Universidad Politécnica Metropolitana de Hidalgo

Programa Educativo de Maestría en Inteligencia Artificial

"Estadística Aplicada"

Reporte: Tarea 5

Flores García Katherine Itzel
(253220100)
Morales Hernández Emmanuel
(253220003)

26 de octubre de 2025

Índice

Índice de figuras

Índice de cuadros

1. OBJETIVO

Objetivo general

Construir y evaluar un modelo de **regresión logística** utilizando un dataset de Kaggle para identificar las variables que mejor explican una variable binaria dependiente y analizar la calidad del modelo.

Objetivos específicos

- Analizar y describir el dataset seleccionado, incluyendo todas las variables numéricas y la variable dependiente.
- Estimar la ecuación de regresión logística con todas las variables y analizar la interpretación de los coeficientes.
- Evaluar el modelo mediante métricas como pseudo R-cuadrada, Log-Likelihood, LL-Null y LLR p-value.
- Interpretar los resultados de los coeficientes, errores estándar, valores z y significancia estadística de las variables independientes.
- Construir la matriz de confusión, graficar la función logística junto con los datos y determinar la precisión del modelo.
- Seleccionar y construir un modelo optimizado con un máximo de cuatro variables independientes según criterios estadísticos.

2. INTRODUCCIÓN

En la actualidad, el análisis de datos se ha convertido en una herramienta esencial para la toma de decisiones en diversos ámbitos, desde la investigación científica hasta la industria y los negocios. La disponibilidad de plataformas como Kaggle permite a los equipos acceder a una amplia variedad de datasets que pueden ser utilizados para entrenar modelos predictivos y realizar análisis estadísticos avanzados.

En este trabajo, se propone utilizar un dataset seleccionado de Kaggle que contenga al menos diez variables numéricas y una variable binaria como dependiente. Se busca aplicar la regresión logística para comprender cómo las variables independientes influyen en la variable dependiente y evaluar la eficacia del modelo mediante métricas estadísticas y gráficas.

El código completo utilizado en este análisis puede visualizarse en el siguiente link a Colab, donde se incluyen todos los pasos desde la carga del dataset hasta la construcción y evaluación del modelo de regresión logística.

Este análisis permitirá identificar las variables más relevantes y medir el desempeño predictivo del modelo, proporcionando información útil para la interpretación de los resultados.

La tabla a continuación muestra ejemplos representativos de los registros incluidos en el dataset.

Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	...
40	M	ATA	140	289	0	Normal	...
49	F	NAP	160	180	0	Normal	...
37	M	ATA	130	283	0	ST	...
48	F	ASY	138	214	0	Normal	...
54	M	NAP	150	195	0	Normal	...
39	M	NAP	120	339	0	Normal	...
45	F	ATA	130	237	0	Normal	...
54	M	ATA	110	208	0	Normal	...
...	...	...	...	...	...	...	...

Cuadro 1: Primeras filas del Heart Failure Prediction Dataset.

3. Desarrollo

3.1. PARTE I

El dataset utilizado para esta actividad es el **Heart Failure Prediction Dataset**, obtenido el **18 de octubre de 2025** de la plataforma *Kaggle*, elaborado por **FEDESORIANO**[1]. El conjunto de datos está disponible en el siguiente enlace: Heart Failure Prediction Dataset.

Este dataset está enfocado en enfermedades cardiovasculares, una de las **principales causas de mortalidad a nivel mundial**. Contiene **11 variables** que permiten predecir la presencia de insuficiencia cardíaca, incluyendo factores de riesgo como hipertensión, diabetes, edad, sexo y nivel de colesterol, entre otros.

El análisis de estas variables facilita la identificación de pacientes con **alto riesgo cardiovascular**, lo que permite aplicar estrategias de prevención y tratamiento oportunas. Las variables incluidas en el dataset son las siguientes:

- **Age:** Edad del paciente (numérica, en años).
- **Sex:** Sexo del paciente (categórica, M = masculino, F = femenino).
- **ChestPainType:** Tipo de dolor torácico (categórica):
  - `ATA` = angina típica,
  - `NAP` = angina no típica,
  - `ASY` = asintomático,
  - `TA` = dolor atípico.
- **RestingBP:** Presión arterial en reposo (numérica, mmHg).
- **Cholesterol:** Nivel de colesterol sérico (numérica, mg/dL).
- **FastingBS:** Glucosa en ayunas (binaria, 1 = $>$120 mg/dL, 0 = $\leq$120 mg/dL).
- **RestingECG:** Electrocardiograma en reposo (Normal / ST / LVH).

- **MaxHR:** Frecuencia cardíaca máxima alcanzada (latidos por minuto).
- **ExerciseAngina:** Angina inducida por ejercicio (Y = sí, N = no).
- **Oldpeak:** Depresión del segmento ST inducida por ejercicio.
- **ST_Slope:** Pendiente del segmento ST durante ejercicio (Up / Flat / Down).
- **HeartDisease:** Variable dependiente binaria (0 = no, 1 = sí).

3.1.1. Análisis del Dataset

Este análisis permite conocer rápidamente:

- El número de registros no nulos por columna y el tipo de dato (‘df.info()‘).
- Estadísticas descriptivas básicas de las variables numéricas (‘df.describe()‘).
- Las categorías únicas presentes en cada variable categórica, lo que facilita su comprensión y preparación para modelos predictivos.

```
# Resumen de la informacion general del dataset
display(df.info())

# Estadisticas descriptivas de las variables numericas
display(df.describe())

# Identificacion de columnas de tipo objeto (categoricas)
col_obj = df.select_dtypes(include='object').columns

# Mostrar valores unicos de cada columna categorica
for column in col_obj:
    print(f"==='{column}'===")
    display(df[column].unique())
```

Columna	Non-Null Count	Dtype
Age	918	int64
Sex	918	object
ChestPainType	918	object
RestingBP	918	int64
Cholesterol	918	int64
FastingBS	918	int64
RestingECG	918	object
MaxHR	918	int64
ExerciseAngina	918	object
Oldpeak	918	float64
ST_Slope	918	object
HeartDisease	918	int64

Cuadro 2: Resumen de las columnas, conteo de valores no nulos y tipos de datos del Heart Failure Prediction Dataset.

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

Figura 1: Análisis de la distribución del Dataset

3.1.2. Separación de variables

En esta etapa se realiza la separación entre las variables independientes (`X`) y la variable dependiente (`y`), preparándolas para el análisis de regresión logística:

```
# Separacion de variables independientes y dependiente
X = df.drop('HeartDisease', axis=1)  # Variables independientes
y = df['HeartDisease']               # Variable dependiente

# Identificacion de columnas numericas
col_num = X.select_dtypes(include='number').columns
```

Este paso permite:

- Definir claramente cuál es la variable a predecir (`HeartDisease`).
- Seleccionar las variables que se utilizarán como predictores.
- Identificar las columnas numéricas para aplicar análisis estadísticos o transformaciones específicas.

3.1.3. Limpieza de valores nulos

En esta etapa se realiza la separación entre las variables independientes (`X`) y la variable dependiente (`y`), preparándolas para el análisis de regresión logística:

```
# Separacion de variables independientes y dependiente
X = df.drop('HeartDisease', axis=1)  # Variables independientes
y = df['HeartDisease']               # Variable dependiente

# Identificacion de columnas numericas
col_num = X.select_dtypes(include='number').columns
```

Este paso permite:

- Definir claramente cuál es la variable a predecir (`HeartDisease`).
- Seleccionar las variables que se utilizarán como predictores.
- Identificar las columnas numéricas para aplicar análisis estadísticos o transformaciones específicas.

	0
RestingBP	1
Cholesterol	172

Cuadro 3: Resultados de la limpieza de datos

3.1.4. Modelo de imputación

Para manejar valores faltantes en el dataset, se utiliza un modelo de **imputación iterativa** (`IterativeImputer`) con un `RandomForestRegressor` como estimador. Este método permite predecir los valores faltantes basándose en las demás variables del dataset.

```
    # Configurar IterativeImputer con RandomForestRegressor
imputer = IterativeImputer(
    estimator=RandomForestRegressor(
        n_estimators=100,           # Numero de arboles
        max_depth=10,               # Profundidad para evitar overfitting
        random_state=42,
        n_jobs=-1                   # Usar todos los cores
    ),
    max_iter=10,                    # Iteraciones maximas
    random_state=42,
    initial_strategy='median',      # Inicializacion con mediana
    imputation_order='ascending',   # Menos faltantes primero
    verbose=1
)
```

```
    # Realizar la imputacion
X[columns_with_zeros] = imputer.fit_transform(X[columns_with_zeros])
# Redondear RestingBP y Cholesterol a enteros (variables discretas)
X[columns_with_zeros] = X[columns_with_zeros].round().astype(int)
```

Durante la ejecución del `IterativeImputer` con `RandomForestRegressor`, se obtuvieron los siguientes mensajes que indican la convergencia del algoritmo y los cambios en cada iteración:

```
[IterativeImputer] Completing matrix with shape (918, 2)
[IterativeImputer] Change: 79.09199999999998, scaled tolerance: 0.603
[IterativeImputer] Change: 98.03282084774568, scaled tolerance: 0.603
[IterativeImputer] Change: 32.56462456117043, scaled tolerance: 0.603
[IterativeImputer] Change: 8.521020423501142, scaled tolerance: 0.603
[IterativeImputer] Change: 0.6216106996030817, scaled tolerance: 0.603
[IterativeImputer] Change: 8.526512829121202e-14, scaled tolerance: 0.603
[IterativeImputer] Early stopping criterion reached.
```

Interpretación:

- Cada línea muestra el cambio promedio en los valores imputados durante cada iteración.
- El algoritmo detiene las iteraciones cuando el cambio se vuelve muy pequeño comparado con la tolerancia escalada ('scaled tolerance'), indicando que la imputación ha convergido.
- Esto asegura que los valores faltantes han sido estimados de manera estable y confiable.

3.1.5. Revisión del Dataset imputado

```
1
2 # Mostrar informaciOn
3 display(X.isnull().sum())
4 display(X.describe())
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.538126	245.295207	0.233115	136.809368	0.887364
std	9.432617	17.990127	54.526364	0.423046	25.460334	1.066570
min	28.000000	80.000000	85.000000	0.000000	60.000000	-2.600000
25%	47.000000	120.000000	213.000000	0.000000	120.000000	0.000000
50%	54.000000	130.000000	241.000000	0.000000	138.000000	0.600000
75%	60.000000	140.000000	271.750000	0.000000	156.000000	1.500000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000

Figura 2: Resultados del modelo de imputación

3.1.6. Codificación

Para preparar las variables categóricas para el modelo de regresión logística, se aplicaron técnicas de codificación según el tipo de variable:

- Variables **ordinales** (con más de dos categorías y un orden implícito): `ChestPainType`, `RestingECG`, `ST_Slope`, se codificaron usando `OrdinalEncoder`.
- Variables **nominales** (con dos categorías sin orden): `Sex` y `ExerciseAngina`, se codificaron con **One Hot Encoding** o mapeo a 0 y 1.

