

PROYECTO FINAL. Modelos y Aprendizajes

Autor: Eduardo Báez Agama

A. DESCRIPCIÓN DEL PROYECTO

Supongamos que usted trabaja en el servicio de salud y recibe muestras que provienen de mujeres con cáncer de mama.

Los médicos han extraído características y las han anotado, su trabajo es crear un modelo que sea capaz de identificar si un paciente tiene o no cáncer.

Recordemos que un falso positivo no es tan preocupante como un falso negativo, ya que en el futuro se le hacen más pruebas a las pacientes y hay oportunidades de descubrir que estábamos en un error.

Sin embargo, un falso negativo puede llevar a que el cáncer se desarrolle sin supervisión durante más tiempo del necesario y podría llevar a daños más graves o incluso la muerte de la paciente.

Teniendo esto en cuenta, desarrolla un modelo que funcione lo mejor posible y explica qué decisiones has tomado en su elaboración y por qué.

A entregar obligatoriamente:

Link a un repositorio público de Github que contenga al menos:

- Un archivo Jupyter Notebook con todas las celdas ejecutadas en orden. Es decir, que antes de subir el archivo a github habéis limpiado el notebook y luego lo habéis ejecutado desde el principio).
- Un archivo Readme en el que se explica el proyecto y el ejercicio. Tened en cuenta que este repositorio puede servir como CV en el futuro y que los recruiters suelen mirar los archivos Readme.md
- Una carpeta data con el dataset.

En el notebook debe aparecer el proceso de preprocesado de datos desde los archivos originales a ser posible.

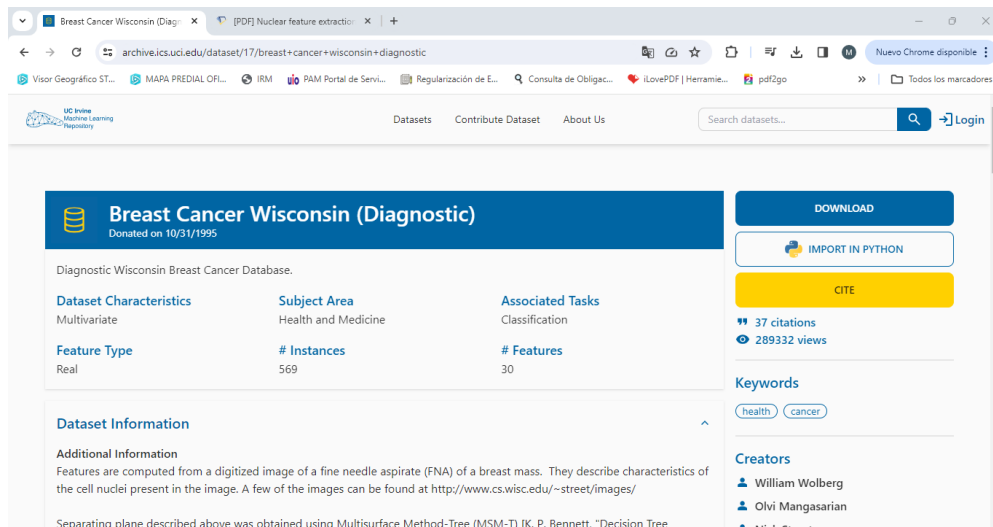
En el notebook debéis probar al menos con 3 modelos, evaluarlos y decidir cual es el mejor, justificando la respuesta en base a las matrices de confusión que aparecen al evaluar el error en training y en test.

El dataset y su descripción aparecen aquí:

<https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>

B. DESARROLLO DEL PROYECTO

1. Revisión preliminar de la dirección URL en donde se encuentra el Dataset y su descripción:



Breast Cancer Wisconsin (Diagnostic)
Donated on 10/31/1995

Diagnostic Wisconsin Breast Cancer Database.

