Momento de Retroalimentación: Módulo 2 Análisis y Reporte sobre el desempeño del modelo

Alan Eduardo Aquino Rosas
A01366912
Modulo 2

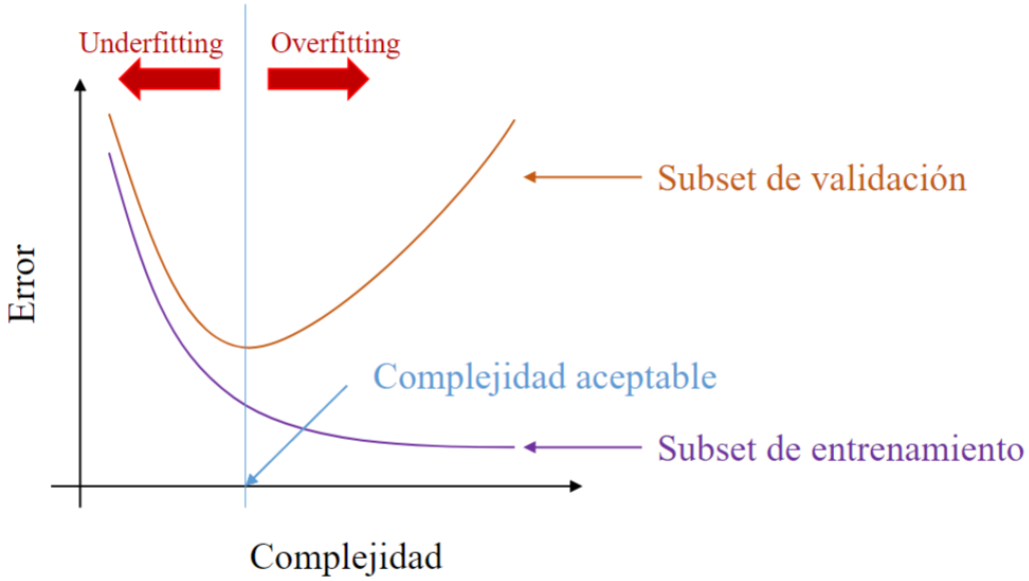
La actividad consistía en elegir un modelo de los vistos en clases y analizar el comportamiento en cuestión de rendimiento a base de diferentes “hyperparameters” que se le puede cambiar a la configuración del modelo seleccionado.

Para esta actividad se utilizó un modelo de MLP Classifier de Sci-Kit Learn, este es un algoritmo de ML Supervisado, llamado Neural Networks.

Teoría de Rendimiento

Para identificar que afecta en el rendimiento de una red neuronal primero debemos entender que es una red neuronal.

Cuando nosotros utilizamos un algoritmo como logistic regresión, para alguna aplicación en la industria, existe el problema que muchas veces el modelo de regresión logística simple no puede ajustarse bien a los datos y hacer buenas predicciones. Normalmente, cuando sucede esto es debido a la complejidad del data set, pues una línea no puede separar los datos de manera adecuada. Cuando un modelo no se puede ajustar bien a los datos debido a su complejidad tan simple, se le llama que es un modelo que tiene una alta bias y por ende tiene underfitting. Para solucionar esto, lo que se hace es crear nuevas características con un mayor orden, que nos puedan ofrecer la capacidad de ajustar el data set mejor. Sin embargo, si esto no se realiza de una manera adecuada puede existir la posibilidad que el modelo haga overfitting y regresemos al mismo problema.



Debido a esto existen las redes neuronales, donde básicamente el modelo se compone de capas y en cada capa existe una cierta cantidad de unidades. En si cada unidad se podría tomar como si fuera un algoritmo de regresión logística. Esto nos permite que el mismo modelo pueda sacar nuevas características más interesantes a través de las conexiones entre las capas y así poder computar funciones mas interesantes que ajusten mejor a un dataset especifico. Debido a la naturaleza de las redes neuronales entre mas capas y unidades nos va a permitir hacer modelos más complejos.

Por ende, las redes neuronales normalmente son algoritmos que normalmente tienen a ser máquinas de overfitting(high varias/low bias) y esto incrementa mas entre mas capas y neuronas debido a que se van a computar funciones más complejas.

Los hyperparametros son valores que podemos modificar de nuestro modelo para que este se comporte de una manera diferente, normalmente buscando que se mejore el desempeño.