```
    """TIPO DE VARIABLES CATEGORICAS
'Sex' ---> Nominal (2) ---> One Hot Encoding
'ChestPainType' ---> Ordinal (4)
'RestingECG' ---> Ordinal (3)
'ExerciseAngina' ---> Nominal (2)---> One Hot Encoding
'ST_Slope' ---> Ordinal (3)
"""
# Codificacion Ordinal Encoder
encoder = OrdinalEncoder()
X['ChestPainType'] = encoder.fit_transform(X[['ChestPainType']])
X['RestingECG'] = encoder.fit_transform(X[['RestingECG']])
X['ST_Slope'] = encoder.fit_transform(X[['ST_Slope']])

# Codificacion One Hot Encoding
X['Sex'] = X['Sex'].map({'M': 1, 'F': 0})
X['ExerciseAngina'] = X['ExerciseAngina'].map({'Y': 1, 'N': 0})
X.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	40	1	1.0	140	289	0	1.0	172	0	0.0	2.0
1	49	0	2.0	160	180	0	1.0	156	0	1.0	1.0
2	37	1	1.0	130	283	0	2.0	98	0	0.0	2.0
3	48	0	0.0	138	214	0	1.0	108	1	1.5	1.0
4	54	1	2.0	150	195	0	1.0	122	0	0.0	2.0

Figura 3: Resultados de codificación

3.1.7. Estandarización

Para asegurar que todas las variables numéricas tengan la misma escala y evitar que algunas dominen el modelo de regresión logística, se aplica una **estandarización** utilizando `StandardScaler` de `scikit-learn`. Esto transforma los valores para que tengan media 0 y desviación estándar 1.

```
    # Estandarizacion de valores StandardScaler
scaler = StandardScaler()
X[col_num] = scaler.fit_transform(X[col_num])
X.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope
0	-1.433140	1	1.0	0.415002	0.801972	-0.551341	1.0	1.382928	0	-0.832432	2.0
1	-0.478484	0	2.0	1.527329	-1.198151	-0.551341	1.0	0.754157	0	0.105664	1.0
2	-1.751359	1	1.0	-0.141161	0.691874	-0.551341	2.0	-1.525138	0	-0.832432	2.0
3	-0.584556	0	0.0	0.303769	-0.574259	-0.551341	1.0	-1.132156	1	0.574711	1.0
4	0.051881	1	2.0	0.971166	-0.922904	-0.551341	1.0	-0.581981	0	-0.832432	2.0

Figura 4: Resultados de estandarización

3.2. PARTE II

Antes de construir el modelo, se realiza la **división de los datos** en conjuntos de entrenamiento y prueba, con el fin de evaluar el desempeño del modelo en datos no vistos durante el ajuste:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42, shuffle = True)
```

3.2.1. Creación del modelo de regresión logística

Una vez preprocesadas y codificadas las variables, se ajusta un modelo de **regresión logística** utilizando el conjunto de entrenamiento[2, 3]:

```
from sklearn.linear_model import LogisticRegression

# Creación del modelo de regresión logística sin regularización
modelo = LogisticRegression(penalty=None)

# Ajuste del modelo con los datos de entrenamiento
modelo.fit(X_train, y_train)
```

3.2.2. Predicción

Una vez ajustado el modelo de regresión logística, se realizan predicciones sobre el conjunto de prueba para evaluar su desempeño:

```
# Prediccion de la variable dependiente para los datos de prueba
prediccion = modelo.predict(X_test)

# Convertir a DataFrame para visualizar los resultados
predicciones = pd.DataFrame(prediccion, columns=['Prediction'])

# Mostrar las primeras 3 predicciones
predicciones.head(3)
```

	Predicción
0	0
1	1
2	1

Cuadro 4: Resultados de la predicción

3.2.3. Coeficientes e Intercepto del Modelo

A continuación se presentan los coeficientes estimados por el modelo de Regresión Logística junto con el intercepto. Estos valores se obtuvieron tras el ajuste del modelo con las variables estandarizadas:

```python
# Obtener coeficientes e intercepto
coeficientes = modelo.coef_[0]
intercepto = modelo.intercept_[0]
feature_names = X_train.columns.tolist()

print("\n--- ECUACION DE REGRESION LOGISTICA ---")
print(f"\nIntercepto (\\beta_0): {intercepto:.6f}")
print("\nCoeficientes (\\beta_i):")
for i, (nombre, coef) in enumerate(zip(feature_names, coeficientes)):
    print(f"  \\beta_{i+1} ({nombre}): {coef:.6f}")
```

Los resultados obtenidos fueron los siguientes:

```
--- ECUACION DE REGRESION LOGISTICA ---

Intercepto (\beta_0): 1.662323

Coeficientes (\beta_i):
  \beta_1 (Age): 0.148690
  \beta_2 (Sex): 1.433789
  \beta_3 (ChestPainType): -0.590140
  \beta_4 (RestingBP): 0.045719
  \beta_5 (Cholesterol): 0.101711
  \beta_6 (FastingBS): 0.488293
  \beta_7 (RestingECG): -0.130859
  \beta_8 (MaxHR): -0.301389
  \beta_9 (ExerciseAngina): 1.179116
  \beta_10 (Oldpeak): 0.441164
  \beta_11 (ST_Slope): -1.712709
```

Interpretación:

- El intercepto ($\beta_0 = 1.662323$) representa el logaritmo de las probabilidades de presentar enfermedad cardiaca cuando todas las variables independientes son cero (en su escala estandarizada).
- Los coeficientes (β_i) indican la dirección y magnitud del efecto de cada variable sobre la probabilidad de tener enfermedad cardiaca.
- Por ejemplo, el coeficiente de `Sex` ($\beta_2 = 1.433789$) sugiere que ser hombre incrementa las probabilidades de padecer enfermedad cardiaca.
- En cambio, variables como `ChestPainType` ($\beta_3 = -0.590140$) y `ST_Slope` ($\beta_{11} = -1.712709$) muestran una relación negativa con la presencia de la enfermedad, indicando que ciertos tipos de dolor de pecho y pendiente del segmento ST están asociados con menor riesgo.

