Cervantes Rubí Brandon

Objetivo.

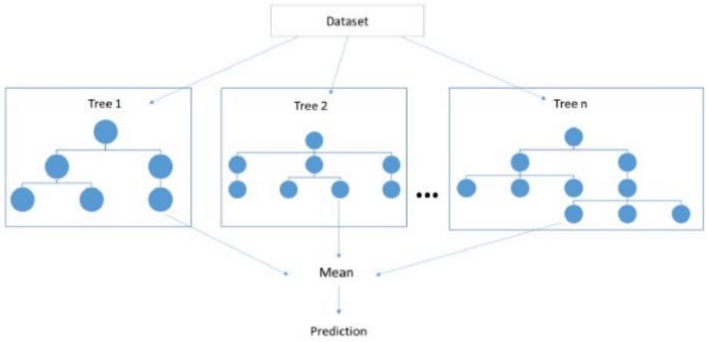
Clasificar registros clínicos de tumores malignos y benignos de cáncer de mama a partir de imágenes digitalizadas.

Características.

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

Los bosques aleatorios son algoritmos que buscan generalizar más las soluciones que nos pueden entregar los árboles aleatorios mediante la combinación de varios de estos árboles en la misma estructura de tal forma que cada uno de ellos aporte una solución similar para después llegar a un consenso evitando además un sobreajuste en los datos.

Al igual que en los árboles de decisión principalmente se trabajará con 2 tipos de estructuras para los bosques aleatorios, primero los bosques compuestos por árboles de clasificación los cuales trabajan con etiquetas como resultados (A, B, C), (0/1), etc. tomando aquella etiqueta que se repita más entre la salida de los árboles como la salida del modelo y después están los bosques compuestos por árboles de regresión, dichos arboles están diseñados para entregar como resultado un valor continuo y para los bosques se busca obtener un promedio con los resultados de cada árbol.



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.

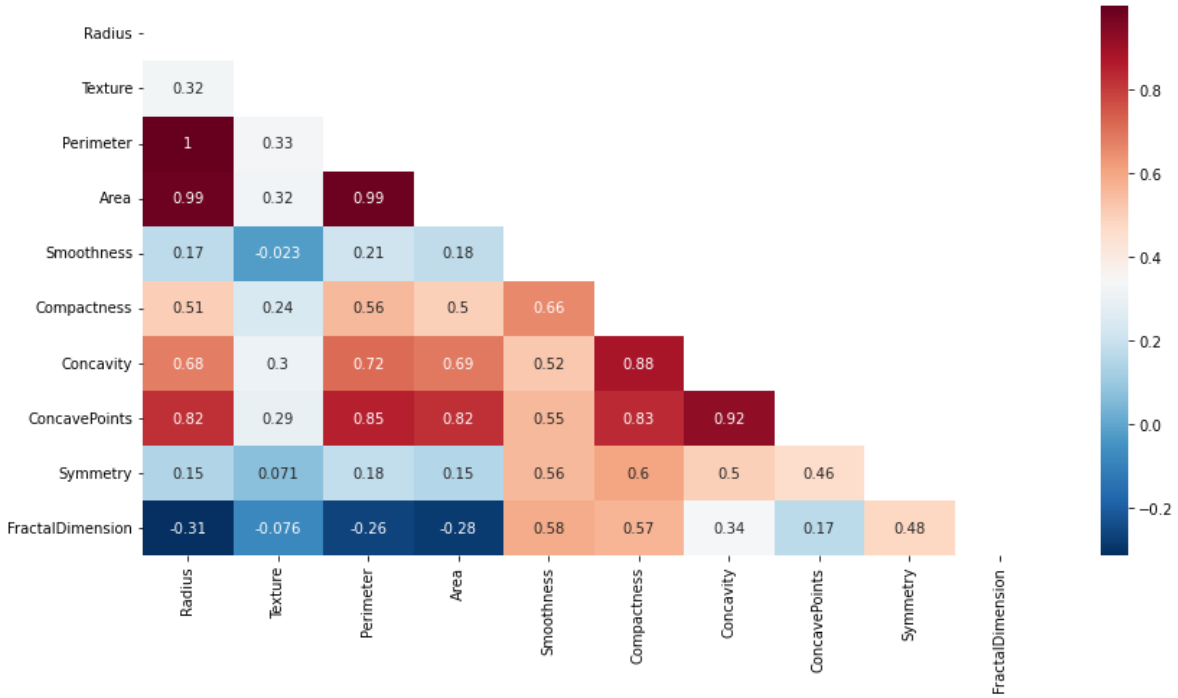
Después tenemos que hacer la lectura de nuestros datos los cuales están relacionados a las características de los tumores presentados en cada uno de las pacientes relacionadas como su área, textura y la etiqueta la cual nos indica si se trata de un tumor maligno o benigno que nos ayudara más adelante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

1.- Texture.

