Proyecto 1
Explorando los Algoritmos de Aprendizaje Supervisado

1st Camacho Hernández José Julián
Instituto Tecnológico de Costa Rica
Reconocimiento de Patrones
Cartago, Costa Rica
jcamacho341@estudiantec.cr

2nd Guillén Fernández José Leonardo
Instituto Tecnológico de Costa Rica
Reconocimiento de Patrones
Cartago, Costa Rica
leoguillen@estudiantec.cr

***Resumen*—This paper will explain and show the performance of three different supervised learning algorithms using three datasets. Linear regression, decision trees and KNN models will be trained and tested to predict target values of the datasets. The adjustment of some hyperparameters of the models and data threatment such as feature engineering will be key to obtain optimal results. Our implementation of the algorithms will also be compared to Sklearn's by computing some metrics like accuracy, recall or F1 score. Results show that some models perform better for an specific dataset and some others tend to overfit.**

***Index Terms*—Machine Learning, KNN, Linear Regression, Logistic Regression, Decision Trees, feature engineering**

I. Introducción

En el siguiente informe se basa en el análisis de 3 algoritmos de aprendizaje supervisado con el cual se inicia explicando la teoría de cada algoritmo.

I-A. Regresión Logística

La regresión logística es un algoritmo de clasificación que se utiliza para predecir la probabilidad de una variable dependiente categórica, la variable dependiente binaria que contiene datos codificados como 1 o 0, significa que solo hay dos clases posibles.

Esta también es llamada la el nombre de la función que la compone Sigmoide, esta función es una curva en forma de s que puede tomar cualquier número de valor real y asignar un valor entre 0 y 1. Por lo que con ese resultado si es menor a 0.5 es un NO y si es mayor a 0.5 es un 1 . Se visualiza su comportamiento en la figura 1

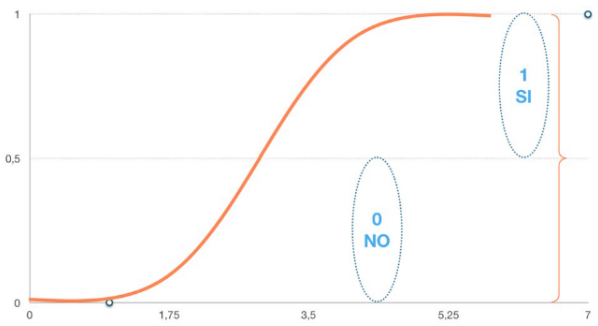

Figura 1. Curva Regresión Logística

I-B. Árboles de decisión

El árbol de decisión de Machine Learning es una estructura de árbol similar a un diagrama de flujo donde un nodo interno representa una característica, la rama representa una regla de decisión y cada nodo hoja representa el resultado, el nodo superior en un árbol de decisión en Machine Learning se conoce como el nodo raíz, Divide el árbol de una manera recursiva llama partición recursiva , esta estructura tipo diagrama de flujo lo ayuda a tomar decisiones.

Se selecciona el mejor atributo utilizando Medidas de selección de atributos ASM para dividir los registros. Hay que hacer que ese atributo sea un nodo de decisión y divida el conjunto de datos en subconjuntos más pequeños recursivamente para cada elemento que una de las condiciones coincida, tanto que no queden más atributos o no hayan más instancias.

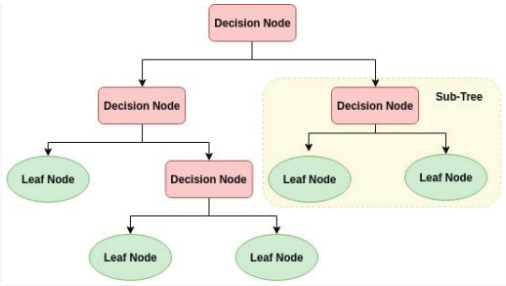

Figura 2. Árbol de decisión con sus terminologías

I-C. KNN

El algoritmo de k vecinos más cercanos , también conocido como KNN, es un clasificador de aprendizaje supervisado no paramétrico, que utiliza la proximidad para hacer clasificaciones o predicciones sobre la agrupación de un punto de datos

individual, para su clasificación se asigna una etiqueta de clase sobre la base de un voto mayoritario, es decir, se utiliza la etiqueta que se representa con más frecuencia alrededor de un unto de datos determinado.

Como métrica generalmente se utiliza la distancia euclidiana, que utiliza la siguiente ecuación:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \quad (1)$$

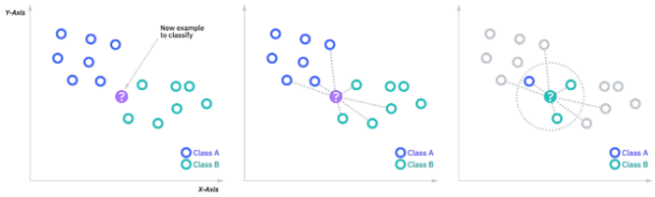

Figura 3. Diagrama KNN

I-D. Regresión Lineal

Una vez comprendidos los algoritmos de aprendizaje supervisado a utilizar, se procede a explicar la metodología del documento, gracias a la explicaciones anteriores vemos como funciona cada uno de los algoritmos y se sabe que se necesita para poder implementarlo.