3.2.4. Ecuación como String

Se construye la ecuación de regresión logística como log-odds, utilizando los coeficientes estimados:

```python
# Construir la ecuacion como string
ecuacion_partes = [f"{intercepto:.4f}"]
i = 0
for nombre, coef in zip(feature_names, coeficientes):
    i += 1
    print(f"x{i} ---> {nombre}")
    signo = "+" if coef >= 0 else ""
    ecuacion_partes.append(f"{signo}{coef:.4f}*x{i}")

ecuacion = " ".join(ecuacion_partes)
print(f"\n--- ECUACION COMPLETA (log-odds) ---")
print(f"log(p/(1-p)) = {ecuacion}")
```

Resultados:

```
x1 ---> Age
x2 ---> Sex
x3 ---> ChestPainType
x4 ---> RestingBP
x5 ---> Cholesterol
x6 ---> FastingBS
x7 ---> RestingECG
x8 ---> MaxHR
x9 ---> ExerciseAngina
x10 ---> Oldpeak
x11 ---> ST_Slope

--- ECUACION COMPLETA (log-odds) ---
log(p/(1-p)) = 1.6623 +0.1487*x1 +1.4338*x2 -0.5901*x3 +0.0457*x4 +0.1017*
   x5 +0.4883*x6 -0.1309*x7 -0.3014*x8 +1.1791*x9 +0.4412*x10 -1.7127*x11
```

3.2.5. Interpretación de coeficientes

En esta sección se realiza la interpretación de los coeficientes obtenidos del modelo de regresión logística. Los coeficientes indican el cambio en el **log-odds** de la variable dependiente (`HeartDisease`) por cada unidad de cambio en la variable independiente, manteniendo constantes las demás variables.

A continuación se presentan los resultados calculados para cada variable independiente.

```python
    print("\n--- INTERPRETACI N DE COEFICIENTES ---")
print("""
Los coeficientes representan el cambio en el LOG-ODDS (logaritmo de las
   probabilidades)
de la variable dependiente por cada unidad de cambio en la variable
   independiente,
manteniendo todas las demás variables constantes.

- Coeficiente POSITIVO: aumenta la probabilidad del evento (HeartDisease
   =1)
- Coeficiente NEGATIVO: disminuye la probabilidad del evento

Para interpretar en términos de ODDS RATIO (más intuitivo):
```

```
  Odds Ratio = exp(coeficiente)

  Si OR > 1: La variable aumenta las probabilidades del evento
  Si OR < 1: La variable disminuye las probabilidades del evento
  Si OR = 1: La variable no tiene efecto
""")

# Calcular Odds Ratios
print("\n--- ODDS RATIOS ---")
for nombre, coef in zip(feature_names, coeficientes):
    odds_ratio = np.exp(coef)
    print(f"  {nombre}: OR = {odds_ratio:.4f}")
    if odds_ratio > 1:
        print(f"     Incrementa las probabilidades en {(odds_ratio-1)
           *100:.2f}%")
    else:
        print(f"     Reduce las probabilidades en {(1-odds_ratio)*100:.2f
           }%")
```

3.3. PARTE III

Para este apartado, se ajustó un modelo de *Regresión Logística* utilizando la librería `statsmodels`, que permite obtener información estadística detallada del ajuste, incluyendo métricas como el *Pseudo R-cuadrado*, el *Log-Likelihood*, el *LL-Null* y el *LLR p-value* [5].

El modelo se entrenó agregando una constante (intercepto) a la matriz de predictores:

```
    # Agregar constante (intercepto) a X_train para statsmodels
X_train_sm = sm.add_constant(X_train)
X_test_sm = sm.add_constant(X_test)

# Entrenar modelo con statsmodels para obtener estadísticas completas
modelo_statsmodels = sm.Logit(y_train, X_train_sm).fit(disp=0)
# Mostrar resumen estadístico del modelo
print(modelo_statsmodels.summary())
```

Logit Regression Results

Dep. Variable:	HeartDisease	No. Observations:	734
Model:	Logit	Df Residuals:	722
Method:	MLE	Df Model:	11
Date:	Tue, 21 Oct 2025	Pseudo R-squ.:	0.4732
Time:	05:01:36	Log-Likelihood:	-266.38
converged:	True	LL-Null:	-505.62
Covariance Type:	nonrobust	LLR p-value:	1.250e-95

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.6636	0.433	3.846	0.000	0.816	2.511
Age	0.1493	0.130	1.146	0.252	-0.106	0.405
Sex	1.4331	0.286	5.010	0.000	0.872	1.994
ChestPainType	-0.5901	0.121	-4.864	0.000	-0.828	-0.352
RestingBP	0.0460	0.121	0.381	0.703	-0.191	0.283
Cholesterol	0.1017	0.129	0.791	0.429	-0.151	0.354
FastingBS	0.4879	0.120	4.076	0.000	0.253	0.723
RestingECG	-0.1306	0.184	-0.710	0.478	-0.491	0.230
MaxHR	-0.3013	0.133	-2.272	0.023	-0.561	-0.041
ExerciseAngina	1.1790	0.255	4.615	0.000	0.678	1.680
Oldpeak	0.4408	0.135	3.272	0.001	0.177	0.705
ST_Slope	-1.7134	0.225	-7.610	0.000	-2.155	-1.272

