



Universidad Nacional Autónoma de México
Facultad de Ingeniería



PRÁCTICA 14

PRONÓSTICO (ÁRBOL DE DECISIÓN)

Minería de Datos

Profesor:

Dr. Molero Castillo Guillermo Gilberto

Grupo 1

Alumna:

Monroy Velázquez Alejandra Sarahí

No. Cuenta: 314000417

OBJETIVO

Obtener un pronóstico del área del tumor de mama a través de un árbol de decisión (regresión).

DESARROLLO

El conjunto de datos con el que se trabajará corresponde a estudios clínicos a partir de imágenes digitalizadas de pacientes con cáncer de mama de Wisconsin (WDBC, Wisconsin Diagnostic Breast Cancer).

Primero comenzamos la importación de bibliotecas correspondientes que nos ayudarán para la realización del código, las cuales son *pandas* para la manipulación y análisis de datos, *numpy* para crear vectores y matrices, *matplotlib* para la generación de gráficas, así como *seaborn* para la visualización de datos. Por último, la biblioteca *files* para subir el archivo csv.

```
import pandas as pd          # Para la manipulación y análisis de datos
import numpy as np           # Para crear vectores y matrices n dimensionales
import matplotlib.pyplot as plt # Para la generación de gráficas a partir de los datos
import seaborn as sns        # Para la visualización de datos basado en matplotlib
%matplotlib inline
```

```
from google.colab import files
files.upload()
```

Elegir archivos Ningún archivo seleccionado Upload widget is only available when the cell has been executed in the current browser
Saving WDBCOriginal.csv to WDBCOriginal.csv
{'WDBCOriginal.csv': b'\xef\xbb\xbfIDNumber,Diagnosis,Radius,Texture,Perimeter,Area,Smoothness,Compactness,Concavity,ConcavePoints,Symmetry,FractalDimension'}

