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

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.

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 viene listo para usar, realizamos 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.

Separación de los datos

Para la evaluación y refinamiento de nuestro modelo, realizamos la separación de nuestros datos de entrenamiento en 2 grupos más, uno para entrenar el modelo y el otro para validarlo de manera iterativa y con diferentes parámetros. Ya cuando estamos satisfechos del desempeño con el subset de datos para validación, procederemos 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, decidimos designar el 80% de nuestros datos de entrenamiento para entrenar nuestro modelo, y el 20% para realizar la validación del mismo.

<table>
<tr><td rowspan="2">Datos de entrenamiento
train_clean.csv</td><td>Datos de entrenamiento (80%)</td></tr>
<tr><td>Datos para validación (20%)</td></tr>
<tr><td>Datos de prueba
test_clean.csv</td><td>Datos de prueba</td></tr>
</table>

Métricas

Para seleccionar el mejor modelo, se utilizará el accuracy score. La elección de este parámetro se debe a que queremos tratar de predecir la mayor cantidad de registros correctamente.

Modelo seleccionado

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

- Random Forest
- Logistic Regression
- XGBoost

Para cada algoritmo, entrenamos el modelo con los datos de entrenamiento y luego utilizamos los datos de validación para evaluar su desempeño. Se realizaron pruebas de "accuracy" con ambos sets de datos para cada modelo y después de las ellas, se eligió la regresión logística para resolver el problema. Esta elección se debe a que este es un modelo que nos permite resolver problemas de clasificación, sin tener un overfitting muy grande, como nos pasa con un 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.

Refinamiento del modelo

Nos encontramos en la etapa de evaluación y refinamiento del modelo, en donde ya seleccionamos el modelo que utilizaremos para resolver el problema, pero debemos de modificar sus hiperparámetros y realizar ajustes para poder maximizar la eficiencia del modelo y que obtengamos los mejores resultados posibles.

El problema del Titanic nos 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 nos 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.

Sabemos que se pueden modificar diferentes parámetros para que el modelo se ajuste mejor a nuestros datos. Los hiperparámetros que modificamos 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.
- **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

Para refinar el modelo de Regresión Logística, es necesario probar con diferentes hiperparámetros, pero tomaría mucho tiempo realizar 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 *Optuma*. 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, corrimos 10,000 pruebas utilizando *Optuma* y nos quedamos con el modelo que tenga mejores resultados. Antes de correr las pruebas, teníamos 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 y nos puede indicar que el modelo tiene algo de overfitting a los datos de entrenamiento.

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

C	2.73997
Penalty	l2
Solver	'lbfgs'
Fit_intercept	True
Class_weight	None
Tol	0.0002
Max_iter	700

Y si entrenamos un modelo lineal con esos hiperparámetros seleccionados, obtenemos los siguientes resultados:

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

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

Ahora, los resultados de training y testing se encuentran muy cercanos, alrededor de 79% de precisión al predecir los resultados, que es mucho mejor de lo que lo hacía el modelo pasado. Esto es bueno ya que el modelo será más versátil porque ya no parece que esté haciendo overfitting, entonces es más probable el obtener buenas predicciones para datos nuevos o distintos a los de entrenamiento.