Representación plot de datasets, selección de características y entrenamiento de modelos

Enlace a github: [github](https://github.com/Javiert54/bigData/tree/main/CEIABD_SNS/UT3%20-%20Algoritmos%20y%20herramientas%20para%20el%20aprendizaje%20supervisado/Actividad%203.4%20-%20Representación%20plot%20de%20datasets%2C%20selección%20de%20características%20y%20entrenamiento%20de%20modelos)

Enlace al recurso utilizado: [Kaggle](https://www.kaggle.com/code/akritiupadhyayks/phishing-detection-with-machine-learning#Logistic-Regression)

El objetivo de esta actividad es poner en práctica los conocimientos adquiridos para el   
preprocesamiento de datos, selección de características y entrenamiento de modelos.

Para ello es necesario seleccionar un Dataset que consideres oportuno, de clasificación o regresión, y distinto a los utilizados en clase.

Los puntos a desarrollar son los siguientes:

Sumario

[1. (5%) Describir el origen y breve explicación del Dataset, así como de cada una de las características. 1](#__RefHeading___Toc3299_3661859914)

[2. (5%) Procesamiento de datos en el dataset: ajustes en características con datos no informados, conversión de variables categóricas, etc. 6](#__RefHeading___Toc3677_3661859914)

[3. Utilizar las siguientes herramientas explicadas en clase para la selección de características: 6](#__RefHeading___Toc3744_3661859914)

[3.1. (10%) Matriz de gráficos de correlación. 6](#__RefHeading___Toc3746_3661859914)

[3.2. (10%) Matriz de gráficos de dispersión. 6](#__RefHeading___Toc3748_3661859914)

[3.3. (10%) SelectKBest. 6](#__RefHeading___Toc3750_3661859914)

### 1. (5%) Describir el origen y breve explicación del Dataset, así como de cada una de las características.

He podido experimentar con dos datasets:

- Uno grande y desbalanceado

- Uno pequeño y balanceado

El que mejores resultados me ha dado ha sido el primero, pero he puesto los dos en el github.

Ambos datasets provienen del resurso, cuyo link está al principio, junto al link de mi github.

Estos datasets ya fueron probados, limpiados y configurados específicamente para la tarea de detectar páginas de phishing, de modo que apenas fue necesario realizar una limpieza de datos.

Dataset Grande (dataset\_full.csv):

- Número total de instancias: 88,647

- Número de páginas web legítimas (Etiquetado como 0 en la característica phishing): 58,000

- Número de páginas web phishing (Etiquetado como 1 en la característica phishing): 30,647

- Número de características: 112 (Incluyendo la característica phishing)

Dataset Pequeño (dataset\_small.csv):

- Número total de instancias: 58,645

- Número de páginas web legítimas (Etiquetado como 0 en la característica phishing): 27,998

- Número de páginas web phishing (Etiquetado como 1 en la característica phishing): 30,647

- Número de características: 112 (Incluyendo la característica phishing)

Aquí especifico todas sus características:

| Característica | Descripción |
| --- | --- |
| qty\_dot\_url | count (.) en la URL |
| qty\_hyphen\_url | count (-) en la URL |
| qty\_underline\_url | count (\_) en URL |
| qty\_slash\_url | count (/) en la URL |
| qty\_questionmark\_url | count (?) en la URL |
| qty\_equal\_url | count (=) en URL |
| qty\_at\_url | recuento (@) en URL |
| qty\_and\_url | recuento (&) en URL |
| qty\_exclamation\_url | count (!) en la URL |
| qty\_space\_url | count ( ) en la URL |
| qty\_tilde\_url | count (~) en la URL |
| qty\_comma\_url | count (,) en la URL |
| qty\_plus\_url | recuento (+) en URL |
| qty\_asterisk\_url | count (\*) en la URL |
| qty\_hashtag\_url | count (#) en la URL |
| qty\_dollar\_url | count ($) en la URL |
| qty\_percent\_url | recuento (%) en URL |
| qty\_tld\_url | longitud del dominio de nivel superior |
| length\_url | Longitud de la URL |
| qty\_dot\_domain | count (.) en el dominio |
| qty\_hyphen\_domain | count (-) en el dominio |
| qty\_underline\_domain | count (\_) en el dominio |
| qty\_slash\_domain | count (/) en dominio |
| qty\_questionmark\_domain | count (?) en el dominio |
| qty\_equal\_domain | count (=) en el dominio |
| qty\_at\_domain | Recuento (@) en dominio |
| qty\_and\_domain | Recuento (&) en dominio |
| qty\_exclamation\_domain | count (!) en el dominio |
| qty\_space\_domain | count ( ) en el dominio |
| qty\_tilde\_domain | count (~) en el dominio |
| qty\_comma\_domain | count (,) en el dominio |
| qty\_plus\_domain | Recuento (+) en dominio |
| qty\_asterisk\_domain | count (\*) en el dominio |
| qty\_hashtag\_domain | count (#) en el dominio |
| qty\_dollar\_domain | count ($) en el dominio |
| qty\_percent\_domain | Recuento (%) en dominio |
| qty\_vowels\_domain | Contar vocales en dominio |
| domain\_length | Longitud del dominio |
| domain\_in\_ip | Dominio URL en formato de dirección IP |
| server\_client\_domain | domain contiene las palabras clave "servidor" o "cliente" |
| qty\_dot\_directory | count (.) en el directorio |
| qty\_hyphen\_directory | count (-) en el directorio |
| qty\_underline\_directory | count (\_) en el directorio |
| qty\_slash\_directory | count (/) en el directorio |
| qty\_questionmark\_directory | count (?) en el directorio |
| qty\_equal\_directory | count (=) en el directorio |
| qty\_at\_directory | count (@) en el directorio |
| qty\_and\_directory | count (&) en el directorio |
| qty\_exclamation\_directory | count (!) en el directorio |
| qty\_space\_directory | count ( ) en el directorio |
| qty\_tilde\_directory | count (~) en el directorio |
| qty\_comma\_directory | count (,) en el directorio |
| qty\_plus\_directory | count (+) en el directorio |
| qty\_asterisk\_directory | count (\*) en el directorio |
| qty\_hashtag\_directory | count (#) en el directorio |
| qty\_dollar\_directory | count ($) en el directorio |
| qty\_percent\_directory | recuento (%) en el directorio |
| directory\_length | Longitud del directorio |
| qty\_dot\_file | count (.) en el archivo |
| qty\_hyphen\_file | count (-) en el archivo |
| qty\_underline\_file | count (\_) en el archivo |
| qty\_slash\_file | count (/) en el archivo |
| qty\_questionmark\_file | count (?) en el archivo |
| qty\_equal\_file | count (=) en el archivo |
| qty\_at\_file | count (@) en el archivo |
| qty\_and\_file | Recuento (&) en el archivo |
| qty\_exclamation\_file | count (!) en el archivo |
| qty\_space\_file | recuento ( ) en el archivo |
| qty\_tilde\_file | count (~) en el archivo |
| qty\_comma\_file | count (,) en el archivo |
| qty\_plus\_file | Recuento (+) en el archivo |
| qty\_asterisk\_file | recuento (\*) en el archivo |
| qty\_hashtag\_file | count (#) en el archivo |
| qty\_dollar\_file | count ($) en el archivo |
| qty\_percent\_file | Recuento (%) en el archivo |
| file\_length | Longitud del archivo |
| qty\_dot\_params | count (.) en parámetros |
| qty\_hyphen\_params | count (-) en parámetros |
| qty\_underline\_params | count (\_) en parámetros |
| qty\_slash\_params | count (/) en parámetros |
| qty\_questionmark\_params | count (?) en parámetros |
| qty\_equal\_params | count (=) en parámetros |
| qty\_at\_params | count (@) en parámetros |
| qty\_and\_params | count (&) en parámetros |
| qty\_exclamation\_params | count (!) en parámetros |
| qty\_space\_params | count ( ) en parámetros |
| qty\_tilde\_params | count (~) en parámetros |
| qty\_comma\_params | count (,) en parámetros |
| qty\_plus\_params | Recuento (+) en parámetros |
| qty\_asterisk\_params | count (\*) en parámetros |
| qty\_hashtag\_params | count (#) en parámetros |
| qty\_dollar\_params | count ($) en parámetros |
| qty\_percent\_params | Recuento (%) en parámetros |
| params\_length | Parámetros Longitud |
| tld\_present\_params | Presencia de TLD en los argumentos |
| qty\_params | Número de parámetros |
| email\_in\_url | correo electrónico presente en la URL |
| time\_response | Tiempo de búsqueda (respuesta) Dominio (búsqueda) |
| domain\_spf | el dominio tiene SPF |
| asn\_ip | Número de AS (o ASN) |
| time\_domain\_activation | Tiempo (en días) de la activación del dominio |
| time\_domain\_expiration | Tiempo (en días) de vencimiento del dominio |
| qty\_ip\_resolved | número de direcciones IP resueltas |
| qty\_nameservers | número de servidores de nombres resueltos (NameServers - NS) |
| qty\_mx\_servers | número de servidores MX |
| ttl\_hostname | Valor de tiempo de vida (TTL) asociado con el nombre de host |
| tls\_ssl\_certificate | Certificado TLS / SSL válido |
| qty\_redirects | Número de redireccionamientos |
| url\_google\_index | comprobar si la URL está indexada en Google |
| domain\_google\_index | comprobar si el dominio está indexado en Google |
| url\_shortened | comprobar si la URL está acortada |
| phishing | es un sitio web de phishing |