Una vez importadas, el *dataframe* se lee y se despliega en pantalla:

```
BCancer = pd.read_csv('WDBCOriginal.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 × 12 columns

Ahora que el *dataframe* está cargado, comenzamos con los pasos vistos en clase.

Como paso extra podemos observar ciertos valores de las variables a través de la función *describe()*.

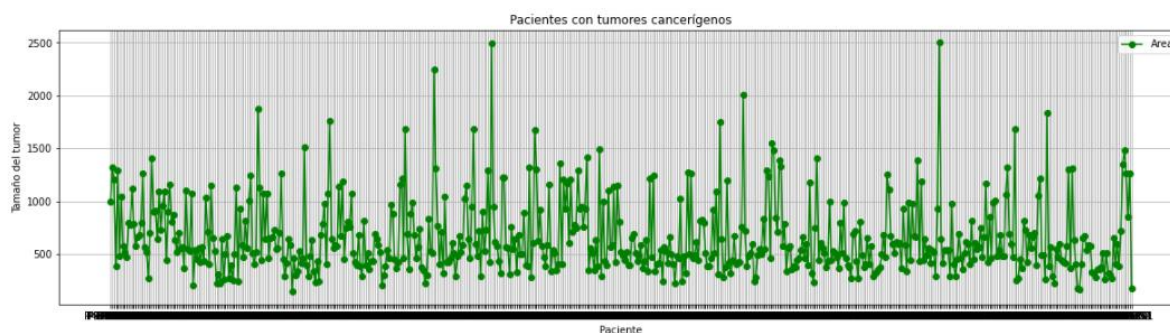
```
BCancer.describe()
```

	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension
count	569.000000	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	0.062798
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440

Evaluación Visual

Graficamos el área, o tamaño del tumor en cada paciente, para visualizar un panorama general de este dato:

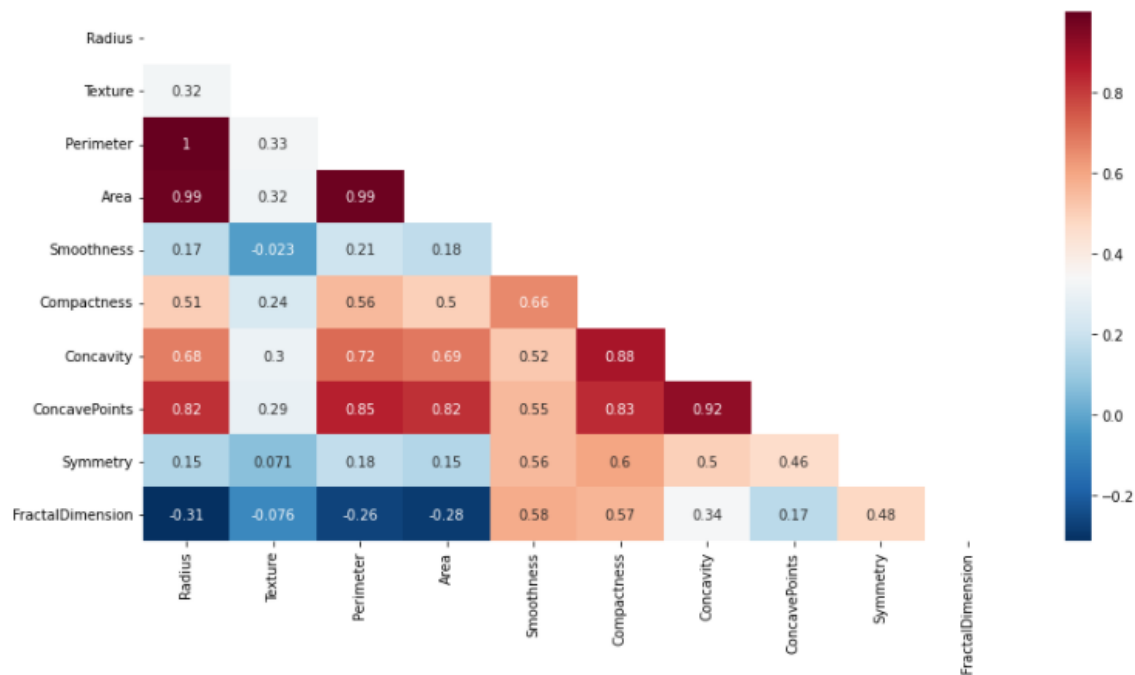
```
plt.figure(figsize=(20, 5))
plt.plot(BCancer['IDNumber'], BCancer['Area'], color='green', marker='o', label='Area')
plt.xlabel('Paciente')
plt.ylabel('Tamaño del tumor')
plt.title('Pacientes con tumores cancerígenos')
plt.grid(True)
plt.legend()
plt.show()
```



Selección de Características

Antes de pasar a la aplicación del algoritmo, se realiza una selección de características, para descartar aquellas variables poco relevantes, o redundantes, en este caso se hará un análisis correlacional de datos. Para ello generamos un mapa de calor mostrando solo la matriz inferior:

```
plt.figure(figsize=(14,7))
MatrizInf = np.triu(BCancer.corr())
sns.heatmap(BCancer.corr(), cmap='RdBu_r', annot=True, mask=MatrizInf)
plt.show()
```



Las variables seleccionadas son:

- 1) Textura [Posición 3]
- 2) Area [Posición 5]
- 3) Smoothness [Posición 6]
- 4) Compactness [Posición 7]
- 5) Symmetry [Posición 10]
- 6) FractalDimension [Posición 11]
- 7) Perimeter [Posición 4] - Para calcular el área del tumor

Aplicación del algoritmo

Para comenzar a aplicar el algoritmo necesitamos importar la librería *sklearn*, ya que utilizaremos los módulos *DecisionTreeRegressor*, *mean_squared_error*, *mean_absolute_error*, *r2_score*, y *model_selection*.

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn import model_selection
```

Lo siguiente será seleccionar las variables predictoras (X) y la variable a pronosticar (Y):

```
X = np.array(BCCancer[['Texture',
                        'Perimeter',
                        'Smoothness',
                        'Compactness',
                        'Symmetry',
                        'FractalDimension']])

pd.DataFrame(X)

#X = np.array(BCCancer[['Radius', 'Texture', 'Perimeter',
#pd.DataFrame(X)
```