Figura 5: Resultados de la regresión logística

3.3.1. Extraer métricas

En esta sección se analizan los principales indicadores estadísticos derivados del modelo de regresión logística, utilizando la librería `statsmodels`. Estos valores permiten evaluar la calidad del ajuste, la capacidad explicativa del modelo y la significancia global de sus parámetros.

```
    pseudo_r2 = modelo_statsmodels.prsquared  # McFadden's Pseudo R_dos
log_likelihood = modelo_statsmodels.llf  # Log-Likelihood del modelo completo
ll_null = modelo_statsmodels.llnull  # Log-Likelihood del modelo nulo
llr = modelo_statsmodels.llr  # Likelihood Ratio (LR)
llr_pvalue = modelo_statsmodels.llr_pvalue  # p-value del LR test
```

3.3.2. Pseudo R-cuadrada (McFadden)

```
# III.1 - Pseudo R_dos
print(f"\n========== III.1: Pseudo R-cuadrada (McFadden) ==========")
print(f"Valor: {pseudo_r2:.4f}")
print("""
INTERPRETACION:
El Pseudo R_dos de McFadden mide la mejora proporcional del modelo completo
```

```
respecto al modelo nulo (solo con intercepto).

Fórmula: Pseudo R_dos = 1 - (Log-Likelihood / LL-Null)

Valores de referencia (McFadden, 1974):
  - 0.2 a 0.4: ajuste EXCELENTE
  - 0.1 a 0.2: ajuste BUENO
  - < 0.1: ajuste POBRE

NO es equivalente al R_dos de regresión lineal. Valores más bajos son normales
y aceptables en regresión logística.
""")
if pseudo_r2 >= 0.2:
    print(f" Con {pseudo_r2:.4f}, tu modelo tiene un AJUSTE EXCELENTE")
elif pseudo_r2 >= 0.1:
    print(f" Con {pseudo_r2:.4f}, tu modelo tiene un AJUSTE BUENO")
else:
    print(f" Con {pseudo_r2:.4f}, tu modelo tiene un AJUSTE POBRE")
```

Obteniendo:

```
    ========== III.1: Pseudo R-cuadrada (McFadden) ==========
Valor: 0.4732

INTERPRETACI N:
El Pseudo R_dos de McFadden mide la mejora proporcional del modelo completo
respecto al modelo nulo (solo con intercepto).

Fórmula: Pseudo R_dos = 1 - (Log-Likelihood / LL-Null)

Valores de referencia (McFadden, 1974):
  - 0.2 a 0.4: ajuste EXCELENTE
  - 0.1 a 0.2: ajuste BUENO
  - < 0.1: ajuste POBRE

NO es equivalente al R_dos de regresión lineal. Valores más bajos son normales
y aceptables en regresión logística.

Con 0.4732, tu modelo tiene un AJUSTE EXCELENTE
```

El valor de 0.4732 indica que el modelo presenta un **ajuste excelente**, lo que sugiere que las variables predictoras explican una proporción importante de la variabilidad en la probabilidad de presentar enfermedad cardíaca.

3.3.3. Log-Likelihood

```
# III.2 - Log-Likelihood
print(f"\n========== III.2: Log-Likelihood ==========")
print(f"Valor: {log_likelihood:.4f}")
print("""
```

```
INTERPRETACION:
El Log-Likelihood (LL) es el logaritmo natural de la función de
    verosimilitud.
Mide qué tan bien el modelo se ajusta a los datos.

- Valores MAS CERCANOS A 0 indican MEJOR ajuste
- Valores MAS NEGATIVOS indican PEOR ajuste
- El LL por si solo NO es interpretable; se usa para COMPARAR modelos

Un modelo perfecto tendria LL = 0, pero esto es imposible en la practica.
""")
print(f" Tu modelo tiene un Log-Likelihood de {log_likelihood:.4f}")
```

Obteniendo:

```
    ========== III.2: Log-Likelihood ==========
Valor: -266.3836

INTERPRETACION:
El Log-Likelihood (LL) es el logaritmo natural de la función de
    verosimilitud.
Mide qué tan bien el modelo se ajusta a los datos.

- Valores MAS CERCANOS A 0 indican MEJOR ajuste
- Valores MAS NEGATIVOS indican PEOR ajuste
- El LL por sí solo NO es interpretable; se usa para COMPARAR modelos

Un modelo perfecto tendría LL = 0, pero esto es imposible en la práctica.

Tu modelo tiene un Log-Likelihood de -266.3836
```

En este caso, el valor de -266.3836 indica un buen ajuste general, ya que es considerablemente mayor (menos negativo) que el del modelo nulo, reflejando una mejora significativa en la capacidad explicativa del modelo.

3.3.4. LL-Null (Log-Likelihood Nulo)

```
# III.3 - LL-Null
print(f"\n========== III.3: LL-Null (Log-Likelihood Nulo) ==========")
print(f"Valor: {ll_null:.4f}")
print("""
INTERPRETACI N:
LL-Null es el Log-Likelihood del modelo "nulo" o baseline.
El modelo nulo SOLO incluye el intercepto (sin variables independientes).

Este valor sirve como punto de referencia para evaluar si tu modelo
    completo
aporta información útil para predecir la variable dependiente.

Si tu modelo completo tiene un LL mucho mejor (más cercano a 0) que LL-
    Null,
significa que tus variables independientes S  ayudan a explicar la
    variable dependiente.
```

```
""")
mejora = ll_null - log_likelihood
print(f"    Mejora del modelo completo vs nulo: {mejora:.4f}")
```

Obteniendo:

```
    ========== III.3: LL-Null (Log-Likelihood Nulo) ==========
Valor: -505.6156

INTERPRETACION:
LL-Null es el Log-Likelihood del modelo "nulo" o baseline.
El modelo nulo SOLO incluye el intercepto (sin variables independientes).

Este valor sirve como punto de referencia para evaluar si tu modelo
    completo
aporta información útil para predecir la variable dependiente.

Si tu modelo completo tiene un LL mucho mejor (más cercano a 0) que LL-
    Null,
significa que tus variables independientes SI ayudan a explicar la
    variable dependiente.

 Mejora del modelo completo vs nulo: -239.2320
```

Valor obtenido: LL-Null $= -505.6156$

El *LL-Null* corresponde al logaritmo de la verosimilitud de un modelo sin variables independientes (solo con intercepto). Sirve como punto de comparación para evaluar cuánto mejora el modelo completo.

$$\text{Mejora} = \text{LL-Null} - \text{Log-Likelihood} = -239.2320$$

Esto indica una mejora sustancial, lo que confirma que las variables independientes aportan información significativa para predecir la enfermedad cardíaca.

