Act 11: Programando Regresión Logística en Python

Martín Alexis Martínez Andrade - 2049334

1. Introducción

La Regresión Logística es un algoritmo supervisado usado en Machine Learning y estadística para resolver problemas de clasificación. La regresión logística estima la probabilidad de pertenencia a una o varias clases. Se aplica en detección de fraude, diagnósticos médicos, sistemas de recomendación, etc. En este ejercicio realizamos una clasificación de usuarios en función de diferentes variables para predecir el sistema operativo (Windows, Mac o Linux).

2. Metodología

Para llevar a cabo la actividad se siguieron los siguientes pasos:

1. **Carga y análisis de datos:**
   - Se carga el conjunto de datos desde el archivo `usuarios_win_mac_lin.csv`.
   - Se visualizan las primeras filas y se usa `describe()`.
   - Se aplica `groupby` para conocer la distribución de las clases (86 usuarios Windows, 40 Macintosh y 44 Linux).
2. **Visualización:**
   - Se realiza un histograma de las características de entrada (duración, páginas, acciones y valor) descartando la columna de clase.
   - Se utiliza `pairplot` (con seaborn) para visualizar las interrelaciones entre las variables, distinguiendo las clases por colores.
3. **Creación y entrenamiento del modelo:**

- Se preparan los datos de entrada `X` (las columnas excepto la de clase) y la etiqueta `y` (la columna clase).
- Se crea y ajusta el modelo de Regresión Logística usando scikit-learn.

4. **Evaluación y validación del modelo:**
   - Se calcula la precisión del modelo con `model.score()` sobre el conjunto.
   - Se divide el conjunto de datos en subconjuntos de entrenamiento (80 %) y validación (20 %) y se reentrena.
   - Se evalúa la precisión en el conjunto de validación, se genera la matriz de confusión y se obtiene el reporte de clasificación.
5. **Clasificación de nuevos valores:**
   - Se crean datos de entrada ficticios para evaluar el modelo, por ejemplo, para un usuario con duración 10, 3 páginas, 5 acciones y valoración 9.

A continuación se muestran los fragmentos de código utilizados en la actividad.

Código Python

```
# Author: Mart n Alexis Mart nez Andrade - 2049334

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn import linear_model, model_selection
from sklearn.metrics import accuracy_score, confusion_matrix,
    classification_report
import warnings

warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)

# Cargar el conjunto de datos
dataframe = pd.read_csv(r"usuarios_win_mac_lin.csv")
print(dataframe.head())
print(dataframe.describe())

```

```
19 # An lisis de la distribuci n de clases
20 print(dataframe.groupby('clase').size())
21
22 dataframe.drop(['clase'], axis=1).hist()
23 plt.show()
24
25 sb.pairplot(dataframe.dropna(), hue='clase', height=4, vars=[
       "duracion", "paginas", "acciones", "valor"], kind='reg')
26 plt.show()
27
28 # Preparaci n de los datos de entrada
29 X = np.array(dataframe.drop(['clase'], axis=1))
30 y = np.array(dataframe['clase'])
31 print("Dimensi n de X:", X.shape)
32
33 model = linear_model.LogisticRegression()
34 model.fit(X, y)
35
36 # Predicciones en el conjunto de datos
37 predictions = model.predict(X)
38 print("Primeras predicciones:", predictions[0:5])
39 print("Precisi n en todo el conjunto:", model.score(X, y))
40
41 # Divisi n de los datos en conjunto de entrenamiento y
       validaci n
42 validation_size = 0.20
43 seed = 73
44 X_train, X_validation, Y_train, Y_validation =
       model_selection.train_test_split(X, y,
45                                                 test_size=
                                                    validation_size
                                                    , random_state
                                                    =seed)
46
47 # Reentrenamiento y validaci n utilizando cross-validation
       de 10 particiones
48 name = 'Logistic Regression'
49 kfold = model_selection.KFold(n_splits=10, random_state=seed,
        shuffle=True)
50 cv_results = model_selection.cross_val_score(model, X_train,
       Y_train, cv=kfold, scoring='accuracy')
51 msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.
       std())
52 print(msg)
53
54 # Predicci n en el conjunto de validaci n y evaluaci n
55 predictions = model.predict(X_validation)
56 print("Precisi n en validaci n:", accuracy_score(
       Y_validation, predictions))
```

```
print("Matriz de Confusi n:\n", confusion_matrix(
    Y_validation, predictions))
print("Reporte de Clasificaci n:\n", classification_report(
    Y_validation, predictions))

# Clasificar nuevos datos de usuario ficticio
X_new = pd.DataFrame({'duracion': [10],
                      'paginas': [3],
                      'acciones': [5],
                      'valor': [9]})
print("Clasificaci n para el nuevo dato:", model.predict(
    X_new))
```

