Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Inteligencia Artificial Avanzada para la Ciencia de Datos I

Ingeniería en Ciencias de Datos y Matemáticas

Selección, configuración y entrenamiento del modelo Reto

Profesores
Daniel Otero Fadul
Hugo Terashima Marin
Gerardo Ibarra Vázquez

Equipo 4

Niza Alvarado Rendon	A01367547
Renata Vargas	A01025281
Andrea Galicia Jimenez	A01177643
Ana Daniela López Dávila	A00831568
Alejandro Murcia Alfaro	A00828513
Iván L. Hernández Buda	A01412476

Monterrey, Nuevo León. 13 de septiembre de 2023

1. Introducción

La revolución tecnológica y la digitalización de la información han llevado a que, en la actualidad, se manejen cantidades inmensas de datos. Según Statista, el volumen de datos creados a nivel mundial ha aumentado de 33 zettabytes en 2018 a una proyección de 175 zettabytes para 2025 [1]. Esta explosión en la generación y almacenamiento de datos ha transformado radicalmente el proceso por el que tomamos decisiones. En este nuevo contexto, la ciencia de datos y los modelos de aprendizaje automático han emergido como herramientas esenciales para la automatización y mejora de la eficiencia en diversas áreas.

Uno de los desafíos más intrigantes en este campo es la predicción de eventos futuros basada en datos históricos. En este contexto, abordamos el caso del hundimiento del Titanic, buscando predecir la supervivencia de los pasajeros a bordo del Titanic. Para ello, recurrimos a técnicas avanzadas de modelado predictivo. Según Hastie, Tibshirani y Friedman en su libro The Elements of Statistical Learning, una técnica avanzada de modelado predictivo se refiere a métodos matemáticos y estadísticos que utilizan algoritmos complejos y técnicas de optimización para anticipar resultados futuros basados en datos pasados [2]. Nuestro objetivo es descubrir los patrones que respondan a la pregunta: ¿Quiénes sobrevivieron y quiénes no?

En este reporte, se explora la problemática abordada y se define el tipo de modelo a utilizar. Se procede a explorar la base de datos e identificar el significado de cada variable. Asimismo, se dan a conocer los modelos de aprendizaje automático utilizados. Se describe el preprocesamiento de datos, la selección de características, la construcción del modelo y la evaluación de cada modelo; estos siendo el SVM, Random Forest y KNN. Finalmente, se selecciona el modelo por refinar y se describe cómo fue el proceso de refinamiento para terminar con una comparación del antes y después.

2. Problemática

Desde el sentido común, se asume que algunos de los factores que influyen en la supervivencia de los pasajeros son: género, clase socioeconómica, edad y familiares abordo. Sin embargo, dentro de la base de datos nos encontramos con más variables que podrían tener un impacto significativo. Por esto es necesario hacer una extensa exploración sobre la relación de nuestras variables para reconocer cómo es que interactúan.

Como parte de la problemática, sera un desafío reconocer si existen o no patrones en los datos. Es crucial que, de identificar patrones, entendamos cómo es que estos afectan dentro del contexto de la situación. Asimismo, se espera identificar a que variables, no tomadas en cuenta, se les atribuye el error de los modelos probados.

Para dar inicio al desarrollo del proyecto, es necesario identificar cuál es el tipo de modelo que será necesario para resolver la problemática. Dado que tratamos de predecir si un pasajero sobrevivió o no al

acontecimiento, se entiende que el modelo deberá ser de **clasificación binaria**. Se utiliza la clasificación binaria en este caso porque el problema implica que solo hay dos resultados posibles (si sobrevivió o no). La clasificación binaria es simple, eficiente y facilita la toma de decisiones y la evaluación del modelo.

Esto se representa de la siguiente forma:

- Sobrevivió: 1
- No sobrevivió: 0

Para cada modelo se implementan unas métricas de evaluación las cuales nos ayudan a evaluar el rendimiento de estos modelos. Las metricas que se utilizaron son la exactitud, la precisión, la sensibilidad (recall) y la puntuación F1. Estas métricas se calculan como sigue:

$$\text{Exactitud} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Puntuación F1} = 2 \times \frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}}$$

Donde TP es Verdadero Positivo, TN es Verdadero Negativo, FP es Falso Positivo y FN es Falso Negativo.

