcula la desviación estándar del modelo en cada iteración del proceso.

Para el proceso de búsqueda óptima de hiper-parámetros se utilizó el método *GridSearchCV* contenido en *scikit-learn*. Este proceso busca un conjunto de valores posibles para los hiper-parámetros de los modelos, y realiza el entrenamiento y validación de los modelos con cada uno de estos hiper-parámetros. Además, aplica el proceso de validación cruzada para cada iteración del mismo y busca el mejor modelo entre los parámetros de configuración, tomando en cuenta las métricas mencionadas anteriormente.

El conjunto de hiper-parámetros considerado para la búsqueda a través de *GridSearchCV* se muestran en la Tabla 5.2, Tabla 5.3 y Tabla 5.4 para los modelos de SVM, MLP y AdaBoost respectivamente.

| Hiper-parámetro | Conjunto de valores |
|---|---|
| Tamaño de capa | Valores del 300 al 600 en incrementos de 100 |
| Función de activación | identity, logistic, tanh y relu |
| Optimizador | lbfgs, sgd y adam |
| Iteraciones máximas | Conjunto de valores del 100 al 1000 en incrementos de 100 |

Tabla 5.3: Conjunto de valores para hiper-parámetros de MLP usados para llevar a cabo el entrenamiento.

| Hiper-parámetro | Conjunto de valores |
|---|---|
| Número de estimadores | Valores del 50 al 100 en incrementos de 1 |
| Tasa de aprendizaje | Valores del 0.1 al 1 en incrementos de 0.01 |
| Algoritmo de AdaBoost | SAMME y SAMME.R |

Tabla 5.4: Conjunto de valores para hiper-parámetros de AdaBoost usados para llevar a cabo el entrenamiento.

## 5.3. Análisis y discusión de resultados

En esta sección se presentan los resultados obtenidos al entrenar los algoritmos de ML con las configuraciones discutidas en la sección 5.2. Cada una de las subsecciones siguientes presenta los resultados considerando las distintas configuraciones, así como la discusión de los mismos. En cada uno de los casos la representación vectorial de los datos descritos al inicio de este capítulo es utilizada como valor de entrada en los algoritmos. Posteriormente, su desempeño es evaluado para realizar el ajuste de los hiper-parámetros y obtener el mejor modelo para el conjunto de datos utilizado. En cada sección se presentan los valores de los hiper-parámetros específicos para cada modelo, así como la precisión promedio obtenida por cada uno de los casos presentados. Además se mostrará el reporte detallado de clasificación del mejor caso para cada modelo de clasificación.

### 5.3.1. Configuración solo con datos no estructurados

Esta configuración consideró solamente los vectores obtenidos a través del algoritmo de representación vectorial. Los resultados obtenidos bajo esta configuración para las máquinas de soporte vectorial se muestran en la Tabla 5.5.

Como se puede observar en la Tabla 5.5 y su gráfica correspondiente (Figura 5.1), el modelo que mostró el mejor desempeño bajo la métrica de precisión obtuvo un 87 %, con una diferencia de 0.021 en la desviación estándar, considerando el modelo con la medida de desviación más baja. También se puede notar que para los mejores casos, se tiene un valor de C y $\gamma$ altos, aunque para los casos más bajos, un valor alto de $\gamma$ compensa un valor bajo de C. También se observa una diferencia mínima entre el caso 1 y el caso 10, que difieren solo con 0.002. El reporte detallado de clasificación para el mejor modelo (caso 1) es presentado en la Tabla 5.6. En esta tabla se puede observar que este modelo se desempeña mejor en la clasificación de cáncer de pecho bajo la métrica de precisión, mientras que el desempeño también es muy bueno considerando la medida de recuperación para el cáncer de hígado,

| Caso # | C Valor | Valor $\gamma$ | Precisión | $\sigma$ |
|---|---|---|---|---|
| 1 | 100 | 0.8 | **0.870** | 0.072 |
| 2 | 100 | 0.6 | 0.868 | 0.080 |
| 3 | 100 | 0.7 | 0.866 | 0.084 |
| 4 | 100 | 0.3 | 0.864 | 0.093 |
| 5 | 1 | 0.5 | 0.861 | 0.059 |
| 6 | 100 | 0.4 | 0.860 | 0.084 |
| 7 | 100 | 0.9 | 0.857 | 0.082 |
| 8 | 1 | 0.8 | 0.855 | 0.052 |
| 9 | 1 | 0.9 | 0.854 | 0.051 |
| 10 | 1 | 0.4 | 0.850 | 0.074 |

Tabla 5.5: Mejores 10 configuraciones de SVM para clasificación de pacientes con cáncer utilizando datos estructurados

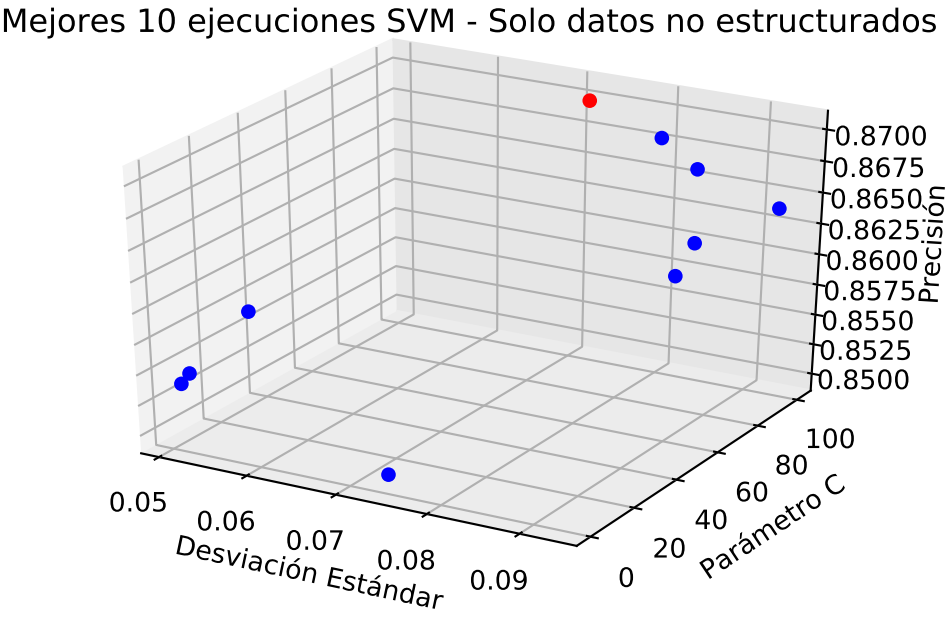



Figura 5.1: Gráfica de los 10 mejores casos de SVM entrenados en la configuración de datos no estructurados.

| | Precisión | Recall | $F_1$-Score |
|---|---|---|---|
| Pecho | 0.92 | 0.80 | 0.86 |
| Hígado | 0.87 | 1.0 | 0.93 |
| Pulmón | 0.83 | 0.83 | 0.83 |
| Promedio | 0.87 | 0.87 | 0.87 |

Tabla 5.6: Puntajes obtenidos por el mejor modelo de SVM entrenado con datos no estructurados

| Caso # | Función de activación | Tamaño de capa escondida | Iter. Max. | Precisión | $\sigma$ |
|---|---|---|---|---|---|
| 1 | Relu | 300 | 500 | **0.890** | 0.065 |
| 2 | Relu | 300 | 200 | 0.865 | 0.109 |
| 3 | Relu | 400 | 800 | 0.864 | 0.057 |
| 4 | Logistic | 300 | 600 | 0.862 | 0.082 |
| 5 | Tanh | 400 | 100 | 0.861 | 0.045 |
| 6 | Logistic | 500 | 700 | 0.860 | 0.064 |
| 7 | Relu | 500 | 100 | 0.859 | 0.042 |
| 8 | Tanh | 300 | 800 | 0.858 | 0.034 |
| 9 | Identity | 400 | 800 | 0.857 | 0.053 |
| 10 | Logistic | 400 | 200 | 0.856 | 0.079 |

Tabla 5.7: Mejores 10 configuraciones de MLP para clasificación de pacientes con cáncer utilizando datos no estructurados

obteniendo una medida de 1.0.

Para el modelo MLP, los hiper-parámetros de los 10 mejores casos son presentados en la Tabla 5.7 y su gráfica correspondiente (Figura 5.2). Se puede observar que el modelo con el mejor desempeño presenta una precisión del 89 % con una diferencia máxima de 0.0034 entre el mejor y el peor caso mostrado. Además, se observa que la función de activación utilizada por los tres primeros casos es la función *relu*, con un tamaño de capa de 300, que corresponde al tamaño de los vectores obtenidos por el algoritmo *Paragraph Vectors*.

El reporte detallado de clasificación para el mejor modelo de MLP se presenta en la Tabla 5.8. Puede observarse como este modelo gana en desempeño a SVM en precisión para el cáncer de pulmón e hígado, y además, muestra un excelente desempeño bajo la métrica de recuperación para el cáncer de pecho.

| | Precisión | Recall | $F_1$-score |
|---|---|---|---|
| Hígado | 0.95 | 0.90 | 0.93 |
| Pecho | 0.73 | 1.00 | 0.85 |
| Pulmón | 0.91 | 0.71 | 0.80 |
| Promedio | 0.89 | 0.87 | 0.87 |

Tabla 5.8: Puntajes obtenidos por el mejor modelo de MLP entrenado con datos no estructurados

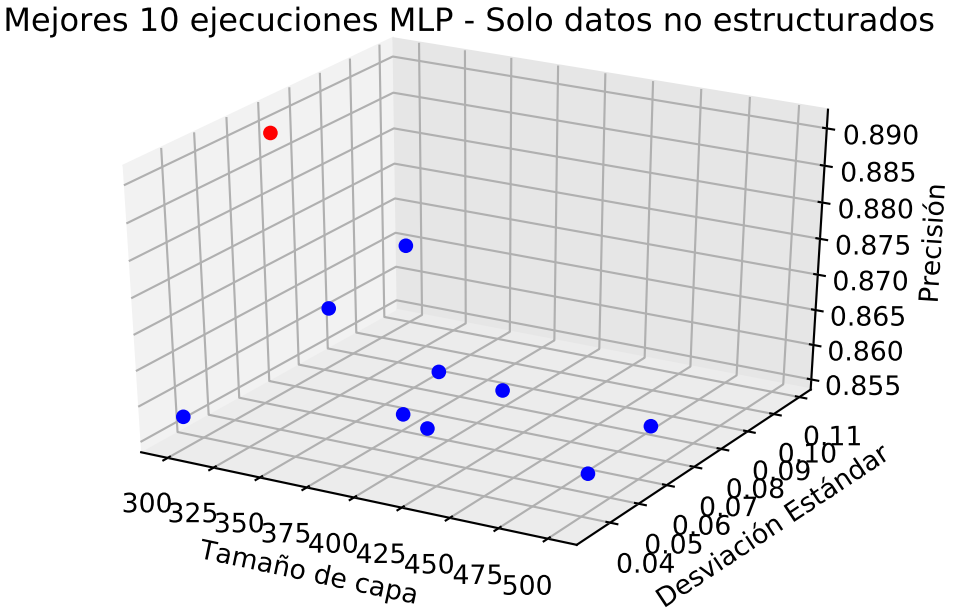



Figura 5.2: Gráfica de los 10 mejores casos de MLP entrenados en la configuración de datos no estructurados.

