Regresión Logística

Valeria Rodríguez

23 de Marzo del 2025

1 Introducción

La regresión logística es un modelo estadístico que estima la probabilidad de que ocurra un evento en función de un conjunto dado de variables independientes. Es comúnmente utilizado para clasificaciones y análisis predictivo. Al medir una probabilidad, el valor de la variable resultado debe estar entre 0 y 1, respetando las normas de la probabilidad.

2 Metodología

Para la realización de esta actividad se llevaron a cabo una serie de pasos encaminados a la recreación del problema de regresión logística presentado en el libro *Aprende Machine Learning* del autor Juan Ignacio Bagnato, páginas 47-56.

2.1 Descarga de archivo .csv

Como primer paso, se descargó el archivo 'usuarios_win_mac_lin.csv' proporcionado por el libro de texto para la realización de esta actividad. Para ello se hizo click en el hipervínculo del archivo y éste se descargó automáticamente.

2.2 Creación de carpeta de trabajo

Seguido de lo anterior, se creó una carpeta llamada Regresion Logística, en la cual se copió el archivo csv descargado. Posteriormente se creó dentro de esta misma carpeta un archivo Python llamado reg logistica para la codificación de la actividad.

2.3 Desarrollo del código

Con ayuda del IDE Visual Studio Code se abrió el archivo .py anteriormente creado y se codificó según las especificaciones del libro. Primeramente, se hicieron los imports al archivo.

```
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
```

Posteriormente, se hace lectura del archivo csv y se imprimen con consola las primeras lineas del archivo así como una breve descripción del contenido.

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

Para continuar con el análisis del contenido del archivo, se hace uso de la función *groupby* para obtener la cantidad de eventos por sistema operativo, en este caso llamado *clase* en el archivo csv.

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

Se hace un análisis de las características de los datos del archivo con diversas gráficas de barras que representen las columnas de la información: *duración, páginas, acciones* y *valor*.

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

Así bien, se analizan los datos por sistema operativo, graficándolos por colores: Windows en azul, Macintosh en verde y Linux en rojo.

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

Ahora bien, se crea el modelo de regresión logística, concatenando en X todas las variables a excepción de *clase*, y esta la definimos como Y. Se obtiene la dimensión de la matriz para verificar la cantidad de registros.

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

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

Se crea el modelo y se ajusta con fit() a las entradas X y las salidas Y. Así bien, se clasifican las entradas X con predict y se revisan las salidas.

```
model = linear_model.LogisticRegression()
model.fit(X,y)
predictions = model.predict(X)
print(predictions[:5])
```

Se confirma el modelo con el *model.score* que arroja la precisión de las predicciones.

```
print("Model score: ", model.score(X,y))
```

Una buena práctica del Machine Learning es subdividir los datos en dos sets, uno de entrenamiento y otro de validación. Es por ello que se dividen los datos en 80% entrenamiento y 20% validación. Así bien, se compila de nuevo el modelo con el 80 porciento de datos de entrada y se calcula de nuevo el score del modelo.

```
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)

name='LogisticRegression'
kfold=model_selection.KFold(n_splits=10,shuffle=True)
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)
```

Se realizan las predicciones con el nuevo modelo dividido y se obtiene el porcentaje de acierto del modelo.

```
predictions=model.predict(X_validation)
print("Accuracy score: ", accuracy_score(Y_validation,predictions))
```

Finalmente, se imprime la matriz de confusión que muestra, los resultados equivocados en cada clase. Así bien, se obtiene el reporte de clasificación.

```
print("Matriz de confusión: ")
print(confusion_matrix(Y_validation,predictions))
print("Reporte de clasificación")
```

print(classification_report(Y_validation, predictions))

Se hace un ejercicio con datos de entrada predefinidos y se prueba en el modelo para predecir la clase del usuario.

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

X_new = pd.DataFrame('duracion': [10], 'paginas': [3], 'acciones': [5], 'valor': [9])
print("Clase estimada: ", model.predict(X_new))

3 Resultados

A continuación se listan los resultados arrojados por cada sección de código ejecutado.

	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

Figure 1: Primeras 5 filas del archivo csv.

	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 2: Descripción del archivo csv.

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

Figure 3: Clasificación por sistema operativo.

Se puede observar que se obtuvieron 86 usuarios Windows, 40 usuarios Mac y 44 Linux.

