Cervantes Rubí Brandon

Objetivo.

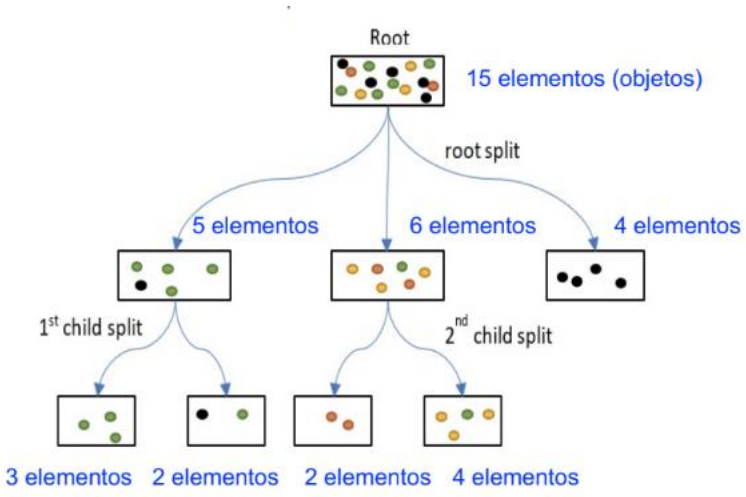
Pronosticar el área del tumor de pacientes con cáncer de mama a través de un árbol de decisión.

Características.

Estudios clínicos a partir de imágenes digitalizadas de pacientes con cáncer de mama de Wisconsin (WDBC, Wisconsin Diagnostic Breast Cancer).

Los árboles de decisión son algoritmos los cuales toman una serie de variables independientes como entrada y las analizan una por una en cada uno de los diferentes niveles del árbol partiendo de una raíz para obtener el determinado valor de una variable dependiente.

En este aspecto tendremos 2 diferentes tipos de árboles de decisión, primero los árboles de clasificación los cuales trabajan con etiquetas como resultados (A, B, C), (0/1), etc. y después están los árboles de regresión, dichos arboles están diseñados para entregar como resultado un valor continuo.



Desarrollo.

Como primer paso tenemos la importación de las librerías necesarias para trabajar con el conjunto de datos de los pacientes a segmentar. `pandas` para la manipulación de los datos, `numpy` para el manejo de matrices, `matplotlib` y `seaborn` para la visualización de estos datos mediante gráficos de diferente tipo dependiendo del análisis.

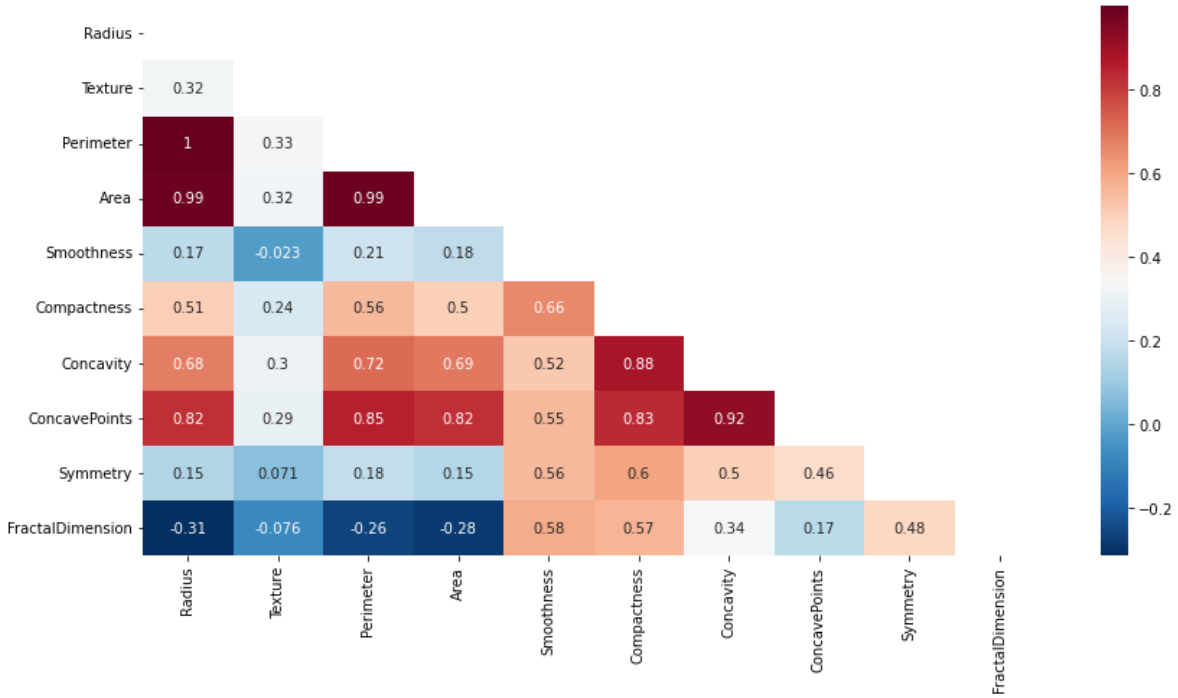
Después tenemos que hacer la lectura de nuestros datos los cuales están relacionados a las características de los tumores presentados en cada uno de las pacientes relacionadas como su área, textura y la etiqueta la cual nos indica si se trata de un tumor maligno o benigno que nos ayudara más adelante para el proceso de entrenamiento del modelo.

