Universidad Nacional
Autónoma de México

Inteligencia Artificial

Bosques Aleatorios (Pronóstico)

Grupo 3

Nombre:
Barreiro Valdez Alejandro

Práctica 13

Profesor: Dr. Guillermo Gilberto Molero Castillo

12 de mayo de 2022

Introducción

En esta práctica se utilizarán pronósticos de bosques aleatorios para predecir el área del tumor de pacientes con indicios de cáncer de mama. Se compararán los resultados con los obtenidos en prácticas pasadas utilizando pronósticos de un árbol. Además se obtendrá la estructura de un árbol que conforma parte del bosque para saber cómo se conforma.

Objetivos

Pronosticar el área del tumor de pacientes con indicios de casos de cáncer de mama a través de bosques aleatorios utilizando estudios clínicos a partir de imágenes digitalizadas de pacientes con cáncer de mama de Wisconsin (WDBC, Wisconsin Diagnostic Breast Cancer).

Desarrollo

Se importan las bibliotecas necesarias para la manipulación de datos.

```
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
```

Se importan los datos del cáncer de mama para utilizarlos.

```
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

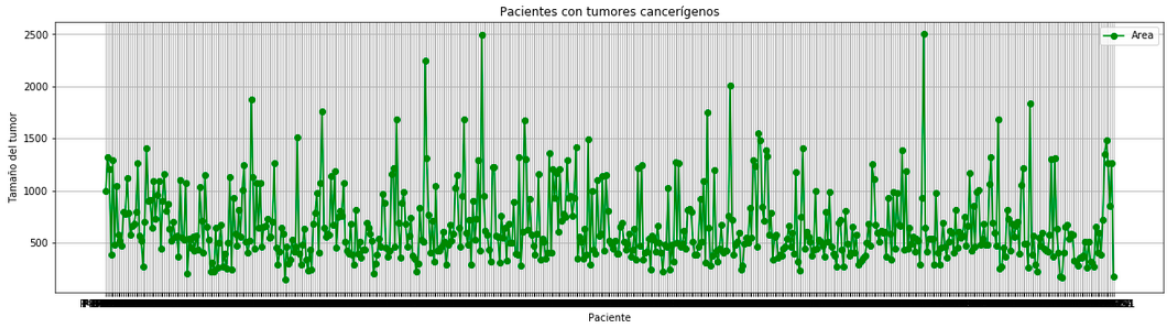
Se describen los datos que se utilizarán en el algoritmo para obtener las estadísticas generales.

```
BCancer.describe()
```

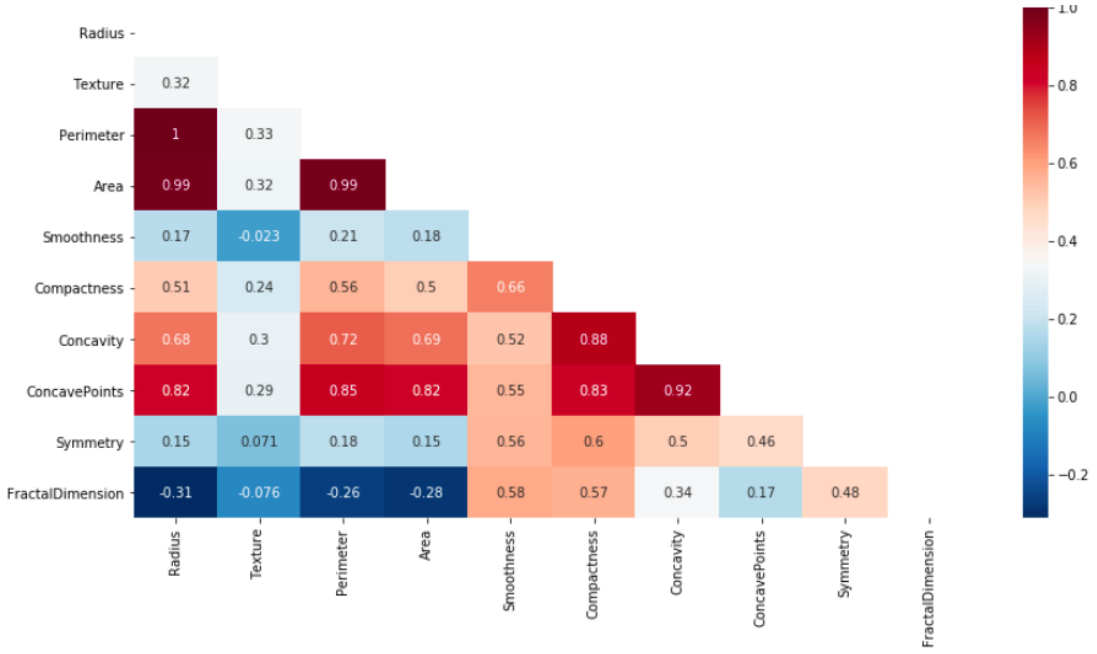
	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry	FractalDimension
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	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

Se realizó una gráfica del área de tumor por paciente para visualizar los datos.