Dataset Characteristics
Multivariate

Subject Area
Health and Medicine

Associated Tasks
Classification

Feature Type
Real

Instances
569

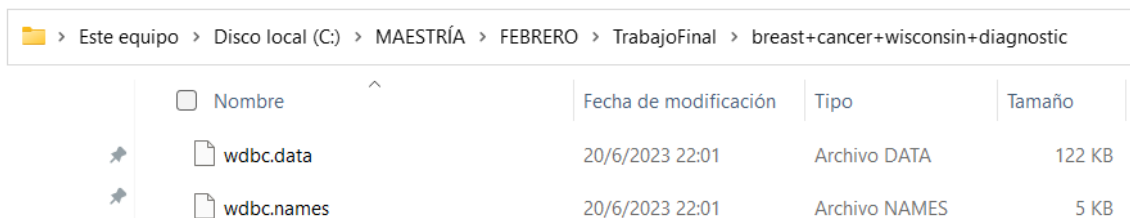
Features
30

Dataset Information
Additional Information
Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at <http://www.cs.wisc.edu/~street/images/>
Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree

Keywords
health cancer

Creators
William Wolberg
Olvi Mangasarian

2. Descarga de datos en un repositorio local:

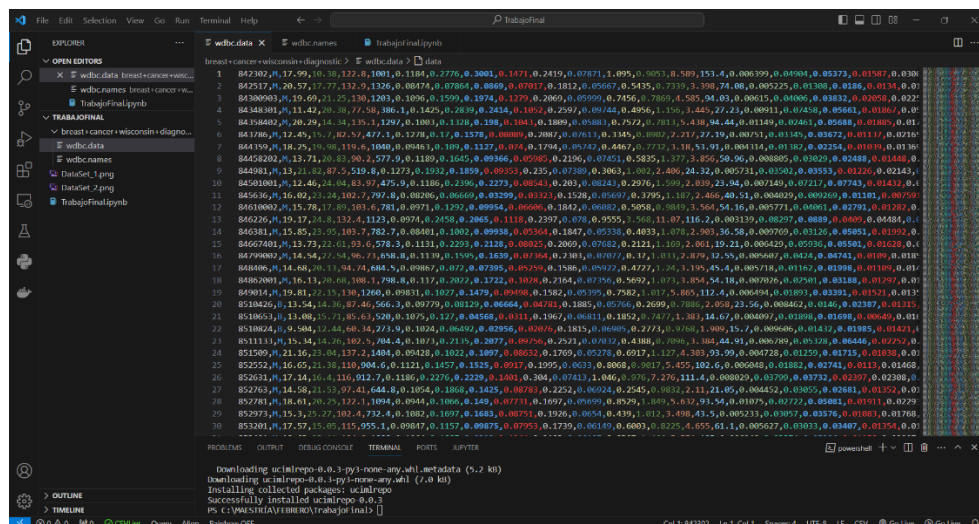


Este equipo > Disco local (C:) > MAESTRÍA > FEBRERO > TrabajoFinal > breast+cancer+wisconsin+diagnostic

Nombre	Fecha de modificación	Tipo	Tamaño
wdbc.data	20/6/2023 22:01	Archivo DATA	122 KB
wdbc.names	20/6/2023 22:01	Archivo NAMES	5 KB

3. Open Folder en Visual Studio Code de la carpeta que contiene los sets de datos:

- wdbc.data



Visual Studio Code interface showing the file explorer on the left with the 'wdbc.data' file selected. The main editor displays the content of 'wdbc.data', which is a list of numerical features for each instance. The file explorer on the left shows the project structure, including the 'wdbc.data' file.