2.- Area.

3.- Smoothness.

4.- Compactness.

5.- Symmetry.

6.- FractalDimension.

```
#Variables predictoras
X = np.array(BCancer[['Texture',
                      'Area',
                      'Smoothness',
                      'Compactness',
                      'Symmetry',
                      'FractalDimension']])
```

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

Similar a la práctica de árboles de clasificación, 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 bosque aleatorio, similar a la práctica anterior, es necesario utilizar la clase `RandomForestClassifier` para crear un objeto a partir de ella sobre la que es posible definir ciertos parámetros de los árboles que conformaran el bosque, así como la cantidad. Nos daremos cuenta de que los parámetros trabajados en esta ocasión son los mismos de la practica anterior para el problema de regresion, 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 y `n_estimators` el cual define el número de árboles o estimadores que participaran para solucionar el problema.

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
ClasificacionBA = RandomForestClassifier(n_estimators=200, max_depth=8,
min_samples_split=4, min_samples_leaf=2, random_state=0)
ClasificacionBA.fit(X_train, Y_train)
Y_Clasificacion = ClasificacionBA.predict(X_validation)
```

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 posiblemente alto.

Para validarlo se utilizó el método de nuestro objeto creado a partir de `RandomForestClassifier` 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
ClasificacionBA.score(X_validation, Y_validation)
```

En un principio se evaluo el modelo con los hiperparametros por defecto, es decir sin una limitante en la altura o los nodos hoja lo cual nos arrojo un modelo con un 93.85% de precisión el cual es muy similar al obtenido por un solo árbol de decisión o la regresión logística variando solamente por décimas.

Se construyeron otras configuraciones modificando los hiperparametros relacionados al número de árboles que están en el bosque aleatorio, para una altura de máximo 8 niveles en el árbol, con un mínimo de 2 nodos hojas y 4 registros en cada uno de ellos a modo de podar los que no cumplan con este requisito.

Los resultados variando el número de árboles se muestran a continuación.

No. de arboles	50	100	150	200
Score (%)	92.98	92.1	92.98	91.22

Analizando esta tabla podemos ver que si bien el numero de estimadores se puede incrementar esto no significa que mientras mas arboles se consideren mejor será el modelo ya que el score se estabiliza cuando incrementa este número en este caso parece que rondara por el 92%, por otra parte, el incrementar el número de estimadores si conlleva un mayor costo computacional y tiempo de ejecución.

Una característica de estos bosques es que generalizan mejor los problemas de pronóstico y en este caso de clasificación, lo podemos apreciar ya que el índice de score no es tan cercano al 100% dejando ese 8% de incertidumbre para nuevos diagnosticos.