Luego de eso se entra en la etapa de diseño del programa desarrollado en el cual se encontrarán diagramas de flujos de cómo se comporta el código implementado para cada uno de los tres algoritmos, viendo paso a paso de que se debe ir haciendo y que hace el algoritmo con los datos ingresados.

Cada algoritmo será testeado con 3 set de datos diferentes, los cuales son *Red Wine Quality*, que trata sobre la calidad de vinos tintos. Luego un set de datos relacionado con los resultados de Arquitectura en Computadores I, que se basa en varias notas con diferentes etiquetas de diferentes semestres y años. Finalmente, el set propio, para el cual se eligió *Heart Disease*. Este contiene datos relacionados con la salud y predicción de una posible enfermedad del corazón.

Con lo anterior se procederá a realizar un análisis de resultados, de cada algoritmo y cada set de datos, con el fin de comparar su comportamiento con base a los demás.

II. Detalles del diseño del programa desarrollo

En la figura 4 se aprecia el algoritmo de regresión logística. Este consta de dos partes: entrenamiento y predicción; con variables iniciales como lo es lo que nos ayuda a graduar el *training*. num_inter sería el numero de iteraciones que vaya a utilizar y fit_intercep sería si se concatena x con una matriz de ceros o no, con un valor de true o false.

Si se entra en la parte de entrenamiento el sistema valida la condición del fit_intercep , luego de ello entra al modo training que depende del número de iteraciones, dependiendo de esta cantidad es la veces que se repetirá el siguiente proceso, inicia sacando el producto matricial de X, que serían los datos ingresados, con la variable self.w, luego aplica el sigmoid a el resultado de ese producto matricial, luego de hacer otro producto matricial con (X.T , el resultado de sigmoid - el target y ingresado), este resultado se multiplicará por el valor de lr y actualiza al valor self.w.

Luego en la etapa de predicción, evalúa la condición del fit_intercep y se aplica el producto matricial de X, que serían los datos ingresados, con self.w que ahora contiene los datos entrenados , y el resultado se le aplica la función sigmoid y este sería el resultado final.

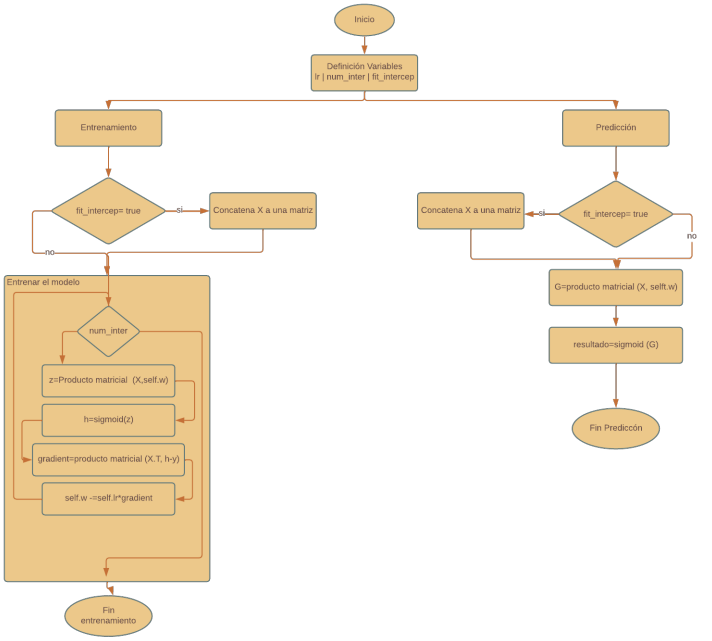

Figura 4. Regresión Logística

En la figura 5 se aprecia el diagrama de flujo de el algoritmo árboles de decisión, contando con dos datos definidos para la estructura como lo son data, serían los datos ingresados y el tarjet_column que sería la columna a evaluar, al igual que el algoritmo anterior se divide en dos partes, Creación del árbol y luego su predicción.

Entrando en la creación de los nodos , se inicia buscando el mejor atributo de división, una vez conocido se procede a crear un nodo con el mejor atributo, y revisa los puntos de parada, que son si todos los valores tienen la misma etiqueta o si no hay más datos, de tener uno de estos, el proceso de creación termina, si aún no cumple estas condiciones, el set de datos se divide en dos grupos y cada grupo crea sus propios nodos , volviendo al punto anterior.

Por la parte de predicción se valida que tenga nodos , si si tiene se busca la media en los datos ingresados , y esta es nero o igual se busca en el árbol below y se hace el mismo proceso anterior y si es mayor hace lo mismo pero con el árbol above, hasta llegar al punto de no tener nodos, el cual es el punto final y ahí termina la predicción.

