Historia de la Regresión Logística

La regresión logística tiene sus raíces en la estadística y la teoría de probabilidades que se desarrollaron a lo largo del siglo XX. Aunque el concepto de modelar eventos binarios se remonta a las primeras aplicaciones de la estadística, fue en la década de 1940 que el uso de la regresión logística se popularizó. En esa época, el estadístico David Cox introdujo el modelo de regresión logística como una forma de manejar problemas de clasificación y análisis de supervivencia. Su trabajo proporcionó una base sólida para el desarrollo de técnicas que permiten analizar y predecir la ocurrencia de eventos en el ámbito médico y social.

La importancia de la regresión logística comenzó a vislumbrarse especialmente en el ámbito de la salud pública. Durante las décadas de 1950 y 1960, los investigadores se dieron cuenta de que podían utilizar esta técnica para identificar factores de riesgo asociados a enfermedades. Por ejemplo, se realizaron estudios sobre el tabaquismo y su relación con el cáncer de pulmón, donde la regresión logística ayudó a cuantificar la probabilidad de desarrollar la enfermedad en función de variables como la cantidad de cigarrillos consumidos y la duración del hábito.

Con el avance de la tecnología y el acceso a grandes volúmenes de datos, la regresión logística se expandió a otros campos. En la década de 1980, se empezaron a utilizar computadoras personales para el análisis estadístico, lo que facilitó la aplicación de modelos de regresión logística en el ámbito empresarial, particularmente en marketing y estudios de mercado. Las empresas comenzaron a utilizar esta técnica para predecir el comportamiento del consumidor, evaluar la efectividad de campañas publicitarias y analizar la lealtad del cliente, lo que llevó a una comprensión más profunda de cómo los consumidores toman decisiones.

A medida que la estadística se consolidó como disciplina, la regresión logística ganó protagonismo en múltiples campos. Durante las décadas de 1960 y 1970, los investigadores comenzaron a aplicar este modelo en la investigación biomédica para entender las relaciones entre las variables clínicas y los resultados de salud. Esto incluyó la identificación de factores de riesgo para enfermedades y la evaluación de tratamientos médicos. La regresión logística se convirtió en una herramienta crucial para los epidemiólogos, quienes la utilizaron para analizar la efectividad de intervenciones en salud pública.

En los años posteriores, con el auge de la informática y la capacidad de procesar grandes volúmenes de datos, la regresión logística fue adoptada por otras disciplinas, como la economía y las ciencias sociales. Se utilizó en la evaluación de políticas públicas, análisis de comportamiento del consumidor y estudios sobre educación y empleo. El desarrollo de software estadístico accesible facilitó aún más su uso, permitiendo a los investigadores no solo aplicar el modelo, sino también interpretar sus resultados de manera más efectiva. Hoy en día, la regresión logística sigue siendo un pilar en el análisis de datos, adaptándose a nuevas metodologías y ampliando su alcance en el análisis predictivo y la inteligencia artificial.

Introducción a la Regresión Logística

La regresión logística es un método estadístico utilizado para modelar una variable dependiente binaria. Es decir, se utiliza para predecir la probabilidad de ocurrencia de un evento categórico (como éxito/fallo, sí/no, positivo/negativo) en función de una o más variables independientes

(predictoras). Este tipo de análisis es esencial en diversas áreas, incluyendo la medicina, la economía, la psicología y el marketing, donde es fundamental entender cómo diferentes factores influyen en resultados discretos.

A diferencia de la regresión lineal, que modela una variable continua, la regresión logística se enfoca en variables de resultado binario. Este enfoque permite modelar la relación no solo entre variables independientes, sino también la naturaleza de la variable dependiente, que en muchos casos se presenta como la ocurrencia o no de un evento. Esto es crucial en campos como la biomedicina, donde los investigadores a menudo se enfrentan a preguntas sobre la efectividad de tratamientos y la probabilidad de recuperación de los pacientes en función de múltiples factores.

La regresión logística no solo se limita a la predicción, sino que también ofrece interpretaciones valiosas sobre las relaciones entre las variables. Por ejemplo, en un estudio sobre la efectividad de un nuevo fármaco, los investigadores pueden utilizar la regresión logística para determinar cómo diferentes factores, como la dosis administrada, la edad y el estado de salud previo de los pacientes, influyen en la probabilidad de respuesta al tratamiento. Esta capacidad de estimar probabilidades permite que los tomadores de decisiones evalúen riesgos y beneficios de manera más efectiva, optimizando así estrategias en diversos contextos.

Además, la regresión logística es especialmente valiosa porque no solo ayuda a predecir resultados, sino que también proporciona información sobre la fuerza y dirección de las relaciones entre las variables. Los coeficientes obtenidos del modelo pueden ser interpretados como la influencia que cada variable independiente tiene sobre la variable dependiente. Por lo tanto, es un método poderoso para el análisis exploratorio y confirmatorio, permitiendo a los investigadores formular hipótesis y teorías basadas en datos empíricos. Su capacidad para manejar variables categóricas y continuas la convierte en una herramienta flexible en el análisis estadístico.

Otro aspecto relevante es la extensión de la regresión logística a modelos más complejos, como la regresión logística multinomial y la regresión logística ordinal, que permiten manejar situaciones en las que la variable dependiente tiene más de dos categorías. Estos modelos han ampliado las aplicaciones de la regresión logística en áreas como la clasificación de textos, análisis de encuestas, y estudios de comportamiento en línea.

Fundamentos de la Regresión Logística

La regresión logística es una técnica estadística clave que se utiliza para modelar la relación entre una variable dependiente binaria y una o más variables independientes. Su versatilidad la hace esencial en múltiples disciplinas, incluyendo la medicina, la economía, el marketing y las ciencias sociales. Al permitir predecir la probabilidad de que ocurra un evento específico, la regresión logística no solo facilita la toma de decisiones informadas, sino que también ayuda a comprender las relaciones subyacentes entre las variables involucradas.

Odds

El concepto de **odds** es fundamental en la regresión logística. Las odds representan la relación entre la probabilidad de que ocurra un evento y la probabilidad de que no ocurra. Se definen matemáticamente como:

$$\text{Odds}(Y=1)=\frac{P(Y=1)}{P(Y=0)}=\frac{P(Y=1)}{1-P(Y=1)}$$

Donde:

- $P(Y=1)$ es la probabilidad de que ocurra el evento de interés (por ejemplo, un resultado positivo).
- $P(Y=0)$ es la probabilidad de que no ocurra el evento (resultado negativo).

Por ejemplo, si una probabilidad de éxito es del 70% ($P(Y=1)=0.7$), las odds se calculan como:

$$\text{Odds}(Y=1)=\frac{0.7}{0.3}\approx 2.33$$

Esto significa que las odds de que ocurra el evento son aproximadamente 2.33 veces mayores que las odds de que no ocurra. El uso de odds es útil ya que permite trabajar con valores que pueden ser tanto positivos como negativos en análisis posteriores, además de ofrecer una forma más intuitiva de comprender la relación entre las probabilidades.

Función Sigmoide

La regresión logística utiliza la función sigmoide para transformar cualquier valor real en un rango entre 0 y 1, lo que la convierte en una herramienta ideal para modelar probabilidades. La función sigmoide se define matemáticamente como:

$$\sigma(z)=\frac{1}{1+e^{-z}}$$

Donde:

- z es la combinación lineal de las variables independientes, expresada como:
  $$z=\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n$$
  En esta ecuación, β_0 es el término independiente (intercepto) y $\beta_1, \beta_2, \ldots, \beta_n$ son los coeficientes asociados a cada variable independiente $X_1, X_2, \ldots, X_n$.

La función sigmoide es crucial ya que garantiza que las probabilidades estimadas siempre estén en el intervalo [0, 1]. Su forma característica permite observar cómo pequeñas variaciones en las variables independientes pueden influir en la probabilidad de que ocurra el evento. Esta propiedad es esencial para la interpretación de los resultados del modelo.

