Basile Álvarez Andrés José

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 2/12/21

Inteligencia Artificial

Grupo III

Semestre 2022-1

Reporte Práctica 13: Pronóstico (Bosques Aleatorios)

Objetivo: Pronosticar el área del tumor de pacientes con indicios de casos de cáncer de mama a través de bosques aleatorios.

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: Perímetro ^2 / 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)

Características Generales:

En esta práctica, utilizaremos el pronóstico por árboles de decisión, utilizando un bosque como una forma de generalización (utilizando varios árboles para mejorar el pronóstico). A esto se le conoce como Bosque Aleatorio, el cual es un algoritmo de aprendizaje automático de gran uso actualmente. Se supone que, al combinar los resultados, los errores se compensan con otros y se tiene una predicción que generaliza mejor al problema.

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-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

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

Una vez hecho lo anterior, se hace una descripción de los datos:

	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	Fracta
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: Parte de la descripción de datos con la función *describe()*.

Una vez hecho lo anterior, graficamos el área del tumor por paciente. En esta gráfica, notamos que hay una gran variabilidad entre los tamaños de tumores, lo cual nos puede llevar a que, al hacer el árbol de decisión, caigamos en sobreajuste.

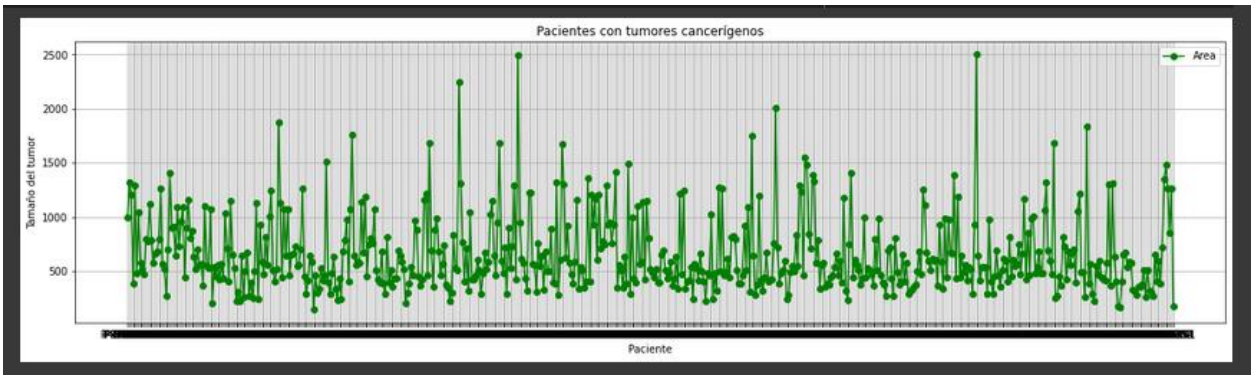

Figura 3: Gráfica de los tumores de acuerdo a su tamaño.

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.

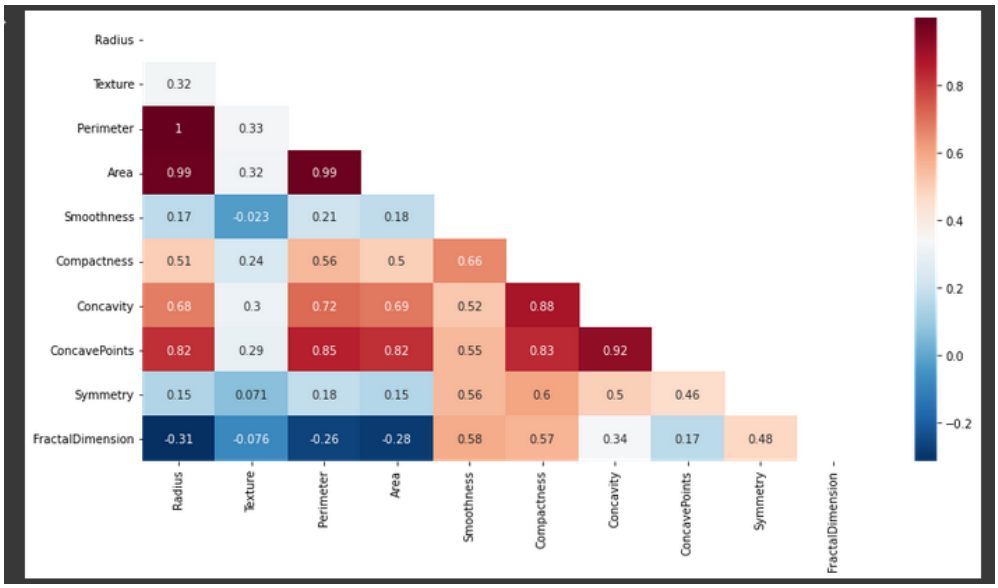

Figura 4: Matriz con las correlaciones entre variables.

Varibles 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]

*7) Perimeter [Posición 4] - Para calcular el área del tumor -

Figura 5: Variables seleccionadas

Para poder aplicar el algoritmo, importamos el *RandomForestRegressor,* las funciones *mean_squared_error*, *mean_absolute_error*, *r2_score* y *model_selection.* 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.

Se seleccionan las variables predictoras (X) y la variable a pronosticar (Y)

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

#X = np.array(BCancer[['Radius', 'Texture', 'Perimeter', 'Smoothness',
#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

Figura 6: 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.

```
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

Figura 7: Parte de la matriz con la variable clase.

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:

```
[13] X_train, 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 × 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

Figura 8: División de los datos.

