Instituto Tecnológico de Estudios Superiores de Monterrey
Escuela de Ingeniería y Ciencias
Concentración Inteligencia artificial avanzada para la ciencia de datos
Monterrey, Nuevo León

TC3006C.103. Inteligencia artificial avanzada para la ciencia de datos I

Implementación de Inteligencia artificial avanzada para crear modelo predictivo sobre la competencia de Kaggle Titanic

Kaggle

Rodrigo de la Garza Martínez Matricula
César Alejandro Cruz Salas A00825747
Francisco José Joven Sánchez Matricula
David Emiliano Mireles Cárdenas Matricula
Mario Javier Soriano Aguilera A01384282
Marlon Brandon Romo López Matricula

1 de Septiembre de 2023

Índice

1. Introducción **2**

2. Problema/Reto **2**

3. Base de datos **2**

4. Modelo de Machine Learning **3**

4.1. Modelos Preliminares Seleccionados . . . 3

4.1.1. Red Neuronal . . . 3

4.1.2. Vecinos más cercanos . . . 6

4.1.3. Árboles . . . 7

4.1.3.1. Clasificador de árbol de decisión . . . 7

4.1.3.2. Bosques aleatorios . . . 8

4.1.3.3. XG Boost . . . 8

4.2. Modelo Definitivo . . . 9

4.3. Refinamiento . . . 9

5. Conclusiones finales **9**

5.1. Redes Neuronales . . . 9

5.2. Vecinos más cercanos . . . 10

5.3. Árboles . . . 10

6. Referencias **10**

1. Introducción

Nuestro objetivo principal es desarrollar un modelo de aprendizaje automático que pueda predecir si un pasajero del Titanic sobrevivió, basado en la información que existe sobre cada pasajero. Este modelo de predicción nos puede ayudar a comprender los factores clave que influyeron en las posibilidades de supervivencia durante el trágico hundimiento del Titanic en 1912. Antes de entrenar un modelo, es esencial encargarse de la limpieza de datos, ya que faltan muchos de los datos de algunos pasajeros, especialmente 'age' y 'cabin'. La falta de datos puede afectar significativamente la calidad de nuestras predicciones, por lo que implementaremos estrategias de usar datos falsos y, posiblemente, eliminación de registros con valores ausentes. Los algoritmos de aprendizaje automático que utilizamos incluyen redes neuronales (NN), k-vecinos más cercanos (KNN), aumento de gradiente (Gradient Boosting) y regresión logística. Cada uno de estos algoritmos tiene sus propias características y ventajas, y explicaremos cómo se comportan en la tarea de predicción de supervivencia de los pasajeros del Titanic.

2. Problema/Reto

El desafío principal en este proyecto reside en la capacidad de nuestro modelo para predecir la supervivencia de los pasajeros del Titanic basada en los datos proporcionados. Esto implica superar desafíos como la imputación de valores faltantes en datos cruciales, la selección adecuada de características relevantes y la elección de algoritmos de aprendizaje automático que puedan modelar la relación entre estas características y la supervivencia. Nuestra misión es desarrollar un modelo que no solo pueda proporcionar predicciones precisas.

3. Base de datos

Para empezar el desarrollo del modelo, utilizamos dos conjuntos de datos similares: 'train.csv' y 'test.csv'. El conjunto de datos 'train.csv' contiene información sobre un subconjunto de los pasajeros a bordo, en total 891 pasajeros forman parte de este subconjunto. Lo más importante es que revela si estos pasajeros sobrevivieron o no, lo que se conoce como verdad fundamental. En contraste, el conjunto de datos 'test.csv' proporciona información similar sobre otros 418 pasajeros, pero no divulga la verdad fundamental para cada uno de ellos. La tarea consiste en utilizar los patrones y relaciones identificados en los datos de 'train.csv' para predecir si los 418 pasajeros del conjunto 'test.csv' sobrevivieron o no. Los atributos conocidos para cada pasajero incluye su ID, clase socioeconómica, nombre, género, edad, número de hermanos/cónyuges a bordo, número de padres/hijos a bordo, número de boleto, tarifa, cabina y puerto de embarque. Es importante destacar que, en los datos de 'train.csv', faltan valores para la edad de 177 pasajeros y para la cabina de 687 pasajeros. En los datos de 'test.csv', 86 pasajeros no tienen información sobre su edad, y 327 no tienen datos sobre la cabina. Para estos datos

