Instituto Tecnológico y de Estudios Superiores de Monterrey

Inteligencia Artificial Avanzada para la Ciencia de Datos (Gpo 101)

Evaluación y Refinamiento de modelo

Equipo 5

Jorge Eduardo De León Reyna - A00829759
David Esquer Ramos - A01114940
Francisco Mestizo Hernández - A01731549
Adrián Emmanuel Faz Mercado - A01570770

Septiembre 4, 2023

1. Introducción

El problema que se está tratando de solucionar es determinar si un pasajero del Titanic sobrevive o no basado en diferentes caraterísticas, como el puerto en el que embarcó, su edad, sexo, entre otros. Esto nos genera un problema de clasificación binaria.

1.1 Set de datos

El set de datos utilizado es el del titanic, obtenido de la plataforma de Kaggle. Para consultarlo a más detalle se puede consultar en el siguiente link. El set tiene 891 registros y cuenta con las siguientes variables: Passenger ID, Survival, Ticket class (PClass), Sex, Age, Number of siblings / spouses aboard the Titanic (sibsp), Number of parents / children aboard the Titanic (parch), Ticket number, Passenger fare, Cabin number, Port of Embarkation.

Debido a que el set de datos no ests listo para usar, se realizó una limpieza de datos. En esta limpieza se realizó un one hot encoding para las variables que eran categóricas. Además, había algunos registros que les faltaban datos como la edad. Basados en su título y su clase determinamos un aproximado de la edad que podrían haber tenido. Para ver con más detalle la limpieza de datos se sugiere visitar esa entrega.

Los datos limpios se pueden encontrar en la siguiente liga.

1.1.1 Separación de los datos

Para la evaluación y refinamiento de nuestro modelo, se realizó la separación de los datos de entrenamiento en 2 grupos más, uno para entrenar el modelo y el otro para validarlo de manera iterativa y con diferentes parámetros. Cuando se estuvo satisfecho del desempeño con el subset de datos para validación, se procedió a usar el de prueba original, y así obtener una estimación final de cómo el modelo se desempeñará en datos completamente nuevos.

Para la división, se decidió designar el 80% de los datos de entrenamiento para entrenar el modelo, y el 20% para realizar la validación del mismo.

Datos de entrenamiento *train_clean.csv*	Datos de entrenamiento (80%)
	Datos para validación (20%)
Datos de prueba	Datos de prueba

test_clean.csv	

2. Métricas

Para la selección del mejor modelo, se utilizó el criterio de "**accuracy**". Esta métrica mide la proporción de predicciones correctas que hizo un modelo en relación con el total de predicciones.

Para calcularla, se utilizan los valores obtenidos de la matriz de confusión, estos valores son los siguientes:

Verdaderos positivos (VP)	Número de casos positivos (1) que el modelo predijo correctamente como positivos.
Verdaderos negativos (VN)	Número de casos negativos (0) que el modelo predijo correctamente como negativos.
Falsos positivos (FP)	Número de casos negativos (0) que el modelo predijo incorrectamente como positivos.
Falsos negativos (FN)	Número de casos positivos (0) que el modelo predijo incorrectamente como negativos.

La fórmula matemática para obtener la "accuracy" o exactitud del modelo es la siguiente:

$$Accuracy = \frac{VP+VN}{VP+VN+FN+FP}$$

Básicamente, consiste en dividir la cantidad total de predicciones correctas entre el número total de predicciones. La elección de esta métrica se debe a que se busca tratar de predecir la mayor cantidad de registros correctamente. Esta métrica se utilizará tanto para el subconjunto de entrenamiento como para el subconjunto de prueba.

3. Modelo seleccionado

En la etapa anterior, se realizaron pruebas con 3 diferentes modelos:

- Random Forest
- Logistic Regression
- XGBoost

Para cada algoritmo, se entrenó el modelo con los datos de entrenamiento y luego se utilizaron los datos de validación para evaluar su desempeño. Se realizaron pruebas de “accuracy” con ambos conjuntos de datos para cada modelo y después de las ellas, se eligió la regresión logística para resolver el problema.

La elección de este modelo permite resolver problemas de clasificación, sin tener un overfitting muy grande, como pasaría con un modelo de random forest. Además, de todos los modelos probados fue el que mejor rendimiento mostró. Para ver con más detalle la selección del modelo se recomienda visitar el siguiente link.