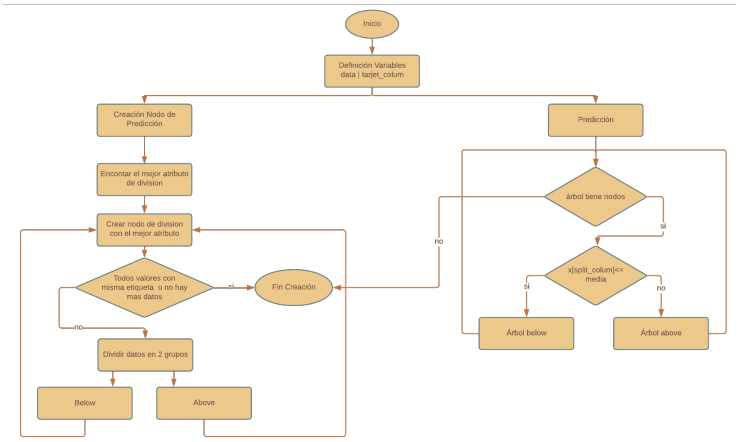

Figura 5. Árboles de decisión

En la figura 6 se puede apreciar el algoritmo KNN donde se empieza definiendo las variables del algoritmo tales como x_train , y_train que son los datos para entrenar,n-neighbors que serían el número de vecinos más cercanos , *weight* que denota *uniform* o *distance* que serían los dos métodos por el cual evaluar y por default n_classes que se definen 2.

Primero se realizan las distancias euclidianas de todos los X_test, para proceder a ordenar los índices y distancias y colocarlos en listas separadas.

Ahora entra la decisión del *weight*, para ver si hay que usar el método *uniform* o *distance*, de ser el primero se buscan los valores máximos en los índices de la lista, se cuentan sus concurrencias en el set de datos y pasa esto a una lista final.

Y por el lado de *distance*, busca obtener las distancias y vecinos y de estos obtener la mínima distancia promedio y con esto finaliza el algoritmo.

A continuación en la figura 7, se presenta el diagrama de flujo del algoritmo de regresión lineal, iniciando por la definición de variables del algoritmo las cuales nos ayudaran a mejorar o disminuir el rendimiento del algoritmo, learning_rate nos ayuda en el momento de actualizar los datos de weight y bias , num_interation es la cantidad de veces que se hará el *training*, y *weight* y *bias* contienen el manejos de datos.

El algoritmo tiene dos estados: *training* y *predict*. El primero inicia los valores en cero y luego hace un producto matricial de x y weight + bias, con el resultado de eso se calculan los gradientes y finaliza con la actualización de datos en *weight* y *bias*.

Y en la predicción, ya con el modelo entrenado y las variables llenas , se procede a realizar el producto matricial,de X con `weight + bias` y esa sería la predicción final.

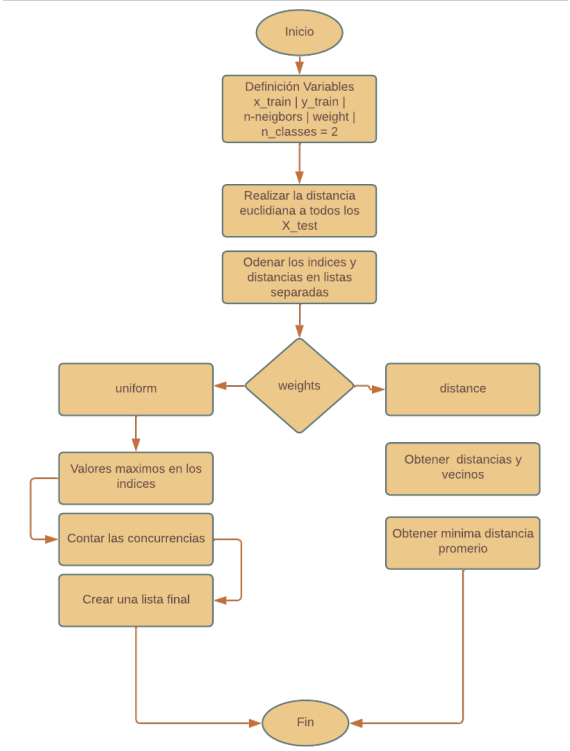

Figura 6. KNN

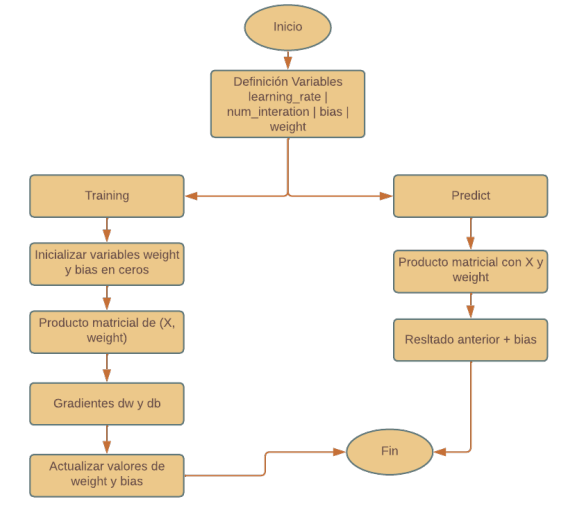

Figura 7. Regresión Lineal

III. Resultados y análisis

En la presente sección, se detallan aspectos relacionados con los resultados obtenidos al llevar a cabo el ciclo de diseño de un problema de reconocimiento de patrones. Se presenta el proceso de selección y *feature engineering* de los datos, la escogencia de los hiper-parámetros de los modelos

implementados y un análisis de los resultados obtenidos a partir de las funciones de predicción de los mismos.

