INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

ESCUELA DE INGENIERÍA Y CIENCIAS

Inteligencia artificial avanzada para la ciencia de datos I TC3006C.102

Modelo de Machine Learning para predicción de supervivencia con datos del Titanic

Machine Learning Model for survival prediction with Titanic data.

Autores - Equipo 5

Francisco Salas Porras - A01177893

J. Andrés Orantes Guillén - A01174130

Jackeline Conant Rubalcava - A01280544

Luis Mario Lozoya Chairez - A00833364

J. Eduardo Corrales Cardoza - A01742328

Profesores

Alfredo Esquivel Jaramillo

Jesús Adrián Rodríguez Rocha

Antonio Carlos Bento

Julio Antonio Juárez Jiménez

Frumencio Olivas Alvarez

Mauricio González Soto

Hugo Terashima Marín

Monterrey, Nuevo León. 08 Septiempre 2024

Índice

Resumen

El presente documento ahonda en la predicción de la supervivencia de los pasajeros del Titanic utilizando técnicas de aprendizaje automático y una base de datos histórica. A partir de los datos disponibles, se inició con la exploración y limpieza de los datos, abordando valores faltantes y realizando la categorización de variables. Posteriormente, se implementaron técnicas de feature engineering para mejorar la capacidad predictiva del modelo en desarrollo.

Para la selección del modelo, se investigaron y compararon diversos algoritmos de clasificación binaria, estos son *Regresión Logística*, *Support Vector Machine*, *Red neuronal*, *Random Forest* y *XGBoost*. Los modelos fueron evaluados utilizando métricas como exactitud, Puntaje F1 y Puntaje ROC-AUC. Los resultados indicaron que el modelo con mejor rendimiento fue el Random Forest, el cual será refinado para optimizar su desempeño.

Abstract

This paper delves into the prediction of the survival of Titanic passengers using machine learning techniques and a historical database. Based on the available data, the data was explored and cleaned, addressing missing values and categorizing variables. Subsequently, feature engineering techniques were implemented to improve the predictive capacity of the model under development.

For model selection, various binary classification algorithms were investigated and compared, these are *Logistic Regression*, *Support Vector Machine*, *Neural Network*, *Random Forest* and *XGBoost*. The models were evaluated using metrics such as accuracy, F1-Score, and ROC-AUC Score. The results indicated that the model with the best performance was Random Forest, which will be refined to optimize its performance.

1. Introducción

El trágico hundimiento del RMS Titanic, en la noche del 14 al 15 de abril de 1912, es uno de los desastres marítimos más infames de la historia. De las aproximadamente 2,224 personas que abordaron, más de 1,500 acabaron perdiendo la vida. Debido a este gran número de gente y la pérdida de registros, no se tienen todos los datos de los pasajeros. Sin embargo, todavía se tienen bases de datos con información de algunos pasajeros del Titanic. Dentro de la plataforma de Kaggle se cuenta con una de éstas, con el propósito de analizar y explorar las características que podrían haber influido en las probabilidades de supervivencia.

En el presente reporte, se emplearán herramientas avanzadas de aprendizaje automático para llevar a cabo un análisis exhaustivo de los datos históricos disponibles sobre los pasajeros del Titanic. Esta metodología no solo presenta un interés académico, sino que también ofrece una oportunidad única para evaluar cómo un manejo más informado y un análisis meticuloso

podrían haber mejorado la gestión de la crisis durante el evento. Mediante el uso de estas técnicas de modelado, este estudio busca proporcionar una comprensión más profunda de las dinámicas sociales y las decisiones que afectaron las probabilidades de supervivencia durante la tragedia. Este enfoque pretende contribuir significativamente al cuerpo de conocimiento en análisis de datos aplicados a estudios históricos y emergencias.

2. Planteamiento del problema

El dataset utilizado para éste reporte contiene diferentes características de los pasajeros a bordo del Titanic. Contiene información general que se describirá y analizará más adelante, siendo la más importante la variable objetivo que se requiere predecir: la supervivencia del pasajero. La investigación se centrará en encontrar un modelo que pueda reducir lo más posible el error de categorización y prediga correctamente si el pasajero sobrevive o no.

3. Conjunto de datos

En la tabla 1 se pueden observar las variables contenidas en el dataset de Kaggle.
Otras consideraciones que también son importantes de mencionar:

- La edad es fraccionaria si es menor de 1. Si la edad es estimada, se presenta en la forma xx.5
- Las relaciones familiares de 'Sibsp' son definidas como:
  - Hermano = hermano, hermana, hermanastro, hermanastra.
  - Esposo = esposo.
- Las relaciones familiares de 'Parch' son definidas como:
  - Padre = madre, padre.
  - Hijo = hija, hijo, hijastra, hijastro.
  - Algunos niños viajaban solo con una niñera, por lo tanto parch=0 para ellos.

4. Limpieza del conjunto de Datos

En esta etapa inicial se hará un anáisis de los datos proporcionados y serán preparados para que puedan ser procesados por los modelos. Aunque hay modelos que son robustos a presencia de datos nulos, *outliers* y otras características, es preferible que se trabajen estos datos cuando sea posible y así poder probar distintos tipos de modelos. También se realizará feature engineering para reducir y manipular la cantidad de variables independientes y proporcionar solo la información más importante al modelo.