4. Optimización de hiper parámetros y regularización

Para obtener los mejores resultados con el modelo elegido se deben modificar sus hiperparámetros y realizar ajustes para poder maximizar la eficiencia y obtener los mejores resultados posibles.

El problema del Titanic provee ya dos grupos de datos separados, uno que se utiliza para el entrenamiento de los datos, y otro para las pruebas del modelo. Sin embargo, es recomendable dividir el conjunto de entrenamiento en dos partes adicionales: una para seguir entrenando el modelo y otra para validarlo. Esta división permite ajustar los hiperparámetros y evaluar el rendimiento del modelo sin tener que usar el conjunto de prueba, proporcionando así una estimación más confiable de cómo se desempeñará el modelo con datos no vistos.

4.1 Hiperparametros del modelo

Se sabe que se pueden modificar diferentes parámetros para que el modelo se ajuste mejor a los datos. Los hiperparámetros que se modificaron fueron los siguientes:

- **Solver (solucionador):** es el algoritmo utilizado para resolver el problema de optimización asociado con la regresión logística. Los solucionadores comunes incluyen "liblinear", "lbfgs", "newton-cg", "sag", y "saga". Cada uno de estos

solucionadores utiliza un enfoque diferente para encontrar los coeficientes óptimos del modelo.

- **Fit_intercept (Ajuste de intercepción):** controla si se debe ajustar o no la intercepción (también conocida como sesgo) en el modelo de regresión logística. Si se establece en "True", el modelo ajustará la intercepción; si se establece en "False", el modelo no ajustará la intercepción y se considerará que pasa por el origen.
- **Class_weight (Peso de clases):** se utiliza para manejar el desequilibrio de clases en un problema de clasificación. Si las clases en el conjunto de datos no están balanceadas (una clase tiene muchas más muestras que otra), puedes usar este parámetro para asignar pesos diferentes a las clases.
- **Tol (Tolerancia)**: controla cuándo se considera que el modelo ha convergido durante el proceso de entrenamiento. Indica la diferencia mínima aceptable entre las iteraciones sucesivas para considerar que el modelo ha convergido. Si la diferencia entre las iteraciones cae por debajo de esta tolerancia, el modelo se considera convergente y el entrenamiento se detiene.
- **max_iter (Número máximo de iteraciones)**: para encontrar un modelo que se ajuste a los datos es necesario correr varias iteraciones. Pero el tener muchas iteraciones puede generar un overfitting, que el código sea muy tardado de correr o que se siga corriendo sin obtener mejores resultados. Por esto es importante cambiar este hiperparámetro, para que el modelo se ajuste bien a los datos y deje de ajustarse cuando no se vea una mejora

4.2 Hiperparametros de regularización

Para el ajuste óptimo de los hiper parámetros del modelo utilizado se hizo uso de la librería *Optuna*. Esta librería tiene como objetivo explorar distintas opciones de hiper paramétros a través de algoritmos y estrategias más eficientes que si se hiciera de manera “manual” de manera que se puedan encontrar los parámetros óptimos para el módulo utilizado, en este caso un modelo de regresión logística. Por otro lado, los hiperparametros optimizados con el uso de *Optuna* fueron los siguientes:

- **Penalty (Penalización)**: controla la regularización aplicada al modelo de regresión logística. La regularización es una técnica que se utiliza para evitar el sobreajuste al penalizar los coeficientes de las características menos importantes. Puede tomar dos valores principales: "l1" Utiliza la norma L1 para penalizar los coeficientes, lo que

puede llevar a la selección automática de características al reducir algunos coeficientes a cero y "l2" el cual utiliza la norma L2 para penalizar los coeficientes, lo que ayuda a evitar coeficientes muy grandes y reduce el impacto de características menos importantes.

- **Parámetro C:** controla la regularización del modelo. La regularización evita el sobreajuste ajustando la fuerza de penalización en la función de costo. Un valor pequeño de "C" aumenta la regularización, haciendo que el modelo sea más conservador y generalice mejor, mientras que un valor grande de "C" disminuye la regularización, permitiendo un ajuste más cercano a los datos de entrenamiento, lo que podría resultar en sobreajuste.

5. Refinamiento del modelo

Para refinar el modelo de Regresión Logística, fue necesario probar con diferentes hiperparámetros lo cual tomaría mucho tiempo al tener que realizar un el proceso a prueba y error hasta encontrar la mejor combinación de hiperparámetros, es por ello que con el fin de optimizar el proceso para encontrar los hiperparametros ideales en esta nueva iteración, se utilizó la librería de *Optuna*. Con esto en cuenta, los hiperparametros seleccionados fueron el resultado de probar con distintas configuraciones hasta encontrar la ideal.

