Actividad 11: Programando Regresión Logística en Python

Gerardo Enrique Torres Flores 2064063

23 de marzo de 2025

1. Introducción

Es una técnica estadística que permite **analizar relaciones entre variables y predecir el valor de una variable a partir de otra**. A partir de un conjunto de datos de entrada (características), nuestra salida será discreta (y no continua), por eso utilizamos Regresión Logística (y no Regresión Lineal). La Regresión Logística es un *Algoritmo Supervisado* que se utiliza para *clasificación*. Vamos a clasificar problemas con dos posibles estados "SI/NO" (binario) o un número finito de "etiquetas" o "clases" (múltiple).

Algunos ejemplos de Regresión Logística son:

- Clasificar si el correo que llega es *Spam* o *No es Spam*.
- Dados unos resultados clínicos de un tumor clasificar en "Benigno" o "Maligno".
- Clasificar el texto de un artículo en categorías como *Entretenimiento, Deportes, Política o Ciencia*.
- A partir del historial bancario, decidir conceder o no un crédito.

En esta actividad, emplearemos un **modelo de Regresión Logística** para predecir la clase de usuarios (por ejemplo, su sistema operativo favorito) con base en variables como duración de uso, páginas visitadas, acciones y valor asignado. Asignaremos los siguientes valores a las etiquetas: **0- Windows 1- Macintosh 2-Linux**

2. Metodología

Para realizar esta actividad se siguieron los siguientes pasos:

1. Importación de librerías y configuración

Se importaron las librerías necesarias para el análisis y la clasificación, además de configurar las herramientas de visualización:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.multiclass import OneVsRestClassifier
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sb
```

2. Lectura y exploración de datos

Se cargó el archivo CSV `usuarios_win_mac_lin.csv` y se exploró la estructura de los datos. Además, se observó cómo se distribuyen las clases:

```
dataframe = pd.read_csv(r"usuarios_win_mac_lin.csv")
dataframe.head()
dataframe.describe()
print(dataframe.groupby('clase').size())
```

3. Visualización de la distribución de datos

Se generaron histogramas de las variables numéricas para entender su comportamiento y se usó `pairplot` para ver cómo se agrupan los datos según la clase:

```
dataframe.drop(columns=['clase']).hist()
plt.show()

sb.pairplot(dataframe.dropna(), hue='clase', height=4, vars=[
    "duracion", "paginas", "acciones", "valor"], kind='reg')
plt.show()
```

4. Preparación de datos y creación del modelo

Se separaron las variables predictoras (X) y la variable objetivo (y). Luego, se definió el modelo de Regresión Logística y se entrenó con los datos completos:

```
model = OneVsRestClassifier(LogisticRegression(solver='liblinear'))
model.fit(X, y)

predictions = model.predict(X)
print(predictions[0:5])
model.score(X,y)
```

5. Validación del modelo

Para evaluar la capacidad de generalización del modelo, se dividieron los datos en entrenamiento y validación, y se realizó una validación cruzada:

```
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, y, test_size=validation_size, random_state=seed)

name='Logistic Regression'
kfold = model_selection.KFold(n_splits=10, shuffle=True, random_state=seed)
cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)

predictions = model.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

6. Predicción con nuevos datos

Finalmente, se creó un nuevo conjunto de características y se utilizó el modelo entrenado para predecir la clase:

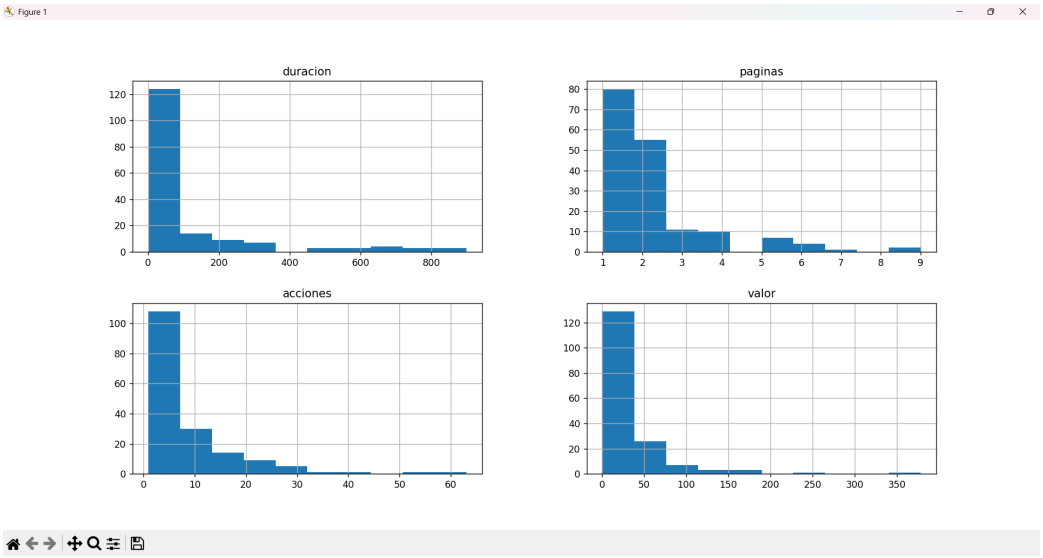
```
X_new = pd.DataFrame([[10, 3, 5, 9]], columns=X.columns)
print(model.predict(X_new))
```