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 = ClasificacionBA.predict(X_validation)
Matriz_Clasificacion = pd.crosstab(Y_validation.ravel(),
                                   Y_Clasificacion,
                                   rownames=['Real'],
                                   colnames=['Clasificación'])
Matriz_Clasificacion
```

Similar a la práctica de árboles de clasificación se utilizó el método `classification_report` para obtener otras métricas que evalúan el desempeño del modelo; `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.

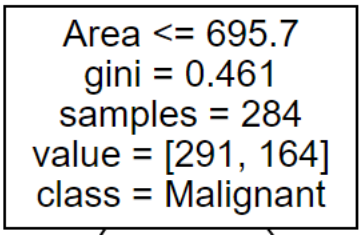
Para las tres configuraciones presentadas se generaron diferentes arboles partiendo de diferentes variables las cuales se detallan, pare generar las gráficas de los árboles y el conjunto de reglas fue necesario trabajar con los módulos `export_graphviz`, `plot_tree` y `export_text` de `sklearn.tree`.

Cervantes Rubí Brandon

Inteligencia Artificial 2022-2, G.3, 08/05/2022
Practica No.12 Clasificación (Bosques aleatorios).

Bosque aleatorio sin configurar los hiperparametros.

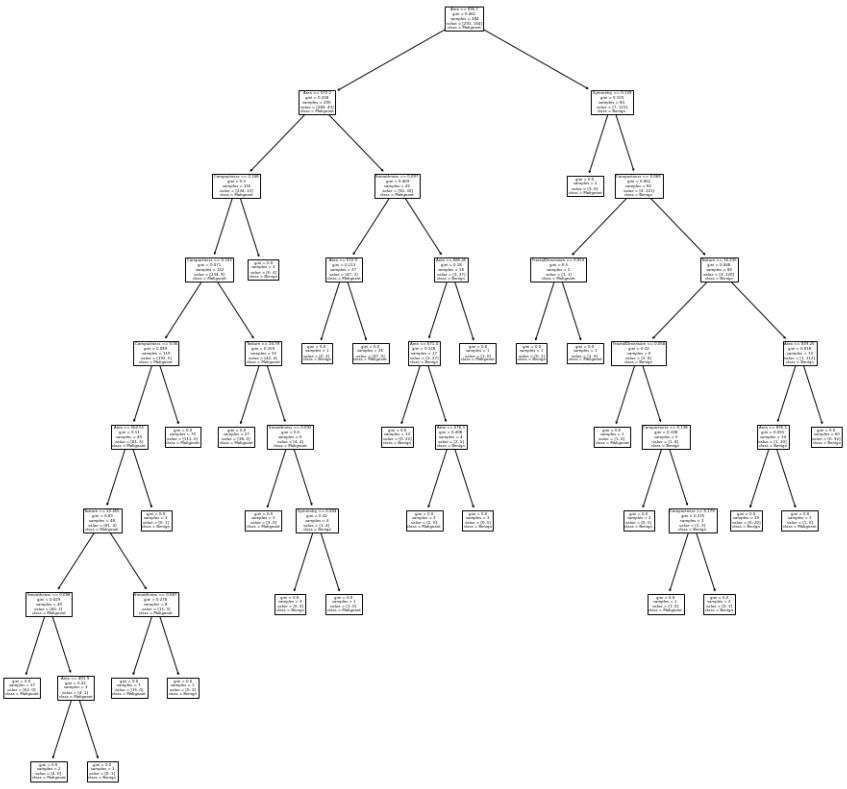
Nodo del que se parte.



	Variable	Importancia
1	Area	0.459875
3	Compactness	0.214218
0	Texture	0.126815
2	Smoothness	0.079435
5	FractalDimension	0.064920
4	Symmetry	0.054738

El árbol generado presenta 9 niveles y el nodo raíz parte del perímetro con un score del 93.85%.

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 simetría.



Matriz de confusión del árbol.

Clasificación / Real	Benign	Malignant
Benign	64	3
Malignant	4	43

Reporte de parámetros.

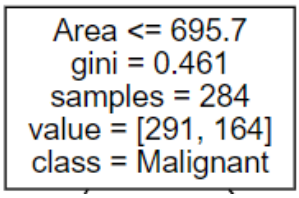
	precision	recall	f1-score	support
Benign	0.94	0.96	0.95	67
Malignant	0.93	0.91	0.92	47
accuracy			0.94	114
macro avg	0.94	0.94	0.94	114
weighted avg	0.94	0.94	0.94	114

Cervantes Rubí Brandon

Inteligencia Artificial 2022-2, G.3, 08/05/2022
Practica No.12 Clasificación (Bosques aleatorios).

Bosque aleatorio con 50 estimadores, una altura máxima de 8, mínimo 2 nodos hoja con 4 registros en cada una de ellas.

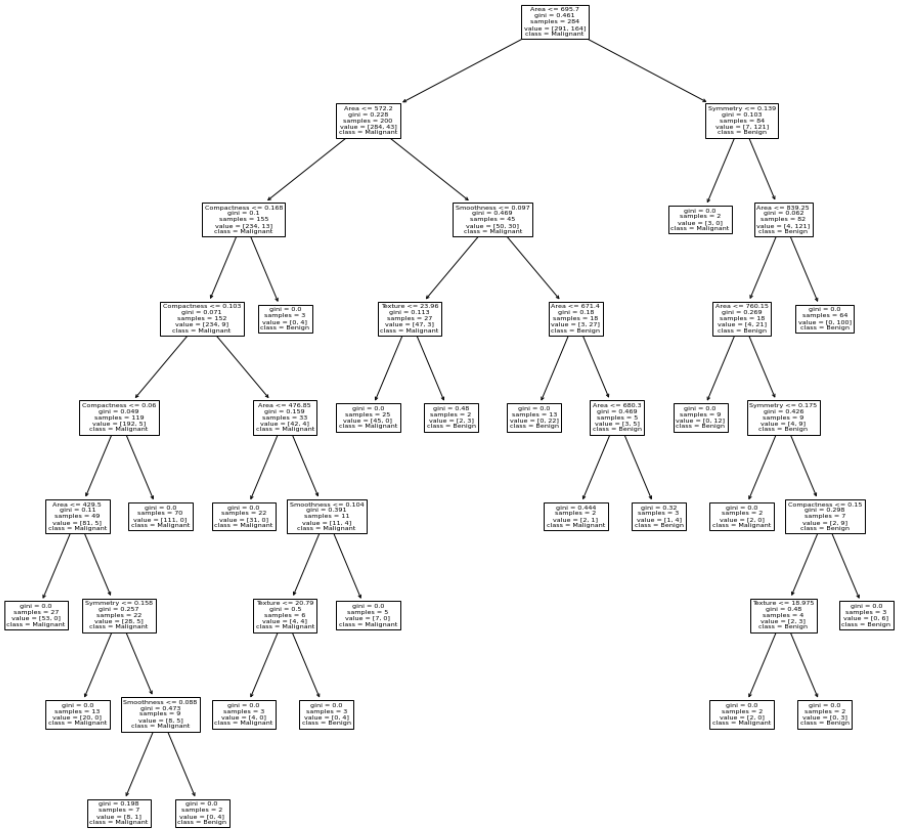
Nodo del que se parte.



	Variable	Importancia
1	Area	0.453440
3	Compactness	0.236248
0	Texture	0.119023
2	Smoothness	0.083636
5	FractalDimension	0.059683
4	Symmetry	0.047971

El árbol generado presenta 8 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 simetría.



Matriz de confusión del árbol.

Clasificación Real	Benign	Malignant
Benign	62	5
Malignant	3	44

Reporte de parámetros.

	precision	recall	f1-score	support
Benign	0.95	0.93	0.94	67
Malignant	0.90	0.94	0.92	47
accuracy			0.93	114
macro avg	0.93	0.93	0.93	114
weighted avg	0.93	0.93	0.93	114

Como ya se ha venido viendo podemos usar la librería `export_test` para generar una lista de reglas que definen la estructura de divisiones que va a tomar el árbol de decisión del bosque se seleccione para hacer clasificaciones con nuevos registros, si se manda a imprimir podemos ver la estructura que toma este árbol del bosque desde otra perspectiva partiendo desde la variable más importante dependiendo de la configuración, en la imagen se muestra el bosque con 50 árboles, una profundidad de 8, 4 registros en los nodos hoja y mínimo 2 nodos hoja de la página anterior.