| Caso | Tasa de aprendizaje | # de estimadores | Precisión | $\sigma$ |
|---|---|---|---|---|
| 1 | 0.939 | 90 | **0.856** | 0.100 |
| 2 | 0.989 | 81 | 0.849 | 0.102 |
| 3 | 0.939 | 93 | 0.845 | 0.105 |
| 4 | 0.939 | 76 | 0.844 | 0.093 |
| 5 | 0.979 | 99 | 0.843 | 0.095 |
| 6 | 0.899 | 90 | 0.841 | 0.151 |
| 7 | 0.589 | 76 | 0.840 | 0.092 |
| 8 | 0.979 | 72 | 0.839 | 0.068 |
| 9 | 0.589 | 78 | 0.838 | 0.073 |
| 10 | 0.899 | 53 | 0.837 | 0.135 |

Tabla 5.9: Mejores 10 configuraciones de AdaBoost para clasificación de pacientes con cáncer utilizando datos no estructurados

| | Precisión | Recall | $F_1$-score |
|---|---|---|---|
| Hígado | 0.58 | 0.85 | 0.69 |
| Pecho | 0.91 | 0.71 | 0.80 |
| Pulmón | 0.73 | 0.61 | 0.67 |
| Average | 0.74 | 0.72 | 0.72 |

Tabla 5.10: Puntajes obtenidos por el mejor modelo de AdaBoost entrenado con datos no estructurados

Por último, para el modelo de *AdaBoost*, la Tabla 5.9 presenta los 10 mejores casos del modelo, así como los hiper-parámetros asociados. Se puede observar que el mejor modelo obtuvo una precisión del 85.6 %, siendo la mas baja entre los modelos entrenados con datos no estructurados. Se puede observar que el algoritmo de *boosting* preferido fue *SAMME*, también considerando una alta tasa de aprendizaje, por arriba de 0.5.

El reporte detallado de clasificación se presenta en la Tabla 5.10 y su gráfica correspondiente (Figura 5.3). Se observa que este modelo tiene una buena precisión para la clasificación del cáncer de pecho (91 %), aunque para los otros tipos de cáncer muestra un desempeño deficiente.

Podemos concluir que para esta configuración el modelo que menor desempeño mostró fue *AdaBoost*, que si bien mostró un buen desempeño en precisión en la clasificación del cáncer de pecho, esta fue menor que los otros dos modelos, además de que en los otros dos tipos de cáncer el desempeño fue muy bajo, mostrando una precisión apenas por arriba del 70 %. En cambio los modelos de SVM y MLP mostraron mejores resultados en promedio. En el caso de MLP muestra su mejor desempeño en precisión para la clasificación de cáncer de hígado y pulmón, siendo estas métricas las más altas para este tipo de cáncer en esta configuración. Sin embargo, el desempeño en precisión para el cáncer de pecho se ve disminuido en gran manera, aunque en este caso la métrica de *recall* muestra su valor mas alto. Para el caso de SVM se muestra un desempeño mas constante a través de los tres tipos de cáncer, manteniendo una precisión por arriba del 80 % en los tres casos, mostrando el mejor desempeño para la clasificación del cáncer de pecho. También se muestra que en el caso de cáncer de hígado, la métrica de recuperación muestra su valor mas alto entre los tres modelos, obteniendo un

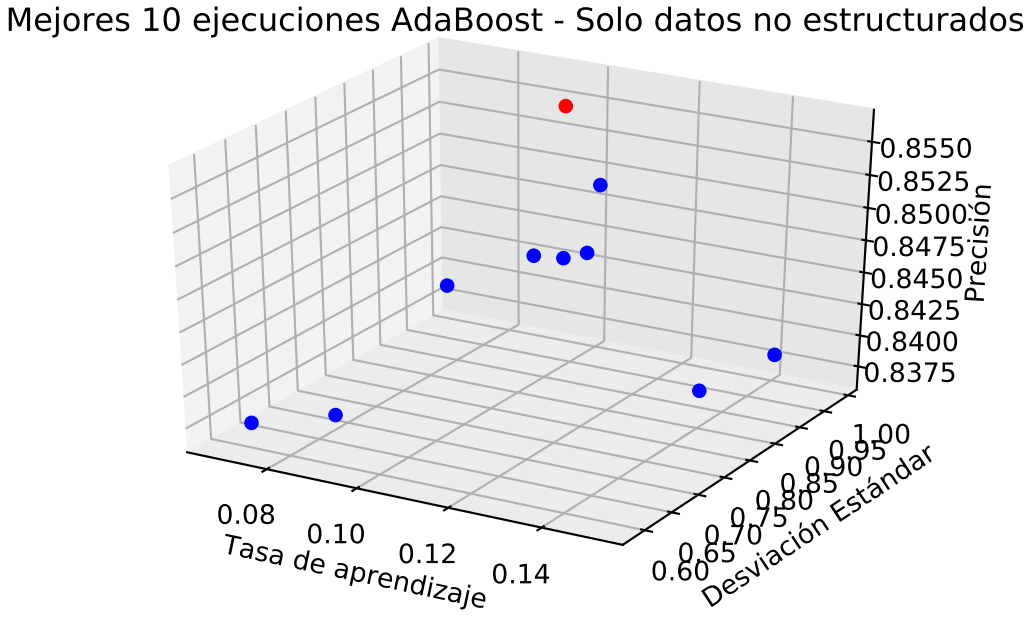



Figura 5.3: Gráfica de los 10 mejores casos de AdaBoost entrenados en la configuración de datos no estructurados.

| Caso # | Valor C | Valor $\gamma$ | Precisión | $\sigma$ |
|---|---|---|---|---|
| 1 | 300 | 0.2 | **0.575** | 0.146 |
| 2 | 1 | 0.8 | 0.557 | 0.158 |
| 3 | 1 | 0.9 | 0.554 | 0.135 |
| 4 | 1 | 0.6 | 0.553 | 0.174 |
| 5 | 1 | 0.4 | 0.540 | 0.221 |
| 6 | 100 | 0.9 | 0.536 | 0.185 |
| 7 | 1 | 0.3 | 0.533 | 0.260 |
| 8 | 1 | 0.7 | 0.532 | 0.143 |
| 9 | 100 | 0.5 | 0.525 | 0.128 |
| 10 | 1 | 0.5 | 0.524 | 0.184 |

Tabla 5.11: Mejores 10 configuraciones de SVM para clasificación de pacientes con cáncer utilizando datos estructurados

| | Precisión | Recall | $F_1$-Score |
|---|---|---|---|
| Pecho | 0.70 | 0.74 | 0.72 |
| Hígado | 0.56 | 0.38 | 0.45 |
| Pulmón | 0.44 | 0.54 | 0.48 |
| Promedio | 0.56 | 0.55 | 0.55 |

Tabla 5.12: Puntajes obtenidos por el mejor modelo de SVM entrenado con datos estructurados

valor de 1.0.

### 5.3.2. Configuración solo con datos estructurados

Esta configuración solo considera los valores procesados de los datos estructurados de los pacientes. Estos valores son los mostrados en la Tabla 4.1.

Los 10 mejores casos para SVM bajo esta configuración se muestran en la Tabla 5.11 y su correspondiente gráfica (5.4). Como se puede observar, la precisión de los modelos bajo esta configuración se mantiene ligeramente sobre el 50 %, donde la mejor configuración presenta un desempeño del 57.5 % en la precisión. También se puede observar que los valores preferidos de C son bajos, considerando que en la mayoría de los casos el valor es 1, que contrasta con los altos valores de C presentes al entrenar este modelo con datos no estructurados. Sin embargo, también cabe mencionar que para esta configuración de igual manera el mejor modelo tiene un valor alto de C (300). El reporte detallado de clasificación para el mejor modelo SVM para esta configuración es presentado en la Tabla 5.12.

En este reporte se puede observar que el desempeño en las tres métricas disminuyó considerablemente en comparación con la configuración utilizando solo datos no estructurados. Sin embargo, se muestra que el cáncer de pecho mostró el mejor desempeño en las tres métricas teniendo un 70 % de precisión, donde en contraste con la configuración de datos estructurados, también muestra que este tipo de cáncer es el que muestra mejor desempeño en la precisión.

Para el modelo MLP, la Tabla 5.13 y su gráfica correspondiente (Figura 5.5) muestran los

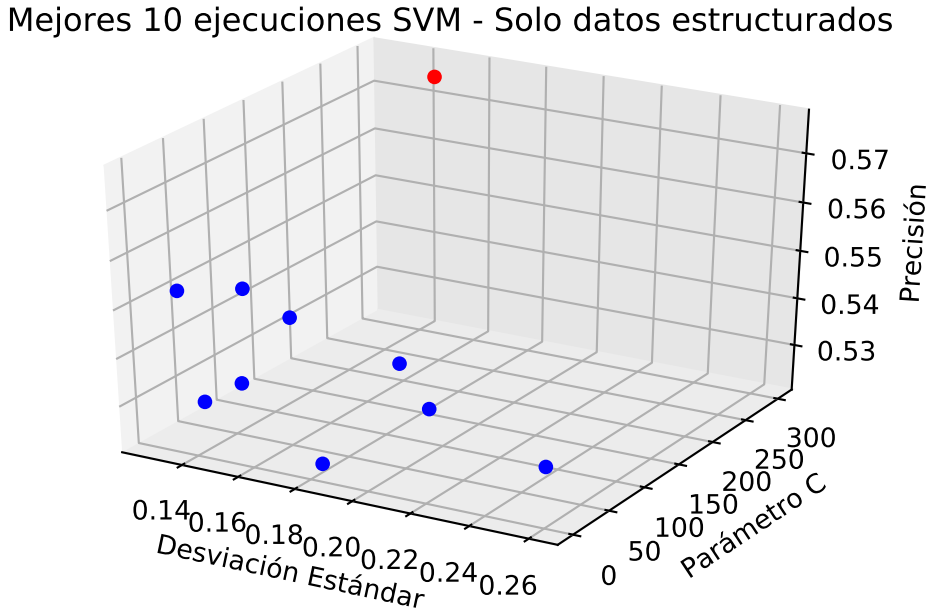



Figura 5.4: Gráfica de los 10 mejores casos de SVM entrenados en la configuración solo con datos estructurados.