Todas las características aportan información relevante. La mayoría de características simplemente es el número de veces que podemos encontrar un carácter en particular dentro de la url (esos caracteres se utilizan como operadores, así que aportan mucha información). Además, entre algunas de estas características, podemos ver que hay otras especialmente interesantes, como:

- url\_google\_index (Comprueba si google indexa la página)

- qty\_redirects (Comprueba si redirecciona a otra página, y de hacerlo cuántas veces lo hace)

- tls\_ssl\_certificate (Comprueba si tiene el certificado tls ssl)

### 2. (5%) Procesamiento de datos en el dataset: ajustes en características con datos no informados, conversión de variables categóricas, etc.

Como se mencionó anteriormente, no fue necesario realizar ningún cambio dentro del dataset, ya que ya venía preparado para utilizarse en este tipo de tareas. Sin embargo, si hubiera tenido que realizar algún cambio de balance, lo habría hecho con esta herramienta: [Phishing Datasets Web App](https://gregavrbancic.github.io/Phishing-Dataset/), que me permitía customizar el dataset que obtengo, por si lo hubiera necesitado para otra cosa.

### 3. Utilizar las siguientes herramientas explicadas en clase para la selección de características:

### 3.1. (10%) Matriz de gráficos de correlación.

Para mostrar la matriz de gráficos de correlación, usé el siguiente código:

import seaborn as sns

import matplotlib.pyplot as plt

def showCorrelationMatrix(dataFrame):

    # Tomar una muestra del dataset (por ejemplo, 1000 filas)

    df\_sample = dataFrame.sample(n=1000, random\_state=1)

    # Selecciono 5 variables (ya que si las selecciono todas, tardará demasiado)

    selected\_columns = df\_sample.columns[:5]

    df\_sample = df\_sample[selected\_columns]

    # Crear la matriz de gráficos de correlación

    sns.pairplot(df\_sample, diag\_kind='kde', markers='+', plot\_kws={'alpha': 0.5})