- wdbc.names

```

wdbc.names
-----
5. Number of Instances: 569
6. Number of Attributes: 32 (10, diagnosis, 30 real-valued input features)
7. Attribute Information
   1. ID number
   2. Diagnosis (M = malignant, B = benign)
   3-32
   Ten real valued features are computed for each cell nucleus:
   a) radius (mean of distances from center to points on the perimeter)
   b) texture (standard deviation of gray-scale values)
   c) perimeter
   d) area
   e) smoothness (local variation in radius lengths)
   f) compactness (perimeter2 / area - 1.0)
   g) concavity (severity of concave portions of the contour)
   h) concave points (number of concave portions of the contour)
   i) symmetry
   j) fractal dimension ("coastline approximation" - 1)
   Several of the papers listed above contain detailed descriptions of
   how these features are computed.
   The mean, standard error, and "worst" or largest (mean of the three
   largest values) of these features were computed for each image,
   resulting in 30 features. For instance, field 1 is Mean Radius, field
   14 is Radius St, field 23 is Worst Radius.
   All feature values are recoded with four significant digits.
8. Missing attribute values: none
9. Class distribution: 357 benign, 212 malignant
  
```

4. Revisión y Ejecución de Import in Python de la página en donde están alojado el dataset.

The screenshot shows the UC Irvine Machine Learning Repository page for the Breast Cancer Wisconsin dataset. A modal window is open, providing instructions on how to import the dataset into Python. The modal includes the following text:

```

Install the ucimlrepo package
pip install ucimlrepo

Import the dataset into your code
from ucimlrepo import fetch_ucirepo

# fetch dataset
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)

# data (as pandas dataframes)
X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets

# metadata
print(breast_cancer_wisconsin_diagnostic.metadata)

# variable information
print(breast_cancer_wisconsin_diagnostic.variables)

View the full documentation
  
```

A red arrow points to the 'IMPORT IN PYTHON' button in the modal window.

5. Descripción del archivo Jupyter Notebook: TrabajoFinal.ipynb

https://github.com/EduardoBaez/Grupo7_ModelosYAprendizajes/blob/40e8af4ee371f42b08ab8cade69d7d4648faf80e/TrabajoFinal.ipynb

▼ Carga de Librerías

```
from ucimlrepo import fetch_ucirepo
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier # Importar MLPClassifier
from sklearn.preprocessing import StandardScaler
```

[1] ✓ 1.8s

Python

Archivos Originales

```
# fetch dataset
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)
```

✓ 1.5s

Python

```
# data (as pandas dataframes)
X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets
```

✓ 0.0s

Python

```
# metadata
int(breast_cancer_wisconsin_diagnostic.metadata)
# variable information
print(breast_cancer_wisconsin_diagnostic.variables)
```

✓ 0.0s

Python

```
{'uci_id': 17, 'name': 'Breast Cancer Wisconsin (Diagnostic)', 'repository_url': 'https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnost
    name      role      type demographic description units \
0          ID      ID  Categorical      None      None      None
1    Diagnosis  Target  Categorical      None      None      None
2    radius1  Feature   Continuous      None      None      None
3    texture1  Feature   Continuous      None      None      None
4    perimeter1  Feature   Continuous      None      None      None
5        area1  Feature   Continuous      None      None      None
6    smoothness1  Feature   Continuous      None      None      None
7    compactness1  Feature   Continuous      None      None      None
8    concavity1  Feature   Continuous      None      None      None
9    concave_points1  Feature   Continuous      None      None      None
```

Carga de Datos breast_cancer_wisconsin_diagnostic utilizando URL

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data"
column_names = ['ID', 'Diagnosis', 'Mean Radius', 'Mean Texture', 'Mean Perimeter', 'Mean Area', 'Mean Smoothness', 'Mean Compactness', 'Mean Concavi
data = pd.read_csv(url, names=column_names, header=None)
print(data)
```

✓ 1.0s

Python

