Desempeño de un Modelo de Machine Learning con framework

Oscar E. Delgadillo Ochoa – A01705935

Resumen

En este documento se va a describir el proceso para la realización de un modelo de clasificación de machine learning con regresión logística haciendo uso del framework de sklearn. Manejando datos de un dataset sobre el arroz obtenido en el Machine Learning Repository UCI. [1]

Palabras claves

- Machine Learning.
- Regresión Logística.
- Matriz de confusión.
- Hiperparámetros.
- Sklearn

Introducción

Para la clase de Inteligencia artificial avanzada para la ciencia de datos vemos temas de machine learning, para este caso en particular vamos a implementar un modelo de regresión logística que es un algoritmo para resolver problemas aprendizaje supervisado, más en específico un problema de clasificación.

La problemática de clasificación es en base un dataset de arroz con los siguientes parámetros:

1. Área: Devuelve el número de píxeles dentro de los límites del grano de arroz.

2. Perímetro: calcula la circunferencia calculando la distancia entre píxeles alrededor de los límites del grano de arroz.

3. Longitud del eje principal: la línea más larga que se puede dibujar en el grano de arroz.

4. Longitud del eje menor: La línea más corta que se puede dibujar en el grano de arroz.

5. Excentricidad: Mide la redondez de la elipse que tiene los mismos momentos que el grano de arroz.

6. Área convexa: devuelve el recuento de píxeles de la capa convexa más pequeña de la región formada por el grano de arroz.

7. Extensión: Devuelve la proporción de la región formada por el grano de arroz a los píxeles del cuadro delimitador

8. Clase: Commeo y Osmancik.

Con los datos que contiene el dataset mencionado anteriormente queremos predecir la clase de arroz que tenemos en nuestras manos.

Metodología

Primero seleccionamos los datos y los convertimos a un DataFrame de pandas, para obtener predicción y dropeamos la información que no sea relevante para lo que vamos a predecir.

Usando el framework de sklearn aplicamos su función de regresión logística teniendo como hiperparámetos principales:

- Solver: ‘sag’
- Penalty: ‘l2’
- Max_iter: 100

Siendo solver un método de optimización del algoritmo con 4 posibles opciones:

- ‘liblinear’: utiliza un algoritmo de descenso coordinado que logra solucionar problemas de optimización realizando una minimización aproximada de manera sucesiva y se recomienda para datasets grandes.

- ‘lbfgs’: almacena solo unos pocos vectores que representan la aproximación implícitamente.

- ‘newton-cg’: este método utiliza la matrix Hessiana para minimizar el error, sin embargo es computacionalmente cara.

- ‘sag’: optimiza la suma de un número finito de funciones convexas suaves.

- ‘saga’: es una variante del sag adecuado para datasets más grandes.

Penalty que pretende reducir el error de generalización del modelo y desincentivar y regular el sobreajuste con 4 posibles configuraciones l1’, ‘l2’, ‘elasticnet’ ‘none’ con las que hay que tener cuidado, porque no funcionan con todos los solvers, se debe seguir la tabla de a continuación

	Solvers				
Penalties	**'liblinear'**	**'lbfgs'**	**'newton-cg'**	**'sag'**	**'saga'**
Multinomial + L2 penalty	no	yes	yes	yes	yes
OVR + L2 penalty	yes	yes	yes	yes	yes
Multinomial + L1 penalty	no	no	no	no	yes
OVR + L1 penalty	yes	no	no	no	yes
Elastic-Net	no	no	no	no	yes
No penalty ('none')	no	yes	yes	yes	yes
Behaviors					
Penalize the intercept (bad)	yes	no	no	no	no
Faster for large datasets	no	no	no	yes	yes
Robust to unscaled datasets	yes	yes	yes	no	no

Tabla 1. Solvers – Penalties

Finalmente, max_iter que es el número máximo de iteraciones necesarias para que los solucionadores converjan.

Con la configuración de arriba obtenemos una precisión en el test de 82%, viendo el comportamiento de negativos y positivos en la siguiente matriz de confusión del test.