4. Modelo de Machine Learning

Los algoritmos de aprendizaje seleccionados, representan una variedad de enfoques y técnicas. Las redes neuronales (NN) es un modelo inspirado en el funcionamiento del cerebro humano, capaz de aprender patrones complejos a partir de datos. Los k-vecinos más cercanos (KNN) utiliza la similitud entre ejemplos cercanos para realizar predicciones. El aumento de gradiente (Gradient Boosting) es una técnica de conjunto que combina múltiples modelos simples para mejorar la precisión. Por último, la regresión logística se utiliza para problemas de clasificación binaria al modelar la relación entre características y probabilidades de pertenecer a una clase. Cada uno de estos algoritmos tiene sus propias ventajas y desafíos, y a lo largo de este informe.

4.1. Modelos Preliminares Seleccionados

4.1.1. Red Neuronal

El modelo de red neuronal creado en este informe se implementa utilizando Keras con TensorFlow. Se compone de 9 capas ocultas y una capa de salida diseñada para tareas de clasificación binaria. Cada capa oculta emplea la función de activación Rectified Linear Unit (ReLU), y sus parámetros de peso se inicializan utilizando el método HeUniform. La capa de salida se compone de dos unidades con una función de activación softmax, adecuada para problemas de clasificación binaria.

El entrenamiento del "Modelo 1çomienza con los datos de entrada y la feature de Survived correspondiente. El entrenamiento se lleva a cabo a lo largo de 800 épocas con una división de datos del 15 % para la validación en cada época y un batch size de 40. En el primer conjunto de gráficos de entrenamiento “Fig. 1”, observamos un notable aumento de la precisión del entrenamiento, que llega hasta el 95 %. Sin embargo, al examinar la precisión de validación, ésta se mantiene en torno al 75 %. Al mismo tiempo, la pérdida de entrenamiento disminuye a lo largo de las épocas, mientras que la pérdida de validación aumenta significativamente “Fig. 2”

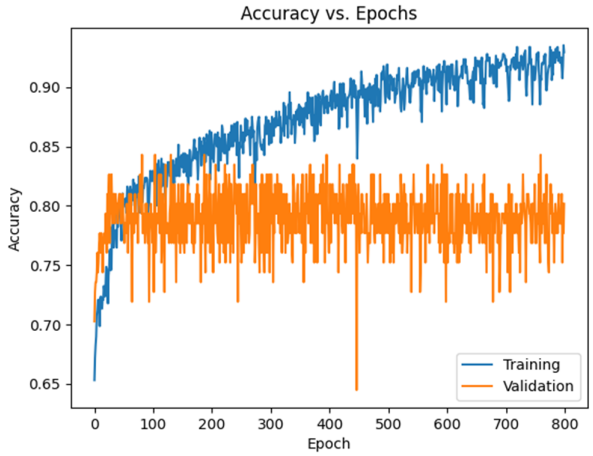

Figura 1: NN Model 1 Accuracy

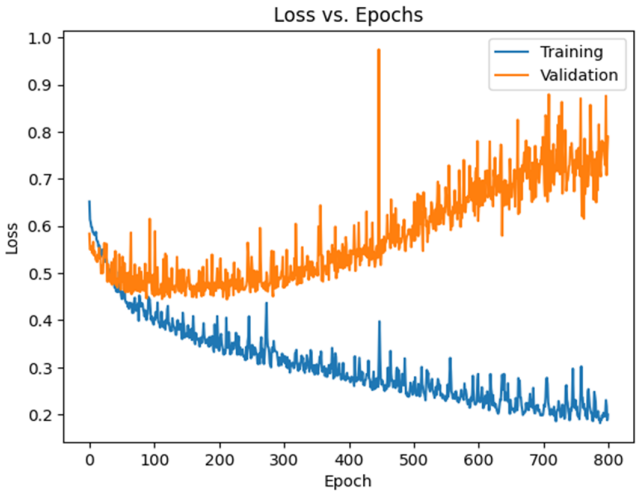

Figura 2: NN Model 1 Loss

Por otro lado, el "Modelo 2.^{en}globa múltiples capas ocultas, inicialmente compuestas por capas densas de 64 y 128 unidades con activación ReLU, reflejando la arquitectura del "Modelo 1". Se incorporan capas de dropout con una tasa del 30 % para evitar el sobreajuste y una capa de batch normalization para mejorar la estabilidad del entrenamiento. El "Modelo 2çoncluye con dos capas ocultas adicionales de 64 unidades cada una y una capa de salida con 2 unidades y activación softmax para la clasificación.

