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 A00825271
César Alejandro Cruz Salas A00825747
Francisco José Joven Sánchez A00830564
David Emiliano Mireles Cárdenas A01384282
Mario Javier Soriano Aguilera A01384282
Marlon Brandon Romo López A00827765

1 de Septiembre de 2023

Índice

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) [5] , k-vecinos más cercanos (KNN)[1], árboles de decisión[3], bosques aleatorios (RF)[4], aumento de gradiente (Gradient Boosting)[6] y regresión logística[7]. Cada uno de estos algoritmos los elegimos ya que tienen sus propias características y ventajas, como lo es la simplicidad del KNN, el buen modelado de datos complejos del NN o la robustez ante los datos ruidosos del RF entre otros, pero las explicaciones de cómo se comportan en la tarea de predicción de supervivencia de los pasajeros del Titanic están más adelante.

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 pueda proporcionar predicciones binarias correctas.

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.

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

[3]El modelo de red neuronal creado en este informe se implementa utilizando Keras con TensorFlow. Se compone de 9 capas ocultas de 64 unidades(neuronas) 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 1" comienza 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, mostrando Overfitting notablemente, ya que también se obtuvo un aumento notable en la precisión del conjunto de entrenamiento mientras que la precisión de validación se estanca o disminuye sugiere que el modelo se esta sobreajustandose a los datos de entrenamiento, para abordar el sobreajuste, se necesita regularización, como la reducción de la complejidad del modelo, la adición de términos de penalización en la función de perdida o el uso de técnicas como la eliminación de características irrelevantes. Estas técnicas nos van ayudar a controlar el overfitting y a mejorar la generalización del modelo. Tambien se puede obervar como se esta obteniendo una varainza alte y un sesgo alto, indicando que hay sobreajuste en nuestro modelo."Fig. 2"

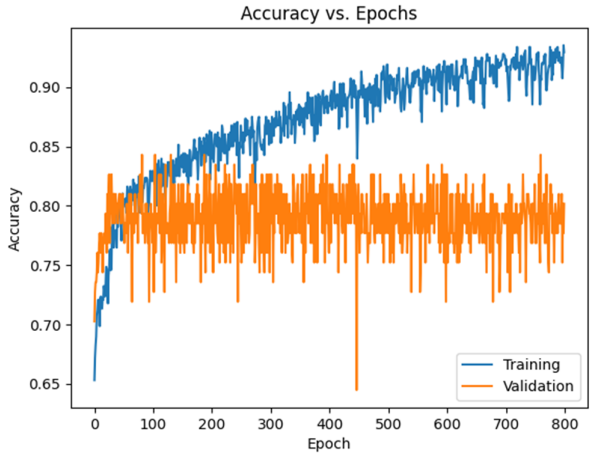


Figura 1: NN Model 1 Accuracy

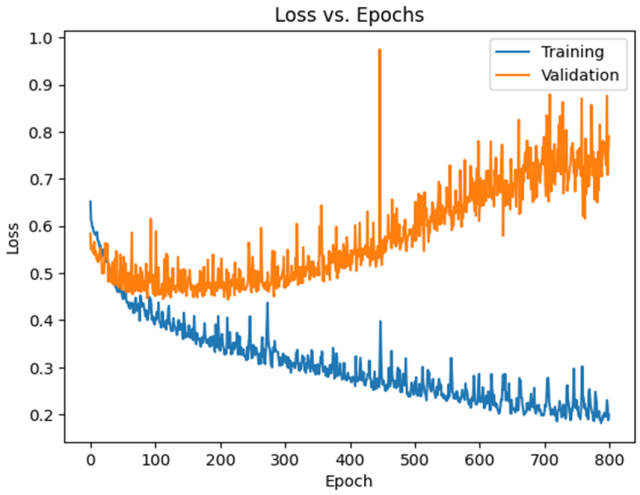


Figura 2: NN Model 1 Loss

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 % 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 como la precisión tanto en los conjuntos de entrenamiento como en los de validación están alcanzando puntos de sobreajuste igual que el anterior modelo. Examinando los gráficos de pérdidas”Fig.4”, de igual forma que el anterior modelo se tiene varianza alta y sesgo bajo, mostrando notablemente sobreajuste en nuestro modelo. En base a la precisión de validación, el modelo esta sobreajustado.

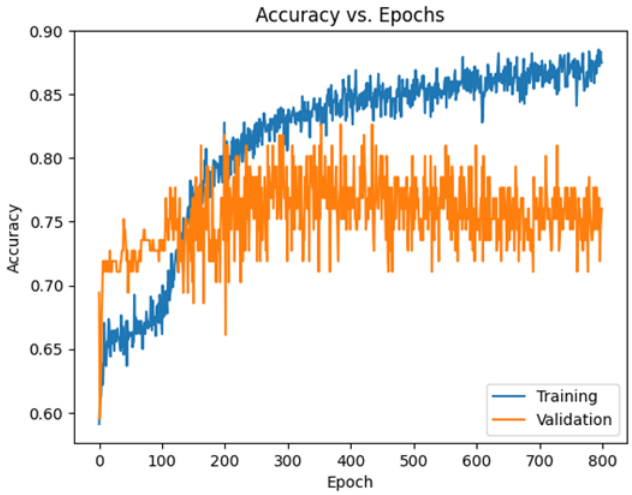


Figura 3: NN Model 2 Accuracy

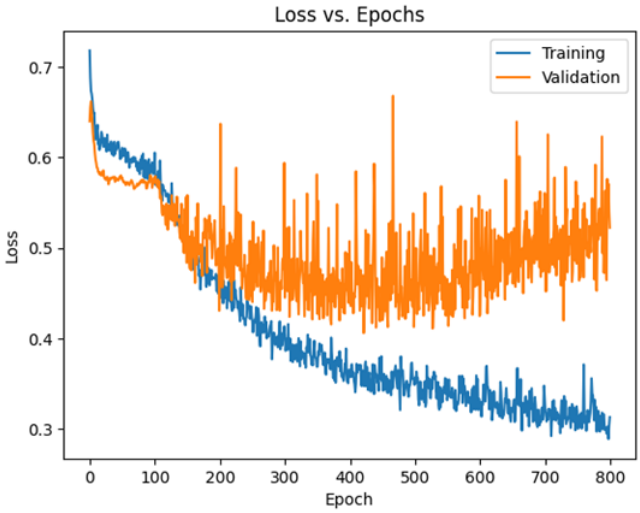


Figura 4: NN Model 2 Loss

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.

Estas dos técnicas las utilizamos con el objetivo de mejorar la eficacia del entrenamiento y evitar problemas como el overfitting. Early Stopping tiene ventajas ya que supervisa continuamente el rendimiento en un conjunto de validacion y detiene el entrenamiento si la perdida de validación no mejora durante un cierto numero de épocas consecutivas, con el propósito de no generar un overfitting. La Reducción de la Tasa de

Aprendizajes es una buena técnica que la escogimos debido a que ajusta la tasa de aprendizaje durante el entrenamiento. Si la perdida de validación no mejora durante un cierto numero de épocas consecutivas, la estrategia reduce la tasa de aprendizaje en un factor de 0.2. Esto ayuda a que el modelo realice un ajuste mas fino y delicado de sus pesos a medida que se acerca al óptimo.

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 en este modelo su ajuste a los datos y en como se evito tener overfitting deteniéndose en la época 70 . En el grafico de perdida, es notable que nuestro modelo se esta ajustando a los datos de forma adecuada deteniéndose acertadamente antes de sobreajustar el modelo, obteniendo un sesgo bajo y una varianza baja. “Fig 6” . En cuanto a los modelos de redes neuronales realizados este fue el mejor modelo.

