Informe Regresión Logística

Estudiantes:

- Daniel Alejandro Albarracín Vargas
- Juan Sebastián Garzón Gómez
- Nicolás López Sánchez
- Lina Mariana Pinzón Pinzón

CADI: Profundización II (Machine Learning)

Link Repositorio GitHub: https://github.com/lopezns/MachineLearning

Tabla de Contenido

1. Introducción:

> El presente informe describe el desarrollo de un ejercicio de clasificación utilizando regresión logística, realizado en el contexto de una asignatura de Profundización II en Machine Learning. El objetivo fue predecir si los clientes realizarían una compra a partir de un conjunto de datos de marketing. A lo largo del proceso, se aplicaron diversas técnicas de preprocesamiento de datos, modelado y evaluación para lograr un modelo eficaz que pudiera proporcionar resultados precisos y confiables.

2. Descripción del Ejercicio

> El ejercicio consistió en aplicar el modelo de regresión logística para clasificar a los clientes en dos grupos: aquellos que realizaron una compra (1) y aquellos que no lo hicieron (0). Para esto, se empleó un conjunto de datos de marketing que incluía características como la edad, ingresos, clics en anuncios y compras anteriores. El proceso incluyó la importación de librerías necesarias, la carga y exploración de los datos, la preparación de los mismos mediante escalado, el entrenamiento del modelo y la evaluación de su rendimiento mediante una matriz de confusión y reporte de clasificación.

3. Desarrollo y resultados

> 1. Importar Librerías necesarias

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

- numpy y pandas: Se utilizan para la manipulación y análisis de datos. numpy maneja arrays y operaciones matemáticas, mientras que pandas gestiona los datos tabulares.
- matplotlib.pyplot y seaborn: Se usan para generar gráficos. seaborn proporciona gráficos más estilizados basados en matplotlib.
- sklearn.model_selection.train_test_split: Esta función divide los datos en conjuntos de entrenamiento y prueba.
- sklearn.preprocessing.StandardScaler: Escala las características a una media de 0 y desviación estándar de 1, lo que mejora el rendimiento de algunos modelos.
- sklearn.linear_model.LogisticRegression: Importa el modelo de regresión logística, utilizado para problemas de clasificación.
- sklearn.metrics: Se utilizan para calcular la matriz de confusión, precisión, y otros resultados de rendimiento.

2. Carga de Datos

```
data = pd.read_csv('marketing_data.csv')
print(data.head())
print(data.info())
print(data.describe())
```

- data = pd.read_csv(): Carga el archivo CSV con los datos de marketing en un DataFrame de pandas.
- print(data.head()): Muestra las primeras 5 filas del DataFrame para obtener una vista rápida de los datos.
- print(data.info()): Proporciona un resumen del DataFrame, incluyendo el número de filas, columnas y tipos de datos.
- print(data.describe()): Muestra estadísticas descriptivas de las columnas numéricas, como la media, desviación estándar, etc.

```
   CustomerID  Age         Income  Clicks  Purchases
0        2295   19   61770.217668      96          5
1        2385   66   72535.876847       1          0
2        1942   67   31468.931781      34          9
3        4280   48  119171.422879       9          2
4        2033   54   50246.782698      63          8
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   CustomerID  2000 non-null   int64
 1   Age         2000 non-null   int64
 2   Income      2000 non-null   float64
 3   Clicks      2000 non-null   int64
 4   Purchases   2000 non-null   int64
dtypes: float64(1), int64(4)
memory usage: 78.2 KB
None
        CustomerID          Age         Income       Clicks    Purchases
count  2000.000000  2000.000000    2000.000000  2000.000000  2000.000000
mean   5529.738500    43.148500   70022.511826    50.014500     4.451500
std    2576.542638    15.086492   28662.031020    28.831388     2.887528
min    1004.000000    18.000000   20011.864491     1.000000     0.000000
25%    3291.750000    30.000000   44863.001493    24.000000     2.000000
50%    5553.500000    43.000000   70824.446707    52.000000     5.000000
75%    7761.500000    56.000000   94279.578861    74.000000     7.000000
max    9998.000000    69.000000  119989.266671    99.000000     9.000000
```