Este modelo es más complejo que el "Modelo 1". En los gráficos de entrenamiento nuevos ”Fig.3”, observamos un aumento constante de la precisión tanto en los conjuntos de entrenamiento como en los de validación. Una observación interesante es cuando la precisión de validación supera temporalmente a la de entrenamiento, seguido de un posterior descenso en la precisión de validación. Examinando los gráficos de pérdidas”Fig.4”, observamos que mientras la pérdida de entrenamiento disminuye constantemente y a veces se cruza con la pérdida de validación, la pérdida de validación, similar al ’Modelo 1’, experimenta una tendencia ascendente.

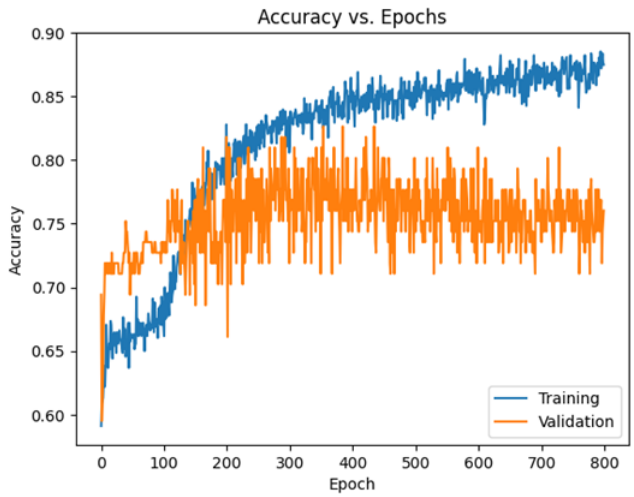

Figura 3: NN Model 2 Accuracy

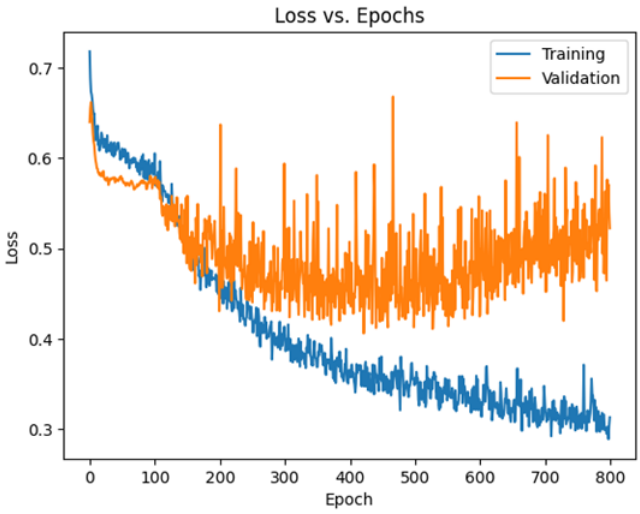

Figura 4: NN Model 2 Loss

Posteriormente, se emplean dos estrategias de entrenamiento cruciales. En primer lugar, se aplica la técnica Early Stopping.Este enfoque supervisa continuamente el rendimiento del modelo en un conjunto de validación y detiene el entrenamiento si la pérdida de validación no mejora durante 30 épocas consecutivas, lo que evita el sobreajuste y favorece una convergencia eficaz.

En segundo lugar, se pone en marcha la estrategia de reducción de la tasa de aprendizaje, facilitada por la función ReduceLROnPlateau. Esta estrategia dinámica ajusta la tasa de aprendizaje durante el entrenamiento. Si la pérdida de validación no mejora durante 15 épocas consecutivas, reduce la tasa de aprendizaje en un factor de 0,2, facilitando el ajuste fino del rendimiento del modelo y una mejor convergencia.

Tras la configuración de estas estrategias, se introduce el "Modelo 3". Este modelo se compila con una configuración distinta, incorporando un optimizador Adam con una tasa de aprendizaje de 0,0001. Se somete a un entrenamiento de 800 épocas utilizando el conjunto de datos de entrenamiento, con estrategias de parada temprana y reducción de la tasa de aprendizaje para un proceso de entrenamiento eficaz y controlado. Posteriormente, el rendimiento del modelo a lo largo del entrenamiento se visualiza mediante gráficos de precisión y pérdida, que ayudan a evaluar el rendimiento y la convergencia del modelo. Estas estrategias contribuyen colectivamente a la eficacia y la gestión del proceso de entrenamiento de redes neuronales, un aspecto fundamental para lograr resultados óptimos en tareas de aprendizaje automático