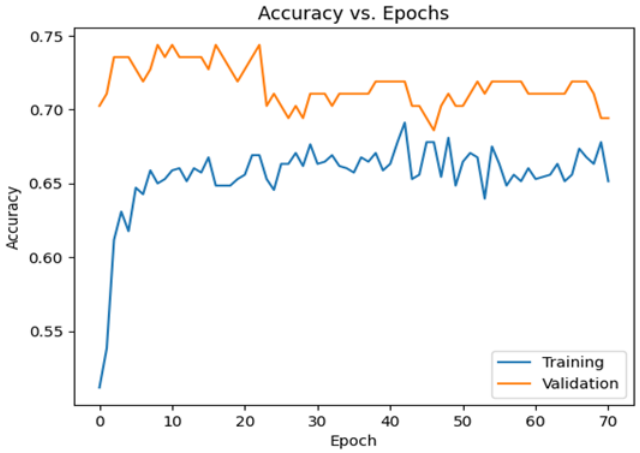


Figura 5: NN Model 3 Accuracy

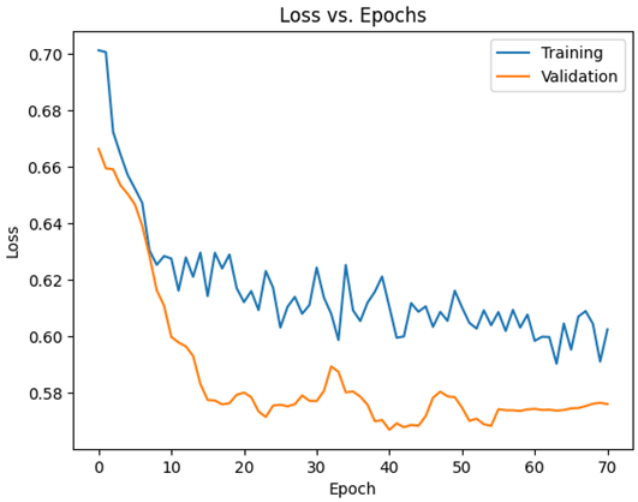


Figura 6: NN Model 3 Loss

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 nos afirma que no sera empleado como modelo final.

4.1.2. Vecinos más cercanos

El algoritmo de K-Nearest Neighbors[2] 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 sumbmits 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

[3]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.
Se llevo dos pruebas de arbol de decisión, donde los parámetros que asignamos eran $maxdepth = 3$, $criterion = "gini"$, $splitter = "best"$ y $minsamplesplit = 2$, 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 parámetros e hiperparámetros que puedan resultar más óptimos para el 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 $maxdepth = 5$ y $njobs = 10$. "Fig.8"

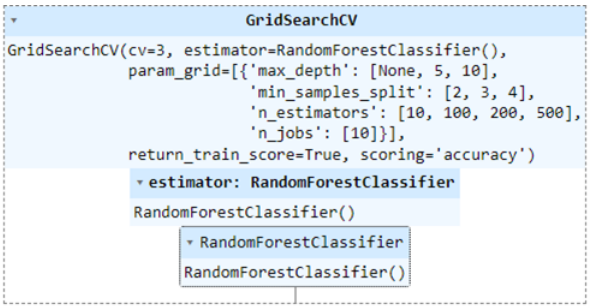


Figura 7: Hiperparámetros Grid Search Bosques Aleatorios

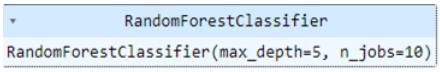


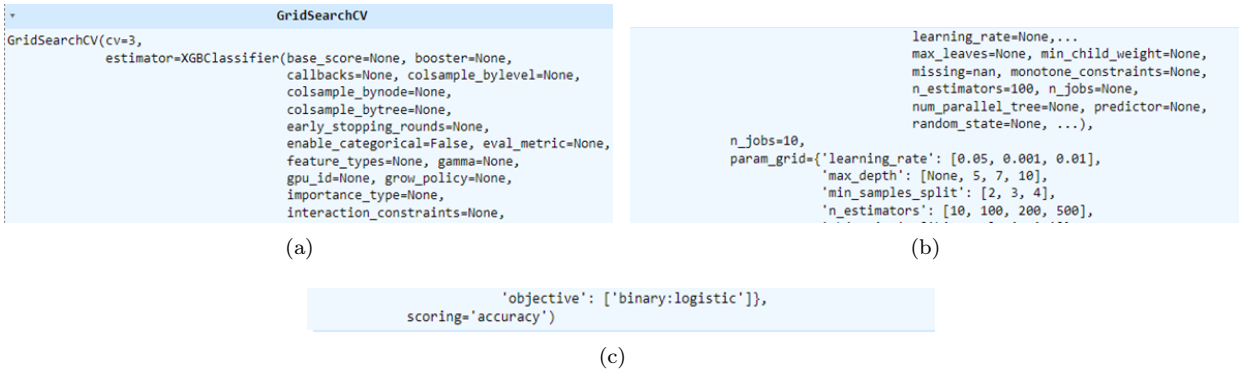
Figura 8: Hiperparámetros para predicción Bosques Aleatorios

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. 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

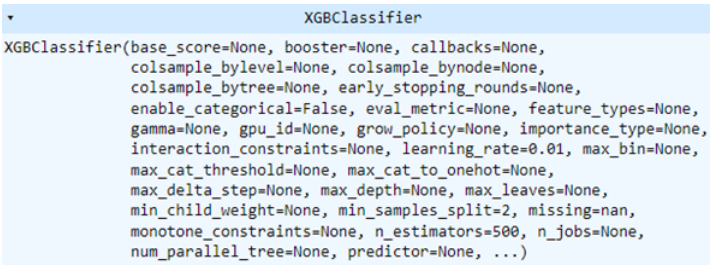


Figura 10: Hiperparámetros para predicción XG Boost

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

4.2. Modelos Definitivos

Los algoritmos que obtuvieron los mejores resultados en nuestra segunda entrega, en términos de precisión, fueron los modelos basados en árboles de decisión bosques aleatorios y el XG-boost. Específicamente, el modelo basado en árboles de decisión alcanzó una precisión del 77.51 en Kaggle, mientras que el modelo de bosques aleatorios obtuvo una precisión del 77.03 en Kaggle. El algoritmo de XG Boost también mostró resultados buenos ya que logró una precisión del 75.59 en Kaggle . Por lo tanto, en este contexto específico, aunque el modelo basado en árboles de decisión fue el algoritmo que proporcionó los mejores resultados, en la siguiente seccion de refinamiento se explorarán estos 3 modelos.

4.3. Refinamiento

Para refinar aún más nuestros modelos basados en árboles de decisión, existe un conjunto de estrategias prometedoras que podríamos implementar. En primer lugar, empleamos técnicas como Grid Search para ajustar los hiper parámetros del modelo y explorar si una configuración óptima mejora la precisión. También implementaremos la eliminación de outliers. Por último, el feature engineering es una herramienta poderosa que podría ayudarnos a crear nuevas características o transformar las existentes para capturar mejor las relaciones cruciales en los datos.

4.4. Modelos con Preprocesamiento Data 1

En esta etapa, se realizó una modificación en el procesamiento de los datos en comparación con el enfoque utilizado en la exploración inicial. En lugar de asignar 1 a los hombres y 0 a las mujeres en la columna "Sexo", se optó por un enfoque diferente. Dado que se conocía que casi el 80 % de las mujeres sobrevivieron, se decidió asignar un valor más alto a las mujeres, es decir, 1, para resaltar su importancia en el modelo.
De manera similar, se aplicó un enfoque parecido a la columna que representa la 'Clase' de los pasajeros. Dado que se tenía información de que los pasajeros de primera clase tenían tasas de supervivencia más altas en comparación con las de tercera clase, se asignó un valor más alto, en este caso 3, a los pasajeros de primera clase, mientras que a los de tercera clase se les asignó un valor más bajo, específicamente 1.
Estas modificaciones se realizaron con el propósito de reflejar la influencia de ciertas características en la supervivencia de los pasajeros de una manera más precisa en el modelo.