3. Resultados

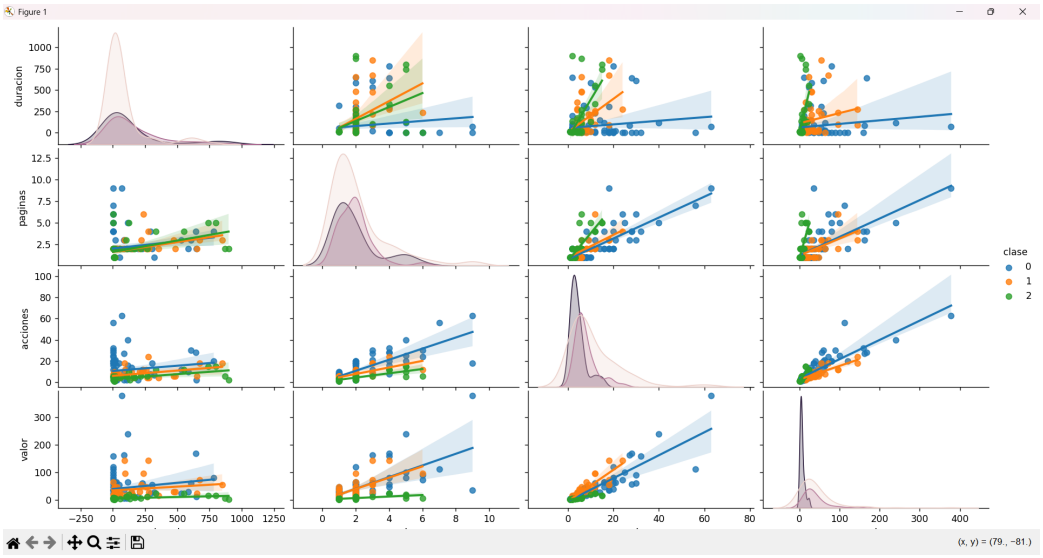
Entre los resultados obtenidos en la actividad se incluyen:

- **Precisión del modelo** (*accuracy*): Se observó la proporción de instancias correctamente clasificadas tanto en la validación cruzada como en la prueba final.
- **Matriz de confusión**: Permite visualizar cuántas instancias de cada clase se clasifican correctamente y cuáles se confunden con otras clases.
- **Reporte de clasificación**: Muestra métricas de evaluación como la precisión (*precision*), exhaustividad (*recall*) y la medida F1 para cada clase.

Visualización de la distribución de datos:



Visualización de concentración lineal de datos:



De manera general, el modelo de Regresión Logística mostró un buen desempeño en la tarea de clasificación de los usuarios según su sistema operativo preferido. A continuación, la **salida de verificación del modelo:**

```
clase
0    86
1    40
2    44
dtype: int64
[2 2 2 2 2]
0.7764705882352941
Logistic Regression: 0.742308 (0.135737)
0.8529411764705882
```

4. Conclusión

A diferencia de las actividades anteriores, se aplicó la **Regresión Logística** para clasificar a los usuarios según su sistema operativo preferido. Se prepararon los datos, se definió el modelo y se evaluó su desempeño a través de la validación cruzada y la matriz de confusión. Los resultados muestran que la Regresión Logística es una técnica efectiva para resolver problemas de clasificación con un conjunto de clases discreto. Este modelo puede adaptarse a distintos escenarios de clasificación, brindando resultados interpretables y eficientes para la toma de decisiones.

```
[[16  0  2]
 [ 3  3  0]
 [ 0  0 10]]
              precision    recall  f1-score   support

           0       0.84      0.89      0.86        18
           1       1.00      0.50      0.67         6
           2       0.83      1.00      0.91        10

    accuracy                           0.85        34
   macro avg       0.89      0.80      0.81        34
weighted avg       0.87      0.85      0.84        34

[2]
```