Regresión Logística

Alain Lobato

31 de Marzo de 2025

1 Introducción

La regresión logística es una técnica de análisis de datos que nos permite encontrar patrones en las relaciones entre variables. A diferencia de la regresión lineal, en este caso, la salida es discreta, es decir, los resultados se limitan a categorías específicas como "sí" o "no". Se considera un algoritmo supervisado de clasificación que se usa en problemas donde las respuestas están dentro de un conjunto finito de opciones.

2 Metodología

Primero, clonamos nuestro repositorio de la materia en nuestra máquina y creamos una carpeta llamada "Logistic Regression". Descargamos el archivo CSV y creamos un archivo llamado `logistic_regression.py` donde desarrollamos nuestro código usando Visual Studio Code.

Iniciamos importando las librerías necesarias e instalando las dependencias en caso de ser necesario.

```
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
```

Figure 1: Importación de librerías necesarias

Ahora leemos el archivo CSV y verificamos su correcta lectura imprimiendo los primeros 5 registros y una tabla con estadísticas generales del dataframe.

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

Figure 2: Lectura del archivo CSV

```
   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
          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
```

Figure 3: Descripción estadística del dataframe

Luego agrupamos los datos por clase y creamos histogramas para visualizar la distribución.

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

Figure 4: Agrupación de datos por clase

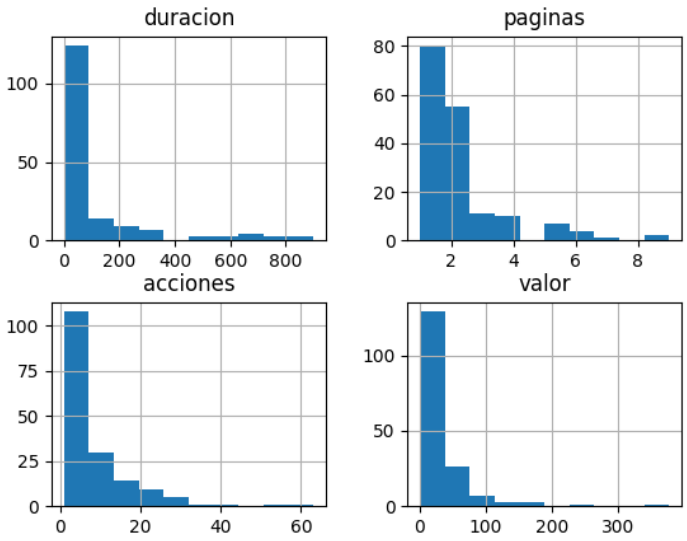

Figure 5: Histogramas de datos

Generamos una gráfica que nos permite visualizar la entrada y salida de usuarios en distintos sistemas operativos.

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

Figure 6: Gráfica de entrada y salida de usuarios en SO

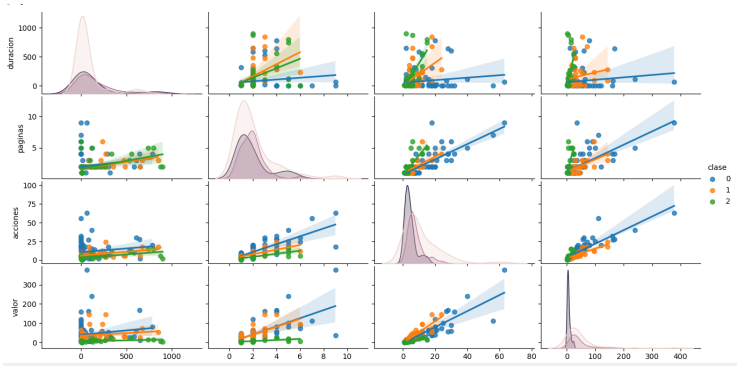

Figure 7: Distribución de usuarios en distintos SO

Ahora creamos nuestro modelo de regresión logística, asignando las variables de entrada y la variable objetivo, para posteriormente entrenarlo.

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

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

Figure 8: Creación y entrenamiento del modelo

Revisamos la precisión del modelo y realizamos una predicción basada en los datos del CSV.

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

Figure 9: Revisión del puntaje del modelo

```
[2 2 2 2 2]
0.7764705882352941
```

Figure 10: Resultados de la predicción

Ahora repetimos el modelo de regresión logística, utilizando el 80% de los datos para entrenamiento.

```
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)
```

Figure 11: Modelo de regresión logística con 80% de datos

Finalmente, realizamos predicciones con nuestro modelo utilizando *cross-validation*.

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

print(confusion_matrix(Y_validation, predictions))

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

Figure 12: Validación cruzada

3 Resultados

Vamos a predecir para un usuario con 10 de duración, 3 páginas, 5 acciones y 9 de valor, obteniendo el siguiente resultado:

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

Figure 13: Predicción del modelo

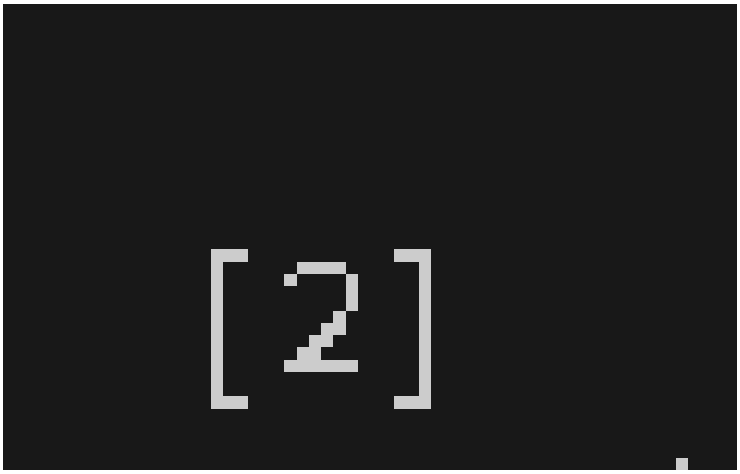

Figure 14: Resultado de la predicción

Como podemos ver, el modelo predice que el usuario pertenece a la categoría 2, lo que indica que es un usuario de Linux.

4 Conclusiones

La regresión logística resulta una herramienta bastante útil en la clasificación de datos, permitiendo predecir categorías específicas a partir de ciertas características. Su implementación en Python con `sklearn` facilita la creación y evaluación de modelos de clasificación. Durante este ejercicio, reforcé mis conocimientos sobre la aplicación práctica de este algoritmo, además de comprender mejor el uso de diferentes métricas para evaluar su rendimiento.

5 Referencias

Materiales de clase (2025). UANL.
Bagnato J. (2019). Aprende Machine Learning. Leanpub.