	0	1	2	3	4	5
0	10.38	122.80	0.11840	0.27760	0.2419	0.07871
1	17.77	132.90	0.08474	0.07864	0.1812	0.05667
2	21.25	130.00	0.10960	0.15990	0.2069	0.05999
3	20.38	77.58	0.14250	0.28390	0.2597	0.09744
4	14.34	135.10	0.10030	0.13280	0.1809	0.05883
...
564	22.39	142.00	0.11100	0.11590	0.1726	0.05623
565	28.25	131.20	0.09780	0.10340	0.1752	0.05533
566	28.08	108.30	0.08455	0.10230	0.1590	0.05648
567	29.33	140.10	0.11780	0.27700	0.2397	0.07016
568	24.54	47.92	0.05263	0.04362	0.1587	0.05884

569 rows × 6 columns

```
Y = np.array(BCancer[['Area']])
pd.DataFrame(Y)
```

	0
0	1001.0
1	1326.0
2	1203.0
3	386.1
4	1297.0
...	...
564	1479.0
565	1261.0
566	858.1
567	1265.0
568	181.0

569 rows × 1 columns

Las variables predictoras son: *Texture*, *Perimeter*, *Smoothness*, *Compactness*, *Symmetry* y *FractalDimension*; mientras que la variable a pronosticar será *Area*.

Se hace la división de los datos, donde se generará el entrenamiento y se despliega tanto la variable `x_train`, como `y_train`:

[illegible]

```
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 × 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 × 1 columns

Luego, se entrena el modelo a través de un Árbol de Decisión (Regresión):

```
PronosticoAD = DecisionTreeRegressor()
PronosticoAD.fit(X_train, Y_train)
```

DecisionTreeRegressor()

Y se genera el pronóstico:

```
#Se genera el pronóstico
Y_Pronostico = PronosticoAD.predict(X_test)
pd.DataFrame(Y_Pronostico)
```

	0
0	420.5
1	344.9
2	477.1
3	278.6
4	575.5
...	...
109	408.2
110	1102.0
111	546.4
112	546.4
113	2250.0

114 rows × 1 columns

Podemos visualizar una comparativa entre el valor del área del tumor que agrupamos en `Y_test` contra el valor de `Y_Pronostico`, el cual resulta del entrenamiento a través del árbol de decisión. Si observamos cada registro, nos damos cuenta de que los valores pronosticados se acercan a los reales:

```
Valores = pd.DataFrame(Y_test, Y_Pronostico)
Valores
```

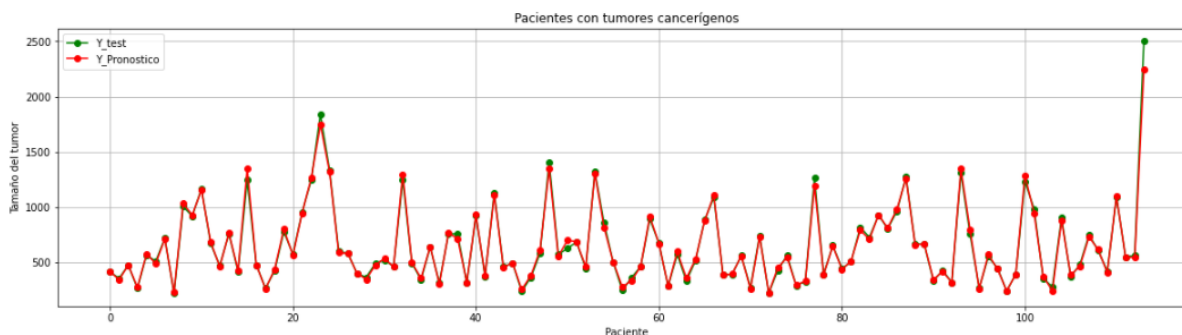
```

