

Regresión Logística

Valeria Ybarra López

Matrícula: 2047880

1. Introducción

La regresión logística es una técnica de análisis de datos que utiliza las matemáticas para identificar relaciones entre dos variables. Estas relaciones se utilizan para predecir el valor de una variable a partir de otra. Normalmente, la predicción tiene un número finito de resultados, como “sí” o “no”.

Esta técnica se aplica cuando los datos de entrada (o características) generan salidas discretas, no continuas, lo que la diferencia de la regresión lineal. Se trata de un Algoritmo Supervisado empleado para la calificación. Permite resolver problemas donde las posibles soluciones son binarias (“sí” o “no”) o pertenecen a un número finito de “clases” o “etiquetas”.

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”.
- El texto de un artículo a analizar es: Entretenimiento, Deportes, Política ó Ciencia.
- A partir de historial bancario conceder un crédito o no.

2. Metodología

Para la realización de esta actividad, se siguieron las instrucciones proporcionadas en la página 39 del libro “Aprenda Machine Learning”.

2.1. Creamos la carpeta Regresión Logística

Se creó una carpeta con el nombre de “Regresión Logística” en donde se guardó el archivo .csv (que contiene los datos de entrada) proporcionado por el libro para poder realizar el código en python, en esa misma carpeta se creó un archivo .py para realizar la actividad.

2.2. Archivo CSV

El archivo CSV nos proporciona con 170 registros de muestra para clasificar si el sistema operativo de un usuario en un sitio web es Windows, Macintosh o Linux (0, 1 y 2 respectivamente). Las características de entrada son:

- Duración de la visita (segundos)
- Páginas vistas en la sesión
- Acciones del usuario (clicks, scrolls, etc.)
- Valor de las acciones (según su importancia)

2.3. Código

Comenzamos el código importando las bibliotecas 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
```

Después leemos el archivo csv y lo asignamos usando Pandas a la variable "dataframe".

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

Usando el método ".head()" imprimiremos los primeros 5 registros guardados en el archivo csv.

```
print(dataframe.head())
```

Continuamos llamando al método ".describe()", el cual nos da información estadística (Media, desvío estándar, valores mínimo y máximo) de nuestro set de datos.

```
print(dataframe.describe())
```

Analizaremos la cantidad de resultados que tenemos de cada tipo haciendo uso de la función groupby:

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

Visualización de Datos

Realizaremos unas visualizaciones que nos podrán ayudar a comprender mejor las características de la información con la que estamos trabajando, al igual que la correlación que hay entre los datos. Para esto primero visualizaremos en formato de historial los cuatro Features de entrada con nombres "duración", "páginas", "acciones" y "valor".

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

También podemos relacionaremos las entradas de a pares, para ver como se concentran linealmente las salidas de usuarios por colores: Sistema Operativo Windows en azul, Macintosh en verde y Linux en rojo.

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

Creamos el Modelo de Regresión Logística

Cargamos las variables de las 4 columnas de entrada en X excluyendo con el método drop(), la columna "clase", esta columna la agregamos en la variable y . Ejecutamos $X.shape$ para comprobar la dimensión de la matriz.

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

Una vez realizado este cambio, creamos nuestro modelo y con el método ".fit()", lo ajustamos al conjunto de entradas X y salidas y .

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

Después de compilar el modelo, lo utilizamos para clasificar todo el conjunto de datos de entrada X mediante el método ".predict(X)". Luego, examinamos algunas de las predicciones generadas y comprobamos que coincidan con las salidas reales del archivo CSV.

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

Después confirmamos que tan preciso es nuestro modelo utilizando "model.score()", esto nos deberá devolver la precisión de las predicciones.

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

Validación del modelo

Una buena práctica en Machine Learning es dividir los datos (de forma aleatoria) en dos conjuntos: uno para entrenamiento (80 %) y otro para validación (20 %), esto asegura que el modelo no utilice el set de validación durante el entrenamiento. Así prevenimos problemas como la sobre-generalización del algoritmo.

```
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)
```

Compilamos de nuevo el modelo de Regresión Logística pero esta vez sólo con el 80 % de los datos y calculamos el nuevo scoring.