4.4.1. Árbol de decisión

En esta sección, primero se llevó a cabo una estandarización estándar, la cual fue aplicada de manera consistente en todos los modelos que se mencionarán posteriormente. Este proceso tiene como objetivo normalizar los datos.
Posteriormente, se implementó un Grid Search utilizando una validación cruzada de 5 divisiones, como para los demas casos de la data 1 y 2, con un amplio rango de hiper-parámetros. Estos hiper parámetros incluyeron la profundidad máxima del árbol, el número mínimo de muestras requeridas para dividir un nodo y el número mínimo de muestras requeridas para formar una hoja.
En la "Fig.11" se presenta el resultado del Grid Search para este modelo. El 'score' es una métrica numérica que indica qué tan bien se desempeña el modelo con esos hiper-parámetros específicos en los datos de validación. Este puntaje se basa en una métrica de evaluación específica, como precisión, exactitud, F1-score, entre otras, dependiendo del problema abordado. En este caso, se obtuvo un puntaje de validación cruzada de 0.82, utilizando los siguientes hiper-parámetros en el árbol:**max depth**=7, **min samples leaf**=4 y **min samples split**==20

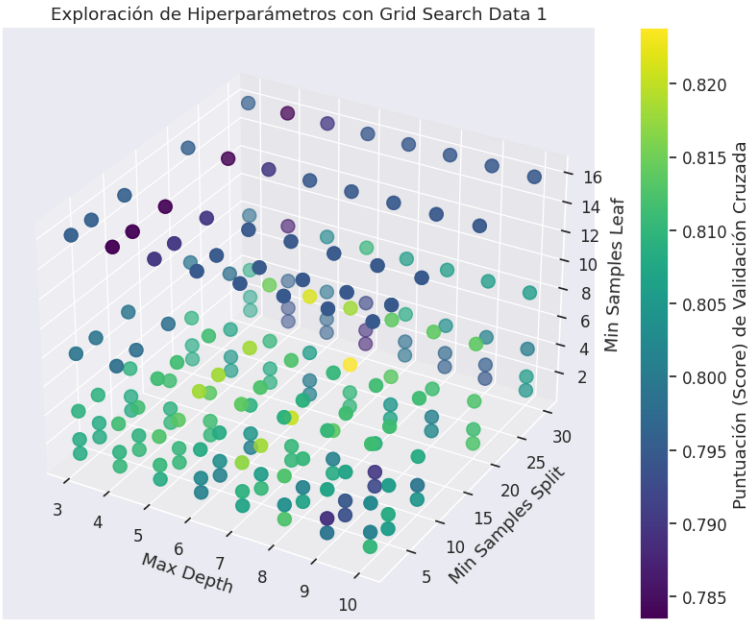


Figura 11: Tree-Gridsearch Data 1

La matriz de confusión fue la siguiente en la figura "Fig.12" nos muestra como tenemos 522 verdaderos negativos o muertes no supervivientes, 256 verdaderos positivos o supervivientes, 27 falsos positivos y 86 falsos negativos.

La matriz de confusión no indica necesariamente la eficacia del modelo en la clasificación, ya sea de sobrevivientes (casos positivos) o de no sobrevivientes (casos negativos). Además de la matriz de confusión, se emplearon otras métricas para evaluar el rendimiento del modelo:

Métrica	Valor
Exactitud	0.8732
Precisión	0.9046
Sensibilidad	0.7485
Especificidad	0.9508
F1-Score	0.8192

El modelo de Árbol de Decisión exhibe un rendimiento notable con una exactitud del 87.32 %, lo que significa que aproximadamente el 87.32 % de las predicciones son correctas. Además, destaca por su alta precisión del 90.46 %, indicando que el 90.46 % de las predicciones positivas son acertadas. No obstante, la sensibilidad es ligeramente inferior (74.85 %), lo que sugiere que podría haber margen para mejorar la detección de casos positivos. Por otro lado, la especificidad del 95.08 % demuestra una sólida habilidad para clasificar correctamente los casos negativos. El F1-Score, que combina precisión y sensibilidad, se sitúa en un destacado 81.92 %, lo que sugiere un equilibrio efectivo entre ambas métricas. En conjunto, este modelo de Árbol de Decisión presenta un rendimiento satisfactorio en la clasificación de los datos.

Finalmenente para este modelo en kaggle se obtuvo un precision de 76.78 %

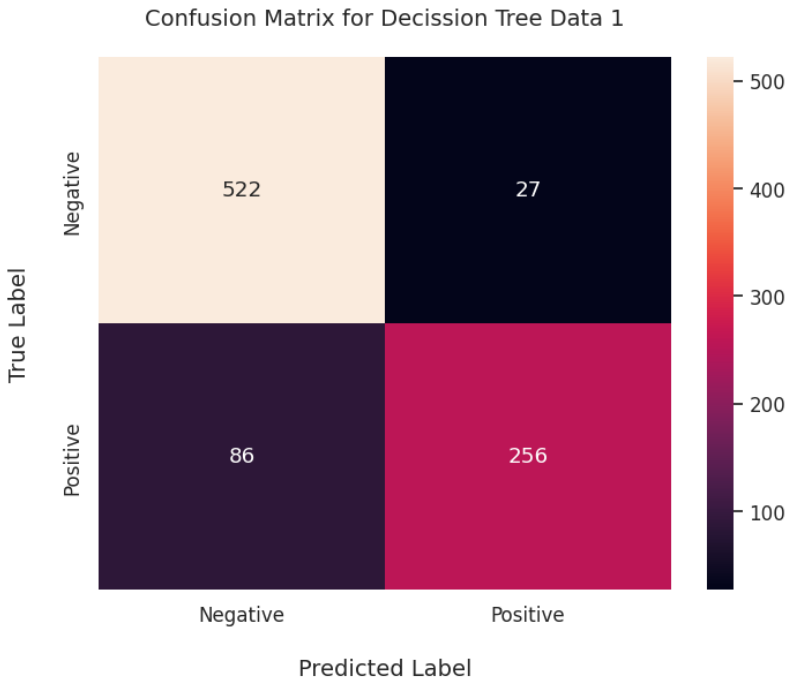


Figura 12: Tree-Confusion Matrix Data 1

4.4.2. Bosque Aleatorio

Para este segundo modelo, al igual que en el primero, se implementó una búsqueda en cuadrícula (Grid Search) para ajustar los parámetros, como el número de estimadores , la profundidad máxima , el número mínimo de muestras para dividir un nodo y la cantidad de trabajos(n-jobs). Cabe destacar que este proceso fue más prolongado, requiriendo casi 10 minutos, lo que lo convierte en una tarea algo pesada en términos de tiempo.

Los resultados del Grid Search indicaron que el mejor conjunto de hiper parámetros consistía en un valor de profundidad máxima de 10, un mínimo de 3 muestras para dividir un nodo y la configuración de 10 trabajos (n-jobs). Esto resultó en un 'score' de validación cruzada de 0.832, como se puede apreciar en la siguiente figura: "Fig.13" , en este caso es el más alto que obtendremos en nuestros,modelos.