```
    ID Diagnosis  Mean Radius  Mean Texture  Mean Perimeter  Mean Area \
0   842302      M      17.99      10.38      122.80      1001.0
1   842517      M      20.57      17.77      132.90      1326.0
2   84300903     M      19.69      21.25      130.00      1203.0
3   84348301     M      11.42      20.38      77.58      386.1
4   84358402     M      20.29      14.34      135.10      1297.0
..    ...      ...      ...      ...      ...      ...
564  926424     M      21.56      22.39      142.00      1479.0
565  926682     M      20.13      28.25      131.20      1261.0
566  926954     M      16.60      28.08      108.30      858.1
567  927241     M      20.60      29.33      140.10      1265.0
568   92751     B       7.76      24.54      47.92      181.0

    Mean Smoothness  Mean Compactness  Mean Concavity  Mean Concave Points \
0      0.11840      0.27760      0.30010      0.14710
1      0.08474      0.07864      0.08690      0.07017
2      0.10960      0.15990      0.19740      0.12790
3      0.14250      0.28390      0.24140      0.10520
4      0.10030      0.13280      0.19800      0.10430
..    ...      ...      ...      ...
564      0.11100      0.11590      0.24390      0.13890
565      0.09780      0.10340      0.14400      0.09791
566      0.08455      0.10230      0.09251      0.05302
567      0.11780      0.27700      0.35140      0.15200
568      0.05263      0.04362      0.09900      0.08000
```

```
Preprocesamiento de Datos

# Eliminar la columna ID
data.drop('ID', axis=1, inplace=True)
# Convertir la columna de Diagnóstico a valores numéricos (Maligno=1, Benigno=0)
data['Diagnosis'] = data['Diagnosis'].map({'M': 1, 'B': 0})

# Se dividen los datos en conjuntos de train y test
X = data.drop('Diagnosis', axis=1)
y = data['Diagnosis']
print("Número de instancias:", X.shape[0])
print("Número de características:", X.shape[1])
print("Número de etiquetas:", len(set(y)))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

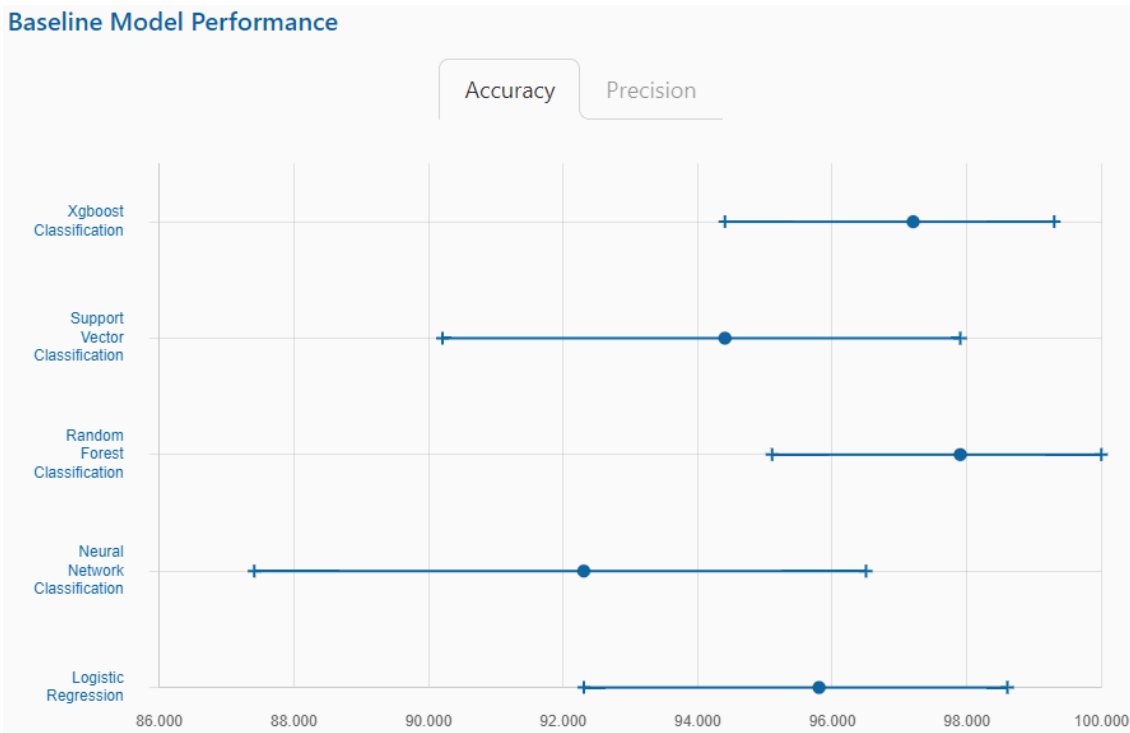
# Se escalan las características para estandarizarlas antes de entrenar el modelo
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Número de instancias: 569
Número de características: 30
Número de etiquetas: 2

```
Definición de Modelos

models = {
    'Logistic Regression': LogisticRegression(max_iter=10000, solver='liblinear'), # Aumentamos el número máximo de iteraciones
    'Random Forest': RandomForestClassifier(),
    'Neural Network': MLPClassifier()
}
```