Variable	Definición	Key
Survival	Supervivencia después del hundimiento	Variable categórica booleana: 0 = No, 1 = Sí
Pclass	Clase socioeconómica	Variable categórica ordinal: 1 = 1ª clase, 2 = 2ª clase, 3 = 3ª clase
Sex	Sexo de la persona	Variable categórica booleana: 0 = Femenino, 1 = Masculino
Age	Edad en años	Variable numérica continua: Tipo: float Rango: [0.17, 80]
Sibsp	Número de hermanos / esposos a bordo	Variable numérica discreta: Tipo: float Rango: [0, 8]
Parch	Número de padres / hijos a bordo	Variable numérica discreta: Tipo: float Rango: [0, 9]
Ticket	Número de boleto	Variable categórica nominal: Tipo: string
Fare	Tarifa del pasajero	Variable numérica continua: Tipo: float Rango: [0, 512.3292]
Cabin	Número de cabina	Variable categórica nominal: Tipo: string
Embarked	Puerto de embarque	Variable categórica nominal: C = Cherburgo, Q = Queenstown, S = Southampton

Tabla 1: Descripción de variables del conjunto de datos del Titanic

4.1. Análisis y visualización

Antes de poder entrenar nuestro modelo predictivo, se requiere realizar un análisis y limpieza de los datos. Se extrajeron todos los datos del Titanic de entrenamiento y evaluación para poder analizarlos dentro de nuestro repositorio de Github. Dentro de éste repositorio se encuentran como 'train.csv' y 'test.csv' para los datos de entrenamiento y evaluación respectivamente. El análisis y limpieza se llevó a cabo en el archivo de Jupyter Notebook 'InicioTitanic.ipynb'

Al descargar y analizar los datos, uno de las primeras observaciones fueron que algunos de los datos contenían valores nulos. Más a detalle, se observó que las variables de 'Survived', 'Age' y 'Cabin' eran los valores más altos, con 418, 263 y 1014 respectivamente. De 'Survived' era esperado pues éstas eran las etiquetas vacías del test set, pero las demás deberán ser reemplazadas. Con valores menos altos, pero que también se tomarán en cuenta, 'Fare' y 'Embarked' tuvieron 1 y 2 valores nulos respectivamente.

Otras observaciones que también se tomaron en cuenta fueron los familiares y títulos de las

personas. Las variables de 'Sibsp' y 'Parch' mostraban tener una pobre cantidad de datos, con muchos de los pasajeros teniendo ninguno de ambos. Por otra parte, todos los nombres dentro del dataset incluían el título de la persona. Los diferentes títulos contenidos en los nombres fueron: 'Mr', 'Mrs', 'Miss' 'Master', 'Don', 'Rev', 'Dr', 'Mme', 'Ms', 'Major', 'Lady', 'Sir', 'Mlle', 'Col', 'Capt', 'the Countess', 'Jonkheer' y 'Dona'

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
1304	1305	NaN	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
1305	1306	NaN	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
1306	1307	NaN	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
1307	1308	NaN	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
1308	1309	NaN	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

1309 rows × 12 columns

Figura 1: Dataframe con datos de entrenamiento y evaluación

4.2. Limpieza y transformación

Previo a la separación de los conjuntos de datos de entrenamiento y evaluación, se inició la limpieza de los datos llenando las columnas con valores faltantes o nulos. Utilizando la función de la moda, que tiene la finalidad de seleccionar el valor que aparece con mayor frecuencia en el conjunto de datos, pudimos rellenar los valores faltantes de 'Fare' y 'Embarked' con 8.05 y 'S', respectivamente. Lo anterior se hizo debido a la poca cantidad de valores nulos en estas columnas. Para el llenado de los datos faltantes en la edad, se analizaron y utilizaron las funciones de la media y la mediana para visualizar mejor los datos, generando inicialmente gráficos de medidas de tendencia central antes de completar los valores faltantes.

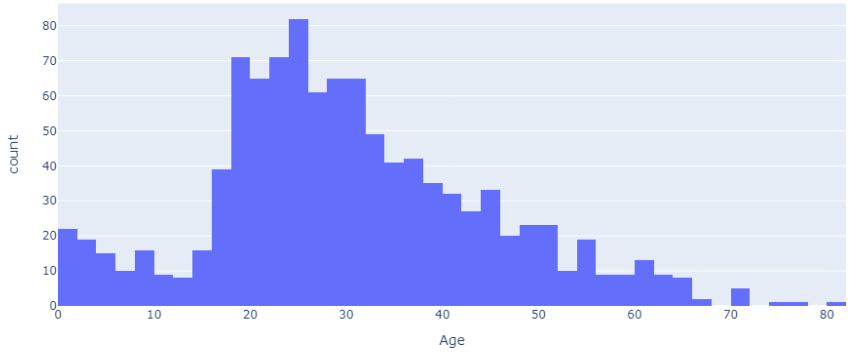

Figura 2: Histograma de las edades de los pasajeros.