En la “Fig 5”, se destaca un aumento sustancial en la precisión de la validación en comparación con la del conjunto de entrenamiento, lo que arroja resultados inesperados en comparación con los modelos previos. Sin embargo, es notable que en el gráfico de pérdida, la pérdida “Fig 6” en el conjunto de entrenamiento es mayor que en el de validación, una observación que contradice las expectativas y sugiere la necesidad de ajustar algunos hiperparámetros. En mi opinión, es crucial explorar estos ajustes para comprender mejor y potenciar el rendimiento del modelo, ya que los resultados actuales son menos favorables en comparación con los modelos previos.

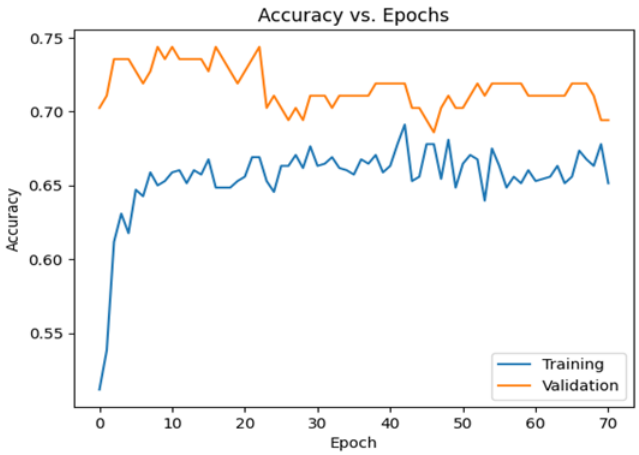

Figura 5: NN Model 3 Accuracy

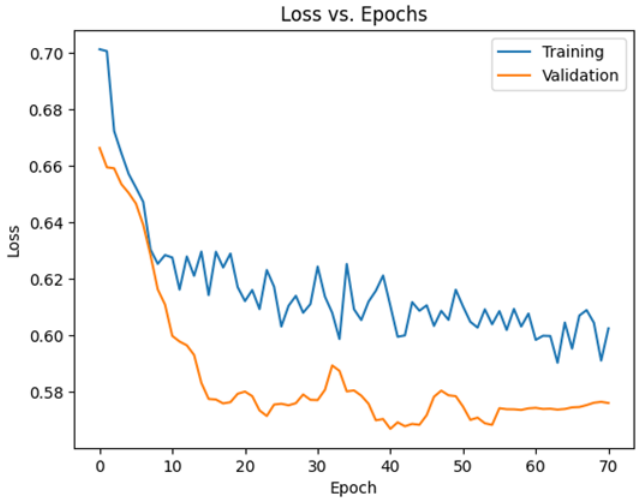

Figura 6: NN Model 3 Loss

Este modelo exhibió un rendimiento insatisfactorio, logrando una precisión de validación del 67.7 % y una pérdida del 59.6 %. Estos resultados son notablemente desfavorables. Además, al evaluar el modelo con el conjunto de pruebas de Kaggle, se obtuvo una precisión del 64 %, lo que confirma la necesidad de mejorar el rendimiento de este modelo.

4.1.2. Vecinos más cercanos

El algoritmo de K-Nearest Neighbors cae dentro de la categoría de algoritmos perezosos al ser uno que simplemente observa los datos alrededor del cada dato analizado y los agrupa en base a su proximidad sin pasar por una etapa de entrenamiento, o más específicamente, se mide la distancia entre un dato y los demás, se ordenan en una lista de manera que los primeros en la lista(los más cercanos) se agrupan para usar el modo para realizar clasificación, o la media si es regresión, obteniendo el resultado que se necesite.

Verdaderamente este método no requiere entrenamiento en un principio si es que ya conoces la cantidad óptima de vecinos a analizar en cada dato, aquí no los conocemos, pero tampoco tenemos el resultado final

de la cantidad de sobrevivientes, por lo tanto podemos seleccionar una de 3 opciones, la primera es elegir un número "al azar", la segunda es estar probando haciendo sumbits en la página de Kaggle, y por último es tomar una muestra del entrenamiento y probar suerte con el número que obtenga mayor precisión al hacer entrenamientos de predicción con la muestra y el dataset completo. En este caso probamos con las últimas 2 opciones, teniendo resultados desde el 37.8 % hasta en los mejores casos un acierto del 67.9 %, por lo que se puede observar que no sería la mejor solución, eso o sería necesario probar con números más grandes para el número de vecinos.