Singularidad en la Función Sigmoide

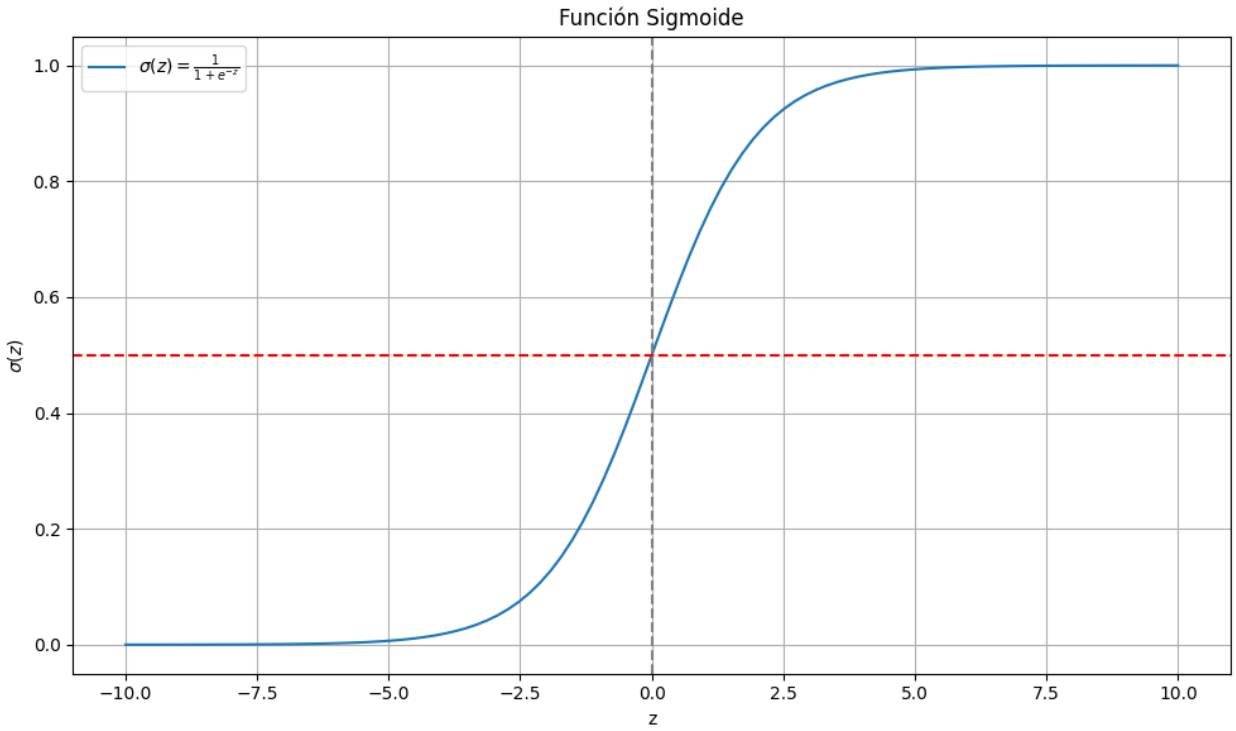
Un aspecto fundamental de la función sigmoide es su singularidad en el punto donde $\sigma(z)=0.5$. Esto ocurre cuando $z=0$, representando un umbral de decisión crítico para clasificar una observación. En este punto, las probabilidades de que ocurra el evento y de que no ocurra son iguales, lo que introduce un balance en la clasificación. Si el valor de z es mayor que 0, la función sigmoide devuelve una probabilidad mayor que 0.5, sugiriendo que el evento es más probable de ocurrir. Por el contrario, si z es menor que 0, la probabilidad cae por debajo de 0.5, indicando que el evento es menos probable.

En la práctica, este umbral puede ajustarse según el contexto del problema. Por ejemplo, en situaciones donde es crítico minimizar los falsos negativos (como en diagnósticos médicos), se puede elegir un umbral más bajo para asegurar que más casos positivos sean identificados, incluso a costa de aumentar el número de falsos positivos. Por otro lado, si el objetivo es minimizar los falsos positivos (como en la detección de fraudes), el umbral puede elevarse. Esta flexibilidad en el ajuste del umbral es fundamental para adaptarse a las prioridades y las consecuencias de los errores de clasificación.

Visualización de la Función Sigmoide

A continuación, se presenta un código que genera la gráfica de la función sigmoide. La visualización de esta función es crucial para entender cómo las variables predictoras influyen en la probabilidad del evento.

```
import numpy as np
import matplotlib.pyplot as plt

# Definir la función sigmoide
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Crear un rango de valores z
z = np.linspace(-10, 10, 100)

# Calcular los valores de la función sigmoide para z
sigma_z = sigmoid(z)

# Crear la gráfica con tamaño personalizado
plt.figure(figsize=(10, 6))

# Graficar la función sigmoide
plt.plot(z, sigma_z, label=r'$\sigma(z) = \frac{1}{1 + e^{-z}}$')
plt.axvline(0, color='grey', linestyle='--', linewidth=1.5)
plt.axhline(0.5, color='red', linestyle='--', linewidth=1.5)
plt.title('Función Sigmoide')
plt.xlabel('z')
plt.ylabel(r'$\sigma(z)$')
plt.grid(True)

# Colocar la leyenda en la parte superior izquierda
plt.legend(loc='upper left')

# Ajustar la disposición para evitar recortes en la gráfica
plt.tight_layout()

# Mostrar la gráfica
plt.show()
```



La visualización revela cómo los valores de z afectan la probabilidad. Para $z=0$, se tiene que $\sigma(z)=0.5$, lo que indica un punto de equilibrio entre los dos posibles resultados. Este umbral es crucial para la clasificación de los eventos, dividiendo la región de alta probabilidad de ocurrencia del evento de la región de baja probabilidad. A medida que z se vuelve más positivo, la probabilidad de que ocurra el evento se incrementa, y a medida que se vuelve más negativo, la probabilidad disminuye. Esta propiedad de la función sigmoide permite interpretar la regresión logística de manera más intuitiva.

Modelo de Regresión Logística

El modelo de regresión logística se expresa como:

$$P(Y=1 \vee X)=\sigma(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n)$$

Donde:

- $P(Y=1 \vee X)$ es la probabilidad de que ocurra el evento de interés (por ejemplo, un resultado positivo).
- β_0 es el término independiente, que representa el log-odds del evento cuando todas las variables independientes son cero.
- $\beta_1, \beta_2, \ldots, \beta_n$ son los coeficientes que reflejan la influencia de cada variable independiente en la probabilidad del evento. Cada coeficiente indica el cambio en el logaritmo de las odds del evento con respecto a un cambio unitario en la variable independiente correspondiente.

La regresión logística asume que la relación entre las variables independientes y el logaritmo de las odds (log-odds) de la variable dependiente es lineal. Esto permite aplicar técnicas de

regresión para estimar los parámetros del modelo, manteniendo la capacidad de hacer inferencias sobre la significancia de cada variable.

Interpretación de los Coeficientes

La interpretación de los coeficientes en la regresión logística es crucial para entender el impacto de cada variable predictora sobre la probabilidad del evento. Cada coeficiente β_i se interpreta de la siguiente manera:

- **Cambio en Odds**: Un incremento de una unidad en X_i cambia las odds del evento en un factor de e^{β_i}. Este factor de cambio se refiere al efecto multiplicativo que tiene el predictor sobre las odds.
- **Signo del Coeficiente**:
  - Si $\beta_i > 0$: Un aumento en X_i incrementa las odds del evento, aumentando así la probabilidad de que ocurra.
  - Si $\beta_i < 0$: Un aumento en X_i disminuye las odds del evento, reduciendo la probabilidad de que ocurra.