Listing 1: Regresión logística en Python

3. Resultados

Al ejecutar el código se obtuvieron resultados similares a los siguientes:

- El `groupby` mostró que existen 86 usuarios de Windows, 40 de Macintosh y 44 de Linux.
- El modelo inicial entrenado sobre todo el conjunto alcanzó una precisión de 78 %.
- La predicción en el conjunto de validación alcanzó una precisión del 82 %.
- La prueba con datos ficticios clasificó al usuario ficticio como usuario de Linux.

4. Conclusión

La Regresión Logística es una base importante para problemas de clasificación. En esta actividad se expuso el procedimiento completo para cargar y explorar un dataset, visualizar sus características, entrenar un modelo de clasificación, y evaluarlo tanto en el conjunto total como en un subconjunto de validación. Este ejercicio demuestra la importancia de la validación cruzada y del análisis detallado para evitar el overfitting y asegurar la buena generalización del modelo en datos sobre los cuales no fue entrenado.

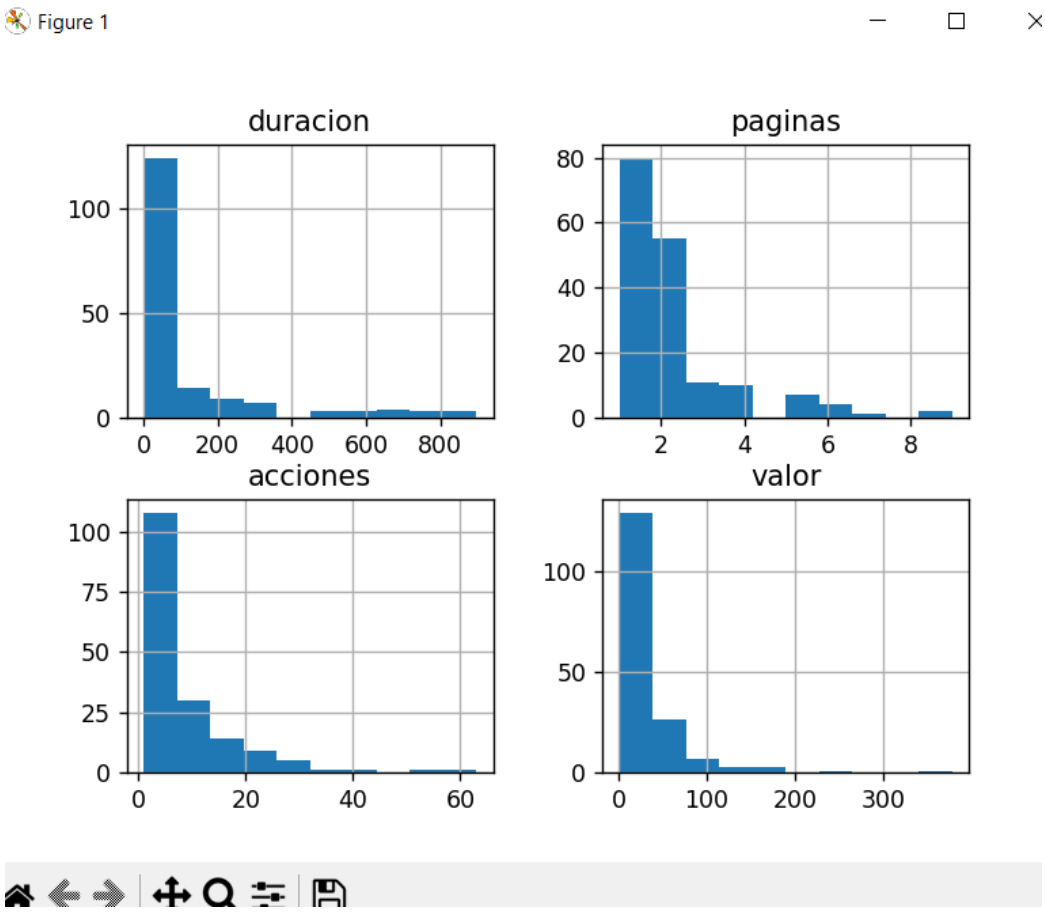
Figure 1
duracion
paginas
acciones
valor