Tenemos que leer los datos con ayuda de pandas el cual nos genera un `dataframe` con 569 registros y 12 columnas asociadas a las características de los tumores presentados, de los cuales podemos hacer una agrupación con el método `goupby` de pandas para conocer la cuenta de los diagnósticos benignos (357 registros) y los malignos (212 registros).

A continuación, se muestra una parte del conjunto de datos leído

	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

Considerando las 12 variables en el modelo se decidió hacer un análisis para la selección de características, realizando un análisis de correlación con la matriz de correlaciones se trazando un mapa de calor para hacer el resultado más legible pues este mapa nos muestra con un indicé de -1 a 1 el nivel de correlación que tiene cada variable con las demás. A continuación, se muestra el mapa de calor obtenido resaltando las relaciones fuertes entre área y perímetro, radio y área, Concavity con la mayoría de las variables, Concave points de igual manera con la mayoría de las variables por lo cual se decidió que se eliminarían del modelo.



Al final del análisis de correlación las variables independientes que se seleccionaron fueron las siguientes.

1.- Texture.

2.- Area.

3.- Smoothness.

4.- Compactness.

5.- Symmetry.

6.- FractalDimension.

Con las variables independientes del modelo seleccionadas se procedió con la aplicación del algoritmo, Para ello fue necesaria la clase `DecisionTreeClassifier, classification_report` de `sklearn`, y por último `model_selection` para la partición del conjunto de entrenamiento y conjunto de pruebas.

En esta ocasión la variable dependiente del modelo tomará el nombre de variable clase y las variables independientes se nombrará variables predictoras. Para la variable clase hizo la discriminación maligno o benigno para el tipo de tumor que presentan los pacientes, como variables predictoras se usaran a cada una de las 6 variables que resultaron del análisis de correlación anterior; Texture, area, smoothness, compactness, symmetry y fractal dimensión. Se hizo un nuevo arreglo de `numpy` con todas estas variables al cual se le llamo `X`, por otra parte, `Y` es el arreglo correspondiente a la columna diagnosis que contiene como valores {Malignant, Benign}

Con estos dos arreglos se utilizó `model_selection` para separar los 2 conjuntos en 4, dos correspondientes a las `X` y `Y` de entrenamiento que representan el 80% del total los datos (455 registros) y otros 2 correspondientes a las `X` y `Y` del conjunto de pruebas que representan el otro 20% (114 registros) del conjunto de datos., por último para utilizar esta librería fue necesario establecer una semilla de aleatoriedad y el atributo `shuffle` en True para que mezcle los registros evitando tomarlos tal y como están lo cual podría llevar a una mala división de los conjuntos.

```
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y,
                                                                    test_size = 0.2,
                                                                    random_state = 1234,
                                                                    shuffle = True
```