4.1.3. Árboles

Los modelos de árboles que existen para hacer predicciones machine learning son algoritmos que utilizan una estructura jerárquica de nodos y ramas para representar las reglas de decisión basadas en las características de los datos. Los modelos de árboles se pueden dividir en dos tipos principales: árboles de regresión y árboles de clasificación. Los árboles de regresión son modelos que predicen una variable continua a partir de las características de los datos. Los árboles de regresión se construyen dividiendo los datos en regiones homogéneas, donde la predicción se hace con el valor medio de la variable objetivo en cada región. Los árboles de clasificación son modelos que predicen una variable categórica a partir de las características de los datos. Los árboles de clasificación se construyen dividiendo los datos en regiones puras, donde la predicción se hace con la clase más frecuente o la probabilidad de cada clase en cada región. Existen diferentes métodos para aprender y mejorar los modelos de árboles, como el el bagging, el boosting y el random forest. Estos métodos varían en la forma de elegir las divisiones, medir la calidad de los nodos, evitar el sobreajuste y combinar varios árboles para aumentar la precisión y la robustez

Grid search es una técnica de optimización que intenta encontrar los mejores valores de los hiperparámetros de un modelo. Consiste en realizar una búsqueda exhaustiva sobre un conjunto de valores específicos para cada hiperparámetro y evaluar el rendimiento del modelo con cada combinación posible. El grid search puede ayudar a los tres modelos anteriores mencionados (clasificador de árbol de decisión, bosque aleatorio y XGBoost) a encontrar los mejores parámetros e hiperparámetros para mejorar la precisión y la eficiencia de las predicciones. Sin embargo, el grid search también tiene algunas limitaciones, como el alto costo computacional, el riesgo de sobreajuste y la dependencia de la elección del rango y la escala de los valores a explorar

4.1.3.1. Clasificador de árbol de decisión

Un clasificador de árbol de decisión es un algoritmo de aprendizaje supervisado que se usa para clasificar o predecir datos basándose en una estructura de árbol jerárquica. Cada nodo del árbol representa una pregunta o una condición sobre una característica de los datos, y cada rama representa una posible respuesta o resultado. El árbol se construye de forma recursiva, buscando el mejor punto de división para cada nodo, hasta que se alcanza un criterio de parada. Los nodos hoja representan las clases o los valores predichos para los datos. Para obtener los mejores parámetros e hiperparámetros del clasificador, se pueden usar diferentes técnicas, como la validación cruzada, la poda, el ajuste de la complejidad o el uso de conjuntos como el

bosque aleatorio. Estas técnicas ayudan a evitar el sobreajuste o el sobreajuste del modelo, y a mejorar su precisión y generalización

Se llevo acabo dos pruebas de arbol de decisión, donde los parámetros que asignamos eran $max_depth = 3$ donde el mejor resultado que se obtuve fue el de $max_depth = 3$ con un 77.51 % de accuracy en Kaggle

4.1.3.2. Bosques aleatorios

Un bosque aleatorio es un algoritmo de aprendizaje supervisado que combina varios árboles de decisión para mejorar la precisión y la robustez de las predicciones. Cada árbol se entrena con un subconjunto de los datos y de las características, elegidos al azar, lo que introduce diversidad y reduce el sobreajuste. La predicción final se obtiene por votación mayoritaria para los problemas de clasificación o por promedio para los problemas de regresión. Para obtener los mejores parámetros e hiperparámetros del bosque aleatorio, se pueden usar diferentes técnicas, como la validación cruzada, el ajuste de la complejidad o la selección de características. Estas técnicas ayudan a encontrar el número óptimo de árboles, la profundidad máxima de cada árbol, el número de características a considerar en cada división y el criterio para medir la calidad de una división.