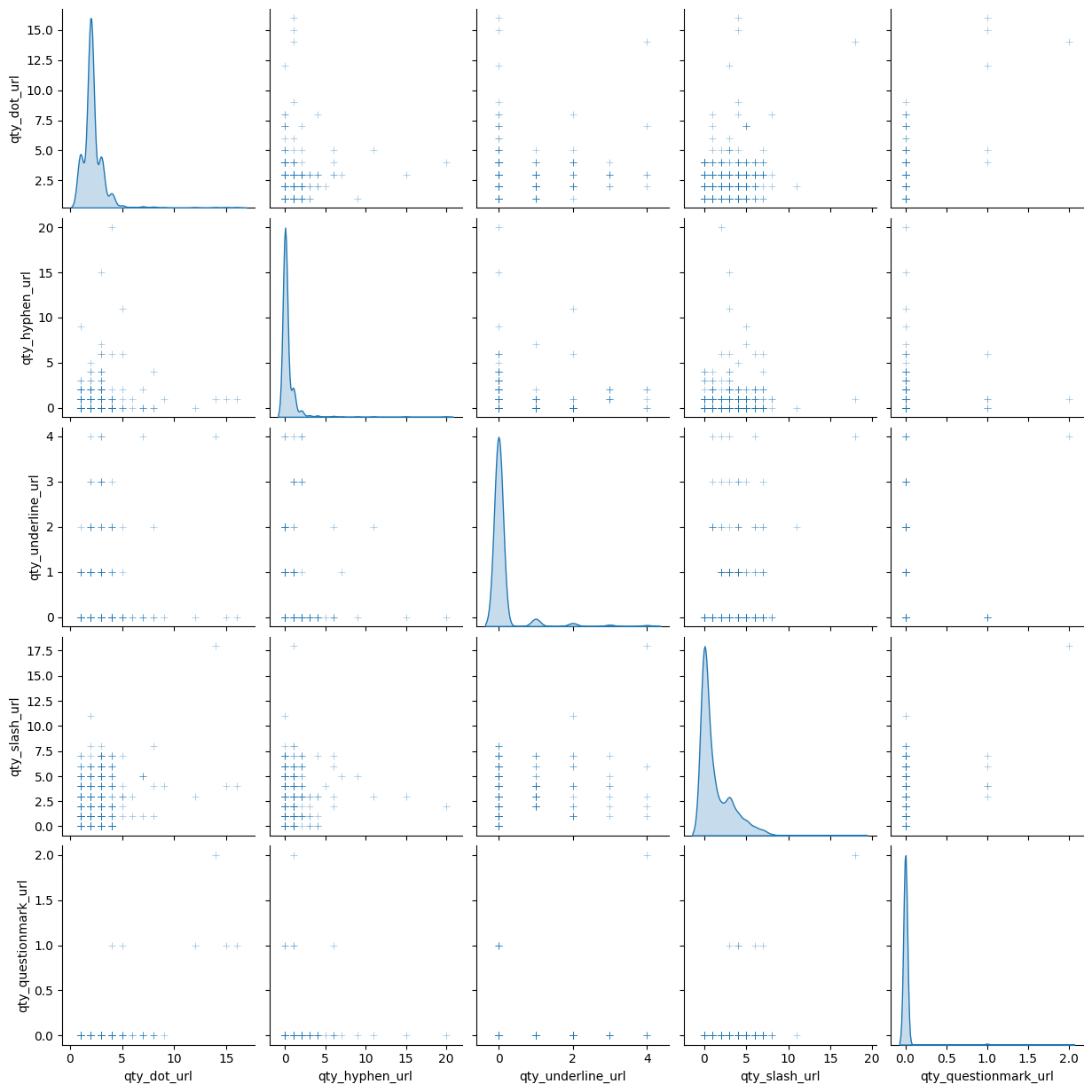
    # Mostrar la gráfica

    plt.show()

correlation = showCorrelationMatrix(df)

correlation

Aquí los resultados:



### 3.2. (10%) Matriz de gráficos de dispersión.

Para mostrar la matriz de gráficos de dispersión, utilicé este código:

def showDispersion(dataFrame):

    # Seleccionar una muestra del conjunto de datos

    df\_sample = dataFrame.sample(n=1000, random\_state=1)

    # Selecciono 5 variables (ya que si las selecciono todas, tardará demasiado)

    selected\_columns = df\_sample.columns[:5]

    df\_sample = df\_sample[selected\_columns]

    # Crear la matriz de gráficos de dispersión

    sns.pairplot(df\_sample)