Con los conjuntos de datos establecidos, para generar el árbol, similar a la práctica anterior, es necesario utilizar la clase `DecisionTreeClassifier` para crear un objeto a partir de ella sobre la que es posible definir ciertos parámetros del árbol. Nos daremos cuenta de que los parámetros trabajados en esta ocasión son los mismos de la practica anterior, con `max_deep` podemos establecer la profundidad del árbol, con `min_samples_split` se define un mínimo de hojas que se tienen que generar y con `min_samples_leaf` podemos controlar el número de registros que definan una hoja de tal forma que los nodos hoja que no cumplan con este mínimo no se consideran para las reglas que se generen al final.

Cuando se crea el objeto con los parámetros establecidos, se usa el método `fit` pasando como parámetros los conjuntos de datos de entrenamiento que corresponden al 80% de los registros. Para generar las métricas de rendimiento del modelo primero es necesario utilizar el método `predict` pasando como parámetros el conjunto de datos de prueba que corresponde a las variables independientes del modelo `X_validation` para generar las clases que el modelo les asigne y comparar los resultados con el conjunto de datos que tiene los valores reales para estos registros, es decir `Y_validation`.

```
#MODIFICANDO LOS HIPERPARAMETROS
ClasificacionAD = DecisionTreeClassifier(max_depth=6, min_samples_split=4, min_samples_leaf=2)
ClasificacionAD.fit(X_train, Y_train)
Y_Clasificacion = ClasificacionAD.predict(X_validation)
Valores = pd.DataFrame(Y_validation, Y_Clasificacion)
Valores
```

Malignant	Malignant
Benign	Benign
Malignant	Malignant
Malignant	Malignant
Malignant	Benign

Con los valores de clase predichos por el modelo (Y_clasificacion) y los valores reales (Y_validation) se creó un nuevo `dataframe` donde podemos hacer una comparación rápida entre ellos observando que en la mayoría de los casos los resultados del estado del tumor son correctos lo cual nos da indicios de un indicé de score alto.

Para validarlo se utilizó el método de nuestro objeto creado a partir de `DecisionTreeClassifier` llamado score y pasando como parámetros los conjuntos de variable clase y predictora de validación, es decir, el que corresponde al 20%.

```
#Se calcula la exactitud promedio de la validación
ClasificacionAD.score(X_validation, Y_validation)
```

Con un mínimo de 2 registros por hoja y la profundidad máxima (8 niveles) se tiene una precisión de 91.23% en comparación con la regresión logística de prácticas anteriores donde se obtuvo un 93.85%. A diferencia de la práctica de pronóstico, en clasificación se obtuvo un menor indicé de score utilizando arboles de decisión para hacer la discriminación de los registros.

De igual manera se hicieron varias pruebas para la configuración de mínimo 4 registros en los nodos hoja y mínimo 2 nodos hojas variando la profundidad de acuerdo con la siguiente tabla.

Profundidad	5	6	7
Score (%)	89.47	92.1	89.47

Por otra parte, si se toma en cuenta a todas las variables del modelo, una profundidad de 7, mínimo 3 registros en cada nodo hoja y mínimo 2 nodos hoja se obtiene un score del 93% el cual es mayor al obtenido cuando se consideraron menos variables a costa de un mayor poder computacional.

En esta ocasión el sobreajuste no parece ser un problema que se presente ya que el indicé del score presenta un rango de incertidumbre de poco menos del 10% representando además un modelo que generaliza bien los casos para el contexto con el que se trabaja.

Además de la precisión de los modelos también podemos obtener la matriz de clasificación o matriz de confusión donde podemos comparar los valores obtenidos por el modelo con los valores reales y de esta manera obtener otras métricas para medir el rendimiento que presente el modelo, en ella podemos a preciar todos los aciertos contabilizados en la diagonal principal y los errores como el complemento de esta diagonal principal.