Es así como el modelo, en función de los atributos, realizará una predicción binaria sobre la supervivencia de una persona específica abordo del Titanic. Los modelos de clasificación binaria más utilizados son la Regresión Logística, Support Vector Machine, Random Forest, entre otros [3]. Sin embargo, la limpieza de los datos será una parte importante al momento de evaluar nuestros modelos. A continuación se describen los datos utilizados para el proyecto.

La predicción de la supervivencia de los pasajeros del Titanic se basa en un problema de clasificación binaria, donde el objetivo es determinar si un individuo sobrevivió o no en función de ciertos atributos o características. En este contexto, el modelo, utilizando los atributos proporcionados, realizará una predicción binaria sobre la supervivencia de una persona específica a bordo del Titanic.

Existen varios algoritmos y técnicas que se utilizan comúnmente para abordar problemas de clasificación binaria. Algunos de los más populares y ampliamente utilizados incluyen:

- **Regresión Logística**: Es un método estadístico que se utiliza para predecir la probabilidad de una variable dependiente categórica basada en una o más variables independientes [4].
- **Support Vector Machine (SVM)**: Es un algoritmo de aprendizaje supervisado que puede ser utilizado tanto para clasificación como para regresión. SVM busca encontrar un hiperplano en un espacio N-dimensional (N es el número de características) que clasifica claramente los puntos de datos [5].
- **Random Forest**: Es un algoritmo de aprendizaje de conjunto que construye múltiples árboles de decisión durante el entrenamiento y produce la clase que es el modo de las clases de los árboles individuales para la clasificación [6].

La calidad y precisión de estos modelos dependen en gran medida de la calidad de los datos de entrada. Por lo tanto, la limpieza y el preprocesamiento de los datos son pasos cruciales antes de entrenar cualquier modelo. Los datos ruidosos, incompletos o incorrectamente etiquetados pueden llevar a modelos inexactos o sesgados. Por lo tanto, es esencial asegurarse de que los datos estén bien preparados y limpios antes de proceder con el modelado.

A continuación, se describen los datos utilizados para el proyecto, destacando su origen, características y cualquier preprocesamiento realizado para asegurar su calidad.

3. Base de Datos

Para esta asignatura se trabajaron con dos base de datos obtenidas directamente desde la plataforma de Kaggle, especificamente en la competencia llamada “Titanic - Machine Learning from Disaster” la cual contiene dos csv con información general sobre las personas que se encontraban en el titanic, uno para entrenar al modelo, “train.csv”, y otro para hacer pruebas, “test.csv” [7]. En ellas podemos notar diferentes variables como lo son:

- **PassengerId:** variable entera que representa el ID de cada pasajero (tipo de variable: int64)
- **Survived:** variable entera, de categoria binaria, la cual representa las personas que sobrevivieron (1) y fallecieron (0). (tipo de variable: int64)
- **Pclass:** el ticket que representa la clase del pasajero, 1 = 1era clase, 2 = 2nda clase, 3 = 3era clase. Hay que tener en cuenta que las clases hacen referencia al estatus socio-economico del pasajero teniendo que 1era clase = sociedad de alta clase, 2nda clase = sociedad de media clase y 3ra clase = sociedad de baja clase. (tipo de variable: int64)
- **Name:** nombre del pasajero (tipo de variable: object). Algo notorio en esta columna es que cada nombre tiene un título asociado (como Mr., Mrs., Master., Miss., Bissette y Bishop). Estos títulos pueden proporcionarnos pistas sobre la edad de la persona. Por ejemplo:

1. “Master.” generalmente indica niños varones entre 0 a 12 años.
2. “Mrs.” se refiere a una mujer casada, lo que sugiere que es poco probable que tenga menos de 16 años (edad de consentimiento).

- **Sex:** sexo del pasajero (tipo de variable: object).
- **Age:** edad en años del pasajero (tipo de variable: float64).
- **SibSp:** número de hermanas, hermanos, esposos o esposas en el barco (tipo de variable: int64).
- **Parch:** número de padres de familia y niños en el barco (tipo de variable: int64).
- **Ticket:** número de ticket (tipo de variable: object).
- **Fare:** tarifa del boleto (tipo de variable: float64).
- **Cabin:** número de cabina del pasajero (tipo de variable: object).
- **Embarked:** puerta de embarcación por donde ingresaron los pasajeros. (C = Cherbourg, Q = Queenstown, S = Southampton) (tipo de variable: object).