| Caso # | Función de activación | Tamaño de capa escondida | Iter. Max. | Precisión | $\sigma$ |
|---|---|---|---|---|---|
| 1 | Identity | 300 | 500 | **0.608** | 0.178 |
| 2 | Identity | 400 | 400 | 0.603 | 0.171 |
| 3 | Tanh | 500 | 700 | 0.598 | 0.219 |
| 4 | Relu | 400 | 400 | 0.597 | 0.266 |
| 5 | Identity | 300 | 100 | 0.592 | 0.257 |
| 6 | Identity | 300 | 600 | 0.589 | 0.197 |
| 7 | Relu | 300 | 500 | 0.588 | 0.188 |
| 8 | Identity | 400 | 900 | 0.587 | 0.175 |
| 9 | Identity | 500 | 400 | 0.585 | 0.209 |
| 10 | Identity | 300 | 800 | 0.583 | 0.222 |

Tabla 5.13: Mejores 10 configuraciones de MLP para clasificación de pacientes con cáncer utilizando datos estructurados

| | Precisión | Recall | $F_1$-score |
|---|---|---|---|
| Hígado | 0.65 | 0.65 | 0.65 |
| Pecho | 0.44 | 0.67 | 0.53 |
| Pulmón | 0.60 | 0.38 | 0.46 |
| Promedio | 0.56 | 0.56 | 0.56 |

Tabla 5.14: Puntajes obtenidos por el mejor modelo de MLP entrenado con datos estructurados

10 mejores casos con sus hipe-parámetros asociados. Se puede observar que el mejor caso gana en desempeño promedio de precisión al modelo de SVM, obteniendo un 60 % de precisión. También es posible mostrar que para este tipo de datos, la función de activación preferida es *Identity*, contrastando con los modelos entrenados solo con datos no estructurados, cuya función de activación preferida fue *Relu*. En el caso del tamaño de capa este valor se quedó cercano a 300, aunque el número de valores de entrada no corresponden a ese número en este caso. El reporte detallado de clasificación se presenta en la Tabla 5.14.

Este reporte muestra como el modelo MLP mantiene valores de precisión más estables a través de la métrica de precisión. Sin embargo, se puede observar una disminución considerable en el desempeño en comparación con el modelo entrenado con datos no estructurados. De igual manera, se puede observar que los tipos de cáncer que mostraron mejor desempeño en precisión fueron el de hígado y pulmón, manteniendo esta cualidad con el modelo entrenado con datos no estructurados.

Para el modelo de AdaBoost bajo esta configuración, la Tabla 5.15 y su gráfica correspondiente (Figura 5.6) presentan los 10 mejores casos del modelo. Como se puede observar en esta tabla, la precisión promedio mas alta fue del 67 %, siendo esta la más alta en esta configuración en comparación con los otros modelos. En contraste con el modelo entrenado con datos no estructurados, en este caso se prefiere una tasa de aprendizaje baja, manteniéndose por debajo de 0.3. El reporte detallado del mejor caso se muestra en la Tabla 5.16.

En este reporte se muestra que los valores de precisión disminuyeron en comparación

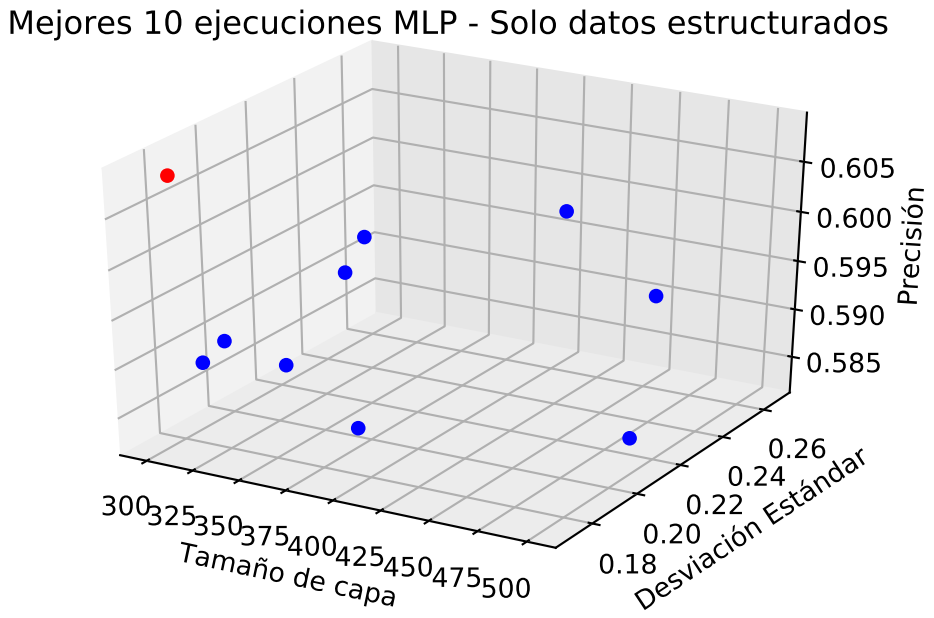



Figura 5.5: Gráfica de los 10 mejores casos de MLP entrenados en la configuración solo con datos estructurados.

| Caso | Tasa de aprendizaje | # de estimadores | Precisión | $\sigma$ |
|---|---|---|---|---|
| 1 | 0.25 | 65 | **0.677** | 0.202 |
| 2 | 0.229 | 53 | 0.671 | 0.239 |
| 3 | 0.259 | 59 | 0.670 | 0.193 |
| 4 | 0.279 | 51 | 0.669 | 0.239 |
| 5 | 0.199 | 61 | 0.668 | 0.237 |
| 6 | 0.189 | 70 | 0.667 | 0.236 |
| 7 | 0.289 | 56 | 0.665 | 0.185 |
| 8 | 0.169 | 81 | 0.664 | 0.234 |
| 9 | 0.299 | 54 | 0.663 | 0.190 |
| 10 | 0.130 | 87 | 0.661 | 0.244 |

Tabla 5.15: Mejores 10 configuraciones de AdaBoost para clasificación de pacientes con cáncer utilizando datos estructurados

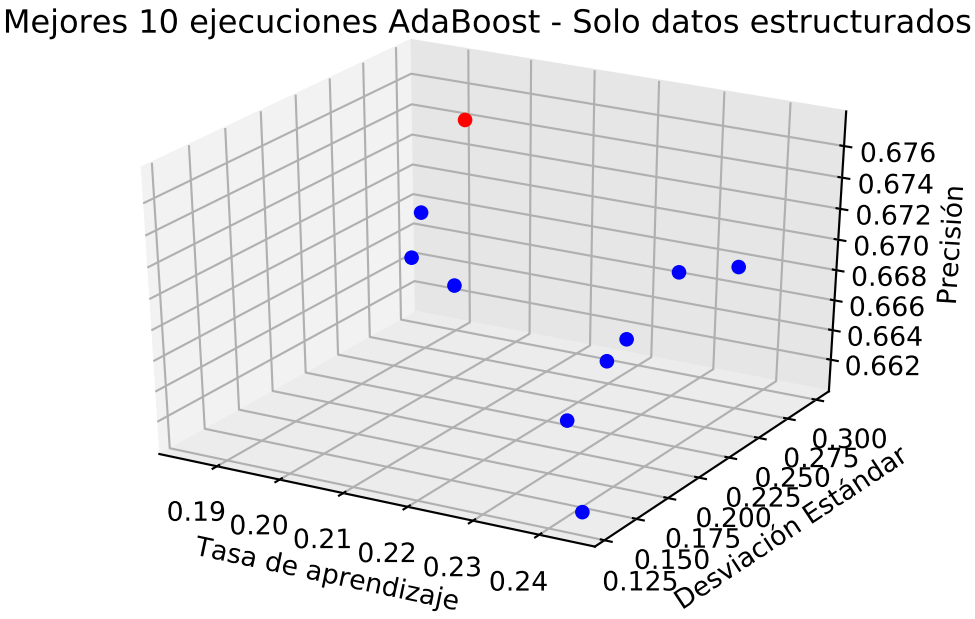



Figura 5.6: Gráfica de los 10 mejores casos de AdaBoost entrenados en la configuración solo con datos estructurados.

| | Precisión | Recall | $F_1$-score |
|---|---|---|---|
| Hígado | 0.46 | 0.46 | 0.46 |
| Pecho | 0.63 | 0.67 | 0.65 |
| Pulmón | 0.46 | 0.43 | 0.44 |
| Promedio | 0.52 | 0.52 | 0.52 |

Tabla 5.16: Puntajes obtenidos por el mejor modelo de AdaBoost entrenado con datos estructurados

| Caso # | C Valor | Valor $\gamma$ | Precisión | $\sigma$ |
|---|---|---|---|---|
| 1 | 100 | 0.1 | **0.84** | 0.069 |
| 2 | 1 | 0.1 | 0.713 | 0.164 |
| 3 | 100 | 0.2 | 0.703 | 0.135 |
| 4 | 1 | 0.2 | 0.701 | 0.170 |
| 5 | 100 | 0.3 | 0.695 | 0.142 |
| 6 | 1 | 0.5 | 0.690 | 0.159 |
| 7 | 1 | 0.3 | 0.680 | 0.168 |
| 8 | 1 | 0.4 | 0.679 | 0.150 |
| 9 | 100 | 0.4 | 0.671 | 0.168 |
| 10 | 100 | 0.6 | 0.667 | 0.157 |

Tabla 5.17: Mejores 10 configuraciones de SVM para clasificación de pacientes con cáncer utilizando datos estructurados y no estructurados

con el modelo entrenado con datos no estructurados. Sin embargo, también se muestra que el cáncer que presentó el mejor desempeño en la clasificación fue el cáncer de pecho, al igual que el modelo entrenado con datos no estructurados.

Para esta configuración los tres modelos considerados tuvieron una disminución considerable en su desempeño. Sin embargo, considerando los resultados obtenidos en la configuración con datos no estructurados, podemos observar cierto patrón en la clasificación. En el caso del modelo SVM y AdaBoost presenta el mejor desempeño para el cáncer de pecho. Aunque bajo la configuración con datos no estructurados el desempeño disminuyo, la característica de que dichos modelos muestran el mejor desempeño para el cáncer de pecho se mantiene. De igual manera, esta característica se mantiene para el modelo MLP para el cáncer de pulmón e hígado.

### 5.3.3. Configuración con datos estructurados y no estructurados

Esta configuración considera el entrenamiento de los modelos de ML utilizando ambos tipos de datos. Estos datos se unen en un archivo en formato csv y se consideran como valores de entrada para los algoritmos.

