Momento de Retroalimentación: Módulo 2 Implementación de una técnica de aprendizaje máquina sin el uso de un framework. (Portafolio Implementación)

Alumno: Ramón Yuri Danzos García

Matricula: A00227838

Maestro: Jesús Adrián Rodríguez Rocha

Materia: Inteligencia artificial avanzada para la ciencia de datos I (Gpo 102)

Monterrey, NL. 7 de septiembre de 2024

Índice

Tabla de contenido

Introducción

En este reporte se busca documentar el proceso de elaboración del portafolio 1 para el modulo de machine learning, explicando de manera clara el paso a paso que se realizó tanto a nivel de código como de documentación, así como argumentar las decisiones tomadas a lo largo del proyecto.

Procedimiento

En el procedimiento se expresa detalladamente el camino que se decide seguir para elaborar el proyecto propuesto.

Repositorio

Realización del repositorio en GitHub, añadiendo Product Backlog para manejar una metodología ágil, así como creación de archivo README.

Enlace a Repositorio: https://github.com/danzosr/Machine_Learning_Logistic_Regression

Problemática

El problema por resolver escogido en Kaggle ha sido “Logistic regression To predict heart disease”.

La Organización Mundial de la Salud calcula que cada año se producen en el mundo 12 millones de muertes por cardiopatías. La mitad de las muertes en Estados Unidos y otros países desarrollados se deben a enfermedades cardiovasculares. El pronóstico precoz de las enfermedades cardiovasculares puede ayudar a tomar decisiones sobre cambios en el estilo de vida de los pacientes de alto riesgo y, a su vez, reducir las complicaciones. Esta investigación pretende identificar los factores de riesgo más relevantes de las enfermedades cardiacas y predecir el riesgo global mediante regresión logística.

Dentro de la página de Kaggle se puede apreciar la descripción de la problemática, así como la definición del dataset utilizado

Enlace a Problemática: https://www.kaggle.com/datasets/dileep070/heart-disease-prediction-using-logistic-regression

Realización del Modelo

El modelo realizado ha sido regresión logística, ya que se adecuaba mejor a los datos obtenidos, siendo que se tienen 15 features con los que se debe crear un modelo que tenga la capacidad de predecir si una persona tiene riesgo de padecer una enfermedad coronaria dentro de 10 años basado en los datos recolectados.

Limpieza y Separación de Dataset

Inicialmente es necesario revisar el dataset original y analizar si existen datos nulos, en caso de serlo buscar la proporción de datos nulos para tomar una decisión sobre que hacer al respecto. Para este caso el dataset cuenta con 15 features, 1 target, y 4238 instancias.

En la ilustración 1 se aprecia la cantidad de datos faltantes por cada feautre que cuenta con datos nulos, se aprecia que el que mayor cantidad de valores desconocidos tiene es “glucose”, en la ilustración 2 se aprecia el porcentaje que representan las cifras nulas. Se denota que en el caso de glucosa el porcentaje de valores faltantes es cercano al 10% por lo que al no ser cantidades tan elevadas se decidió realizar un método de llenado de datos por medio de calcular el promedio +- la varianza y elegir un valor aleatorio entre estos para rellenar los campos vacíos.

