Regresión Logistica en Python

[Carlos Oswaldo Gonzalez Garza]

March 29, 2025

1 Introducción

La regresión logística es una técnica de análisis de datos que utiliza las matemáticas para encontrar las relaciones entre dos factores de datos. Luego, utiliza esta relación para predecir el valor de uno de esos factores basándose en el otro. Normalmente, la predicción tiene un número finito de resultados, como un sí o un no.

2 Metodología

Se siguieron los siguientes pasos para realizar la regresión logistica

1. Para comenzar hacemos los Import necesarios con los paquetes que utilizaremos.
2. Leemos el archivo csv y lo asignamos mediante Pandas a la variable dataframe.
3. A continuación llamamos al método dataframe.describe() que nos dará algo de información estadística básica de nuestro set de datos.
4. visualizamos en formato de historial los cuatro Features de entrada con nombres “duración”, “páginas”,”acciones” y “valor”. Y también podemos interrelacionar 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.
5. Ahora cargamos las variables de las 4 columnas de entrada en X excluyendo la columna “clase” con el método drop(). En cambio agregamos la columna “clase” en la variable y. Ejecutamos X.shape para comprobar la dimensión de nuestra matriz con datos de entrada de 170 registros por 4 columnas.
6. Y creamos nuestro modelo y hacemos que se ajuste (fit) a nuestro conjunto de entradas X y salidas ‘y’.

7. Una vez compilado nuestro modelo, le hacemos clasificar todo nuestro conjunto de entradas X Y confirmamos cuan bueno fue nuestro modelo
8. dividimos nuestros datos de entrada en forma aleatoria
9. Volvemos a compilar nuestro modelo de Regresión Logística pero esta vez sólo con 80 de los datos de entrada y calculamos el nuevo scoring que ahora nos da 74
10. Y ahora hacemos las predicciones
11. También podemos ver el reporte de clasificación con nuestro conjunto de Validación
12. vamos a inventar los datos de entrada de navegación y lo probamos en nuestro modelo

2.1 Código en Python

```
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
from sklearn.preprocessing import StandardScaler

# Mostrar gráficas
plt.show()

# Cargar datos
dataframe = pd.read_csv(r"usuarios_win_mac_lin.csv")

# Análisis descriptivo
print(dataframe.head())
print(dataframe.describe())
print(dataframe.groupby('clase').size())  # Analizar cuántos resultados hay por clase

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

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

```
# Preparar datos para el modelo
X = np.array(dataframe.drop(['clase'], axis=1))
y = np.array(dataframe['clase'])

# Escalar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Entrenar el modelo
model = linear_model.LogisticRegression(max_iter=1000)
model.fit(X_scaled, y)

# Realizar predicciones y mostrar los primeros 5
predictions = model.predict(X_scaled)
print(predictions[0:5])

# Validación y evaluación
model.score(X_scaled, y)
validation_size = 0.20
seed = 7
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split
(X_scaled, y,test_size=validation_size, random_state=seed)

# Validación cruzada
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)" % ('Logistic Regression', cv_results.mean(), cv_results.std())
print(msg)

# Evaluar en datos de validación
predictions = model.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))

# Realizar predicción para nuevos datos
X_new = pd.DataFrame({'duracion': [10], 'paginas': [3], 'acciones': [5], 'valor': [9]})
X_new_scaled = scaler.transform(X_new)  # Escalar los nuevos datos
print(model.predict(X_new_scaled))  # Realizar la predicción
```

3 Resultados

El modelo de regresión lineal generó los siguientes coeficientes:

```
[2 2 2 2 2]
0.77647058823529413
Logistic Regression: 0.743407 (0.115752)
0.852941176471
```

4 Conclusión

n este artículo, exploramos cómo construir un modelo de Regresión Logística en Python para clasificar el Sistema Operativo de los usuarios basándonos en sus características de navegación en un sitio web. A partir de este ejemplo, se puede adaptar a otras tareas en las que se necesite clasificar resultados en valores discretos. Si tuviéramos que predecir valores continuos, sería necesario utilizar Regresión Lineal.