Los 10 mejores casos para el modelo SVM bajo esta configuración se muestran en la Tabla 5.17 y en su gráfica correspondiente (Figura 5.7). Se puede observar que la mejor precisión promedio fue del 84 %, mostrando una diferencia considerable al caso 10, siendo esta diferencia de 0.73, mostrando contraste con la pequeña diferencia que se obtuvo en el modelo entrenado con datos estructurados. También se puede observar que los hiperparámetros prefieren valores pequeños de gama, y valores pequeños de C, en comparación con el modelo entrenado con datos no estructurados. El reporte detallado de clasificación para el mejor modelo de SVM se muestra en la Tabla 5.18. En esta tabla se observa que los valores de precisión para el cáncer de pecho e hígado aumentaron; sin embargo, hubo una caída considerable en el desempeño del cáncer de pulmón. Para la métrica de recuperación, también se observa una considerable disminución en el desempeño para el cáncer de pecho y una disminución más ligera para el cáncer de hígado. Por el contrario, esta configuración mostró una mejora en esta métrica para el cáncer de pulmón.

Para el modelo MLP, los 10 mejores resultados bajo esta configuración se presentan en

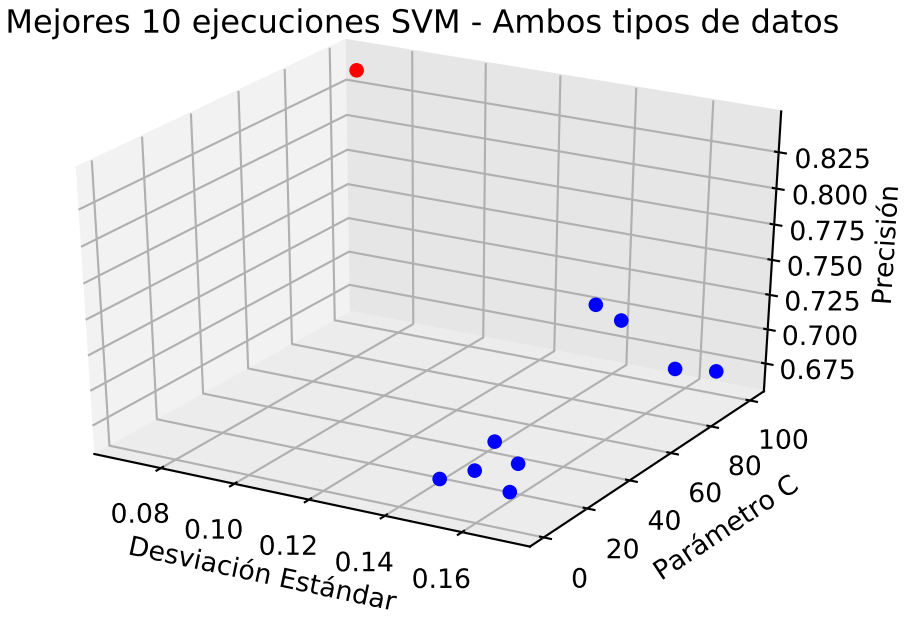



Figura 5.7: Gráfica de los 10 mejores casos de SVM entrenados en la configuración con ambos tipos de datos (estructurados y no estructurados).

| | Precisión | Recall | $F_1$-Score |
|---|---|---|---|
| Pecho | 0.93 | 0.68 | 0.79 |
| Hígado | 0.92 | 0.92 | 0.92 |
| Pulmón | 0.69 | 0.92 | 0.77 |
| Promedio | 0.84 | 0.84 | 0.83 |

Tabla 5.18: Puntajes obtenidos por el mejor modelo de SVM entrenado con datos estructurados y no estructurados

| Caso # | Función de activación | Tamaño de capa escondida | Iter. Max. | Precisión | $\sigma$ |
|---|---|---|---|---|---|
| 1 | Logistic | 400 | 700 | **0.849** | 0.150 |
| 2 | Tanh | 500 | 900 | 0.846 | 0.160 |
| 3 | Identity | 300 | 800 | 0.845 | 0.146 |
| 4 | Logistic | 500 | 500 | 0.844 | 0.147 |
| 5 | Identity | 400 | 100 | 0.841 | 0.148 |
| 6 | Identity | 300 | 200 | 0.840 | 0.149 |
| 7 | Logistic | 500 | 200 | 0.839 | 0.108 |
| 8 | Tanh | 400 | 700 | 0.838 | 0.140 |
| 9 | Identity | 300 | 400 | 0.837 | 0.161 |
| 10 | Identity | 300 | 700 | 0.836 | 0.139 |

Tabla 5.19: Mejores 10 configuraciones de MLP para clasificación de pacientes con cáncer utilizando datos estructurados y no estructurados

| | Precision | Recall | $F_1$-score |
|---|---|---|---|
| Hígado | 0.64 | 0.75 | 0.69 |
| Pecho | 0.94 | 0.89 | 0.92 |
| Pulmón | 0.77 | 0.71 | 0.74 |
| Promedio | 0.80 | 0.80 | 0.80 |

Tabla 5.20: Puntajes obtenidos por el mejor modelo de MLP entrenado con datos estructurados y no estructurados

la Tabla 5.19 y su correspondiente gráfica (Figura 5.8). En este caso, la mejor precisión para el modelo fue del 89.4 %, con poca diferencia entre el mejor y peor caso presentado, siendo esta una diferencia de 0.013. Se muestra que la función de activación preferida por el modelo fue *Identity*; sin embargo, se muestra que en los dos mejores casos la función preferida fue *Logistic* y *Tanh*, respectivamente. La función de activación contrasta con la configuración del modelo con datos no estructurados, en donde se muestra que la función de activación preferida por el modelo fue *Relu*. El reporte detallado de clasificación para el mejor modelo MLP se presenta en la Tabla 5.20. Este reporte muestra como el desempeño de la precisión en la clasificación del cáncer de pecho aumenta considerablemente, incluso siendo este el mejor desempeño mostrado para esta métrica con un 94 % de precisión. Sin embargo, contrario a las dos configuraciones anteriores, el desempeño en la clasificación en los otros dos tipos de cáncer sufrió una baja de desempeño considerable, incluso más que la ganancia en precisión para el cáncer de pecho.

Los resultados para el modelo AdaBoost considerando esta configuración se presentan en la Tabla 5.21 y su gráfica correspondiente (Figura 5.9). Como se observa en la tabla, la mejor precisión encontrada para el modelo fue de 85.2 %, siendo un gran aumento en consideración del modelo entrenado solo con datos estructurados, y manteniéndose con un valor similar comparando con el modelo entrenado con datos no estructurados. Para el caso del parámetro de tasa de aprendizaje, se puede observar que los valores se mantienen entre 0.2 y 0.6, sin llegar a considerar valores tan altos como el modelo entrenado con datos no estructurados. Incluso se puede observar que los valores de la tasa de aprendizaje se mantienen similares a

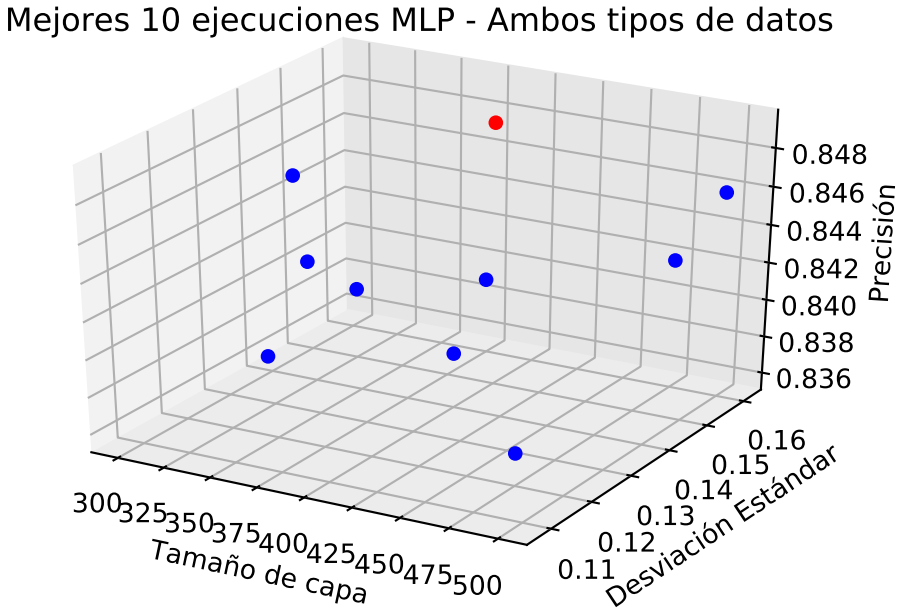



Figura 5.8: Gráfica de los 10 mejores casos de MLP entrenados en la configuración con ambos tipos de datos (estructurados y no estructurados).

| Caso | Tasa de aprendizaje | # de estimadores | Precisión | $\sigma$ |
|---|---|---|---|---|
| 1 | 0.369 | 55 | **0.852** | 0.096 |
| 2 | 0.279 | 72 | 0.850 | 0.118 |
| 3 | 0.569 | 78 | 0.849 | 0.107 |
| 4 | 0.589 | 69 | 0.848 | 0.144 |
| 5 | 0.499 | 99 | 0.847 | 0.100 |
| 6 | 0.329 | 54 | 0.846 | 0.131 |
| 7 | 0.449 | 61 | 0.845 | 0.089 |
| 8 | 0.269 | 55 | 0.844 | 0.057 |
| 9 | 0.259 | 72 | 0.843 | 0.105 |
| 10 | 0.259 | 64 | 0.842 | 0.099 |

Tabla 5.21: Mejores 10 configuraciones de AdaBoost para clasificación de pacientes con cáncer utilizando datos estructurados y no estructurados

| | Precisión | Recall | $F_1$-score |
|---|---|---|---|
| Hígado | 1.0 | 0.83 | 0.91 |
| Pecho | 0.80 | 0.75 | 0.77 |
| Pulmón | 0.70 | 0.82 | 0.76 |
| Promedio | 0.83 | 0.80 | 0.81 |

Tabla 5.22: Puntajes obtenidos por el mejor modelo de AdaBoost entrenado con datos estructurados y no estructurados

los presentados en el entrenamiento con datos estructurados. De igual manera, al considerar el número de estimadores utilizados, en esta configuración no se tiene un número alto, siendo 55 para el mejor caso. Comparando con los modelos de configuraciones anteriores, el número de estimadores para la configuración de datos estructurados consideraba valores altos, cercanos a 100. El reporte detallado de clasificación para el mejor modelo de AdaBoost para esta configuración es presentado en la Tabla 5.22. En este reporte se puede observar que esta configuración permitió que el modelo AdaBoost obtuviera un considerable aumento en la precisión del cáncer de hígado, obteniendo el mejor desempeño en esta métrica a través de todas las configuraciones, con un 100 % de precisión. En comparación con la configuración de datos estructurados, el aumento en el desempeño de las métricas se realizó de manera general para los tres tipos de cáncer. Para el caso de la configuración con datos no estructurados, se observa un decremento en el desempeño de la precisión para el cáncer de pecho y pulmón; sin embargo, la ganancia presentada para el cáncer de hígado sobrepasa la pérdida de la precisión de los otros dos tipos de cáncer.