Por ejemplo, si $\beta_1 = 0.5$, el efecto se interpreta como que un aumento de una unidad en X_1 resulta en un incremento de las odds del evento en un 65% (ya que $e^{0.5} \approx 1.65$). Por otro lado, si $\beta_2 = -0.3$, esto sugiere que un aumento de una unidad en X_2 resulta en una disminución de las odds en un 26% (ya que $e^{-0.3} \approx 0.74$).

Valores Atípicos (Outliers) y su Influencia en la Regresión Logística

Los valores atípicos (outliers) son observaciones que se desvían significativamente de la tendencia general del conjunto de datos. Estos pueden influir de manera notable en el rendimiento del modelo de regresión logística. A continuación, se presentan algunos aspectos clave sobre cómo los outliers afectan a este tipo de análisis:

1. **Influencia en los Coeficientes**: Los outliers pueden alterar las estimaciones de los coeficientes de regresión, llevando a interpretaciones erróneas sobre la relación entre las variables. Un solo valor atípico con un alto nivel de influencia puede cambiar drásticamente el signo y la magnitud de un coeficiente, lo que compromete la validez del modelo.

2. **Ajuste del Modelo**: La presencia de outliers puede afectar el ajuste general del modelo, resultando en una mayor variabilidad en las predicciones y un rendimiento deficiente en datos no observados. Esto es especialmente problemático si el modelo se utiliza para tomar decisiones críticas basadas en las predicciones.

3. **Detección de Outliers**: Es esencial implementar técnicas para detectar y manejar outliers antes de ajustar un modelo de regresión logística. Métodos como el análisis de residuos, boxplots y la distancia de Cook son herramientas útiles para identificar observaciones influyentes. La distancia de Cook, por ejemplo, mide la influencia de cada punto de datos en la estimación de los coeficientes del modelo.

4. **Tratamiento de Outliers**: Dependiendo del contexto, hay varias estrategias para tratar con outliers:

- **Eliminación**: Si se determina que un outlier es un error de entrada de datos o no representa la población de interés, se puede optar por eliminarlo del conjunto de datos.
- **Transformación**: En algunos casos, aplicar transformaciones como la logarítmica puede mitigar el impacto de los outliers, permitiendo un ajuste más robusto del modelo.
- **Modelos Robustos**: Considerar el uso de modelos de regresión robusta que son menos sensibles a la presencia de outliers. Estas técnicas pueden proporcionar estimaciones más estables en presencia de datos atípicos.

Al abordar los valores atípicos adecuadamente, se puede mejorar la calidad del modelo y, por ende, la precisión de las predicciones. Esto es vital en aplicaciones donde las decisiones basadas en modelos pueden tener un impacto significativo.

Evaluación del Modelo

La evaluación del modelo de regresión logística implica el uso de métricas que ayuden a entender su rendimiento en la predicción. Algunas métricas comunes incluyen:

- **Exactitud**: Proporción de predicciones correctas sobre el total de predicciones. Aunque es una métrica intuitiva, puede ser engañosa en conjuntos de datos desbalanceados, donde una clase puede dominar.
- **Precisión**: Proporción de verdaderos positivos sobre el total de positivos predichos, es decir, cuántas de las instancias clasificadas como positivas son realmente positivas.
- **Sensibilidad (o Recall)**: Proporción de verdaderos positivos sobre el total de casos positivos reales. Es vital en situaciones donde es crucial identificar todos los casos positivos, como en el diagnóstico de enfermedades.
- **Especificidad**: Proporción de verdaderos negativos sobre el total de casos negativos reales. Esto ayuda a entender cuántas de las instancias negativas fueron correctamente identificadas.
- **AUC-ROC**: Área bajo la curva de la característica operativa del receptor, que proporciona una medida de la capacidad del modelo para distinguir entre las clases. Un valor de AUC cercano a 1 indica un excelente rendimiento del modelo.

Estas métricas, entre otras, permiten no solo evaluar la eficacia del modelo, sino también realizar comparaciones entre diferentes modelos o configuraciones, optimizando así el proceso de modelización.

Ajuste del Umbral de Decisión

El umbral de 0.5 para clasificar los eventos no es obligatorio y puede ajustarse según las necesidades del problema. Este ajuste se basa en la importancia de los falsos positivos y falsos negativos en el contexto específico:

- **Minimizar Falsos Positivos**: Si es crucial evitar clasificar erróneamente un evento positivo, se puede elevar el umbral por encima de 0.5. Esto es común en diagnósticos médicos donde un falso positivo puede causar ansiedad o tratamientos innecesarios.

- **Minimizar Falsos Negativos**: Si es más importante identificar todos los eventos positivos, el umbral puede reducirse por debajo de 0.5. Esto es fundamental en situaciones donde se requiere la detección de enfermedades graves o la identificación de fraudes financieros.

La elección del umbral debe ser informada por el contexto y las consecuencias asociadas a cada tipo de error. Por ejemplo, en medicina, la pérdida de un caso positivo (falso negativo) puede ser más crítica que el costo de un tratamiento innecesario (falso positivo). Así, es recomendable utilizar herramientas como la curva ROC para decidir el umbral óptimo, sopesando los beneficios y costos de los distintos tipos de errores.

La regresión logística es una herramienta potente y flexible para modelar relaciones entre una variable dependiente binaria y variables independientes. Su capacidad para estimar probabilidades, junto con la interpretación de los coeficientes mediante la transformación logit, la convierte en un enfoque esencial en la toma de decisiones basadas en datos. Al comprender y aplicar correctamente la regresión logística, junto con el ajuste adecuado del umbral de decisión, se pueden tomar decisiones informadas y respaldadas por análisis estadísticos sólidos. La habilidad para interpretar y comunicar los resultados de un modelo de regresión logística es crucial en un entorno donde los datos son cada vez más valorados como un activo estratégico. Además, la integración de esta técnica en procesos de análisis predictivo puede aumentar significativamente la eficacia en la toma de decisiones a través de la identificación de patrones y relaciones clave en los datos.

Ejemplo de uso de regresion logistica (Diagnotico de cancer)

```
# Importa la biblioteca pandas para manipulación de datos
import pandas as pd
# Importa el módulo datasets de sklearn para acceder a conjuntos de datos
from sklearn import datasets
```

Para este ejemplo vamos a utilizar el Wisconsin Breast Cancer Dataset. Es un dataset de imagenes de celulas obtenidas de analisis de personas que sufren un posible cancer de mama.

Las imagenes tienen el siguiente aspecto:

image.png

El siguiente código carga el conjunto de datos de cáncer de mama utilizando la función `datasets.load_breast_cancer()`. Posteriormente, se accede a las claves del diccionario devuelto para explorar la estructura y contenido del conjunto de datos:

- **'data'**: Contiene los datos del conjunto de datos.
- **'target'**: Contiene las etiquetas de clase correspondientes a cada muestra.
- **'frame'**: Puede contener un DataFrame de Pandas si el conjunto de datos está estructurado así.
- **'target_names'**: Nombres asociados a las etiquetas de clase.

- **'DESCR'**: Una descripción detallada del conjunto de datos.
- **'feature_names'**: Nombres de las características (atributos) del conjunto de datos.
- **'filename'**: Nombre del archivo que contiene el conjunto de datos.
- **'data_module'**: Módulo o paquete específico del cual se carga el conjunto de datos.

Este código es útil para cargar conjuntos de datos específicos y explorar sus diferentes componentes y metadatos.

```
# Carga el conjunto de datos de cáncer de mama
cancer_datos = datasets.load_breast_cancer()

# Muestra las claves del diccionario del conjunto de datos
cancer_datos.keys()

dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR',
'feature_names', 'filename', 'data_module'])

# Imprime la descripción del conjunto de datos
# print(cancer_datos["DESCR"])
```

Conjunto de datos de cáncer de mama Wisconsin (diagnóstico)

Características del conjunto de datos:

- **Número de instancias:** 569
- **Número de atributos:** 30 atributos numéricos predictivos y la clase

Información de los atributos:

- Atributos como radio (media de las distancias desde el centro hasta puntos en el perímetro), textura (desviación estándar de los valores de escala de grises), perímetro, área, suavidad (variación local en las longitudes del radio), compacidad (perímetro^2 / área - 1.0), concavidad (severidad de porciones cóncavas del contorno), puntos cóncavos (número de porciones cóncavas del contorno), simetría y dimensión fractal ("aproximación de la línea costera" - 1).

