

Basile Álvarez Andrés José

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 11/10/21

Inteligencia Artificial

Grupo III

Semestre 2022-1

Reporte Práctica 9: Clasificación con Regresión Logística

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

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: Desviación 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: $\text{Perímetro}^2 / \text{Área} - 1$
- 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+\(Diagnostic\)](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic))

Características Generales:

En esta práctica, utilizaremos la clasificación por regresión logística para predecir etiquetas que nos permitan conocer si un tumor es benigno o maligno. Para ello, construiremos un modelo a través de un conjunto de entrenamiento y utilizaremos un conjunto de prueba para evaluar dicho modelo.

La regresión logística es un tipo de aprendizaje supervisado cuyo objetivo es predecir valores binarios (0 o 1). Para este tipo de aprendizaje, se utiliza la función logística o sigmoide, la cual se muestra a continuación:

$$\text{Función logística} = \frac{1}{1 + e^{-(a+bX)}}$$

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 *RGeofisicos.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 x 12 columns

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

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 del mismo. Como se muestra en la figura 2, utilizamos el valor de *Diagnosis* como *hue* para el *pairplot* de correlación entre las variables. Además, 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.

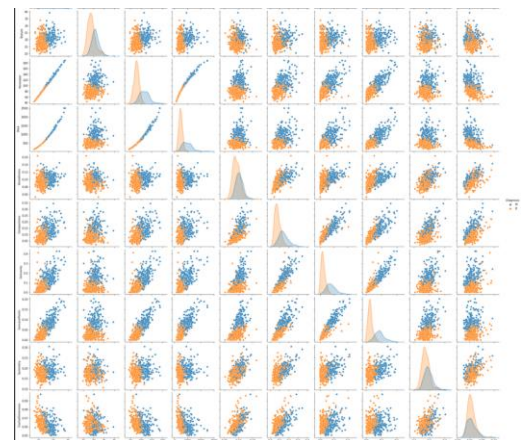


Figura 2: Pairplot con los datos de cáncer de mama utilizando *Diagnosis* como *Hue*.

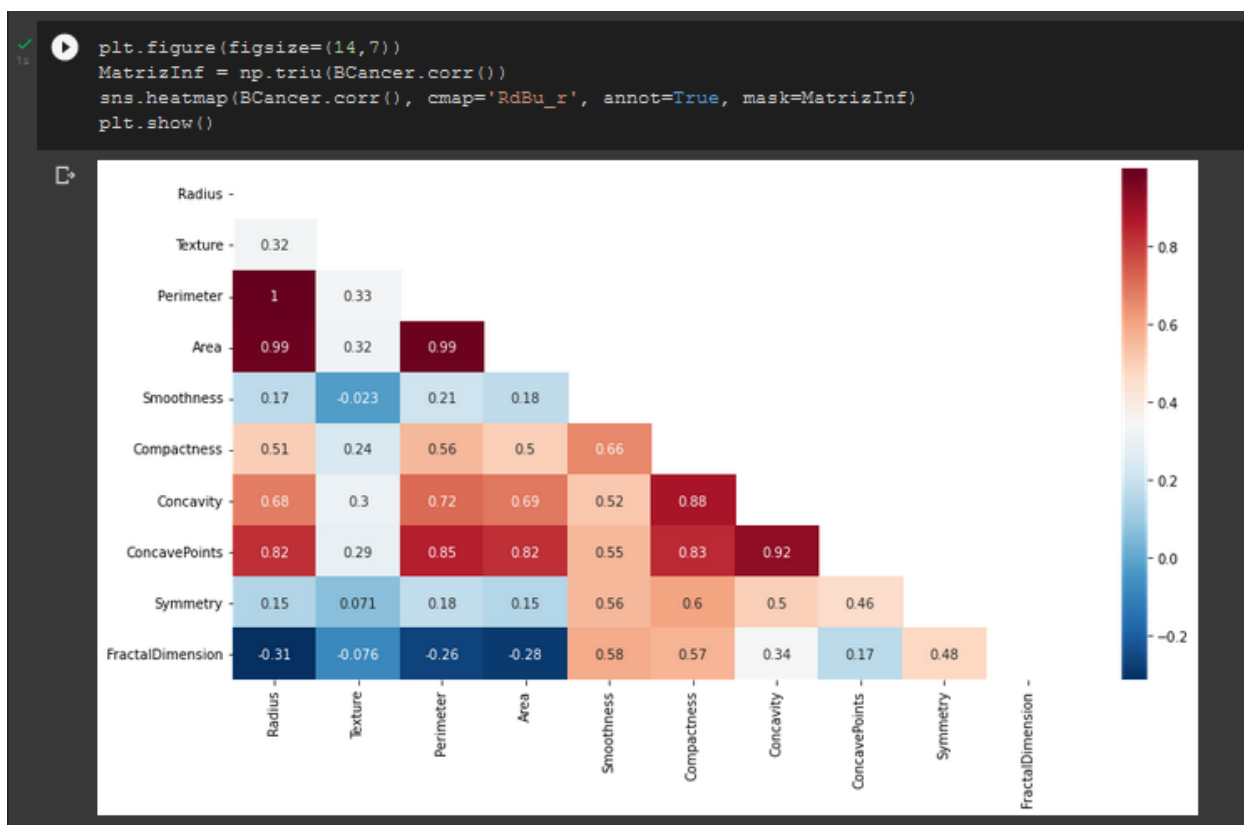


Figura 3: Matriz con las correlaciones entre variables.

Una vez hecho lo anterior, se seleccionan las variables: Texture, Area, Smoothness, Compactness, Symmetry y Fractal Dimension. Luego, tenemos que definir las variables predictoras y las variables clase. Para ello, primero cambiamos los valores del diagnóstico (maligno o benigno) de “M” y “B” a “0” y “1”, con la intención de trabajar con valores binarios para la clasificación.

```
[5] BCancer = BCancer.replace({'M': 0, 'B': 1})
BCancer
```

	IDNumber	Diagnosis	Radius	Texture	Perimeter	Area	Smoothness	Compactness	Concavity	ConcavePoints	Symmetry
0	P-842302	0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	P-842517	0	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	P-84300903	0	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	P-84348301	0	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	P-84358402	0	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...
564	P-926424	0	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	P-926682	0	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	P-926954	0	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	P-927241	0	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	P-92751	1	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows x 12 columns

Figura 4: Parte del DataFrame con el diagnóstico como valores binarios

Definimos entonces un *array* con las variables predictoras que nos permitirán predecir el valor del diagnóstico en la clasificación, como se muestra en la figura 5.

```
#Variables predictoras
X = np.array(BCancer[['Texture', 'Area', 'Smoothness', 'Compactness', 'Symmetry', 'FractalDimension']])
#X = BCancer.iloc[:, [3, 5, 6, 7, 10, 11]].values #iloc para seleccionar filas y columnas según su posición
pd.DataFrame(X)
```

	0	1	2	3	4	5
0	10.38	1001.0	0.11840	0.27760	0.2419	0.07871
1	17.77	1326.0	0.08474	0.07864	0.1812	0.05667
2	21.25	1203.0	0.10960	0.15990	0.2069	0.05999
3	20.38	386.1	0.14250	0.28390	0.2597	0.09744
4	14.34	1297.0	0.10030	0.13280	0.1809	0.05883
...
564	22.39	1479.0	0.11100	0.11590	0.1726	0.05623
565	28.25	1261.0	0.09780	0.10340	0.1752	0.05533
566	28.08	858.1	0.08455	0.10230	0.1590	0.05648
567	29.33	1265.0	0.11780	0.27700	0.2397	0.07016
568	24.54	181.0	0.05263	0.04362	0.1587	0.05884

569 rows x 6 columns

Figura 5: Parte de la matriz con las variables predictoras.

