

Inteligencia Artificial
Act10: Regresión Logística

Arturo Garza Rodríguez

March 2025

1. Introducción

¿Qué es una regresión logística?

La regresión logística es un modelo estadístico utilizado para predecir la probabilidad de que ocurra un evento binario, es decir, con dos posibles resultados, como “sí” o “no”. Se basa en la relación entre una variable dependiente categórica y varias variables independientes, que pueden ser tanto categóricas como cuantitativas. Este modelo estima la probabilidad de un evento mediante una función logística, y se ajusta utilizando el método de máxima verosimilitud a través de iteraciones sucesivas.

2. Metodología

2.1. Requerimientos

2.1.1. Jupyter notebook

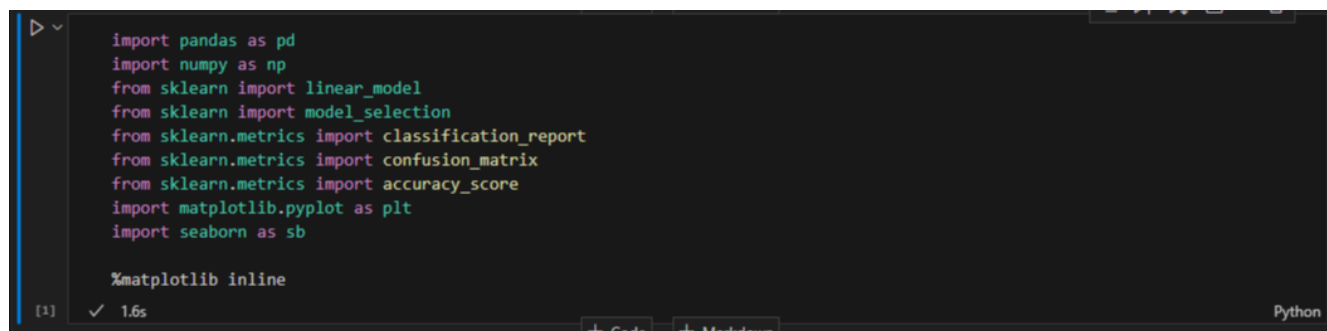
Igual que en el caso de la regresión lineal, estaremos codificando en una Jupyter Notebook, lo que nos permitirá ir probando parte por parte y viendo los resultados al momento.

2.1.2. Descargar dataset

En el libro 'Aprende Machine Learning' de Juan Ignacio Bagnato se nos proporciona un dataset de nombre 'usuarios_win_mac_lin.csv' que descargamos y utilizamos en este ejercicio.

2.2. Desarrollo de código

2.2.1. Imports



```
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

%matplotlib inline
```

[1] ✓ 1.6s Python

Figura 1: Se hace uso de herramientas similares en el caso de la regresión lineal, con la diferencia de que se tienen diferentes criterios de optimalidad.

2.2.2. Definición de dataset

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

[3] ✓ 0.0s Python

| | 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 2: Se lee el dataset usando la librería pandas y se imprimen los primeros 5 registros, para ver el formato del dataset, a través de `data.head()`

```
dataframe.describe()
```

[4] ✓ 0.0s Python

| | 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 |

Figura 3: `data.describe()` es útil para obtener una visión rápida de la distribución y características de los datos en el DataFrame.

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

[5] ✓ 0.0s Python

```
clase
0    86
1    40
2    44
dtype: int64
```

Figura 4: A través de esta sentencia encontramos la cantidad de registros asociados a cada clase.

2.2.3. Análisis gráfico

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

[6] ✓ 0.2s Python

Figura 5: Se genera un histograma para cada una de las variables con excepción de 'clase'.