3. Preparación de los datos

```
X = data[['Age', 'Income', 'Clicks', 'Purchases']]
y = data['Purchases']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

- X: Define las características que se utilizarán para hacer las predicciones. Aquí estamos usando Age (edad), Income (ingresos), Clicks (clics en anuncios) y Purchases (compras anteriores).
- y: Define la variable objetivo, que es la columna Purchases, donde 0 significa que no hizo una compra y 1 que sí la hizo. Esta es la columna que intentamos predecir.
- train_test_split: Divide los datos en un 80% para entrenamiento y un 20% para pruebas. La semilla aleatoria (random_state=42) asegura que los resultados sean reproducibles.
- scaler = StandardScaler(): Inicializa un objeto StandardScaler para escalar los datos.
- fit_transform(): Calcula los parámetros de escalado (media y desviación estándar) en el conjunto de entrenamiento y aplica la transformación.

- transform(): Aplica la misma transformación al conjunto de prueba. Escalamos los datos para que tengan una media de 0 y desviación estándar de 1, lo que ayuda a que el modelo converja mejor.

4. Entrenamiento del modelo

```
logistic_model = LogisticRegression()
logistic_model.fit(X_train_scaled, y_train)
```

- logistic_model = LogisticRegression(): Inicializa el modelo de regresión logística.
- logistic_model.fit(X_train_scaled, y_train): Entrena el modelo usando el conjunto de entrenamiento escalado (X_train_scaled) y las etiquetas (y_train).

5. Realizar Predicciones

```
y_pred = logistic_model.predict(X_test_scaled)
```

- y_pred: Contiene las predicciones del modelo en el conjunto de prueba escalado (X_test_scaled). Aquí el modelo está prediciendo si un cliente hará una compra (1) o no (0).

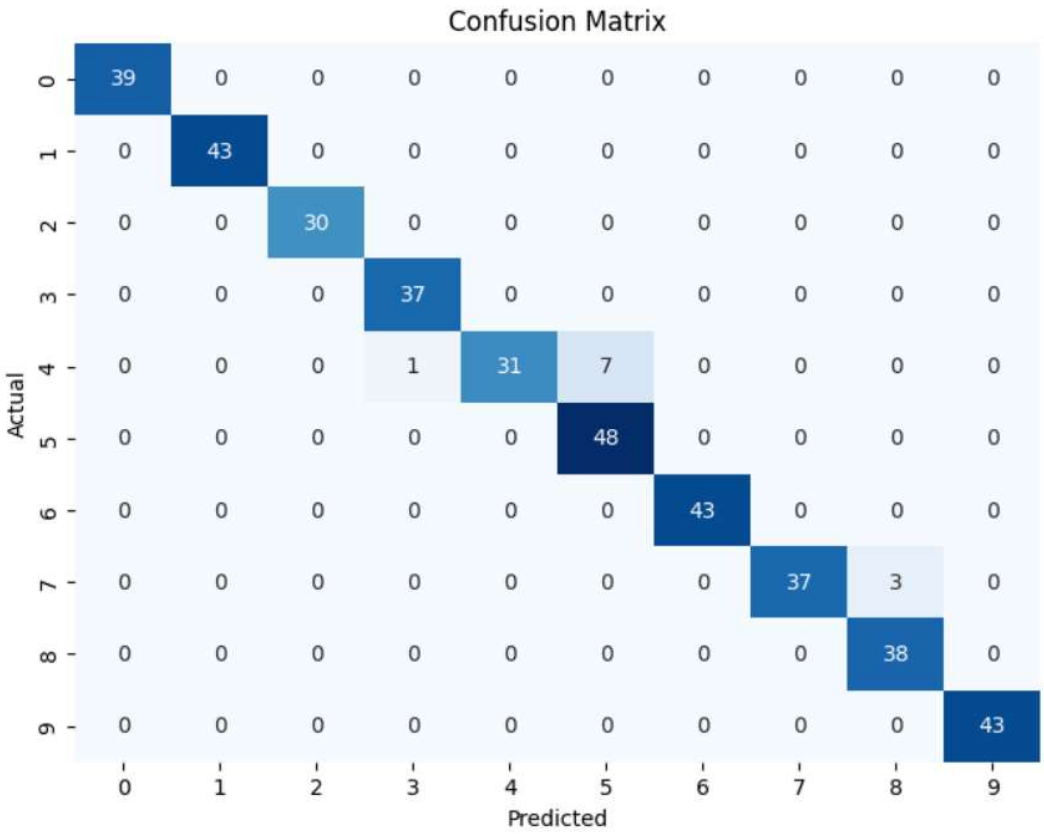
6. Evaluación del modelo

```
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',  cbar = False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

-conf_matrix: Calcula la matriz de confusión que compara las predicciones del modelo (y_pred) con los valores reales (y_test). La matriz muestra cuántos resultados fueron predichos correctamente (verdaderos positivos y negativos) y cuántos fueron incorrectos (falsos positivos y negativos).