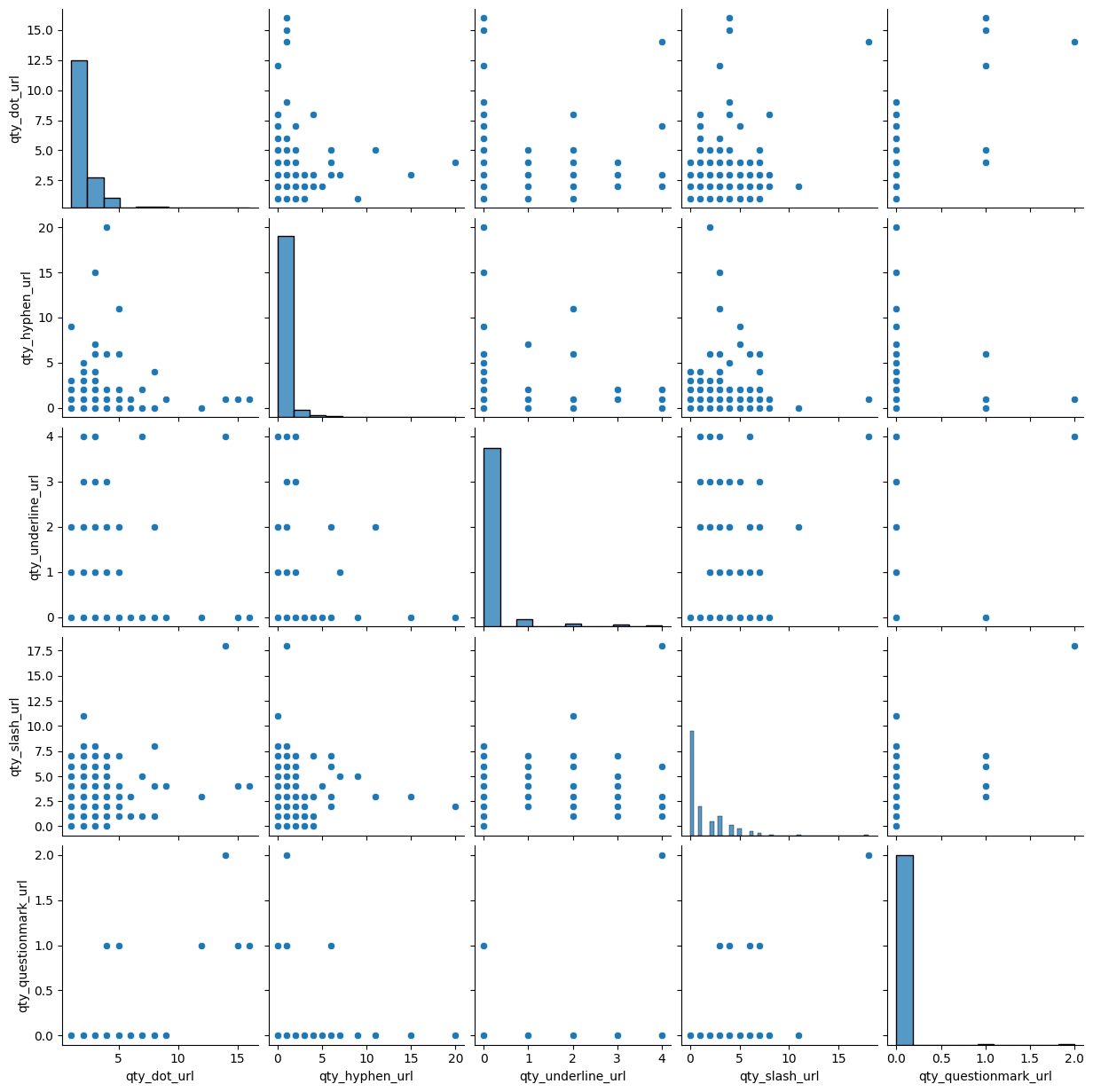
    # Mostrar la matriz

    plt.show()

dispersion = showDispersion(df)

dispersion

Aquí los resultados:



### 3.3. (10%) SelectKBest.

SelectKBest nos permite seleccionar las mejores características del dataset para no tener que gastar demasiado tiempo en el entrenamiento (ya que tendremos que tener menos características en cuenta).