Analizando un poco más a detalle ambas bases de datos, pudimos notar que el test existen 418 registros mientras que en el train existen 891. El propósito de usar conjuntos etiquetado es entrenar un modelo de aprendizaje automático y luego evaluar su capacidad para hacer predicciones precisas en datos no etiquetados (test). El conjunto de entrenamiento se utiliza para entrenar al modelo, mientras que el conjunto de prueba se utiliza para medir su rendimiento en datos no vistos, lo que ayuda a garantizar que el modelo pueda generalizar correctamente y evitar problemas de sobreajuste. Esta división es esencial para evaluar y ajustar los modelos de manera efectiva.

4. Modelos de Machine Learning

a) Support Vector Machine

Preprocesamiento de Datos:

Se carga primeramente los datos limpios de entrenamiento guardados en un archivo “.csv”. Posteriormente, se eliminan variables categóricas irrelevantes (como “Ticket”, “PassengerId”, “Name”, “CabinEncode” y “Deck”) para preparar el conjunto de datos para el modelado. Las variables categóricas sobrantes se convierten en variables dummy utilizando la función pd.getDummies. Se grafica el mapa de calor de correlación entre las variables sobrantes (o relevantes) para identificar relaciones entre características.

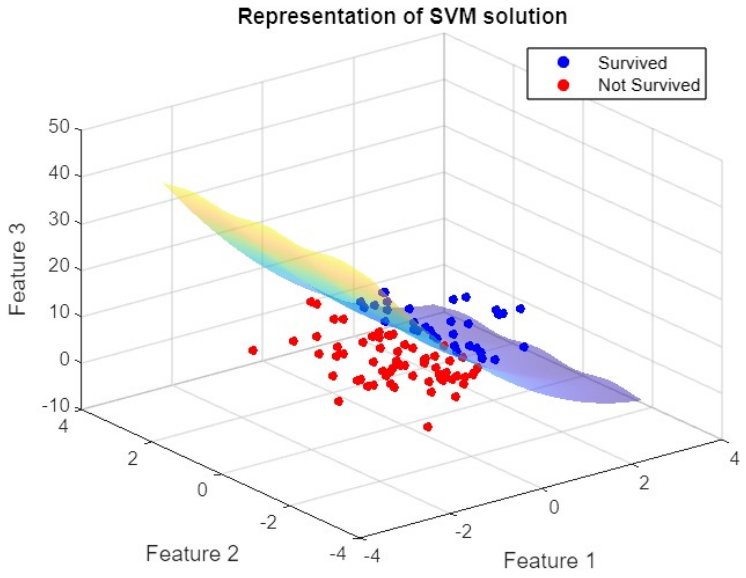
Selección de características:

Se calcula la correlación de cada característica con la variable objetivo “Survived” y se almacena en “corrSurvived”. Se consideran las características con una correlación menor a 0.2 con “Survived” para eliminar (featuresToDrop). Estas características de baja correlación se eliminan del conjunto de datos, dando lugar a dataFiltered. Se calcula una nueva matriz de correlación (corMatrixFiltered) basada en las características restantes de alta correlación. Se grafica el mapa de calor de esta matriz de correlación para visualizar las relaciones entre las variables restantes.

Construcción del Modelo SVM:

1. Se importa el modelo SVM de sklearn.svm.

2. Se crea una instancia del modelo SVM con hiperparámetros especificados, como:
   - **C**: Parámetro de penalización del término de error (parámetro de regularización)
   - **kernel**: Especifica el tipo de kernel (por ejemplo, ’poly’ para un kernel polinómico)
   - **gamma**: Coeficiente del kernel (auto para 1/n_características)
   - **class_weight**: Equilibrar clases ajustando los pesos de manera inversamente proporcional a las frecuencias de clase
   - **random_state**: Semilla para el generador de números aleatorios para reproducibilidad
3. El modelo SVM se entrena en los datos de entrenamiento utilizando `.fit()`.
4. Evaluación del Modelo:



Cálculo de métricas

Se hacen predicciones en los datos de prueba utilizando el modelo SVM entrenado. Se calcula la exactitud del modelo utilizando "accuracy_score" de sklearn.metrics. La puntuación de exactitud se imprime en la consola.

