

Basile Álvarez Andrés José

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 30/11/21

Inteligencia Artificial

Grupo III

Semestre 2022-1

Reporte Práctica 12: Clasificación con Árboles de Decisión

Objetivo: Clasificar registros clínicos de tumores malignos y benignos de cancer de mama a partir de imágenes digitalizadas.

Fuente de datos: Estudios clínicos a partir de imágenes digitalizadas de pacientes con cáncer de mama de Wisconsin (WDBC, Wisconsin Diagnostic Breast Cancer), donde:

- ID number: Identifica al paciente (valor discreto).
- Diagnosis: Diagnóstico (M=maligno, B=benigno).
- Radius: Media de las distancias del centro y puntos del perímetro.
- Texture: Desvaición estándar de la escala de grises.
- Perimeter: Valor del perímetro del cáncer de mama.
- Area: Valor del área del cáncer de mama.
- Smoothness: Variación de la longitud del radio.
- Compactness: $\text{Perímetro}^2 / \text{Area} - 1$
- Concavity: Caída o gravedad de las curvas de nivel.
- Concave Points: Número de sectores de contorno cóncavo.
- Symmetry: Simetría de la imagen
- Fractal dimensión: Aproximación de frontera – 1.

[https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

Características Generales:

En esta práctica, utilizaremos la clasificación por árboles de decisión, en donde se calcula la entropía como medida de incertidumbre para después calcular la ganancia de información y así poder seleccionar el mejor atributo e iterar hasta que todos los elementos en un conjunto de datos sean clasificados.

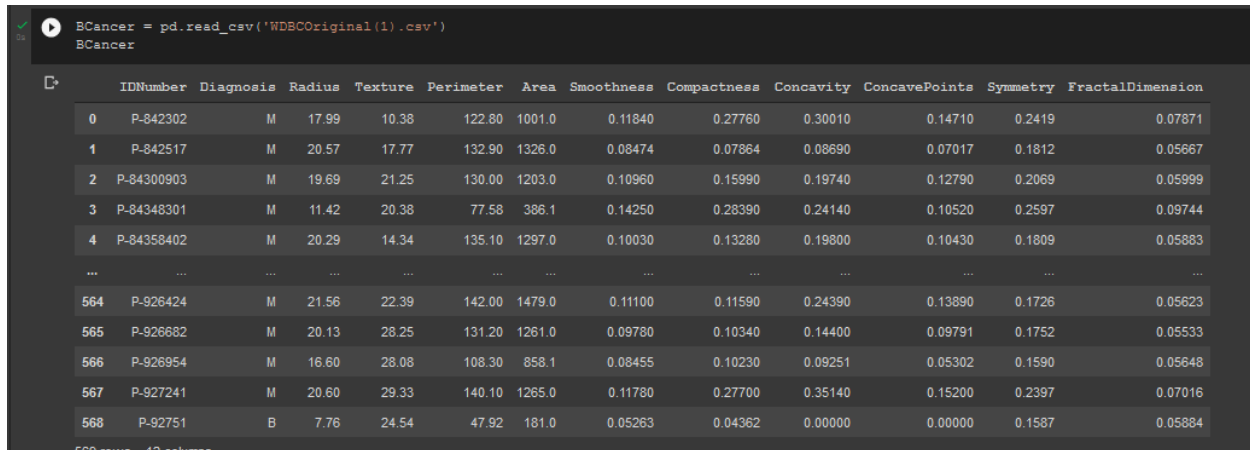
Los árboles de decisión para clasificar se utilizan cuando la variable respuesta tiene 2 o más categorías, utilizando un árbol de clasificación estándar o de tipo C4.

Desarrollo

Primeramente, tenemos que definir aquellas bibliotecas de Python que nos serán útiles para importar, limpiar y analizar los datos contenidos en el archivo separado por comas

WDBCOriginal.csv. Estas serán: *pandas* (manipulación y análisis de datos), *matplotlib* (para la creación de gráficas y visualización de los datos), *numpy* para utilizar vectores y matrices de n dimensiones.

Más tarde, importamos el archivo *WDBCOriginal.csv* y lo primero que hacemos es guardarlo en un DataFrame. La importación del archivo se realizó a partir del explorador de archivos que abrimos utilizando el comando *files.upload()*. Una vez importados los datos, mostramos el DataFrame que los contiene:



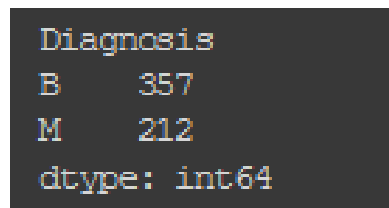
```
Bcancer = pd.read_csv('WDBCOriginal(1).csv')
Bcancer
```

	IDNumber	Diagnosis	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension
0	P-842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871
1	P-842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667
2	P-84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999
3	P-84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744
4	P-84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883
...
564	P-926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623
565	P-926882	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533
566	P-926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648
567	P-927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016
568	P-92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884

569 rows x 12 columns

Figura 1: Data Frame con algunos de los datos de tumores de mama.

Una vez hecho lo anterior, utilizamos la función *groupby* para conocer cuántos datos tenemos en cada una de las dos clasificaciones para los tumores (benigno o maligno):



```
Diagnosis
B      357
M      212
dtype: int64
```

Figura 2: *Groupby* por diagnóstico del Data Frame con los datos de tumores de mama.

Una vez hecho lo anterior, utilizamos una matriz de correlaciones con el propósito de seleccionar variables significativas en el conjunto de datos y reducir la dimensionalidad de este. En la figura 3 se muestra la matriz de correlación de Pearson, utilizando colores para distinguir las variables que tienen una mayor dependencia y así poder realizar la selección de variables.

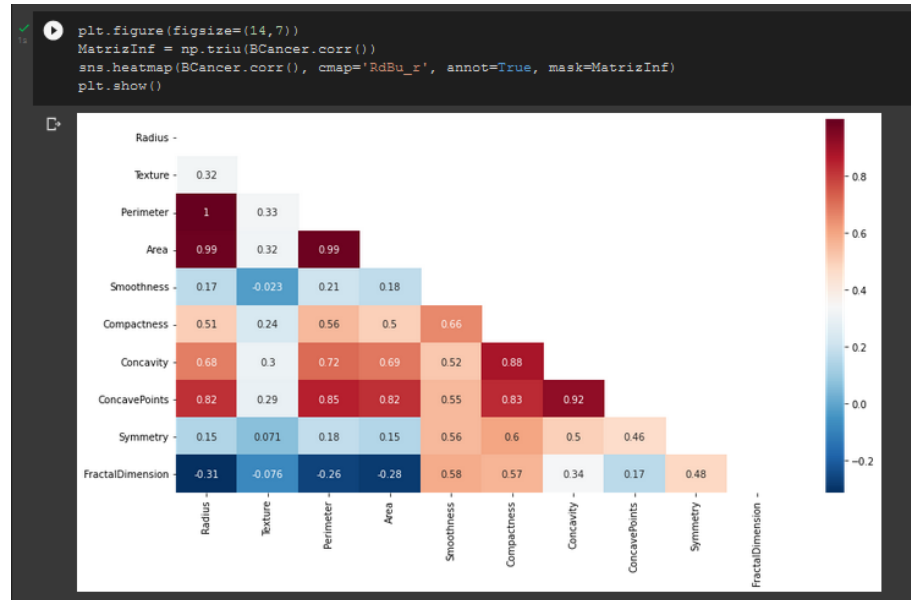


Figura 3: Matriz con las correlaciones entre variables.

Variables seleccionadas:

- 1) Textura [Posición 3]
- 2) Area [Posición 5]
- 3) Smoothness [Posición 6]
- 4) Compactness [Posición 7]
- 5) Symmetry [Posición 10]
- 6) FractalDimension [Posición 11]

Figura 4: Variables seleccionadas

Definimos entonces un *array* con las variables predictoras que nos permitirán pronosticar el valor del diagnóstico en la clasificación, como se muestra en la figura 5.

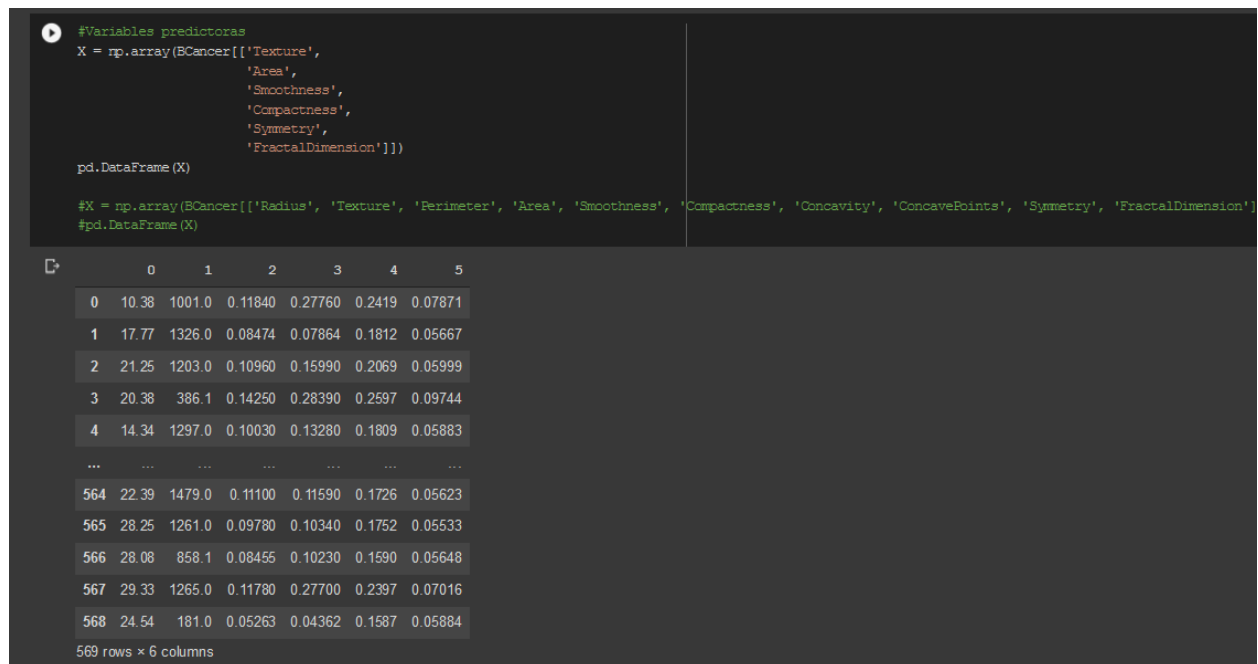


Figura 5: Parte de la matriz con las variables predictoras.

Además, definimos un vector con la variable clase o variable a predecir *Diagnosis*, como se muestra en la figura 6.

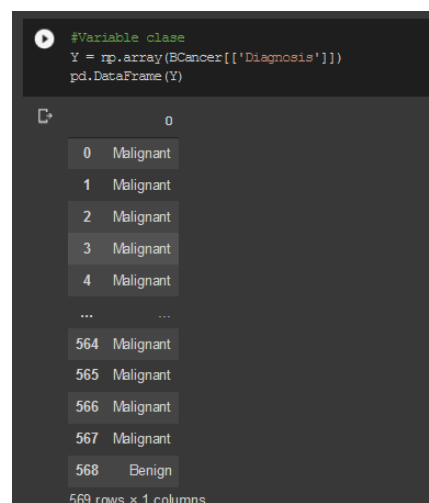


Figura 6: Parte de la matriz con la variable clase.

Para poder aplicar el algoritmo, debemos importar las funciones *DecisionTreeClassifier* de *sklearn.tree*, *classification_report*, *confusion_matrix* y *accuracy_score* de *sklearn.metrics* y *model_selection* de *sklearn*.

Realizamos la división de los datos para determinar el tamaño del conjunto de pruebas y del conjunto de entrenamiento, de manera similar a lo que se realizó en prácticas pasadas:

```
[ ] X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y,
                                                                                   test_size = 0.2,
                                                                                   random_state = 0,
                                                                                   shuffle = True)
```

pd.DataFrame(X_train)

	0	1	2	3	4	5
0	17.53	310.8	0.10070	0.07326	0.1890	0.06331
1	21.98	359.9	0.08801	0.05743	0.2016	0.05977
2	14.86	800.0	0.09495	0.08501	0.1735	0.05875
3	17.84	451.1	0.10450	0.07057	0.1900	0.06635
4	22.44	466.5	0.08192	0.05200	0.1544	0.05976
...
450	19.98	1102.0	0.08923	0.05884	0.1550	0.04996
451	24.04	475.9	0.11860	0.23960	0.2030	0.08243
452	18.32	278.6	0.10090	0.05956	0.1506	0.06959
453	18.22	288.1	0.06950	0.02344	0.1653	0.06447
454	23.93	403.5	0.09261	0.10210	0.1388	0.06570

455 rows × 6 columns

pd.DataFrame(Y_train)

	0
0	Benign
1	Benign
2	Benign
3	Benign
4	Benign
...	...
450	Malignant
451	Malignant
452	Benign
453	Benign
454	Benign

455 rows × 1 columns

Figura 7: División de los datos.

Una vez hecho lo anterior, se entrena el modelo a través de un árbol de decisión. El objeto para generar el modelo de tipo regresor se define como *DecisionTreeClassifier()* y una vez definido,

podemos entrenar el modelo para obtener el árbol utilizando la función *fit()*. Generamos la clasificación:

```
#Se etiquetan las clasificaciones
Y_Clasificacion = ClasificacionAD.predict(X_validation)
pd.DataFrame(Y_Clasificacion)
```

	0
0	Malignant
1	Benign
2	Benign
3	Benign
4	Benign
...	...
109	Benign
110	Benign
111	Malignant
112	Malignant
113	Malignant

114 rows × 1 columns

Figura 8: Generamos las clasificaciones.

Y, una vez generado, comparamos el valor real con el valor clasificado, obteniendo que:

```
Valores = pd.DataFrame(Y_validation, Y_Clasificacion)
Valores
```

	0
Malignant	Malignant
Benign	Benign
Benign	Benign
Benign	Benign
Benign	Benign
...	...
Benign	Malignant
Benign	Benign
Malignant	Malignant
Malignant	Malignant
Malignant	Benign

114 rows × 1 columns

Figura 9: Comparación del valor pronosticado y el valor real.

Generamos la matriz de clasificación para observar los elementos que fueron clasificados de manera correcta y aquellos que no. Además, generamos el reporte de la clasificación; observando que se tuvo una exactitud del 91.228%. Vemos también que obtuvimos una precisión del 93% para la clasificación de tumores benignos y un 89% para la clasificación de tumores malignos.

```
#Matriz de clasificación
Y_Clasificacion = ClasificacionAD.predict(X_validation)
Matriz_Clasificacion = pd.crosstab(Y_validation.ravel(),
                                   Y_Clasificacion,
                                   rownames=['Real'],
                                   colnames=['Clasificación'])

Matriz_Clasificacion
```

	Benign	Malignant
Benign	62	5
Malignant	5	42

```
[20] #Reporte de la clasificación
print('Criterio: \n', ClasificacionAD.criterion)
print('Importancia variables: \n', ClasificacionAD.feature_importances_)
print("Exactitud", ClasificacionAD.score(X_validation, Y_validation))
print(classification_report(Y_validation, Y_Clasificacion))

#describir reporte, cómo se está comportando el modelo.
```

```
Criterio:
gini
Importancia variables:
[0.08121515 0.69406154 0.04198305 0.15624813 0.02015288 0.00633925]
Exactitud 0.9122807017543859
```

	precision	recall	f1-score	support
Benign	0.93	0.93	0.93	67
Malignant	0.89	0.89	0.89	47
accuracy			0.91	114
macro avg	0.91	0.91	0.91	114
weighted avg	0.91	0.91	0.91	114

Figura 10: Reporte del modelo de clasificación

Luego, se obtuvo la importancia de cada una de las variables para nuestro modelo utilizando la función *feature_importance*, obteniendo que el área fue la variable con mayor importancia para nuestra clasificación con un 69%, mientras que la dimensión fractal fue la menos importante, con un porcentaje del 0.6%.

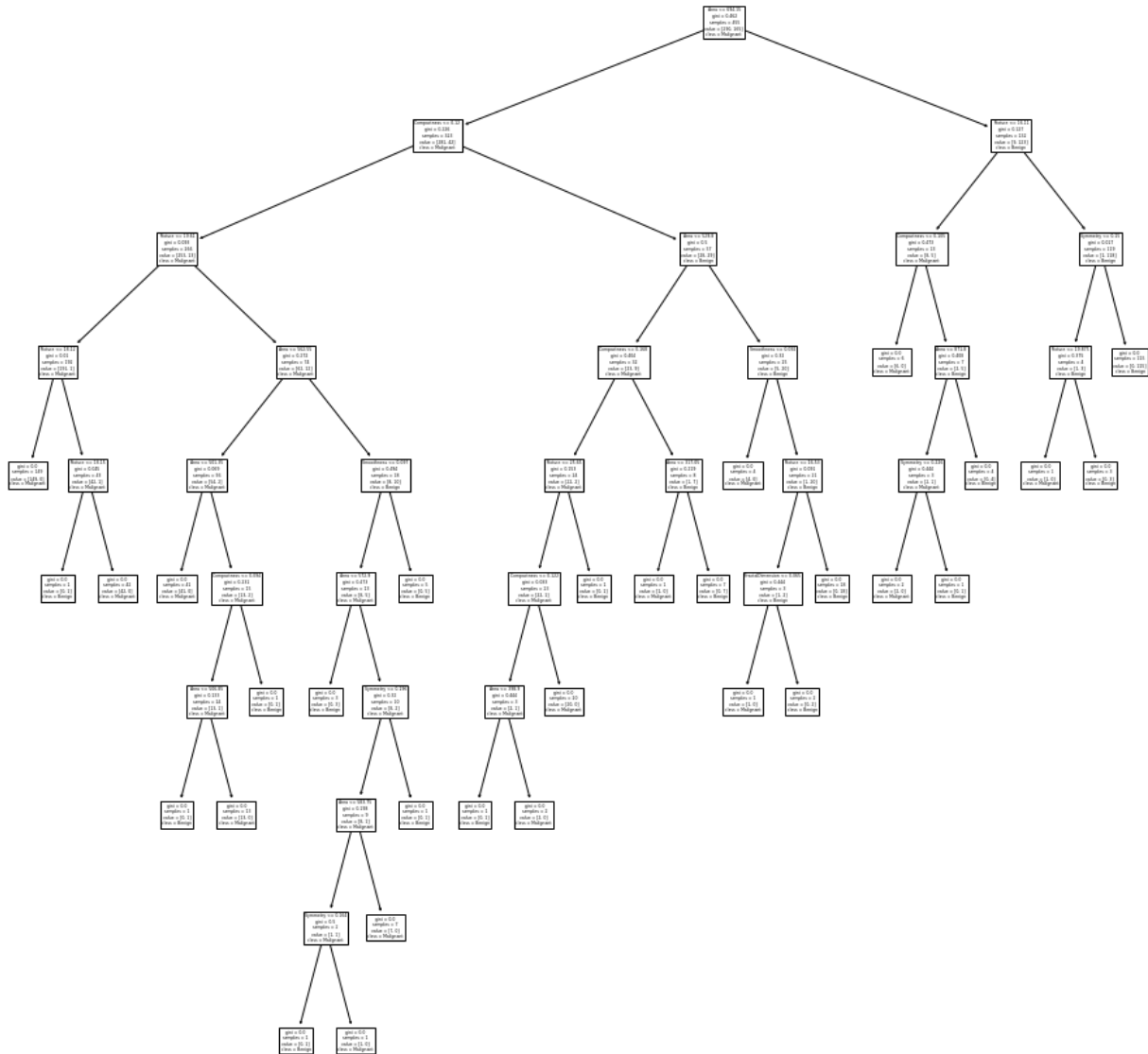


Figura 11: Árbol generado a partir de nuestro modelo de clasificación, donde vemos que los nodos hoja tienen más de un elemento en su mayoría, lo cual representa un ajuste correcto.

Más tarde, importamos *export_text* para visualizar el árbol recién creado en forma de texto, observando cada uno de los elementos que lo conforman. Además, como ya contamos con el modelo hecho, podemos ahora hacer nuevas clasificaciones únicamente ingresando los valores de las variables que seleccionamos para predecir. De esta forma, por ejemplo, para valores de 'Texture': [10.38], 'Area': [1001.0], 'Smoothness': [0.11840], 'Compactness': [0.27760], 'Symmetry': [0.2419], 'FractalDimension': [0.07871], obtenemos que el modelo lo clasifica como tumor maligno. Mientras que, para valores de 'Texture': [24.54], 'Area': [181.0], 'Smoothness': [0.05263], 'Compactness': [0.04362], 'Symmetry': [0.1587], 'FractalDimension': [0.05884], el modelo lo clasifica como tumor benigno (probablemente porque cuenta con un área mucho menor que el tumor del caso anterior).

Y justamente eso es lo más importante de realizar este tipo de modelos: que su clasificación pueda ser utilizada por un usuario (médico) para ingresar valores de los tumores de los pacientes y obtener un posible diagnóstico o una especie de “segunda opinión” sobre cómo se podría clasificar el tumor.

Además, se realizó una prueba con un número diferente de elementos en cada nodo hoja (en este caso un número mínimo de 2) y al cambiar la profundidad máxima del árbol a 8. Al hacer esto, cambió la exactitud a un valor menor (90.35%) y obtuvimos una precisión en la clasificación de tumores benignos del 94% y del 86% para tumores malignos.

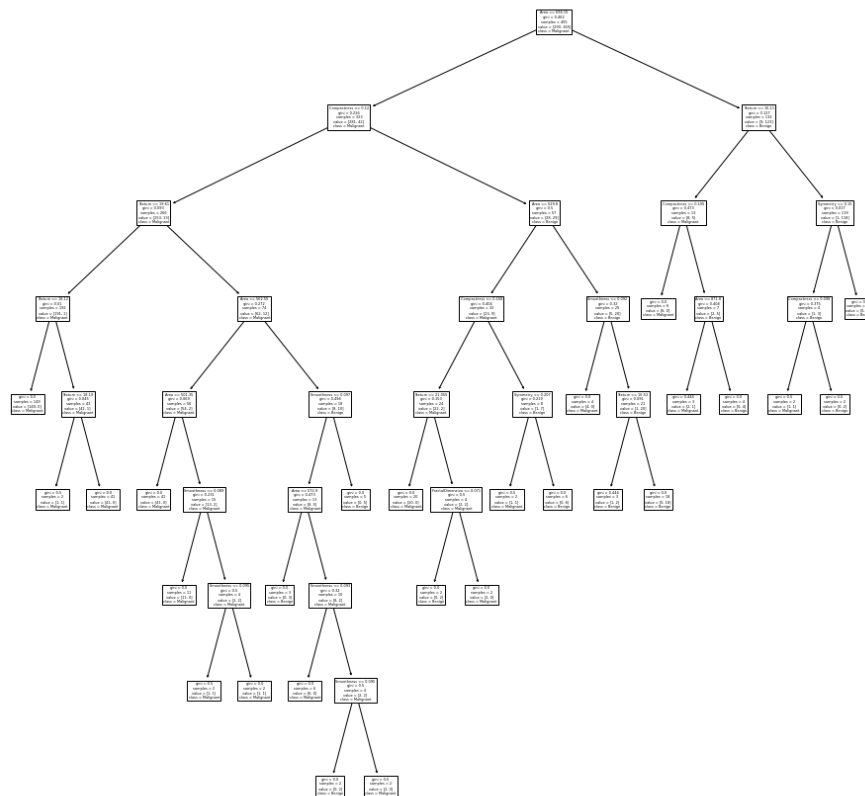


Figura 12: Árbol generado a partir de nuestro modelo de clasificación, variando el número mínimo de elementos en las hojas y el número máximo de profundidad del árbol.

Conclusiones

A lo largo de esta práctica, tuvimos como principal objetivo el obtener clasificaciones de si un tumor es maligno o benigno utilizando árboles de decisión. Para ello, utilizamos los árboles de decisión que se encuentran entre las funciones de Python, teniendo especial cuidado en obtener modelos con el ajuste adecuado, probando para ello distintos valores de *max_depth*, *min_samples_split* y de *min_samples_leaf*, con el objetivo de obtener el mejor ajuste posible.

En los árboles de decisión construidos, observamos que el primero de ellos fue el que contó con una mayor exactitud y mejores parámetros en general, por lo que ese sería el modelo que utilizaría para clasificar valores de área en los tumores.

En general, creo que se cumplieron los objetivos de la práctica y que ésta fue útil para comprender de mejor manera un ejemplo de aprendizaje supervisado sencillo y las posibles aplicaciones de este tipo de algoritmos.