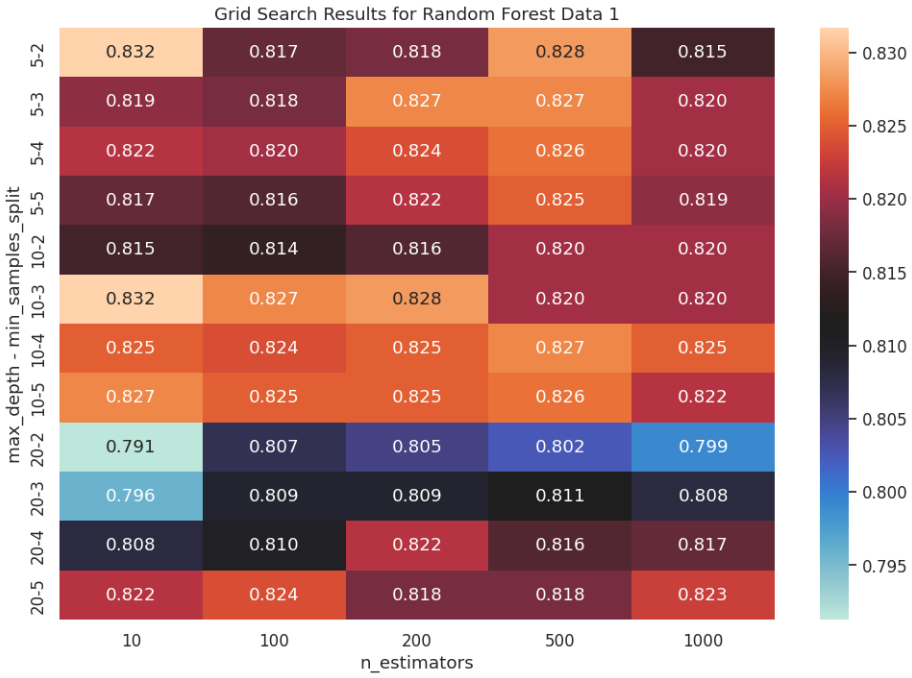


Figura 13: Random Forest-Grid Search Data 1

La matriz de confusión fue la siguiente en la figura "Fig.14" nos muestra como tenemos 529 verdaderos negativos o muertes no supervivientes, 300 verdaderos positivos o supervivientes, 20 falsos positivos y 42 falsos negativos. Lo que nos hace notar que contra el árbol, obtuvo mejores resultados para los datos de entrenamiento

Como se mencionó previamente la matriz de confusión, por sí sola, no es suficiente para determinar la eficacia del modelo en la clasificación. Junto con la matriz de confusión, se utilizaron otras métricas para una evaluación más completa del rendimiento del modelo:

Métrica	Valor
Exactitud	0.930
Precisión	0.9375
Sensibilidad	0.8771
Especificidad	0.96357
F1-Score	0.9063

Los resultados muestran que el modelo tiene una alta exactitud, con un valor del 93.0 %, lo que significa que clasifica correctamente la mayoría de las instancias. La precisión, que mide la proporción de verdaderos positivos entre las predicciones positivas, es del 93.75 %, lo que indica que las predicciones positivas son altamente confiables. La sensibilidad, que evalúa la capacidad del modelo para identificar casos positivos, es del 87.71 %, lo que sugiere que el modelo es efectivo para detectar sobrevivientes. La especificidad, que mide la capacidad de identificar correctamente casos negativos, es del 96.36 %, lo que indica una buena capacidad para predecir no sobrevivientes. Finalmente, el puntaje F1, que combina precisión y sensibilidad, es del

90.63 %, lo que resalta un equilibrio entre la capacidad de predicción positiva y negativa del modelo. Estas métricas en conjunto demuestran un rendimiento sólido del modelo de Bosque Aleatorio en la clasificación de supervivencia en este contexto.

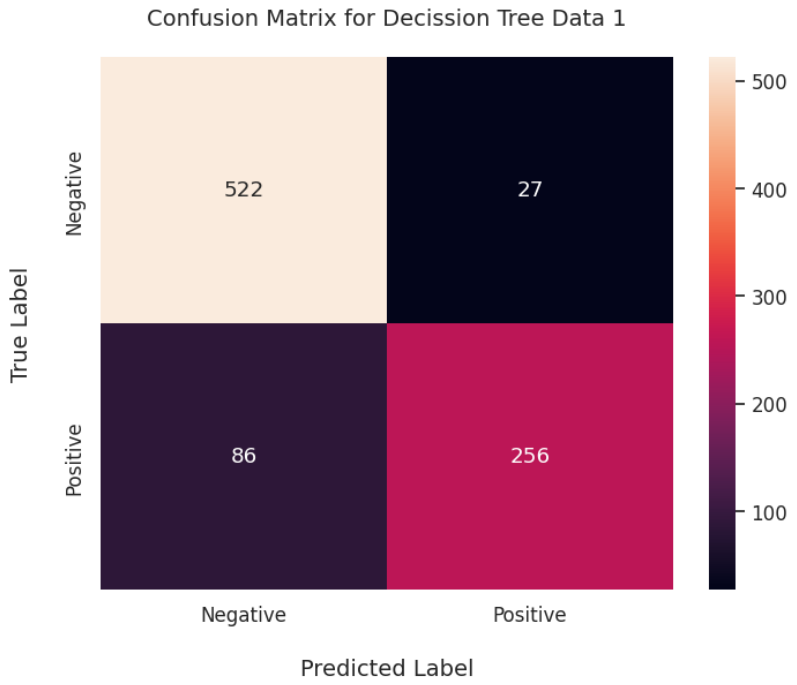


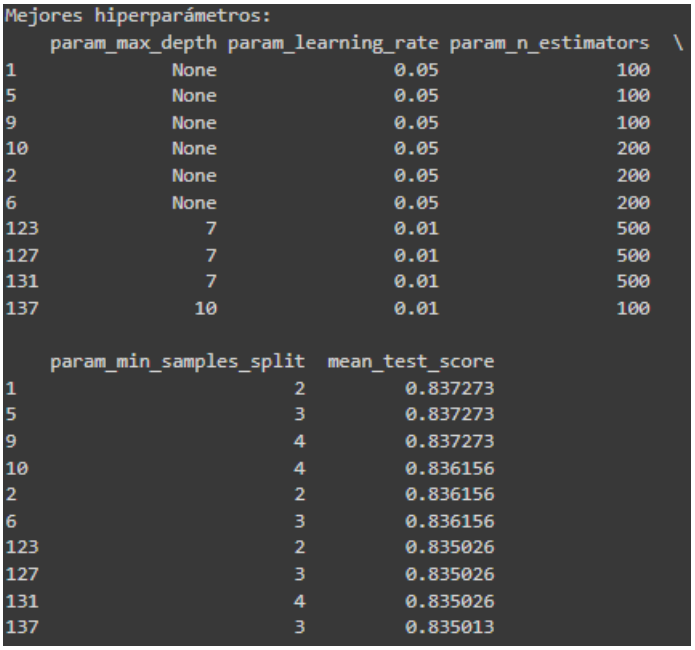
Figura 14: Random Forest-Confusion Matrix Data 1

Este modelo en kaggle se obtuvo un precision de 77.99 %

4.4.3. XGBoost

En este modelo, se emplearon hiperparámetros clave como la profundidad máxima (max depth), la tasa de aprendizaje (learning rate), el número de estimadores (n estimator), y el mínimo de muestras requeridas para dividir un nodo (min samples split) en un proceso de Grid Search. Además, se incorporó una estrategia de detención temprana (early stopping) con un límite de 10 iteraciones sin mejora con el propósito de evitar la utilización innecesaria de recursos computacionales y tiempo, dado que el proceso de búsqueda de hiper-parámetros implica probar numerosas combinaciones. La métrica de evaluación utilizada en este caso fue la pérdida logarítmica (logarithmic loss).

Como se puede ver la ”Fig.15”,se obtuvo para este caso el mejor resultado de hiper-parametros era de un score con valor de 0.837273.



```
Mejores hiperparámetros:
    param_max_depth param_learning_rate param_n_estimators  \
1              None                0.05                100
5              None                0.05                100
9              None                0.05                100
10             None                0.05                200
2              None                0.05                200
6              None                0.05                200
123               7                0.01                500
127               7                0.01                500
131               7                0.01                500
137              10                0.01                100

    param_min_samples_split  mean_test_score
1                         2         0.837273
5                         3         0.837273
9                         4         0.837273
10                        4         0.836156
2                         2         0.836156
6                         3         0.836156
123                       2         0.835026
127                       3         0.835026
131                       4         0.835026
137                       3         0.835013
```

Figura 15: XGB-Gridsearch Data 1

La matriz de confusión fue la siguiente en la figura ”Fig.16” nos muestra como tenemos 529 verdaderos negativos o muertos no supervivientes, 300 verdaderos positivos o supervivientes, 20 falsos positivos y 42 falsos negativos. Lo que nos hace notar que contra el árbol, obtuvo mejores resultados para los datos de entrenamiento.