3.3.5. LLR p-value (Prueba de Razón de Verosimilitudes)

```
# III.4 - LLR p-value
print(f"\n========== III.4: LLR p-value (Likelihood Ratio Test) ==========
    ")
print(f"Valor: {llr_pvalue:.10f}")
print(f"Estadístico LR: {llr:.4f}")
print("""
INTERPRETACION:
El LLR p-value prueba la hipótesis nula de que el modelo nulo (solo
    intercepto)
es tan bueno como el modelo completo (con todas las variables).

Hipótesis:
  H0: El modelo nulo es suficiente (las variables NO aportan información)
  H1: El modelo completo es mejor (las variables SI aportan información)

Criterio de decision (alpha = 0.05):
```

```
  - Si p-value < 0.05: RECHAZAMOS H₀ - El modelo completo es
      significativamente mejor
  - Si p-value ≥ 0.05: NO rechazamos H₀ - El modelo no es
      significativamente mejor
""")
if llr_pvalue < 0.001:
    print(f"- Con p-value = {llr_pvalue:.2e} (< 0.001), el modelo es
       ALTAMENTE SIGNIFICATIVO")
    print("   Las variables independientes explican de manera
       significativa la variable dependiente")
elif llr_pvalue < 0.05:
    print(f"- Con p-value = {llr_pvalue:.4f} (< 0.05), el modelo es
       SIGNIFICATIVO")
else:
    print(f"- Con p-value = {llr_pvalue:.4f} (≥ 0.05), el modelo NO es
       significativo")
```

Obteniendo:

```
    ========== III.4: LLR p-value (Likelihood Ratio Test) ==========
Valor: 0.0000000000
Estadístico LR: 478.4641

INTERPRETACION:
El LLR p-value prueba la hipótesis nula de que el modelo nulo (solo
   intercepto)
es tan bueno como el modelo completo (con todas las variables).
Hipótesis:
  H₀: El modelo nulo es suficiente (las variables NO aportan información)
  H₁: El modelo completo es mejor (las variables SI aportan información)

Criterio de decisión (alpha = 0.05):
  - Si p-value < 0.05: RECHAZAMOS H₀ - El modelo completo es
      significativamente mejor
  - Si p-value ≥ 0.05: NO rechazamos H₀ - El modelo no es
      significativamente mejor

- Con p-value = 1.25e-95 (< 0.001), el modelo es ALTAMENTE SIGNIFICATIVO
   Las variables independientes explican de manera significativa la
      variable dependiente
```

Resultados:

$$\text{Estadístico LR} = 478.4641, \quad p\text{-value} = 1.25 \times 10^{-95}$$

El *LLR p-value* contrasta dos hipótesis:

$$H_0 : \text{El modelo nulo (sin variables) es suficiente}$$

$$H_1 : \text{El modelo completo es significativamente mejor}$$

Con un valor de $p < 0.001$, se rechaza contundentemente H_0, concluyendo que el modelo completo es altamente significativo. Esto demuestra que las variables independientes incluidas explican de manera estadísticamente significativa la variable dependiente.

3.4. PARTE IV

Se analizan los resultados obtenidos del modelo de regresión logística completo, con base en los valores de la tabla generada mediante `modelo_statsmodels.summary()` y los coeficientes extraídos en un `DataFrame` personalizado.

```
    resultados_df = pd.DataFrame({
    'Variable': ['Intercepto'] + feature_names,
    'coef': modelo_statsmodels.params.values,
    'std_err': modelo_statsmodels.bse.values,
    'z': modelo_statsmodels.tvalues.values,
    'p_value': modelo_statsmodels.pvalues.values,
    'CI_lower': modelo_statsmodels.conf_int()[0].values,
    'CI_upper': modelo_statsmodels.conf_int()[1].values
})

print("\n--- TABLA COMPLETA DE RESULTADOS ---")
print(resultados_df.to_string(index=False))
```

Obteniendo:

```
--- TABLA COMPLETA DE RESULTADOS ---
      Variable      coef  std_err         z      p_value  CI_lower  CI_upper
     Intercepto  1.663575 0.432525  3.846189 1.199690e-04  0.815841  2.511309
           Age  0.149275 0.130274  1.145851 2.518568e-01 -0.106058  0.404608
           Sex  1.433095 0.286038  5.010147 5.438835e-07  0.872470  1.993720
 ChestPainType -0.590115 0.121322 -4.864054 1.150056e-06 -0.827901 -0.352329
     RestingBP  0.046043 0.120790  0.381180 7.030700e-01 -0.190701  0.282787
   Cholesterol  0.101749 0.128710  0.790525 4.292211e-01 -0.150519  0.354017
     FastingBS  0.487935 0.119700  4.076314 4.575527e-05  0.253327  0.722543
    RestingECG -0.130590 0.183870 -0.710227 4.775633e-01 -0.490969  0.229790
         MaxHR -0.301256 0.132578 -2.272285 2.306933e-02 -0.561105 -0.041407
ExerciseAngina  1.179007 0.255460  4.615228 3.926649e-06  0.678314  1.679699
       Oldpeak  0.440785 0.134703  3.272266 1.066891e-03  0.176772  0.704799
      ST_Slope -1.713367 0.225141 -7.610191 2.736916e-14 -2.154635 -1.272098
```