```
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()
```



Se realiza la selección de variables utilizando un mapa de calor y las variables seleccionadas son: textura, área, smoothness, compactness, symmetry, FractalDimension y perímetro.



Se importan algunas bibliotecas para estadísticas sobre el modelo y la creación del bosque aleatorio.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn import model_selection
```

Se seleccionan las variables que serán las predictoras y las variables a predecir para generar *dataFrames* a partir de ellas. Los conjuntos de datos generados son los siguientes.

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

#X = np.array(BCancer[['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

```
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

Se generó la división de datos para separar los datos de entrenamiento y de prueba.

```
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

Se crea el bosque aleatorio utilizando la biblioteca de *sklearn*. La salida del modelo creado se ve de la siguiente manera.

```
PronosticoBA = RandomForestRegressor(n_estimators = 100, random_state=0, max_depth=8, min_samples_leaf=2, min_sample
PronosticoBA.fit(X_train, Y_train)

#PronosticoBA = RandomForestRegressor(n_estimators=100, max_depth=8, min_samples_split=4, min_samples_leaf=2, random
#PronosticoA.fit(X_train, Y_train)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=8, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=2,
                      min_samples_split=4, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

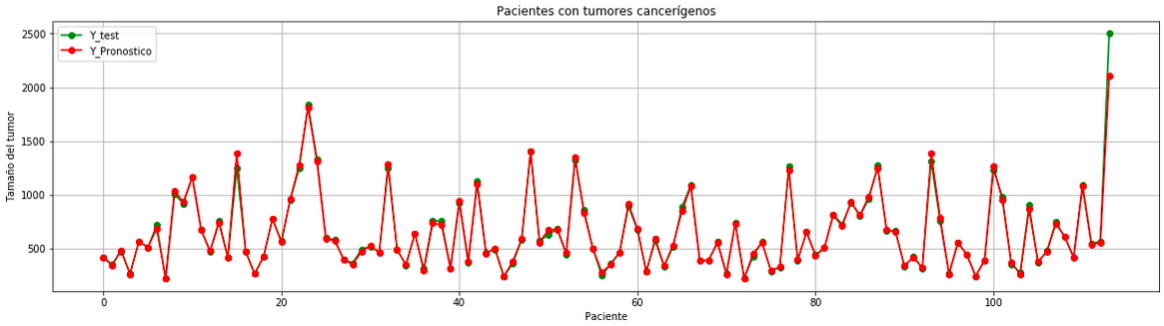
Se generan los pronósticos y los valores reales para poder obtener el *score* de los pronósticos. Con 100 estimadores se tiene un *score* de .986. Este *score* puede cambiar con los estimadores utilizados y con la selección de variables. Con 50 estimadores el *score* baja a .985 y con 200 estimadores cambia a .986 que es muy parecido al primer valor.