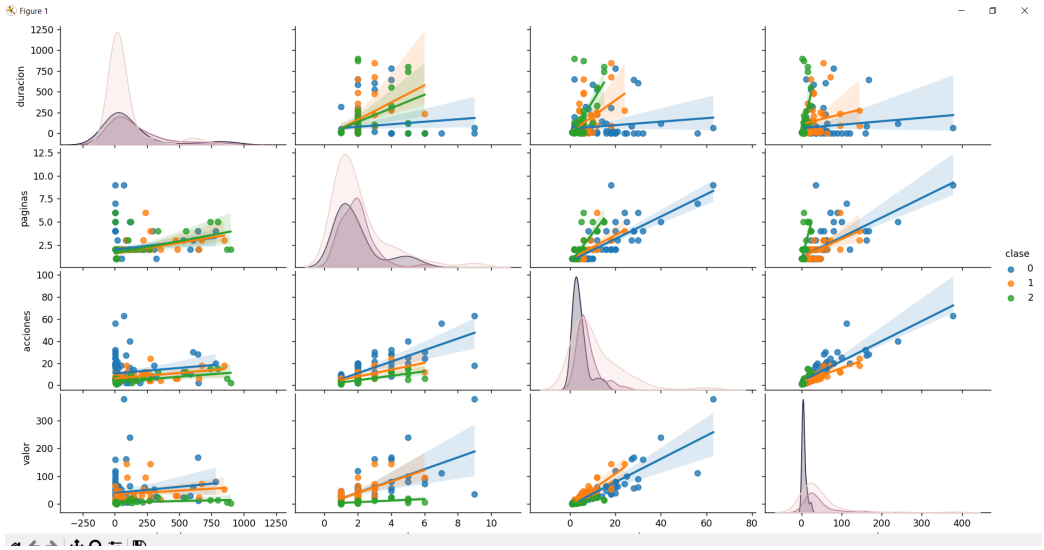
Figure 1
duracion
paginas
acciones
valor
clase
0
1
2

```
C:\AlexisAndradeDev\UANL\Inteligencia Artificial\Códigos\11>python main.py
   duracion  paginas  acciones  valor  clase
0       7.0        2         4      8      2
1      21.0        2         6      6      2
2      57.0        2         4      4      2
3     101.0        3         6     12      2
4     109.0        2         6     12      2
         duracion     paginas    acciones       valor       clase
count  170.000000  170.000000  170.000000  170.000000  170.000000
mean   111.075729    2.041176    8.723529   32.676471    0.752941
std    202.453200    1.500911    9.136054   44.751993    0.841327
min      1.000000    1.000000    1.000000    1.000000    0.000000
25%     11.000000    1.000000    3.000000    8.000000    0.000000
50%     13.000000    2.000000    6.000000   20.000000    0.000000
75%    108.000000    2.000000   10.000000   36.000000    2.000000
max    898.000000    9.000000   63.000000  378.000000    2.000000
clase
0    86
1    40
2    44
dtype: int64
Dimensión de X: (170, 4)
Primeras predicciones: [2 2 2 2 2]
Precisión en todo el conjunto: 0.7823529411764706
Logistic Regression: 0.714835 (0.099402)
Precisión en validación: 0.8235294117647058
Matriz de Confusión:
 [[16  1  1]
 [ 4  4  0]
 [ 0  0  8]]
Reporte de Clasificación:
              precision    recall  f1-score   support

           0       0.80      0.89      0.84        18
           1       0.80      0.50      0.62         8
           2       0.89      1.00      0.94         8

    accuracy                           0.82        34
   macro avg       0.83      0.80      0.80        34
weighted avg       0.82      0.82      0.81        34

Clasificación para el nuevo dato: [2]
```