En bosques aleatorios se hizo uso de grid search para establecer los valores óptimos de los hiperparámetros "Fig.7". Asignando una cuadrícula de distintos valores para poder realizar las predicciones se obtuvo que los mejores fueron con un $max_depth = 5$ y $n_jobs = 10$. "Fig.8"

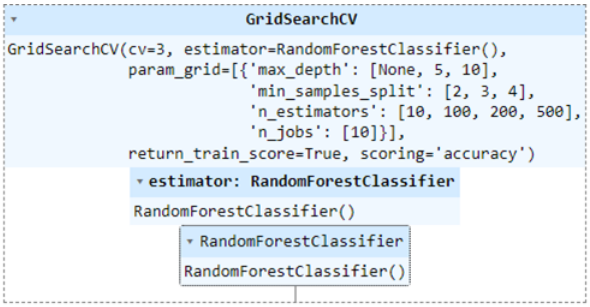

Figura 7: Hiperparámetros Grid Search Bosques Aleatorios

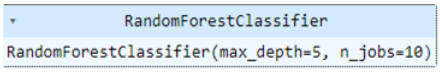

Figura 8: Hiperparámetros para predicción Bosques Aleatorios

En este modelo se obtuvo un accuracy del 77.03 % en Kaggle

4.1.3.3. XG Boost

XGBoost es un algoritmo de aprendizaje supervisado que utiliza el refuerzo de gradientes para mejorar la precisión y la eficiencia de las predicciones. XGBoost combina varios árboles de decisión que se entrenan de

forma secuencial, corrigiendo los errores de los árboles anteriores. XGBoost también incorpora regularización, paralelización y optimización para lograr un rendimiento superior y una mayor escalabilidad. Para obtener los mejores parámetros e hiperparámetros de XGBoost, se pueden usar diferentes técnicas, como la validación cruzada, el ajuste de la complejidad o la selección de características. Estas técnicas ayudan a encontrar el número óptimo de árboles, la profundidad máxima de cada árbol, la tasa de aprendizaje, el criterio para medir la calidad de una división y los parámetros de regularización. También se usó Grid Search para establecer los valores óptimos de los hiperparametros "Fig.9". Asignando una cuadrícula de distintos valores para poder realizar las predicciones, los mejores hiperparámetros son los que se muestran en la "Fig.10".

(a) (b)

(c)

Figura 9: Hiperparámetros Grid Search XG Boost

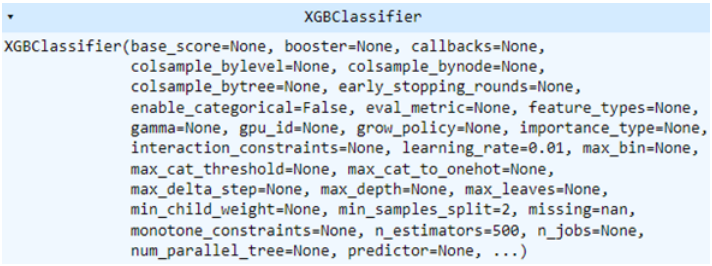

Figura 10: Hiperparámetros para predicción XG Boost

En este modelo se obtuvo un accuracy del 75.59 % en Kaggle

4.2. Modelo Definitivo

4.3. Refinamiento

5. Conclusiones finales

5.1. Redes Neuronales

En esta entrega, se observó un rendimiento deficiente en la implementación de las redes neuronales. Desde nuestro punto de vista, existen áreas de mejora importantes. En primer lugar, consideramos que aplicar un estratificado al dividir el conjunto de datos puede proporcionar muestras más representativas y mejorar el

rendimiento general. Además, la incorporación de técnicas como el backpropagation podría contribuir a la optimización del modelo. También es fundamental revisar y ajustar los hiperparámetros en el último modelo, ya que parece haber un problema no identificado que afecta negativamente la optimización del mismo. Estos enfoques pueden ayudar a elevar el rendimiento de las redes neuronales en futuras iteraciones.

5.2. Vecinos más cercanos

Se puede ver que la implementación del algoritmo no ha sido la mejor para realizar la predicción, y debido a su naturaleza perezosa sería difícil mejorarlo sin tener que probar un gran número de veces de manera manual, esto sin tener nada garantizado, por lo que creemos que sería mejor no continuar usándolo.

5.3. Árboles

Se puede decir que nuestro modelos de árboles fueron los mejores pero aún hay áreas de mejora, podemos implementar grid search a la clasificación de árboles de decisión para ver si un ajuste de hiperparametros puede mejorar nuestro accuracy en ese modelo. En general se cree que técnicas como discretización de los datos, eliminación de outliers, verificación de parámetros más significativos y feature engineering podrían provocar que obtengamos mejores resultados.

6. Referencias

Grus, J. (2015). Data science from scratch: first principles with Python. First edition. Sebastopol, CA, O'Reilly.

Geron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly.

¿Qué es KNN? | IBM. (s. f.). https://www.ibm.com/mx-es/topics/knn