  Los valores de media, error estándar y "peor" (o mayor entre los tres valores peores) de estas características se calcularon para cada imagen, resultando en 30 características. Por ejemplo, el campo 0 es Radio Medio, el campo 10 es Radio SE y el campo 20 es Radio Peor.

- **Clase:**
  - WDBC-Maligno
  - WDBC-Benigno

Estadísticas resumidas:

	Mínimo	Máximo
Radio (media)	6.981	28.11
Textura (media)	9.71	39.28
Perímetro (media)	43.79	188.5
Área (media)	143.5	2501.0
Suavidad (media)	0.053	0.163
Compacidad (media)	0.019	0.345
Concavidad (media)	0.0	0.427
Puntos cóncavos (media)	0.0	0.201
Simetría (media)	0.106	0.304
Dimensión fractal (media)	0.05	0.097
Radio (error estándar)	0.112	2.873
Textura (error estándar)	0.36	4.885
Perímetro (error estándar)	0.757	21.98
Área (error estándar)	6.802	542.2
Suavidad (error estándar)	0.002	0.031
Compacidad (error estándar)	0.002	0.135
Concavidad (error estándar)	0.0	0.396
Puntos cóncavos (error estándar)	0.0	0.053
Simetría (error estándar)	0.008	0.079
Dimensión fractal (error estándar)	0.001	0.03
Radio (peor)	7.93	36.04
Textura (peor)	12.02	49.54
Perímetro (peor)	50.41	251.2
Área (peor)	185.2	4254.0
Suavidad (peor)	0.071	0.223
Compacidad (peor)	0.027	1.058
Concavidad (peor)	0.0	1.252
Puntos cóncavos (peor)	0.0	0.291
Simetría (peor)	0.156	0.664
Dimensión fractal (peor)	0.055	0.208

Valores faltantes: Ninguno

Distribución de clases: 212 - Maligno, 357 - Benigno

Creadores: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

Fecha: Noviembre de 1995

Este conjunto de datos es una copia del conjunto de datos UCI ML Breast Cancer Wisconsin (Diagnostic).

Enlace al conjunto de datos

Los atributos describen características de los núcleos celulares presentes en imágenes de aspirados de aguja fina (FNA) de masas mamarias.

El plano separador descrito anteriormente se obtuvo utilizando el método Multisurface Method-Tree (MSM-T), un método de clasificación que utiliza programación lineal para construir un árbol de decisión.

Preparación de datos para facilitar su análisis

A continuación se accede al atributo `target_names` de `cancer_datos` para obtener los nombres asociados a las etiquetas de clase del conjunto de datos.

```
# Accede a los nombres de las clases
cancer_datos["target_names"]
```

```
array(['malignant', 'benign'], dtype='<U9')
```

Se crea un DataFrame `cancer_df` utilizando los datos (`data`) del conjunto de datos de cáncer de mama (`cancer_datos`). Las columnas del DataFrame se etiquetan con los nombres de las características (`feature_names`) del conjunto de datos.

```
# Crea un DataFrame utilizando los datos del conjunto de datos de
cáncer de mama
cancer_df = pd.DataFrame(cancer_datos["data"],
                         columns=cancer_datos["feature_names"]
                        )
```

Se añade una nueva columna llamada `objetivo` al DataFrame `cancer_df`, que contiene las etiquetas de clase (`target`) del conjunto de datos.

```
# Añade la columna objetivo al DataFrame
cancer_df["objetivo"] = cancer_datos["target"]
```

El dataset contiene los valores medios de ciertos parámetros del núcleo de las células mostradas en las imágenes, así como dichos valores para la célula con características más preocupantes.

```
# Muestra la forma del DataFrame
cancer_df.shape
```

```
(569, 31)
```

```
# Muestra las primeras filas del DataFrame
cancer_df.head()
```

```
   mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0        17.99         10.38          122.80     1001.0
0.11840
1        20.57         17.77          132.90     1326.0
0.08474
2        19.69         21.25          130.00     1203.0
0.10960
3        11.42         20.38           77.58      386.1
0.14250
4        20.29         14.34          135.10     1297.0
0.10030

   mean compactness  mean concavity  mean concave points  mean
symmetry  \
0           0.27760          0.3001              0.14710
0.2419
1           0.07864          0.0869              0.07017
0.1812
2           0.15990          0.1974              0.12790
0.2069
3           0.28390          0.2414              0.10520
0.2597
4           0.13280          0.1980              0.10430
0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst
area  \
0                 0.07871  ...          17.33           184.60
2019.0
1                 0.05667  ...          23.41           158.80
1956.0
2                 0.05999  ...          25.53           152.50
1709.0
3                 0.09744  ...          26.50            98.87
567.7
4                 0.05883  ...          16.67           152.20
1575.0

   worst smoothness  worst compactness  worst concavity  worst concave
points  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
```

```
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   worst symmetry  worst fractal dimension  objetivo
0          0.4601                  0.11890         0
1          0.2750                  0.08902         0
2          0.3613                  0.08758         0
3          0.6638                  0.17300         0
4          0.2364                  0.07678         0

[5 rows x 31 columns]
```

Se calcula la distribución de las etiquetas de clase en la columna `objetivo` del DataFrame `cancer_df` utilizando `value_counts(True)`, lo que proporciona la frecuencia relativa de cada etiqueta.

```
# Calcula y muestra la distribución de las etiquetas en la columna objetivo
cancer_df["objetivo"].value_counts(True)
```

```
objetivo
1    0.627417
0    0.372583
Name: proportion, dtype: float64
```

Después, se reemplazan los valores en la columna `objetivo` del DataFrame `cancer_df`. Se sustituyen los valores 1 por 0 y los valores 0 por 1 utilizando el método `replace()`.

```
# Reemplaza los valores en la columna objetivo
cancer_df["objetivo"] = cancer_df["objetivo"].replace({1:0,0:1})
```

A continuación, se importan las bibliotecas necesarias para la división de datos y la creación del modelo de regresión.

```
# Importa las bibliotecas necesarias
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

Regresión Lineal

Se divide el DataFrame `cancer_df` en dos conjuntos de datos: `train_df` y `test_df`, utilizando la función `train_test_split` con un tamaño de prueba del 20% (`test_size=0.2`).

```
# Divide el DataFrame en conjuntos de entrenamiento y prueba
train_df, test_df = train_test_split(cancer_df, test_size=0.2,
random_state=42)
```

Las variables de entrenamiento se definen como las características del conjunto de datos, obtenidas de `cancer_datos["feature_names"]`.