En resumen, este código carga un conjunto de datos limpios, los preprocesa eliminando variables irrelevantes y creando variables dummy para características categóricas. Luego realiza una selección de características basada en la correlación con la variable objetivo y aplica un modelo SVM con hiperparámetros especificados para clasificación binaria (sobrevivió o no sobrevivió).

La exactitud del modelo se evalúa utilizando los datos de prueba. El objetivo es construir un modelo predictivo que clasifique con exactitud los resultados de supervivencia basados en las características seleccionadas y el algoritmo SVM. La exactitud del modelo fue de 0.72 sobre los datos de validación. Y de 0.66 sobre los datos de prueba proporcionados por la competencia de Kaggle. En cuanto a las métricas adicionales, el recall del modelo resultó en 0.72, y la puntuación F1 de 0.69.

b) Random Forest

Preprocesamiento de Datos:

Para la limpieza de este modelo, realizamos una serie de modificaciones en los dos sets de entrenamiento y test. Para comenzar, nos enfocamos en la variable de Edad y notar que había muchos valores nulos (177 en el set de entrenamiento y 86 en el de prueba). Esto fue considerando que la edad era la principal variable en un pasajero para determinar su supervivencia. Como eso es un porcentaje sumamente representativo del set de entrenamiento, usaremos Random Forest para llenar estos valores . Para esto, haremos uso de las variables dummy para las variables categóricas y de esta manera puedan ser procesadas por el modelo. Después de eso, quitaremos las columnas que no utilizamos para el análisis: Name, Ticket y Cabin.

Modelo

El modelo Random Forest es el conjunto de múltiples árboles de decisión donde cada árbol está entrenado para un subconjunto de los datos, para que al final se unan para formar una predicción o una clasificación mucho más fundamentada.

La formulación básica del algoritmo Random Forest se puede expresar como sigue:

$$Y = f(X_1, X_2, \ldots, X_k) + \epsilon$$

donde Y es la variable objetivo, $X_1, X_2, \ldots, X_k$ son las características y ϵ es el error aleatorio.

El modelo se entrena utilizando el método `fit` en el conjunto de entrenamiento.

```
rf.fit(X_train, y_train)
```

Donde también se calcularon las métricas para analizar la evaluación del modelo mencionadas al inicio del documento (exactitud, precisión, sensibilidad (recall) y puntuación F1).

El modelo de Random Forest es un algoritmo robusto y versátil para tareas de clasificación y regresión. Su capacidad para manejar un gran número de características y su flexibilidad en la configuración de parámetros lo convierten en una excelente opción para una variedad de problemas de aprendizaje automático.

Los parámetros del modelo de Random Forest se eligieron de la siguiente manera:

- **n_estimators=200**: Se eligieron 200 árboles para asegurar que el modelo sea lo suficientemente complejo como para capturar patrones en los datos.
- **max_depth=10**: Se limitó la profundidad máxima de los árboles a 10 para evitar el sobreajuste.
- **min_samples_leaf=2**: Se requieren al menos 2 muestras para formar una hoja, lo que ayuda a evitar árboles demasiado complejos.
- **min_samples_split=2**: Se requieren al menos 2 muestras para dividir un nodo, lo que también ayuda a evitar el sobreajuste.
- **bootstrap=False**: Se decidió no utilizar bootstrap para darle al modelo la oportunidad de utilizar todo el conjunto de datos para el entrenamiento.

El modelo de Random Forest funciona construyendo múltiples árboles de decisión durante el entrenamiento y luego promediando los resultados para mejorar la precisión y controlar el sobreajuste. En el contexto del Titanic, cada árbol toma en cuenta diferentes combinaciones de características y condiciones para hacer una predicción sobre si un pasajero sobrevivirá o no.

Por ejemplo, un árbol podría considerar principalmente la clase del pasajero y el sexo, mientras que otro podría centrarse en la edad y el número de hermanos o cónyuges a bordo. Al final, el modelo toma la "votación"de todos los árboles para hacer una predicción final. [8]

El modelo logró una exactitud de aproximadamente 0.8268 en el conjunto de validación y un puntaje de 0.7887 en la competencia de Kaggle, lo que indica un rendimiento bastante bueno. Las métricas adicionales como el recall 0.8713 y la puntuación F1 0.9113 también mostraron resultados prometedores.