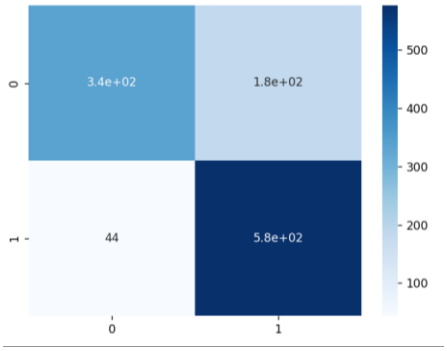

Figura 1. Confusion matrix model 1

Para el mejoramiento del modelo se modificaron los hiperparámetos a:

- Solver: ‘lbfgs’
- Penalty: ‘none’
- Max_iter: 5000

Obteniendo una mejor precisión

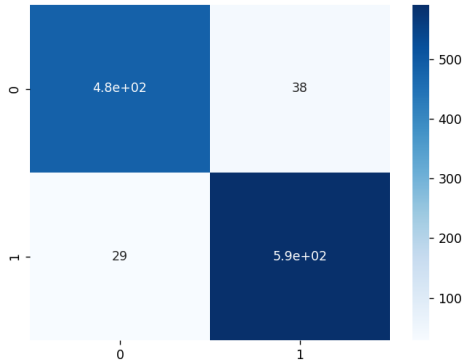

Figure 2. Confusion matrix model 2

Objetivos

Conseguir un modelo con un buen porcentaje de predicción para, siendo en este caso de aproximadamente 94%, viendo también el mejoramiento en la siguiente matriz de confusión del test.

Análisis y resultados

En las gráficas de aprendizaje de abajo podemos observar el sesgo y la varianza del modelo y así mismo determinar si el modelo esta ‘underfitting’ que se da cuando el sesgo (bias) es muy grande, esta ‘overfitting’ que ocurre cuando el sesgo es bajo, pero la varianza es alta, o esta ‘fit’ que es cuando el sesgo es bajo y la varianza es baja.

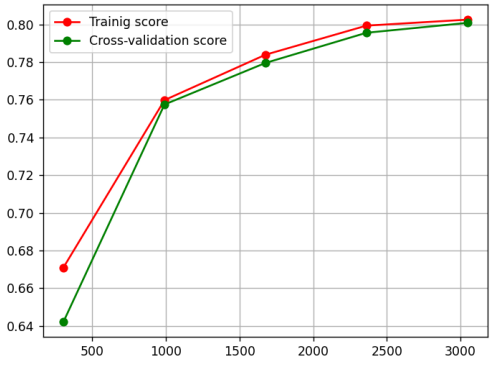

Figura 3. Learning curve model 1

Para la gráfica de arriba podemos ver como el modelo al principio tiene underfitting porque tiene un bias alto tanto en train como en test, pero con forme pasan las iteración converge hasta tener un fit.

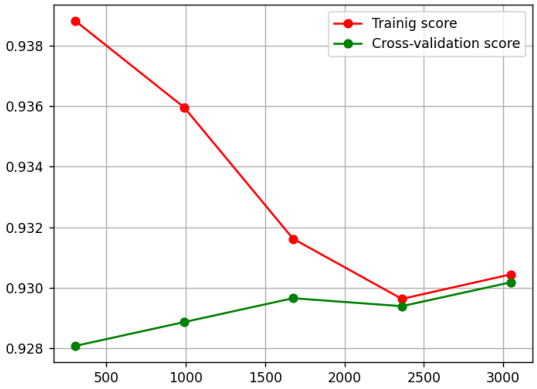

Figura 4. Learning curve model 2

Por lo mencionado antes podemos decir que el modelo en este caso se da un fit porque al principio tiene un underfitting, pero con el pasar de las iteraciones se disminuye el bias hasta que converge y obtenemos un modelo fitteado mejor que el anterior.

Esto también lo podemos decir obteniendo los errores del train y del test, que en nuestro primer modelo son bajos, pero al cambiar algunos de los hiperparámetros de framework mencionados anteriormente se obtienen errores aún más bajos en train y test.

Conclusiones

Los resultados fueron satisfactorios y considero que obtuve conocimiento sobre los temas de machine learning vistos en esta unidad de formación. Además, fue bastante interesante conocer las maneras para mejorar el comportamiento de nuestro modelo.

Bibliografía

[1] UCI. (2022). UCI Machine Learning Repository: Rice (Cammeo and Osmancik) Data Set. Retrieved 8/09/2022, from https://archive.ics.uci.edu/ml/datasets/Rice+%28Cammeo+and+Osmancik%29

[2] Repositorio de GitHub con el modelo de regresión logística en python: https://github.com/Oscar19260/ML_models_with_framework