```
# Define las variables de entrenamiento
variables_entrenamiento = cancer_datos["feature_names"]
```

La variable objetivo se establece como "objetivo", que representa las etiquetas de clase modificadas en el DataFrame `cancer_df`.

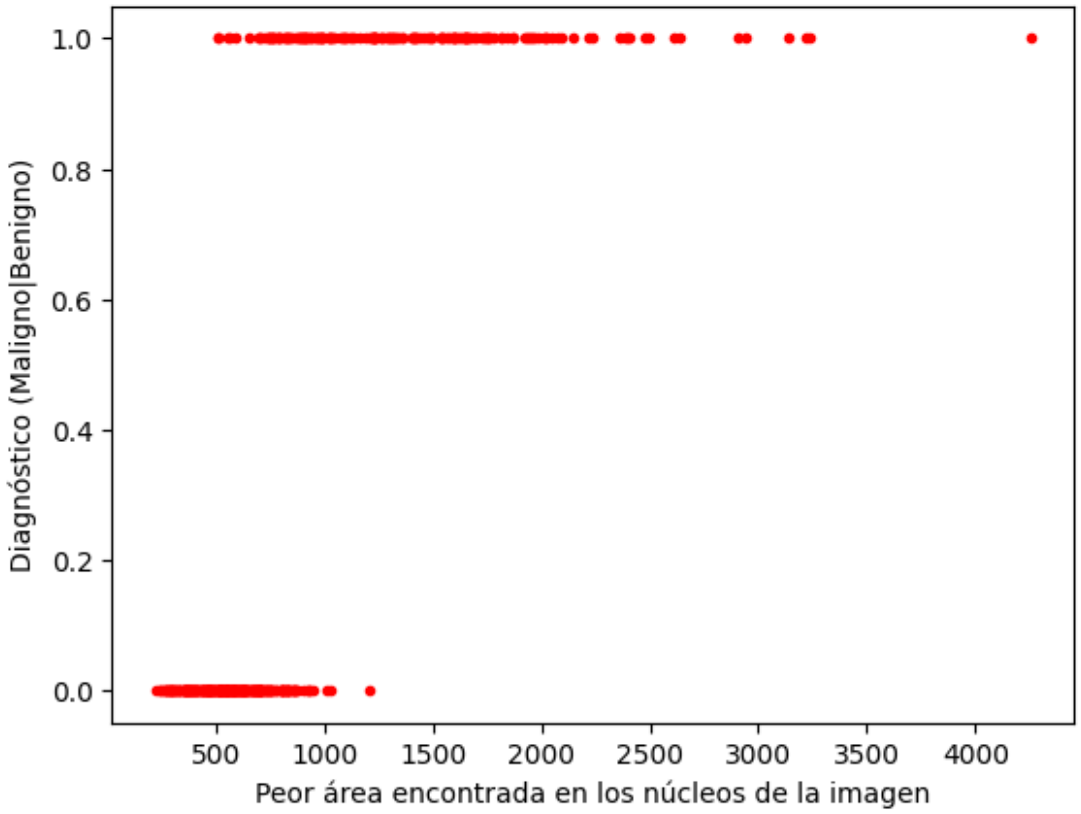
```
# Define la variable objetivo
variable_objetivo = "objetivo"
```

Se establece una columna para graficar, que en este caso es "worst area".

```
# Establece la columna para la gráfica
columna_entrenamiento = "worst area"
```

Se grafica la relación entre el área peor encontrada en los núcleos de la imagen y el diagnóstico.

```
# Grafica los datos de entrenamiento
plt.plot(train_df[columna_entrenamiento], train_df.objetivo, '.r')
plt.xlabel("Peor área encontrada en los núcleos de la imagen")
plt.ylabel("Diagnóstico (Maligno|Benigno)");
```



Se crea un modelo de regresión lineal (`LinearRegression()`) y se asigna a la variable `modelo_ols`.

```
# Crea el modelo de regresión lineal
modelo_ols = LinearRegression()
```

El modelo se entrena utilizando los datos de entrenamiento (`train_df`) especificando la columna de características (`columna_entrenamiento`) y la variable objetivo (`variable_objetivo`) mediante el método `fit()`.

```
# Entrena el modelo
modelo_ols.fit(train_df[[columna_entrenamiento]],
train_df[variable_objetivo])

LinearRegression()
```

Se realizan predicciones sobre los datos de prueba (`test_df`) utilizando el modelo entrenado mediante `predict()` y se guardan en la variable `predicciones`.

```
# Realiza predicciones sobre el conjunto de prueba
predicciones = modelo_ols.predict(test_df[[columna_entrenamiento]])
```

Se muestran las primeras 10 predicciones (`predicciones[:10]`) para verificar el rendimiento inicial del modelo.

```
# Muestra las primeras 10 predicciones
predicciones[:10]
```

```
array([0.24879738, 0.98062866, 0.54329156, 0.14894901, 0.11322288,
       0.95291011, 1.20299302, 0.58702527, 0.20999141, 0.27405206])
```

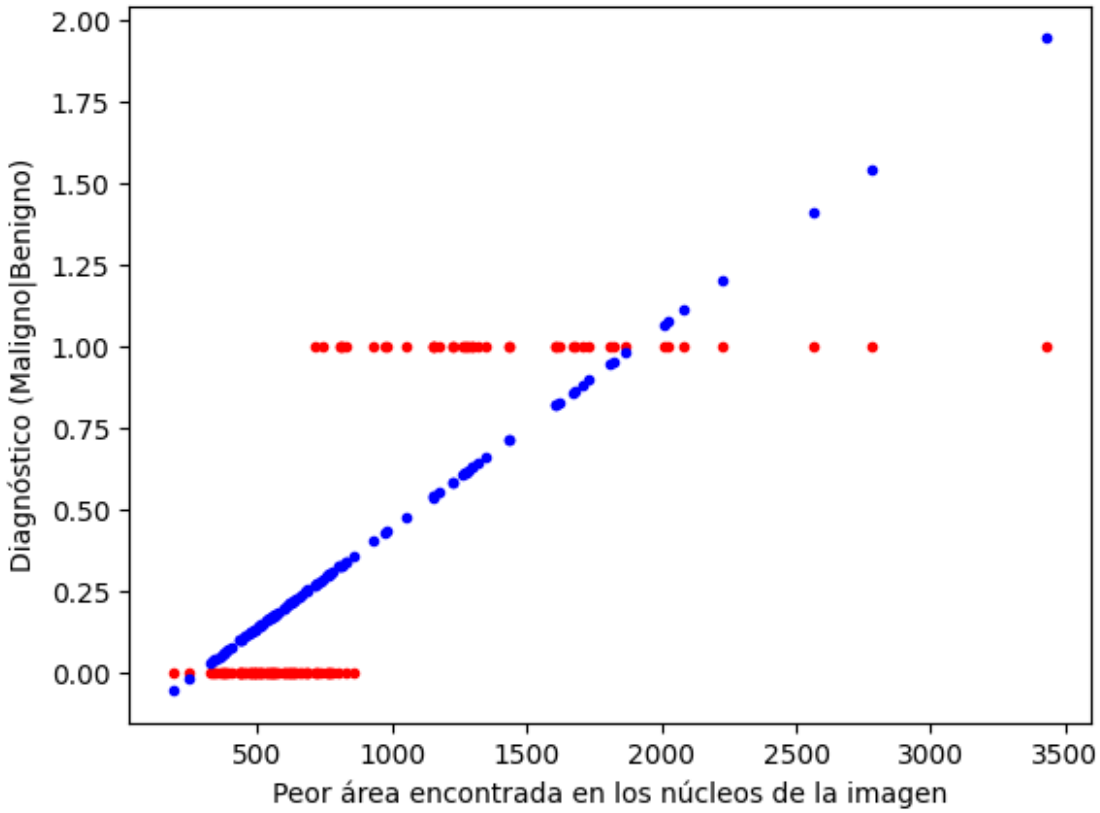
Las predicciones mostradas son un array de valores numéricos que representan las estimaciones del modelo de regresión lineal sobre el conjunto de datos de prueba (`test_df`). Cada valor en el array corresponde a la probabilidad estimada de pertenecer a la clase positiva o negativa (dependiendo de cómo se haya definido la codificación de las etiquetas).

Por ejemplo, para los primeros 10 ejemplos en el conjunto de prueba, las predicciones indican la probabilidad de que cada uno de ellos pertenezca a la clase objetivo, después de haber sido transformados por el modelo de regresión lineal entrenado en las características proporcionadas en `train_df`.

Se grafica la comparación entre las etiquetas de clase verdaderas y las predicciones.