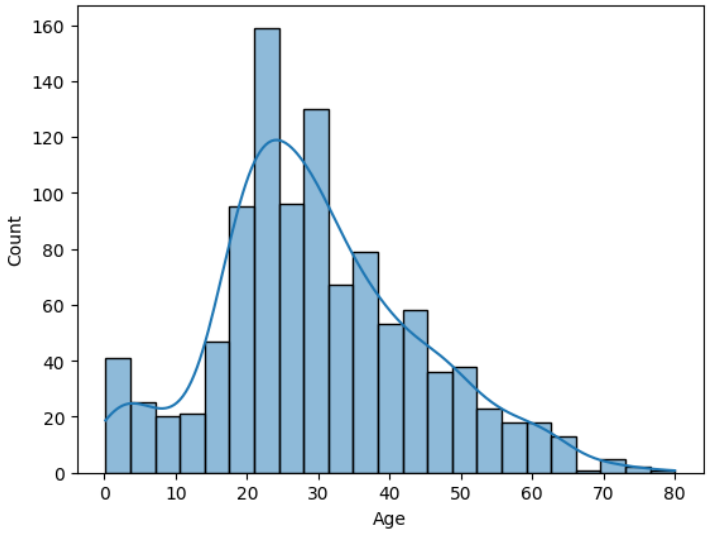

Figura 3: Histograma con línea de tendencia de las edades de los pasajeros.

Adicionalmente a los gráficos que se pueden visualizar previamente, los cuales muestran las edades de los pasajeros, se obtuvieron las medidas de tendencia central presentes en la columna de 'Age'. Como se mencionó previamente, se utilizó la función de la media para calcular el promedio de todas las edades, que resultó ser de 29.88 años, un valor que se refleja claramente en el histograma. La mediana de las edades de los pasajeros se estableció en 28 años, mientras que la moda fue de 24 años. Además, se determino que el cuartil inferior es de 21 años y el cuartil superiror es de 39 años. Estos datos sugieren que la mayoría de los pasajeros se encontraban en un rango de edad 21-39 años. Adicionalmente, se identificó que el pasajero con mayor edad a bordo del Titanic tenía 80 años y el de menor edad tenía menos de un año de edad.

Debido a la falta de normalidad en la distribución se determinó el uso de diferentes medidas de desviación estándar para la imputación de datos a través de los títulos que contenían los nombres de los pasajeros. Se hizo una visualización de las edades que se puede observar en

la figura 4.

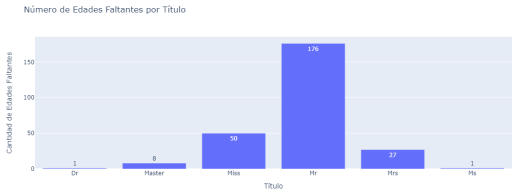

Figura 4: Gráficos de Barras de Edades faltantes por título

Se puede observar que las edades faltantes se encuentran distribuidas en 6 títulos: ”Dr”, ”Master”, ”Miss”, ”Mr”, ”Mrs” y ”Ms”. Donde Mr es el que cuenta con un mayor número de edades faltantes. Teniendo este conocimiento se procedió a generar gráficas para conocer la distribución de las edades faltantes en función de los títulos.

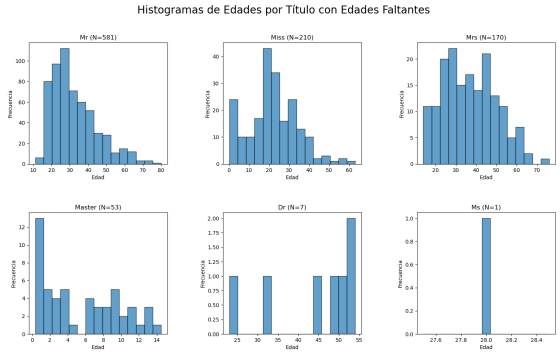

Figura 5: Histogramas de Edades por Título con edades faltantes.

Basándose en la figura 5 se seleccionaron los métodos de imputación para cada título. Se buscó que los datos imputados no alteraran la distribución inicial de los datos, por lo que se decidió utilizar la mediana móvil para los títulos “Mr”, “Miss” y “Mrs”, dado que cuentan con más de 100 datos. Además, en estos títulos, parecen existir valores que pueden sesgar la imputación, por lo que la mediana, al ser robusta, permite que el sesgo no se proyecte al imputar los valores.

Para los títulos ”Master” y ”Dr”, se optó por el uso de la media móvil, ya que no cuentan con suficientes datos para aplicar la mediana. Además, se considera que la distribución de los datos no se verá influida por la imputación con esta medida, y la cantidad de datos que serán imputados es baja. Para el título “Ms” se utilizó la moda debido a solo haber un dato. Tras haber terminado la imputación de los datos, se generó la gráfica 6, la cual muestra la distribución de las edades con los datos imputados.

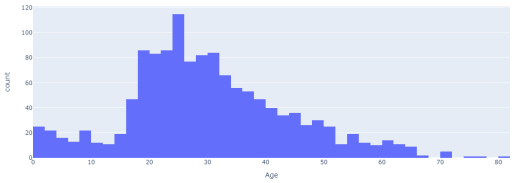

Figura 6: Histograma de las edades de los pasajeros con datos faltantes imputados

Al terminar la imputación de los datos, se procedió con el análisis de variables para realizar su transformación y eliminar aquellas que se consideraron irrelevantes. Se generó la variable #_relatives, la cual indica la cantidad de parientes de la persona que se encuentran a bordo del Titanic. Esta variable se generó sumando las variables SibSp y Parch. Se realizó la eliminación de variables que no se consideraron relevantes por los datos contenido en ellas, estas fueron: ”Name”, ”Ticket”, ”Title” y ”Fare”. Teniendo esto, se aplicó el método ’get dummies’ en las variables ”Embarked” y ”Sex”, convirtiendo estos valores categóricos en valores booleanos, simplificando de esta manera los datos al dividirlos en columnas y permitiendo que posteriormente puedan ser procesados por los algoritmos de Machine Learning que se vayan a utilizar.