```
|--- Area <= 695.70
|   |--- Area <= 572.20
|   |   |--- Compactness <= 0.17
|   |   |   |--- Compactness <= 0.10
|   |   |   |   |--- Compactness <= 0.06
|   |   |   |   |   |--- Area <= 429.50
|   |   |   |   |   |   |--- class: 0.0
|   |   |   |   |   |--- Area >  429.50
|   |   |   |   |   |   |--- Symmetry <= 0.16
|   |   |   |   |   |   |   |--- class: 0.0
|   |   |   |   |   |   |--- Symmetry >  0.16
|   |   |   |   |   |   |   |--- Smoothness <= 0.09
|   |   |   |   |   |   |   |   |--- class: 0.0
|   |   |   |   |   |   |   |--- Smoothness >  0.09
|   |   |   |   |   |   |   |   |--- class: 1.0
|   |   |   |   |--- Compactness >  0.06
|   |   |   |   |   |--- class: 0.0
|   |   |   |--- Compactness >  0.10
|   |   |   |   |--- Area <= 476.85
|   |   |   |   |   |--- class: 0.0
```

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.

```
#Paciente P-842302 (1) -Tumor Maligno-
PacienteID1 = pd.DataFrame({'Texture': [10.38],
                            'Area': [1001.0],
                            'Smoothness': [0.11840],
                            'Compactness': [0.27760],
                            'Symmetry': [0.2419],
                            'FractalDimension': [0.07871]})
ClasificacionBA.predict(PacienteID1)
```

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

Conclusiones.

A lo largo de esta practica se implementaron diferentes configuraciones de bosques aleatorios para lograr predecir el tipo de tumor del que un paciente padece tomando como base sus características mas importantes como el área, la textura, etc. dicha selección de características importantes se hizo mediante un análisis de correlación para reducir el numero inicial de variables y tener un mejor desempeño. Además, es posible generar varios bosques configurando los hiperparametros correctos, el primero y de los mas importantes es el numero de arboles de los cuales se va a componer el bosque, podemos además controlar las características de estos árboles como lo son la altura, y las características de los nodos hoja.

Cada una de las configuraciones generadas da como resultado diferentes métricas, aunque a medida que se aumentó el número de estimadores se puede ver que se estabiliza el score del modelo, sin embargo, al tener que usar mas arboles para llegar al mismo desempeño afectamos el tiempo que toma el algoritmo en ejecutarse por lo cual hay que tratar de llegar el mejor resultado con la menor cantidad de árboles generalizando de buena manera el problema dependiendo del contexto con el que se trabaje.

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)