```
name='Logistic Regression'
kfold = model_selection.KFold(n_splits=10, random_state=None)
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)
```

Realizamos predicciones de clasificación usando el "cross validation set", el conjunto apartado previamente. El código muestra cómo generar las predicciones y calcular la precisión.

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

Reporte de Resultados del Modelo

Generamos una matriz de confusión que compara las predicciones del modelo con los valores reales, ayudando a evaluar el rendimiento del modelo y detectar errores de clasificación

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

La función "classification_report()" genera un informe detallado del rendimiento del modelo de clasificación. Este informe incluye métricas clave para evaluar cómo el modelo clasifica diferentes clases.

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

Clasificación de nuevos valores

Por último, inventaremos datos de entrada de navegación de un usuario ficticio:

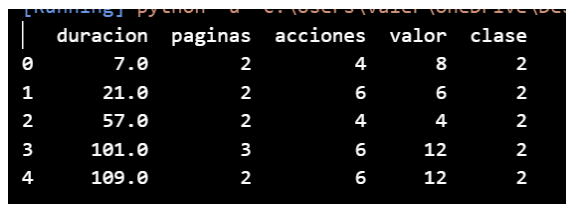
- Tiempo Duración: 10
- Paginas visitadas: 3
- Acciones al navegar: 5
- Valoración: 9

Lo probamos en el modelo:

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

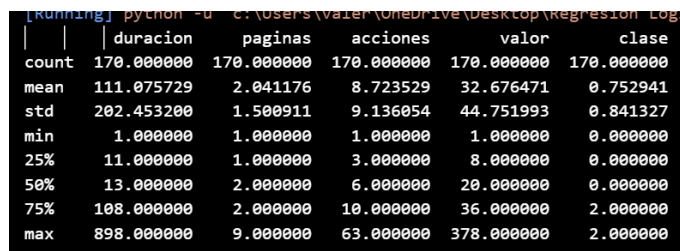
3. Resultados

Visualización de datos:



	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: Primeras 5 filas.



	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 2: Descripción de los datos estadísticos.

```

clase
0      86
1      40
2      44
dtype: int64

```

Figura 3: Función "groupby()". Vemos que tenemos 86 usuarios "Clase 0", es decir Windows, 40 usuarios Mac y 44 de Linux.

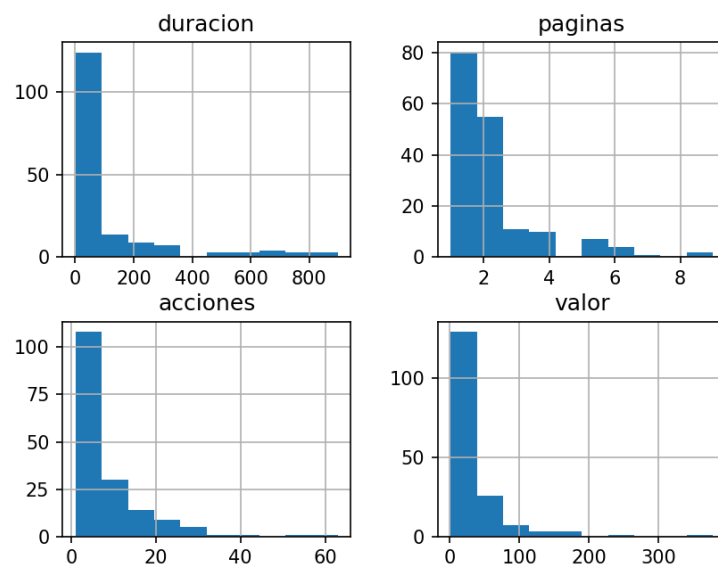


Figura 4: Histogramas de los cuatro Features de entrada: "duración", "páginas", "acciones" y "valor".

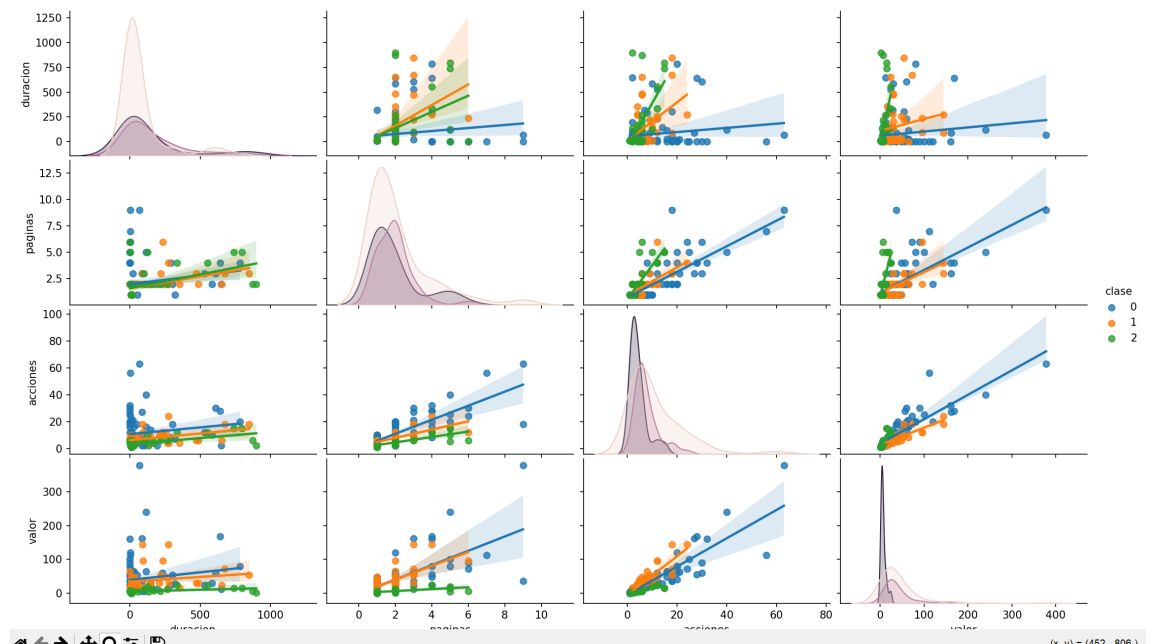


Figura 5: Podemos ver una serie de gráficos donde cada variable es comparada con las demás, con puntos codificados por color según la clase, y líneas de regresión para analizar la tendencia lineal entre las variables.

Modelo de Regresión Logística:

```
(170, 4)
```

Figura 6: Dimensión de la matriz después de excluir la columna “clase” de los datos de entrada y la cambiamos a la variable y.

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

Figura 7: Los primeros cinco valores en el array “predictions”, las cuales representan las clases predichas por el modelo para las primeras cinco filas de (X).

```
0.7764705882352941
```

Figura 8: Resultado de usar el método “model.score()”, para confirmar cuan bueno fue el modelo (precisión de las predicciones).

Validación del modelo:

Logistic Regression: 0.728571 (0.094186)

Figura 9: Resultado de compilar el modelo de Regresión Lógica con 80 % de los datos de entrada.

0.8529411764705882

Figura 10: Resultado de la predicción utilizando nuestro “cross validation set”. Debido a que el tamaño de datos que teníamos era pequeño nuestros aciertos fueron del 85 %.

Reporte de Resultados del Modelo:

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

Figura 11: Resultado de la matriz de confusión, muestra cuantos resultados equivocados tuvo de cada clase (los que no están en la diagonal).

		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 12: Podemos ver que el reporte de clasificación del conjunto de validación muestra 18 registros de Windows, 6 de Mac y 10 de Linux. Por ejemplo, para Macintosh hubo 3 aciertos y 3 fallos (recall de 0.5). El F1-score, que equilibra precisión y recall, tiene un promedio de 84 %, lo cual es bastante positivo.

Clasificación de Nuevos Valores:

[2]

Figura 13: El resultado de inventar los datos de entrada; clasifica un usuario tipo 2, es decir, de Linux.

4. Conclusión

Al realizar esta actividad me quedó claro que, la Regresión Logística es una herramienta clave en el análisis de datos y la clasificación, pues esta nos ayuda a predecir resultados discretos a partir de características específicas. El poder implimentarlo en Python facilita la creación de modelos útiles, como en este ejercicio de clasificación de sistemas operativos según patrones de navegación web. Durante la realización del código pude reforzar mis conocimientos sobre el uso y funcionamiento de los distintos métodos y funciones que se utilizan en la predicción haciendo uso de la técnica de regresión logística.

5. Referencias

- Bagnato, J. (2020). Aprende Machine Learning en Español.
AWS.(2024).¿Qué es la regresión logística?. Recuperado de <https://aws.amazon.com/es/what-is/logistic-regression/>