Basándose en la limpieza y adaptación de valores previamente realizada, se seleccionaron los valores más importantes que se utilizarán en el modelo de aprendizaje automático mediante la creación de un mapa de calor, que tiene el propósito de mostrár la correlación entre los datos ”Pclass”, ”Age”, ”SibSp”, ”Parch”, ”#_Relatives”, ”Embarked_C”, ”Embarked_Q”, ”Embarked_S”, ”Sex_f” y ”Sex_m” con el valor ”Survival” de los pasajeros.

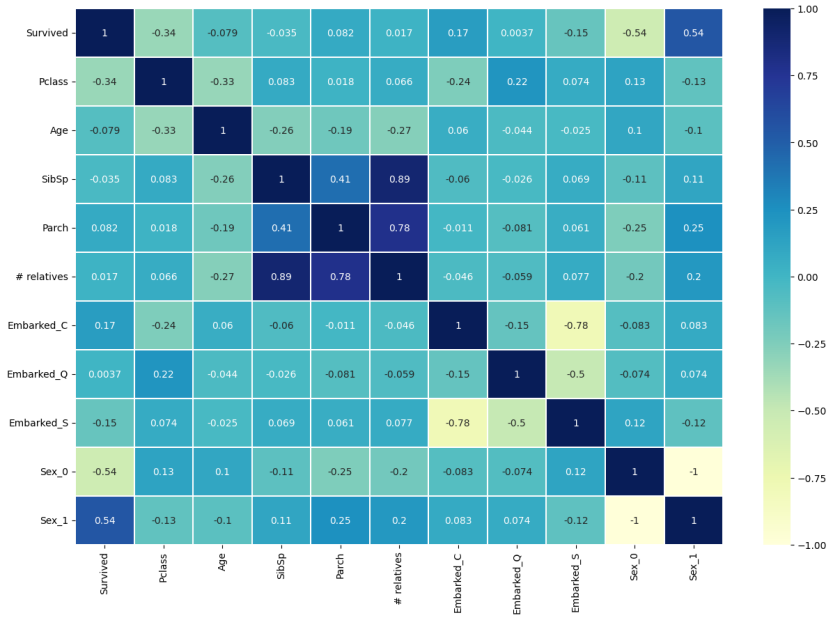

Figura 7: Mapa de calor para la comparación de valores.

Se pudo notar que el foco de la correlación se centro en la variable objetivo, ”Survived”, y

su relación con el resto de las variables. Las variables que mostraron las correlaciones más significativas, tanto positivas como negativas, con la variable objetivo fueron: ”Sex_0”, ”Sex_1”, ”Pclass”. Sin embargo, estas observaciones iniciales resultaron insuficientes para una selección definitiva de variables. Se realizó una investigación sobre técnicas que permitieran una mejor obtención de variables clave. De esta investigación se obtuvo un selector secuencial que proporcionó las características que más beneficiarían a un modelo seleccionado. En base al mapa de calor 7 y el selector secuencial se procedió a hacer la eliminación de las características ”SibS” y ”Parch”, puesto que, de acuerdo al selector secuencial, no resultaban relevantes. Finalmente se guardó y generó la base de datos final, que se utilizará en la siguiente etapa de Selección, configuración y entrenamiento del modelo.

4.3. Conjunto de Datos final

Variable	**Definición**	**Key**
Survival	Supervivencia después del hundimiento	Variable categórica booleana: 0 = No, 1 = Sí
Pclass	Clase socioeconómica	Variable categórica ordinal: 1 = 1ª clase, 2 = 2ª clase, 3 = 3ª clase
Sex_f	Sexo femenino	Variable categórica booleana: 0 = No, 1 = Sí
Sex_m	Sexo masculino	Variable categórica booleana: 0 = No, 1 = Sí
Age	Edad en años	Variable numérica continua: Tipo: float Rango: [0.17, 80]
#_relatives	Suma de Sibsp y Parch	Variable numérica discreta: Tipo: float Rango: [0, 10]
Embarked_C	Puerto de embarque Cherburgo	Variable categórica booleana: 0 = No, 1 = Sí
Embarked_Q	Puerto de embarque Queenstown	Variable categórica booleana: 0 = No, 1 = Sí
Embarked_S	Puerto de embarque Southampton	Variable categórica booleana: 0 = No, 1 = Sí

Tabla 2: Descripción de variables finales para entrenamiento

El conjunto final mostrado en la tabla 2, cuenta con 9 columnas. Dentro de este conjunto, la columna “Survival” es la variable a predecir, mientras que las otras 8 son las variables predictoras. El conjunto incluye 6 variables categóricas booleanas, una categórica ordinal, una numérica continua y una numérica discreta. Este conjunto se exportó a un archivo CSV para

ser utilizado en la siguiente fase. Es importante recordar que este conjunto proviene de los datos de los archivos iniciales “train.csv” y “test.csv”, por lo que se debe dividir para que pueda ser utilizado por la siguiente etapa.

Primero se divide en los datos originales de entrenamiento y prueba. Ya que se mantuvieron ordenados, simplemente con seleccionar los primeros 891 datos se puede reobtener los datos de entrenamiento. De igual manera los 418 datos restantes siguen siendo los datos de prueba originales. Ahora éstos datos pueden ser divididos entre sus variables características y la variable objetivo, obteniendo así los datasets necesarios para poder entrenar y probar los modelos.