Se utilizaron otras métricas para una evaluación más completa del rendimiento del modelo:

Métrica	Valor
Exactitud	0.9057
Precisión	0.92434
Sensibilidad	0.82163
Especificidad	0.958157
F1-Score	0.86996

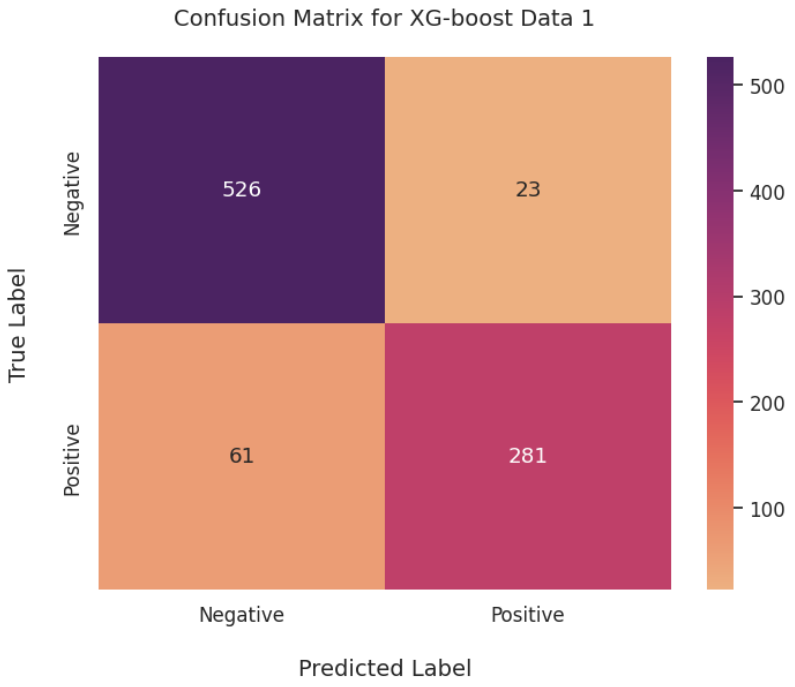


Figura 16: XG-Boost-Confusion Matrix Data 1

El XG-Boost, utilizando la data 1 obtubo un 75.559 % de precisión en kaggle.

4.5. Modelos con Preprocesamiento Data 2

En la segunda etapa de procesamiento de datos, se aplicó una técnica similar a la utilizada en la primera data, pero con una variación en la codificación de las variables “Sexo” y “Clase”. En lugar de asignar 1 a los hombres y 0 a las mujeres en la columna “Sexo”, se optó por una estrategia distinta. Considerando que se tenía información que indicaba que casi el 80 % de las mujeres sobrevivieron, se decidió asignar un valor más alto, es decir, 1, a las mujeres, con el fin de resaltar su importancia en el modelo.
De manera análoga, se implementó un enfoque similar en la columna que representaba la “Clase” de los pasajeros. Dado que se disponía de datos que indicaban que los pasajeros de primera clase tenían tasas de supervivencia significativamente mayores en comparación con los de tercera clase, se asignó un valor más alto, en este caso 3, a los pasajeros de primera clase, mientras que a los de tercera clase se les asignó un valor más bajo, específicamente 1.
Estas adaptaciones se llevaron a cabo con el propósito de reflejar de manera más precisa la influencia de estas características en la supervivencia de los pasajeros en el model

4.5.1. Árbol de decisión

Se implementó un Grid Search utilizando una validación, con un amplio rango de hiper-parámetros. Estos hiper parámetros incluyeron la profundidad máxima del árbol, el número mínimo de muestras requeridas para dividir un nodo y el número mínimo de muestras requeridas para formar una hoja.
En la ”Fig.17” se presenta el resultado del Grid Search para este modelo.Para este caso, se obtuvo un pun-

taje de validación cruzada de 0.82, utilizando los siguientes hiper-parámetros en el árbol: [Lista de hiper-parámetros utilizados]. **max depth**=6, **min samples leaf**=8 y **min samples split**==20

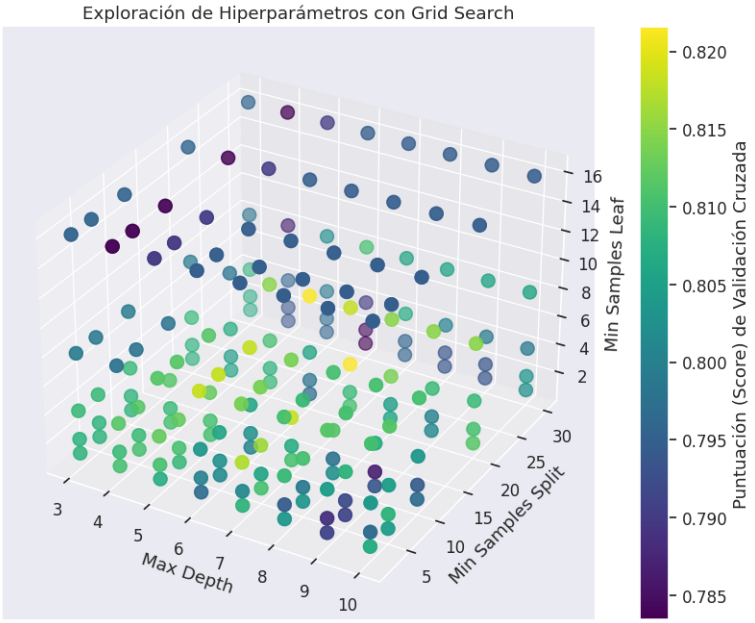


Figura 17: Tree-Gridsearch Data2

La matriz de confusión fue la siguiente en la figura "Fig.18" nos muestra como tenemos 511 verdaderos negativos o muertes no supervivientes, 259 verdaderos positivos o supervivientes, 38 falsos positivos y 83 falsos negativos.Igualmente para este caso los resultados son decentes.
Además de la matriz de confusión, se emplearon otras métricas para evaluar el rendimiento del modelo:

Métrica	Valor
Exactitud	0.8641
Precisión	0.8720
Sensibilidad	0.7573
Especificidad	0.9307
F1-Score	0.810641

Estos resultados, en conjunto con las métricas adicionales, sugieren un desempeño decente del modelo en la clasificación de sobrevivientes y no sobrevivientes en este conjunto de datos. La exactitud alcanza un valor de 0.8641, la precisión es de 0.8720, la sensibilidad es 0.7573, la especificidad es 0.9307, y el F1-Score es 0.810641.

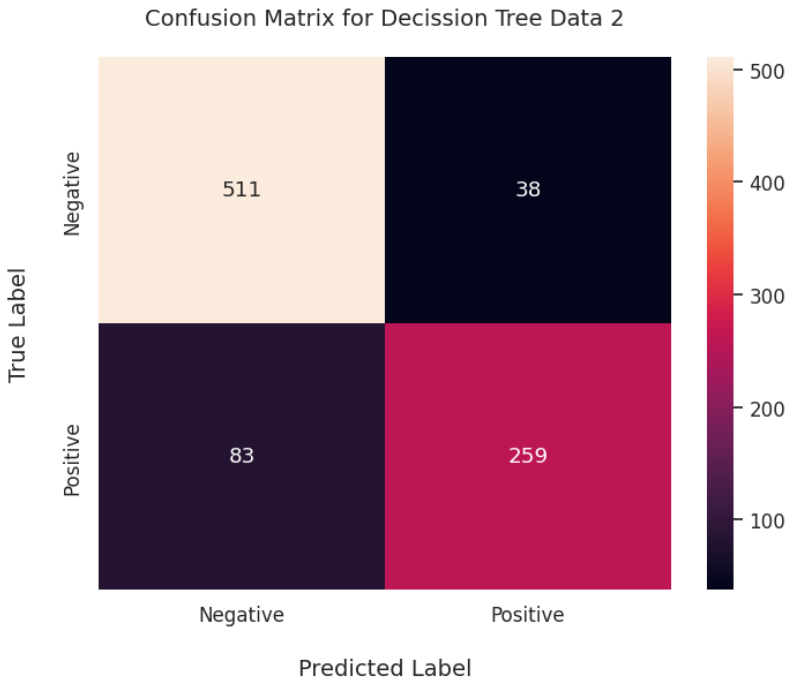


Figura 18: Tree-Confusion Matrix Data 2

El árbol de decisión con la data 2 obtuvo un 75.12 % de precision en kaggle.