Para obtener la mejor combinación de hiperparámetros, se corrieron 10,000 pruebas utilizando *Optuna* y se seleccionó el modelo que tuvo mejores resultados. Antes de correr las pruebas, se tenía un modelo que daba los siguientes resultados:

```
Logistic Regression sin optimización de hiperparámetros
Logistic Regression Training : 0.846
Logistic Regression Testing : 0.775
```

Se puede ver que tiene un buen desempeño en training y en testing ya que el accuracy está por arriba de 70. De todas formas, los valores se encuentran un poco separados lo cual podría indicar que el modelo presenta un nivel de overfitting a los datos de entrenamiento.

Los hiper parámetros que hacen el modelo lo más óptimo posible son:

C	0.0126412354368

Penalty	l2
Solver	'saga'
Fit_intercept	False
Class_weight	Balanced
Tol	2.61438480
Max_iter	1200

6. Resultados

Al entrenar un modelo lineal con los hiperparámetros seleccionados, se obtienen los siguientes resultados:

```
Logistic Regression con hiperparámetros optimizados
Accuracy Training: 0.7923728813559322
Accuracy Testing: 0.797752808988764

Matriz de Confusión
array([[88, 18],
       [18, 54]])
```

Los resultados para training y testing se encuentran muy cercanos, alrededor de 79% de accuracy al predecir los resultados, que es un poco peor que el modelo anterior. Pero esto no presenta un problema ya que con el refinamiento y regularización se puede asegurar que el modelo será más versátil porque ya no muestra indicios de overfitting, lo que hace más probable el obtener buenas predicciones para datos nuevos o distintos a los de entrenamiento, aunque tenga un rendimiento un poco menor.

Igualmente, se comprueba esta mejoría del modelo viendo una curva ROC del desempeño antes y después del refinamiento. Si la curva se encuentra muy alejada de la diagonal central, tiene un valor diagnóstico perfecto, mientras que si se acerca mucho, es altamente probable que el modelo confunda las clases y no haga una clasificación correcta.

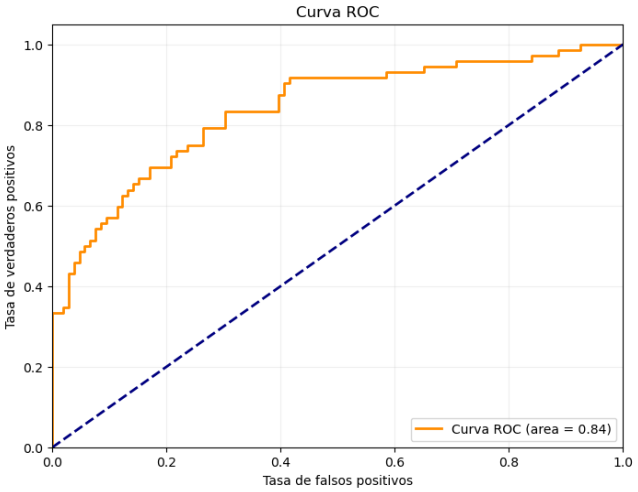

Figura 1.1 Curva ROC para el modelo sin refinamiento

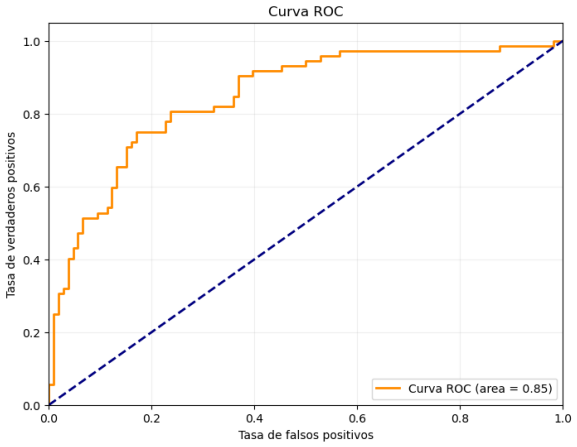

Figura 1.2 Curva ROC para el modelo después del refinamiento

En la figura 1.1 se indica que el área bajo la curva es de 0.84 para el modelo sin refinamiento, mientras que en la figura 1.2 el área bajo la curva es de 0.85. Por lo tanto, existe una pequeña mejoría del modelo.