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

Se utilizo `classification_report` para obtener algunas de estas métricas; `recall` la cual debemos maximizar ya que es un índice que mide la habilidad del modelo para encontrar todas las clases relevantes en el data set, la `precisión` del modelo para asignar las clases y el `valor F1` cuyo objetivo es combinar la precisión y la exhaustividad (`recall`) en un solo indicé. Como todos estos índices varían según la configuración del árbol se detallarán debajo en el resumen de cada uno de los modelos.

```
#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))
```

Con el atributo `feature_importances_` podemos generar una lista ordenada con la importancia de cada una de las variables que se consideraron, a partir de esta lista se creó un dataframe ordenado por valor de importancia en el que podemos observar variaciones dependiendo de la configuración.

Para las tres configuraciones presentadas se generaron diferentes arboles partiendo de diferentes variables las cuales se detallan, pare generar las gráficas de los árboles y el conjunto de reglas fue necesario trabajar con los módulos `export_graphviz`, `plot_tree` y `export_text` de `sklearn.tree`.

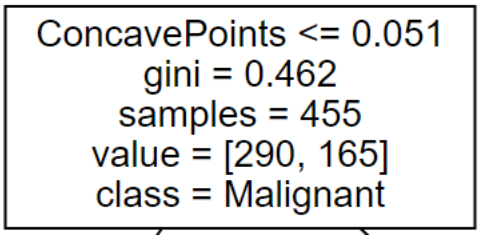
Cervantes Rubí Brandon

Inteligencia Artificial 2022-2, G.3, 03/05/2022
Practica No.12 Clasificación (Árbol de decisión).

Árbol con todas las variables del modelo.

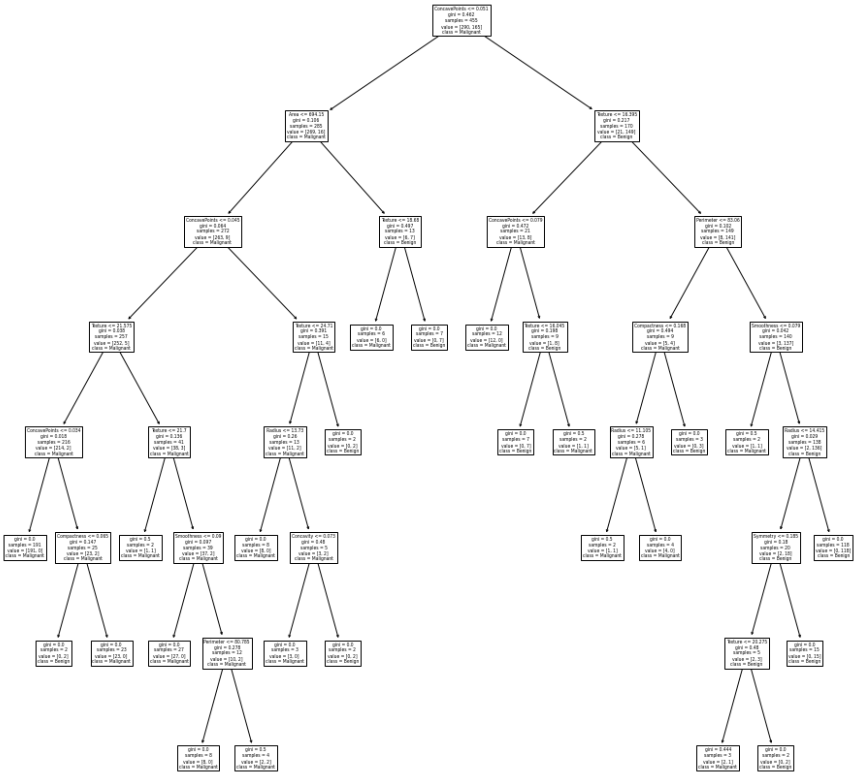
	Variable	Importancia
7	ConcavePoints	0.755958
1	Texture	0.116267
5	Compactness	0.031812
3	Area	0.031220
2	Perimeter	0.030337
6	Concavity	0.011823
0	Radius	0.009819
4	Smoothness	0.006852
8	Symmetry	0.005911
9	FractalDimension	0.000000

Nodo del que se parte.



