Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo. (Portafolio Análisis)

Miguel Ángel Pérez López A01750145

Regresión logística

La regresión logística es un tipo de análisis de regresión para predecir una variable y. Sirve para estudiar las relaciones entre el conjunto x y la y. Necesita una o más entradas de variables independientes, una variable dependiente y una función de activación (por ejemplo la sigmoide). Sólo funciona con variables numéricas.

Dataset

Se utilizó el dataset winequality_red de Kaggle que tiene 11 variables independientes numéricas de tipo flotante y una variable “y” binaria que define si el vino es de clase es de buena calidad o no.

El dataset no necesitó imputación de datos o cambio de variables categóricas a numéricas.

Escalamos todas las variables independientes X.

Separación de datos

Para el modelo de este dataset separamos los datos en train, test y validation de la siguiente forma:

Test size: 20%

Del train size se destinó el 25% para el validation set. Por lo tanto:

Train size: 60%

Validation size: 20%

Procedimiento

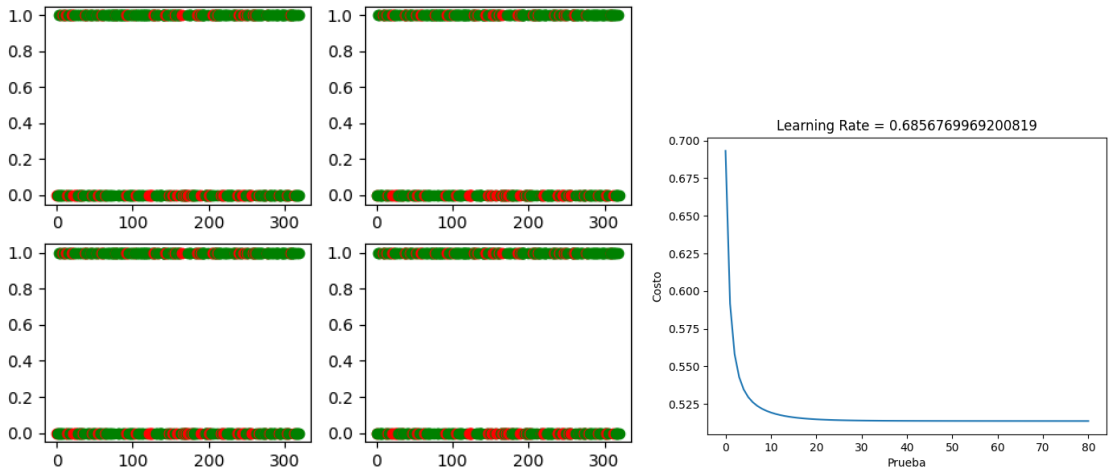
Crearemos una lista con épocas y learning rates aleatorios. Vamos a entrenar el modelo y usaremos el conjunto de validación varias veces con las distintas épocas y learning rates de la lista antes mencionada. Con esos hiper parámetros calcularemos la precisión de cada corrida y guardaremos los hiper parámetros que den la mejor precisión.

De esta forma tenemos hiper parámetros que podrían dar una buena precisión con el conjunto de test.

Para mejorar el desempeño del modelo utilizaremos una técnica de regularización llamada coarse to fine. Vamos a probar el modelo con un rango menor y mayor alrededor de los hiper parámetros obtenidos. Esto nos da una gran probabilidad de encontrar mejores hiper parámetros que resulten en una mejor precisión.