Para concluir, en general para esta configuración, se reflejó un aumento en el desempeño de las métricas para un tipo de cáncer en particular. Los modelos MLP y SVM muestran un aumento considerable en un tipo de cáncer, que en otras configuraciones se mostraba como el de menor desempeño. Sin embargo, derivado de este aumento también presentan pérdidas de desempeño en las métricas de los tipos de cáncer restantes. Un caso particular fue AdaBoost, que en comparación con la configuración estructurada, el aumento a través de todas sus métricas se hizo muy notable, incluso obteniendo la mejor métrica de precisión para la clasificación de cáncer de hígado. De igual manera, teniendo pérdida de desempeño

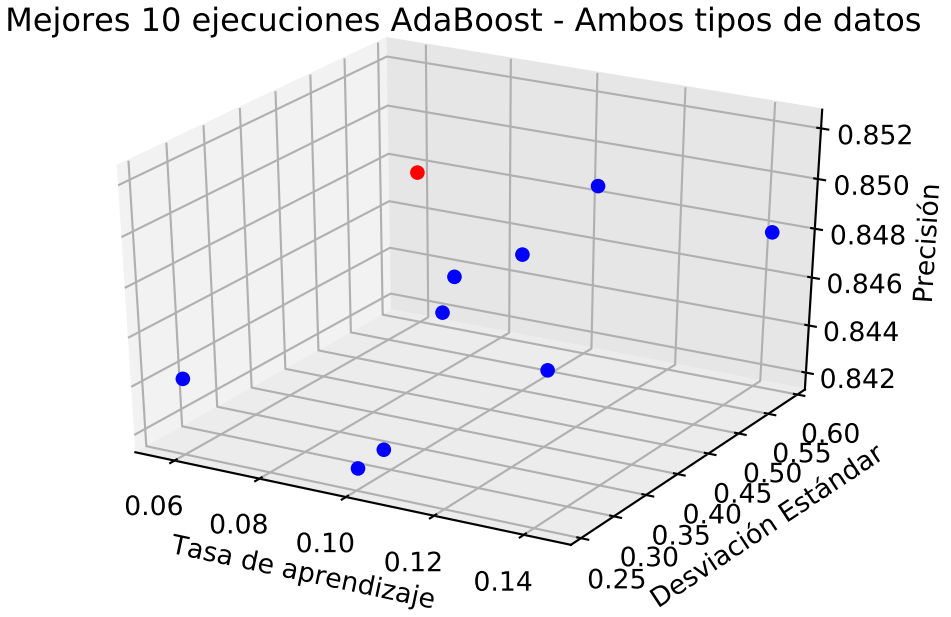



Figura 5.9: Gráfica de los 10 mejores casos de AdaBoost entrenados en la configuración con ambos tipos de datos (estructurados y no estructurados).

en los otros tipos de cáncer, pero dicha pérdida no siendo tan pronunciada como el caso de los modelos SVM y MLP. Se puede considerar que la integración de ambos tipos de datos presentan un aumento considerable en las métricas de los tipos de cáncer que anteriormente presentaban el menor desempeño. Sin embargo, como resultado, se tiene una pérdida considerable en el desempeño de la clasificación de los tipos de cáncer restantes.

# Capítulo 6

# Conclusiones y trabajo a futuro

Esta tesis presenta una metodología para la clasificación de pacientes con cáncer de pecho, pulmón e hígado, considerando la inclusión tanto de datos estructurados como no estructurados. Dicha metodología consideró técnicas de PLN para la extracción de conocimiento a partir de notas médicas, escritas como texto libre. También se incluyeron en el análisis los datos de estudios de laboratorio y demográficos que son realizados de manera rutinaria a pacientes que ingresan a los hospitales.

Los resultados mostraron que la precisión en la clasificación muestra valores más estables cuando se entrenan los modelos utilizando datos no estructurados. En particular, el modelo SVM resulta el mejor clasificador de cáncer de pecho bajo esta configuración, así como el modelo MLP para los tipos de cáncer restantes.

De igual manera se observó que utilizar solamente datos estructurados para la clasificación muestra un deterioro en el desempeño de las métricas mostrando un desempeño en la precisión por debajo del 50 %. Esto podría ser derivado que estos tipos de datos son muy genéricos. Los datos considerados son información de rutina recabada para los pacientes ingresados al hospital.

Por otro lado, al entrenar los modelos con ambos tipos de datos se observa un aumento en el desempeño de algunos tipos de cáncer. Incluso los mejores valores en el desempeño de precisión se obtuvieron mediante el entrenamiento de los modelos incluyendo datos estructurados y no estructurados. Siendo el modelo MLP para la clasificación de cáncer de pecho, obteniendo un 94 % de precisión, y *Adaboost* para el cáncer de hígado, obteniendo un 100 % de precisión. Sin embargo, este aumento en el desempeño de métricas para algunos tipos de cáncer, también presenta una disminución para los otros tipos de cáncer, incluso mostrando mayor disminución mayor que la ganancia en precisión obtenida.

Basado en la consistencia de los resultados y métricas de desempeño estables, se considera al modelo MLP como el que mejor desempeño muestra en la clasificación de tipo de cáncer, entrenado con datos no estructurados, mostrando una precisión promedio del 89 %.

Con base en los resultados obtenidos, la metodología propuesta presenta las siguientes características, considerando solamente su aplicación a la base de datos MIMIC-II[43]:

- Se cuenta con un módulo para el pre-procesamiento de notas médicas en forma de texto libre. A través de este proceso las notas son preparadas para que un módulo posterior pueda transformar dichas notas en una representación necesaria para ser utilizada por algoritmos de ML.

- El módulo de representación vectorial permite que las notas pre-procesadas sean utilizadas como valores de entrada al proceso. Esto para que cada nota sea transformada en una representación vectorial que está lista para utilizarse como valores de entrada en algoritmos de ML.
- Considerando el manejo de datos estructurados, se cuenta con el módulo de pre-procesamiento de este tipo de datos. De esta manera, permite que información estructurada de bases de datos clínicas (estudios de laboratorio, datos demográficos, etc.) sea utilizada en conjunto con la representación vectorial de las notas médicas como datos de entrada para algoritmos de ML.
- El módulo de clasificación permite que la información obtenida de procesos anteriores sea utilizada para clasificar a los pacientes. Este módulo incluye el uso de los algoritmos SVM, MLP y Adaboost.
- Gracias al proceso de evaluación de desempeño y mejora de modelo, fue posible obtener el modelo que mejor desempeño presentó en cada tipo de cáncer.

Como trabajo a futuro se pueden hacer mejoras desde las distintas etapas de la metodología propuesta.

- En la etapa de extracción de información, es posible implementar bases de datos con una estructura más óptima para el registro de datos, que puede ser adaptada para una generación continua de información. Además, se puede considerar una integración transparente con los sistemas actuales, para causar el mínimo impacto en la operación. Por otra parte, es posible realizar un pre-procesamiento previo al almacenamiento, para así disminuir la carga de trabajo posterior.
- En el área de Procesamiento de Lenguaje Natural, es posible integrar distintas técnicas para la representación de texto. Incluso, es posible utilizar diccionarios y sistemas de modelado de lenguaje orientadas específicamente al campo clínico, como es el caso de UMLS.
- En el área de ML es posible la integración de algoritmos de aprendizaje profundo (*deep learning*), como lo son las FFNN, para manejar grandes cantidades de datos, dado las características de la información generada. La integración de algoritmos de aprendizaje profundo a estas tareas puede suponer una mejora en el desempeño de la clasificación, considerando también una mayor variedad de datos de pacientes.

# Bibliografía


[1] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[2] W. Van Der Aalst, "Process mining: Overview and opportunities," *ACM Transactions on Management Information Systems (TMIS)*, vol. 3, no. 2, p. 7, 2012.

[3] M. K. Leung, A. Delong, B. Alipanahi, and B. J. Frey, "Machine learning in genomic medicine: a review of computational problems and data sets," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 176–197, 2016.

[4] S. Fodeh and Q. Zeng, "Mining big data in biomedicine and health care," *Journal of Biomedical Informatics*, vol. 63, pp. 400–403, 2016.

[5] S. N. Limited, "Health sciences," https://www.nature.com/subjects/health-sciences#:~:text=Definition,improve%20the%20treatment%20of%20patients, 2020, accesado: 15-06-2020.

[6] B. Röhrig, J. B. du Prel, D. Wachtlin, and M. Blettner, "Types of study in medical research: part 3 of a series on evaluation of scientific publications," *Dtsch Arztebl Int*, vol. 106, no. 15, pp. 262–8, 2009.

[7] K. Zhang and D. Demner-Fushman, "Automated classification of eligibility criteria in clinical trials to facilitate patient-trial matching for specific patient populations," *Journal of the American Medical Informatics Association*, vol. 24, no. 4, pp. 781–787, 2017.

[8] N. C. Institute, "Nci dictionary of cancer terms," https://www.cancer.gov/publications/dictionaries/cancer-terms/def/cancer, 2020, accesado: 15-04-2020.

[9] I. A. for Research on Cancer, "Global cancer observatory," https://gco.iarc.fr/tomorrow/home, 2020, accesado: 08-05-2020.

[10] P. B. Jensen, L. J. Jensen, and S. Brunak, "Mining electronic health records: towards better research applications and clinical care," *Nature Reviews Genetics*, vol. 13, no. 6, pp. 395–405, 2012.

[11] C. Lee, Z. Luo, K. Y. Ngiam, M. Zhang, K. Zheng, G. Chen, B. C. Ooi, and W. L. J. Yip, *Big healthcare data analytics: Challenges and applications*. Springer, 2017, pp. 11–41.

[12] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and structural biotechnology journal*, vol. 13, pp. 8–17, 2015.

[13] P. Um, *Cancer, Definition.* Boston, MA: Springer US, 2015, pp. 65–65. [Online]. Available: https://doi.org/10.1007/978-1-4899-7475-4_106

[14] E. Britannica, "Cancer incidence and mortality in the united states," https://bidi.uam.mx:6402/levels/collegiate/article/cancer/106118, 2020, accesado: 30-06-2020.