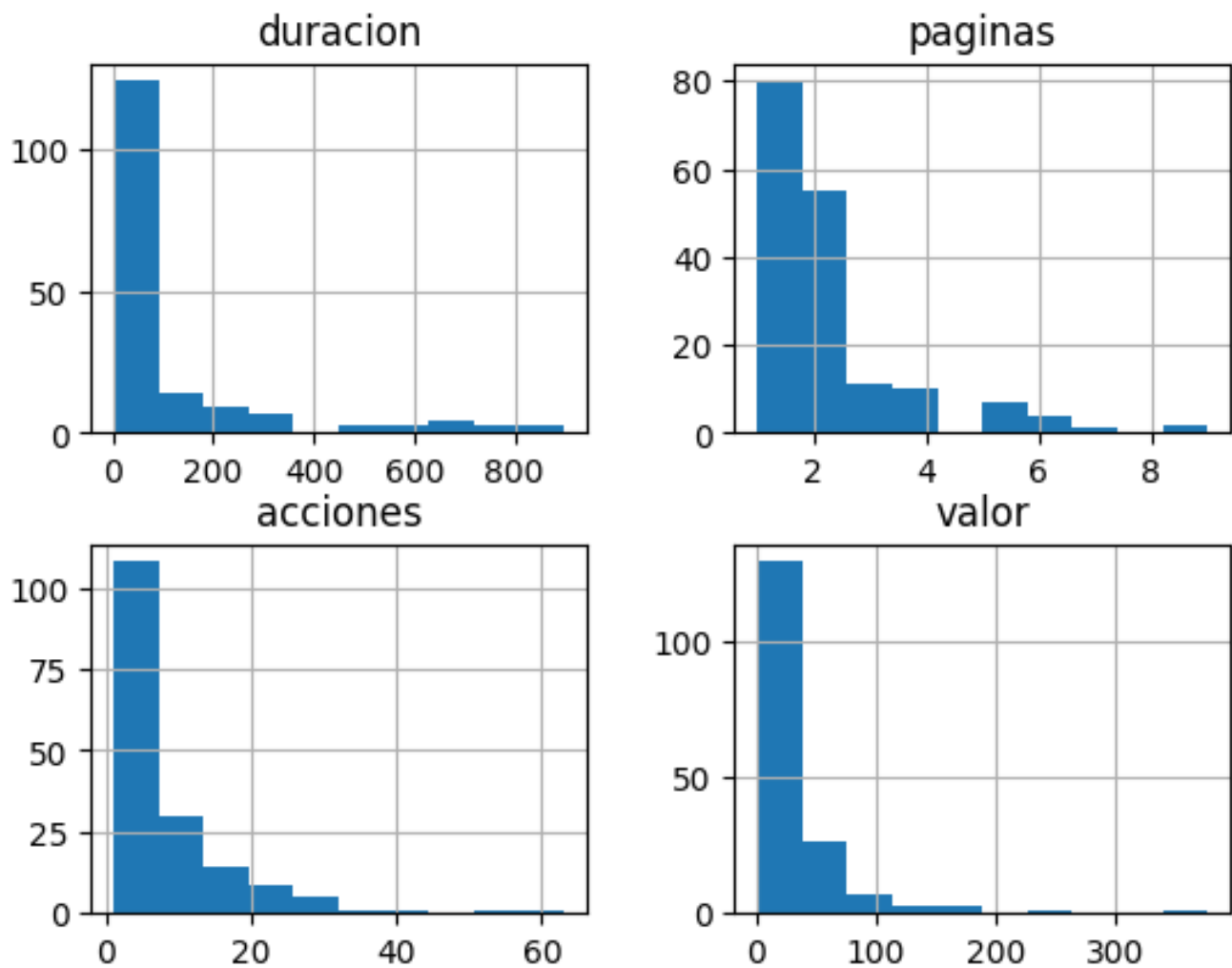


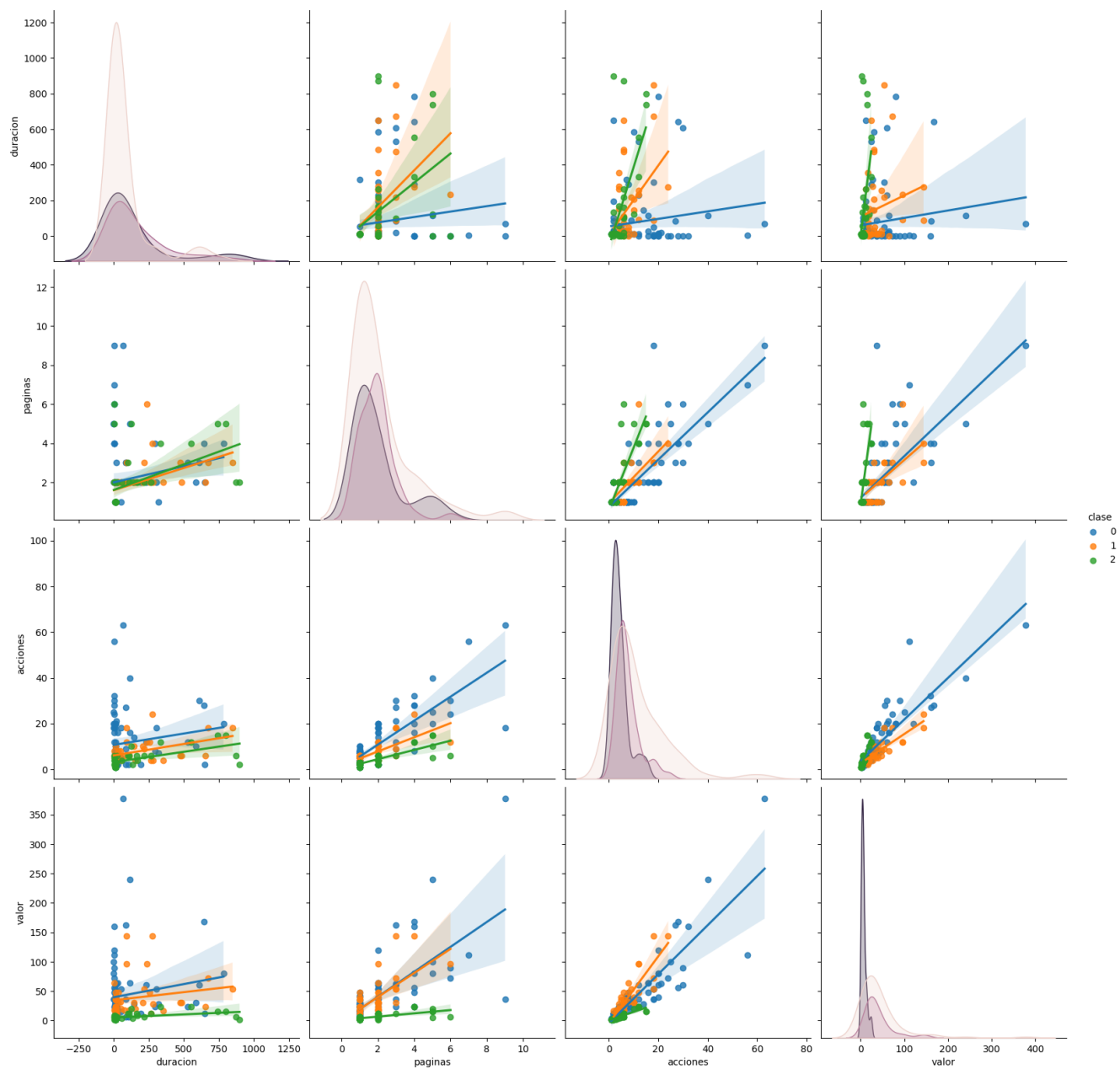
Figura 6: Podemos ver gráficamente entre qué valores e comprenden sus mínimos y máximos y en qué intervalos concentran la mayor densidad de registros.

```

> sb.pairplot(dataframe.dropna(), hue='clase', height=4, vars=["duracion", "paginas", "acciones", "valor"], kind='reg')
[7] ✓ 3.8s Python
... <seaborn.axisgrid.PairGrid at 0x284d1714dd0>