III-A. Red Wine Quality

El primer set de datos que se utiliza es el denominado *Red Wine Quality*. Este consiste en una recopilación de la información relacionada con las variantes tinto y blanco del vino portugués "Vinho Verde". [1] Consta de 1599 muestras, las cuales están compuestas por 11 propiedades fisicoquímicas que buscan caracterizar la calidad del vino, en un rango del 3 al 8.

El primer paso de manejo que se le realiza a los datos es convertir el problema a uno de clasificación binaria. Esto se realiza al agrupar los valores del rango antes mencionado. En este caso, se determina que si la calidad tiene los valores 3, 4 o 5, se considerará como un vino de mala calidad, por lo que su valor de calidad será 0. Caso contrario, si está en el rango de 6 al 8, será considerado como un buen vino y su calidad estará representada con un 1.

Como parte del pre-procesamiento se analiza la detección de valores atípicos o datos *outliers*. Para esto, se determinó que los valores que se encuentran fuera de 1.5 veces el rango inter-cuartílico se considerarán valores atípicos por lo que serán eliminados. Esto con el fin de reducir el impacto desproporcionado que estos pueden generar en los resultados del modelo.

Como este set de datos no cuenta con valores nulos o faltantes, no se analiza el caso.

Seguidamente, respecto a la selección de *features* como parte del *feature engineering*, a partir del análisis del set da datos que se presenta en el paper: RED AND WHITE WINE DATA ANALYSIS. PREDICT QUALITY OF WINE [2], se decide eliminar la columna de *free sulfur dioxide* debido a que se encuentra directamente relacionado o tiene una correlación alta con el *feature 'total sulfur dioxide'*, ya es este último es la suma del *free sulfur dioxide* y otros tipos. De manera similar se descarta la densidad debido a que depende del porcentaje de alcohol y el contenido de azúcar. Esto se realiza para evitar sesgos por dependencia lineal entre los *features*.

Se analiza que los rangos de los datos son muy variados, por ejemplo, un *feature* tiene valores dentro del intervalo [0.01 - 0.61], mientras que otro se encuentra entre [6 - 289]. Esto genera que el modelo tome en cuenta en mayor medida la de números más elevados. Como no se considera que un *feature* sea más importante que otro en esta clasificación, se toma la decisión de estandarizar los datos. Esto con el fin de que todos se encuentren en un rango similar y así sean tomados en cuenta de manera igualitaria por el modelo.

Una vez se finaliza con el pre-procesamiento y *feature engineering* de los datos, se pasa a la división del set en conjuntos para entrenamiento y pruebas. Esto se realiza de manera estratificada, es decir, subdividiendo de tal forma que exista igual representación de clases en cada conjunto para reducir la posibilidad de sesgos por este motivo.

Para este set de datos, se realizarán las predicciones respecto a su calidad utilizando tres diferentes modelos: la regresión logística, los árboles de decisión, y el KNN. A partir de ello, se obtendrán métricas que indican el desempeño de los modelos, como la exactitud, la precisión, el *recall*, el *F1 score*, al AUC y se grafica el ROC.

Para este caso concreto, como la proporción de falsos positivos, o falsos negativos no es crítica, se brindará mayor importancia al análisis del *F1 score*, que considera armónicamente tanto la precisión como el *recall*.

III-A1. ***Regresión Logística****:* En primer lugar se presentan los resultados de la predicción utilizando el modelo de Regresión Logística. Estos corresponden a los mejores resultados, al utilizar un *learning rate* de 0.03, 50000 iteraciones y un umbral de 0.5.

En el cuadro I se presentan los valores obtenidos para las métricas mencionadas anteriormente al utilizar el modelo elaborado por el grupo de trabajo.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.7414	0.7695	0.7442	0.7567	0.7412
Test	0.7531	0.7917	0.7364	0.7631	0.7546

Cuadro I
RESULTADOS MODELO PROPIO

Se logra observar que al predecir con el conjunto de *testing* se obtiene un valor de *F1 Score* de 0.7631, que se considera aceptable, pero se podría mejorar, ya que el ideal sería de 1. Además, se nota que todas las métricas tienen valores cercanos al mencionado, y que su AUC indica que el modelo funciona de manera aceptable.