```
r2_score(Y_test, Y_Pronostico)
```

```
0.9862774348475016
```

Con la configuración de 100 estimadores se generó una gráfica para observar la diferencia entre los pronósticos generados por el modelo y los datos reales. La gráfica que se obtuvo fue la siguiente.

```
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()
```



También se obtuvieron los parámetros de este modelo los cuales se enseñan a continuación.

```
Criterio:
 mse
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
```

También, se obtuvo la importancia para el modelo de cada una de las variables que se utilizan. De estos datos se pudo obtener que la variable más importante para el modelo generado es el perímetro.

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

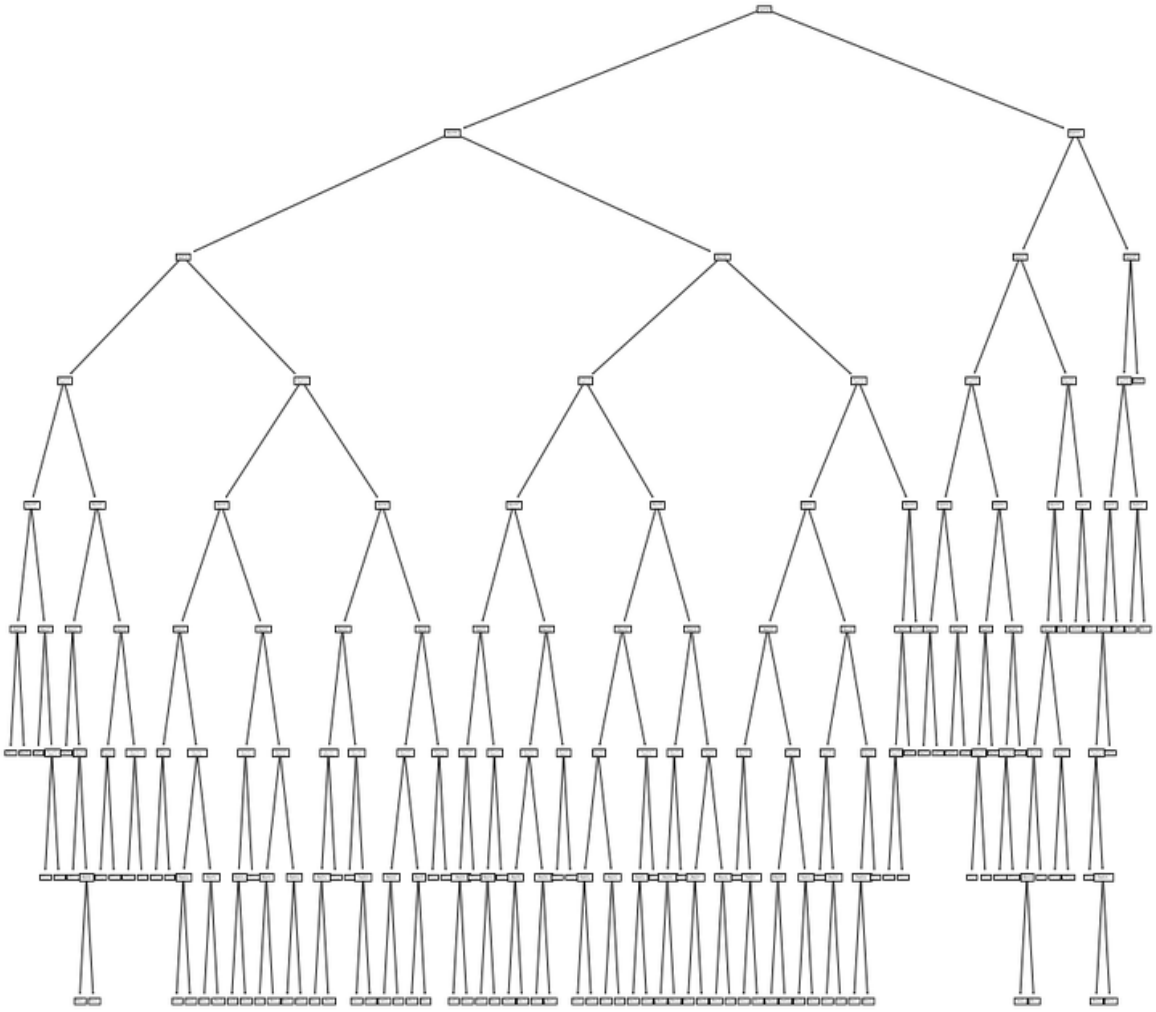
	Variable	Importancia
1	Perimeter	0.994995
3	Compactness	0.001248
0	Texture	0.001152
5	FractalDimension	0.000974
4	Symmetry	0.000855
2	Smoothness	0.000777

Después se grafica uno de los estimadores de los bosques para graficar y poder observar los datos que contiene y la manera en que trabaja.