[15] F. Bray, J. Ferlay, I. Soerjomataram, R. L. Siegel, L. A. Torre, and A. Jemal, "Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries," *CA: a cancer journal for clinicians*, vol. 68, no. 6, pp. 394–424, 2018.

[16] B. Chabner and D. L. Longo, *Harrison's manual of oncology*, 2014.

[17] E. Britannica. (2020) Breast cancer. Accesado: 30-06-2020. [Online]. Available: https://bidi.uam.mx:6402/levels/collegiate/assembly/view/212227

[18] E. Britannica. (2020) Lung cancer. Accesado: 30-06-2020. [Online]. Available: https://bidi.uam.mx:6402/levels/collegiate/assembly/view/56767

[19] L. Deng and Y. Liu, *Deep Learning in Natural Language Processing.* Springer, 2018.

[20] C. C. Aggarwal, *Machine Learning for Text.* Cham: Springer International Publishing, 2018.

[21] J. B. Lovins, "Development of a stemming algorithm," *Mech. Translat. & Comp. Linguistics*, vol. 11, no. 1-2, pp. 22–31, 1968.

[22] C. D. Manning, C. D. Manning, and H. Schütze, *Foundations of statistical natural language processing.* MIT press, 1999.

[23] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[25] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, Conference Proceedings, pp. 1188–1196.

[26] T. Mitchell, *Machine learning*, 1st ed., ser. Mcgraw-Hill International Edit. McGraw Hill Higher Education, 1997.

[27] S. Marsland, *Machine learning: an algorithmic perspective.* Chapman and Hall/CRC, 2011.

[28] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms.* Wiley Publishing, 2001.

[29] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[30] E. Alpaydin, *Introduction to Machine Learning*. The MIT Press, 2010.

[31] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[32] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[33] M. Minsky and S. A. Papert, *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[34] Merriam-Webster, "Boost definition," https://www.merriam-webster.com/dictionary/boost, 2020, accesado: 30-06-2020.

[35] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[36] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *icml*, vol. 96. Citeseer, Conference Proceedings, pp. 148–156.

[37] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[38] L. Minor, "Stanford medicine 2020 health trends report," 2020. [Online]. Available: https://med.stanford.edu/dean/healthtrends.html

[39] C. Durinx, J. McEntyre, R. Appel, R. Apweiler, M. Barlow, N. Blomberg, C. Cook, E. Gasteiger, J.-H. Kim, and R. Lopez, "Identifying elixir core data resources," *F1000Research*, vol. 5, 2016.

[40] K. Tomczak, P. Czerwińska, and M. Wiznerowicz, "The cancer genome atlas (tcga): an immeasurable source of knowledge," *Contemporary oncology*, vol. 19, no. 1A, p. A68, 2015.

[41] A. E. Kennedy, M. J. Khoury, J. P. Ioannidis, M. Brotzman, A. Miller, C. Lane, G. Y. Lai, S. D. Rogers, C. Harvey, and J. W. Elena, "The cancer epidemiology descriptive cohort database: A tool to support population-based interdisciplinary research," *Cancer Epidemiology and Prevention Biomarkers*, vol. 25, no. 10, pp. 1392–1401, 2016.

[42] "Seer incidence database - seer data and software," 2020. [Online]. Available: https://seer.cancer.gov/data/

[43] M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, and R. G. Mark, "Multiparameter intelligent monitoring in intensive care ii: A public-access intensive care unit database," *Critical Care Medicine*, vol. 39, no. 5, pp. 952–960, 2011.

[44] G. Clifford, D. Scott, and M. Villarroel, "User guide and documentation for the mimic ii database," 01 2009.

[45] D. Delen, G. Walker, and A. Kadam, "Predicting breast cancer survivability: a comparison of three data mining methods," *Artificial intelligence in medicine*, vol. 34, no. 2, pp. 113–127, 2005.

[46] M. Montazeri, M. Montazeri, M. Montazeri, and A. Beigzadeh, "Machine learning models in breast cancer survival prediction," *Technology and health care : official journal of the European Society for Engineering and Medicine*, vol. 24, no. 1, pp. 31–42, 2016.

[47] L. Tapak, N. Shirmohammadi-Khorram, P. Amini, B. Alafchi, O. Hamidi, and J. Poorolajal, "Prediction of survival and metastasis in breast cancer patients using machine learning classifiers," *Clinical Epidemiology and Global Health*, 2018.

[48] N. A. Farooqui and Ritika, "A study on early prevention and detection of breast cancer using three-machine learning techniques," *International Journal of Advanced Research in Computer Science U6 - Journal Article*, vol. 9, no. Special Issue 2, p. 37, 2018.

[49] S. Sharma, A. Aggarwal, and T. Choudhury, "Breast cancer detection using machine learning algorithms," in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. IEEE, Conference Proceedings, pp. 114–118.

[50] N. Liu, E.-S. Qi, M. Xu, B. Gao, and G.-Q. Liu, "A novel intelligent classification model for breast cancer diagnosis," *Information Processing & Management*, vol. 56, no. 3, pp. 609–623, 2019.

[51] C. M. Lynch, B. Abdollahi, J. D. Fuqua, A. R. de Carlo, J. A. Bartholomai, R. N. Balgemann, V. H. van Berkel, and H. B. Frieboes, "Prediction of lung cancer patient survival via supervised machine learning classification techniques," *International Journal of Medical Informatics*, vol. 108, pp. 1–8, 2017.

[52] M. I. Faisal, S. Bashir, Z. S. Khan, and F. H. Khan, "An evaluation of machine learning classifiers and ensembles for early stage prediction of lung cancer," in *2018 3rd International Conference on Emerging Trends in Engineering, Sciences and Technology (ICEEST)*. IEEE, Conference Proceedings, pp. 1–4.

[53] N. K. Kumar, D. Vigneswari, M. Kavya, K. Ramya, and T. L. Druthi, "Predicting non-small cell lung cancer: A machine learning paradigm," *Journal of Computational and Theoretical Nanoscience*, vol. 15, no. 6-7, pp. 2055–2058, 2018.

[54] T. M. Deist, F. J. Dankers, G. Valdes, R. Wijsman, I. Hsu, C. Oberije, T. Lustberg, J. van Soest, F. Hoebers, and A. Jochems, "Machine learning algorithms for outcome prediction in (chemo) radiotherapy: An empirical comparison of classifiers," *Medical physics*, vol. 45, no. 7, pp. 3449–3459, 2018.

[55] J. A. Bartholomai and H. B. Frieboes, "Lung cancer survival prediction via machine learning regression, classification, and statistical techniques," in *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, Conference Proceedings, pp. 632–637.

[56] K. Pradeep and N. Naveen, "Lung cancer survivability prediction based on performance using classification techniques of support vector machines, c4. 5 and naive bayes algorithms for healthcare analytics," *Procedia computer science*, vol. 132, pp. 412–420, 2018.

[57] W. Ksiazek, M. Abdar, U. R. Acharya, and P. Plawiak, "A novel machine learning approach for early detection of hepatocellular carcinoma patients," *Cognitive Systems Research*, vol. 54, pp. 116–127, 2019.

[58] M. Sato, K. Morimoto, S. Kajihara, R. Tateishi, S. Shiina, K. Koike, and Y. Yatomi, "Machine-learning approach for the development of a novel predictive model for the diagnosis of hepatocellular carcinoma," *Scientific reports*, vol. 9, no. 1, pp. 1–7, 2019.

[59] M. B. Priya, P. L. Juliet, and P. Tamilselvi, "Performance analysis of liver disease prediction using machine learning algorithms," *International Research Journal of Engineering and Technology (IRJET)*, vol. 5, no. 1, pp. 206–211, 2018.

[60] A. S. Rahman, F. J. M. Shamrat, Z. Tasnim, J. Roy, and S. A. Hossain, "A comparative study on liver disease prediction using supervised machine learning algorithms," *International Journal of Scientific & Technology Research*, vol. 8, no. 11, pp. 419–422, 2019.

[61] K. AlicjaSala, "Comparison of machine learning algorithms for prediction mortality in patients with hepatocellular carcinoma," *International Journal of Scientific Engineering and Technology*, vol. 7, no. 9, pp. 85–89.

[62] E. H. Abdelaziz, S. M. Kamal, K. El-Bhanasy, and R. Ismail, "The application of data mining techniques and feature selection methods in the risk classification of egyptian liver cancer patients using clinical and genetic data," in *Proceedings of the 2019 8th International Conference on Software and Information Engineering*, Conference Proceedings, pp. 200–205.

[63] S. Bozkurt, F. Gimenez, E. S. Burnside, K. H. Gulkesen, and D. L. Rubin, "Using automatically extracted information from mammography reports for decision-support," *Journal of Biomedical Informatics*, vol. 62, pp. 224–231, 2016.

[64] T. A. Patel, M. Puppala, R. O. Ogunti, J. E. Ensor, T. He, J. B. Shewale, D. P. Ankerst, V. G. Kaklamani, A. A. Rodriguez, S. T. C. Wong, and J. C. Chang, "Correlating mammographic and pathologic findings in clinical decision support using natural language processing and data mining methods: Natural language processing," *Cancer*, vol. 123, no. 1, pp. 114–121, 2017.

[65] A. K. AAlAbdulsalam, J. H. Garvin, A. Redd, M. E. Carter, C. Sweeny, and S. M. Meystre, "Automated extraction and classification of cancer stage mentions fromunstructured text fields in a central cancer registry," *AMIA Summits on Translational Science Proceedings*, vol. 2018, p. 16, 2018.

[66] A. W. Forsyth, R. Barzilay, K. S. Hughes, D. Lui, K. A. Lorenz, A. Enzinger, J. A. Tulsky, and C. Lindvall, "Machine learning methods to extract documentation of breast cancer symptoms from electronic health records," *Journal of pain and symptom management*, vol. 55, no. 6, pp. 1492–1499, 2018.

[67] A. E. Gerevini, A. Lavelli, A. Maffi, R. Maroldi, A.-L. Minard, I. Serina, and G. Squassina, "Automatic classification of radiological reports for clinical care," *Artificial intelligence in medicine*, vol. 91, pp. 72–81, 2018.

[68] B. Koopman, G. Zuccon, A. Nguyen, A. Bergheim, and N. Grayson, "Extracting cancer mortality statistics from death certificates: A hybrid machine learning and rule-based approach for common and rare cancers," *Artificial Intelligence In Medicine*, vol. 89, pp. 1–9, 2018.

[69] Z. Zeng, S. Espino, A. Roy, X. Li, S. A. Khan, S. E. Clare, X. Jiang, R. Neapolitan, and Y. Luo, "Using natural language processing and machine learning to identify breast cancer local recurrence," *BMC bioinformatics*, vol. 19, no. 17, pp. 65–74, 2018.