4.5.2. Bosque Aleatorio

Los resultados del Grid Search indicaron que el mejor conjunto de hiper parámetros consistía en un valor de profundidad máxima de 10, un mínimo de 4 muestras para dividir un nodo y la configuración de 10 trabajos (n-jobs). Esto resultó en un 'score' de validación cruzada de 0.826, como se puede apreciar en la siguiente figura: "Fig.19" .

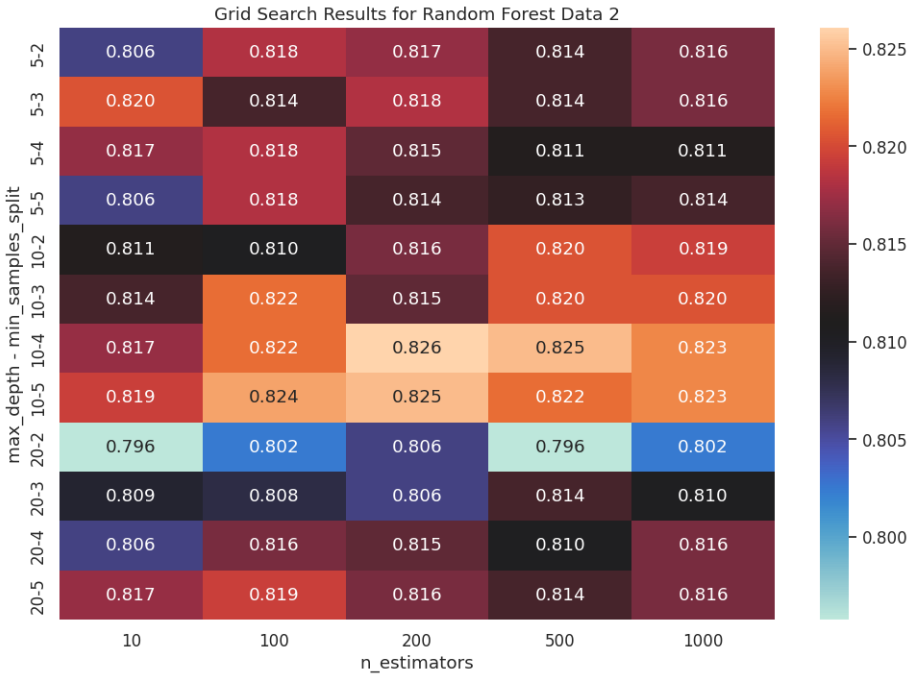


Figura 19: Random Forest-Grid Search Data 2

La matriz de confusión fue la siguiente en la figura "Fig.20" nos muestra como tenemos 534 falsos positivos o muertes no supervivientes, 290 verdaderos positivos o supervivientes, 15 falsos positivos y 52 falsos negativos. Lo que nos hace notar que contra el árbol, obtuvo mejores resultados para los datos de entrenamiento

Tenemos otras métricas para una evaluación más completa del rendimiento del modelo:

Métrica	Valor
Exactitud	0.9248
Precisión	0.9508
Sensibilidad	0.8479
Especificidad	0.97267
F1-Score	0.89644

En comparación con el árbol de decisión, este modelo obtuvo mejores resultados en los datos de entrenamiento. Las métricas adicionales respaldan aún más el rendimiento sólido del modelo, con una exactitud de 0.9248, una precisión de 0.9508, una sensibilidad de 0.8479, una especificidad de 0.97267 y un F1-Score de 0.89644. Estos valores indican una capacidad sólida para clasificar a los sobrevivientes y no sobrevivientes en el conjunto de datos.

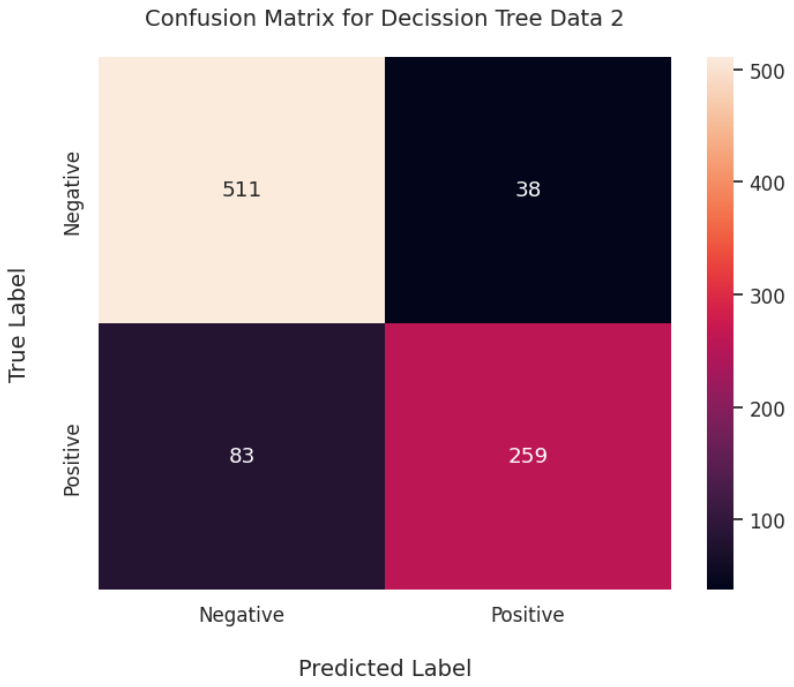


Figura 20: Random Forest-Confusion Matrix Data 2

El bosque aleatorio con la data 2 obtuvo un 77.51 % de precision en kaggle.

4.5.3. XGBoost

Como se puede observar la "Fig.21",se obtuvo para este caso el mejor resultado de hiper-parametros era de un score con valor de 0.836162.

Mejores hiperparámetros:

	param_max_depth	param_learning_rate	param_n_estimators	param_min_samples_split	mean_test_score
22	5	0.05	200	4	0.836162
18	5	0.05	200	3	0.836162
14	5	0.05	200	2	0.836162
127	7	0.01	500	3	0.836137
131	7	0.01	500	4	0.836137
123	7	0.01	500	2	0.836137
23	5	0.05	500	4	0.833915
15	5	0.05	500	2	0.833915
19	5	0.05	500	3	0.833915
10	None	0.05	200	4	0.833909

Figura 21: XGB-Gridsearch Data 2

La matriz de confusión fue la siguiente en la figura “Fig.22” nos muestra como tenemos 526 verdaros negativos o muertes no supervivientes, 282 verdaderos positivos o supervivientes, 23 falsos positivos y 60 falsos negativos. Lo que nos hace notar que contra el árbol, obtuvo mejores resultados para los datos de entrenamiento

se utilizaron otras métricas para una evaluación más completa del rendimiento del modelo:

Métrica	Valor
Exactitud	0.90684
Precisión	0.92459
Sensibilidad	0.824
Especificidad	0.9581057
F1-Score	0.87171

La tabla indica indica que, en comparación con el árbol de decisión, el modelo XGBoost ha obtenido mejores resultados en los datos de entrenamiento, especialmente en la clasificación de sobrevivientes y no sobrevivientes. Además de la matriz de confusión, se utilizaron otras métricas para evaluar el rendimiento general del modelo. El modelo XGBoost alcanzó una exactitud del 0.90684, una precisión del 0.92459, una sensibilidad del 0.824, una especificidad del 0.9581057 y un F1-Score del 0.87171. Estos valores reflejan una capacidad sólida para clasificar a los pasajeros en función de su supervivencia en el conjunto de datos.