```

Figura 7: El código genera una matriz de gráficos de dispersión con una línea de regresión ajustada, visualizando las relaciones entre las variables duración, páginas, acciones y valor, y cloreando los puntos según la columna clase, después de eliminar las filas con valores nulos.



2.2.4. Modelo regresor

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

✓ 0.0s Python

(170, 4)

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

✓ 0.0s Python

Figura 8: Declaramos los dos vectores que utilizaremos en la regresión logística como X y y , y encontramos que las dimensiones son de 170 x 4.

Definimos el modelo de regresión logística a través de `linear_model.LogisticRegression()` y lo entrenamos con los valores de X y y .

```
predictions = model.predict(X)
print(predictions[:5])
```

[10] ✓ 0.0s Python

... [2 2 2 2 2]

```
model.score(X, y)
```

[11] ✓ 0.0s Python

... 0.7823529411764706

Figura 9: Después de entrenar al modelo, encontramos que las predicciones para los primeros 5 registros de X son '2', lo que nos dice que dados los parámetros, son usuarios de Linux.

```
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
)
```

[12] ✓ 0.0s Python

+ Code + Markdown

```
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)
```

[14] ✓ 0.1s Python

3. Resultados

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

[15] ✓ 0.0s Python
... 0.8529411764705882

print(confusion_matrix(Y_validation, predictions))

[17] ✓ 0.0s Python
... [[16  0  2]
     [ 3  3  0]
     [ 0  0 10]]
```

Figura 10: El código realiza predicciones con el modelo sobre los datos de validación, luego imprime la exactitud y la matriz de confusión comparando las predicciones con los valores reales.

```
print(classification_report(Y_validation, predictions))

[16] ✓ 0.0s Python
...
      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      0.85      0.85         34
  macro avg       0.89      0.80      0.81         34
 weighted avg       0.87      0.85      0.84         34
```

Figura 11: El código imprime un informe de clasificación que muestra métricas como la precisión, la recuperación y la puntuación F1, comparando las predicciones con los valores reales en los datos de validación.

```
X_new = pd.DataFrame({'duracion': [10], 'paginas': [3], 'acciones': [5], 'valor': [9]})
model.predict(X_new)

[18] ✓ 0.0s Python
... c:\Users\artur\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\validation.py:2732: UserWarning: X has fea
warnings.warn(
...
array([2])
```

Figura 12: El código crea un nuevo DataFrame con un conjunto de valores para las características duracion, paginas, acciones y valor, y luego utiliza el modelo previamente entrenado para hacer una predicción sobre esos nuevos datos.

4. Conclusión

En conclusión, la regresión logística es una herramienta estadística poderosa para modelar y predecir la probabilidad de eventos binarios en función de variables independientes. A lo largo del trabajo, se destacó su capacidad para analizar relaciones entre variables y clasificar datos en categorías, utilizando el método de máxima verosimilitud para estimar los parámetros del modelo. Además, la evaluación del modelo con métricas como la precisión, la matriz de confusión y el informe de clasificación permite medir su desempeño y efectividad. En general, la regresión logística es fundamental para problemas de clasificación, brindando interpretaciones claras y aplicables en diversas áreas como la medicina, marketing y análisis de riesgos.

5. Referencias

- Material de clase
- <https://aws.amazon.com/es/what-is/logistic-regression/>
- <https://www.ibm.com/mx-es/topics/logistic-regression>
- <https://www.sciencedirect.com/science/article/abs/pii/S1138359323001661>