c) K-Nearest neighbor

Preprocesamiento de Datos: Para la limpieza de este modelo se realizaron diversas modificaciones específicamente en nuestra base de datos de entrenamiento, "train". Primeramente se combinaron algunas

variables que podrian manejarse juntas para evitar datos atípicos innecesarios como lo son combinar los padres de familia junto con los hermanos asumiendo que son de la misma familia. Después se procedió a eliminar columas que no son necesarias para este modelo como el nombre de los pasajeros, el ticket, la tarifa del boleto y la cabina de cada pasajero. Finalmente para tener un poco mas de orden con los datos restantes al momento de limpiarlos, se decidió por crear unas variables dummies para poder tener la información por categorías cambiando los tres tipos de embarcamiento (S, C, Q) por números (1, 2, 3) y de igual manera pero con el genero (masculino = 1 y femenino=0). Todo este mismo procedimiento se realizo de igual manera para la base de datos "test"para así utilizarla en la implementación del modelo.

División de datos de entrenamiento y validación: Primero, dividimos nuestros datos en dos partes: datos de entrenamiento y datos de prueba. Lo hicimos para poder enseñar a nuestro modelo y luego evaluar cuán bien podemos hacer predicciones en datos nuevos.

División de datos de entrenamiento y prueba: Utilizamos estos datos para enseñarle a nuestro modelo cómo predecir quién sobrevivió y quién no en el barco. En Xtrain, incluimos características como la clase de boleto (Pclass), la edad (Age), el tamaño de la familia (Family) y la tarifa del boleto (Fare). ytrain contenía información sobre si cada pasajero sobrevivió (1) o no (0).

Datos de prueba (Xtest): Estos datos son para evaluar qué tan bien podemos hacer predicciones nuestro modelo en nuevos datos. También tienen las mismas características que Xtrain, pero no tienen las etiquetas de supervivencia.

Entrenamiento de un modelo KNN: Luego, elegimos utilizar un algoritmo llamado K Vecinos Más Cercanos (KNN) para construir un modelo de predicción. El número 17 que ajustamos como n neighbors es una configuración que podemos cambiar según nuestras necesidades.

Entrenamiento del modelo: El modelo KNN se entrena utilizando los datos de entrenamiento (Xtrain y ytrain). Esto significa que el modelo aprende de estos datos cómo se relacionan las características (como la clase de boleto, la edad, la familia y la tarifa) con la supervivencia.

Realización de predicciones: Una vez que el modelo está entrenado, lo usamos para hacer predicciones sobre quién sobrevivió en un conjunto de datos de prueba (Xtest). El modelo tomó las características de cada pasajero en Xtest y trató de predecir si sobrevivieron o no. Luego, guardamos esas predicciones en la variable ypred.

Verificación de la longitud de los resultados: Finalmente, verificamos que la cantidad de predicciones en ypred coincidiera con la cantidad de datos en el conjunto de prueba (Xtest). Esto era importante para asegurarnos de que el modelo hiciera una predicción para cada pasajero en el conjunto de prueba. Si la longitud de ypred y Xtest no coincidiera, podría indicar un problema en nuestro código.

Rendimiento: Después de entrenar nuestro modelo K Vecinos Más Cercanos (KNN), evaluamos su rendimiento utilizando una métrica llamada exactitud, que mide la proporción de predicciones correctas en

comparación con el total de predicciones realizadas.

En nuestros datos de prueba, el modelo alcanzó una exactitud de aproximadamente 0.702. Esto significa que acertó 70 %. de las veces al predecir si un pasajero sobrevivió o no en función de las características proporcionadas.

Sin embargo, es importante señalar que al cargar nuestras predicciones en la plataforma Kaggle, obtuvimos un puntaje de exactitud de aproximadamente 0.60 en el conjunto de datos de prueba de Kaggle. Esto sugiere que nuestro modelo podría beneficiarse de mejoras adicionales. Además, en las métricas adicionales, se obtuvó un recall de 0.69 y una puntuación F1 de 0.70.

En resumen, aunque nuestro modelo tiene un rendimiento decente con una exactitud del 70 %., aún estamos trabajando en mejorarlo para lograr resultados aún más precisos, especialmente teniendo en cuenta la puntuación que obtuvimos en Kaggle.

