Código para Regresión Logística

Realizado por:

- Samuel Leonardo Tarazona Arciniegas
- Julian Andres Leon Montoya

Enlace repositorio

- https://github.com/03SamuelTarazona/MachineLearning

📖 Resumen / propósito

El objetivo de este código es aplicar un modelo de **Regresión Logística** para la clasificación de correos electrónicos en dos categorías:

- **HAM (0):** correos legítimos.
- **SPAM (1):** correos no deseados.

El proceso incluye:

1. Carga y exploración del dataset.
2. Separación de variables (características y etiqueta).
3. Preprocesamiento de datos (escalado con `StandardScaler`).
4. División de datos en entrenamiento y prueba.
5. Entrenamiento del modelo de regresión logística.
6. Evaluación del modelo con métricas como **F1-Score** y reporte de clasificación.
7. Visualización de los resultados mediante **matriz de confusión**.

🔷 1. Cargar el dataset

Se importa el archivo `dataset_spam_ham.csv` y se visualizan las primeras filas para inspección.

```
import pandas as pd

df = pd.read_csv("dataset_spam_ham.csv")

print("Primeras filas del dataset:") print(df.head())
```

🔷 2. Separar variables (X) y etiqueta (y)

Se eliminan las columnas de clase de las variables predictoras (X) y se almacena en y la variable objetivo:

```
X = df.drop("clase", axis=1)  # Features y = df["clase"]                # Target (0 = HAM, 1 = SPAM)
```

🔷 3. Transformación de valores (Preprocesamiento)

Se aplica StandardScaler para normalizar los datos y mejorar el rendimiento de la regresión logística.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler() X_scaled = scaler.fit_transform(X)
```

🔷 4. División en entrenamiento y prueba

Se realiza una división 80% entrenamiento y 20% prueba, estratificando para mantener la proporción de clases.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split( X_scaled, y, test_size=0.2, random_state=42, stratify=y )
```

🔷 5. Entrenamiento del modelo de Regresión Logística

Se entrena el modelo usando el solver liblinear con un número máximo de iteraciones de 1000.

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression(max_iter=1000, solver="liblinear")
log_reg.fit(X_train, y_train)
```

🔷 6. Evaluación del modelo

Se realizan predicciones sobre el conjunto de prueba y se calculan métricas de desempeño.

```
from sklearn.metrics import classification_report, f1_score

y_pred = log_reg.predict(X_test)

# Métrica principal: F1-Score f1 = f1_score(y_test, y_pred) print(f"\n✅ F1-Score del modelo: {f1:.3f}")

# Reporte detallado print("\nReporte de clasificación:")
print(classification_report(y_test, y_pred, target_names=["HAM (0)", "SPAM (1)"]))
```

🔷 7. Ecuación de la Regresión Logística

Se muestran los coeficientes de cada variable y el intercepto del modelo.

```
print("\nCoeficientes del modelo (uno por cada feature):") for feature, coef in zip(df.drop("clase", axis=1).columns, log_reg.coef_[0]): print(f"{feature}: {coef:.4f}")

print(f"\nIntercepto (bias): {log_reg.intercept_[0]:.4f}")
```

🔷 8. Visualización de resultados (Matriz de confusión)

Se grafica la matriz de confusión para analizar los aciertos y errores del modelo.

```
import matplotlib.pyplot as plt import seaborn as sns from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5)) sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["HAM (0)", "SPAM (1)"], yticklabels=["HAM (0)", "SPAM (1)"])
plt.title("Matriz de Confusión - Regresión Logística") plt.xlabel("Predicción")
plt.ylabel("Real") plt.show()
```

📊 Interpretación de resultados

- F1-Score: mide el equilibrio entre precisión y recall. Un valor cercano a 1 indica un excelente desempeño.
- Reporte de clasificación: incluye precisión, recall y F1-score por cada clase (HAM y SPAM).
- Matriz de confusión: permite visualizar el número de aciertos y errores:
- La diagonal muestra las predicciones correctas.
- Los valores fuera de la diagonal indican errores de clasificación.
- Coeficientes: permiten interpretar qué variables aumentan o disminuyen la probabilidad de que un correo sea SPAM.