Fecha 09 09, 2022

Inteligencia artificial avanzada para la ciencia de datos I

Análisis y Reporte sobre el desempeño del modelo (septiembre 2022)

Gerardo Peña Pérez A01701474

RESUMEN: Se tomó un modelo de regresión logística para poder predecir si un tumor es maligno o benigno. Este modelo fue desarrollado usando scikit-learn. Se logró un valor de accuracy cercano al 97% incluso haciendo un cross validation, lo que nos dice que el modelo no está en estado de overfit. Aquí se explica el proceso llevado a cabo, así como los resultados obtenidos

CONCEPTOS CLAVE: Machine Learning, regresión lineal, framework, scikit-learn.

I. INTRODUCCIÓN

La regresión logística es un algoritmo que resulta útil para resolver problemas de clasificación binaria, es decir, en el que se tiene que predecir entre 2 diferentes categorías, o bien predecir la probabilidad de que algo pertenezca a una clase, esto basándose en diversos datos de entrada. En este caso se usará dicho algoritmo para predecir si un tumor es maligno o benigno, tomando como entrada características de la masa a analizar, tales como el radio, la textura, la concavidad, entre otras.

Se usa el data set de Wisconsin Breast Cancer Database (January 8, 1991), el cual cuenta con 569 instancias y 32 atributos. Uno de dichos atributos es la variable por pronosticar, es decir la clase de tumor, teniendo un valor de 2 al ser benigno y 4 al ser maligno.

II. SEPARACIÓN DE LOS DATOS

Se realizó una etapa de entrenamiento y una de pruebas, para esto, el set de datos fue separado en 2 subsets (train y test), esto para poder verificar que el modelo esté aprendiendo correctamente y que no esté aprendiendo los valores específicos en lugar de arrojar predicciones reales, a esto se le llama estado de "over fitting", en el que, a pesar de que se llegara a lograr un error de entrenamiento bajo, el error de validación terminaría siendo bastante alto, debido a que el modelo no reaccionaría bien al enfrentarse a datos que no haya visto durante su entrenamiento, pero de esto hablaré más adelante.

Para dividir los datos se usó la función "train_test_split" del modulo de model_selection de scikit-learn. A dicha función le doy los conjuntos de datos que quiero que divida y una semilla que inicializa el generador de números pseudoaleatorios que básicamente me ayuda a que la división sea aleatoria. Al no definir el tamaño de cada subset, automáticamente forma el set de entrenamiento con un 75% del total y el de pruebas con 25%

III. RESULTADOS

Una vez entrenado el modelo, se procede a crear un arreglo con las predicciones generadas por el modelo para así poder evaluar el desempeño de este probando con datos que no conoció en la etapa de entrenamiento. Haciendo uso de la función "accuracy_score" podemos conocer que la precisión con la que el modelo genera predicciones es de 0.9714285714285714 , es decir que ronda cerca del 97% de precisión.

Para poder describir el desempeño del modelo de clasificación generado se usó una matriz de confusión. Esta es la matriz obtenida con los datos de prueba:

[[117 1]
[4 53]]

Podemos notar que las predicciones acertadas suman 170. Mientras que tuvimos 4 casos de falsos positivos y tan solo 1 caso de falso negativo.

En la siguiente gráfica podemos ver el accuracy que el modelo logró en la etapa del entrenamiento comparado con el logrado en el cross validation:

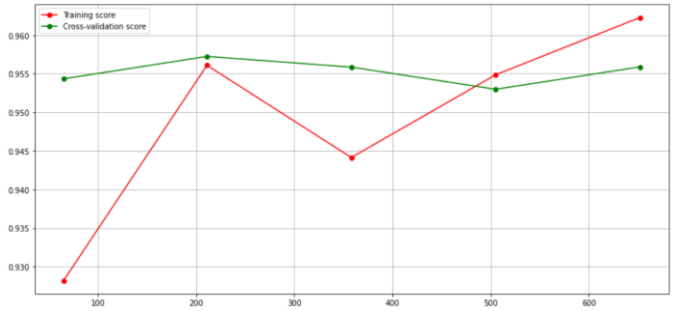

Figura 1.1. Accuracy en el entrenamiento y en el cross validation

IV. NIVEL DE AJUSTE DEL MODELO

Para saber si el modelo estaba en un estado de overfitting se realizó un cross validation haciendo 5 divisiones del dataset y se obtuvo un accuracy promedio de 0.9513874614594039, lo
cual nos dice que el modelo en efecto entrenó bien y no solo para los valores de entrenamiento, evitando así el estado de overfitting

V. DETECCIÓN DE BIAS Y VARIANZA

Ya que conocemos el accuracy con el que el modelo hace predicciones, el cual fue suficientemente alto (0.95138), podemos concluir que el modelo tiene un sesgo bajo y una varianza baja, ya que no hay tanto error dentro de las predicciones, además de que en la figura 1.1 se nota que el accuracy de pruebas no sube, si no que se mantiene descendiendo. Por lo tanto, podemos volver a confirmar que el modelo no está en estado de underfitting u overfitting.

VI. REGULARIZACIÓN

En este caso su obtuvo un nivel aceptable de error y accuracy, pero se puede calibrar el modelo para mejorar su desempeño, para esto se utiliza algún método de regularización. En este caso usé la regularización Ridge (L2).
Se usó la librería “Ridge” para generar un modelo de tipo “ridgemodel” y así ver si hay alguna mejora en cuanto al accuracy con el que el modelo hace predicciones.
Al hacer cross validation en el nuevo modelo, se encontró un accuracy promedio de 0.7711623135065906
La precisión con la que se hacen predicciones bajó drásticamente, esto se debe a que al ya haber tenido un modelo en estado de fit e intentar regularizarlo, terminamos sobre simplificando el modelo.