      0
420.5  416.2
344.9  357.6
477.1  476.7
278.6  269.4
575.5  568.9
...
408.2  419.8
1102.0 1094.0
546.4  551.7
546.4  565.4
2250.0 2501.0
114 rows × 1 columns
```

Los valores anteriores, los desplegamos en una gráfica para poder realizar una comparativa visual:

```

plt.figure(figsize=(20, 5))
plt.plot(Y_test, color='green', marker='o', label='Y_test')
plt.plot(Y_Pronostico, color='red', marker='o', label='Y_Pronostico')
plt.xlabel('Paciente')
plt.ylabel('Tamaño del tumor')
plt.title('Pacientes con tumores cancerígenos')
plt.grid(True)
plt.legend()
plt.show()
```



Si imprimimos nuestro score nos daremos cuenta de que nuestro modelo ha sido entrenado con mucha exactitud:

```
r2_score(Y_test, Y_Pronostico)

0.9916144429382922
```

Obtención de los parámetros del modelo & conformación del modelo de pronóstico

En este paso obtenemos los parámetros del modelo, los cuales incluyen el criterio, la importancia de las variables, el MSE, RMSE y el score. Luego también imprimimos las variables y las ordenamos de acuerdo con su importancia

```
print('Criterio: \n', PronosticoAD.criterion)
print('Importancia variables: \n', PronosticoAD.feature_importances_)
print("MAE: %.4f" % mean_absolute_error(Y_test, Y_Pronostico))
print("MSE: %.4f" % mean_squared_error(Y_test, Y_Pronostico))
print("RMSE: %.4f" % mean_squared_error(Y_test, Y_Pronostico, squared=False)) #True devuelve MSE, False devuelve RMSE
print("Score: %.4f" % r2_score(Y_test, Y_Pronostico))
```

```
Criterio:
squared_error
Importancia variables:
[3.58291782e-04 9.93495986e-01 1.16856034e-03 1.76332984e-03
 1.73624584e-03 1.47758656e-03]
MAE: 17.9842
MSE: 1119.5988
RMSE: 33.4604
Score: 0.9916
```

```
Importancia = pd.DataFrame({'Variable': list(BCancer[['Texture', 'Perimeter', 'Smoothness',
'Compactness', 'Symmetry', 'FractalDimension']]),
'Importancia': PronosticoAD.feature_importances_}).sort_values('Importancia', ascending=False)

Importancia
```

	Variable	Importancia
1	Perimeter	0.993496
3	Compactness	0.001763
4	Symmetry	0.001736
5	FractalDimension	0.001478
2	Smoothness	0.001169
0	Texture	0.000358

- El error absoluto medio (MAE) del algoritmo es 17.9, que es alrededor de 2.7% de la media de todos los valores de la variable 'Area' (654.88). Esto significa que el algoritmo realiza un notable pronóstico.
- Además, se tiene un Score de 0.9916, el cual indica que el pronóstico del Area del tumor se logrará con un 99% de efectividad.

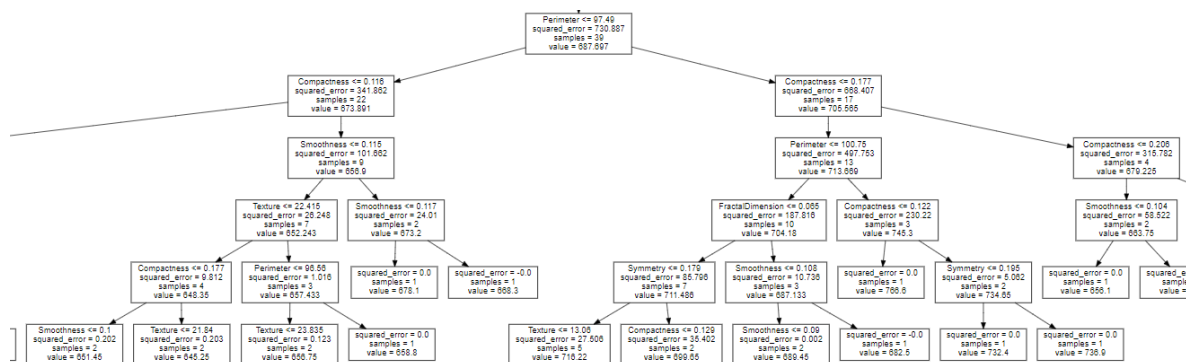
- Por otro lado, los pronósticos del modelo final se alejan en promedio 33.46 (RMSE) unidades del valor real.

Lo siguiente será instalar el paquete `graphviz` importar la biblioteca y el módulo `export_graphviz` de `sklearn` ya que imprimiremos la forma del árbol.