Al comparar los datos obtenidos al predecir con el conjunto de *training*, se observa que todas las métricas menos el *recall*, son mayores para el conjunto de *testing*, por lo que se podría mencionar que existe un *underfitting* muy leve, ya que el error es un poco más alto en el entrenamiento.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.7403	0.7680	0.7442	0.7559	0.7310
Test	0.7448	0.7833	0.7287	0.7551	0.7462

Cuadro II
RESULTADOS MODELO SKLEARN

Al predecir este set de datos con Regresión Logística de *Sklearn*, se obtienen valores muy similares a los presentados con el modelo propio. Por lo que se analiza que las mejoras pueden presentarse en el *feature engineering* en lugar de mejorar los modelos.

Adicionalmente, como los datos obtenidos entre el conjunto de *training* y *testing* son tan similares y variantes, no se puede concluir claramente si hay *overfitting* o *underfitting* en este caso.

III-A2. ***Árboles de decisión****:* Seguidamente, se realizan las predicciones con los árboles de decisión. En este caso el modelo de *Sklearn* mediante *GridSearch* determinó que el parámetro óptimo es una profundidad máxima de 14.

Como se presenta en el cuadro III, se nota que la implementación desarrollada por el grupo de trabajo obtuvo resultados perfectos, al alcanzar el valor ideal de 1 en todas las métricas

tanto para el conjunto de prueba como de entrenamiento. Esto significa que el modelo logró predecir correctamente todos los datos.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	1.0	1.0	1.0	1.0	1.0
Test	1.0	1.0	1.0	1.0	1.0

Cuadro III
RESULTADOS MODELO PROPIO

Como se presenta en el cuadro IV a continuación, el modelo de Sklearn obtuvo calificación perfecta en el *training*; sin embargo, en el *testing* su desempeño disminuyó significativamente. Esto presenta un caso muy marcado de *overfitting* en el modelo, ya que el modelo fue exageradamente bueno al entrenar, pero fue aceptable en las pruebas.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	1.0	1.0	1.0	1.0	1.0
Test	0.7615	0.7813	0.7752	0.778	0.7603

Cuadro IV
RESULTADOS MODELO SKLEARN

III-A3. ***KNN:*** Respecto a las predicciones con el algoritmo de KNN, *GridSearch* determinó que es óptimo analizar con k=17 y método de distancias. Sin embargo, en las pruebas con el modelo propio se obtuvieron mejores resultados con k=5 y un peso uniforme.

En el siguiente cuadro se presentan los resultados con el modelo propio, donde para el conjunto de *testing* se obtuvo un *F1 Score* de 0.7529, similar a los otros modelos analizados.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.8157	0.8232	0.8391	0.8311	0.8137
Test	0.7280	0.7388	0.7674	0.7529	0.72463

Cuadro V
RESULTADOS MODELO PROPIO

Se analiza que se obtuvieron valores de métricas mejores a la hora de entrenar, por lo que en este caso el modelo presentó un leve *overfitting*.

Respecto a las métricas de *Sklearn*, se nota que tuvo desempeño perfecto en el *training*, pero a la hora de probar el mismo bajó. Este es otro caso de *overfitting* en el modelo.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	1.0	1.0	1.0	1.0	1.0
Test	0.7950	0.8226	0.7907	0.8063	0.7953

Cuadro VI
RESULTADOS MODELO SKLEARN

Se analiza que este fue el modelo con el que se obtuvo el mejor *F1 Score*, con un valor bastante aceptado de 0.8063.

III-B. Notas del curso de Arquitectura de Computadores 1

El siguiente set de datos consiste en una recopilación de las notas de los estudiantes del curso de Arquitectura de Computadores I en los 4 semestres anteriores al actual. En este caso, se requiere considerar la nota del proyecto individual 1, proyecto grupal 1, examen 1 y tarea 1, con el fin de que los modelos predigan si la nota será mayor o menor a 67.5, es decir, que clasifiquen si el estudiante aprobará o no el curso.

Los datos de cada semestre se encuentran en archivos `.csv` aparte por lo que en primera instancia se cargan por separado. Para cada uno, primeramente se realiza el proceso de extracción de los *features* que se deben analizar. Estos corresponden a los anteriormente mencionados: proyectos 1, examen 1, tarea 1 y el *target* que en este caso será la nota final sin redondear. Las demás columnas se eliminan.

Seguidamente, con el fin de unificar los 4 sets, se procede al renombramiento de algunas columnas. De esta manera será posible combinar los archivos por medio del *match* en los nombres de sus columnas.

Al analizar el conjunto de datos, se observa que existen valores nulos o faltantes. Respecto a esto, se decide que si en una observación se presenta este caso en cualquiera de los *features* de interés, la misma será descartada. Esto debido a que si se da este caso, puede significar que el estudiante abandonó el curso, o su nota no fue registrada correctamente, por lo que se considera como un caso atípico que puede afectar el rendimiento del modelo. Esto con el fin mantener la consistencia de los datos, y reducir la posibilidad de que los modelos brinden predicciones erróneas por este motivo.

Una vez realizado lo anterior, se nota que los datos del modelo son de tipo *string* por lo que se debe realizar la conversión a decimal con punto flotante para el manejo numérico de los mismos.