5. Selección, configuración y entrenamiento del modelo

El desarrollo de esta etapa requiere la investigación de los modelos que pueden ser utilizados en el contexto del problema. Dado que se busca clasificar la supervivencia de los pasajeros, este es un problema de clasificación binaria, por lo que los modelos que se elegirán tienen que ser capaces de enfrentarse efizcamente a este problema[6].

Debido a que los datos de entrenamiento contienen una variable objetivo, se busca utilizar técnicas de aprendizaje supervisado. Como primer acercamiento se pensó en utilizar regresión logística y *Support Vector Machine*, debido a que estos están diseñados especialmente para problemas de clasificación binaria, esperando que tengan un buen desempeño en estos algoritmos [2]. Además se investigaron otros modelos para probar su eficacia en el problema, estos son *Random Forest*, XG Boost y Red Neuronal [5].

Recapitulando los modelos que se tomarán en cuenta:

- Regresión Logística
- Support Vector Machine
- Red neuronal
- Random Forest
- XG Boost

5.1. Selección

A continuación se presenta una descripción de los modelos que se tomarán entrenarán para, posteriormente, seleccionar el que cuente con mejores puntajes en clasificación.

5.1.1. Regresión Logística

Esta técnica de Machine Learning permite una clasificación de datos rápida, pues puede procesar grandes volúmenes de datos con pocos recursos. Este modelo ofrece flexibilidad,

puesto que puede realizar clasificación de datos tanto binaria como multiclase. Además este modelo tiene un buen rendimiento para predicciones binarias, por lo que resulta verdaderamente adecuado para el problema de clasificación que se le pondrá[16]. Este modelo será implementado a través de la librería Sci-kit Learn, la función que define el modelo cuenta con hasta 15 parámetros [11] de los cuales serán configurados más adelante.

5.1.2. Support Vector Machine

Es una técnica de clasificación supervisada que permite alta dimensionalidad. Ésta técnica busca eficientizar la clasificación de datos al dividir los datos con una línea o hiperpláno [7]. Es muy versátil, eficiente en memoria y es comúnmente utilizado para problemas donde hay menos instancias que features. Funciona a través de los kernels, los cuales pueden definir cómo se hará la división entre los vectores de soporte del dataset, pudiendo incluso elevar los datos a dimensiones donde puedan ser divididos más fácilmente. Este modelo será implementado a través de la librería Sci-kit Learn, la función que define el modelo cuenta con hasta 15 parámetros [15] de los cuales serán configurados más adelante.

5.1.3. Red Neuronal

Este método de inteligencia artificial utiliza aprendizaje inspirado en el cerebro humano. Basado en el modelo del perceptrón, el cual buscaba emular el comportamiento de una nuerona, la red neuronal contiene múltiples de éstos perceptrones interconectados entre sí para poder hacer una mejor predicción [13]. Estos se pueden modificar entre distintas capas y números de neuronas para poder llegar a distintas arquitecturas y mejor funcionamiento del modelo. Este modelo será implementado a través de la librería Sci-kit Learn, la función que define el modelo cuenta con hasta 23 parámetros [12] de los cuales serán configurados más adelante.

5.1.4. Random Forest

Técnica de machine learning de ensamble basada en la implementación de múltiples árboles de decisión para luego combinarlas y llegar a la salida deseada. Algunas de los beneficios de este modelo es que tiene menor riesgo de overfitting, es muy flexible y también puede ayudar a determinar la importancia de los features [8]. Este modelo será implementado a través de la librería Sci-kit Learn, la función que define el modelo cuenta con hasta 19 parámetros [14] de los cuales serán configurados más adelante.

5.1.5. XG Boost

También es una técnica de machine learning de ensamble basada en la utilización de árboles de decisión. Es una versión escalada y paralelizable del árbol de decisión con impulso de gradiente, el cual se centra en generar múltiples modelos débiles de los datos de los cuales

se basan en el gradiente del error para mejorar su rendimiento. Esto permite minimización del bias y del underfitting [10]. Este modelo será implementado a través de la librería XGBoost, la función que define el modelo cuenta con hasta 40 parámetros [17] de los cuales serán configurados más adelante.

5.2. Configuración

Para la realización de la configuración es importante notar que los parámetros configurados no son todos los disponibles en la función. Los parámetros utilizados fueron investigados con el objetivo de optimizar el modelo para este problema en específico, y los parámetros que no se mencionan se mantienen con los valores predeterminados. Se debe mencionar que, en los modelos Red Neuronal y *Random Forest*, se utilizó el parámetro *random_state* debido a que el resultado del modelo cambia si no se coloca este parámetro. Ésto para poder reproducir la implementación correctamente. A continuación se presentan las configuraciones para cada uno de los algoritmos:

Regresión Logística

3 son los parámetros que se configurarán fuera de lo predeterminado:

- **class_weight**: Este parámetro se encarga de asignar pesos a las clases. A este parámetro se le ha colocado "**balanced**", para que los pesos sean inversamente propocionales a la frecuencia de las clases.
- **solver**: Este parámetro se refiere al algoritmo de optimización del problema que se usa. Se asignó "**liblinear**"pues sirve para conjuntos de datos pequeños y es para clasificación binaria[11].
- **max_iter**: Este parámetro se refiere al número máximo de iteraciones para que los solvers converjan. Se asignó el valor **500**.