El árbol generado presenta 7 niveles y el nodo raíz parte del perímetro con un score del 93.85%.

ConcavePoints es la variable que presenta un mayor nivel de entropía o que mayor importancia toma para el modelo, por el contrario, la menos importante es Dimensión Fractal.



Matriz de confusión del árbol.

Clasificación Real	Benign	Malignant
Benign	64	3
Malignant	4	43

Reporte de parámetros.

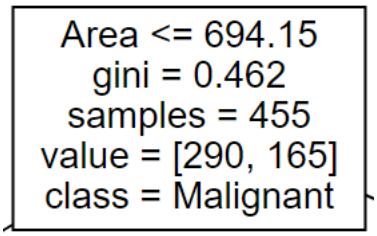
	precision	recall	f1-score	support
Benign	0.94	0.96	0.95	67
Malignant	0.93	0.91	0.92	47
accuracy			0.94	114
macro avg	0.94	0.94	0.94	114
weighted avg	0.94	0.94	0.94	114

Cervantes Rubí Brandon

Inteligencia Artificial 2022-2, G.3, 03/05/2022
Practica No.12 Clasificación (Árbol de decisión).

Árbol con las variables seleccionadas del análisis de correlación.

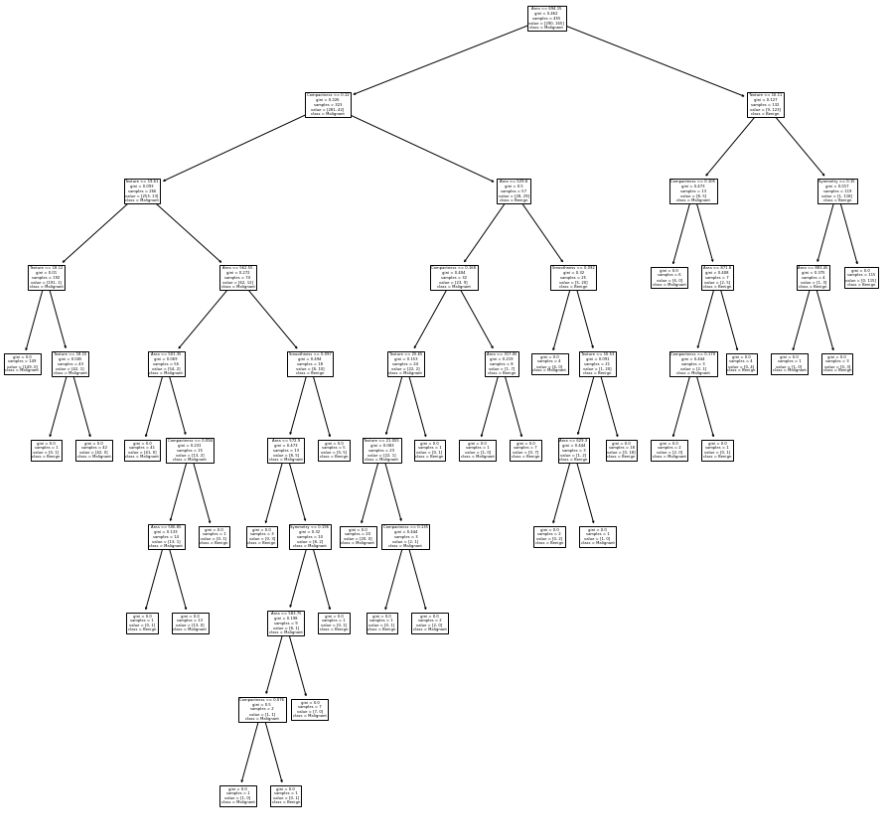
Nodo del que se parte.