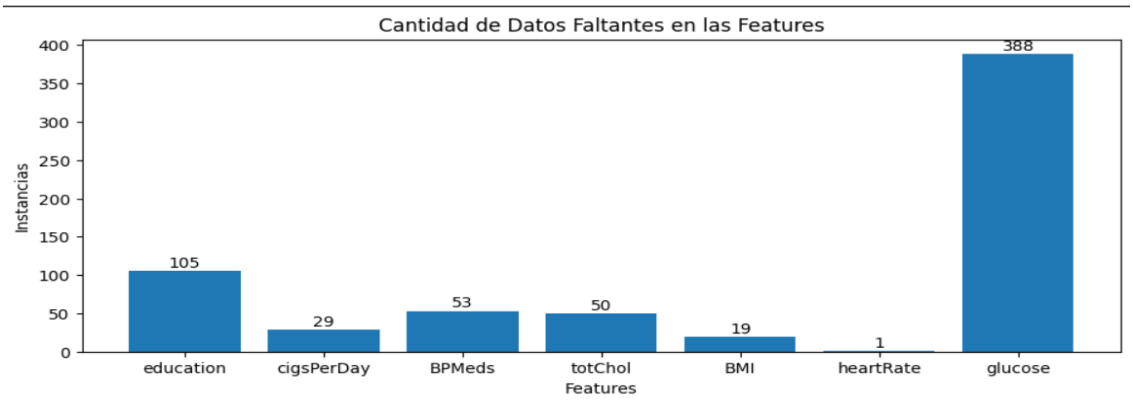

Ilustración 1: Datos Faltantes Dataset Original

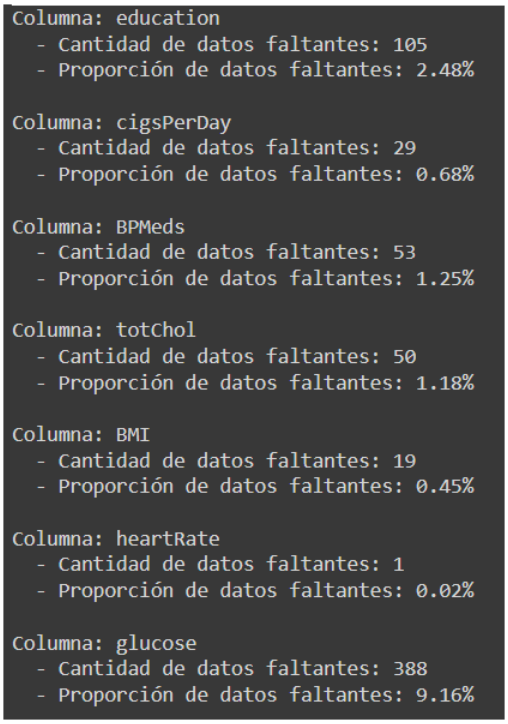

Ilustración 2: Porcentaje Datos Faltantes Dataset Original

Con el dataset limpio se aplicó feature Re-Scaling en el dataset para asemejar los valores y que no exista tanta distancia entre estos, a continuación, se utilizó una proporción 60-20-20 para el Train, Test y Validation set, utilizando la librería Numpy para elegir índices del dataset de forma aleatoria para asegurar que los dos sets de datos se mantengan equilibrados.

Entrenamiento Inicial y Predicciones

Para el entrenamiento inicial del modelo se utilizaron valores para Alpha de 0.01, un umbral de 0.5, y un total de mil epochs, en la ilustración 3 se puede apreciar como baja el error en relación con la cantidad de iteraciones que se realizan, en cambio en la ilustración 4 se puede percibir una comparación entre las primeras 10 predicciones y los valores reales.