Además, definimos un vector con la variable clase o variable a predecir *Diagnosis*, como se muestra en la figura 6.

```
#Variable clase
Y = np.array(BCancer[['Diagnosis']])
pd.DataFrame(Y)
```

	0
0	0
1	0
2	0
3	0
4	0
...	...
564	0
565	0
566	0
567	0
568	1

569 rows x 1 columns

Figura 6: Parte de la matriz con la variable clase.

Además, realizamos un *scatter plot* con los valores de tumores malignos y benignos para observar su distribución según su textura y su área (se podría utilizar otras variables).

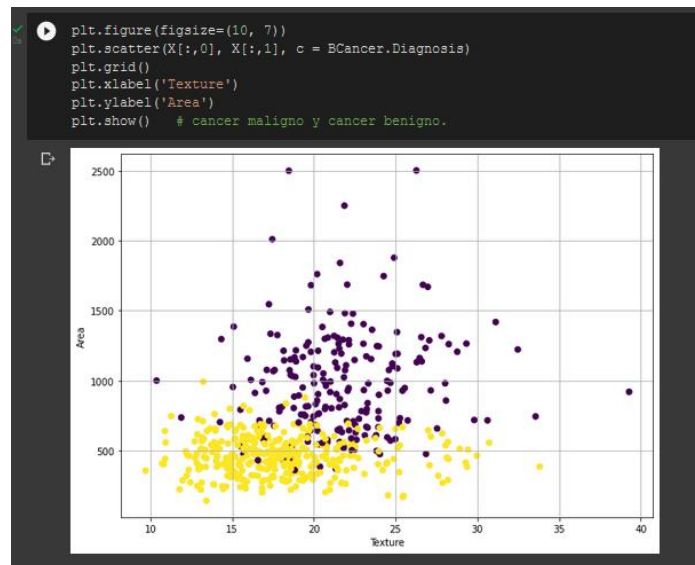


Figura 7: *Scatter plot* con los tumores malignos y benignos según su textura y su área.

Para la aplicación del algoritmo de regresión logística, primero tenemos que hacer la importación de las funciones de *linear_model* y *model_selection* de la biblioteca *sklearn* y *classification_report*, *confusion_matrix* y *accuracy_score* de *sklearn.metrics*, respectivamente.

Una vez hecho lo anterior, podemos utilizar la función *model_selection*, pasando como parámetros las variables predictoras y la variable clase, así como el tamaño del test (en un primer caso se utilizará un grupo de prueba del 20% de la cantidad total de datos), el *random state* (para fijar la posición de los pseudoaleatorios y reproducir los mismos resultados en distintas ejecuciones) y *shuffle* (con valor *True* para que los datos de entrenamiento y prueba sean elegidos de manera aleatoria en el conjunto de datos).

Utilizamos la función *Logistic_Regression* y *fit* para crear un objeto para hacer el entrenamiento con los datos de entrada definidos y crear el modelo de clasificación. Luego, generamos las predicciones probabilísticas de los datos de prueba, mostrando en la primera columna del DataFrame de la figura 8 la probabilidad del diagnóstico (si es mayor al 50%, el valor será de 1 y en caso contrario será de 0) y en la segunda columna mostraremos la exactitud de dicha probabilidad.