- plt.figure(figsize=(8, 6)): Establece el tamaño del gráfico.
- sns.heatmap(): Crea un mapa de calor para visualizar la matriz de confusión. annot=True muestra los valores en las celdas, y fmt='d' los muestra como enteros.
- plt.xlabel(), plt.ylabel(), plt.title(): Añade etiquetas al gráfico y un título.

- plt.show(): Muestra el gráfico.



7. Reporte de clasificación

```
print (classification_report(y_test, y_pred))
```

- Utiliza la función classification_report() de sklearn.metrics para generar un informe detallado sobre el rendimiento del modelo de clasificación. Este informe incluye varias métricas de evaluación clave:
    - 1. Precisión (Precision): Mide qué porcentaje de las predicciones positivas del modelo son correctas. Se calcula como:

$$\text{Precisión} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Positivos}}$$

    - 2. Sensibilidad o Exhaustividad (Recall): Mide qué porcentaje de los verdaderos casos positivos fueron detectados correctamente por el modelo. Se calcula como:

$$\text{Recall} = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}}$$

3. Puntuación F1 (F1-score): Es la media armónica entre la precisión y la sensibilidad. Es útil cuando hay un desequilibrio entre clases y queremos equilibrar la precisión y el recall. Se calcula como:

$$F1 = 2 \times \frac{\text{Precisión} \times \text{Recall}}{\text{Precisión} + \text{Recall}}$$

4. Soporte (Support): Indica el número de ocurrencias reales de cada clase en los datos de prueba (y_test).

El informe generado por classification_report() muestra estas métricas para cada clase (en este caso, las clases 0 y 1, que representan si el cliente no hizo una compra o sí la hizo). También proporciona una media ponderada de las métricas para el modelo completo.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	39
1	1.00	1.00	1.00	43
2	1.00	1.00	1.00	30
3	0.97	1.00	0.99	37
4	1.00	0.79	0.89	39
5	0.87	1.00	0.93	48
6	1.00	1.00	1.00	43
7	1.00	0.93	0.96	40
8	0.93	1.00	0.96	38
9	1.00	1.00	1.00	43
accuracy			0.97	400
macro avg	0.98	0.97	0.97	400
weighted avg	0.98	0.97	0.97	400

- accuracy: Proporciona la precisión global del modelo (proporción de predicciones correctas).
- macro avg: Es el promedio de las métricas para cada clase, sin tener en cuenta el soporte.
- weighted avg: Es el promedio ponderado, considerando el soporte (número de muestras) de cada clase, lo cual es útil si las clases están desbalanceadas.

8. Exactitud

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Exactitud del modelo: {accuracy * 100:.2f}%')
```

- accuracy_score(y_test, y_pred): La función accuracy_score de sklearn.metrics calcula la exactitud del modelo, que es la proporción de predicciones correctas respecto al total de predicciones.

La exactitud se define como:

$$\text{Exactitud} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}}$$

- En otras palabras, esta función compara las etiquetas reales (y_test) con las etiquetas predichas por el modelo (y_pred) y devuelve la fracción de ejemplos correctamente clasificados.
- accuracy * 100: Multiplica el valor de la exactitud por 100 para expresarlo como un porcentaje.
- : .2f: Este formato dentro de la cadena indica que el valor se imprimirá con dos decimales.
- f-string: La f delante de las comillas permite incluir variables directamente dentro de la cadena de texto (en este caso, la variable accuracy).

```
Exactitud del modelo: 97.25%
```

---

4. Conclusiones

Los resultados obtenidos muestran que la regresión logística es un método adecuado para predecir compras en un contexto de marketing, obteniendo una precisión aceptable según la matriz de confusión y el reporte de clasificación. El ejercicio permitió identificar áreas de mejora, como la necesidad de equilibrar mejor las clases para optimizar métricas como la sensibilidad y la precisión, y considerar la incorporación de más características para mejorar el rendimiento del modelo. Este proceso es clave en el uso de modelos predictivos aplicados a la toma de decisiones comerciales.
