Módulo 2 Implementación de una técnica de aprendizaje máquina sin el uso de un framework.

Portafolio Implementación

Luis Ubaldo Balderas Sánchez - A01751150

Jueves, 05 de septiembre de 2025

Inteligencia artificial avanzada para la ciencia de datos

Grupo 101

1. Crea un repositorio de GitHub para este proyecto.
    a. https://github.com/Luiss1715/-Implementacion_algoritmo_ML

2. Programa uno de los algoritmos vistos en el módulo (o que tu profesor de módulo autorice) sin usar ninguna biblioteca o framework de aprendizaje máquina, ni de estadística avanzada. Lo que se busca es que implementes manualmente el algoritmo, no que importes un algoritmo ya implementado.

    a. Se implementó una Regresión logística y el modelo logra predecir clases binarios ya que se usó una función de activación sigmoide. El modelo utiliza una combinación lineal de las variables de entrada y aplica la función sigmoide para obtener una probabilidad entre 0 y 1.

    Durante el entrenamiento, se ajustan los pesos y el bias con el método de descenso de gradiente, minimizando el error de clasificación y una vez entrenado, el modelo permite predecir la clase de nuevos datos comparando la probabilidad contra un umbral (por defecto 0.5). Finalmente, para poder trabajar con cualquier dataset, las variables categóricas se transforman automáticamente con one-hot encoding, convirtiendo cada categoría en columnas binarias (0/1)

    b. Para la evaluación se usaron la matriz de confusión y métricas estándar: accuracy, precision, recall y F1-score.

3. Prueba tu implementación con un set de datos y realiza algunas predicciones. Las predicciones las puedes correr en consola o las puedes implementar con una interfaz gráfica apoyándote en los visto en otros módulos.

Para probar el modelo se utilizaron 2 datasets.

1. un dataset de mushrooms donde todas las variables son categóricas y la clase objetivo es binaria

2. un dataset de tenis donde la clase objetivo es binaria y se predice si se juega o no.

4. Tu implementación debe de poder correr por separado solamente con un compilador, no debe de depender de un IDE o de un “notebook”. Por ejemplo, si programas en Python, tu implementación final se espera que esté en un archivo .py no en un Jupyter Notebook.

Requisitos del modelo:

- Dataset en CSV con encabezados.
- La columna objetivo (--target) puede ser 0/1 o texto binario común (por ejemplo: yes/no, true/false, e/p, etc.).
- Las columnas categóricas se transforman automáticamente con one-hot encoding.

El modelo corre a través de la terminal con las siguientes opciones:

- python implementacion_ML.py --data <ruta_al_csv> --target <columna_objetivo> [opciones]

Opciones

--data (obligatorio): ruta al archivo CSV de entrada.

--target (obligatorio): nombre de la columna objetivo binaria.

--test-size (opcional, default 0.3): fracción para el conjunto de prueba.

--lr (opcional, default 0.1): learning rate del descenso de gradiente.

--epochs (opcional, default 1000): número de épocas de entrenamiento.

--train-out (opcional, default train_dataset.csv): ruta donde guardar el CSV de entrenamiento.

--test-out (opcional, default test_dataset.csv): ruta donde guardar el CSV de prueba.

5. Añade al repositorio un documento PDF en el que presentes los resultados de tu algoritmo al ser ejecutado.

   Resultados con dataset de mushroom:

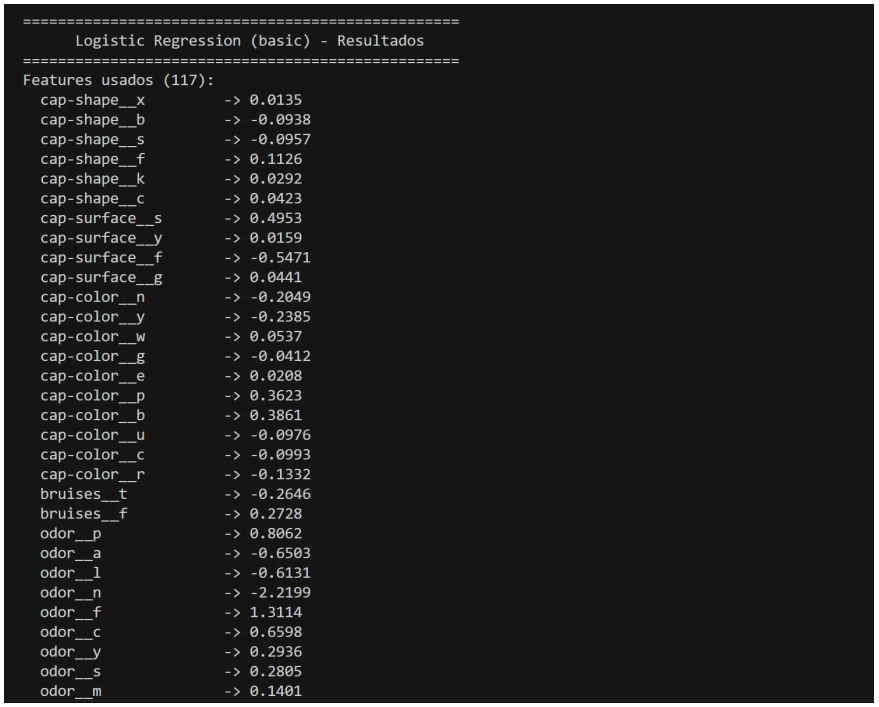

Imagen 1. Resultados con dataset Mushroom

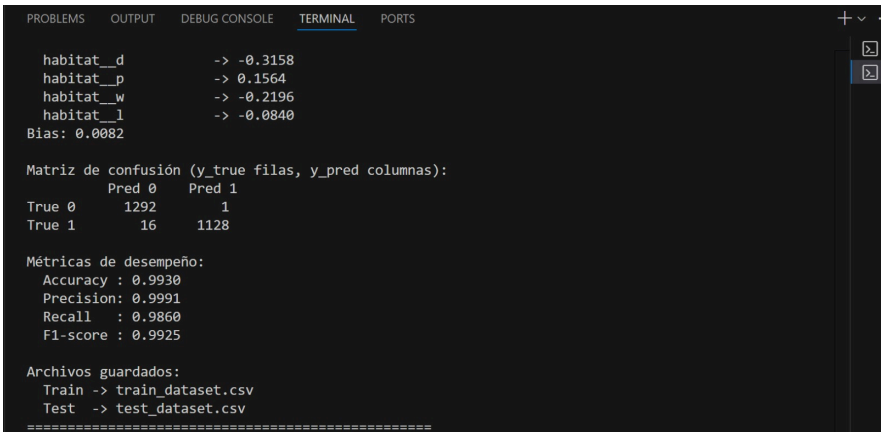

Imagen 2. Resultados con dataset Mushroom

6. Debes de indicar cuál fue el dataset con el que se entrenó y cuál con el que se probó.

train dataset: train_dataset.xlsx

test dataset: test_dataset.xlsx

7. Justificar utilizando matriz de confusión y métricas (tu seleccionas las adecuadas)

   Interpretación de resultados:

   True 0 = 1292 → 1292 casos que eran de la clase 0 (negativos, por ejemplo comestible) fueron clasificados correctamente.

   FP = 1 → solo 1 error de falso positivo (un caso negativo fue predicho como positivo).

   FN = 16 → 16 falsos negativos (casos positivos que el modelo no detectó).

   TP = 1128 → 1128 verdaderos positivos (predijo correctamente la clase positiva, por ejemplo venenoso).

   Accuracy = 0.9930 → el 99.3% de las predicciones fueron correctas.

   Precision = 0.9991 → de todas las predicciones positivas, el 99.9% fueron realmente positivas.

   Recall = 0.9860 → detecta el 98.6% de los positivos.

   F1-score = 0.9925 → buen equilibrio entre precisión y recall.

8. Un breve análisis y conclusión sobre el desempeño obtenido.

   El modelo de regresión logística implementado desde cero mostró un desempeño sobresaliente. Con un accuracy de 99.3%, una precisión de 99.9% y un recall de 98.6%, el algoritmo logra clasificar casi todos los ejemplos de manera correcta, con muy pocos falsos positivos y falsos negativos. Esto refleja que la técnica de one-hot encoding para variables categóricas y el entrenamiento mediante descenso de gradiente permitieron capturar adecuadamente los patrones del dataset. En conclusión, los resultados confirman que la regresión logística es un modelo adecuado y efectivo para este tipo de problema de clasificación binaria.

   Finalmente, en el repositorio se encuentran cargados 3 datasets para poder probar el modelo de regresión con alguno de ellos.