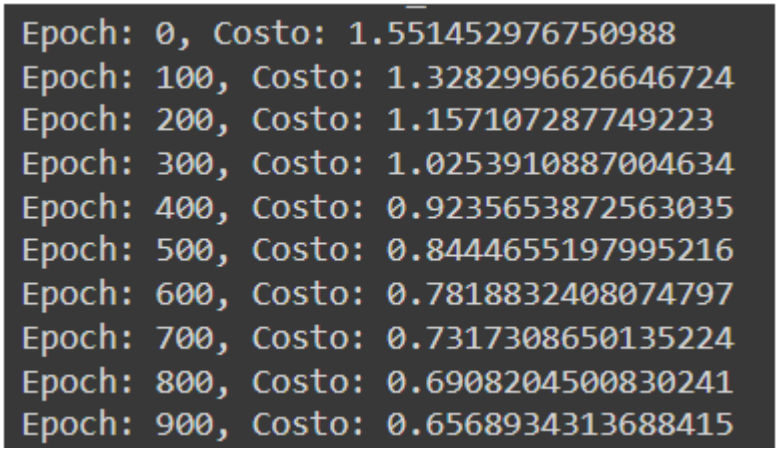

Ilustración 3: Relación Costo - Epoch

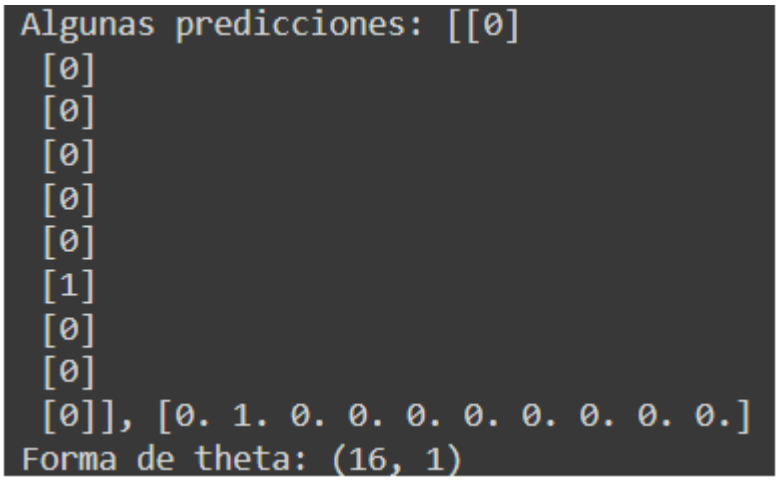

Ilustración 4: Comparación Predicción - Valor

Evaluación del Modelo

A la hora de comenzar a evaluar el modelo se realizaron 4 métodos distintos (agregados al MSE), siendo estos: matriz de confusión, precisión and recall, y accuracy.

Debido a que dentro de la matriz de confusión (ilustración 5) se identificó una inclinación bastante grande de datos hacia un “True Negative” se calculó el porcentaje de positive y negative del target en el dataset global, denotando una desproporción de los datos, siendo un 84.80% de los valores del label negativos y solo un 15.2% positivos, para estos casos se cuenta con diversos

métodos para balancear esta desproporción, entre ellos se puede implementar dentro del dataset upsampling, downsampling, modificar el valor del umbral.

Dentro de la ilustración 5 se puede apreciar los valores de precisión y recall que produjo el modelo, siendo estos bastante bajos, que complementado con la matriz de confusión se puede denotar que el modelo tiene una inclinación a predecir resultados como negativos, esto dado muy probablemente a causa del desbalance en los datos

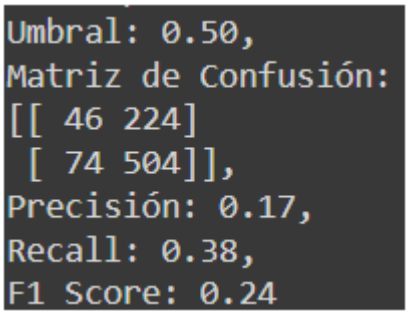

Ilustración 5: Evaluación Inicial Modelo

Mejora y Optimización del Modelo

Dentro de esta sección se expondrá en orden cronológico las modificaciones realizadas al modelo en busca de mejorar su capacidad para realizar predicciones correctas.

El primer proceso realizado fue modificar el valor del umbral, que originalmente se situó en 0.5, el valor más común que se suele utilizar, se probó el modelo con umbrales desde 0.1 hasta 0.9 aumentando en 0.1 el valor de este, dando como resultado valores de precisión muy bajos, algo que por las características del problema deseamos aumentar en la mayor medida posible, ya que como se indica en la descripción de Kaggle, al predecir si un paciente tendrá problemas cardiacos en los próximos 10 años, es preferible no detectar algunos positivos antes que detectar un negativo incorrectamente.

Al notar que el valor del umbral no alteraba los resultados, se optó por realizar downsampling en el dataset original, reestructurando en una proporción de 50:50 entre positivos y negativos; además de agregar a la evaluación del modelo F1 Score, que sirve para encontrar un punto medio entre pecision y recall. Para evaluar si este método era más efectivo se utilizaron valores para el umbral desde 0.01 hasta 1 con intervalos de 0.01, en la ilustración 6 se puede apreciar que para todos los casos el mejor umbral fue 0.01, dando un recall del 98% y una precisión del 50%

```
Mejor Precision: 0.50 en umbral 0.01
Mejor Recall: 0.98 en umbral 0.01
Mejor F1 Score: 0.66 en umbral 0.01
```

Ilustración 6: Evaluación Modelo

Los resultados mostrados en la ilustración 6 fueron con un total de 101 epochs y un valor de 0.01 para Alpha, ya que solo se buscaba encontrar si el modelo mejoraba su precisión al realizar downsampling, que en efecto sucedió, ahora se realizó grid search para evaluar los mejores valores de los hiperparametros del modelo, en la ilustración 7 se puede apreciar los valores que se utilizaron para medir esto.

```
alpha_values = [0.001, 0.01, 0.1]
epochs_values = [1000, 1500, 2000]
umbrales = np.arange(0.2, 0.51, 0.01)
```

Ilustración 7: Hiperparametros

Se obtuvo como resultado que el mejor valor para aplha es 0.01, la cantidad de epochs fue 2000 y el umbral 0.22, pero justo después de esta prueba se modificó la función de log_regresion para que no funcione por cantidad de epochs, sino por la diferencia entre el epoch actual con el previo, agregando un nuevo hiperparametro a la ecuación.

Realizando un grid search con diferentes valores para Alpha, utilizando cinco mil epochs por modelo, se buscó localizar el mejor valor de este para un modelo entrenado, al igual que revisar de forma manual cual podría ser un buen número de epochs, coincidiendo en que cinco mil eran incluso demasiados, ya que la disminución del error era minúscula a partir de los 500 en la mayoría de los casos, dando el mismo resultado para Alpha con valores 0.01, 0.1, 0.5 y 1; aunque en estos llegando al mismo error en un epoch diferente, el que menos iteraciones requirió fue el valor de 1, por lo que se optó por utilizar este, dando saltos grandes en el entrenamiento del modelo.

Finalmente se decidió olvidar los experimentos anteriormente realizados, comenzando de cierta forma desde 0, pero manteniendo ciertos factores, por ejemplo, el número de epochs sigue siendo 500, Alpha se probó con los valores: 0.1, 0.5, 1.0. Para el umbral se encontró un patrón, en el que los mejores resultados se daban cuando este estaba en un rango entre 0.4 y 0.6.

En la ilustración 8 se puede ver el resultado final para cada valor de Alpha, datos con los cuales llegamos a la conclusión de este problema, notando que para Alpha 0.5 y 1 es el mejor valor de accuracy, nos incita a querer irnos por alguno de estos, aunque como para este problema en particular es deseable priorizar la precisión del modelo, el valor de 0.1 para Alpha entra en competencia al tener el valor más alto de precisión.

Para elegir los valores específicos para los hiperparametros se analizó el valor de cada uno de estos en diferentes umbrales, y como se puede apreciar en la ilustración 8 para Alpha 0.1 el umbral de la mejor precisión y el del mejor accuracy son muy cercanos, justo esto es lo que motiva revisar más a fondo esto, llegando a la conclusión de que aquí se encuentra el mejor modelo. Finalmente se utilizó un umbral de 0.56, ya que este aunque no tenía el máximo de ninguno, si tenía la mayor estabilidad entre estos, maximizando la precisión y accuracy, sin sacrificar demasiado el recall, como se puede apreciar en la ilustración 9.

```
Mejores combinaciones de alpha, epochs y umbral:
Alpha: 0.1, Epochs: 501, Umbral: 0.57, Mejor Precision: 0.72, Umbral: 0.40, Mejor Recall: 0.77, Umbral: 0.44, Mejor F1 Score: 0.69, Umbral: 0.55, Mejor Accuracy: 0.69
Alpha: 0.5, Epochs: 501, Umbral: 0.61, Mejor Precision: 0.70, Umbral: 0.40, Mejor Recall: 0.79, Umbral: 0.40, Mejor F1 Score: 0.69, Umbral: 0.48, Mejor Accuracy: 0.70
Alpha: 1.0, Epochs: 501, Umbral: 0.61, Mejor Precision: 0.70, Umbral: 0.40, Mejor Recall: 0.79, Umbral: 0.40, Mejor F1 Score: 0.69, Umbral: 0.48, Mejor Accuracy: 0.70
```