Como el objetivo es determinar si el estudiante pasó o no, es necesario convertir el *target* a binario. Para esto, si la nota final sin redondear es menor a 67.5 será un 0 y si es mayor, un 1.

Posteriormente, se analiza que todos los *features* de interés se encuentran en una escala de 0 a 100. Por el contexto específico del problema, se decide aplicar una ponderación de los datos por medio de un escalamiento correspondiente al porcentaje de cada rubro en la nota final del curso. Esto debido a que un 100 en el proyecto grupal 1, que representa un 13.12 % de la nota, puede ser más determinante para que un estudiante pase, en comparación con un 100 en la tarea 1, que vale solo 3.75 %.

Para este set de datos se toman las mismas métricas que para el anterior, y se analizará en mayor detalle el *F1 score* y el *recall*, ya que por la naturaleza del set de datos se debe prestar atención a la tasa de falsos negativos que arroja el modelo, para no afectar a un estudiante que sí obtuvo el puntaje para pasar.

III-B1. ***Regresión Logística:*** Primeramente, se realizan las predicciones con los modelos de regresión logística tanto propio como de *Sklearn*.

En el cuadro VII a continuación, se presentan las métricas obtenidas con el modelo propio. Al igual que para el set de datos de *Red Wine*, los mejores resultados se presentaron con un *leaning rate* de 0.03, 50000 iteraciones y un umbral de 0.5.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.9118	0.9310	0.9643	0.9474	0.8155
Test	0.9444	1.0	0.9333	0.9655	0.9667

Cuadro VII
RESULTADOS MODELO PROPIO

Se observa que se obtienen valores de *recall* = 0.9333 y *F1 Score* = 0.9655. En el contexto del set de datos, estos resultados son en gran medida buenos, ya que indican que el modelo tiene una buena tasa de verdaderos positivos y detecta muy bien los falsos negativos que serían importantes en este caso.

En comparación con el desempeño en el entrenamiento, se observa que en todas las métricas, exceptuando el *recall*, se obtienen valores mayores. Sin embargo, no se considera suficiente para que se catalogue como *overfitting* o *underfitting*.

Seguidamente se presentan las métricas para el modelo de *Sklearn*. Se nota un desempeño ligeramente mejor para el conjunto de *testing*.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.9118	0.9310	0.9643	0.9474	0.8155
Test	0.9444	0.9375	1.0	0.9677	0.8333

Cuadro VIII
RESULTADOS MODELO SKLEARN

De manera general, el modelo de regresión logística logró predecir de muy buena manera si un estudiante va a pasar o no según las notas consideradas.

III-B2. ***Árboles de decisión****:* Al predecir con el modelo de árbol de decisión propio, al igual que para el set pasado, se obtienen valores perfectos de métricas. Esto refuerza la idea de que el algoritmo implementado es muy bueno clasificando.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	1.0	1.0	1.0	1.0	1.0
Test	1.0	1.0	1.0	1.0	1.0

Cuadro IX
RESULTADOS MODELO PROPIO

El modelo de *Sklearn* arrojó datos bastante buenos también, con *recall* y *F1 Score* de 0.9333. Se analiza que se obtuvieron métricas levemente mejores en el entrenamiento por lo que se podría señalar un ligero *overfitting* en el modelo.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.9853	0.9825	1.0	0.9910	0.9583
Test	0.8889	0.9333	0.9333	0.9333	0.8

Cuadro X
RESULTADOS MODELO SKLEARN

III-B3. ***KNN****:* Con el algoritmo de KNN también se presentaron resultados favorables. *GridSearch* determinó que para este caso la cantidad de vecinos cercanos ideal es de k=3, con un peso uniforme. Los mejores resultados utilizando el modelo propio se obtuvieron con esos mismos parámetros.

Al analizar el cuadro III-B3, se observa que una vez más los valores de *recall* y *F1 Score* son muy buenos, llegando inclusive al 1 en el primero.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.9265	0.9322	0.9821	0.9565	0.8244
Test	0.9444	0.9375	1.0	0.9677	0.8333

Cuadro XI
RESULTADOS MODELO PROPIO

Respecto al modelo de *Sklearn*, se nota que con el conjunto de *testing* se obtienen métricas perfectas.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.9265	0.9322	0.9822	0.9566	0.8245
Test	1.0	1.0	1.0	1.0	1.0

Cuadro XII
RESULTADOS MODELO SKLEARN

En términos generales, para el set de datos de las Notas de Arquitectura de Computadores I, todos los modelos lograron resultados muy buenos. El peor modelo quizá fue el de regresión logística de *Sklearn* al tener valores de AUC menores a los demás. Sin embargo, se puede concluir que el *feature engineering* aplicado al set fue bastante correcto.