5. Discusión

A continuación un resumen de los resultados de las métricas en el conjunto de entrenamiento para cada modelo:

	SVM	**Random Forest**	**KNN**
Exactitud	0.7226	0.9349	0.7017
Precisión	0.6718	0.9551	0.7078
Recall	0.7166	0.8713	0.6871
Puntuación F1	0.6935	0.9113	0.6973

La evaluación de los modelos en Kaggle se realizó al generar un archivo .csv conformado por una columna de Passenger ID y otra columna con las predicciones. A continuación, las mejores evaluaciones de los modelos.

Mejores Predicciones:

- SVM: 0.72
- Random Forest: 0.7887
- KNN: 0.6

El modelo con mejor exactitud fue el de Random Forest con exactitud de 0.7887 y el peor fue K-Means con exactitud de 0.6.

6. Selección del modelo

Con base en los resultados obtenidos, podemos concluir que el modelo de Random Forest es el que muestra un mejor comportamiento. Esto debido a que es el modelo de mejor puntaje en la predicción de la competencia en Kaggle.

7. Refinamiento del modelo

7.1. Ajuste de Hiperparámetros

Para realizar el ajuste de hiperparámetros, se recurrió a la herramienta GridSearchCV de la biblioteca scikit-learn. La técnica GridSearchCV nos permite realizar la validación cruzada, es decir, se ejecuta a través de los diferentes parámetros que se introducen en la cuadrícula de parámetros y se extraen los mejores valores y combinaciones de parámetros [9].

Es por esto que, una vez se define nuestro modelo, se proporciona una cuadrícula de hiperparámetros para probarse durante la búsqueda. Esta cuadrícula incluye los siguientes hiperparámetros:

- **n_estimators**: El número de árboles de decisión que se construirán en el bosque aleatorio. Se tomaron en cuenta los valores 50, 100 y 200 árboles.
- **max_features**: El número máximo de atributos que se considerarán al dividir un nodo durante la construcción de cada árbol. Se tomaron en cuenta las opciones auto, sqrt y log2.
- **max_depth**: La profundidad máxima permitida para cada árbol en el bosque, es importante para evitar el sobreajuste del modelo. Se tomaron en cuenta los valores 10, 20, 30 y la opción None, es decir, sin límite.
- **min_samples_split**: El número mínimo de muestras requeridas para dividir un nodo interno, define cuán grande debe ser un nodo antes de intentar dividirlo. Se tomaron en cuenta los valores 2, 5 y 10 muestras.
- **min_samples_leaf**: El número mínimo de muestras requeridas para formar una hoja en un árbol, define cuán pequeñas pueden ser las hojas. Se tomaron en cuenta los valores 1, 2 y 4 muestras.
- **bootstrap**: Controla si se debe realizar un muestreo con reemplazo o sin reemplazo al construir cada árbol en el bosque aleatorio, es decir, si se permite que una misma muestra se seleccione más de una vez. Se toman en cuenta las opciones True y False.

Esta cuadrícula es utilizada en la validación cruzada con cinco divisiones para evaluar el rendimiento de cada conjunto de hiperparámetros. Esto quiere decir que se divide el conjunto de datos de entrenamiento en

5 partes aproximadamente iguales, donde 4 de los conjuntos se utilizan para el entrenamiento del modelo y el número 5 se utiliza en la evaluación del rendimiento. Este proceso se repite cinco veces, pues son cinco partes en las que se divide el conjunto de datos de entrenamiento. De esta forma nos aseguramos de que cada conjunto sirva como conjunto de prueba exactamente una vez.

Asimismo, la validación cruzada tiene como parámetro definido *n_jobs = -1*. Es decir, que se utilizan todos los núcleos de CPU disponibles en el sistema. Este parámetro permite que la búsqueda de hiperparámetros se realice de manera más rápida.

7.2. Resultados de búsqueda

A continuación se dan a conocer los mejores hiperparámetros para ajustar nuestro modelo de la mejor forma posible.

- **n_estimators**: *200*, es la cantidad de árboles que se construirán en el bosque aleatorio.
- **max_features**: *auto*, es equivalente a sqrt en este contexto, y significa usar la raíz cuadrada del número total de características como cantidad máxima de atributos que se consideran al dividir un nodo durante la construcción de cada árbol.
- **max_depth**: *20*, la profundidad máxima permitida para cada árbol es de este valor.
- **min_samples_split**: *2*, es la cantidad mínima de muestras para dividir un nodo interno.
- **min_samples_leaf**: *4*, es la cantidad mínima de muestras para formar una hoja en un árbol.
- **bootstrap**: *False*, no se permite que una misma muestra se seleccione más de una vez al constriuir cada árbol en el bosque aleatorio.

