Momento de Retroalimentación: Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución.

Portafolio Implementación

Luis Ubaldo Balderas Sánchez - A01751150

Miércoles, 10 de septiembre de 2025

Inteligencia artificial avanzada para la ciencia de datos

Grupo 101

Crea un repositorio de GitHub para este proyecto.

- https://github.com/Luiss1715/-Implementacion_algoritmo_ML_libreria

Programa uno de los algoritmos vistos en el módulo (o que tu profesor de módulo autorice) haciendo uso de una biblioteca o framework de aprendizaje máquina. Lo que se busca es que demuestres tu conocimiento sobre el framework y cómo configurar el algoritmo.

Se implementó una Regresión logística. En concreto se utilizó la clase LogisticRegression de sklearn.linear_model, que ya implementa internamente el algoritmo de regresión logística. Esta librería nos permitió:

1. Configurar fácilmente hiperparámetros como max_iter (número de iteraciones) o el solver (algoritmo de optimización).
2. Entrenar el modelo directamente con .fit(X_train, y_train).
3. Generar predicciones con .predict(X_test).
4. Obtener probabilidades de clase con .predict_proba(X_test).

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)

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

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.

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 binaria (0/1) o texto binario común (ej: yes/no, true/false, pass/fail).

Las columnas categóricas se transforman automáticamente con one-hot encoding.

Las columnas numéricas se escalan con StandardScaler.

Ejecución en terminal:

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

Opciones disponibles:

- --data (obligatorio): ruta al archivo CSV de entrada.
- --target (obligatorio): nombre de la columna objetivo (binaria o multiclase).
- --test-size (opcional, default = 0.3): fracción para el conjunto de prueba.
- --epochs (opcional, default = 1000): número de iteraciones máximas del optimizador (max_iter en scikit-learn).
- --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.

Ejemplo de uso:

- python modelo_con_libreria.py --data heart.csv --target HeartDisease

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

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

Resultados:

```
PS C:\Users\Luis\Documents\7_semestre\Uresti\Tarea2_ModeloConLibreria> python modelo_con_libreria.py --data mushroo
ms.csv --target class

================================================
      Logistic Regression (scikit-learn) - Resultados
================================================
Features usados (117):
  cap-shape__x         -> -0.1811
  cap-shape__b         -> 0.2465
  cap-shape__s         -> -0.5448
  cap-shape__f         -> 0.0007
  cap-shape__k         -> 0.0033
  cap-shape__c         -> 0.5268
  cap-surface__s       -> 0.2815
  cap-surface__y       -> -0.0268
  cap-surface__f       -> -0.7975
  cap-surface__g       -> 0.5942
```

Imagen 1. Resultados con dataframe mushroom.csv

```
Bias: 0.1695

Matriz de confusión (y_true filas, y_pred columnas):
          Pred 0    Pred 1
True 0      1293         0
True 1         0      1144

Métricas de desempeño:
  Accuracy : 1.0000
  Precision: 1.0000
  Recall   : 1.0000
  F1-score : 1.0000

Archivos guardados:
  Train -> train_dataset.csv
  Test  -> test_dataset.csv
================================================

PS C:\Users\Luis\Documents\7_semestre\Uresti\Tarea2_ModeloConLibreria>
```

Imagen 2. Métricas del resultado

Un breve análisis y conclusión sobre el desempeño obtenido.

1. Bias (intercepto) = 0.1695

    El bias es el término independiente de la regresión logística.

    En este caso es positivo y pequeño, lo cuál indica un ligero sesgo hacia predecir la clase positiva, pero en la práctica lo importante es que los coeficientes + bias permiten separar perfectamente las clases.

2. Matriz de confusión

    1293 casos negativos (0) se predijeron correctamente como 0.

    1144 casos positivos (1) se predijeron correctamente como 1.

    No hubo errores → 0 falsos positivos y 0 falsos negativos.

    Esto significa que el modelo clasificó perfectamente todos los ejemplos del conjunto de prueba.

3. Métricas

    Accuracy = 1.0 → 100% de aciertos.

    Precision = 1.0 → todos los predichos como positivos realmente lo son.

    Recall = 1.0 → el modelo encontró todos los positivos reales.

    F1-score = 1.0 → balance perfecto entre precisión y exhaustividad.

Dado que el resultado fue “perfecto” probemos con el dataframe heart.csv

test dataset: test_dataset (1).xlsx

train dataset: train_dataset (1).xlsx

```
PS C:\Users\Luis\Documents\7_semestre\Uresti\Tarea2_ModeloConLibreria> python modelo_con_libreria.py --data heart.c
sv --target HeartDisease
C:\Users\Luis\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.13_qbz5n2kfra8p0\LocalCache\local-packages\P
ython313\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning: lbfgs failed to converge after 10
00 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=1000).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(

================================================
      Logistic Regression (scikit-learn) - Resultados
================================================
Features usados (20):
```

Imagen 3. resultados con data frame heart.csv

```
  Bias: -1.1618

  Matriz de confusión (y_true filas, y_pred columnas):
            Pred 0    Pred 1
  True 0        95        22
  True 1        20       138

  Métricas de desempeño:
    Accuracy : 0.8473
    Precision: 0.8625
    Recall   : 0.8734
    F1-score : 0.8679

  Archivos guardados:
    Train -> train_dataset.csv
    Test  -> test_dataset.csv
  ================================================
```

Imagen 4. Métricas del resultado con dataframe heart.csv

1. Bias (intercepto) = -1.1618

   El sesgo negativo indica que, en ausencia de variables predictoras, el modelo tiende a favorecer la clase 0 (negativa). Es decir, la línea de decisión está ligeramente desplazada hacia predecir negativos.

2. Matriz de confusión

95 verdaderos negativos (TN): clase 0 correctamente identificada.

138 verdaderos positivos (TP): clase 1 correctamente identificada.

22 falsos positivos (FP): predijo 1 pero en realidad era 0.

20 falsos negativos (FN): predijo 0 pero en realidad era 1.

El modelo comete errores en ambos sentidos (FP y FN), pero la mayoría de las predicciones son correctas.

3. Métricas

Accuracy = 0.8473 (84.7%) → el modelo acierta en casi 85 de cada 100 casos.

Precision = 0.8625 → de los predichos como positivos, ~86% realmente lo eran.

Recall = 0.8734 → el modelo identificó correctamente ~87% de los positivos reales.

F1-score = 0.8679 → balance adecuado entre precisión y exhaustividad.

Un breve análisis y conclusión sobre el desempeño obtenido.

En los dos experimentos realizados con el modelo de Regresión Logística se observaron comportamientos contrastantes. En el primer caso, el modelo alcanzó un desempeño perfecto (accuracy, precision, recall y F1-score = 1.0), lo que indica una separación total de las clases en el conjunto de prueba. Sin embargo, este resultado también sugiere la necesidad de validar con más datos para descartar sobreajuste o fuga de información. En el segundo caso, el modelo obtuvo un rendimiento sólido pero no perfecto (accuracy ≈ 85%, precision ≈ 86%, recall ≈ 87%), mostrando un balance adecuado entre precisión y exhaustividad, aunque con algunos errores de clasificación (falsos positivos y falsos negativos). En conclusión, la regresión logística demostró ser efectiva para la tarea de clasificación: puede alcanzar desempeños sobresalientes cuando las variables discriminan claramente las clases, y mantiene resultados competitivos en escenarios más complejos, lo que la convierte en una herramienta confiable siempre que se acompañe de una correcta validación.