Se escogieron estos modelos con base a la siguiente explicación de Accuracy de la página de explicación del Dataset:



▼ Entrenamiento de Modelos y Cálculo de Matrices de Confusión

```
# Entrenamiento de modelos
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)

# Cálculo y muestra de matrices de confusión de Entrenamiento y Prueba
cm_train = confusion_matrix(y_train, y_pred_train)
cm_test = confusion_matrix(y_test, y_pred_test)

print(f"Modelo: {name}")
print("Accuracy de entrenamiento:", accuracy_score(y_train, y_pred_train))
print("Accuracy de prueba:", accuracy_score(y_test, y_pred_test))
report = classification_report(y_test, y_pred_test)
print(f"Reporte de Clasificación:\n{report}")

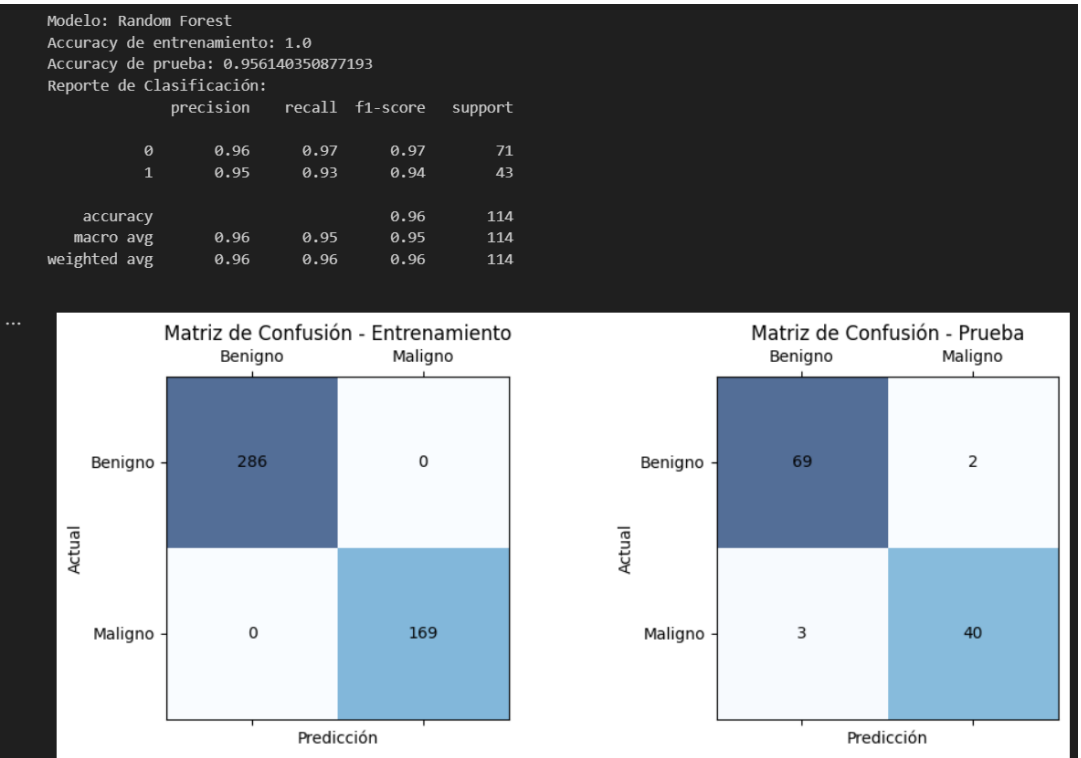
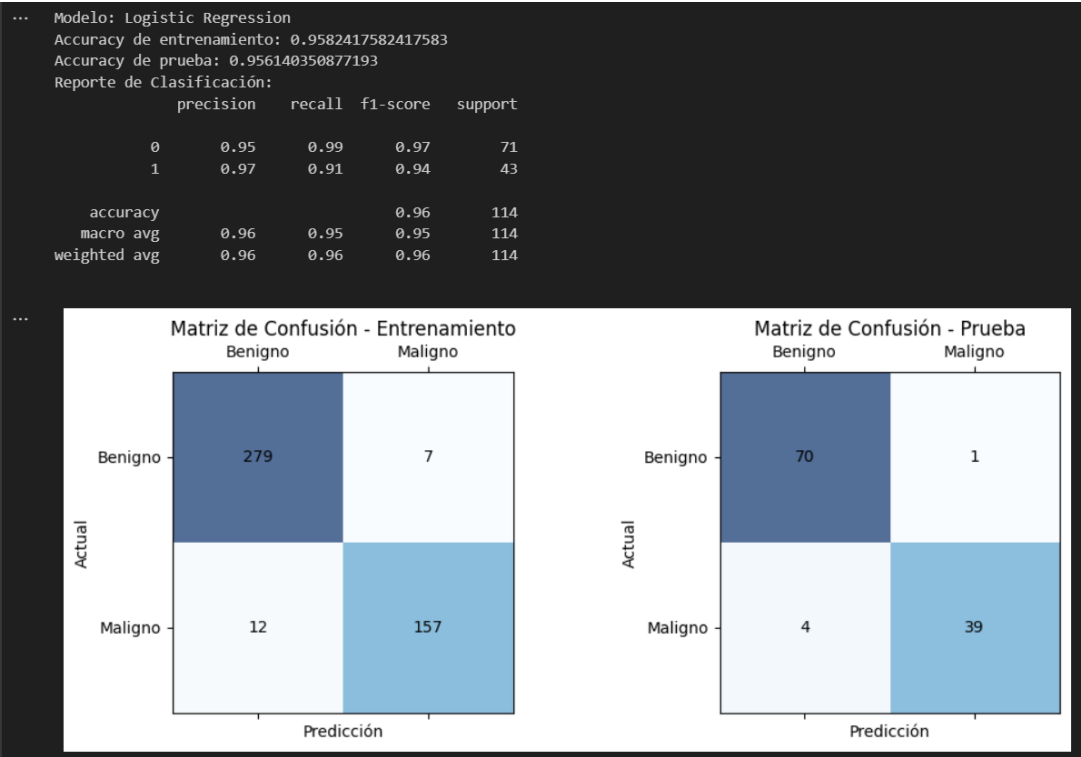
fig, axes = plt.subplots(1, 2, figsize=(10, 4))
labels = ['Benigno', 'Maligno']

# Matriz de confusión para el conjunto de entrenamiento
axes[0].matshow(cm_train, cmap=plt.cm.Blues, alpha=0.7)