7.3. Comparación de modelos

Una vez se obtuvo el modelo ajustado, se presenta una comparación de los resultados de validación del modelo antes y después del ajuste de hiperparámetros.

Métrica	**Antes del Ajuste**	**Después del Ajuste**
Exactitud	0.8156	0.9349
Precisión	0.7971	0.9551
Recall	0.7432	0.8713
Puntuación F1	0.7692	0.9113

Estos resultados resaltan la eficacia del proceso de ajuste de hiperparámetros, el cual permitió al modelo alcanzar un rendimiento significativamente mejorado en todas las métricas evaluadas. La precisión y el puntaje F1, en particular, experimentaron un aumento notorio, lo que indica una mejor capacidad del modelo ajustado para clasificar correctamente las instancias positivas y negativas. Esta mejora en el rendimiento del modelo es fundamental para su utilidad en el proyecto.

Asimismo, se realizó nuevamente la entrega de resultados en la competencia para poder comparar la exactitud del modelo después del ajuste pero con los datos de prueba. Antes del ajuste, se tenía una exactitud de 0.7887 y, después del ajuste, Kaggle brindó una exactitud de 0.82.

8. Conclusión

Como se menciono anteriormente, se decidio trabajar con el modelo de Random Forest no solo por el hecho de haber sido el modelo con el mejor score en su momento sino que se observo conforme su aplicacion que cuenta con una capacidad para manejar datos faltantes y ruidosos, su robustez frente al overfitting, el manejo automático de la importancia de características, la capacidad para lidiar con desequilibrio de clases, su facilidad de uso y disponibilidad en bibliotecas populares.

Crear el modelo es uno de los pasos mas importantes; sin embargo, durante este proceso se fue indagando sobre el afinamiento de los hiperparametros para que este modelo bien hecho pueda mejorar aun mas. Este afinamiento de los hiperparámetros es esencial para mejorar el rendimiento, prevenir el sobreajuste, ahorrar recursos, adaptarse a los datos, optimizar costos y obtener modelos más interpretables en el aprendizaje automático. Ajustar los hiperparámetros correctamente marca la diferencia entre un modelo efectivo y uno que no lo es; por lo que gracias a esto se pudo obtener un $f_1 score$ muy bueno al finalizar el modelo.

Es fundamental emplear modelos de machine learning como K-Means, Support Vector Machines (SVM) y Random Forest debido a su versatilidad y eficacia en la resolución de una amplia gama de problemas. K-Means es útil para la segmentación de datos, SVM brilla en la clasificación y regresión, y Random Forest ofrece alta precisión y resistencia al sobreajuste. Estos modelos son adaptables, interpretables y pueden manejar datos ruidosos, lo que los convierte en herramientas esenciales para la toma de decisiones basada en datos y la solución de problemas en el mundo real.

En conclusión, una comprensión profunda permite tomar decisiones informadas, interpretar resultados y resolver problemas de manera efectiva, lo que es esencial en la aplicación exitosa del aprendizaje automático en una amplia variedad de aplicaciones y sectores. En última instancia, la comprensión completa de un modelo de machine learning es la clave para obtener resultados precisos y confiables en la toma de decisiones basada en datos.

Bibliografía

[1] G. Moreno, "Infografía: A la espera de un big bang de datos," 4 2019. [Online]. Available: https://es.statista.com/grafico/17734/cantidad-real-y-prevista-de-datos-generados-en-todo-el-mundo/

[2] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction.* Springer, 2009, vol. 2.

[3] F. Parra, "6 Métodos de clasificación — Estadística y Machine learning con R." [Online]. Available: https://bookdown.org/content/2274/metodos-de-clasificacion.html

[4] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression.* John Wiley & Sons, 2013.

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

[6] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] W. Cukierski, "Titanic - machine learning from disaster," 2012. [Online]. Available: https://www.kaggle.com/competitions/titanic

[8] G. Biau, "Analysis of a random forests model," *The Journal of Machine Learning Research*, vol. 13, pp. 1063–1095, 2012.

[9] R. KeepCoding, "¿Qué es GridSearchCV? — KeepCoding Bootcamps," 6 2023. [Online]. Available: https://keepcoding.io/blog/que-es-gridsearchcv/