```
!pip install graphviz
```

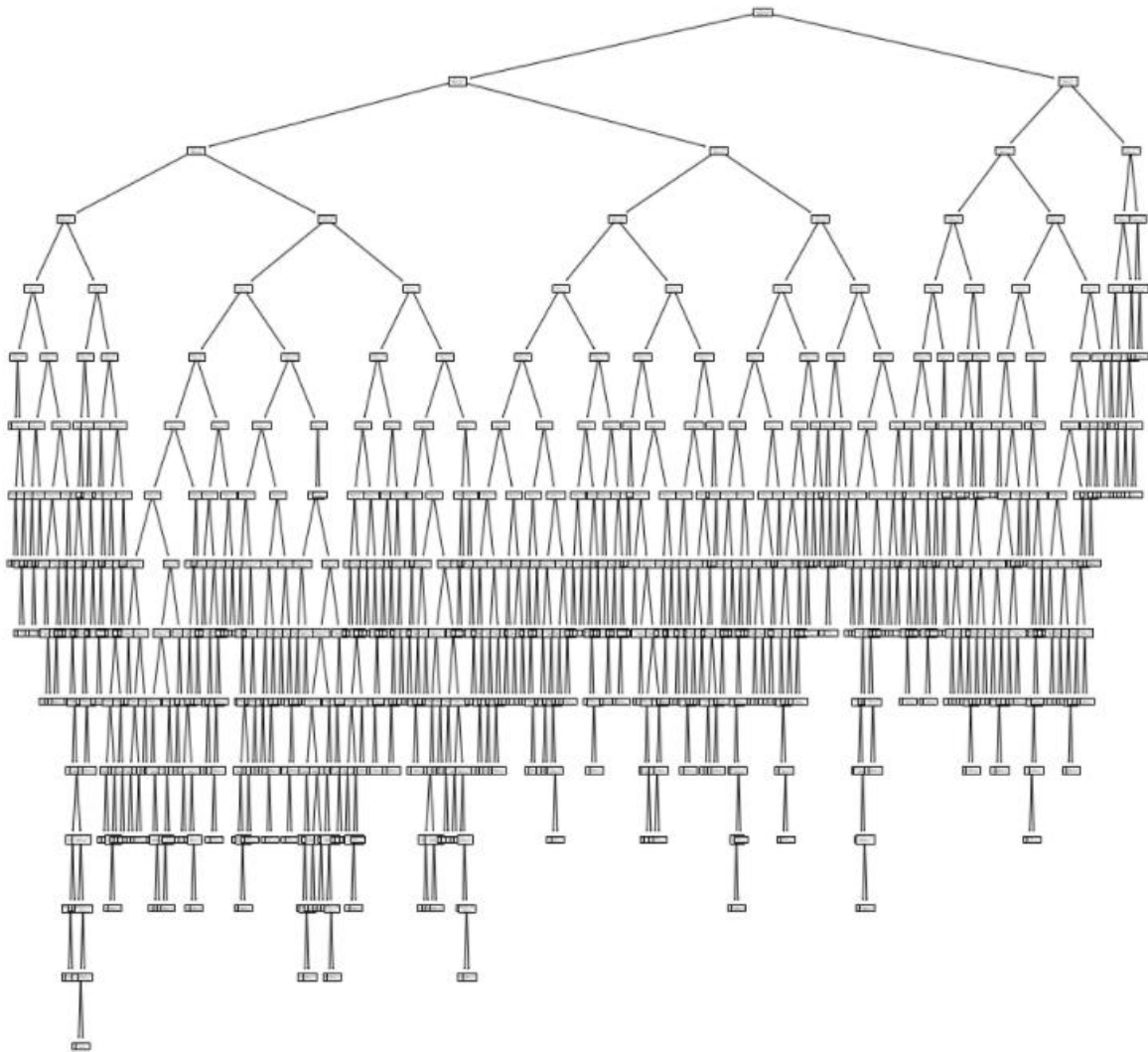
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (0.10.1)