III-C. Heart Disease Dataset

En este set de datos llamado *Heart Disease Dataset* [3], que utiliza datos recopilados de enfermedades cardiacas y cuenta con 303 datos los cuales que cuentan con 14 columnas, las cuales son: edad, sexo, tipo de dolor torácico (4 valores), presión arterial en reposo, colesterol sérico en mg/dl, Azúcar en sangre en ayunas $\geq$ 120 mg/dl, resultados electrocardiográficos en reposo (valores 0,1,2), frecuencia cardíaca máxima alcanzada, angina inducida por el ejercicio, old peak = depresión del ST inducida por el ejercicio en relación con el reposo, la pendiente del segmento ST de ejercicio máximo, número de vasos principales (0-3) coloreados por fluoroscopia, tal: 0 = normal; 1 = defecto fijo; 2 = defecto reversible.

Aparte de estas columnas la número 14 es un *target* el cual será solo 0 o 1 representando si está enfermo o no está enfermo.

Luego en el manejo de datos y estandarización, se toman los valores target, sex, fbs, exang que son enteros y se convierten en flotantes para poder operarlos. En el conjunto de datos toman relaciones diferentes como por ejemplo la edad, la presión arterial en reposo y la frecuencia máxima, son valores grandes por lo que toman gran valor en la predicción, y datos como el sexo, la condición de azúcar en la sangre en ayunas , angina inducida por ejercicio y *target*, son datos binarios donde solo se visualizan 0 o 1 y en el caso de los demás datos están valorados en estados de 0,1,2.

Una vez procesados los datos se pasa a la división del set de datos para el conjunto de entrenamiento y pruebas, donde al igual que para los conjuntos pasados se genera de manera estratificada.

A continuación se analizan ciertas métricas que se necesitan para evaluar el desempeño de los algoritmos en este caso ver el desempeño de los algoritmo tanto con nuestra implementación como con la implementación de *sklearn*, con el set de datos de *Heart Diseases*, dichas métricas son *Accuracy, Precision, Recall, F1 Score*, AUC.

Se tomaron los resultados reiteradas ocasiones para conseguir el mejor caso y para cada implementación se obtuvieron los resultados que contienen el algoritmo regresión logística, Árboles de decisión y KNN respectivamente.

III-C1. ***Regresión Logística:*** En las siguientes tablas, se observan los resultados para el algoritmo de regresión lineal, con dos implementaciones, la propia y la implementación de *Sklearn*.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.85714	0.86915	0.90476	0.87735	0.85194
Test	0.86885	0.90322	0.87878	0.87500	0.87067

Cuadro XIII
RESULTADOS MODELO PROPIO

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.83516	0.82608	0.90476	0.86363	0.82251
Test	0.83606	0.81081	0.90909	0.85714	0.82954

Cuadro XIV
RESULTADOS MODELO SKLEARN

Comparando los datos, se observa que nuestra implementación es más exacta que la de *sklearn*, teniendo un mayor número de predicciones correctas. Luego, la precisión se observa que de la misma manera se tiene más precisión en la implementación propia que en la otra, lo cual va relacionado con la exactitud del modelo.

Posteriormente, se tiene la métrica de *recall*, en la cual se tiene el caso contrario a la precisión, es mayor el procesamiento de buenos datos por lo que se tiene un bajo promedio de falsos positivos. En esta métrica nos vamos a centrar puesto que el análisis de los datos se relacionan directamente con esta métrica, ya que cuando se detectan falsos negativos, lo cual es crítico en términos de salud, es mejor fallar en la detección de una persona que esta sana y decirle que esta enferma, que una persona que está enferma y se le diga que está bien. Por este motivo nos centramos en tener las menores posibilidades de falsos negativos tratando de tener un *recall* lo más cercano a 1.

La siguiente métrica en *F1 score* representa la combinación de precisión y *recall*, y analiza qué tan efectivo es el modelo, en los datos se observa que es más efectivo el modelo propio.

Finalmente el AUC que indica qué tan bueno es el modelo y se aprecia que con diferencia de 0.05 el modelo propio es superior al modelo de *sklearn*.

III-C2. ***Árboles de decisión:*** En las tablas siguientes, se observa el algoritmo de árboles de decisión, igualmente con dos implementaciones, la propia y la implementación de *sklearn*.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	1.0	1.0	1.0	1.0	1.0
Test	0.7375	0.76687	0.75438	0.74850	0.73798

Cuadro XV
RESULTADOS MODELO PROPIO

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.88461	0.90384	0.89523	0.89952	0.88268
Test	0.77049	0.78787	0.78787	0.78787	0.76893

Cuadro XVI
RESULTADOS MODELO SKLEARN

Al analizar y comparar los datos, vemos que nuestra implementación es menos exacta que la de *sklearn*, de la misma manera es más precisa esta segunda. Se aprecia que todos los valores son mayores para *sklearn*. Esto muestra que es mejor para identificar las posibilidades de falsos positivos y falsos negativos, que como se mencionó anteriormente, nos deja claro el identificar esta métrica con el *recall*, y viendo el *F1 score*.