Este es el código utilizado:

from sklearn.feature\_selection import SelectKBest, f\_classif

selector = SelectKBest(score\_func=f\_classif, k=50)  # Selecciona las 50 mejores características

Xtrain\_new = selector.fit\_transform(Xtrain, ytrain)

Xtest\_new = selector.transform(Xtest)

model = GaussianNB()

model.fit(Xtrain\_new, ytrain)

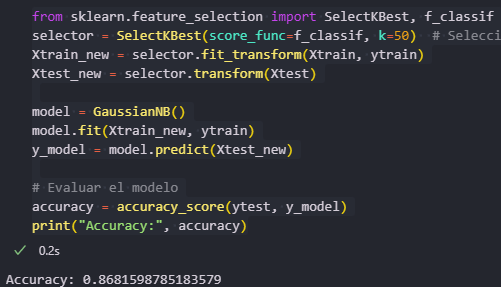
y\_model = model.predict(Xtest\_new)

# Evaluar el modelo

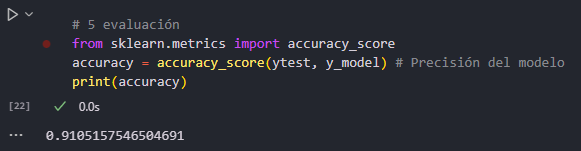
accuracy = accuracy\_score(ytest, y\_model)

print("Accuracy:", accuracy)

Si entrenamos al modelo con las 50 características que ha seleccionado, obtenemos una precisión del 87%:



Sin embargo, al no utilizar SelectKBest, obtenemos una precisión del 91%:



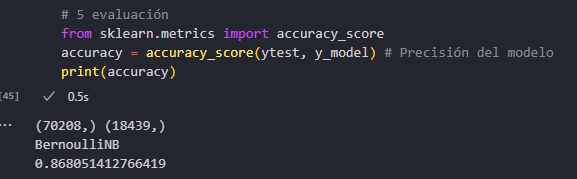
### 4. (5%) Una pequeña reflexión sobre la elección de las características elegidas.

Desconozco por qué entrenar el modelo con las características recomendadas baja tanto la precisión. Me parece raro, ya que también he probado a usar todas las características junto con SelectKBest, pero sigo sin obtener la misma precisión.

### 5. Con las librerías para NaiveBayes vistas en clase, entrenar el modelo que consideres más adecuado.

Entre los modelos de Naive Bayes, encontramos el BernoulliNB, que se especializa en clasificaciones binarias.

Parece ideal para clasificar URLs en phishing/no phishing, así que lo pruebo:



Como se puede ver, obtengo una precisión decente, pero aún inferior a la obtenida con el NB Gaussiano (91%). De modo que el más adecuado para seguir, a juzgar por los resultados, es el NB Gaussiano. Y es por eso que será el modelo que utilizaré.

### 5.1. (10%) Sin utilizar Cross Validation.

Para mostrar los resultados de la comparación de usar y no usar cross-validation, he usado el siguiente código:

from sklearn.model\_selection import cross\_val\_score

# 1. Elegimos clasificador

from sklearn.naive\_bayes import GaussianNB

# 2. Instanciamos el modelo

model = GaussianNB()

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

print(model.\_\_class\_\_.\_\_name\_\_)

for column in Xtrain.columns:

    Xtrain[column] = label\_encoder.fit\_transform(Xtrain[column])

    # Apply label encoding to each column in Xtrain

for column in Xtest.columns:

    Xtest[column] = label\_encoder.fit\_transform(Xtest[column])

    # Now you can fit your model

model.fit(Xtrain, ytrain)

# Evaluar el modelo sin usar cross-validation

y\_model = model.predict(Xtest)

accuracy = accuracy\_score(ytest, y\_model)

print("Accuracy sin cross-validation:", accuracy)

# Aplicar validación cruzada

scores = cross\_val\_score(model, Xtrain, ytrain, cv=5)  # cv=5 para 5-fold cross-validation

print("Cross-validation scores:", scores)

print("Mean cross-validation score:", scores.mean())

* Aquí temenos el resultado sin cross-validation:



### 5.2. (15%) Utilizando Cross Validation.

* Aquí tenemos el resultado con cross-validation:



### 6. (5%) Obtener una conclusión sobre los resultados obtenidos en la predicción y evaluación al utilizar o no Cross Validation.

- Sin Cross-Validation: La precisión obtenida puede ser específica de la partición de datos utilizada y no necesariamente representativa del rendimiento general del modelo.

- Con Cross-Validation: Los puntajes de validación cruzada y su promedio proporcionan una evaluación más robusta y confiable del rendimiento del modelo, ya que consideran múltiples particiones de los datos.