Support Vector Machine

La configuración de este algoritmo es la siguiente:

- **C**: Este es un parámetro de regularización que controla el equilibrio en los límites de decisión y la clasificación correcta de los datos[9]. El valor dado a este parámetro es **10**.
- **gamma**: Este parámetro se encarga de ajustar el modelo al conjunto de datos, entre mayor el valor asignado, mayor el ajuste a los datos. El valor asignado fue **0.1**, buscando que el algoritmo se mantenga flexible en la definición de fronteras y no se llegue a overfitting.
- **class_weight**: Este parámetro se encarga de asignar pesos a las clases. A este parámetro se le ha colocado "**balanced**", para que los pesos sean inversamente proporcionales a la frecuencia de las clases.

- **Kernel**: Este parámetro define la función que la SVM usa para organizar y entender de mejor manera los datos. El *kernel* utilizado fue **rbf**, puesto que este es el predeterminado.

Red Neuronal

La implementación de la red neuronal cuenta con la siguiente configuración de parámetros:

- **hidden_layer_sizes**: Este parámetro define la cantidad de capas ocultas para el modelo y el número de neuronas en cada una. Las capas definidas son la siguientes: **(64, 128, 128, 64)**. Por lo que el modelo tiene 4 capas ocultas con 64, 128, 128 y 64 nodos respectivamente.
- **activation**: Este parámetro define las funciones de activación de las capas ocultas del modelo[12]. Se definió la función **logistic** para las capas, debido al tipo de problema de clasificación que se busca realizar.
- **max_iter**: Este parámetro se refiere al número máximo de iteraciones para que los solvers converjan. Se asignó el valor **500**.

Random Forest

Para el algoritmo Random Forest se tendrá la siguiente configuración:

- **n_estimators**: Este parámetro define la cantidad de árboles que se generararán. Se establecieron **200**, que es el doble del valor predeterminado.
- **max_depth**: Este parámetro se refiere a la profundidad máxima de los árboles. Se colocó un valor límite de **10**, con el objetivo de evitar *overfitting*.
- **min_samples_split**: Este parámetro establece el número mínimo de muestras requeridas para dividir un nodo interno. Se asignó el valor **10** para evitar divisiones en nodos con pocas muestras.

XG Boost

Para el algoritmo XG Boost la configuración es la siguiente:

- **n_estimators**:Este parámetro define la cantidad de árboles que se generarán. Se establecieron **300**, lo que es un valor típico del modelo.
- **max_depth**: Este parámetro se refiere a la profundidad máxima de los árboles. Se colocó un valor límite de **12**.
- **learning_rate**: Este parámetro reduce el impacto de cada uno de los árboles a generar. Se colocó **0.01**.

- **objective=**: Este parámetro define qué tipo de problema se busca resolver y cómo se va a ajustar el modelo. Se seleccionó **binary:logistic**, donde se utiliza regresión logística para classificar de forma binaria.[4]

5.3. Entrenamiento

Para decidir cual será el modelo que se implementará en esta etapa, para la predicción de la supervivencia en el conjunto de prueba, se tomarán en cuenta los resultados obtenidos en diferentes métricas. Es importante notar que las métricas permiten encontrar los puntos fuertes y débiles del modelo, sin embargo, no hay un métrica específica que permita saber directamente cual es la calidad general de un modelo[6]. Las métricas que se utilizarán para la evaluación son: Matriz de Confusión, exactitud, Puntuación F1 y ROC-AUC Score. La matriz de confusión es una métrica que permite una visualización matricial de los resultados de predicciones de los modelos, en la diagonal principal de la matriz se observan los valores clasificados correctamente, mientras que en los otros elementos representan predicciones verdaderas que son falsas y predicciones falsas que en realidad son verdaderas. La exactitud, se refiere a la frecuencia con la que el clasificador es correcto. La Puntuación F1 calcula la media armónicado de la precisión y exhaustividad[3]. El puntaje ROC-AUC mide la habilidad del modelo de distinguir entre clases, entre mejor puntaje ROC, hay mejor entendimiento de las clases[1]. Dentro de la tabla 3 se desglosaron los valores de las matrices de confusión en Verdadero Positivo(VP), Verdadero Negativo(VN), Falso Positivo(FP) y Falso Negativo(FN).

Modelo	VP	VN	FP	FN	Exactitud	Puntaje F1	Puntaje ROC AUC
Regresión Logística	62	84	21	12	0.815	0.789	0.818
Support Vector Machine	61	83	22	13	0.804	0.777	0.807
Red Neuronal	48	94	11	26	0.793	0.721	0.771
Random Forest	54	95	10	20	0.832	0.782	0.817
XG Boost	54	93	12	20	0.821	0.771	0.807

Tabla 3: Métricas de cada Modelo

En base a la tabla 3 se hará un análisis de cada métrica obtenida por modelo.

5.3.1. Exactitud

La exactitud es fundamental, pues representa las predicciones correctas de entre el total de predicciones. En las evaluaciones se puede notar que el algoritmo Random Forest es el que cuenta con mayor exactitud, al tener un valor de 0.832, seguido por el XGBoost con 0.821. La regresión logística y el SVM tienen exactitudes ligeramente inferiores (0.815 y 0.804, respectivamente), mientras que la Red Neuronal tiene la exactitud más baja con 0.793.

5.3.2. Puntaje F1

En este puntaje se puede notar el mejor desempeño de parte de la Regresión Logística con 0.789, y después sigue Random Forest con 0.782. Mientras que la Red Neuronal cuenta con el puntaje más bajo con un valor de 0.721.