Figura 6: Resultados del DataFrame

3.4.1. Interpretación de coeficientes

```
    # IV.1 - Interpretación de coeficientes
print(f"\n--- IV.1: Interpretación de cada coeficiente ---")
print("""
El coeficiente (coef) representa el cambio en el LOG-ODDS por cada unidad
de cambio en la variable independiente (manteniendo otras variables
   constantes).

En datos ESTANDARIZADOS (como los tuyos):
  - El coeficiente representa el cambio en log-odds por cada DESVIACION
     ESTANDAR
    de cambio en la variable
  - Esto permite COMPARAR directamente la importancia de las variables
```

```
  - Variables con |coeficiente| mayor tienen mayor impacto en la predicció
    n
""")

for idx, row in resultados_df.iterrows():
    if row['Variable'] == 'Intercepto':
        continue
    var = row['Variable']
    coef = row['coef']
    odds_ratio = np.exp(coef)

    print(f"\n{var}:")
    print(f"  Coeficiente: {coef:.6f}")
    print(f"  Odds Ratio: {odds_ratio:.6f}")

    if coef > 0:
        print(f"   Efecto POSITIVO: aumentar 1 desviación estándar en {var
            }")
        print(f"     multiplica las probabilidades de HeartDisease por {
            odds_ratio:.4f}")
    else:
        print(f"   Efecto NEGATIVO: aumentar 1 desviación estándar en {var
            }")
        print(f"     multiplica las probabilidades de HeartDisease por {
            odds_ratio:.4f}")
```

Teniendo como resultados del análisis:

```
    --- IV.1: Interpretación de cada coeficiente ---

El coeficiente (coef) representa el cambio en el LOG-ODDS por cada unidad
de cambio en la variable independiente (manteniendo otras variables
   constantes).

En datos ESTANDARIZADOS (como los tuyos):
  - El coeficiente representa el cambio en log-odds por cada DESVIACION
    ESTANDAR
    de cambio en la variable
  - Esto permite COMPARAR directamente la importancia de las variables
  - Variables con |coeficiente| mayor tienen mayor impacto en la predicció
    n


Age:
  Coeficiente: 0.149275
  Odds Ratio: 1.160992
  - Efecto POSITIVO: aumentar 1 desviación estándar en Age
     multiplica las probabilidades de HeartDisease por 1.1610

Sex:
  Coeficiente: 1.433095
  Odds Ratio: 4.191652
  - Efecto POSITIVO: aumentar 1 desviación estándar en Sex
     multiplica las probabilidades de HeartDisease por 4.1917

```

```
ChestPainType:
  Coeficiente: -0.590115
  Odds Ratio: 0.554263
  - Efecto NEGATIVO: aumentar 1 desviación estándar en ChestPainType
     multiplica las probabilidades de HeartDisease por 0.5543

RestingBP:
  Coeficiente: 0.046043
  Odds Ratio: 1.047119
  - Efecto POSITIVO: aumentar 1 desviación estándar en RestingBP
     multiplica las probabilidades de HeartDisease por 1.0471

Cholesterol:
  Coeficiente: 0.101749
  Odds Ratio: 1.107105
  - Efecto POSITIVO: aumentar 1 desviación estándar en Cholesterol
     multiplica las probabilidades de HeartDisease por 1.1071

FastingBS:
  Coeficiente: 0.487935
  Odds Ratio: 1.628950
  - Efecto POSITIVO: aumentar 1 desviación estándar en FastingBS
     multiplica las probabilidades de HeartDisease por 1.6289

RestingECG:
  Coeficiente: -0.130590
  Odds Ratio: 0.877578
  - Efecto NEGATIVO: aumentar 1 desviación estándar en RestingECG
     multiplica las probabilidades de HeartDisease por 0.8776

MaxHR:
  Coeficiente: -0.301256
  Odds Ratio: 0.739888
  - Efecto NEGATIVO: aumentar 1 desviación estándar en MaxHR
     multiplica las probabilidades de HeartDisease por 0.7399

ExerciseAngina:
  Coeficiente: 1.179007
  Odds Ratio: 3.251143
  - Efecto POSITIVO: aumentar 1 desviación estándar en ExerciseAngina
     multiplica las probabilidades de HeartDisease por 3.2511

Oldpeak:
  Coeficiente: 0.440785
  Odds Ratio: 1.553927
  - Efecto POSITIVO: aumentar 1 desviación estándar en Oldpeak
     multiplica las probabilidades de HeartDisease por 1.5539

ST_Slope:
  Coeficiente: -1.713367
  Odds Ratio: 0.180258
  - Efecto NEGATIVO: aumentar 1 desviación estándar en ST_Slope
     multiplica las probabilidades de HeartDisease por 0.1803
```

3.4.2. Interpretación de std err

```
    # IV.2 - Interpretación de std err
print(f"\n--- IV.2: Interpretación del Error Estándar (std err) ---")
print("""
El error estándar mide la VARIABILIDAD o INCERTIDUMBRE del coeficiente
    estimado.

- Error estándar DIMINUTO: el coeficiente es más PRECISO y CONFIABLE
- Error estándar GRANDE: el coeficiente tiene más INCERTIDUMBRE

El error estándar se usa para:
  1. Calcular el estadístico z: z = coef / std_err
  2. Construir intervalos de confianza
  3. Evaluar la precisión de las estimaciones
""")

print("\nVariables ordenadas por precisión (menor error estándar):")
resultados_ordenados = resultados_df[resultados_df['Variable'] != '
    Intercepto'].sort_values('std_err')
print(resultados_ordenados[['Variable', 'coef', 'std_err']].to_string(
    index=False))
```