```
# Grafica los datos de prueba y las predicciones
plt.plot(test_df[columna_entrenamiento], test_df.objetivo, '.r')
plt.plot(test_df[columna_entrenamiento], predicciones, '.b')
plt.xlabel("Peor área encontrada en los núcleos de la imagen")
plt.ylabel("Diagnóstico (Maligno|Benigno)")
```

```
Text(0, 0.5, 'Diagnóstico (Maligno|Benigno)')
```



Función Logística

La función logística es una función matemática utilizada comúnmente en modelos de regresión logística y en la teoría de sistemas de control. Su forma general es:

$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}}$$

Donde:

- L es el valor máximo que la función puede alcanzar (en términos de probabilidad, generalmente se fija en 1).
- k es la tasa de crecimiento, que determina la pendiente de la curva logística. Un valor mayor de k indica un crecimiento más rápido.
- x_0 es el valor de la variable independiente x en el cual la función alcanza la mitad de su valor máximo. Este parámetro desplaza la curva hacia la izquierda o hacia la derecha.

La función logística transforma cualquier valor real x en un rango entre 0 y L. Es especialmente útil en situaciones donde se requiere modelar probabilidades que deben estar entre 0 y 1, como en clasificación binaria. Por lo tanto, en el contexto de la regresión logística, la salida de la función se interpreta como la probabilidad de que una observación pertenezca a una de las dos clases.

La implementación de la función logística en Python se realiza de la siguiente manera:

```
# Define la función logística
def funcion_logistica(x,L=1,k=0.1,x0=0):
    return L / (1+np.exp(-k*(x-x0)))
```

Se generan `predicciones_probabilidades` aplicando la función logística a cada predicción en `predicciones` mediante `map(funcion_logistica, predicciones)` y se almacenan en una lista.

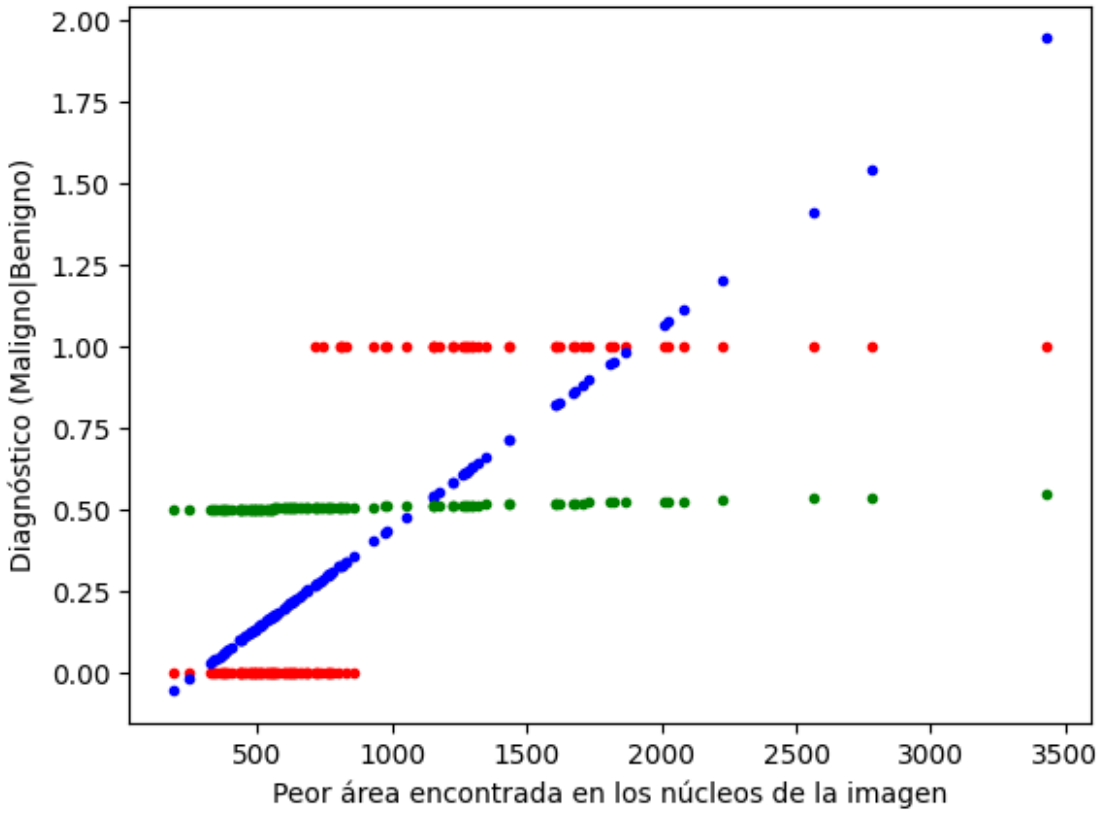
```
# Genera las predicciones probabilísticas
predicciones_probabilidades =
list(map(funcion_logistica,predicciones))
```

Se realiza una visualización gráfica utilizando Matplotlib:

- Se grafican los puntos de prueba en rojo (`test_df[columna_entrenamiento], test_df.objetivo, '.r'`).
- Se grafican las predicciones del modelo en azul (`test_df[columna_entrenamiento], predicciones, '.b'`).
- Se grafican las predicciones transformadas por la función logística en verde (`test_df[columna_entrenamiento], predicciones_probabilidades, '.g'`).

Se etiquetan los ejes del gráfico: el eje x representa el "Área Peor encontrada en los núcleos de la imagen", y el eje y representa "Diagnóstico (Maligno|Benigno".

```
# Grafica las predicciones y las probabilidades
plt.plot(test_df[columna_entrenamiento], test_df.objetivo, '.r')
plt.plot(test_df[columna_entrenamiento], predicciones, '.b')
plt.plot(test_df[columna_entrenamiento],
predicciones_probabilidades,'.g')
plt.xlabel("Peor área encontrada en los núcleos de la imagen")
plt.ylabel("Diagnóstico (Maligno|Benigno)")

Text(0, 0.5, 'Diagnóstico (Maligno|Benigno)')
```



Función Logística con Parámetro Parcial

Se define una función parcial `funcion_logit_k5` utilizando `partial` de la biblioteca `functools`, que aplica la función logística con un parámetro específico $k=5$ sobre los datos de entrada. Esto permite crear una versión de la función logística que tiene un crecimiento más rápido en comparación con la función logística estándar.

El uso de `partial` es útil cuando deseamos fijar uno o más argumentos de una función y crear una nueva función con los parámetros restantes. En este caso, se fija el parámetro $k=5$, lo que significa que la pendiente de la función logística se incrementará, permitiendo una transición más rápida entre los valores de salida.

```
# Importa la función partial
from functools import partial

# Define la función parcial para la función logística
funcion_logit_k5 = partial(funcion_logistica, k=5)
```

Se generan `predicciones_probabilidades` aplicando la función parcial `funcion_logit_k5` a cada predicción en `predicciones` mediante `map(funcion_logit_k5, predicciones)` y se almacenan en una lista.