Ilustración 8: Resultados Finales Modelo

```
Umbral: 0.55,
Matriz de Confusión: [73 31],[ 49 109], Precisión: 0.70, Recall: 0.60, F1 Score: 0.65, Accuracy: 0.69
Umbral: 0.56,
Matriz de Confusión: [71 29],[ 51 111], Precisión: 0.71, Recall: 0.58, F1 Score: 0.64, Accuracy: 0.69
Umbral: 0.57,
Matriz de Confusión: [69 27],[ 53 113], Precisión: 0.72, Recall: 0.57, F1 Score: 0.63, Accuracy: 0.69
Umbral: 0.58,
```

Ilustración 9: Valores con Alpha 1

Los valores para los hiperparametros utilizados en la versión final son los siguientes:

Alpha: 0.1

Epochs: 500

Umbral: 0.56

Con los valores para los hiperparametros propuestos anteriormente se puede conseguir un modelo con un buen fitt, ya que un 69% de accuracy nos indica que tiene una buena generalización para predecir datos futuros, eliminando las sospechas de una varianza alta evitando el overfitting, por otro lado asegura estar cubiertos contra un alto bias por el buen comportamiento que tiene no solo en los datos de prueba, sino en el test, que es el punto principal que nos indica un bajo bias.

Considerando todo podemos concluir que se ha encontrado modelo con una buena generalización, que en promedio tiene un acierto del 72% en la identificación de positivos, y un 69% de rendimiento general, valores que nos aseguran librarnos del under u overfitting.

Conclusiones Personales

Al iniciar este portafolio pensé que lo más complicado sería la realización del modelo en sí, después se me presentó el primer reto incluso antes de siquiera comenzar con el modelado, esta adversidad fue la limpieza de los datos, que afortunadamente ya lo había realizado para el dataset del reto, por lo que pude utilizar la lógica implementada en ello para este caso, permitiéndome limpiar de forma acorde los datos, aunque después note un desbalance impresionante en los datos, que estaba alterando el aprendizaje de mi modelo, por lo que tuve que implementar algún método para corregir este desbalance, comencé modificando el umbral pero me di cuenta que tenía que realizar up o downsampling, por lo que opté por downsampling para batallar en menor medida y porque la cantidad de instancias me lo permitía.

Corregido el dataset comencé a jugar con los hiperparametros de muchas maneras, obteniendo resultados distintos en la evaluación del modelo, hasta que finalmente decidí mejor realizar un grid search buscando los mejores valores para cada hiperparametro, llegando a la conclusión expuesta anteriormente.

Todo esto me hizo comprender que lo verdaderamente interesante en un modelo no es la versión inicial de este, es modificar los hiperparametros en busca de obtener valores óptimos para estos, algo que te llevará a tomar decisiones, ya que es casi imposible encontrar ciertos valores exactos para cada parámetro que optimice el resultado en cada evaluación, usualmente tendrás que optar entre una mayor precisión o recall y basado en esto incluso si prefieres tener un buen accuracy o prefieres optimizar los positivos o negativos, es realmente interesante este aspecto y como elegir que factores afectan tu modelo y que decisión tomarás para llegar al objetivo planteado.