En el caso de linear regresión y logistic regresión, el hyperparametro más ocupado es el tamaño de paso, que nos permite llevar control de que tan largo quieres el paso en el algoritmo de optimización de gradiente descendiente.

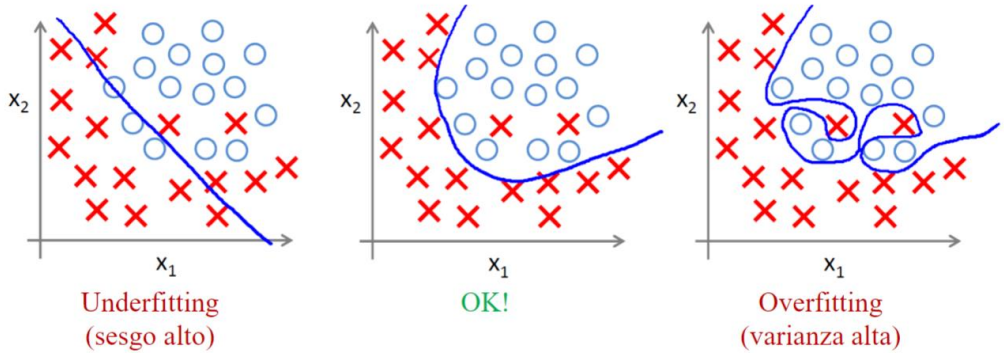
Igualmente, si ocupamos regularización, otro hyperparametro podría ser el tamaño de lamda o incluso si requerimos una función más compleja, podríamos ocupar de hiperparametro las características polinomiales que existen.

En el caso de redes neuronales, que es el algoritmo que se analizara, si tenemos underfiting(alto bias,baja varianza) podemos aumentar el numero de capas y neuronas para que modele funciones mas complejas. En caso contrario, si existe overfitting podemos obtener mas datos para tratar que

el modelo generaliza mejor y utilizar algún tipo de regularización. Igualmente, se podría bajar el numero de capas en la red neuronal, pero está probado que realmente esto no sirve del todo.

Cabe mencionar, que en análisis del modelo, no se incluyó el hyperparametro de lamba de regularización, porque la libreria de scikit learn, automáticamente hace la regularización de tipo L2.Este tipo de regularización esta basada en castigar a pesos altos en la función de costo para bajar la complejidad del modelo.

Cuando hablamos de generalización nos referimos a un punto medio, donde el modelo no tiene overfitting ni underfitting y normalmente al momento de evaluar un modelo es al punto al que queremos llegar.



Para checar que un modelo no tenga underfitting, normalmente basta con checar que el costo de entrenamiento sea alto pues el modelo no ajusta bien a los datos de entrenamiento. En contrario, cuando se trata de un modelo que necesitamos saber si tiene overfitting, normalmente se divide mínimamente el dataset en training set y validation set. Esto con el propósito de identificar si el modelo realmente generalizo bien y no memorizo completamente. Si un modelo sufre de overfitting tiende a tener un costo de entrenamiento muy bueno debido a que se ajusto perfectamente a los datos conocidos, sin embargo, para subconjunto de validation que son datos desconocidos tiene un rendimiento notoriamente menor. En base al resultado del validation set se toman decisiones de que modelo y hyperparametros son los mejores para la aplicación del problema.

Características de Dataset

Ahora, entendido esto podemos regresar y analizar nuestro modelo.

El modelo fue aplicado en un data set consistía en personas con anemia y sin anemia y sus características.

El modelo se implemento con el objetivo de poder clasificar en base a ciertas características, si una persona tenía o no anemia.

El data set se puede encontrar en kaggle con el siguiente enlace:

https://www.kaggle.com/datasets/biswaranjanrao/anemia-dataset

El dataset tiene la siguiente estructura:

Características (variables independientes)

- Gender: 0 - male, 1 - female
- Hemoglobin: Hemoglobin is a protein in your red blood cells that carries oxygen to your body's organs and tissues and transports carbon dioxide from your organs and tissues back to your lungs
- MCH: MCH is short for "mean corpuscular hemoglobin." It's the average amount in each of your red blood cells of a protein called hemoglobin, which carries oxygen around your body.
- MCHC: MCHC stands for mean corpuscular hemoglobin concentration. It's a measure of the average concentration of hemoglobin inside a single red blood cell.
- MCV: MCV stands for mean corpuscular volume. An MCV blood test measures the average size of your red blood cells.