5.3.3. Puntaje ROC AUC

En este caso, la Regresión Logística cuenta con el mayor puntaje ROC AUC de 0.818, seguido por el Random Forest con 0.817, el XGBoost y el SVM con 0.807. Por otro lado el modelo con menor puntaje es la Red Neuronal con 0.771. Esto indica que los dos mejores, en distinción de clases, son Regresión Logística y Random Forest.

5.3.4. Matriz de Confusión

A continuación se presenta una descripción de las matrices de confusión obtenidas. Contextualizando las matrices de confusión, los verdaderos positivos representan los supervivientes que fueron clasificados correctamente, mientras que los verdaderos negativos son los pasajeros que no sobrevivieron clasificados correctamente. Además los falsos positivos son pasajeros clasificados como supervivientes cuando en realidad no sobrevivieron y los falsos negativos son pasajeros clasificados como no supervivientes cuando en realidad sobrevivieron.

- **Regresión Logística**: 84 verdaderos negativos, 21 falsos positivos, 12 falsos negativos y 62 verdaderos positivos.
- **SVM**: 83 verdaderos negativos, 22 falsos positivos, 13 falsos negativos y 61 verdaderos positivos.
- **Red Neuronal**: 94 verdaderos negativos, 11 falsos positivos, 26 falsos negativos, 48 verdaderos positivos.
- **Random Forest**: 95 verdaderos negativos, 10 falsos positivos, 20 falsos negativos, 54 verdaderos positivos.
- **XG Boost**: 93 verdaderos negativos, 12 falsos positivos, 20 falsos negativos, 54 verdaderos positivos.

Random Forest, Red Neuronal y XGBoost son los modelos que muestran un mayor número de verdaderos negativos, sin embargo, la Red Neuronal presenta el mayor número de falsos negativos, por lo que se puede sospechar una subestimación de la clase positiva. Por otro lado la mejor clasificación de verdaderos positivos es generada por la Regresión Logística, aunque también genera un gran número de falsos positivos junto con el modelo de Support Vector Machine.

5.4. Conclusión

Considerando las métricas evaluadas a cada modelo, se ha determinado que el modelo más balanceado para la predicción de supervivencia del conjunto de datos de la situación problema es Random Forest. Este algoritmo cuenta con la mayor exactitud(0.832) y con buenos puntajes F1(0.782) y ROC-AUC(0.817). Aunque la regresión logística muestra un buen rendimiento de igual forma, se optará por **Random Forest**. La razón de esta decisión es que la regresión logística es un modelo mucho más sencillo, pero para el objetivo de éste reporte se busca un modelo más complejo que tenga buen rendimiento y que pueda demostrar las habilidades obtenidas durante el curso.

6. Evaluación y Refinamiento del Modelo

Como se mencionó anteriormente, el modelo que se utilizará para predecir la supervivencia de los pasajeros del Titanic con los datos de prueba será *Random Forest*. No obstante, antes de realizar la clasificación, se aplicarán técnicas de optimización para encontrar los mejores hiperparámetros, lo que permitirá obtener un modelo más preciso para hacer las predicciones.

Previo a la utilización de estos métodos, se realizarán dos divisiones. La primera será entre el conjunto de datos de entrenamiento y el de prueba; el conjunto de entrenamiento cuenta con etiquetas, mientras que el de prueba no las tiene y será utilizado para evaluar la clasificación del modelo con los nuevos hiperparámetros. La segunda división se hará dentro del conjunto de entrenamiento, separándolo en un subconjunto de entrenamiento y otro de validación, que se emplearán para entrenar y evaluar el modelo.

6.1. Refinamiento

Para encontrar los mejores hiperparámetros del modelo, se utilizarán las técnicas *Grid Search* y *Randomized Search*, además que también se hará una configuración manual. La configuración de ambos métodos será similar en cuanto a los parámetros que se buscarán. Para ***Grid Search***, la configuración es la siguiente:

- **n_estimators**: 5, 10, 50, 200, 250
- **max_features**: sqrt, log2
- **max_depth**: 4, 6, 8, 10, 12
- **criterion**: gini, log_loss, entropy
- **max_leaf_nodes**: 2, 5, 10

Para ***Randomized search***, la configuración es la siguiente:

- **n_estimators**: range(200, 1600)

- **criterion**: gini, log_loss, entropy
- **max_depth**: range(2, 51)
- **min_samples_split**: range(2, 11)
- **max_features**: sqrt, log2
- **bootstrap**: True, False

Con ***Grid search*** se obtuvo la siguiente configuración de híperparámetros:

- **n_estimators**: 200
- **max_features**: sqrt
- **max_depth**: 10
- **criterion**: gini
- **max_leaf_nodes**: 10

Con ***Randomized search*** se obtuvo la siguiente configuración de hiperparámetros:

- **n_estimators**: 1209
- **criterion**: gini
- **max_depth**: 4
- **min_samples_split**: 6
- **max_features**: sqrt
- **bootstrap**: True

La configuración manual de los hipérparámetros es la siguiente:

- **n_estimators**: 200
- **criterion**: gini
- **max_depth**: 10
- **min_samples_split**: 10
- **max_features**: sqrt

6.2. Evaluación

Para evaluar el desempeño de los modelos generados se utilizarán nuevamente Exactitud, Puntaje F1, Puntaje ROC-AUC y Matriz de confusión. La tabla 4 muestra los resultados de las métricas de evaluación

Modelo	VP	VN	FP	FN	Exactitud	Puntaje F1	Puntaje ROC-AUC
Grid Search	53	92	13	21	0.81	0.75	0.79
Randomized Search	52	92	13	22	0.80	0.74	0.78
Manual	54	95	10	20	0.83	0.78	0.82

Tabla 4: Métricas de Random Forest con diferentes hiperparámetros

6.2.1. Exactitud

El ajuste manual tiene la mayor exactitud (0.83), superando a Grid Search(0.81) y Randomized Search (0.80). Esto sugiere que el ajuste manual ha logrado una mejor configuración de los hiperparámetros para este conjunto de datos en términos de clasificación general correcta.

