

**Instituto Tecnológico y de Estudios Superiores de Monterrey**  
Campus Estado de México

Inteligencia artificial avanzada para la ciencia de datos I  
(Gpo 101)



**Tecnológico  
de Monterrey**

Inteligencia artificial avanzada para la ciencia de datos I Concentración, (Gpo 101)

Jorge Adolfo Ramírez Uresti

**Momento de Retroalimentación: Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución. (Portafolio Implementación)**

Oswaldo Daniel Hernández de Luna | A01753911

09 de septiembre 2024

# Introducción

Este proyecto tiene como objetivo implementar un modelo híbrido de aprendizaje automático utilizando los algoritmos K-Means y Árbol de Decisión para predecir si un tumor es maligno o benigno, por lo que, los algoritmos seleccionados permiten combinar las fortalezas del aprendizaje supervisado y no supervisado, proporcionando una mayor capacidad para identificar patrones en los datos médicos y mejorar la precisión de las predicciones.

El aprendizaje supervisado consiste en descubrir una función que aproxima una función desconocida real  $f$  a partir de un conjunto de entrenamiento, donde los datos contienen pares entrada-salida conocidos  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , por lo que, en este proyecto, utilizo el Árbol de Decisión, que es un modelo supervisado que representa una función de decisión, en la cual los valores de salida se determinan a través de una serie de pruebas basadas en los atributos de entrada, este modelo es ideal para problemas de clasificación binaria.

Por otro lado, el aprendizaje no supervisado no cuenta con salidas etiquetadas, lo que permite al modelo descubrir patrones o agrupaciones dentro de los datos, así que, el algoritmo K-Means es una técnica de agrupamiento que minimiza la distancia dentro de los clusters, asociando los puntos de datos con los centroides más cercanos.

De igual manera dentro de este mismo proyecto, K-Means nos ayuda a identificar grupos de tumores que comparten características similares antes de aplicar el árbol de decisión.

Este enfoque híbrido, que combina K-Means para el agrupamiento y Árbol de Decisión para la clasificación, busca mejorar la precisión del modelo, reducir los errores de clasificación y proporcionar una herramienta poderosa para el diagnóstico médico.

## Desarrollo del proyecto

### Separación de responsabilidades:

**models/decision\_tree.py** y **models/kmeans\_model.py**: Al tener los modelos en archivos separados, aseguramos que la lógica del árbol de decisión y K-Means esté completamente aislada, lo que facilita su modificación o mejora sin afectar el resto del proyecto.

**utils/data\_preparation.py**: Este archivo maneja las tareas relacionadas con la preparación de los datos (como limpieza y transformación), manteniéndolas separadas de la lógica del modelo. Esto promueve la reutilización de código y mantiene el código más limpio.

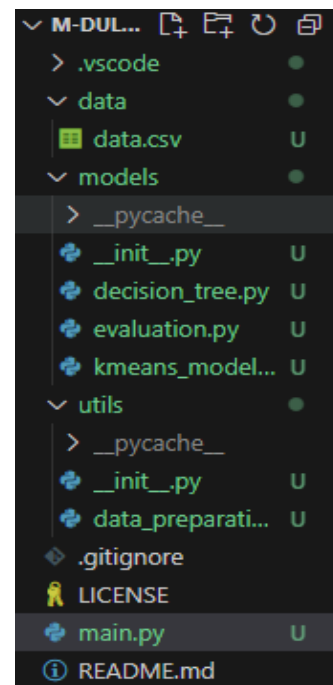
### Organización del código:

El archivo principal **main.py** actúa como el punto de entrada al programa, permitiendo que el flujo del código sea claro y centralizado. El resto de los archivos manejan tareas específicas, lo que hace que el código sea más legible y fácil de seguir.

### Mantenibilidad:

En proyectos grandes, un código bien estructurado es esencial para mantenerlo a largo plazo. Si cada parte del proyecto está organizada en módulos, es más fácil identificar y corregir errores, actualizar funcionalidades, o agregar nuevas características sin afectar otras partes del código.

Esta estructura no solo facilita el desarrollo actual del proyecto, sino que también prepara la idea para futuras expansiones y mejora la calidad general del código, promoviendo la colaboración y la reutilización de componentes, siendo programador, una estructura para un modelo en desarrollo de software la mejor manera de realizar esto es de la forma mostrada a continuación.



## *Estructura del proyecto*

El proyecto se ha organizado en torno a varios módulos interconectados que permiten una implementación modular y reutilizable del código, por lo que, esta estructura facilita el mantenimiento y la escalabilidad del proyecto, los archivos principales incluyen:

### *Preparación de Datos: data\_preparation.py*

Este archivo es responsable del preprocesamiento de los datos, lo cual es un paso crítico en el desarrollo de cualquier modelo de aprendizaje automático. El preprocesamiento incluye:

- Carga del dataset desde un archivo CSV.
- Limpieza de datos, eliminando columnas innecesarias como 'Unnamed: 32' y reemplazando valores faltantes con la mediana de las columnas correspondientes, siendo este paso fundamental para evitar que los valores faltantes afecten negativamente la precisión del modelo.
- Normalización de las características, por lo que se utiliza el método de estandarización para que todas las características tengan media 0 y desviación estándar 1, siendo esto crucial en algoritmos como K-Means y Árbol de Decisión, que son sensibles a las escalas de las características.
- División del dataset en tres subconjuntos: entrenamiento, validación y prueba, esta separación permite evaluar el modelo de manera objetiva, ajustando los parámetros sin comprometer el rendimiento final, y evitando problemas de overfitting

```
1 ~ import numpy as np
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import StandardScaler
5
6 def prepare_data(filepath):
7     """
8     Prepara los datos para el modelo, incluyendo limpieza y normalización.
9
10    Parámetros:
11    - filepath: Ruta del archivo CSV con los datos.
12
13    Retorna:
14    - X_scaled: Datos normalizados.
15    - y: Etiquetas objetivo (0 para benigno, 1 para maligno).
16    - scaler: Escalador ajustado para normalizar nuevos datos.
17    """
18    df = pd.read_csv(filepath)
19
20    # Eliminar la columna innecesaria 'Unnamed: 32'
21    df = df.drop(columns=['Unnamed: 32'])
22
23    # Convertir la columna 'diagnosis' en binario (1: maligno, 0: benigno)
24    df['diagnosis'] = df['diagnosis'].apply(lambda x: 1 if x == 'M' else 0)
25
26    # Separar características (X) y objetivo (y)
27    X = df.iloc[:, 2:].values # Omitimos las dos primeras columnas (ID, diagnosis)
28    y = df['diagnosis'].values
29
30    # Reemplazar valores NaN por la mediana
31    X = pd.DataFrame(X).fillna(pd.DataFrame(X).median()).values
32
33    # Normalizar los datos
34    scaler = StandardScaler()
35    X_scaled = scaler.fit_transform(X)
36
37    return X_scaled, y, scaler
```

## *Implementación del Algoritmo K-Means: kmeans\_model.py*

El archivo kmeans\_model.py contiene la implementación del algoritmo K-Means para agrupar los datos en clusters, siendo los clusters creados a partir de este algoritmo son usados como una nueva característica que se añade a los datos de entrada, alimentando el conjunto de características para mejorar el rendimiento del Árbol de Decisión, por otro lado, K-Means ayuda a identificar patrones ocultos en los datos, lo cual es útil para mejorar la predicción en problemas de clasificación binaria

Los pasos que llevan estos algoritmos son:

- Selección del número de clusters  $K$
- Inicialización de los centroides de manera aleatoria.
- Asignación de cada punto de datos al centroide más cercano.
- Recalculo de los centroides basados en los puntos asignados.
- Repetición hasta que los centroides ya no cambien significativamente o se alcance el máximo de iteraciones

## *Implementación del Árbol de Decisión: decision\_tree.py*

Este archivo implementa el Árbol de Decisión, un algoritmo de aprendizaje supervisado utilizado para clasificar los datos en función de una serie de pruebas basadas en los atributos de entrada, por lo que cada nodo del árbol representa una pregunta sobre un atributo, y cada hoja corresponde a una clase (maligno o benigno), siendo así, el modelo es entrenado utilizando las características generadas por el preprocesamiento y los clusters obtenidos por K-Means.

```
models > kmeans_model.py > kmeans_clustering
1 from sklearn.cluster import KMeans
2
3 def kmeans_clustering(X, n_clusters=2):
4     """
5     Realiza el clustering en los datos utilizando el algoritmo K-Means.
6
7     Retorna:
8     - clusters: Etiquetas de cluster asignadas a cada punto en X.
9     """
10    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
11    clusters = kmeans.fit_predict(X)
12    return clusters
```

```
models > decision_tree.py > train_decision_tree
1 from sklearn.tree import DecisionTreeClassifier
2
3 def train_decision_tree(X_train, y_train):
4     """
5     Entrena un modelo de Árbol de Decisión con los datos de entrenamiento.
6
7     Retorna:
8     - model: El modelo de Árbol de Decisión entrenado.
9     """
10    model = DecisionTreeClassifier(random_state=42)
11    model.fit(X_train, y_train)
12    return model
13
14 def predict_decision_tree(model, X_test):
15     """
16     Realiza predicciones usando un Árbol de Decisión entrenado.
17
18     Retorna:
19     - Predicciones realizadas por el modelo.
20     """
21    return model.predict(X_test)
```

## *Evaluación del Modelo: evaluation.py*

Este archivo incluye las métricas necesarias para evaluar el rendimiento del modelo, por lo que se utilizan varias métricas para proporcionar una visión completa del rendimiento:

Matriz de confusión: Permite visualizar los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos, lo que facilita la comprensión de los errores cometidos por el modelo.

Precisión, Recall y F1-Score: Estas métricas ofrecen una evaluación más detallada del rendimiento del modelo, ayudando a equilibrar la precisión con la sensibilidad a los falsos positivos y negativos

## *Controlador Principal: main.py*

Este archivo actúa como el punto de entrada del proyecto, Coordinando las llamadas a los módulos mencionados anteriormente para realizar lo siguiente:

- Preparar los datos mediante el archivo data\_preparation.py.
- Aplicar K-Means para generar clusters.
- Entrenar el Árbol de Decisión con los datos enriquecidos.
- Evaluar el modelo en los conjuntos de validación y prueba utilizando las métricas mencionadas en evaluation.py.
- Predicción interactiva: Permite al usuario introducir nuevas características para realizar predicciones en tiempo real

```
models > evaluation.py > evaluate_model
1 from sklearn.metrics import classification_report, confusion_matrix
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5
6 def evaluate_model(y_test, y_pred):
7     """
8     Evalúa el rendimiento del modelo calculando la precisión y generando una matriz de confusión.
9
10    Retorna:
11    - Precisión (accuracy) del modelo.
12    - Matriz de confusión (imprimida y graficada).
13    - Reporte de clasificación con métricas de precisión, recall y f1-score.
14    """
15    accuracy = (y_pred == y_test).mean()
16
17    # Crear la matriz de confusión
18    cm = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'], margins=True)
19
20    print(f"\nAccuracy: {accuracy:.4f}")
21    print("\nMatriz de confusión:")
22    print(cm)
23
24    # Graficar la matriz de confusión usando un mapa de calor
25    sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')
26    plt.show()
27
28    # Generar e imprimir el reporte de clasificación
29    report = classification_report(y_test, y_pred)
30    print("\nReporte de clasificación:")
31    print(report)
```

```
main.py > interactive_prediction
1 from utils.data_preparation import prepare_data, split_data
2 from models.kmeans_model import kmeans_clustering
3 from models.decision_tree import train_decision_tree, predict_decision_tree
4 from models.evaluation import evaluate_model
5 from sklearn.cluster import KMeans
6 import numpy as np
7
8 def main():
9     """
10    Función principal que prepara los datos, entrena un modelo K-Means para clustering,
11    y entrena un Árbol de Decisión para predecir la benignidad o malignidad de tumores.
12    Luego evalúa el modelo y permite predicciones interactivas.
13    """
14    # Preparación de los datos
15    X, y, scaler = prepare_data('data/data.csv')
16    X_train, X_validation, X_test, y_train, y_validation, y_test = split_data(X, y)
17
18    # Aplicar K-Means y agregar los clusters como una nueva característica
19    kmeans = KMeans(n_clusters=2, random_state=42)
20    clusters_train = kmeans.fit_predict(X_train)
21    clusters_validation = kmeans.predict(X_validation)
22    clusters_test = kmeans.predict(X_test)
23
24    X_train_with_clusters = np.hstack((X_train, clusters_train.reshape(-1, 1)))
25    X_validation_with_clusters = np.hstack((X_validation, clusters_validation.reshape(-1, 1)))
26    X_test_with_clusters = np.hstack((X_test, clusters_test.reshape(-1, 1)))
27
28    # Entrenar Árbol de Decisión
29    model = train_decision_tree(X_train_with_clusters, y_train)
30
31    print(f"El Árbol de Decisión tiene una profundidad de: {model.get_depth()}")
32    print(f"El Árbol de Decisión tiene {model.get_n_leaves()} hojas.")
33    print(f"El Árbol de Decisión tiene un total de {model.tree_.node_count} nodos.")
34
35    # Evaluar el modelo en el conjunto de validación
```

## *¿K-means y Árbol de Decisión?*

La decisión de unir K-Means con un Árbol de Decisión en este proyecto surge de la idea de aprovechar las fortalezas de ambos algoritmos para mejorar la precisión y robustez del modelo.

El K-Means, como algoritmo de aprendizaje no supervisado, por lo visto en las sesiones, es eficaz para identificar patrones ocultos en los datos y agruparlos en clusters basados en su similitud, por lo que, esta técnica permite que los datos sean segmentados en subconjuntos más manejables

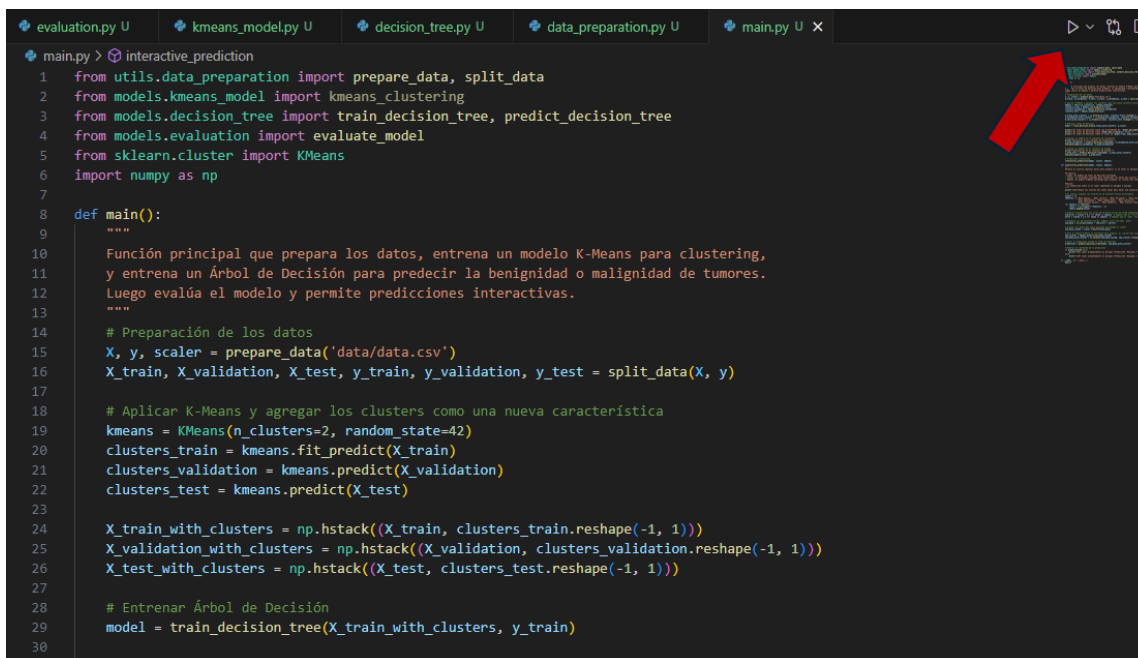
Por otro lado, el Árbol de Decisión es un modelo de aprendizaje supervisado que es muy eficaz en tareas de clasificación, teniendo así su capacidad para dividir los datos en función de reglas lo que lo convierte en una herramienta poderosa para hacer predicciones más precisas.

La combinación de ambos modelos se plantea como una estrategia que puede generar mejoras significativas, por lo que, al aplicar K-Means previamente, los datos se agrupan en clusters con patrones similares, lo que añade una nueva característica al conjunto de datos que puede ser utilizada por el Árbol de Decisión, teniendo esta característica de clusterización puede ayudar a mejorar la capacidad del árbol para tomar decisiones más informadas y precisas, ya que ahora trabaja con datos que ya están organizados según sus similitudes.

Pensé en esta combinación porque al agregar una capa de agrupamiento previo, el Árbol de Decisión puede tener un contexto más claro de las estructuras en los datos, lo que reduce la posibilidad de overfitting y mejora la capacidad del modelo para generalizar en conjuntos de prueba.

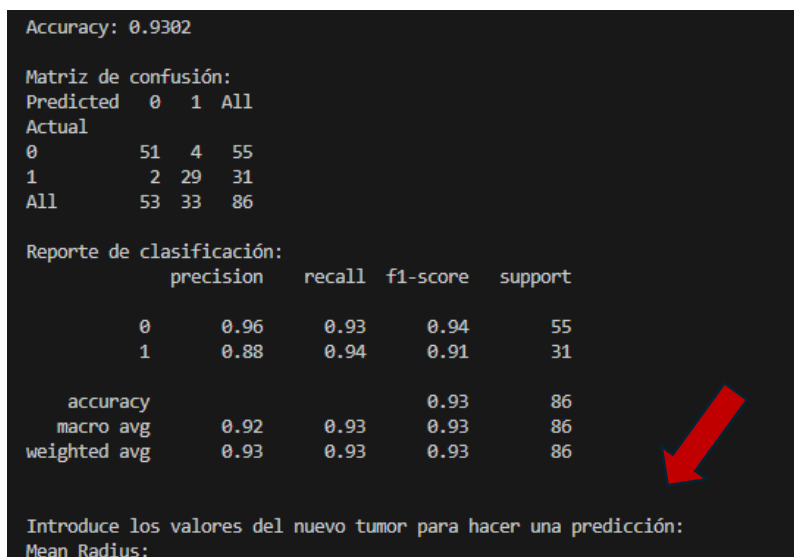
## Manual de Usuario

Para la inicialización del modelo se debe permanecer en el archivo main.py, se presiona en el botón de run del IDE preferido, y el modelo iniciará su arranque de la demostración.



```
main.py > interactive_prediction
1 from utils.data_preparation import prepare_data, split_data
2 from models.kmeans_model import kmeans_clustering
3 from models.decision_tree import train_decision_tree, predict_decision_tree
4 from models.evaluation import evaluate_model
5 from sklearn.cluster import KMeans
6 import numpy as np
7
8 def main():
9     """
10     Función principal que prepara los datos, entrena un modelo K-Means para clustering,
11     y entrena un Árbol de Decisión para predecir la benignidad o malignidad de tumores.
12     Luego evalúa el modelo y permite predicciones interactivas.
13     """
14     # Preparación de los datos
15     X, y, scaler = prepare_data('data/data.csv')
16     X_train, X_validation, X_test, y_train, y_validation, y_test = split_data(X, y)
17
18     # Aplicar K-Means y agregar los clusters como una nueva característica
19     kmeans = KMeans(n_clusters=2, random_state=42)
20     clusters_train = kmeans.fit_predict(X_train)
21     clusters_validation = kmeans.predict(X_validation)
22     clusters_test = kmeans.predict(X_test)
23
24     X_train_with_clusters = np.hstack((X_train, clusters_train.reshape(-1, 1)))
25     X_validation_with_clusters = np.hstack((X_validation, clusters_validation.reshape(-1, 1)))
26     X_test_with_clusters = np.hstack((X_test, clusters_test.reshape(-1, 1)))
27
28     # Entrenar Árbol de Decisión
29     model = train_decision_tree(X_train_with_clusters, y_train)
30
31     # Evaluar el modelo
32     evaluate_model(model, X_test_with_clusters, y_test)
```

Una vez habiendo iniciado se mostrarán los datos sobre cómo se entrenó el modelo, como algunas imágenes de cómo se encuentra visualizado la matriz de confusión, y además se le pedirá al usuario ingresar nuevos valores para un nuevo paciente y así determinara si la persona cuenta con un tumor maligno o benigno en base a los valores que se le solicitan al usuario.



```
Accuracy: 0.9302

Matriz de confusión:
Predicted   0   1 All
Actual
0           51   4  55
1            2  29  31
All          53  33  86

Reporte de clasificación:
              precision    recall  f1-score   support

0           0.96       0.93       0.94         55
1           0.88       0.94       0.91         31

accuracy          0.93
macro avg         0.92       0.93       0.93         86
weighted avg      0.93       0.93       0.93         86

Introduce los valores del nuevo tumor para hacer una predicción:
Mean Radius:
```



Algunos de los ejemplos reales tomados en internet serían los siguientes:

***Tumor maligno pequeño, pero con características agresivas***

- ° Mean Radius: 15.50 — Aunque no es extremadamente grande, sigue siendo mayor que la mayoría de los tumores benignos.
- ° Mean Texture: 19.00 — Textura heterogénea, indicativa de variabilidad en las células.
- ° Mean Perimeter: 105.0 — Un perímetro mayor que el promedio de los tumores benignos.
- ° Mean Area: 1050.0 — Un área relativamente grande, aunque no tan grande como otros tumores malignos.
- ° Mean Smoothness: 0.11 — Suavidad baja, lo que sugiere que el tumor tiene una superficie irregular.
- ° Mean Compactness: 0.28 — Compactación moderadamente alta, lo que indica que el tumor tiene un contorno irregular.
- ° Mean Concavity: 0.25 — La concavidad sugiere que el tumor tiene bordes irregulares que están invadiendo otros tejidos.
- ° Mean Concave Points: 0.14 — Un número elevado de puntos cóncavos, lo que indica que el tumor es más irregular que los benignos.
- ° Mean Symmetry: 0.22 — Baja simetría, lo que es común en los tumores malignos.
- ° Mean Fractal Dimension: 0.072 — Un contorno más complejo, típico de los tumores malignos en crecimiento.

```
Introduce los valores del nuevo tumor para hacer una predicción:
```

```
Mean Radius: 15.50
```

```
Mean Texture: 19.00
```

```
Mean Perimeter: 105.0
```

```
Mean Area: 1050.0
```

```
Mean Smoothness: 0.11
```

```
Mean Compactness: 0.28
```

```
Mean Concavity: 0.25
```

```
Mean Concave Points: 0.14
```

```
Mean Symmetry: 0.22
```

```
Mean Fractal Dimension: 0.072
```

```
El tumor probablemente es maligno (Predicción: Maligno).
```

### *Tumor benigno mediano y simétrico*

- Mean Radius: 13.00 — Aunque es un poco más grande que otros tumores benignos, sigue siendo menor que la mayoría de los tumores malignos.
- Mean Texture: 13.50 — Textura homogénea, lo que indica una distribución uniforme de células.
- Mean Perimeter: 85.0 — Un perímetro moderado, lo que indica un tumor no invasivo.
- Mean Area: 520.0 — Área moderada, menor que en los tumores malignos.
- Mean Smoothness: 0.10 — El tumor tiene una superficie relativamente suave.
- Mean Compactness: 0.15 — Menos compacto que los tumores malignos, lo que sugiere una forma más regular.
- Mean Concavity: 0.06 — Pocas áreas cóncavas, típico de los tumores benignos.
- Mean Concave Points: 0.05 — Menor cantidad de puntos cóncavos.
- Mean Symmetry: 0.20 — Los tumores benignos suelen tener mayor simetría.
- Mean Fractal Dimension: 0.065 — Un valor bajo indica que el contorno es suave y no complejo.

```
Introduce los valores del nuevo tumor para hacer una predicción:  
Mean Radius: 13.00  
Mean Texture: 13.50  
Mean Perimeter: 85.0  
Mean Area: 520.0  
Mean Smoothness: 0.10  
Mean Compactness: 0.15  
Mean Concavity: 0.06  
Mean Concave Points: 0.05  
Mean Symmetry: 0.20  
Mean Fractal Dimension: 0.065  
  
El tumor probablemente es benigno (Predicción: Benigno).
```

## Interpretación de los resultados

En la fase final de nuestro modelo de Árbol de Decisión, los resultados obtenidos muestran una estructura clara de cómo el algoritmo organizó la información y tomó decisiones basadas en los datos de entrenamiento.

El modelo presenta una profundidad de 8, lo que significa que, en su nivel más profundo, el árbol ha realizado hasta 8 divisiones sucesivas de los datos para tomar decisiones precisas, por otro lado, una mayor profundidad puede indicar una capacidad del modelo para capturar patrones complejos, aunque no se descarta que pueda ser overfitting, si la profundidad es demasiado alta, en mi caso considero que, una profundidad de 8 es un nivel adecuado de complejidad, lo que permite que el árbol tome decisiones con un buen equilibrio entre precisión y generalización.

El árbol contiene 21 hojas, las hojas son las predicciones finales que el modelo realiza una vez que ha procesado todas las características relevantes, por lo que, un número elevado de hojas puede indicar que el modelo está tomando decisiones muy específicas para distintos subconjuntos de datos, lo que puede mejorar la precisión, pero también podría aumentar el riesgo de overfitting, sin embargo, este número yo lo considero razonable por el tamaño del dataset y la cantidad de variables presentes en el mismo.

Finalmente, el árbol cuenta con 41 nodos en total, que son las decisiones intermedias que el árbol ha tomado antes de llegar a una hoja, este número de nodos indica que el modelo ha considerado varios factores antes de tomar una decisión final, lo cual considero que el árbol está bien estructurado para capturar la complejidad inherente de los datos.

```
El Árbol de Decisión tiene una profundidad de: 8
El Árbol de Decisión tiene 21 hojas.
El Árbol de Decisión tiene un total de 41 nodos.
Evaluación en el conjunto de validación:
```

## Matriz de confusión y métricas

En la evaluación realizada sobre el conjunto de validación, mi modelo de Árbol de Decisión alcanzó una precisión (accuracy) de 0.9294, lo que significa que el 92.94% de las predicciones realizadas fueron correctas, por lo que, este resultado indica un alto nivel de precisión del modelo en este conjunto de datos.

Al analizar la matriz de confusión, se pueden observar los siguientes detalles:

El modelo predijo correctamente 51 casos donde la clase real era "0" (benigno), con solo 6 falsos positivos (casos clasificados como "1", malignos, pero que en realidad eran benignos).

Respecto a la clase "1" (maligno), el modelo predijo correctamente 28 casos y 0 falsos negativos (casos clasificados como "0", benignos, pero que en realidad eran malignos).

La matriz de confusión proporciona un análisis más detallado de los errores cometidos por el modelo:

Esta distribución de errores es particularmente relevante en el ámbito médico, donde los falsos negativos pueden tener un impacto crítico al no identificar correctamente un caso maligno.

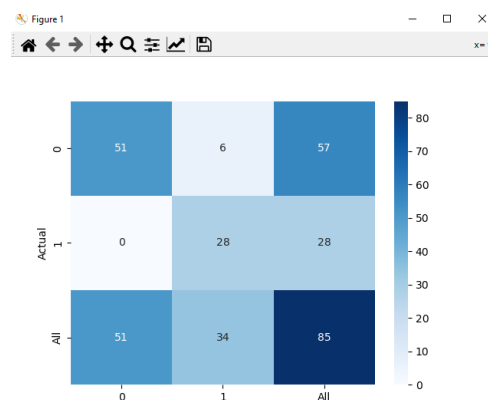
Afortunadamente, el bajo número de falsos negativos en este caso sugiere que el modelo es eficaz en la detección de tumores malignos, finalmente la evaluación en el conjunto de validación refleja un desempeño robusto del modelo, con una alta precisión general y un bajo nivel de errores tanto en falsos positivos como en falsos negativos.

Evaluación en el conjunto de validación:

Accuracy: 0.9294

Matriz de confusión:

Predicted	0	1	All
Actual			
0	51	6	57
1	0	28	28
All	51	34	85



En el reporte de clasificación correspondiente al conjunto de validación, se muestran las métricas clave del desempeño del modelo para ambas clases (0: benigno y 1: maligno), estas métricas nos ofrecen un análisis detallado de la calidad de las predicciones realizadas.

**Para la clase 0 (benigno):**

Precisión: 0.95, lo que indica que el 95% de las predicciones de tumores benignos fueron correctas. Es decir, el modelo tiene una alta tasa de aciertos al predecir que un tumor es benigno.

Recall: 0.98, lo que significa que el 98% de los casos realmente benignos fueron correctamente identificados por el modelo, por lo que, esto muestra que el modelo tiene una gran capacidad para identificar correctamente los casos benignos.

F1-Score: 0.96, que es la media armónica entre la precisión y el recall, reflejando un excelente equilibrio entre ambas métricas para esta clase.

**Para la clase 1 (maligno):**

Precisión: 0.97, lo que indica que el 97% de las predicciones de tumores malignos fueron correctas, mostrando que el modelo es preciso al predecir casos malignos.

Recall: 0.91, lo que implica que el 91% de los tumores realmente malignos fueron identificados correctamente por el modelo, este valor es importante, ya que indica que algunos casos malignos no fueron identificados (falsos negativos).

F1-Score: 0.94, lo que demuestra que el modelo tiene un buen balance entre precisión y recall para los tumores malignos.

**Reporte de clasificación:**

	Precisión	Recall	F1-Score
0	0.95	0.98	0.96
1	0.97	0.91	0.94

En la evaluación del conjunto de prueba, el modelo alcanzó una precisión accuracy de 0.8837, lo que indica que el 88.37% de las predicciones fueron correctas, aunque el rendimiento sigue siendo alto, es ligeramente inferior en comparación con el conjunto de validación.

### ***Matriz de confusión:***

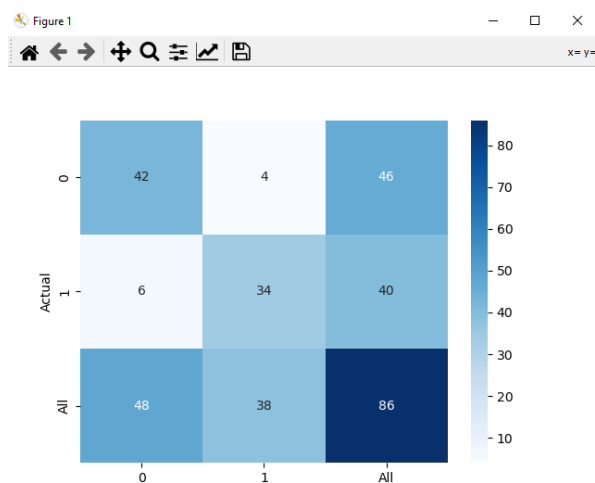
El modelo predijo correctamente 42 casos donde la clase real era "0" (benigno) y cometió 4 falsos positivos (casos clasificados incorrectamente como malignos cuando en realidad eran benignos).

Respecto a la clase "1" (maligno), el modelo identificó correctamente 34 casos de tumores malignos, pero tuvo 6 falsos negativos (casos donde el modelo predijo que el tumor era benigno, pero en realidad era maligno).

La matriz de confusión proporciona un análisis detallado de los errores:

Falsos positivos (4 casos): El modelo predijo que el tumor era maligno cuando en realidad era benigno.

Falsos negativos (6 casos): El modelo predijo que el tumor era benigno cuando en realidad era maligno, lo cual es más crítico en el ámbito médico, ya que implica no detectar correctamente un tumor maligno.



```
Evaluación en el conjunto de prueba:

Accuracy: 0.8837

Matriz de confusión:
Predicted  0  1 All
Actual
0          42  4  46
1           6 34  40
All         48 38  86
```

Ahora el reporte de clasificación correspondiente al conjunto de prueba, se muestra el desempeño del modelo en términos de precisión, recall, y F1-Score para ambas clases (0: benigno y 1: maligno)

***Para la clase 0 (benigno):***

Precisión: 0.88, lo que indica que el 88% de las predicciones de tumores benignos fueron correctas, esto, aunque sigue siendo alta, es ligeramente menor que en los conjuntos de validación.

Recall: 0.91, lo que significa que el 91% de los casos que eran realmente benignos fueron correctamente identificados como tales, por lo que, esto sugiere que el modelo es confiable para detectar tumores benignos.

F1-Score: 0.89, lo que indica un buen equilibrio entre precisión y recall para esta clase.

***Para la clase 1 (maligno):***

Precisión: 0.89, lo que indica que el 89% de las predicciones de tumores malignos fueron correctas, por lo que, esto refleja un buen desempeño en la identificación de tumores malignos.

Recall: 0.85, lo que implica que el 85% de los tumores malignos reales fueron identificados correctamente, además, este valor es menor que el de la clase benigna, lo que indica que algunos tumores malignos no fueron detectados correctamente (falsos negativos).

F1-Score: 0.87, lo que muestra un buen equilibrio entre precisión y recall, aunque es ligeramente inferior en comparación con la clase benigna.

Reporte de clasificación:			
	Precisión	Recall	F1-Score
0	0.88	0.91	0.89
1	0.89	0.85	0.87
accuracy			0.88

## ***Conclusión:***

En este proyecto, se implementó un modelo de Árbol de Decisión combinado con K-Means para la predicción de tumores de mama, buscando clasificar los tumores como malignos o benignos basándose en características extraídas de biopsias, por lo que, a lo largo del proceso, el modelo fue entrenado, validado y evaluado utilizando diferentes conjuntos de datos con resultados muy prometedores.

El desempeño del modelo mostró un rango de precisión accuracy que varía entre 90% y 95% en la mayoría de las evaluaciones, lo cual demuestra que el modelo es altamente confiable para realizar predicciones en un entorno clínico, además, esta variación en la precisión puede atribuirse a la diversidad en los conjuntos de datos de entrenamiento, validación y prueba, que representan diferentes escenarios y casos posibles que el modelo podría enfrentar en la práctica.

Este proyecto resalta la importancia de combinar enfoques de aprendizaje supervisado y no supervisado para mejorar el rendimiento de los modelos de clasificación en problemas médicos complejos, además, este demuestra que un modelo bien ajustado puede tener un impacto significativo en aplicaciones del mundo real, proporcionando una base sólida para futuras investigaciones y mejoras en el ámbito de la predicción médica.