for i in range(cm_train.shape[0]):
    for j in range(cm_train.shape[1]):
        axes[0].text(x=j, y=i, s=cm_train[i, j], va='center', ha='center')
axes[0].set_xticks(range(len(labels)))
axes[0].set_xticklabels(labels)
axes[0].set_yticks(range(len(labels)))
axes[0].set_yticklabels(labels)
axes[0].set_xlabel('Predicción')
axes[0].set_ylabel('Actual')
axes[0].set_title('Matriz de Confusión - Entrenamiento')
```

```
# Matriz de confusión para el conjunto de prueba
axes[1].matshow(cm_test, cmap=plt.cm.Blues, alpha=0.7)
for i in range(cm_test.shape[0]):
    for j in range(cm_test.shape[1]):
        axes[1].text(x=j, y=i, s=cm_test[i, j], va='center', ha='center')
axes[1].set_xticks(range(len(labels)))
axes[1].set_xticklabels(labels)
axes[1].set_yticks(range(len(labels)))
axes[1].set_yticklabels(labels)
axes[1].set_xlabel('Predicción')
axes[1].set_ylabel('Actual')
axes[1].set_title('Matriz de Confusión - Prueba')

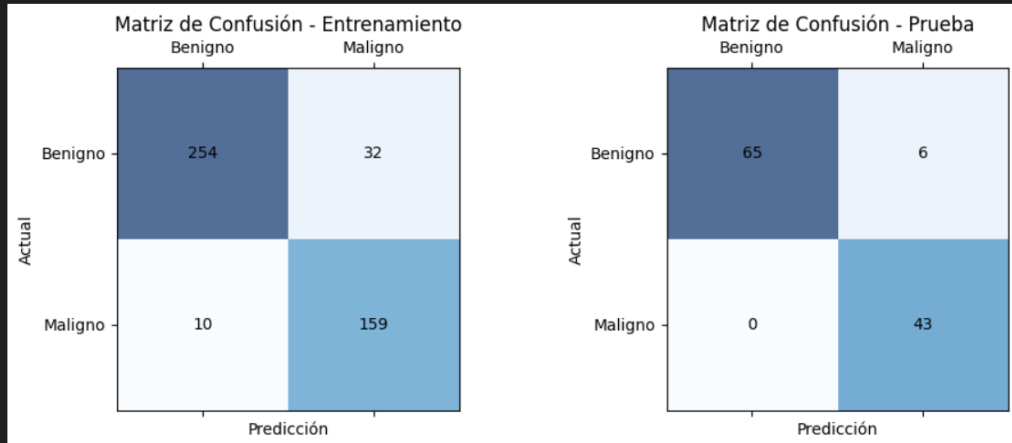
plt.tight_layout()
plt.show()
print("-----")
```