```
# Genera las predicciones probabilísticas utilizando la función
logística con k=5
```

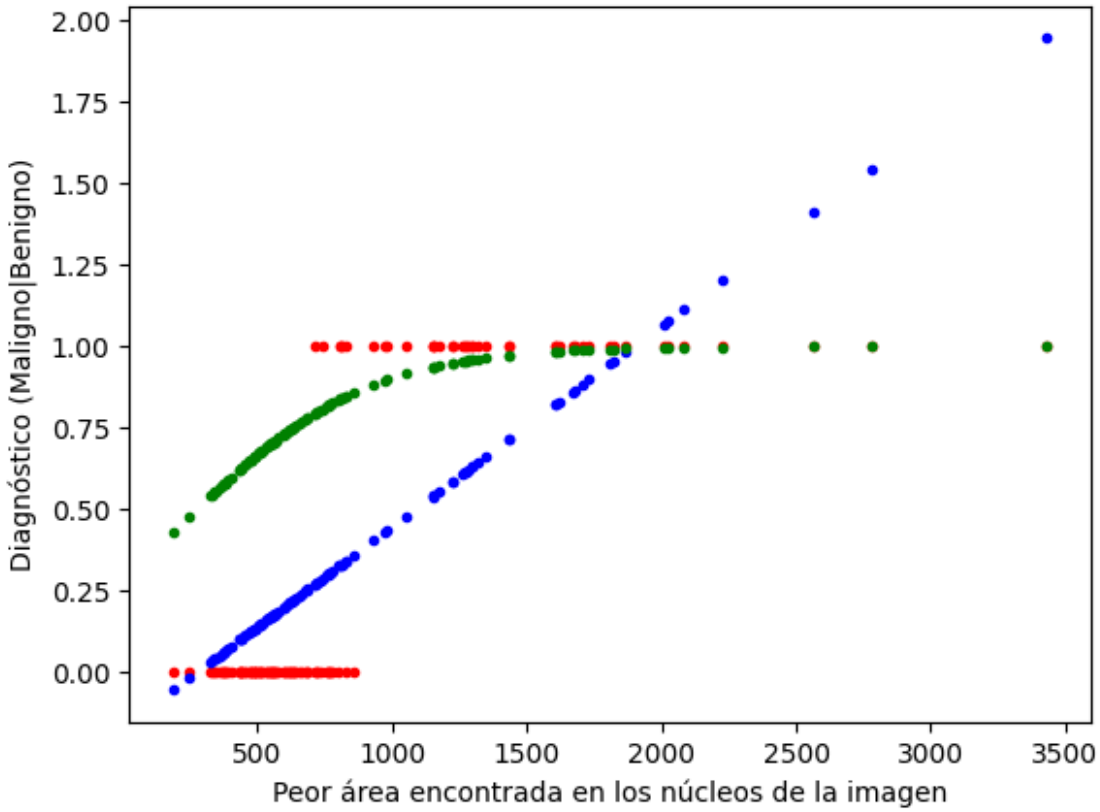
```
predicciones_probabilidades = list(map(funcion_logit_k5,
predicciones))
```

Se realiza una visualización gráfica utilizando Matplotlib:

- Se grafican los puntos de prueba en rojo (`test_df[columna_entrenamiento], test_df.objetivo, '.r'`).
- Se grafican las predicciones del modelo en azul (`test_df[columna_entrenamiento], predicciones, '.b'`).
- Se grafican las predicciones transformadas por la función logística con `k=5` en verde (`test_df[columna_entrenamiento], predicciones_probabilidades, '.g'`).

Se etiquetan los ejes del gráfico: el eje x representa el "Área Peor encontrada en los núcleos de la imagen", y el eje y representa "Diagnóstico (Maligno|Benigno)".

```
# Grafica las predicciones y las probabilidades
plt.plot(test_df[columna_entrenamiento], test_df.objetivo, '.r')
plt.plot(test_df[columna_entrenamiento], predicciones, '.b')
plt.plot(test_df[columna_entrenamiento],
predicciones_probabilidades,'.g')
plt.xlabel("Peor área encontrada en los núcleos de la imagen")
plt.ylabel("Diagnóstico (Maligno|Benigno)");
```



Regresión Logística

Se importa el clasificador de regresión logística.

```
# Importa el clasificador de regresión logística
from sklearn.linear_model import LogisticRegression
```

Se crea la matriz de características `X` utilizando las variables de entrenamiento (`variables_entrenamiento`) del DataFrame `cancer_df`.

```
# Crea la matriz de características
X = cancer_df[variables_entrenamiento]
```

Se define el vector objetivo `y` utilizando la variable objetivo (`variable_objetivo`) del DataFrame `cancer_df`.

```
# Define el vector objetivo
y = cancer_df[variable_objetivo]
```

Luego, se dividen los datos en conjuntos de entrenamiento (`X_train`, `y_train`) y prueba (`X_test`, `y_test`) utilizando la función `train_test_split`. Se especifica que el tamaño del conjunto de prueba será del 20% del conjunto de datos original (`test_size=0.2`).

```
# Divide los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,
random_state=42)
```

Se crea un clasificador de regresión logística (`LogisticRegression`) utilizando el solver "liblinear" y se asigna a la variable `clf`.

```
# Crea el clasificador de regresión logística
clf = LogisticRegression(solver="liblinear")
```

El clasificador se entrena utilizando los datos de entrenamiento (`X_train`, `y_train`) mediante el método `fit()`.

```
# Entrena el clasificador
clf.fit(X_train, y_train)

LogisticRegression(solver='liblinear')
```

Se realizan predicciones sobre los datos de prueba (`X_test`) utilizando el clasificador entrenado mediante `predict()` y se guardan en la variable `predicciones`.

```
# Realiza predicciones sobre el conjunto de prueba
predicciones = clf.predict(X_test)
```

Se muestran las primeras 10 predicciones (`predicciones[:10]`) para evaluar el rendimiento inicial del clasificador.

```
# Muestra las primeras 10 predicciones
predicciones[:10]

array([0, 1, 1, 0, 0, 1, 1, 1, 0, 0])
```

El array anterior contiene las predicciones del modelo de regresión logística para un conjunto de datos de prueba.

- Los valores `0` indican que el modelo predice que es probable que los casos correspondientes pertenezcan a la clase negativa (benigno, en este contexto).
- Los valores `1` indican que el modelo predice que es probable que los casos correspondientes pertenezcan a la clase positiva (maligno, en este contexto).

Estas predicciones se compararían con las etiquetas reales de los datos de prueba para evaluar la precisión y el rendimiento del modelo en la clasificación de casos benignos y malignos.

Se generan las probabilidades de predicción (`predicciones_probabilidades`) utilizando el clasificador de regresión logística (`clf`) sobre los datos de prueba (`X_test`) mediante el método `predict_proba()`.

```
# Genera las probabilidades de predicción
predicciones_probabilidades = clf.predict_proba(X_test)
```

Se muestran las primeras 10 filas de las probabilidades de predicción (`predicciones_probabilidades[:10]`), que representan la probabilidad estimada de pertenecer a cada clase para cada muestra en el conjunto de prueba.

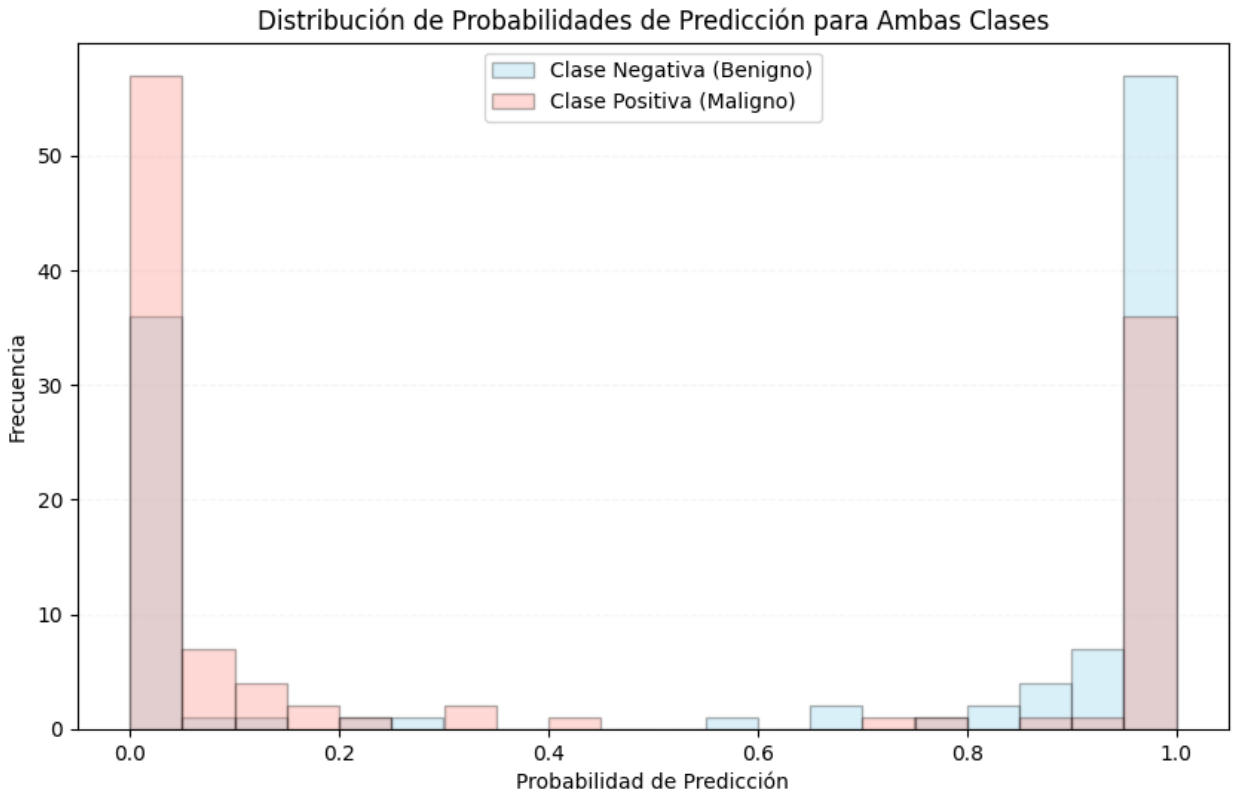
```
# Muestra las primeras 10 probabilidades de predicción
predicciones_probabilidades[:10]

