



## Universidad Autónoma de Nuevo León

## Licenciatura en Ciencias Computacionales

# A10: Regresión Logística en Python

**Materia: Inteligencia Artificial**

**Maestro: Luis Ángel Gutiérrez Rodríguez**

**Rebeca Jaramillo Camarillo**

**Matrícula: 2132988**

**Grupo: 031**

**Fecha: 23 de marzo de 2025**

# 1. Introducción

En estadística, la regresión logística es un tipo de modelo de regresión que sirve para predecir el resultado de una variable categórica. Es decir, la regresión logística se usa para modelar la probabilidad de que una variable categórica tome un determinado valor en función de las variables independientes. El modelo de regresión logística más habitual es la regresión logística binaria, en la cual solo hay dos posibles resultados: ≪fracaso≫ o ≪éxito≫ (distribución de Bernoulli). El ≪fracaso≫ se representa con el valor 0, mientras que el ≪éxito≫ se representa con el valor 1.

Por ejemplo, la probabilidad de que un alumno apruebe un examen en función de las horas que ha dedicado al estudio se puede estudiar mediante un modelo de regresión logística. En este caso, suspender sería el resultado de ≪fracaso≫ y, por otro lado, aprobar sería el resultado de ≪éxito≫.

Otros ejemplos:

- Clasificar si el correo que llega es Spam o No es Spam.
- Dados unos resultados clínicos de un tumor clasificar en “Benigno” o “Maligno”.
- El texto de un artículo a analizar es: Entretenimiento, Deportes, Política ó Ciencia.
- A partir de historial bancario conceder un crédito o no.

En este documento, se presenta el paso a paso de una actividad de regresión logística en Python, el cual utiliza datos de entrada para clasificar si el usuario que visita un sitio web usa como sistema operativo Windows, Macintosh o Linux. La información de entrada son 4 características de una web que utiliza Google Analytics y son:

- Duración de la visita en Segundos
- Cantidad de Páginas Vistas durante la Sesión
- Cantidad de Acciones del usuario (click, scroll, uso de checkbox, sliders,etc)
- Suma del Valor de las acciones (cada acción lleva asociada una valoración de importancia)

Como la salida es discreta, se asignaran los siguientes valores a las etiquetas:

- 0 - Windows
- 1 - Macintosh
- 2 - Linux

# 2. Metodología

A continuación, se muestran los pasos para resolver el ejercicio:

## 2.1. Importar librerías necesarias

```
import pandas as pd
import numpy as np
from sklearn import linear_model
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
```

## 2.2. Leer el archivo csv y cargarlo como un dataset de Pandas

El siguiente código se encarga de cargar y leer los datos de entrada. Además, permite visualizar información estadística básica del set de datos, así como el numero de registros de cada sistema operativo.

```
dataframe = pd.read_csv(r"usuarios_win_mac_lin.csv")
print(dataframe.head())

print(dataframe.groupby('clase').size())
```

## 2.3. Visualización de datos

Visualizacio gráfica en fromato de histograma de los cuatro Features de entrada.

```
dataframe.drop('clase', axis=1).hist()
plt.show()

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

## 2.4. Crear y entrenar el Modelo de Regresión Logística

Se cargan las variables de las 4 columnas de entrada en X excluyendo la columna "clase" con el métododrop(). En cambio, se agrega la columna"clase" en la variable y. Se ejecuta X.shape para comprobar la dimensión de la matriz con datos de entrada de 170 registros por 4 columnas. Luego se crea el modelo y se ajusta (fit) al conjunto de entradas X y salidas 'y'.

```
X = dataframe.drop('clase', axis=1)
y = dataframe['clase']
print(X.shape)

model = linear_model.LogisticRegression(max_iter=1000)
model.fit(X,y)
```

Una vez compilado el modelo, se clasifica el conjunto de entradas X utilizando el método "predict(X)" y se revisa que las salidas coincidan con las salidas reales del archivo csv. Luego se confirma cuan bueno fue el modelo utilizando model.score(), el cual devuelve la precisión media de las predicciones.

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

## 2.5. Validación del modelo

Se dividen los datos de entrada de forma aleatoria (mezclados) utilizando 80 % de registros para entrenamiento y 20 % para validar.

```
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)
```

Después, se vuelve a compilar el modelo de Regresión Logística, pero esta vez sólo con 80 % de los datos de entrada.