	Variable	Importancia
1	Area	0.701193
3	Compactness	0.170925
0	Texture	0.076840
2	Smoothness	0.041983
4	Symmetry	0.009059
5	FractalDimension	0.000000

El árbol generado presenta 9 niveles y el nodo raíz parte del perímetro con un score del 92.98%.

El área es la variable que presenta un mayor nivel de entropía o que mayor importancia toma para el modelo, por el contrario, la menos importante es Dimensión Fractal.



Matriz de confusión del árbol.

Clasificación Real	Benign	Malignant
Benign	63	4
Malignant	4	43

Reporte de parámetros.

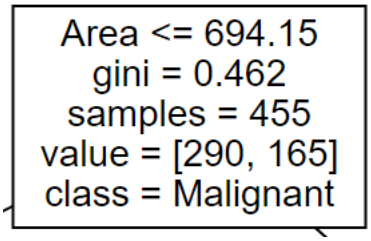
	precision	recall	f1-score	support
Benign	0.94	0.94	0.94	67
Malignant	0.91	0.91	0.91	47
accuracy			0.93	114
macro avg	0.93	0.93	0.93	114
weighted avg	0.93	0.93	0.93	114

Cervantes Rubí Brandon

Inteligencia Artificial 2022-2, G.3, 03/05/2022
Practica No.12 Clasificación (Árbol de decisión).

Árbol con las variables seleccionadas del análisis de correlación, altura máxima de 6 niveles con mínimo 4 registros en los nodos hoja y 2 nodos hoja.

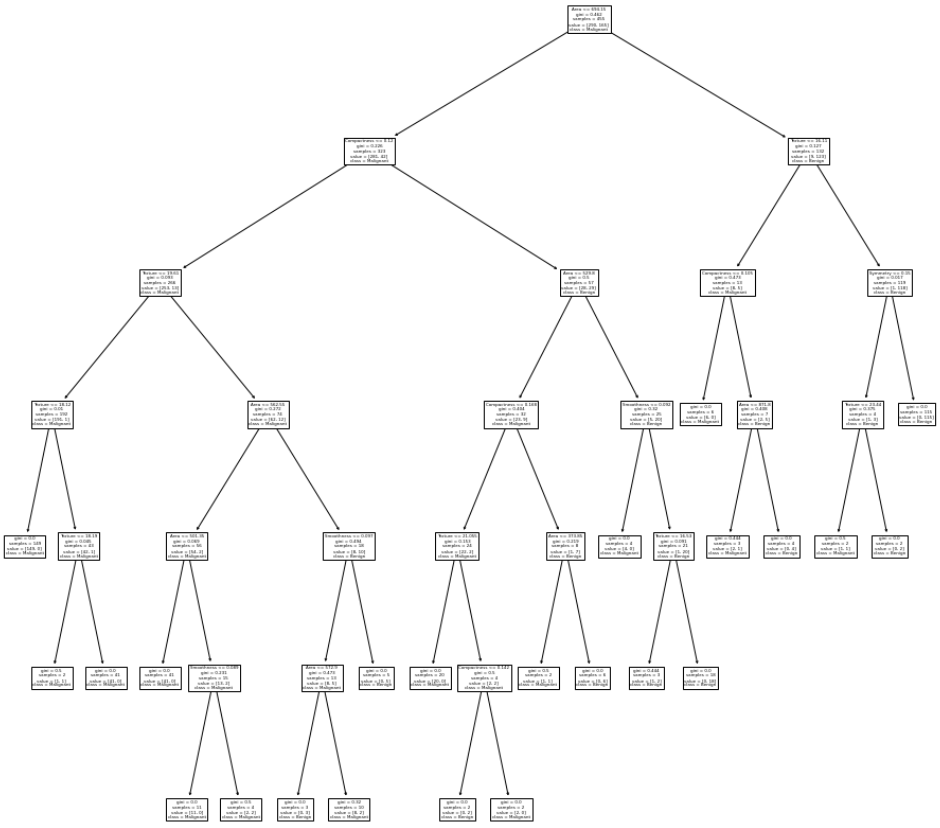
Nodo del que se parte.