[70] Z. Zeng, L. Yao, A. Roy, X. Li, S. Espino, S. E. Clare, S. A. Khan, and Y. Luo, "Identifying breast cancer distant recurrences from electronic health records using machine learning," *Journal of Healthcare Informatics Research*, vol. 3, no. 3, pp. 283–299, 2019.

[71] P.-H. Chen, H. Zafar, M. Galperin-Aizenberg, and T. Cook, "Integrating natural language processing and machine learning algorithms to categorize oncologic response in radiology reports," *Journal of digital imaging*, vol. 31, no. 2, pp. 178–184, 2018.

[72] Y. Si and K. Roberts, "A frame-based nlp system for cancer-related information extraction," in *AMIA Annual Symposium Proceedings*, vol. 2018. American Medical Informatics Association, Conference Proceedings, p. 1524.

[73] B. Udelsman, I. Chien, K. Ouchi, K. Brizzi, J. A. Tulsky, and C. Lindvall, "Needle in a haystack: natural language processing to identify serious illness," *Journal of palliative medicine*, vol. 22, no. 2, pp. 179–182, 2019.

[74] R. Lou, D. Lalevic, C. Chambers, H. M. Zafar, and T. S. Cook, "Automated detection of radiology reports that require follow-up imaging using natural language processing feature engineering and machine learning classification," *Journal of digital imaging*, pp. 1–6, 2019.

[75] E. K. Gupta, R. Thammasudjarit, and A. Thakkinstian, "Nlp automation to read radiological reports to detect the stage of cancer among lung cancer patients," in *Proceedings of the 2019 Workshop on Widening NLP*, Conference Proceedings, pp. 138–141.

[76] M. G. Seneviratne, J. M. Banda, J. D. Brooks, N. H. Shah, and T. M. Hernandez-Boussard, "Identifying cases of metastatic prostate cancer using machine learning on electronic health records," in *AMIA Annual Symposium Proceedings*, vol. 2018. American Medical Informatics Association, Conference Proceedings, p. 1498.

[77] A. Brown and J. Kachura, "Natural language processing of radiology reports in patients with hepatocellular carcinoma to predict radiology resource utilization," *Journal of the American College of Radiology*, vol. 16, no. 6, pp. 840–844, 2019.

[78] C. for Disease Control and P. (CDC). (2009) International classification of diseases,ninth revision (icd-9). Accesado: 15-07-2020. [Online]. Available: https://www.cdc.gov/nchs/icd/icd9.htm

[79] "English stopwords." [Online]. Available: https://www.ranks.nl/stopwords

[80] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.

[81] E. D. D. Team, "DL4J: Deep Learning for Java," 2016. [Online]. Available: https://github.com/eclipse/deeplearning4j

[82] A. S. Foundation, "Commons CSV," 2020. [Online]. Available: https://commons.apache.org/proper/commons-csv/

[83] P. G. D. Group, "PostgreSQL," 2020. [Online]. Available: https://www.postgresql.org/

[84] P. S. Foundation, "Python," 2020. [Online]. Available: https://www.python.org/

[85] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[86] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[87] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14. Montreal, Canada, Conference Proceedings, pp. 1137–1145.

# Apéndice A

# Carta de aceptación a revista indexada

Computación y Sistemas

AN INTERNATIONAL JOURNAL OF COMPUTING SCIENCE AND APPLICATIONS

ISSN 1405-5546 (print)
ISSN 2007-9737 (electronic)

Apartado Postal 75-546
C.P. 07738 México, D.F.
Tel (+52)-55-5729-6000
Ext. 56518, 56643
Fax Ext. 56607
computacion-y-sistemas@cic.ipn.mx

http://cys.cic.ipn.mx

Dublin, Ireland, September 9th, 2019

## ACCEPTANCE CERTIFICATE

This is to confirm that

*Erick E. Montelongo-González, José A. Reyes-Ortiz, Beatriz A. González-Beltrán*

The paper number 01, entitled "**Machine Learning Models for Cancer Type Classification with Unstructured Data**" has been selected for to be published in Computación y Sistemas (CyS) Special Issue on Language & Knowledge Engineering

We need from you the confirmation that you accept the following terms:

1. You agree your paper to be published in the above-mentioned journal (CyS)
2. The corresponding registration fee will be paid by you or one of the paper authors
3. You agree to take into account the reviewers comments and modify your paper accordingly
4. You agree to format the paper following "entirely" the author guidelines required by CyS, providing source files of your paper
5. The maximum number of pages of your paper is 12

In case, any of the above-mentioned conditions can't be answered positive, we can't guarantee the publication of your paper.

Congratulations

Best regards,
**CyS Editor**

# Apéndice B

# Código fuente

Listado B.1: Clase java con métodos para pre-procesamiento de notas médicas

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import opennlp.tools.tokenize.SimpleTokenizer;
import opennlp.tools.tokenize.TokenizerModel;
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;

/**
 *
 * @author Erick Montelongo
 */
public class MedicalNotePreprocessing{

    /*Lista que tendrá las subnotas de el archivo*/
    private ArrayList<String> medicalNotes;
    /*Lista que contiene cada una de las palabras vacías*/
    private ArrayList<String> stopWords;
    /*Variable que contiene la ruta del archivo de la nota médida*/
    private String filePath =

    /*Constructor que recibe la ruta del archivo a pre-procesar,
    así como la ruta del archivo que contiene la lista de palabras vacías*/
    public MedicalNotePreprocessing(String file, String sWord){
        medicalNotes = new ArrayList<>();
        stopWords = new ArrayList<>();
        filePath = file;
        stopWordPath = sWord;
```

```
    }

    /*Método que carga en la varaible correspondiente la lista de palabras vac
        ías*/
    private void getStopWords(){
        try {
            String temp;
            FileReader reader = new FileReader(new File(stopWordPath));
            BufferedReader bf = new BufferedReader(reader);
            while((temp = bf.readLine()) != null){
                stopWords.add(temp);
            }
        } catch (IOException ex) {
            Logger.getLogger(MedicalNotePreprocessing.class.getName()).log(
                Level.SEVERE, null, ex);
        }
    }


    /*Metodo que realiza el proceso de segmentación del archivo*/
    public void segmentFile() {
        try{
            Reader reader = Files.newBufferedReader(Paths.get(filePath));
            CSVParser cvsParser = new CSVParser(reader, CSVFormat.EXCEL.
                withFirstRecordAsHeader());
            for(CSVRecord record: cvsParser){
                medicalNotes.add(record.get("TEXT"));
            }
        }
        catch(IOException e){
            e.printStackTrace();
        }
    }

    /*Método que realiza la eliminación de caracteres especiales. Recibe el
        patrón
    a eliminar, así como la cadena con la que se sustituirá*/
    public void eliminateNoise(String pattern, String replace) {
        ArrayList<String> auxMed = new ArrayList<>();
        StringBuilder sb = new StringBuilder();
        for(String s: medicalNotes){
            try{
                SimpleTokenizer tokenizer = SimpleTokenizer.INSTANCE;
                for(String token: tokenizer.tokenize(s)){
                    if(!token.matches(pattern) && token.length() > 2)
                        sb.append(token.toLowerCase()).append("␣");
                }
            }
            catch(Exception e){
                e.printStackTrace();
            }
            auxMed.add(sb.toString());
            sb.delete(0, sb.length());
        }
        medicalNotes = auxMed;
    }
```

```
    /*Método que itera sobre cada palabra y revisa si es una palabra vacía.
    En caso de ser así la elimina.*/
    public void eliminateStopWords() {
        ArrayList<String> auxMed = new ArrayList<>();
        StringBuilder aux = new StringBuilder();
        getStopWords();
        for(String s: medicalNotes){
            for(String k: s.split("␣")){
                if(!stopWords.contains(k.toLowerCase())){
                    aux.append(k);
                    aux.append("␣");
                }
            }
            auxMed.add(aux.toString());
            aux.delete(0, aux.length());
        }
        medicalNotes = auxMed;
    }

    /*Método que regresa todas las subnotas como una sola cadena de texto*/
    public String toSingleString() {
        StringBuilder sb = new StringBuilder();
        for(Iterator<String> it = medicalNotes.iterator(); it.hasNext(); ){
            String note = it.next();
            if(note.matches("(\\n)+"))
                it.remove();
            sb.append(note);
            sb.append("\\n");
        }
        return sb.toString();
    }

}
```

Listado B.2: Clase java con métodos para generar los vectores de los documentos y enviarlos a un archivo csv

```
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.List;
import java.io.File;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.Arrays;
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVPrinter;
import org.datavec.api.util.ClassPathResource;
import org.deeplearning4j.models.embeddings.inmemory.InMemoryLookupTable;
import org.deeplearning4j.models.embeddings.learning.impl.sequence.DM;
import org.deeplearning4j.models.embeddings.loader.WordVectorSerializer;
import org.deeplearning4j.models.paragraphvectors.ParagraphVectors;
import org.deeplearning4j.models.word2vec.VocabWord;
import org.deeplearning4j.models.word2vec.Word2Vec;
import org.deeplearning4j.text.documentiterator.FileLabelAwareIterator;
import org.deeplearning4j.text.documentiterator.LabelAwareIterator;
import org.deeplearning4j.text.documentiterator.LabelledDocument;
```

```
import org.deeplearning4j.text.tokenization.tokenizer.preprocessor.
    CommonPreprocessor;
import org.deeplearning4j.text.tokenization.tokenizerfactory.
    DefaultTokenizerFactory;
import org.deeplearning4j.text.tokenization.tokenizerfactory.TokenizerFactory;
import org.nd4j.linalg.api.ndarray.INDArray;
import org.nd4j.linalg.primitives.Pair;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 *
 * @author Erick Montelongo
 */
public class ParagraphVectorsClassifier {

    ParagraphVectors  paragraphVectors;
    LabelAwareIterator iterator;
    TokenizerFactory tokenizerFactory;
    /*Tamanio de la capa de salida. Es el número de características,
    que tendrá el vector*/
    private static final int LAYER_SIZE = 300;
    /*Ruta donde se guardará el archivo csv generado del vector*/
    private static final String PARAGRAPH_VECTORS_CSV = "./paragraphVectors.
        csv";

    private static final Logger log = LoggerFactory.getLogger(
        ParagraphVectorsClassifier.class);

 public static void main(String[] args) throws Exception {

      ParagraphVectorsClassifier modelo = new ParagraphVectorsClassifier();
      modelo.crearModelo();
      modelo.generaVectores();

    }

    void crearModelo()  throws Exception {

      /*Ruta dentro del ClassPath que contiene las notas*/
      ClassPathResource resource = new ClassPathResource("data");
      System.out.println(resource.getURL().toString());