```
import graphviz
from sklearn.tree import export_graphviz
```



Como la imagen del árbol es muy grande para visualizarla de forma general, imprimimos de forma distinta:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(16,16))
plot_tree(PrognosticoAD, feature_names = ['Texture', 'Perimeter', 'Smoothness',
                                           'Compactness', 'Symmetry', 'FractalDimension'])
plt.show()
```



Para una mejor legibilidad imprimimos de esta manera:

```
from sklearn.tree import export_text
Reporte = export_text(PronosticoAD, feature_names = ['Texture', 'Perimeter', 'Smoothness',
                                                    'Compactness', 'Symmetry', 'FractalDimension'])
print(Reporte)
```

```
|--- Perimeter <= 110.60
|   |--- Perimeter <= 84.01
|   |   |--- Perimeter <= 70.18
|   |   |   |--- Perimeter <= 60.40
|   |   |   |   |--- Perimeter <= 53.68
|   |   |   |   |   |--- Perimeter <= 45.85
|   |   |   |   |   |   |--- value: [143.50]
|   |   |   |   |   |   |--- Perimeter > 45.85
|   |   |   |   |   |       |--- Texture <= 22.62
|   |   |   |   |   |       |   |--- Perimeter <= 52.49
|   |   |   |   |   |       |   |   |--- value: [201.90]
|   |   |   |   |   |       |   |--- Perimeter > 52.49
|   |   |   |   |   |       |   |   |--- value: [203.90]
|   |   |   |   |   |       |   |--- Texture > 22.62
|   |   |   |   |   |       |   |   |--- Smoothness <= 0.08
|   |   |   |   |   |       |   |   |   |--- Compactness <= 0.05
|   |   |   |   |   |       |   |   |   |   |--- value: [181.00]
|   |   |   |   |   |       |   |   |   |   |--- Compactness > 0.05
|   |   |   |   |   |       |   |   |   |   |   |--- value: [178.80]
|   |   |   |   |   |       |   |   |   |   |--- Smoothness > 0.08
|   |   |   |   |   |       |   |   |   |   |   |--- value: [170.40]
|   |   |   |   |   |       |   |--- Perimeter > 53.68
|   |   |   |   |   |       |   |--- Perimeter <= 56.10
```

La imagen del árbol es grande, pero se puede leer en el siguiente orden:

1. La decisión que se toma para dividir el nodo.
2. El tipo de criterio que se usó para dividir cada nodo.
3. Cuantos valores tiene ese nodo.
4. Valores promedio.
5. Por último, el valor pronosticado en ese nodo.

Nuevos pronósticos

Para nuevos pronósticos implementamos el siguiente bloque de código, donde cada variable predictora tiene un valor; esto servirá para predecir el valor del area:

```
AreaTumorID1 = pd.DataFrame({'Texture': [10.38],
                              'Perimeter': [120.8],
                              'Smoothness': [0.11840],
                              'Compactness': [0.27760],
                              'Symmetry': [0.2419],
                              'FractalDimension': [0.07871]})

PronosticoAD.predict(AreaTumorID1)

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:439: UserWarning: X has feature names, but {self.__class__.__name__} was fitted without "array([980.5])"
```