También se aprecia su superioridad en rendimiento y efectividad, y con el AUC se menciona que *sklearn* es superior. Sin embargo, ninguno de los dos son tan efectivos , porque tiene un valor de rendimiento del 76 % y se podrían aplicar modelos con mejor rendimiento.

En este caso si me muestra *overfitting* en las dos implementaciones, se observa que las métricas en el set de *train* es mucho mayor que el set de prueba o test.

III-C3. ***KNN:*** Y en este último apartado, como se muestra en la tabla, se analiza el set de datos *Heart Disease* con el algoritmo KNN. Se nota que el modelo propio es superior en todas las métricas, recalcando que es más exacto, más preciso y su *recall* es bastante bueno.

Esto permite analizar los falsos positivos y se observa que este modelo es bastante apto para la implementación de detecciones del set de datos sobre enfermedades del corazón, ya que presenta un *F1 Score* bastante aceptable y superior al modelo de *sklearn*. Con respecto al AUC, no son tan diferentes; sin embargo, es mayor el propio mencionando que su rendimiento es un poco superior al de *sklearn*.

Y por último analizando esto datos vemos que en el caso del modelo propio la diferencia de métricas entre en train y el test , es relativamente poca, pero en el caso del modelo de sklearn si se ve gran diferencias , de la misma manera identificando el *overfitting*.

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	0.88461	0.90384	0.89523	0.89952	0.88268
Test	0.85245	0.83333	0.93939	0.87323	0.84740

Cuadro XVII
RESULTADOS MODELO PROPIO

Set	Accuracy	Precision	Recall	F1 Score	AUC
Train	1.0	1.0	1.0	1.0	1.0
Test	0.81967	0.82352	0.84848	0.83582	0.8171

Cuadro XVIII
RESULTADOS MODELO SKLEARN

III-D. Puntos Extra: Predicción de nota

En este apartado, se debe predecir la nota del estudiante utilizando el set de datos de Arquitectura de Computadores I. Para realizarla, se requiere la consideración de las siguiente notas: proyecto individual 1, proyecto grupal 1, examen 1 y taller 1.

El proceso de *feature engineering* es muy similar al detallado en la sección III-B, ya que se trata del mismo conjunto de datos. Se hace la selección de los *features* de interés, que este caso incluye el taller 1 en lugar de la tarea 1. Otra diferencia es que para el modelo de regresión lineal se realiza una normalización de los datos para obtener mejores resultados.

Los mejores parámetros del modelo fueron un *learning rate* de 0.005 con 1000 iteraciones.

En el cuadro III-D a continuación, se presentan las primeras 19 notas esperadas y las predichas por el modelo. A simple vista se puede observar que la mayoría de las predicciones se acercan significativamente las notas esperadas.

	Notas esperadas	**Notas predichas**
0	70.08	64.362009
1	88.17	91.039615
2	67.94	59.705608
3	76.16	73.313171
4	73.41	82.034184
5	83.63	75.892441
6	77.60	84.878215
7	50.20	50.995912
8	75.20	76.480774
9	79.32	74.876613
10	79.64	76.652433
11	86.24	89.269442
12	61.86	61.416467
13	69.40	69.475268
14	68.30	74.566519
15	74.67	68.139549
16	66.38	65.208167
17	72.41	66.293834
18	87.47	89.403680

Cuadro XIX
RESULTADOS REGRESIÓN LINEAL

Para tener un parámetro de error general, se toma en consideración el MSE *(Mean Squared Error)* entre los valores esperados y predichos. El mejor de los resultados que fue el que se presentó anteriormente, tuvo un $MSE = 24{,}675$. Esto indica que el error de predicción es relativamente bajo, por lo que se concluye que el modelo logra el objetivo de predecir la nota final de un estudiante basado en las notas de interés.

IV. CONCLUSIONES

En conclusión, en este documento se pudo observar las fortalezas y debilidades de cada algoritmo, tal como el hecho de que los árboles de decisión son fáciles de interpretar y pueden manejar datos no lineales pero dependiendo de la implementación pueden ser propensos a overfitting. En el caso de KNN, es útil para datos con patrones no lineales, pero puede ser costoso para grandes conjuntos de datos y en el caso de regresión logística, es útil para clasificación binaria y puede manejar datos lineales, pero no está flexible como los algoritmos anteriores.

También se concluyó en que no hay un algoritmo mejor, sino que depende del set de datos que se vaya a analizar y dependiendo de este, se elegirá el modelo más apto para la tarea.

Por este motivo es importante la medición de métricas como la precisión, *accuracy, recall* y *F1 Score*, para determinar cuál es el mejor para cada problema en específico.

REFERENCIAS

[1] UCI Machine Learning. *Red Wine Quality*. URL: https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009.

[2] Gregory D. Nelson. *RED AND WHITE WINE DATA ANALYSIS: PREDICT QUALITY OF WINE*. URL: https://scholarworks.calstate.edu/downloads/mg74qp13p.

[3] M Yasser H. *Heart Disease Dataset*. URL: https://www.kaggle.com/datasets/yasserh/heart-disease-dataset?resource=download.