Resultados

Entrenamiento con conjunto de validación



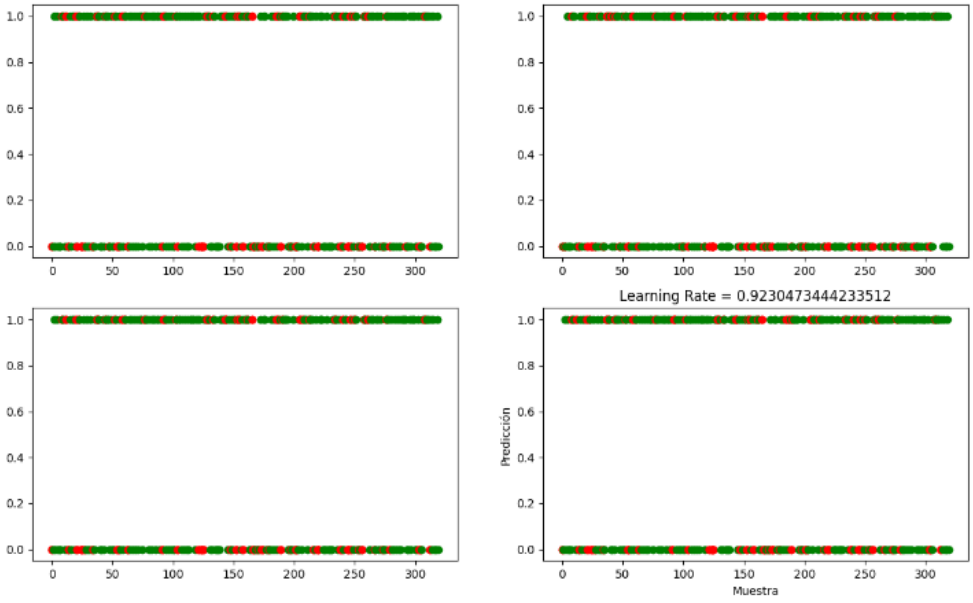
Predicciones correctas, Predicciones incorrectas

Como podemos observar en las 4 corridas, hay un gran porcentaje de puntos verdes (predicciones correctas).

El grado de bias o sesgo es bajo casi medio porque hay varios puntos verdes pero en el eje x en medio hay muchos puntos rojos. En general la mayoría son predicciones correctas alcanzando más del 70% de accuracy. La gráfica de costo nos indica que el costo no es bajo y que no va a llegar a 0. Por lo tanto el bias es un relativamente medio.

El grado de varianza es **bajo** porque las 4 corridas se ven muy similares. Esto nos indica que hay poca dispersión en los resultados y que las predicciones no cambian drásticamente dependiendo la corrida. La gráfica de costo muestra que hay poca dispersión porque el costo va disminuyendo y no muestra “picos”.

Predicción con conjunto de test después de usando técnicas de regularización



Predicciones correctas, Predicciones incorrectas

```
[74.55683003128259, 0.8700982676679923, 6470]
```

Resultados de la última corrida: [Precisión, alpha, épocas]

Estas gráficas son de las predicciones con valores de test y tenemos resultados muy similares a los del training e incluso mejores.

El grado de varianza es **bajo**. Esto nos indica que hay poca dispersión en los resultados. El conjunto de test es más grande y no parece que los errores hayan incrementaron proporcionalmente al número de datos.

El nivel de ajuste del modelo final es **fit**. Definitivamente no es overfitting porque el error en el test no incrementó mucho. Debido a que el learning rate se calculó aleatorio con un rango entre 0 y 1, el learning rate fue alto y el costo no se pudo reducir más. Al tener un costo mayor a 0.5, tiene sentido que la precisión no sea mayor a 75% (como se ve en el screenshot de arriba).

Conclusión

El método de regresión logística fue muy útil para predecir la clase de los vinos. Después de varias iteraciones vi que las técnicas de regularización son muy buenas para encontrar mejores hiper parámetros aunque como vimos en clase, ocupan más recursos y es mucho más tardado el entrenamiento.

Se hicieron varias pruebas con un gran número de épocas (50 mil-100mil) y un learning rate bajo (0.05-0.1) y terminaban con overfitting teniendo una precisión menor a la mostrada en el screenshot de arriba). Por lo que para evitar que el código tardara demasiado y que no tuviera overfitting, decidí mantener las épocas en un rango de mil a 10 mil iteraciones. El bias fue medio y se podría disminuir con otras técnicas de machine learning. La precisión de más de 70% considero que es muy bueno en especial porque los resultados no mostraron overfitting.