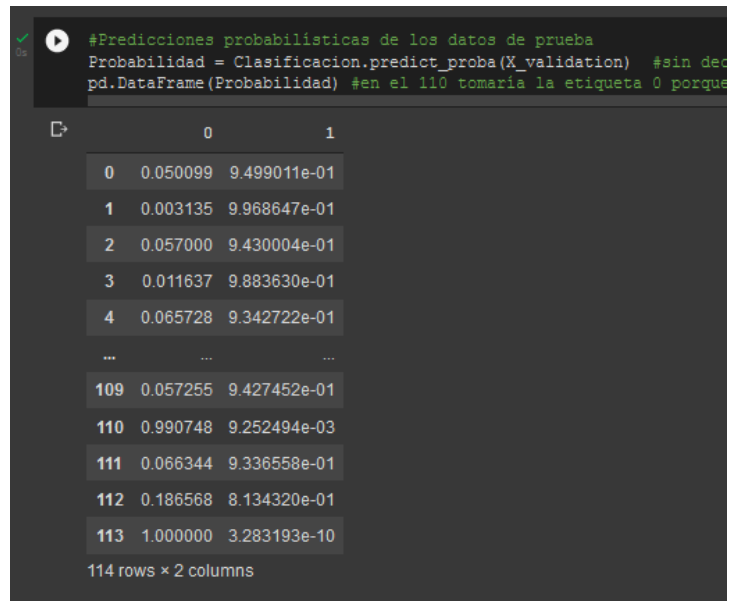


Figura 8: Probabilidades de los datos de prueba. La primera columna representa el valor de la probabilidad (si es mayor a .5, el valor será de 1 y será de 0 en caso contrario) y en la segunda columna se muestra la exactitud de dicha probabilidad.

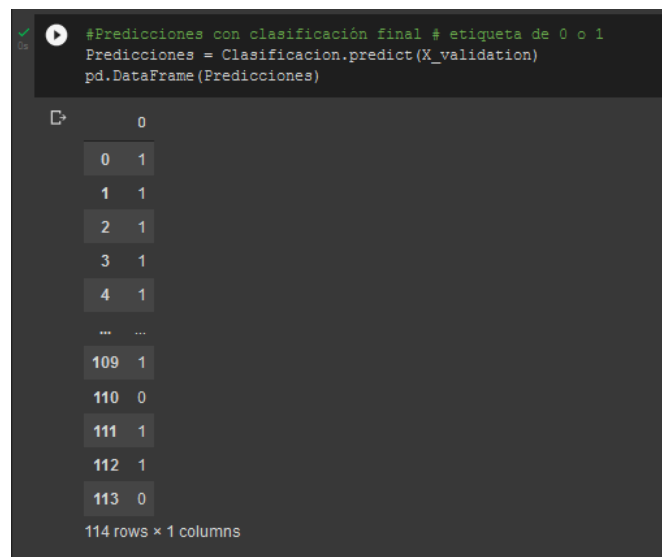
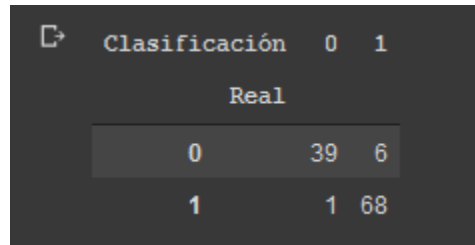


Figura 9: Predicciones con clasificación final (etiqueta de 0 o de 1).