Clases (Variable dependiente)

- Results: 0- not anemic, 1-anemic

Preprocesamiento

Antes de realizar entrenar y comparar los modelos cabe mencionar que se realizo un preprocesamiento de la característica “Gender” debido a que tenía label enconding,esto puede sesgar al modelo porque defines implícitamente que mujer>hombre, por lo que se cambió por 1-hot enconding, y se creó una columna nueva para hombres y una columna para mujeres.

Igualmente se divido el dataset en training (75%) y validation set(25%) para poder identificar que pasa con el modelo.

Ahora comenzaremos a definir nuestros modelos.

La primera iteración del modelo fue la siguiente:

```
#---------------------------------Red Neuronal con 1 capa-----------------------------------

#Configuracion de red neuronal para clasificacion
anemia_nn_1=MLPClassifier(random_state = 0,
                          hidden_layer_sizes = (5),
                          activation = "relu",
                          verbose = False,
                          solver = "adam",
                          learning_rate = "adaptive",
                          max_iter = 1000)
```

Se estableció un modelo de una capa con cinco neuronas, con funciones de activacion relu y un learning rate adaptativo.

Esto nos permitió tener los siguientes resultados:

```
3         14.9  16.0  31.4  87.5
4         14.7  22.0  28.2  99.5
Training Score 0.4187793427230047
Testing Score 0.4887640449438202
```

Como se puede ver no son resultados muy alentadores, debido a que el training Score es muy bajo nos indica que el modelo no es suficientemente complejo para ajustar a las características, sufre de un bias alto, por ende, existe un underfitting.

Tomando esto en cuenta, se aumentó las capas y neuronas del modelo para que pueda hacer funciones mas complejas que ajusten mejor a los datos.

```
#---------------------------------Red Neuronal con 2 capas-----------------------------------

#Configuracion de red neuronal para clasificacion
anemia_nn_2=MLPClassifier(random_state = 0,
                          hidden_layer_sizes = (5,5),
                          activation = "relu",
                          verbose = False,
                          solver = "adam",
                          learning_rate = "adaptive",
                          max_iter = 1000)

#Entrenamiento red neuronal
anemia_nn_2.fit(train_x,train_y)
```

En lugar de ocupar una capa de 5 neuronas, ser ocuparon 2 capas de 5 neuronas cada una.

El resultado fue el siguiente:

```
12 12
---------------------Red Neuronal con 2 capas--
Training Score 0.952112676056338
Testing Score 0.9634831460674157
--------------------------Predicciones--------
```

Con el simple hecho de aumentar una capa más, el modelo se pudo ajustar mejor a los datos del entrenamiento y debido a esto ya no tenemos underfitting. Igualmente, en el validation set nos dio un puntaje muy alto dándonos a entender que el modelo generalizo adecuadamente y predice correctamente en valores nunca vistos.

Aunque el modelo, ya se considera adecuado, se buscó obtener un puntaje más alto en el los 2 datasets y se modificó la arquitectura de la red neuronal con 2 capas más de 5 neuronas.

```
#Configuracion de red neuronal para clasificacion
anemia_nn_3=MLPClassifier(random_state = 0,
                          hidden_layer_sizes = (5,5,5,5),
                          activation = "relu",
                          verbose = False,
                          solver = "adam",
                          learning_rate = "adaptive",
                          max_iter = 1000)

#Entrenamiento red neuronal
anemia_nn_3.fit(train_x,train_y)
```

Los resultados fueron los siguientes:

```
Training Score 0.9568075117370892
Testing Score 0.9662921348314607
```

El modelo ya no tuvo una mejoría en los scores significativa.

Finalmente, comparando los resultados se elige el segundo modelo con una arquitectura de 2 capas por el balance que tiene, ya que no tiene underfitting o overfiting y generaliza bien, sin tener un costo computacional tan alto.

Por ultimo estas son las métricas de rendimiento del modelo seleccionado:

```
----------Metricas de Desempeño del modelo seleccionado(2 capas)----------
[[175   7]
 [  6 168]]
==========================================
Metricas de rendimiento para modelo numero
Exactitud     : 0.9634831460674157
Precision     : 0.9668508287292817
Exhaustividad : 0.9615384615384616
Puntaje F1    : 0.9641873278236914
==========================================
```