      /*Se construye un iterador para el conjunto de datos. En este caso toma
          el nombre
      de la carpeta como la clase a la que pertenece el documento*/
      iterator = new FileLabelAwareIterator.Builder()
              .addSourceFolder(resource.getFile())
              .build();

      tokenizerFactory = new DefaultTokenizerFactory();
      tokenizerFactory.setTokenPreProcessor(new CommonPreprocessor());



      /*Parametros utilizados para configurar al algoritmo*/
      paragraphVectors = new ParagraphVectors.Builder()
```

```
            .learningRate(0.020)
            .minLearningRate(0.001)
            .batchSize(32)
            .epochs(45)
            .iterate(iterator)
            .trainWordVectors(true)
            .layerSize(LAYER_SIZE)
            .tokenizerFactory(tokenizerFactory)
            .sequenceLearningAlgorithm("org.deeplearning4j.models.embeddings
                .learning.impl.sequence.DM")
            .build();

  paragraphVectors.fit();
  /*Método utilizado para escribir el modelo generado*/
  WordVectorSerializer.writeParagraphVectors(paragraphVectors, "
      paragraphVectorModel.txt");
}

/*Método utilizado para generar los vectores de párrafos*/
void generaVectores() throws IOException {

 /*
  Se itera de nuevo sobre el iterator para obtener el valor numérico (
      documentAsVector)
  de cada uno de los documentos.
 */
 MeansBuilder meansBuilder = new MeansBuilder(
     (InMemoryLookupTable<VocabWord>)paragraphVectors.getLookupTable(),
       tokenizerFactory);
 iterator.reset();
 while(iterator.hasNextDocument()){
     LabelledDocument document = iterator.nextDocument();
     INDArray documentAsCentroid = meansBuilder.documentAsVector(document)
         ;
     toCSVFile(documentAsCentroid, document.getLabels().get(0));
 }



}

/*Método que sirve para escribir los valores del vector en un archivo csv
    */
private void toCSVFile(INDArray row, String label){
    try{
    BufferedWriter writer = Files.newBufferedWriter(Paths.get(
        PARAGRAPH_VECTORS_CSV), StandardOpenOption.APPEND);
    CSVPrinter csvPrinter = new CSVPrinter(writer, CSVFormat.DEFAULT);
    String[] aux = new String[LAYER_SIZE+1];
    for(int i = 0; i < LAYER_SIZE; i++)
        aux[i] = String.valueOf(row.getDouble(0, i));
    aux[LAYER_SIZE] = label;
    System.out.println(aux[0]);
    csvPrinter.printRecord((Object[]) aux);
    csvPrinter.flush();
    }
    catch(IOException e){
```

```
            e.printStackTrace();
        }

    }


}
```

Listado B.3: Clase de utilidad de java para obtener la representación en vector de un documento, en base a un modelo de Doc2Vec previamente entrenado

```
import org.deeplearning4j.models.embeddings.inmemory.InMemoryLookupTable;
import org.deeplearning4j.models.word2vec.VocabWord;
import org.deeplearning4j.models.word2vec.wordstore.VocabCache;
import org.deeplearning4j.text.documentiterator.LabelledDocument;
import org.deeplearning4j.text.tokenization.tokenizerfactory.TokenizerFactory;
import org.nd4j.linalg.api.ndarray.INDArray;
import org.nd4j.linalg.factory.Nd4j;

import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;

/**
 *
 * @author Erick Montelongo
 */
/**
 * Clase de utilidad para construir vectores de parrafo en base a
    un modelo de paragraph vectors
 */
public class MeansBuilder {
    private VocabCache<VocabWord> vocabCache;
    private InMemoryLookupTable<VocabWord> lookupTable;
    private TokenizerFactory tokenizerFactory;

    public MeansBuilder(InMemoryLookupTable<VocabWord> lookupTable,
        TokenizerFactory tokenizerFactory) {
        this.lookupTable = lookupTable;
        this.vocabCache = lookupTable.getVocab();
        this.tokenizerFactory = tokenizerFactory;
    }

    /**
     * Método que regresa un vector en base a un documento
     */
    public INDArray documentAsVector(LabelledDocument document) {
        List<String> documentAsTokens = tokenizerFactory.create(document.
            getContent()).getTokens();
        AtomicInteger cnt = new AtomicInteger(0);
        for (String word: documentAsTokens) {
            if (vocabCache.containsWord(word)) cnt.incrementAndGet();
        }
        INDArray allWords = Nd4j.create(cnt.get(), lookupTable.layerSize());
        cnt.set(0);
        for (String word: documentAsTokens) {
            if (vocabCache.containsWord(word))
                allWords.putRow(cnt.getAndIncrement(), lookupTable.vector(word
```

```
                    ));
            }
            INDArray mean = allWords.mean(0);
            return mean;
        }
}
```

Listado B.4: Script de python para entrenamiento y búsqueda de hyper-parámetros óptimos para SVM, MLP y AdaBoost

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jul 30 22:38:47 2019

@author: Erick Montelongo
"""
import pandas as pd
import numpy as np

"""Función utilizada para el manejo de valores faltantes de datos
    estructurados"""
def missingValuesStrategy(dataframe, numericStrategy, categoricalStrategy):
    from sklearn.impute import SimpleImputer
    imputer = SimpleImputer(missing_values=np.nan, strategy=numericStrategy)
    frame = dataframe.iloc[:,6:19]
    imputer.fit(frame)
    dataframe.iloc[:,6:19] = imputer.transform(frame.values[:,0:13])
    imputer = SimpleImputer(strategy=categoricalStrategy)
    frame = dataframe.iloc[:,0:6]
    imputer.fit(frame)
    dataframe.iloc[:,0:6] = imputer.transform(frame.values[:,0:6])

"""Función utilizada para el escalamiento entre -1 y 1 de los datos
    estructurados"""
def standarizeMinMaxFeatures(dataframe):
    from sklearn import preprocessing
    min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-1,1))
    frame = dataframe.iloc[0:,6:19]
    min_max_scaler.fit(frame)
    dataframe.iloc[0:,6:19] = min_max_scaler.transform(frame.values[:,0:13])

"""Función utilizada para la codificación de las características alfanuméricas
    """
def encodeCategoricalFeatures(dataframe):
    dataframe = pd.concat([pd.get_dummies(dataframe.iloc[:,0:6]),dataframe],
        axis=1)
    dataframe.drop(['sex','marital_status','ethnicity','religion','
        admission_type','admission_source'], axis=1,inplace=True)
    return dataframe

"""Función utilizada para el proceso de entrenamiento, búsqueda de parámetros
    y medición del
desempenio de un clasificador"""
def process(X_train,X_test, Y_train, Y_test, classificator, tunedParameters,
    outputFile):
    scores = ['precision']
    f = open(outputFile,"w+")
```

```
    for score in scores:
        print("Ajustando hyper-parametros para %s" % score)
        print()
        #Se considera un valor de CV=5
        clf = GridSearchCV(classificator, tunedParameters, cv=5, scoring='%
            s_macro' % score)
        clf.fit(X_train,Y_train)
        print("Los mejores parametros encontrados en el conjunto:")
        print()
        print(clf.best_params_)
        f.write("Best Param: "+str(score)+":"+str(clf.best_params_)+"\r\n")
        print()
        print("Valores del grid en el conjunto:")
        print()
        means = clf.cv_results_['mean_test_score']
        stds = clf.cv_results_['std_test_score']
        for mean, std, params in zip(means, stds, clf.cv_results_['params']):
            print("%0.3f (+/-%0.03f) for %r"
                  % (mean, std * 2, params))
            f.write("%0.3f (+/-%0.03f) for %r \n"
                  % (mean, std * 2, params))
        print()
        print("Reporte detallado de clasificación:")
        print()
        print()
        y_true, y_pred = Y_test, clf.predict(X_test)
        print(classification_report(y_true, y_pred))
        f.write(str(classification_report(y_true, y_pred)))
        print()

"""Archivo csv que contiene los datos estructurados y no estructurados"""
dataframe = pd.read_csv("allData.csv", header=0, index_col=0)
Y = dataframe.iloc[:,319]
missingValuesStrategy(dataframe, 'mean', 'most_frequent')
standarizeMinMaxFeatures(dataframe)
dataframe = encodeCategoricalFeatures(dataframe)
#X = dataframe.iloc[:,19:319] # Datos no estructurados
#X = dataframe.iloc[:,0:52]   # Datos estructurados
X = dataframe.iloc[:,0:352] # Todos los datos

"""División del conjunto de datos en 80% entrenamiento y 20% prueba"""
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20)

"""Importación de GridSearchCV para busqueda de parámetros óptimos en un
    conjunto
de hyper-parámetros definidos"""
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
"""Importe de modelos de ML"""
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier

"""Hyper-parámetros a optimizar para SVM, así como los distintos valores que
    tomarán
    Incremento de gamma desde 0.1 a 1 en pasos de 0.1
```

```
    Incremento de C de 1 a 1000 en incrementos de 100
    Kernel rbf"""
svm_tuned_parameters = [{'kernel':['rbf'],
                         'gamma':np.arange(0.1,1,0.1),
                         'C':np.arange(1,1000,100)}]

"""Hyper-parámetros a optimizar para MLP, así como los distintos valores que
    tomarán
    Tamanio de capa de 300 a 600 en incrementos de 100
    Funciones de activación: 'identity', 'logistic', 'tanh', 'relu'
    Optimizador: lbfgs', 'sgd', 'adam'
    Iteraciones máximas: de 100 a 10000 en incrementos de 1000"""
mlp_tuned_parameters = [{'hidden_layer_sizes':np.arange(300,600,100), '
    activation':['identity', 'logistic', 'tanh', 'relu'],
                     'solver':['lbfgs', 'sgd', 'adam'], 'max_iter':np.arange
                        (100,1000,100)}]

"""Hyper-parámetros a optimizar para AdaBoost, así como los distintos valores
    que tomarán
    Número de estimadores: de 50 a 100 en incrementos de 1
    Tasa de aprendizaje: de 0.1 a 1 en incrementos de 0.01
    Algoritmo de AdaBoost: 'SAMME','SAMME.R'
"""
adaboost_tuned_parameters = [{'n_estimators':np.arange(50,100,1),
                              'learning_rate':np.arange(0.1,1,0.01),
                              'algorithm':['SAMME','SAMME.R']}]


"""En esta sección se indica el clasificador que se probará, así como el
conjunto de parámetros (definidos anteriormente) de los cuales buscará el ó
    ptimo,
además del archivo donde se depositará la salida. En este caso se esta
    buscando
mediante AdaBoostClassifier, el conjunto definido por
    adaboost_tuned_parameters, y
su salida se depositará en  AdaBoost_BestParams_all_take.txt"""
process(X_train, X_test, Y_train, Y_test, AdaBoostClassifier(),
    adaboost_tuned_parameters,"AdaBoost_BestParams_all_take.txt")
```