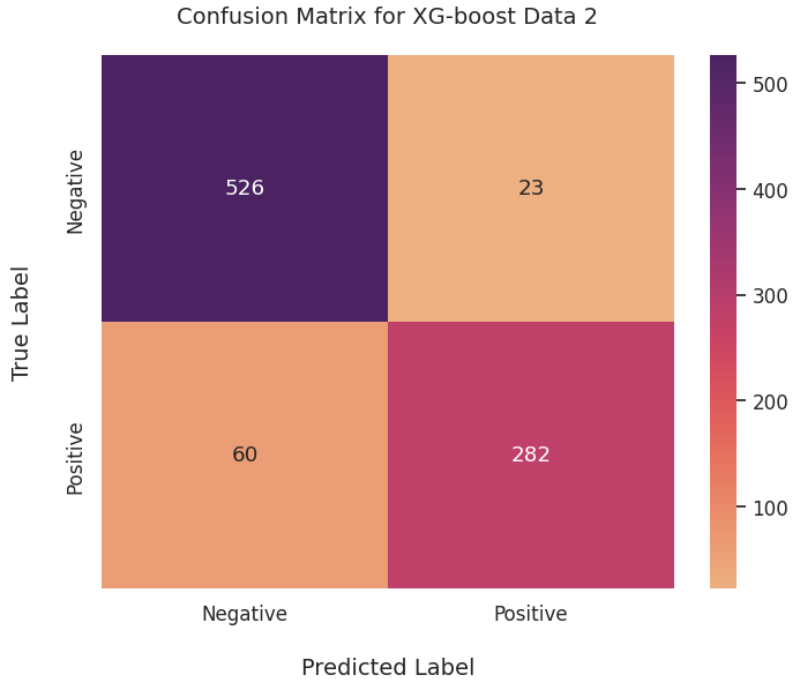


Figura 22: XG-Boost-Confusion Matrix Data 2

XG-Boost con la data 2 obtuvo un 76.076 % de precisión en kaggle.

4.6. Implementación de ingeniería de características Data 3

La ingeniería de características es un paso importante para hacer modelos de machine learning más efectivos. Un modelo solo puede ser tan bueno como los datos que usa, así que trabajar en mejorar esos datos puede marcar la diferencia en el éxito del proyecto de machine learning.

En nuestro caso, creamos nuevas características a partir de los datos existentes. Por ejemplo, clasificamos las edades en cinco grupos: infancia, adolescencia, edad adulta, mediana edad y ancianos. También hicimos algo similar con la variable de embarcación, dividiéndola en tres categorías según las opciones en los datos. Lo mismo hicimos con la clase social y si tenían o no una cabina.

Estos cambios se basaron en un análisis previo que indicaba que estas variables podrían ser importantes para la clasificación. Por ejemplo, notamos que los niños menores de 4 años tenían más probabilidades de sobrevivir, lo que sugería que esta característica podría ser relevante.

A continuación se hablara de los resultados de los tres modelos de arboles implementados para observar sus resultados:

4.6.1. Árbol de decisión

En este modelo, logramos alcanzar una precisión (Accuracy) del 0.8451 en nuestros conjuntos de datos de entrenamiento y prueba.

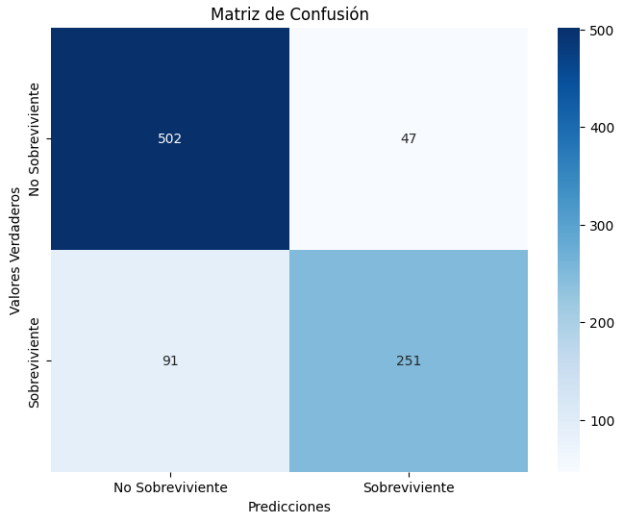


Figura 23: Tree-Confusion Matrix Data 3

Los resultados de la matriz de confusión revelan que el modelo presenta un buen desempeño en la clasificación tanto de sobrevivientes (casos positivos) como de no sobrevivientes (casos negativos) “Fig.23”. Aquí están las métricas de evaluación obtenidas:

Métrica	Valor
Exactitud	0.8451
Precisión	0.8422
Sensibilidad	0.7105
Especificidad	0.7339
F1-Score	0.7843

La alta precisión y exactitud indican que el modelo realiza una sólida tarea en la clasificación general. Sin embargo, la sensibilidad es ligeramente más baja, lo que sugiere margen para mejorar la identificación de casos de sobrevivientes. La alta especificidad muestra que el modelo es eficiente en la identificación de no sobrevivientes. El valor F1-Score es razonable, considerando el equilibrio entre precisión y sensibilidad. Resultado en Kaggle 72.96 % con los hiperparámetros mostrados "Fig.24".

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6, max_features='auto')
```

Figura 24: Tree-Gridsearch Data 3

4.6.2. Bosque aleatorio

En este modelo, alcanzamos una precisión (Accuracy) del 0.8462 en nuestros conjuntos de datos de entrenamiento y prueba.

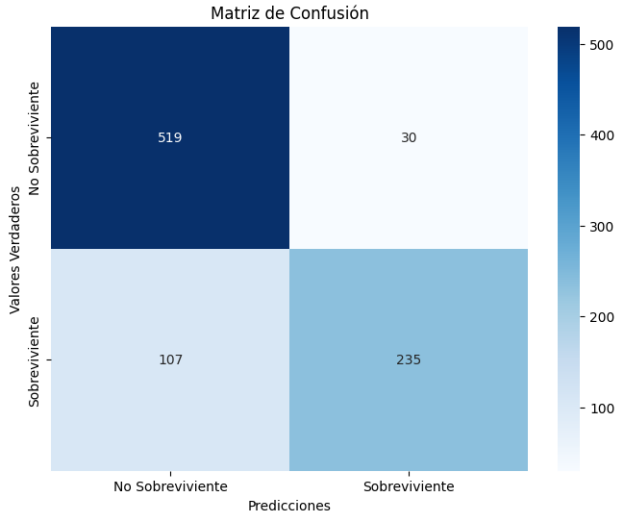


Figura 25: Random Forest-Confusion Matrix Data 3

Los resultados de la matriz de confusión evidencian que el modelo presenta un buen rendimiento en la clasificación de sobrevivientes (casos positivos) y no sobrevivientes (casos negativos) "Fig.25". Aquí están las métricas de evaluación obtenidas:

Métrica	Valor
Exactitud	0.8462
Precisión	0.8868
Sensibilidad	0.6871
Especificidad	0.9451
F1-Score	0.7743

La alta precisión y exactitud indican que el modelo realiza una sólida tarea en la clasificación general. Sin embargo, la sensibilidad es ligeramente más baja, lo que sugiere margen para mejorar la identificación de casos de sobrevivientes. La alta especificidad muestra que el modelo es eficiente en la identificación de no sobrevivientes. El valor F1-Score es razonable, considerando el equilibrio entre precisión y sensibilidad. Resultado en Kaggle 77.51 % con hiperparámetros "Fig.26".