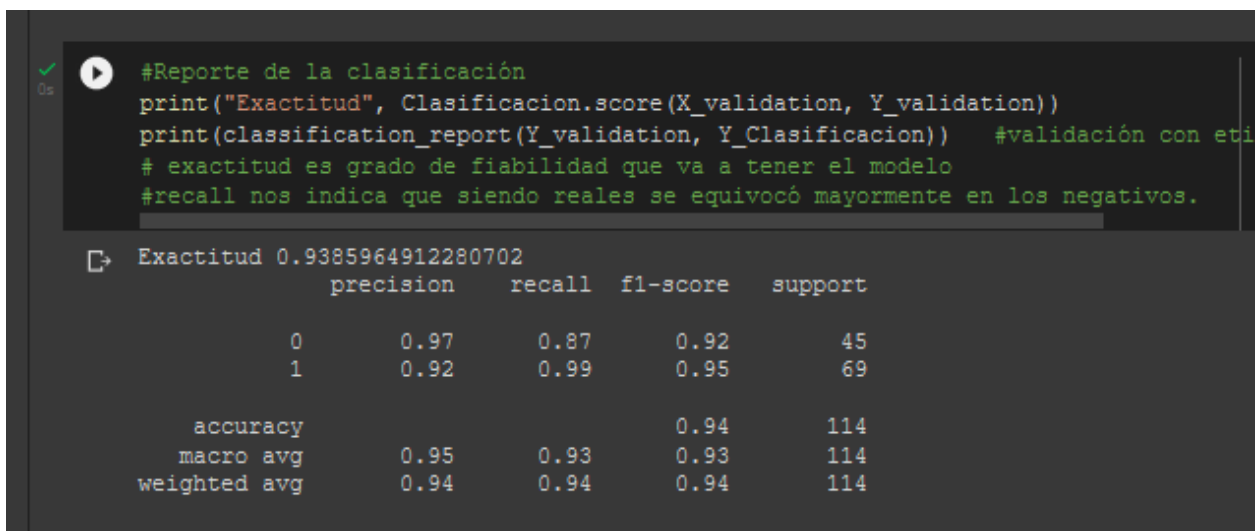
Luego, utilizamos la función *score* para calcular la exactitud promedio de la validación, obteniendo que, para un grupo de prueba del 20%, la exactitud promedio de la validación fue de 0.938596. Más tarde, utilizamos la matriz de clasificación vs el dato real para conocer qué datos de la clasificación realizada difieren de los datos reales (arrojando falsos positivos o falsos negativos).



	Clasificación 0	Clasificación 1
Real 0	39	6
Real 1	1	68

Figura 10: Matriz de clasificación para conocer los falsos positivos y falsos negativos.

Utilizamos la función *classification_report* para obtener un reporte de la clasificación recién realizada. En este reporte, obtenemos el valor de exactitud (grado de fiabilidad del modelo), así como otros datos como la precisión (qué tanto se va a reproducir la clasificación) o el *recall* (que en este caso nos indica que, siendo reales, se equivocó mayormente en los negativos).

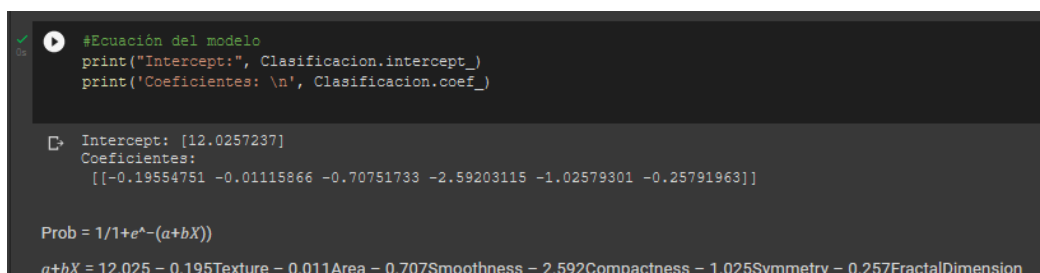


```
#Reporte de la clasificación
print("Exactitud", Clasificacion.score(X_validation, Y_validation))
print(classification_report(Y_validation, Y_Clasificacion)) #validación con et
# exactitud es grado de fiabilidad que va a tener el modelo
#recall nos indica que siendo reales se equivocó mayormente en los negativos.
```

Exactitud	0.9385964912280702			
	precision	recall	f1-score	support
0	0.97	0.87	0.92	45
1	0.92	0.99	0.95	69
accuracy			0.94	114
macro avg	0.95	0.93	0.93	114
weighted avg	0.94	0.94	0.94	114

Figura 11: Reporte de la clasificación.