Una vez hecho lo anterior, se genera el pronóstico de bosques aleatorios y comparamos con los valores reales:

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

	0
0	420.781134
1	348.203769
2	487.185543
3	266.820671
4	569.064344
...	...
109	418.142585
110	1080.452036
111	543.014357
112	554.268093
113	2105.021452

114 rows × 1 columns

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

	0
420.781134	416.2
348.203769	357.6
487.185543	476.7
266.820671	269.4
569.064344	568.9
...	...
418.142585	419.8
1080.452036	1094.0
543.014357	551.7
554.268093	565.4
2105.021452	2501.0

114 rows × 1 columns

Figura 9: Pronóstico con árboles aleatorios.

Obtenemos los parámetros del modelo, viendo que tenemos una exactitud de 0.9863. Vemos otros parámetros:

```
Criterio:
 squared_error
Importancia variables:
 [1.15182354e-03 9.94994772e-01 7.76652894e-04 1.24814848e-03
 8.55020149e-04 9.73582886e-04]
MAE: 16.7955
MSE: 1832.1701
RMSE: 42.8039
Score: 0.9863
```

Figura 10: Obtención de los parámetros del modelo

El error absoluto medio (MAE) del algoritmo es 16.79, que es alrededor de 2.5% de la media de todos los valores de la variable Área (654.88). Esto significa que el algoritmo realiza pronóstico muy aceptable. Además, se tiene un Score de 0.9863, el cual indica que el pronóstico del área del tumor se logrará con un 98.6% de efectividad. Por otro lado, los pronósticos del modelo final se alejan en promedio 42.8 (RMSE) unidades del valor real.

Luego, se obtuvo la importancia de cada una de las variables para nuestro modelo utilizando la función *feature_importances* para bosques aleatorios, obteniendo que el perímetro fue la variable con mayor importancia para nuestra clasificación con un 99.4%, mientras que la *smoothness* fue la menos importante, con un porcentaje del 0.7%.

	Variable	Importancia
1	Perimeter	0.994995
3	Compactness	0.001248
0	Texture	0.001152
5	FractalDimension	0.000974
4	Symmetry	0.000855
2	Smoothness	0.000777

Figura 11: Importancia de las variables para el bosque aleatorio.

Hecho lo anterior, importamos *graphviz,* un visualizador de árboles. Con esta biblioteca, mostramos el árbol de manera gráfica (uno de los árboles dentro del bosque aleatorio, en este caso el árbol con el índice 13):

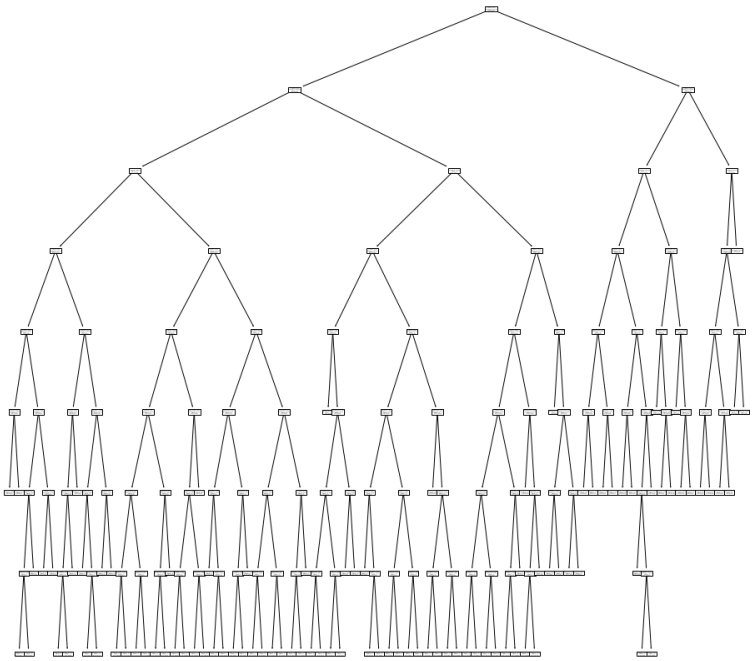

Figura 12: Árbol generado a partir de nuestro modelo de pronóstico para bosques aleatorios, en este caso el árbol con índice 13 dentro del bosque.

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], ‘Perimeter’: [122.8], 'Smoothness': [0.11840], 'Compactness': [0.27760], 'Symmetry': [0.2419], 'FractalDimension': [0.07871], obtenemos que el modelo pronostica un valor de área de 1028.757.

Y justamente eso es lo más importante de realizar este tipo de modelos: que su pronóstico pueda ser utilizada por un usuario (médico) para ingresar valores de los tumores de los pacientes y obtener un posible valor para el área del tumor.

Conclusiones

A lo largo de esta práctica, tuvimos como principal objetivo el obtener pronósticos del área de tumores utilizando árboles aleatorios. Al obtener los parámetros del modelo creado con bosques aleatorios, notamos que hay un peor desempeño en el pronóstico en comparación con el pronóstico que puede realizar un árbol de decisión (el cual tuvo efectividad del 99%, comparado con el 98% obtenido por el bosque aleatorio de esta práctica). No obstante, el pronóstico realizado por el bosque aleatorio tiene valores que son muy aceptables para hacer pronósticos y, en mi opinión, muy probablemente para otros conjuntos de datos se desempeñen mejor que un único árbol de decisión.

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. No obstante, esperaba un mejor resultado al utilizar bosques aleatorios en comparación con lo que se realizó para árboles de decisión (pronóstico).