Regresión Logística en Python

31 de Marzo de 2025

1 Introducción

La regresión logística es un método estadístico fundamental que modela la probabilidad de que una observación pertenezca a una categoría específica. A diferencia de la regresión lineal, la variable dependiente es categórica (binaria o multiclase). En la regresión logística estándar, la variable dependiente solo puede tomar los valores 0 o 1, pero también existe la regresión logística multinomial, que predice una probabilidad para todos los posibles valores de la variable dependiente.

El modelo se basa en la función logística (sigmoide):

$$p(y = k) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{1}$$

donde:

- p es la probabilidad de que y tome un valor k.
- y es la variable dependiente.
- k es un valor binario: 0 o 1.
- e es el número de Euler: 2.7182818284...
- x es la variable independiente.
- β_0 es el intercepto.
- β_1 es la pendiente.

2 Metodología

Para realizar el ejercicio de regresión logística, se siguieron los siguientes pasos:

```
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
```

```
[2]:
```

```
#cargamos el dataframe
dataframe = pd.read_csv("usuarios_win_mac_lin.csv")
dataframe.head()
```

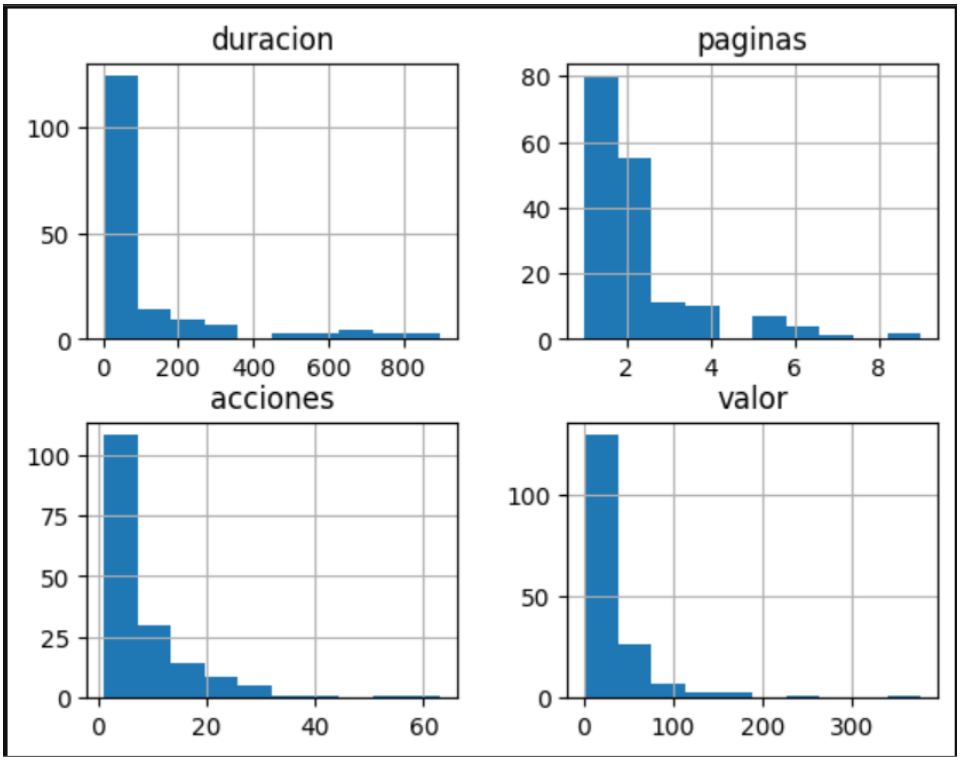
Figure 1: Importación de bibliotecas y preparación de datos.