```
RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_split=4, n_estimators=10,
                       n_jobs=10)
```

Figura 26: Random Forest-Gridsearch Data 3

4.6.3. XGBoost

En este modelo, alcanzamos una precisión (Accuracy) del 0.9158 en nuestros conjuntos de datos de entrenamiento y prueba.

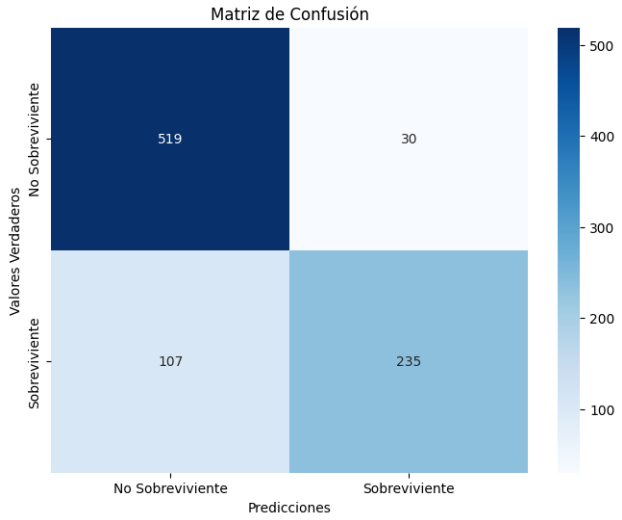


Figura 27: XGB-Confusion Matrix Data 3

Los resultados de la matriz de confusión demuestran que el modelo presenta un excelente desempeño en la clasificación de sobrevivientes (casos positivos) y no sobrevivientes (casos negativos) "Fig.27". Aquí están las métricas de evaluación obtenidas:

Métrica	Valor
Exactitud	0.9158
Precisión	0.9293
Sensibilidad	0.8450
Especificidad	0.9590
F1-Score	0.8851

La alta precisión y exactitud indican que el modelo realiza una tarea sobresaliente en la clasificación general. La sensibilidad es sólida, lo que sugiere que el modelo identifica eficazmente casos de sobrevivientes. La alta especificidad muestra que el modelo es eficiente en la identificación de no sobrevivientes. El valor F1-Score es alto, lo que refleja un equilibrio óptimo entre precisión y sensibilidad. Resultado en Kaggle 76.31 % con hiperparámetros "Fig.28".

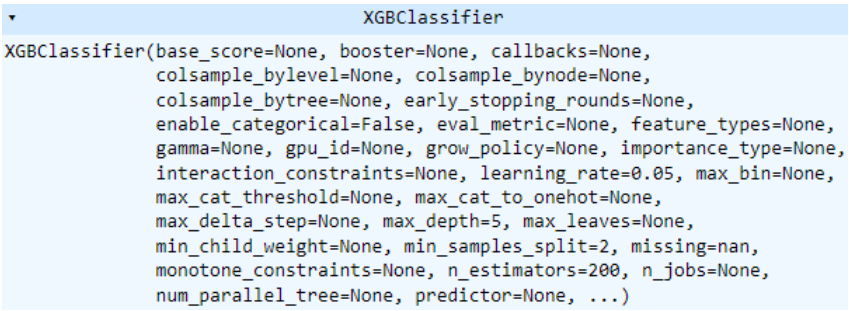


Figura 28: XGB–Gridsearch Data 3

5. Conclusiones de modelos

5.1. El mejor modelo

En el contexto de la competencia en Kaggle, el modelo de Bosque Aleatorio con Data 1 logró un destacado rendimiento, alcanzando una precisión del 77.99 %. Para lograr este éxito, se aplicó una búsqueda en cuadrícula para ajustar los hiperparámetros, como la profundidad máxima del árbol, el número mínimo de muestras requeridas para dividir un nodo y otros parámetros clave.

En cuanto al rendimiento en los datos de entrenamiento, el modelo mostró un excelente desempeño. Se encontró que el mejor conjunto de hiperparámetros incluía una profundidad máxima de 10, un mínimo de 3 muestras para dividir un nodo y la configuración de 10 trabajos paralelos (n-jobs). Esto resultó en un puntaje de validación cruzada (score) de 0.832, el más alto de todos nuestros modelos.

Además de la métrica de validación cruzada, se utilizaron diversas métricas de evaluación, incluyendo la precisión, sensibilidad, especificidad y el puntaje F1, todas las cuales indicaron un sólido rendimiento del modelo en la clasificación de la supervivencia.

En resumen, el modelo de Bosque Aleatorio con Data 1 demostró ser altamente efectivo tanto en la competencia de Kaggle como en los datos de entrenamiento, consolidando su posición como el mejor modelo en este escenario.

5.2. 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.

5.3. 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.

5.4. 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. Llegamos a esta conclusión tomando en cuenta que los árboles de decisión pueden ser un modelo susceptible a los outliers. Creemos que la discretización de los datos nos permitiría tener más control sobre el desarrollo de los árboles, permitiendonos manualmente probar diferentes intervalos en las variables continuas para observar diferentes comportamientos en los modelos. La verificación de parametros más significativos nos permitiría no tomar en cuenta parametros que contribuyen poco a la explicación en la clasificación. Es incluso posible que lo poco que contribuyen sea aleatiorio. Por último, el feature engineering nos permitiría mejorar el rendimiento del modelo al reducir la dimensionalidad, además de representar de manera más enriquecedora los patrones subyacentes a la clasificación (siempre y cuando se realice de manera correcta).

5.5. Clasificador con Potenciación de Gradiente

El modelo de clasificación utilizando potenciación de gradiente (gradient boosting) resultó ser capaz de predecir con considerable certeza. Con una precisión de setenta por ciento, es probable que realizar ajustes en los hiperparámetros además de ingeniería de características lograría que el entrenamiento del modelo lo llevara a una precisión por encima del noventa por ciento.

6. Conclusión Final

En este estudio, se evaluaron diferentes modelos de machine learning para predecir la supervivencia de pasajeros en el Titanic. El modelo de Bosque Aleatorio con Data 1 demostró ser el más efectivo, logrando una precisión del 77.99 % en la competencia de Kaggle y un puntaje de validación cruzada de 0.832 en los datos de entrenamiento. Este modelo se destacó tanto en la clasificación de sobrevivientes como en la de no sobrevivientes.

En contraste, las Redes Neuronales presentaron un desempeño deficiente, sugiriendo la necesidad de ajustes en la implementación y exploración de técnicas adicionales como el backpropagation.

El algoritmo de Vecinos más Cercanos mostró limitaciones y se sugiere su reconsideración para futuras iteraciones del estudio.

Los modelos basados en Árboles ofrecieron buenos resultados, pero se identificaron áreas de mejora, como la implementación de Grid Search para ajuste de hiperparámetros y técnicas como la discretización de datos, eliminación de outliers y feature engineering para potenciar el rendimiento.

Finalmente, el clasificador con Potenciación de Gradiente demostró una capacidad de predicción significativa, con una precisión del 70 %. Se sugiere que ajustes en los hiperparámetros y una ingeniería de características más detallada podrían elevar la precisión por encima del 90 %.

En resumen, este estudio proporciona valiosos insights sobre los modelos de machine learning aplicados a la predicción de supervivencia en el Titanic, destacando la efectividad del Bosque Aleatorio y señalando áreas de mejora para futuras investigaciones en este campo.

Referencias

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

[2] IBM, *¿Qué es KNN?* s. f. dirección: `https://www.ibm.com/mx-es/topics/knn`.

[3] IBM, *(s. f.). ¿Qué es un árbol de decisión?*
Dirección: `https://www.ibm.com/mx-es/topics/decision-trees`.

[4] IBM, *(s. f.). ¿Qué es un bosque aleatorio?*
Dirección: `https://www.ibm.com/mx-es/topics/random-forest`.

[5] IBM, *(s. f.). What are neural networks?* Dirección: `https://www.ibm.com/topics/neural-networks`.

[6] A. Saini, *(2023). Gradient Boosting Algorithm: A complete guide for beginners. Analytics Vidhya.*
dirección: `https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/`.

[7] TIBCOSoftwa, *(s. f.). ¿Qué es la regresión logístic.*
dirección: `https://www.tibco.com/es/reference-center/what-is-logistic-regression`.