Posteriormente, se obtiene la ecuación del modelo con la cual podremos predecir si un tumor es maligno o benigno para los datos de dicho tumor en cualquier paciente:



```
#Ecuación del modelo
print("Intercept:", Clasificacion.intercept_)
print('Coeficientes: \n', Clasificacion.coef_)
```

Intercept: [12.0257237]
 Coeficientes:
 [[-0.19554751 -0.01115866 -0.70751733 -2.59203115 -1.02579301 -0.25791963]]

Prob = 1/1+e^{-(a+bX)}

a+bX = 12.025 - 0.195Texture - 0.011Area - 0.707Smoothness - 2.592Compactness - 1.025Symmetry - 0.257FractalDimension

Figura 12: Ecuación del modelo que podremos aplicar para clasificar cualquier tumor como benigno o maligno.

Finalmente, realizamos el mismo procedimiento, pero para un valor de grupo de prueba diferente. En este caso, utilizaremos un grupo de prueba del 30%. Generando las probabilidades, tenemos que:

```
#Predicciones probabilísticas de los datos de prueba
Probabilidad = Clasificacion.predict_proba(X_validation) #
pd.DataFrame(Probabilidad) #en el 110 tomaria la etiqueta 0
```

	0	1
0	0.046909	0.953091
1	0.002439	0.997561
2	0.049871	0.950129
3	0.010857	0.989143
4	0.056630	0.943370
...
166	0.104829	0.895171
167	0.648838	0.351162
168	0.063938	0.936062
169	0.080063	0.919937
170	0.893648	0.106352

171 rows x 2 columns

Figura 13: Probabilidades para un grupo de prueba del 30%.

Para este grupo de prueba, se obtuvo una exactitud promedio de la validación de 0.8888 y una matriz de clasificación como se muestra en la figura 14.

Clasificación	0	1
Real		
0	52	14
1	5	100

Figura 14: Matriz de clasificación

```
Exactitud 0.8888888888888888
```

	precision	recall	f1-score	support
0	0.91	0.79	0.85	66
1	0.88	0.95	0.91	105
accuracy			0.89	171
macro avg	0.89	0.87	0.88	171
weighted avg	0.89	0.89	0.89	171

Figura 15: Reporte de la clasificación para grupo de prueba del 30%.

```
#Ecuación del modelo
print("Intercept:", Clasificacion.intercept_)
print('Coeficientes: \n', Clasificacion.coef_)
```

```
Intercept: [12.47173007]
Coeficientes:
[[-0.21394488 -0.01132925 -0.62040938 -2.37073435 -0.85512965 -0.24306719]]
```

Figura 16: Coeficientes para modelo de clasificación con grupo de prueba del 30%.

Conclusiones

A lo largo de esta práctica, tuvimos como principal objetivo el obtener la clasificación de registros clínicos de tumores malignos y benignos de cáncer de mama a partir de imágenes digitalizadas. Para ello, utilizamos el algoritmo de regresión logística con dos grupos de prueba diferentes (del 20% y del 30% del total del conjunto de datos) y utilizando 6 variables predictoras (textura, área, smoothness, compactness, symmetry, fractal dimension).

Al utilizar la regresión logística, logramos hacer una clasificación a partir de los registros médicos digitalizados y obtener un modelo matemático que nos permita encontrar la clasificación (benigno o maligno) de tumores a partir de ciertos datos (los de las variables predictoras).

Este tipo de algoritmo es muy utilizado en diversas aplicaciones. Si bien en esta práctica utilizamos este tipo de aprendizaje para datos médicos, esto se podría traducir a datos de todo tipo como para obtener clasificaciones para datos de experimentos científicos, o la clasificación de si un solicitante es considerado bueno o malo como para darle un crédito bancario.

Vimos que, para los distintos tamaños de grupos de prueba, obtuvimos valores diferentes en los modelos matemáticos resultantes. Observamos que, en este caso, la exactitud del modelo fue mayor para un grupo de prueba menor y que en general la precisión del modelo también fue mejor. Se podrían hacer pruebas para distintos tamaños y determinar cuál es el que genera el modelo que se ajuste de mejor manera a los datos con los que se cuenta, evitando caer en un ajuste excesivo o sobre-ajuste de los datos.

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.