```
name = "LogisticRegression"
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)
```

### 2.6. Reporte de resultados

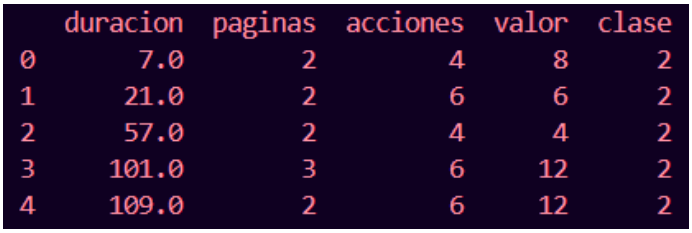
```
print("\nMatriz de confusion:")
print(confusion_matrix(Y_validation, predictions))
print("\nReporte de clasificacion:")
print(classification_report(Y_validation, predictions))

# Clasificacion de nuevos valores
X_new = pd.DataFrame({'duracion': [10], 'paginas': [3], 'acciones': [5], '
    valor': [9]})
print("\nPrediccion para nuevos datos:", model.predict(X_new))
```

# 3. Resultados

A continuación, se presentan los resultados del análisis de Regresión Lineal Múltiple

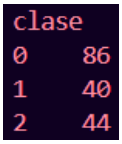
Primeros 5 valores del dataframe:



| | 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 |

Figura 1: Paso 2

Distribución de clase: 86 usuarios de Windows, 40 usuarios Mac y 44 de Linux



| clase | |
|---|---|
| 0 | 86 |
| 1 | 40 |
| 2 | 44 |

Figura 2: Paso 2

Histogramas de los 4 datos de entrada: “duración”, “páginas”, ”acciones” y “valor”

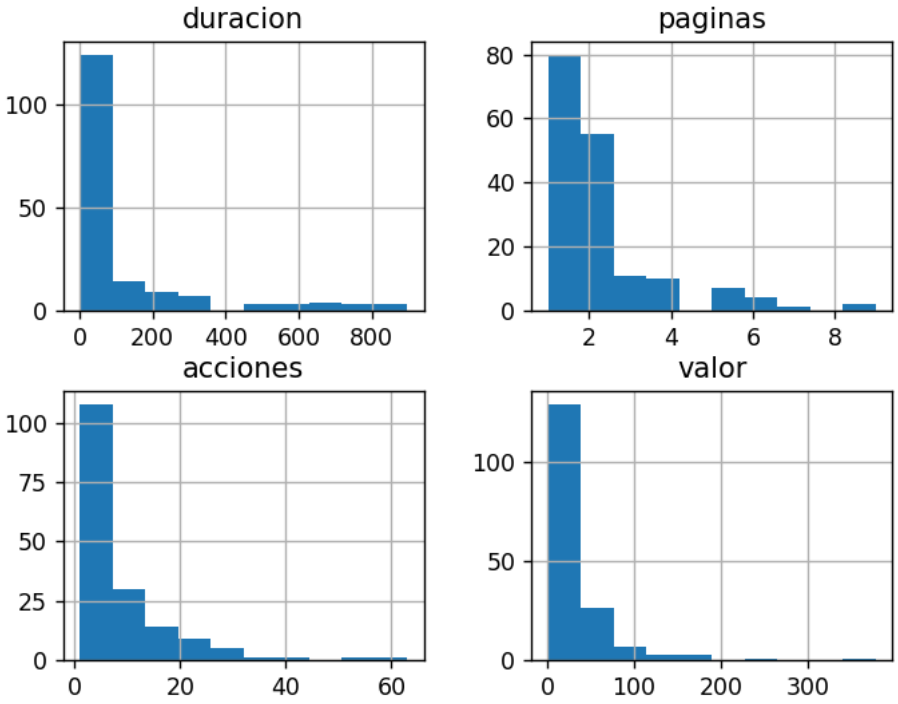



Figura 3: Paso 2

Interrelación de las entradas de a pares, para ver cómo se concentran linealmente las salidas de usuarios por colores: Sistema Operativo Windows en azul, Mac en en verde y Linux en naranja

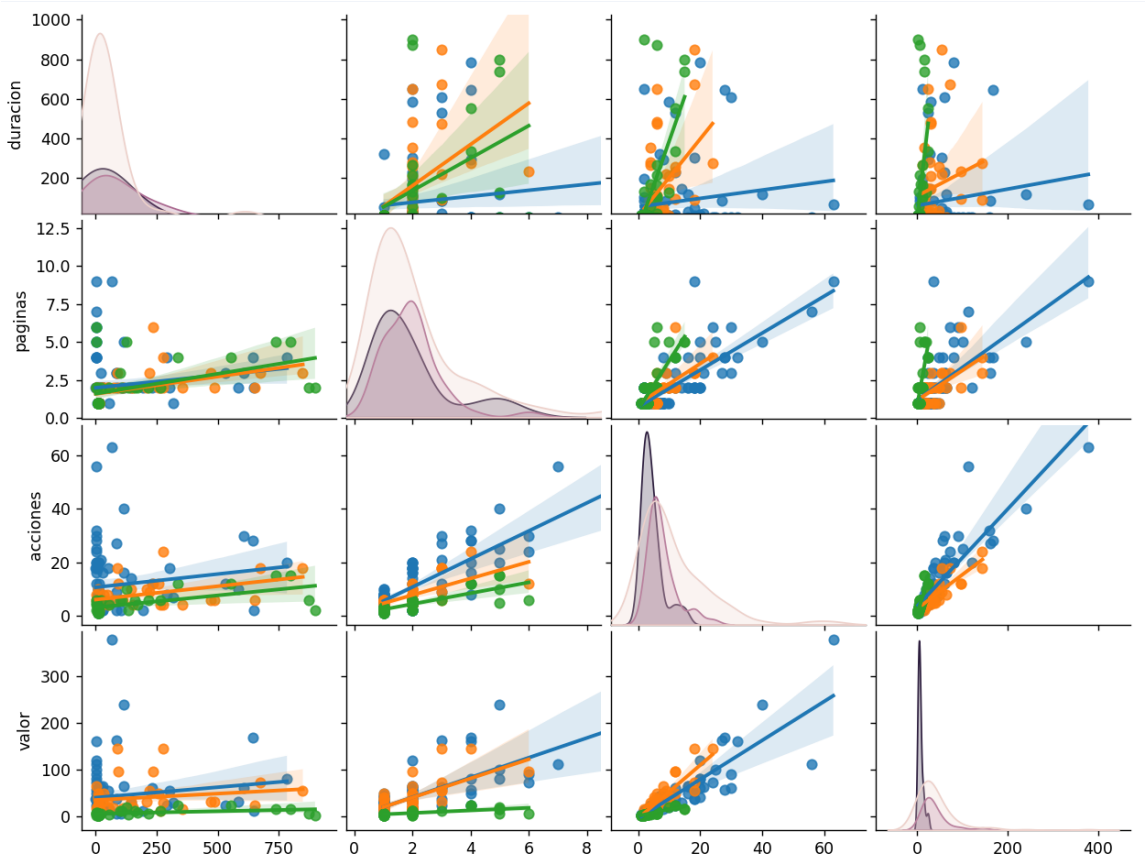



Figura 4: Paso 2

La precisión media de las predicciones del modelo es del 77 %.
Despues de volver a compilar el modelo de Regresión Logística, pero esta vez sólo con 80 % de los datos de entrada, el nuevo scorig es de 72 %.
Luego se hacen las predicciones (clasificación) y el acierto es del 85 %