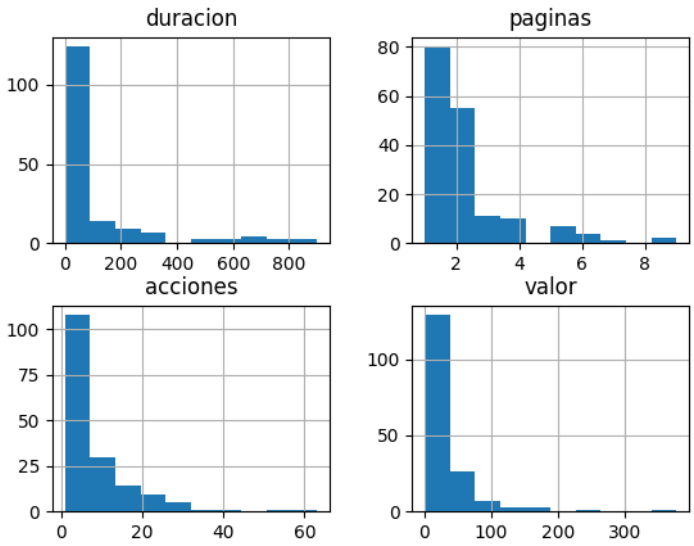

Figure 4: Características de los usuarios.

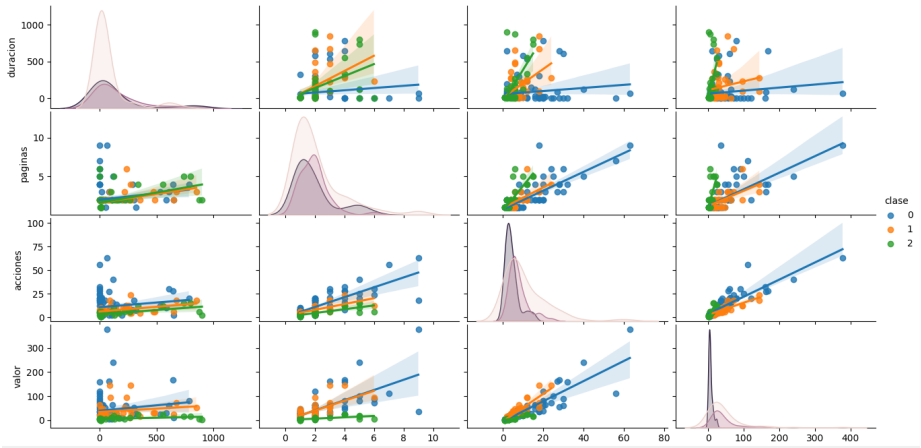

Figure 5: Salidas de usuarios por sistema operativo.

```
(170, 4)
```

Figure 6: Dimensión de la matriz de datos.

Se obtuvo una matriz con 170 registros por 4 columnas.

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

Figure 7: Predicciones del modelo para la variable y.

El modelo predice que los usuarios serán de clase 2, es decir, Linux, lo que coincide con el archivo de datos.

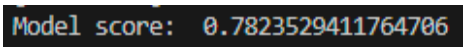

Figure 8: Precisión media de las predicciones del modelo.

La precisión del modelo fue de 78%.

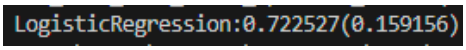

Figure 9: Score del modelo subdividido.

Se obtuvo un Score de 72% en el modelo subdividido.

Figure 10: Precisión media de las predicciones del modelo subdivido

La precisión del modelo subdividido fue de 85%.

```
Matriz de confusión:
[[16  0  2]
 [ 3  3  0]
 [ 0  0 10]]
```

Figure 11: Matriz de confusión del modelo

Se observa que se estimaron 3 usuarios que eran Mac como usuarios Windows y 2 usuarios Linux que eran en realidad Windows.

```
Reporte de clasificación
              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 12: Reporte de clasificación del modelo

Se puede observar 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). La valoración f1 fue de 0.84, un buen valor para el modelo.

```
Clase estimada:  [2]
```

Figure 13: Clase estimada para usuario dado valores específicos

Con el modelo subdividido, se estima que un usuario con las características dadas será de Linux.

4 Conclusión

Considero que la regresión logística predijo considerablemente bien a los usuarios de una página según sus características. Además, el uso de buenas prácticas de Machine Learning como la subdivisión de los datos en entrenamiento y estimación ayuda significativamente a obtener estimaciones más realistas y poder revisar las mismas. Durante la ejecución de la codificación fue necesario hacer ajustes de sintaxis en diversos comandos del código proporcionado por el libro debido a diferencias en las versiones de Python utilizadas, sin embargo, estas se resolvieron fácilmente.

References

IBM. (s.f.). ¿Qué es la regresión logística? IBM.
Material de clase. (2025). UANL