Tecnológico de Monterrey

Implementación de una técnica de aprendizaje máquina sin el uso de un framework

Maria Fernanda Argueta Wolke A00830194

Inteligencia artificial avanzada para la ciencia de datos I

(Gpo 102)

Regresión Logística

Para este portafolio se estuvo trabajando en la implementación de una regresión logística. La cual fue elaborada en Python sin el uso de frameworks y a partir de funciones generadas en clase. Para entrar más en contexto, la regresión logística permite estimar la probabilidad que ocurra un evento en función de un conjunto de datos y variables independientes.

En este caso se estará trabajando con una regresión logística binomial, esta regresión permite predecir un label siempre y cuando este sus valores sean 0s y 1s. La regresión binaria permite modelar las relaciones entre una variable categórica binaria y un conjunto de variables explicativas. Para este proyecto se estará trabajando con una base de datos extraída de Kaggle, específicamente la de ¨Sleep Health and Lifestyle Dataset¨

Sleep Health and Lifestyle contiene información de 400 pacientes, en donde se detallan datos sobre su sueño y estilo de vida en general. Se cuenta específicamente con la información sobre el género, edad, ocupación, duración del sueño, calidad del sueño, nivel de actividad física, niveles de estrés, categoría de IMC, presión arterial, frecuencia cardíaca, pasos diarios y la presencia o ausencia de trastornos del sueño.

Para la evaluación del modelo de regresión logística se tomaron en cuenta dos variables independientes las cuales fueron horas de sueño y nivel de estrés para hacer predicciones sobre la calidad de sueño.Por otro lado, se determina la calidad del sueño en una escala de 1-10. Por ello se tuvo que establecer un rango para convertir los datos de la variable en binarios para ajustar y poder predecir la calidad de sueño pudiera con una regresión logística binomial.

Sin embargo, se pudo observar que los valores de calidad de sueño oscilan entre 4 y 9. Se propuso un rango de 1 a 6 como un sueño de calidad baja, y en un rango de 7 a 10

como un sueño de muy buena calidad. Para lograrlo se generó una nueva columna en la que se indica con 1 si la calidad es alta y con un 0 si es baja en base a los criterios mencionados anteriormente.

Explicación del código

Principalmente el código se encuentra dividido en dos archivos, uno que contiene las funciones necesarias para realizar la regresión (model_fun.py). Por otro lado, se cuenta con el archivo principal que carga los datos, entrena el modelo y genera predicciones (main.py). En la misma carpeta se encuentra la base de datos extraida de kaggle (Sleep_health.csv).

Model_fun.py

Model fun permite crear el modelo de regresión logística, este incluye a su vez dos funciones las cuales se llaman entre sí para el correcto funcionamiento del modelo.

- **sigmoid_function**

  Parámetros: X (array o DataFrame): Matriz de valores numéricos.

  Retorno: array: Matriz de probabilidades resultantes de aplicar la función sigmoide a cada elemento de X.

  Descripción: Aplica la función sigmoide a cada valor en la entrada X. La función sigmoide se utiliza en la regresión logística para convertir el resultado lineal en una probabilidad entre 0 y 1.

- **log_regression4**

Parámetros: X (array o DataFrame): Se espera que sea una matriz de tamaño (n_samples, n_features).

y (array): Valores de la variable objetivo.

alpha (float): Tasa de aprendizaje para la actualización de los parámetros del modelo.

epochs (int): Número máximo de épocas para el entrenamiento del modelo.

Retorno: numpy array: Los parámetros óptimos del modelo después del entrenamiento.

Descripción: Entrena un modelo de regresión logística utilizando el descenso de gradiente. Actualiza los parámetros del modelo en cada iteración basada en la tasa de aprendizaje y el gradiente calculado. La función también implementa un criterio de parada basado en el cambio en la función de pérdida entre épocas.

- manual_split

Parámetros: X (DataFrame): Características del conjunto de datos.

y (Series): Valores de la variable objetivo.

test_size (float): Proporción del conjunto de datos que se utilizará para la prueba.

random_state (int, opcional): Semilla para la generación de números aleatorios.

Retorno: X_train (DataFrame): Conjunto de entrenamiento de las características.

X_test (DataFrame): Conjunto de prueba de las características.

y_train (Series): Conjunto de entrenamiento de la variable objetivo.

y_test (Series): Conjunto de prueba de la variable objetivo.

train_index (list): Índices utilizados para el conjunto de entrenamiento.

test_index (list): Índices utilizados para el conjunto de prueba.

Descripción: Divide el conjunto de datos en conjuntos de entrenamiento y prueba de acuerdo con el tamaño de prueba especificado. La partición es aleatoria, y se utiliza la semilla proporcionada para garantizar la reproducibilidad.

- **manual_scale**

  Parámetros: X_train (DataFrame): Conjunto de características de entrenamiento.

  X_test (DataFrame): Conjunto de características de prueba.

  Retorno: X_train_scaled (DataFrame): Conjunto de características de entrenamiento escalado.

  X_test_scaled (DataFrame): Conjunto de características de prueba escalado.

  Descripción: Escala las características del conjunto de entrenamiento y del conjunto de prueba utilizando la media y la desviación estándar del conjunto de entrenamiento. Esto asegura que los datos de prueba se escalen de la misma manera que los datos de entrenamiento.

- **calculate_accuracy**

Parámetros: y_test (array): Valores verdaderos de la variable objetivo para el conjunto de prueba.

y_pred (array): Valores predichos por el modelo.

Retorno: float: Precisión del modelo, calculada como la proporción de predicciones correctas entre el total de predicciones.

Descripción: Calcula la precisión del modelo comparando las predicciones del modelo con los valores verdaderos del conjunto de prueba. La precisión se define como la proporción de predicciones correctas respecto al total de predicciones.

Main.py

main.py es el script principal que carga los datos, entrena un modelo de regresión logística utilizando las funciones de model_fun.py, y evalúa el rendimiento del modelo en un conjunto de prueba. También muestra ejemplos de predicciones para casos individuales.

- Carga el DataFrame: Lee un archivo CSV para cargar el conjunto de datos.
- Preprocesamiento:
    - Crea una columna binaria de calidad del sueño.
    - Selecciona características y variable objetivo.
    - Divide el conjunto de datos en entrenamiento y prueba.
    - Escala las características.
- Entrenamiento del Modelo:

    - Utiliza log_regression4 para entrenar el modelo de regresión logística con los datos preprocesados.

- Evaluación:
    - Calcula la precisión del modelo utilizando calculate_accuracy.
    - Imprime la precisión del modelo.
- Predicción:
    - Realiza predicciones sobre un índice específico del conjunto de prueba.
    - Imprime la probabilidad predicha, la clase predicha y el valor real de calidad del sueño.

Conclusiones

- Se generó correctamente un modelo de regresión logística en Python sin el uso de librerías externas. Explorando así más a fondo el funcionamiento de cada componente que conforma el programa final.
- El modelo predijo la calidad de sueño de los sujetos en base a su nivel de estrés y horas de sueño. Logrando resultados con un 98% de accuracy según las métricas obtenidas en el programa.
- La implementación del modelo proporcionó una experiencia valiosa para el aprendizaje tanto en la construcción como en el ajuste del modelo de regresión logística para adaptarlo a un nuevo conjunto de datos.

Referencias.

- Sleep Health and Lifestyle Dataset. (2023). Kaggle. https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset