

Basile Álvarez Andrés José

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 25/11/21

Inteligencia Artificial

Grupo III

Semestre 2022-1

## Reporte Práctica 11: Pronóstico con Árboles de Decisión

Objetivo: Realizar el pronóstico de si un tumor es maligno o benigno utilizando árboles de decisión.

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: Desviación 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:  $\text{Perímetro}^2 / \text{Área} - 1$
- 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 regresión logística para predecir etiquetas que nos permitan conocer si un tumor es benigno o maligno. Para ello, construiremos un modelo a través de un conjunto de entrenamiento y utilizaremos un conjunto de prueba para evaluar dicho modelo.

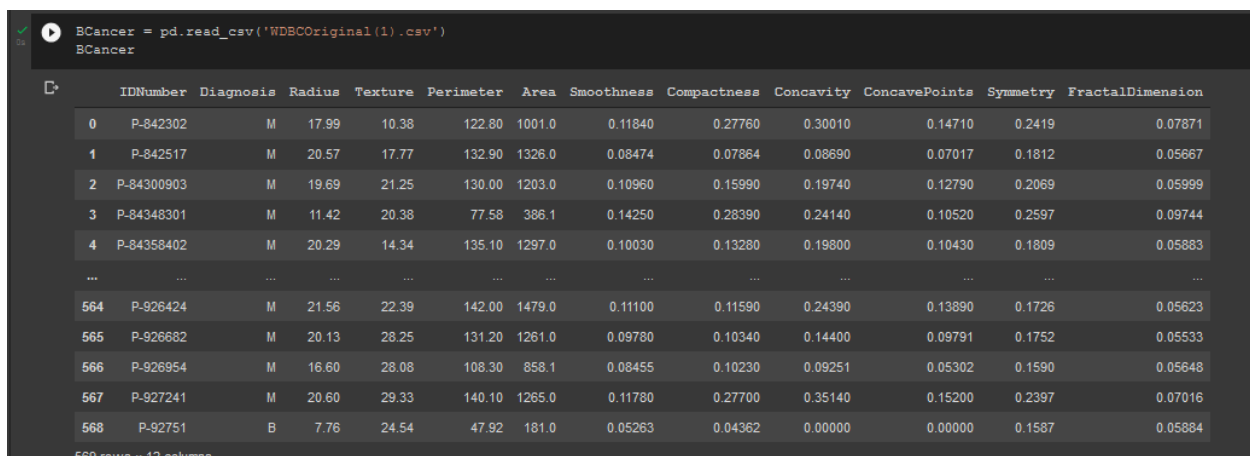
La regresión logística es un tipo de aprendizaje supervisado cuyo objetivo es predecir valores binarios (0 o 1). Para este tipo de aprendizaje, se utiliza la función logística o sigmoide, la cual se muestra a continuación:

$$\text{Función logística} = \frac{1}{1 + e^{-(a+bX)}}$$

## 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 *RGeofisicos.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 *WDBCOoriginal.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:



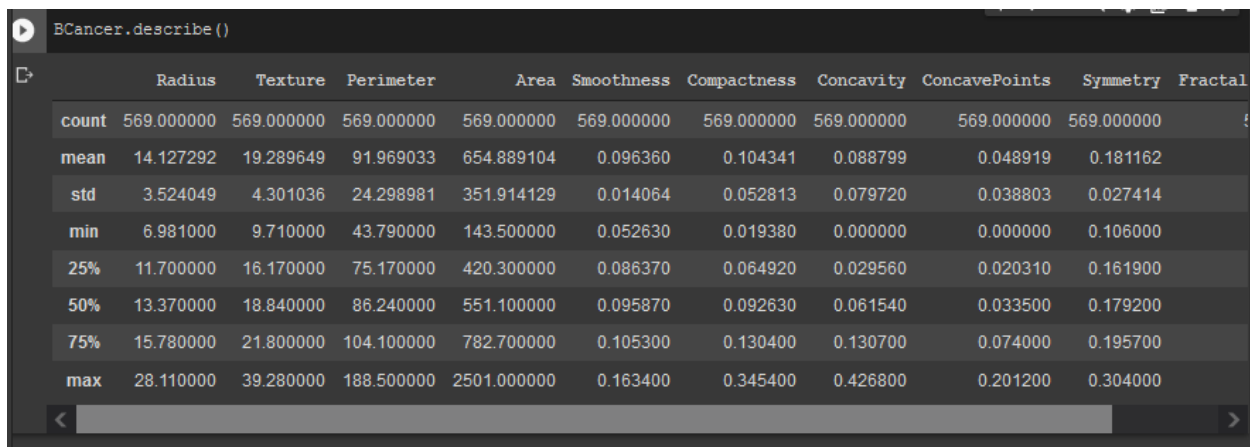
```
Bcancer = pd.read_csv('WDBCOoriginal(1).csv')
Bcancer
```

	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-926682	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.

Hecho lo anterior, realizamos una descripción de los datos utilizando la función *describe()*, esto es importante debido a que una de las formas de dividir un árbol de decisión es utilizando promedios, los cuales se muestran dentro de la descripción de los datos. En la figura 2, se observa la descripción de los datos.



```
Bcancer.describe()
```

	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	Fractal
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	!
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	

Figura 2: Descripción del Data Frame con los datos de tumores de mama.

También realizamos una gráfica del área del tumor por paciente para visualizar los datos.

## 2) Gráfica del área del tumor por paciente

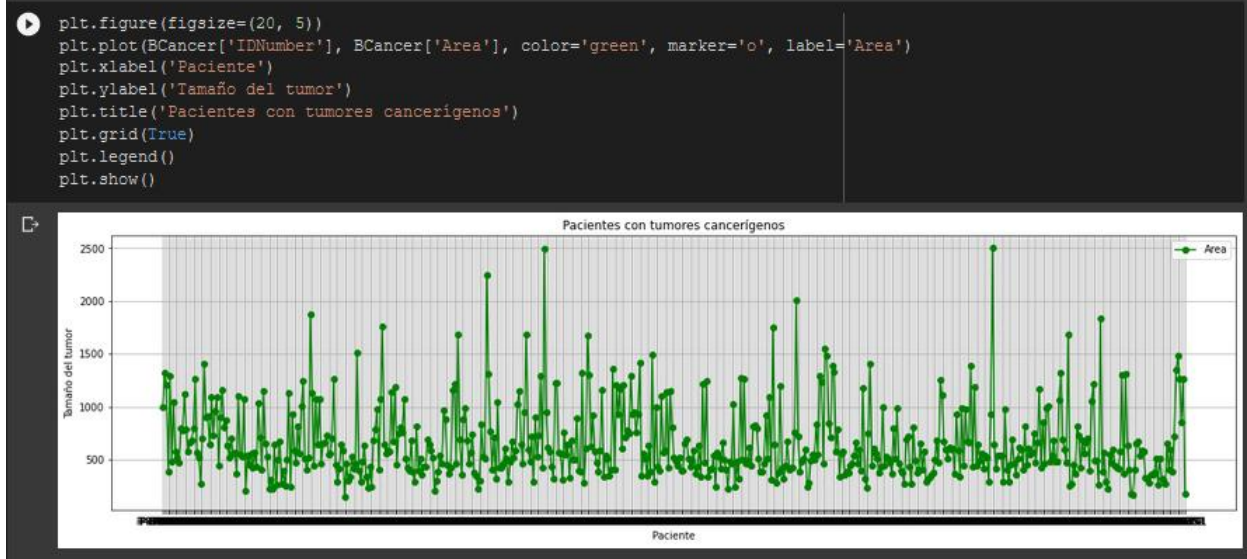


Figura 3: Gráfica del área de los tumores de mama.

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.

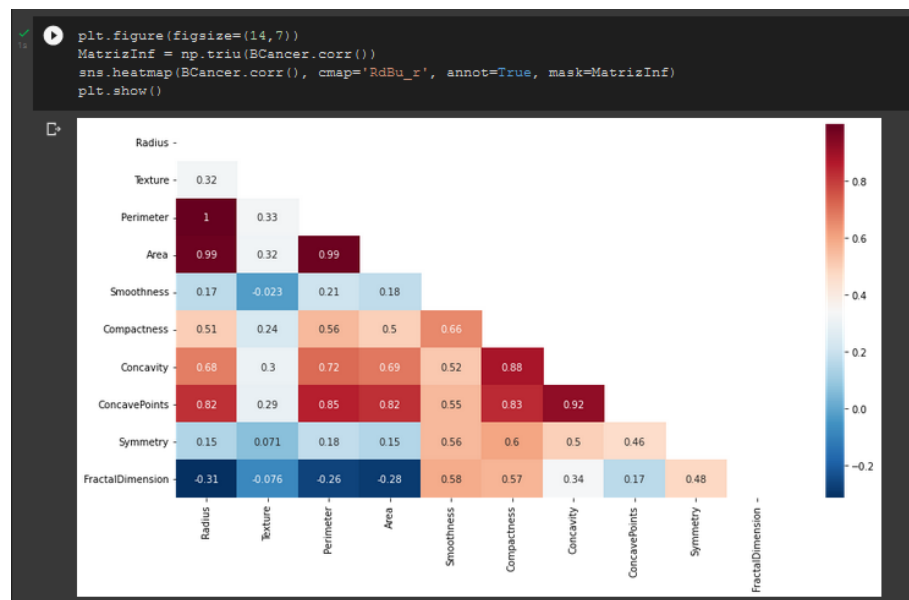


Figura 4: Matriz con las correlaciones entre variables.

Para poder aplicar el algoritmo, debemos importar las funciones *DecisionTreeRegressor* de *sklearn.tree*, *mean\_squared\_error*, *mean\_absolute\_error*, *r2\_score* de *sklearn.metrics* y *model\_selection* de *sklearn*.

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.

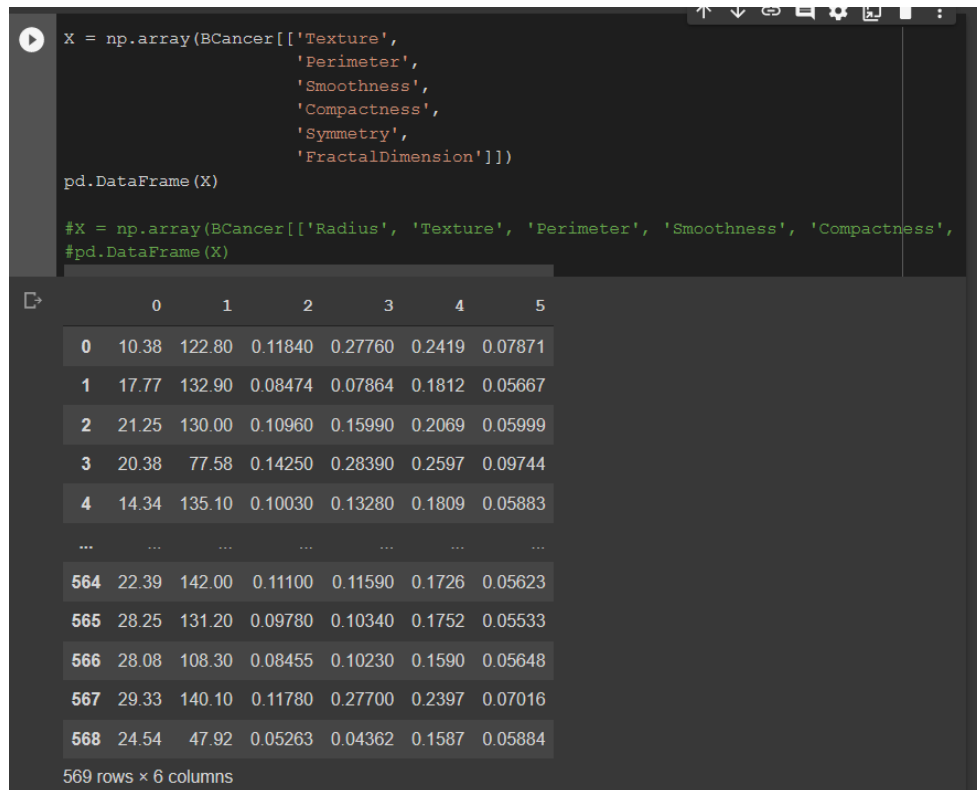


Figura 5: Parte de la matriz con las variables predictoras.

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

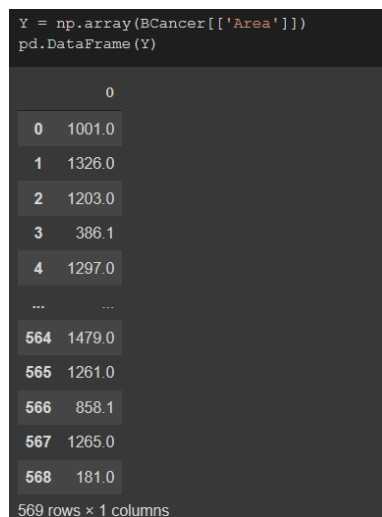


Figura 6: Parte de la matriz con la variable clase.

Más tarde, 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_test, Y_train, Y_test = model_selection.train_test_split(X, Y,
                                                             test_size = 0.2,
                                                             random_state = 1234,
                                                             shuffle = True)
```

```
[ ] pd.DataFrame(X_train)      #
    #pd.DataFrame(X_test)
```

	0	1	2	3	4	5
0	18.22	84.45	0.12180	0.16610	0.1709	0.07253
1	22.44	71.49	0.09566	0.08194	0.2030	0.06552
2	20.76	82.15	0.09933	0.12090	0.1735	0.07070
3	23.84	82.69	0.11220	0.12620	0.1905	0.06590
4	18.32	66.82	0.08142	0.04462	0.2372	0.05768
...	...	...	...	...	...	...
450	15.18	88.99	0.09516	0.07688	0.2110	0.05853
451	15.10	141.30	0.10010	0.15150	0.1973	0.06183
452	18.60	81.09	0.09965	0.10580	0.1925	0.06373
453	18.70	120.30	0.11480	0.14850	0.2092	0.06310
454	13.78	81.78	0.09667	0.08393	0.1638	0.06100

455 rows x 6 columns

```
pd.DataFrame(Y_train)
#pd.DataFrame(Y_test)
```

	0
0	493.1
1	378.4
2	480.4
3	499.0
4	340.9
...	...
450	587.4
451	1386.0
452	481.9
453	1033.0
454	492.1

455 rows x 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 *DecisionTreeRegressor()* y una vez definido, podemos entrenar el modelo para obtener el árbol utilizando la función *fit()*. Generamos el pronóstico:

```
Se genera el pronóstico

[16] #Se genera el pronóstico
Y_Pronostico = PronosticoAD.predict(X_test) #para hacer la validación del modelo
pd.DataFrame(Y_Pronostico)
```

	0
0	420.5
1	355.3
2	477.1
3	278.6
4	571.0
...	...
109	408.2
110	1110.0
111	546.4
112	552.4
113	2499.0

114 rows × 1 columns

Figura 8: Generamos el pronóstico.

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

```
Valores = pd.DataFrame(Y_test, Y_Pronostico) #valor real vs valor pronosticado
Valores
```

	0
420.5	416.2
355.3	357.6
477.1	476.7
278.6	269.4
571.0	568.9
...	...
408.2	419.8
1110.0	1094.0
546.4	551.7
552.4	565.4
2499.0	2501.0

114 rows × 1 columns

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

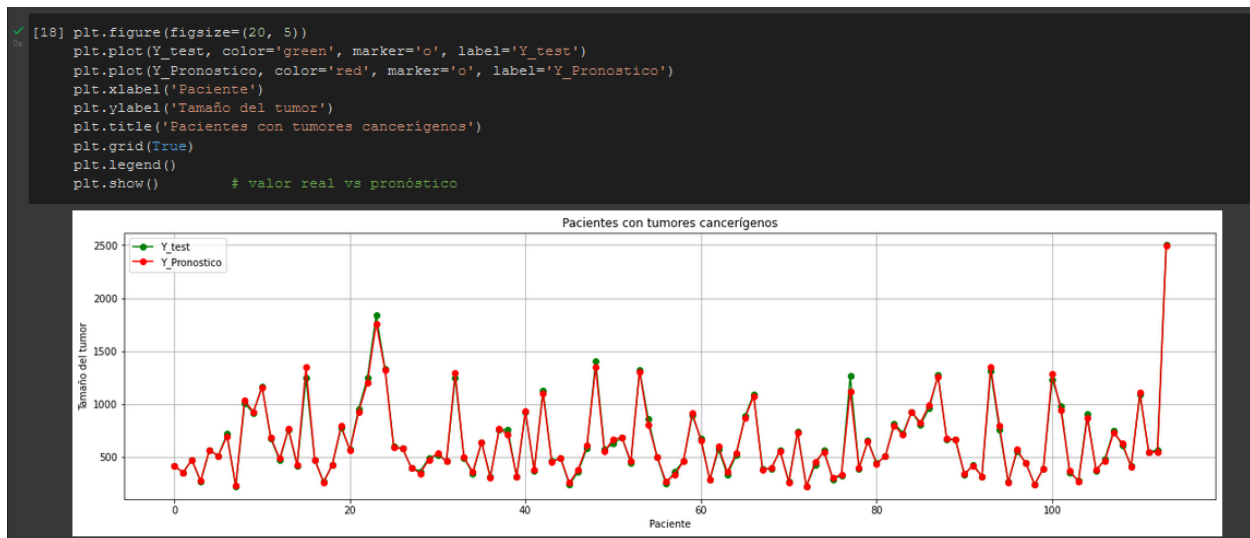


Figura 10: Gráfica de comparación del valor pronosticado y el valor real.

Probando la exactitud del modelo vs los valores reales, obtenemos un  $r2\_score$  de 0.994962. Obteniendo los parámetros del modelo:

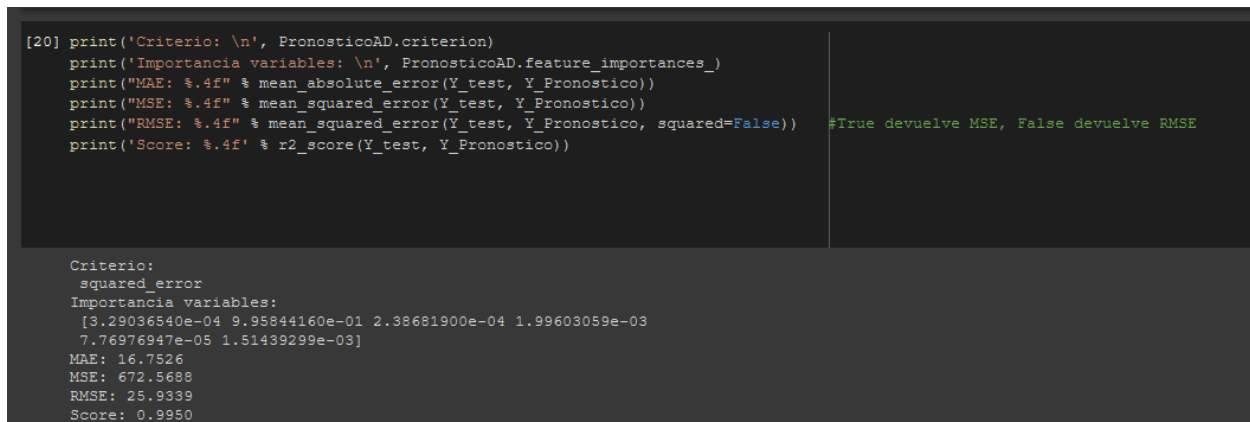


Figura 11: Parámetros del modelo generado.

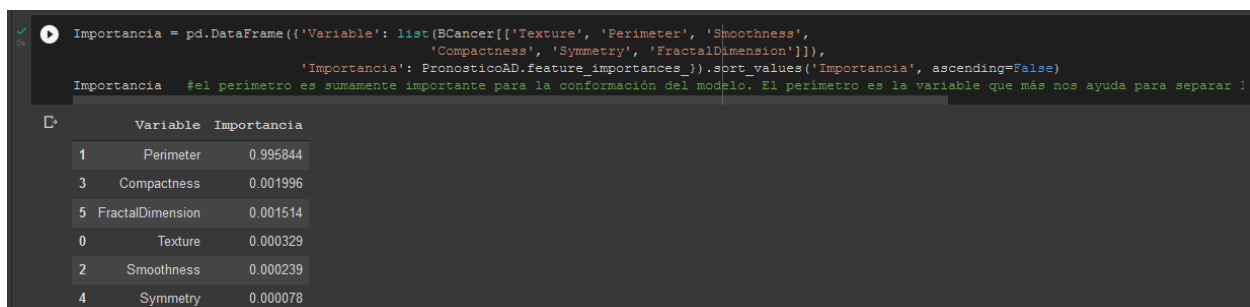


Figura 12: Además, encontramos la importancia de cada una de las variables del pronóstico, obteniendo que la más importante en este caso fue el *perímetro*.

De MAE o error absoluto medio, obtenemos un valor de 16.75, lo cual es alrededor del 2% de la media de los valores de la variable *Area*, indicando que es un pronóstico bastante acertado. El score de 0.995 indica que el pronóstico del área del tumor se logrará con un 99% de efectividad. Además, el valor de RMSE (25.93), indica que los valores del modelo final se alejan en promedio 25.93 unidades del valor real.

Hecho lo anterior, utilizamos el instalador *pip* para instalar *graphviz*, un visualizador de árboles. Con esta biblioteca, mostramos el árbol de manera gráfica:

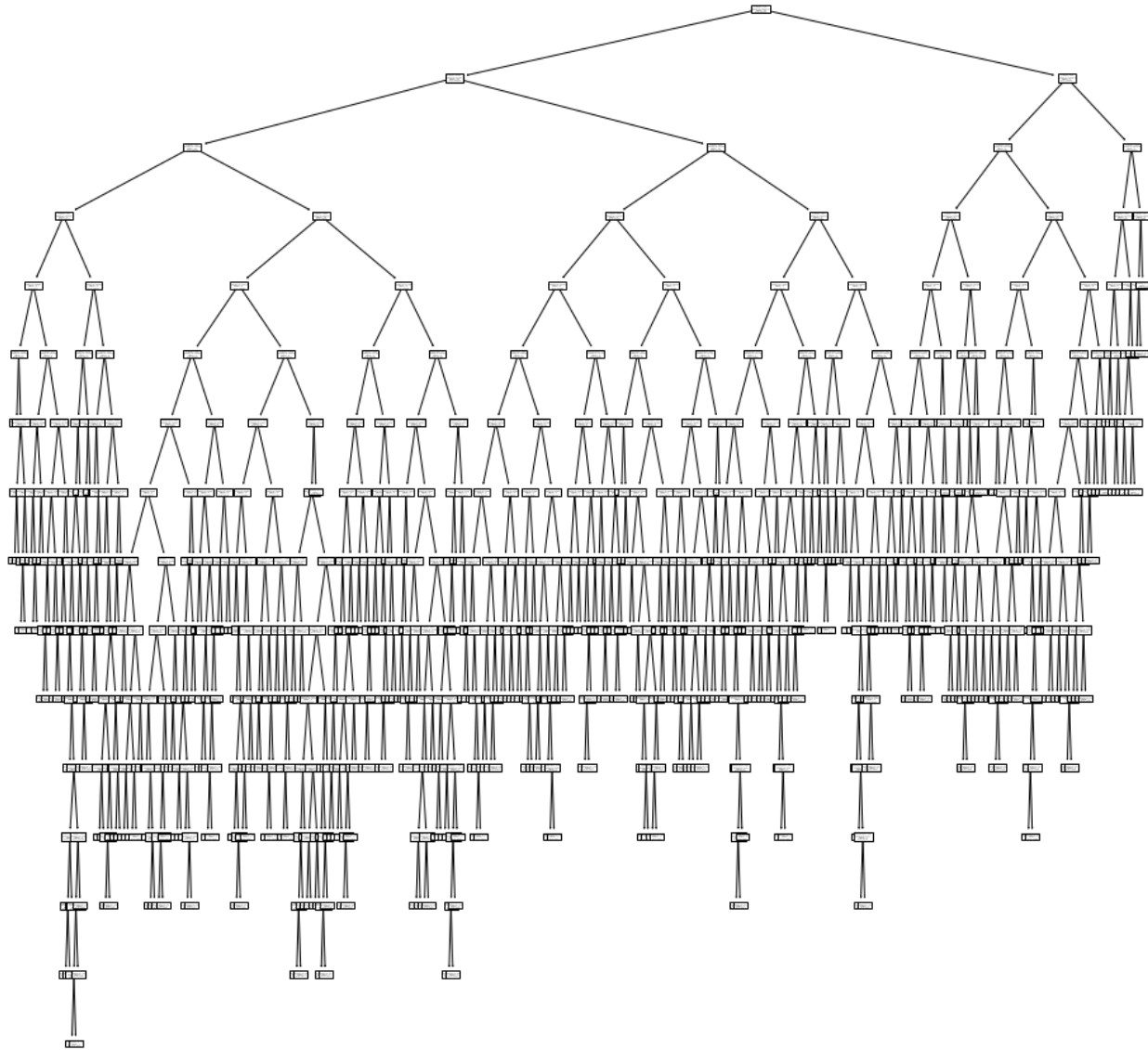


Figura 13: Visualización gráfica del árbol. En este caso, hay hojas que tienen un único valor, lo que puede llevar a un sobre ajuste. Por ello, utilizaremos un mínimo de dos elementos por cada nodo.



De la misma manera, utilizamos la función *export\_text* para generar un reporte textual de los valores obtenidos en el árbol de pronóstico. Además, podemos utilizar el objeto *PronosticoAD* de *DecisionTreeRegressor* para hacer predicciones de valores ingresando el valor de *Texture*, *Perimeter*, *Smoothness*, *Compactness*, *Symmetry* y *FractalDimension*. Por ejemplo, para un valor de *Texture* de 10.38, *Perimeter* de 122.8, *Smoothness* de 0.118, *Compactness* de 0.277, *Symmetry* de 0.2419 y *FractalDimension* de 0.0787, obtenemos una predicción de 1001.

Cambiando el valor de *max\_depth* a 8, el número *min\_samples\_split* a 4 y el número mínimo de elementos por hoja *min\_samples\_leaf* a 2, con la intención de obtener un mejor ajuste. De manera gráfica, vemos que el valor obtenido con el árbol y el valor real se ven de la siguiente manera:

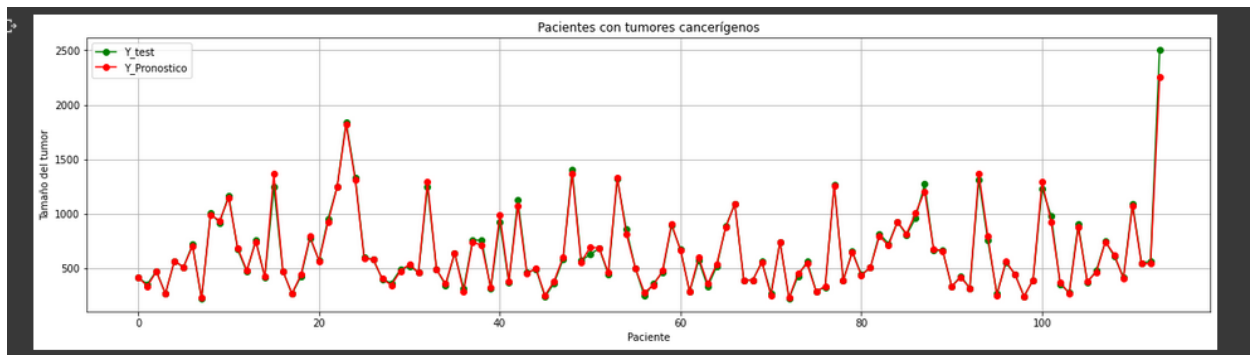


Figura 14: Comparación del valor real y el valor pronosticado con las modificaciones realizadas en la obtención del árbol.

Para esta configuración, obtenemos un *r2\_score* de 0.9914, con los siguientes parámetros del modelo:

```
Criterio:
  squared_error
Importancia variables:
[1.74813546e-04 9.96222481e-01 1.13849950e-04 2.21681716e-03
 1.53423900e-04 1.11861405e-03]
MAE: 18.7071
MSE: 1140.7676
RMSE: 33.7753
Score: 0.9915
```

Figura 15: Parámetros del nuevo modelo.

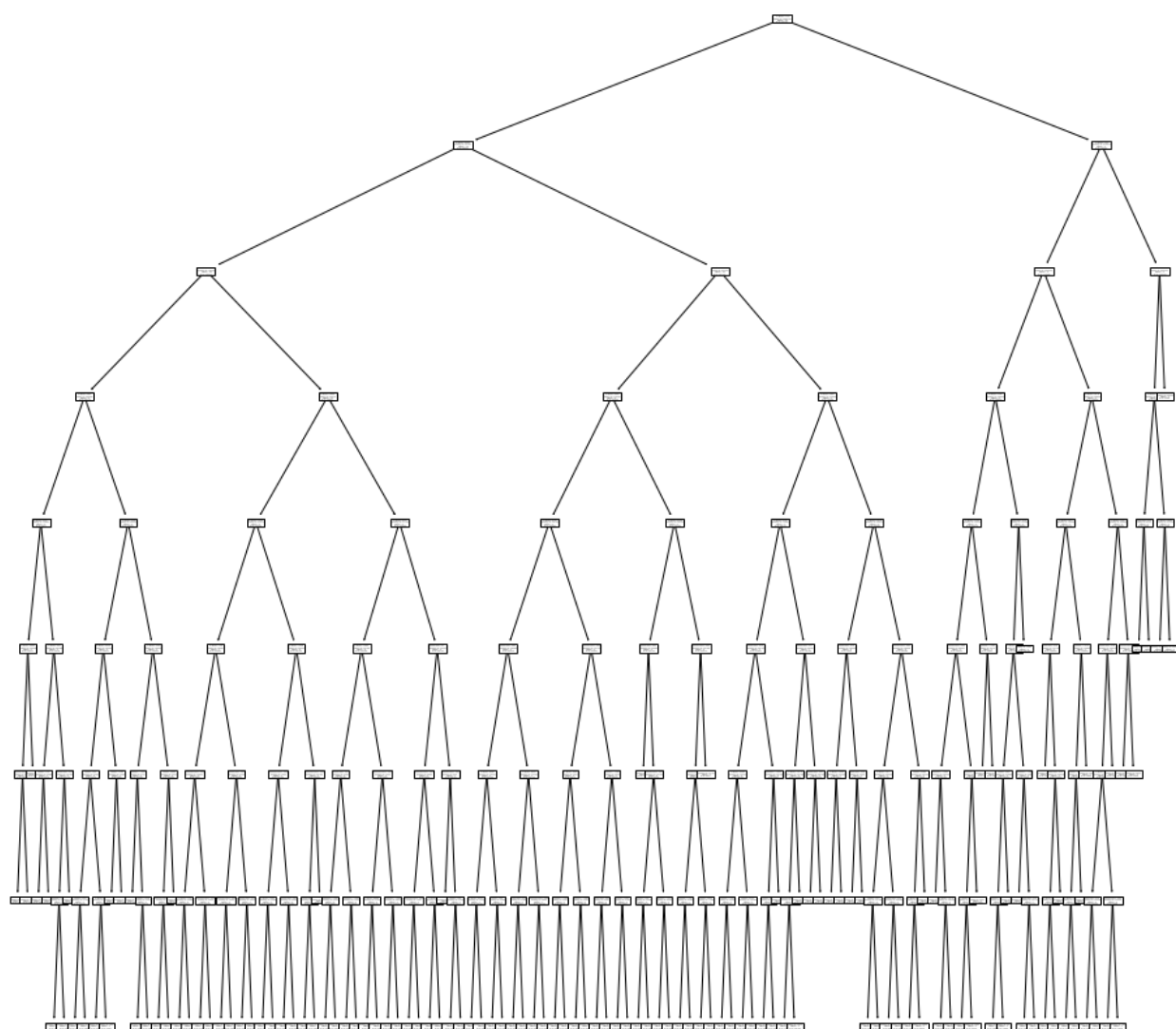


Figura 16: Nuevo árbol de regresión generado.

Para un valor mínimo de elementos por hoja de 15, obtenemos lo siguiente:

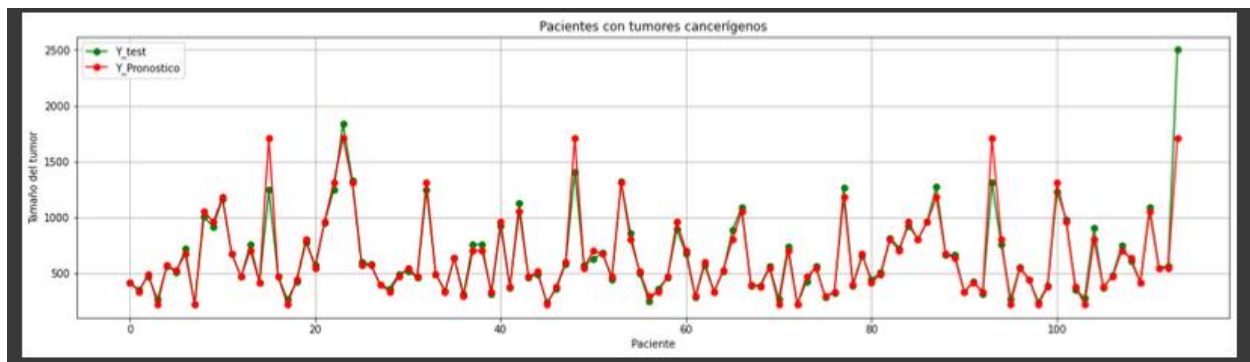


Figura 17: Comparación entre el valor real y el valor pronosticado.

```

Criterio:
squared_error
Importancia variables:
[0.00000000e+00 9.99828237e-01 0.00000000e+00 1.71763487e-04
 0.00000000e+00 0.00000000e+00]
MAE: 42.3001
MSE: 10832.6403
RMSE: 104.0800
Score: 0.9189

```

Figura 18: Parámetros del nuevo modelo.

Con este nuevo valor de elementos por hoja, obtuvimos un ajuste con *score* mucho más bajo que para los dos casos anteriores, por lo que convendría más utilizar los modelos previamente expuestos.

## Conclusiones

A lo largo de esta práctica, tuvimos como principal objetivo el obtener pronósticos 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 un mejor *score* y mejores parámetros en general, por lo que ese sería el modelo que utilizaría para predecir 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.