array([[8.35085685e-01, 1.64914315e-01],
       [1.58795266e-09, 9.99999998e-01],
       [1.54456572e-03, 9.98455434e-01],
       [9.97196083e-01, 2.80391664e-03],
       [9.99455088e-01, 5.44911540e-04],
       [2.50242715e-10, 1.00000000e+00],
       [4.59632332e-13, 1.00000000e+00],
       [9.66246758e-03, 9.90337532e-01],
       [9.97351323e-01, 2.64867714e-03],
       [9.88282275e-01, 1.17177253e-02]])
```

A continuación, se crea un histograma de las probabilidades de predicción utilizando Matplotlib, que proporciona una representación visual de la distribución de las probabilidades asignadas a las clases.

El histograma muestra cuántas predicciones están agrupadas en diferentes rangos de probabilidad, lo cual es útil para ver si el modelo tiene una alta confianza en sus predicciones

(probabilidades cercanas a 0 o 1) o si muchas predicciones se encuentran en rangos intermedios, lo que indicaría menor confianza.

```
# Crea un histograma de las probabilidades de predicción para ambas clases (benigno y maligno)
plt.figure(figsize=(10, 6))  # Ajusta el tamaño del gráfico

# Histograma para la clase negativa (benigno)
plt.hist(predicciones_probabilidades[:, 0], bins=20, color="skyblue", edgecolor="black", alpha=0.3, label="Clase Negativa (Benigno)")

# Histograma para la clase positiva (maligno)
plt.hist(predicciones_probabilidades[:, 1], bins=20, color="salmon", edgecolor="black", alpha=0.3, label="Clase Positiva (Maligno)")

# Título y etiquetas de los ejes
plt.title("Distribución de Probabilidades de Predicción para Ambas Clases")
plt.xlabel("Probabilidad de Predicción")
plt.ylabel("Frecuencia")
plt.legend(loc="upper center")  # Muestra la leyenda en el gráfico
plt.grid(axis="y", linestyle="--", alpha=0.1)
plt.show()
```



Se crea un DataFrame `probs_df` utilizando las probabilidades de predicción (`predicciones_probabilidades`) generadas por el clasificador de regresión logística.

```
# Crea un DataFrame con las probabilidades de predicción
probs_df = pd.DataFrame(predicciones_probabilidades)
```

Se realiza una copia de los datos de prueba (`X_test`) y se reinicia el índice para crear el DataFrame `X`. Luego, se añaden las siguientes columnas:

- "objetivo": Contiene las etiquetas reales (`y_test`) convertidas a una lista.
- "prediccion": Contiene las predicciones realizadas por el clasificador.

```
# Reinicia el índice y crea una copia de los datos de prueba
X = X_test.reset_index().copy()
X["objetivo"] = y_test.tolist()
```

A continuación, se concatenan `X` con `probs_df` a lo largo del eje 1 (columnas) para combinar los datos originales con las probabilidades de predicción.

```
# Combina los datos de prueba con las probabilidades de predicción
X["prediccion"] = predicciones
X = pd.concat([X, probs_df], axis=1)
```

Se seleccionan las columnas "objetivo", "prediccion", 0 (probabilidad de pertenecer a la clase 0) y 1 (probabilidad de pertenecer a la clase 1) del DataFrame `X` y se muestran las primeras 20 filas (`head(20)`) para examinar las etiquetas reales, las predicciones del modelo y las probabilidades asignadas a cada clase.

```
# Muestra las primeras 20 filas de comparación
X[["objetivo", "prediccion", 0, 1]].head(20)
```

	objetivo	prediccion	0	1
0	0	0	8.350857e-01	0.164914
1	1	1	1.587953e-09	1.000000
2	1	1	1.544566e-03	0.998455
3	0	0	9.971961e-01	0.002804
4	0	0	9.994551e-01	0.000545
5	1	1	2.502427e-10	1.000000
6	1	1	4.596323e-13	1.000000
7	1	1	9.662468e-03	0.990338
8	0	0	9.973513e-01	0.002649
9	0	0	9.882823e-01	0.011718
10	0	0	9.384783e-01	0.061522
11	1	1	4.841174e-04	0.999516
12	0	0	9.901019e-01	0.009898
13	1	1	2.445759e-01	0.755424
14	0	0	9.977836e-01	0.002216
15	1	1	1.101263e-03	0.998899
16	0	0	9.975245e-01	0.002476

```
17          0           0  9.997321e-01  0.000268
18          0           0  9.982903e-01  0.001710
19          1           1  3.942000e-08  1.000000
```

Estos resultados muestran la comparación entre las etiquetas reales (`objetivo`) y las predicciones del modelo (`prediccion`), junto con las probabilidades asignadas a cada clase (0 y 1). Esta información es crucial para evaluar el rendimiento del modelo y entender cómo las probabilidades se distribuyen entre las clases, lo que puede ser útil para ajustar el umbral de decisión o para otros análisis posteriores.

Matriz de Confusión

La matriz de confusión es una herramienta de evaluación del rendimiento de un modelo de clasificación. Proporciona una representación tabular que muestra la cantidad de predicciones correctas e incorrectas, desglosadas en cuatro categorías clave:

- **Verdaderos Positivos (VP)**: Casos donde el modelo predijo correctamente la clase positiva.
- **Verdaderos Negativos (VN)**: Casos donde el modelo predijo correctamente la clase negativa.
- **Falsos Positivos (FP)**: Casos donde el modelo predijo la clase positiva, pero en realidad era negativa. También se le llama error tipo I.
- **Falsos Negativos (FN)**: Casos donde el modelo predijo la clase negativa, pero en realidad era positiva. También se le llama error tipo II.

La matriz de confusión permite a los analistas ver de manera clara en qué casos el clasificador ha acertado y en cuáles ha fallado. La diagonal de la matriz representa los aciertos (verdaderos positivos y verdaderos negativos), mientras que los valores fuera de la diagonal representan los errores de clasificación. Esta información es crucial para evaluar la efectividad del clasificador y ajustar los parámetros o elegir diferentes algoritmos si es necesario.

A continuación se crea y se visualiza la matriz de confusión utilizando la biblioteca `sklearn.metrics`.

```
# Importa la función para crear la matriz de confusión
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Genera la matriz de confusión
matriz_confusion = confusion_matrix(y_test, predicciones)

# Visualiza la matriz de confusión utilizando un mapa de calor
plt.figure(figsize=(8, 6))
sns.heatmap(matriz_confusion, annot=True, fmt='d', cmap='Blues',
xticklabels=cancer_datos["target_names"],
yticklabels=cancer_datos["target_names"])
plt.xlabel("Predicción")
plt.ylabel("Realidad")
```

```
plt.title("Matriz de Confusión")
plt.show()
```