C. PRESENTACIÓN DE RESULTADOS



Modelo: Neural Network
 Accuracy de entrenamiento: 0.9076923076923077
 Accuracy de prueba: 0.9473684210526315
 Reporte de Clasificación:

	precision	recall	f1-score	support
0	1.00	0.92	0.96	71
1	0.88	1.00	0.93	43
accuracy			0.95	114
macro avg	0.94	0.96	0.95	114
weighted avg	0.95	0.95	0.95	114



D. CONCLUSION

Los Resultados se basan en la consideración de que un Falso Positivo no es tan preocupante como un **Falso Negativo**, ya que en el futuro se le hacen más pruebas a las pacientes y hay oportunidades de descubrir que estábamos en un error.

		Predicción			
		Positivo	Negativo		
Actual	Positivo	Verdaderos Positivos	Falsos Negativos	←	dato real = 1 dato predicho = 0
	Negativo	Falsos Positivos	Verdaderos Negativos	←	dato real = 0 dato predicho = 0

↑ dato real = 1 dato real = 0
 dato predicho = 1 dato predicho = 1

Por tal motivo el mejor modelo comparativamente de los tres analizados es: **Random Forest**, ya que su cantidad de Falsos Negativos es 2 versus su cantidad de Falsos Positivos que es 3; adicionalmente, su Recall en valor 0 es de 0.97 y en Valor 1 es de 0.93