6.2.2. Puntaje F1

El puntaje F1, que combina precisión y exhaustividad, es más alto en el ajuste manual (0.78), en comparación con Grid Search(0.75) y Randomized Search (0.74). Esto implica que el ajuste manual maneja mejor el equilibrio entre falsos positivos y falsos negativos.

6.2.3. Puntaje ROC-AUC

El ROC-AUC mide la capacidad del modelo para distinguir entre clases. Nuevamente, el ajuste manual tiene un mejor desempeño (0.82) que Grid Search(0.79) y Randomized Search (0.78). Esto indica que el modelo ajustado manualmente tiene una mayor capacidad para discriminar correctamente entre clases positivas y negativas.

6.2.4. Matriz de Confusión

El ajuste manual reduce tanto los falsos positivos (FP = 10) como los falsos negativos (FN = 20), esto implica una mejor capacidad del modelo para clasificar correctamente tanto las clases positivas como negativas. Por otro lado, Grid Search y Randomized Search muestran resultados similares en términos de falsos positivos y falsos negativos. Grid Search tiene un mejor desempeño al reducir los falsos negativos (FN = 21), en comparación con Randomized Search (FN = 22), aunque ambos presentan el mismo número de falsos positivos (FP = 13). En cuanto a verdaderos positivos (VP) y verdaderos negativos (VN), los tres modelos son bastante similares, con Grid Search logrando 53 VP, Randomized Search 52, y el ajuste manual 54. Las

diferencias en estos valores son pequeñas, indicando que los tres modelos tienen un rendimiento comparable, con variaciones mínimas en la capacidad para clasificar correctamente.

7. Conclusión

Con base en los resultados obtenidos de las métricas, se concluyó que se procederá con el ajuste manual de los datos. Asimismo, si se tuviera que elegir un algoritmo para optimizar el modelo de predicción Random Forest, se optaría por Grid Search, ya que presenta las segundas métricas más adecuadas. La diferencia de desempeño se podría deber a la flexibilidad que uno tiene para experimentar con el modelo de forma más precisa. Aunque las búsquedas automatizadas pueden ser útiles, pueden terminar sin explorar las mejores combinaciones de configuraciones.

8. Referencias

[1] Evidently AI. *How to explain the ROC curve and ROC AUC score?* URL: https://www.evidentlyai.com/classification-metrics/explain-roc-curve#:~:text=A%20score%20of%200.5%20indicates,accuracy%2C%20precision%2C%20or%20recall.. (accessed: 30.08.2024).

[2] Data Camp. *Clasificación en machine learning: Introducción*. URL: https://www.datacamp.com/es/blog/classification-machine-learning. (accessed: 29.08.2024).

[3] N. Chauhan. *Métricas De Evaluación De Modelos En El Aprendizaje Automático*. URL: https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico. (accessed: 30.08.2024).

[4] XGBoost Developers. *XGBoost parameter documentation*. URL: https://xgboost.readthedocs.io/en/latest/parameter.html. (accessed: 31.08.2024).

[5] A. Gozhulovskyi. *Choosing a Model for Binary Classification Problem*. URL: https://medium.com/@andrii.gozhulovskyi/choosing-a-model-for-binary-classification-problem-f211f7a4e263. (accessed: 29.08.2024).

[6] Qlik Help. *Puntuación de modelos de clasificación binaria*. URL: https://help.qlik.com/es-ES/cloud-services/Subsystems/Hub/Content/Sense_Hub/AutoML/scoring-binary-classification.htm. (accessed: 29.08.2024).

[7] IBM. *What are support vector machines (SVMs)?* URL: https://www.ibm.com/topics/support-vector-machine. (accessed: 30.08.2024).

[8] IBM. *What is random forest?* URL: https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,Decision%20trees. (accessed: 31.08.2024).

[9] B. Mohtadi. *In Depth: Parameter tuning for SVC*. URL: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769. (accessed: 30.08.2024).

[10] Nvidia. *What is XGBoost?* URL: https://www.nvidia.com/en-us/glossary/xgboost/#:~:text=What%20is%20XGBoost%3F,%2C%20classification%2C%20and%20ranking%20problems.. (accessed: 31.08.2024).

[11] Scikit-Learn. *LogisticRegression*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. (accessed: 30.08.2024).

[12] Scikit-Learn. *MLPClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. (accessed: 31.08.2024).

[13] Scikit-Learn. *Neural network models (supervised)*. URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html. (accessed: 30.08.2024).

[14] Scikit-Learn. *RandomForestClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. (accessed: 31.08.2024).

[15] Scikit-Learn. *SVC*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. (accessed: 31.08.2024).

[16] Amazon Web Services. *¿Qué es la regresión logística?* URL: https://aws.amazon.com/es/what-is/logistic-regression/. (accessed: 30.08.2024).

[17] DMLC XGBoost. *Python API Reference*. URL: https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier. (accessed: 31.08.2024).