```
Estimador = PronosticoBA.estimators_[99]
Estimador
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=8,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=2, min_samples_split=4,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=1396067212, splitter='best')
```

Se grafica el árbol de decisión como en prácticas pasadas y se tiene una versión donde se puede observar cada uno de los campos.



Además se generó una versión del modelo donde se describe dicho estimador utilizando texto. Este texto es bastante extenso, pero puede ser más claro que el diagrama.

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

```
|--- Perimeter <= 108.20
|   |--- Perimeter <= 81.49
|   |   |--- Perimeter <= 67.96
|   |   |   |--- Perimeter <= 59.23
|   |   |   |   |--- Perimeter <= 53.84
|   |   |   |   |   |--- Perimeter <= 49.84
|   |   |   |   |   |   |--- value: [179.90]
|   |   |   |   |   |--- Perimeter >  49.84
|   |   |   |   |   |   |--- value: [203.57]
|   |   |   |   |--- Perimeter >  53.84
|   |   |   |   |   |--- Perimeter <= 56.10
|   |   |   |   |   |   |--- value: [226.52]
|   |   |   |   |   |--- Perimeter >  56.10
|   |   |   |   |   |   |--- Perimeter <= 58.88
|   |   |   |   |   |   |   |--- value: [245.75]
|   |   |   |   |   |   |--- Perimeter >  58.88
|   |   |   |   |   |   |   |--- value: [261.93]
|   |   |   |--- Perimeter >  59.23
|   |   |   |   |--- Perimeter <= 64.43
```

Por último, como en prácticas anteriores, se genera una predicción para uno de los datos que se utilizan y predecir el área del tumor. La predicción que se generó es la siguiente.

```
AreaTumorID1 = pd.DataFrame({'Texture': [10.38],
                             'Perimeter': [122.8],
                             'Smoothness': [0.11840],
                             'Compactness': [0.27760],
                             'Symmetry': [0.2419],
                             'FractalDimension': [0.07871]})
PronosticoBA.predict(AreaTumorID1)
```

```
array([1028.75772738])
```

Se puede observar cómo se tiene menos sobreajuste ya que se tienen varios árboles en conjunto que permiten generar un modelo global. Cuando solo se tiene un árbol esto puede desencadenar en sobreajuste. Se recomienda utilizar bosques aleatorios en vez de un solo árbol.

Conclusiones

En esta práctica se logró utilizar bosques aleatorios para generar pronósticos para predecir el área de un tumor de pacientes de cáncer de mamá. Se pudo observar cómo generar bosques aleatorios utilizando Python y *sklearn*. Se ajustaron los diferentes parámetros para generar un bosque aleatorio de 100 árboles. Se compararon los resultados obtenidos con aquellos de un solo árbol y se concluyó que se tiene mejor *score* y que se tiene un modelo que no cae tan fácilmente en sobreajuste como los árboles. El bosque utiliza a los árboles como componentes y permite que se tengan modelos más generales. También se pudo obtener la estructura de uno de los árboles que conforma el bosque. La estructura de este árbol es similar a la vista en prácticas pasadas y la forma en que se obtiene es similar. Con esta práctica se pudo observar la utilidad y eficiencia de utilizar los bosques aleatorios para generar pronósticos y cómo estos son preferibles a solo utilizar un árbol.