	Variable	Importancia
1	Area	0.706966
3	Compactness	0.163812
0	Texture	0.075177
2	Smoothness	0.051623
4	Symmetry	0.002422
5	FractalDimension	0.000000

El árbol generado presenta 9 niveles y el nodo raíz parte del perímetro con un score del 91.23%.

El área es la variable que presenta un mayor nivel de entropía o que mayor importancia toma para el modelo, por el contrario, la menos importante es Dimensión Fractal.



Matriz de confusión del árbol.

Clasificación Real	Benign	Malignant
Benign	62	5
Malignant	4	43

Reporte de parámetros.

	precision	recall	f1-score	support
Benign	0.94	0.93	0.93	67
Malignant	0.90	0.91	0.91	47
accuracy			0.92	114
macro avg	0.92	0.92	0.92	114
weighted avg	0.92	0.92	0.92	114

De manera similar a como se hizo con la práctica de regresión nos ayudamos de `export_test` la cual permite generar una lista de reglas que definen la estructura de divisiones que va a tomar el árbol de decisión para hacer clasificaciones para nuevos registros, si se manda a imprimir podemos ver la estructura que toma el árbol desde otra perspectiva partiendo desde la variable más importante dependiendo de la configuración, en la imagen se muestra el árbol con una profundidad de 6, 4 registros en los nodos hoja y mínimo 2 nodos hoja de la página anterior.

```
|--- Area <= 694.15
|   |--- Compactness <= 0.12
|   |   |--- Texture <= 19.61
|   |   |   |--- Texture <= 18.12
|   |   |   |   |--- class: Benign
|   |   |   |--- Texture >  18.12
|   |   |   |   |--- Texture <= 18.19
|   |   |   |   |   |--- class: Benign
|   |   |   |   |--- Texture >  18.19
|   |   |   |   |   |--- class: Benign
|   |   |--- Texture >  19.61
|   |   |   |--- Area <= 562.55
|   |   |   |   |--- Area <= 501.35
|   |   |   |   |   |--- class: Benign
```

Y por último es posible estimar nuevos valores para el área del tumor dando como entrada un nuevo registro con la estructura de un dataframe al método `predict` de nuestro modelo de la siguiente manera, para el registro que se muestra en la parte baja se obtuvo un pronóstico de tumor maligno.

```
PacienteID1 = pd.DataFrame({'Texture': [10.38],
                            'Area': [1001.0],
                            'Smoothness': [0.11840],
                            'Compactness': [0.27760],
                            'Symmetry': [0.2419],
                            'FractalDimension': [0.07871]})
ClasificacionAD.predict(PacienteID1)
```

```
array(['Malignant'], dtype=object)
```

Conclusiones.

Durante esta práctica se trabajó con árboles de decisión para resolver un problema de clasificación relacionado a etiquetar a pacientes con tumores los cuales pueden ser malignos o benignos dependiendo de sus características. Para lograrlo se modelo el problema en términos de diferentes conjuntos de variables independientes, en una primera instancia con las 10 variables que el modelo tiene como base y después haciendo un análisis correlacional para eliminar variables que aporten lo mismo.

Además de modificar las variables modificamos los hiperparametros del modelo definiendo una altura máxima, un número mínimo de nodos hoja y un criterio de poda para eliminar aquellos nodos hoja que no cumplan con un mínimo de registros en ellos para su entrenamiento. Cada uno de los modelos generados se comparó en términos de su score y su matriz de confusión para llegar a aquel que logre generalizar mejor la información de entrada evitando así un sobreajuste para que el modelo seleccionado pueda tener un buen rendimiento con nuevos registros que es la finalidad principal al momento de crear estas herramientas.

Fuentes.

pandas documentation — pandas 1.4.1 documentation. (2022). Pandas. Recuperado 2022, de https://pandas.pydata.org/docs/index.html#

sklearn.metrics.classification_report. (2022). Scikit-Learn. Recuperado 2022, de https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

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