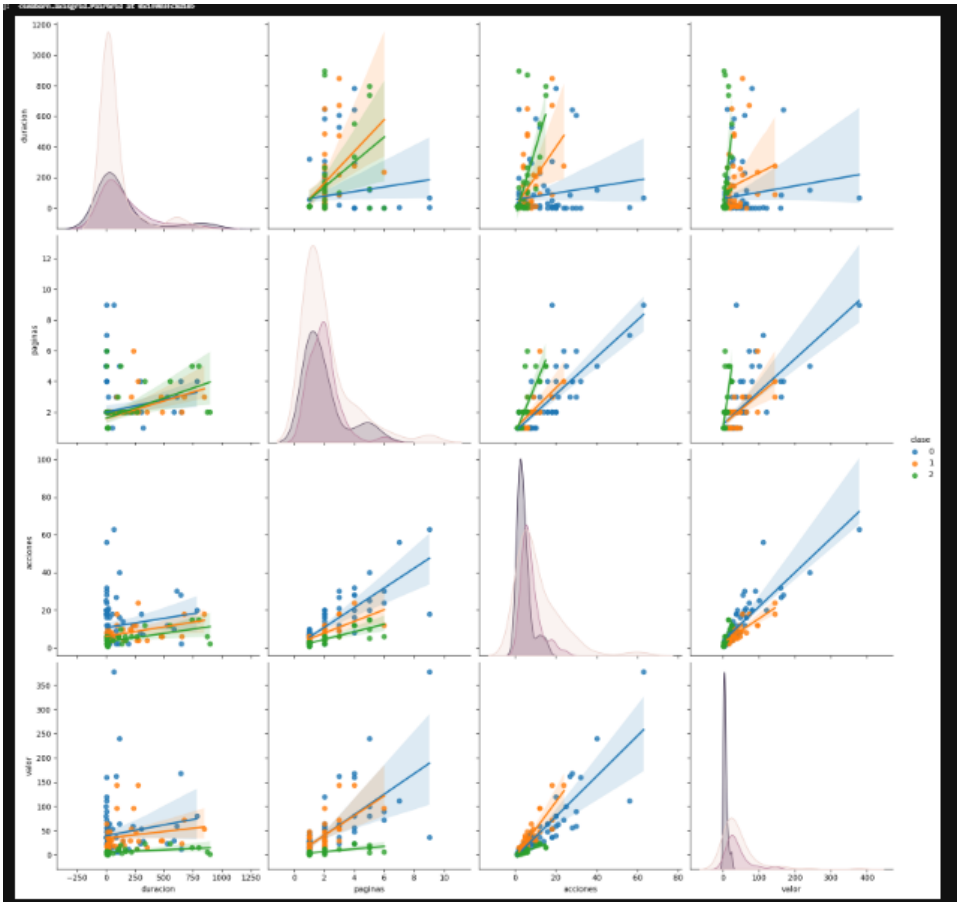
2.1 Visualización General de los Datos de Entrada

La columna del dataset de clase tiene tres posibles valores: 0, 1 y 2, que corresponden al sistema operativo del usuario. Esta variable será nuestra variable dependiente, y el objetivo es predecir su valor en función de las variables independientes.

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

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





2.2 Creación y Ajuste del Modelo

Para este modelo, la variable dependiente es la columna de clase y las variables independientes son las otras cuatro columnas numéricas del dataset. Se establecen los valores de X y Y y se ajusta el modelo de acuerdo con estos valores.

```
#Modelo de regresion logica
X = np.array(dataframe.drop(['clase'], axis=1))
y = np.array(dataframe['clase'])
X.shape
```

```
[14]:

(170, 4)

[16]:
```

```
#Creamos el modelo de regresion logica
model = linear_model.LogisticRegression()
model.fit(X, y)

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

```
[2 2 2 2 2]
```

Figure 2: Creación del modelo.

```
#Precicion del modelo
model.score(X, y)

[17]:

0.7823529411764706
```

Figure 3: Ajuste del modelo.

2.3 Validación del Modelo

En esta parte se intenta confirmar la precisión del modelo creando un conjunto de datos de validación. Posteriormente, se utiliza el método de validación cruzada para dividir los datos de entrenamiento en n partes y evaluar cada una de ellas con los datos de validación.

```
validation_size = .2
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 = 'Regresion Logistica'
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)

Regresion Logistica: 0.704945 (0.145247)
```

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

0.8529411764705882
```

Figure 4: Validación del modelo.

3 Resultados

Para verificar los resultados, se imprime la matriz de confusión y el reporte de clasificación del conjunto de validación Y y las predicciones.

```
print(confusion_matrix(Y_validation, predictions))

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

Figure 5: Matriz de confusión.

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

              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
```

Figure 6: Resultados.

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

```
C:\Python312\Lib\site-packages\sklearn\utils\validation.py:2732: User
Warning: X has feature names, but LogisticRegression was fitted witho
ut feature names
  warnings.warn(
```

```
[25]:

array([2])
```

Figure 7: Prueba del modelo.

4 Conclusión

Los modelos de regresión logística son herramientas estadísticas utilizadas para calcular la probabilidad de una variable dependiente binaria en función de un conjunto de variables independientes, siempre que exista una relación logística entre ellas.

Existen dos tipos principales de regresión logística:

- Regresión logística estándar: la variable dependiente solo puede tomar los valores 0 o 1.
- Regresión logística multinomial: la variable dependiente puede tomar más valores.

En este caso, se creó un modelo logístico multinomial que intenta mostrar la relación entre la duración, páginas, acciones y valor de los usuarios y la clase de sistema operativo que usan. Evaluando el modelo mediante su puntaje y validándolo con técnicas avanzadas para prevenir el *overfitting*, se observa que el modelo tiene buena precisión para predecir el tipo de sistema operativo del usuario.