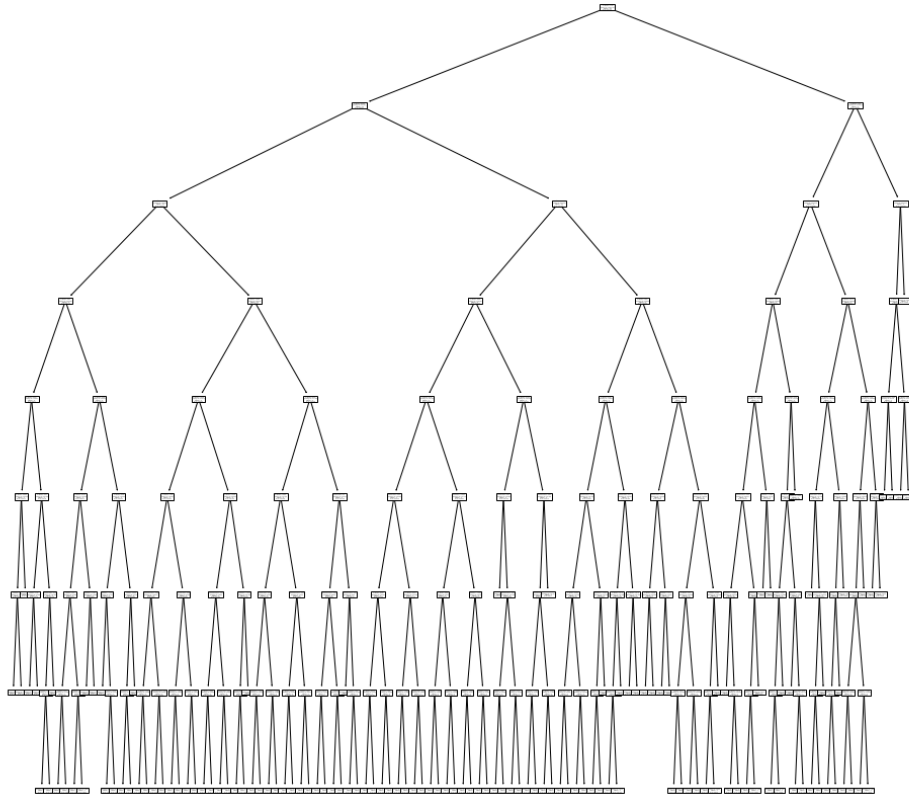
Arreglando el sobreajuste

Si resegamos a observar la forma del árbol, veremos que existe un sobreajuste, para evitar eso, al momento de entrenar el modelo debemos indicar la máxima profundidad a la cual puede llegar el árbol, la cantidad mínima de datos que debe de tener un nodo hoja, y la cantidad mínima de datos para que un nodo de decisión se pueda dividir.

```
PronosticoAD = DecisionTreeRegressor(max_depth=8, min_samples_split=4, min_samples_leaf=2)
PronosticoAD.fit(X_train, Y_train)

DecisionTreeRegressor(max_depth=8, min_samples_leaf=2, min_samples_split=4)
```

Repetimos el proceso desde obtener el pronóstico hasta la impresión del árbol:



Si realizamos el mismo nuevo pronóstico, vemos que el resulta cambia, ya que el entrenamiento y el modelo cambio.

```
AreaTumorID1 = pd.DataFrame({'Texture': [10.38],
                              'Perimeter': [120.8],
                              'Smoothness': [0.11840],
                              'Compactness': [0.27760],
                              'Symmetry': [0.2419],
                              'FractalDimension': [0.07871]})
PronosticoAD.predict(AreaTumorID1)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:439: Us
f"X has feature names, but {self.__class__.__name__} was fit
array([991.83333333])
```

CONCLUSIÓN

En esta sesión pusimos en práctica el algoritmo de árbol de decisión mediante regresión para pronosticar valores, volviendo a trabajar con el set de datos de los estudios clínicos de cáncer de mama, a mi parecer tanto el algoritmo de

regresión lineal como estas son muy buenas herramientas, sin embargo, con esta práctica entendí que muchas veces se necesitara hacer ajustes al modelo entrenado para mejorar su desempeño, cosa que en el algoritmo lineal no lo hicimos.