```
0.7764705882352941
LogisticRegression: 0.720330 (0.151123)
0.8529411764705882
```

Figura 5: Precisión media de las predicciones

Finalmente, se observan los resultados del modelo, como la "matriz de confusión" donde muestra cuantos resultados equivocados tuvo de cada clase (los que no están en la diagonal). En este caso, predijo 3 usuarios que eran Mac como usurarios de Windows, y 2 usuarios Linux que eran Windows.

```
[[16  0  2]
 [ 3  3  0]
 [ 0  0 10]]
```

Figura 6: "Matriz de confusión"

También se observa el reporte de clasificación con el conjunto de Validación. En este caso, se observa que se utilizaron como "soporte" 18 registros windows, 6 de mac y 10 de Linux (total de 34 registros). Se puede ver la precisión con que se acertaron cada una de las clases, por ejemplo, de Macintosh tuvo 3 aciertos y 3 fallos (0.5 recall).

| | 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 |

Figura 7: Reporte de clasificación

# 4. Conclusión

Aunque este ejercicio de regresión logística mostró resultados satisfactorios (matriz de confusión, métricas de clasificación y capacidad predictiva para nuevos datos), es importante destacar que se trabajó con un conjunto de datos limitado con fines de aprendizaje. En la práctica, para obtener predicciones más precisas, es necesario utilizar un mayor número de datos (en este caso fueron 170).