Obteniendo:

```
    --- IV.2: Interpretación del Error Estándar (std err) ---

El error estándar mide la VARIABILIDAD o INCERTIDUMBRE del coeficiente
    estimado.

- Error estándar DIMINUTO: el coeficiente es más PRECISO y CONFIABLE
- Error estándar GRANDE: el coeficiente tiene más INCERTIDUMBRE

El error estándar se usa para:
  1. Calcular el estadístico z: z = coef / std_err
  2. Construir intervalos de confianza
  3. Evaluar la precisión de las estimaciones


Variables ordenadas por precisión (menor error estándar):
      Variable      coef  std_err
     FastingBS  0.487935 0.119700
     RestingBP  0.046043 0.120790
 ChestPainType -0.590115 0.121322
   Cholesterol  0.101749 0.128710
           Age  0.149275 0.130274
         MaxHR -0.301256 0.132578
       Oldpeak  0.440785 0.134703
    RestingECG -0.130590 0.183870
      ST_Slope -1.713367 0.225141
ExerciseAngina  1.179007 0.255460
           Sex  1.433095 0.286038
```

3.4.3. Variables significativas

```
    # IV.3 - Variables significativas
print(f"\n--- IV.3: Variables significativas (alpha = 0.05) ---")
print("""
Un variable es SIGNIFICATIVA si su p-value < 0.05 (nivel de significancia
    común).

Esto significa que hay menos del 5% de probabilidad de que el coeficiente
    observado
sea debido al azar (rechazamos H₀: β = 0).

Variables significativas: REALMENTE explican parte de la variable
    dependiente
Variables NO significativas: NO aportan información estadísticamente
    relevante
""")

variables_significativas = resultados_df[
    (resultados_df['Variable'] != 'Intercepto') &
    (resultados_df['p_value'] < 0.05)
].sort_values('p_value')

print(f"\nVariables SIGNIFICATIVAS (p < 0.05):")
if len(variables_significativas) > 0:
    print(variables_significativas[['Variable', 'coef', 'p_value']].
        to_string(index=False))
    print(f"\n {len(variables_significativas)} de {len(feature_names)}
        variables SI explican la variable dependiente")
else:
    print("NINGUNA variable es significativa")

variables_no_significativas = resultados_df[
    (resultados_df['Variable'] != 'Intercepto') &
    (resultados_df['p_value'] >= 0.05)
]

print(f"\nVariables NO SIGNIFICATIVAS (p ≥ 0.05):")
if len(variables_no_significativas) > 0:
    print(variables_no_significativas[['Variable', 'coef', 'p_value']].
        to_string(index=False))
    print("\n Estas variables podrían ser candidatas para eliminación")
else:
    print("Todas las variables son significativas")
```

Obteniendo:

```
    --- IV.3: Variables significativas (alpha = 0.05) ---

Un variable es SIGNIFICATIVA si su p-value < 0.05 (nivel de significancia
    común).

Esto significa que hay menos del 5% de probabilidad de que el coeficiente
    observado
sea debido al azar (rechazamos H₀: β = 0).
```

```
Variables significativas: REALMENTE explican parte de la variable dependiente
Variables NO significativas: NO aportan información estadísticamente relevante


Variables SIGNIFICATIVAS (p < 0.05):
      Variable      coef      p_value
      ST_Slope -1.713367 2.736916e-14
           Sex  1.433095 5.438835e-07
 ChestPainType -0.590115 1.150056e-06
ExerciseAngina  1.179007 3.926649e-06
     FastingBS  0.487935 4.575527e-05
       Oldpeak  0.440785 1.066891e-03
         MaxHR -0.301256 2.306933e-02

- 7 de 11 variables SI explican la variable dependiente

Variables NO SIGNIFICATIVAS (p ≥ 0.05):
   Variable      coef  p_value
        Age  0.149275 0.251857
  RestingBP  0.046043 0.703070
Cholesterol  0.101749 0.429221
 RestingECG -0.130590 0.477563

- Estas variables podrían ser candidatas para eliminación
```

3.4.4. Variables con mayor poder explicativo (valor z)

```python
    # IV.4 - Variables con mayor poder explicativo (valor z)
print(f"\n--- IV.4: Variables con mayor poder explicativo (según |z|) ---")
print("""
El estadístico z (valor z) mide cuántas desviaciones estándar el coeficiente
está alejado de cero.

- |z| GRANDE: La variable tiene un efecto FUERTE y SIGNIFICATIVO
- |z| DIMINUTO: La variable tiene un efecto DEBIL o NO SIGNIFICATIVO

Valores de referencia:
  |z| > 2.576 - p < 0.01 (muy significativo)
  |z| > 1.960 - p < 0.05 (significativo)
  |z| < 1.960 - p ≥ 0.05 (no significativo)
""")

top_variables = resultados_df[resultados_df['Variable'] != 'Intercepto'].copy()
top_variables['abs_z'] = np.abs(top_variables['z'])
top_variables = top_variables.sort_values('abs_z', ascending=False)

print("\nVariables ordenadas por |z| (poder explicativo):")
```

```
print(top_variables[['Variable', 'coef', 'z', 'abs_z', 'p_value']].head
    (10).to_string(index=False))

top_2 = top_variables.head(2)
print(f"\n- Las 2 variables que MEJOR explican la variable dependiente son
    :")
for idx, row in top_2.iterrows():
    print(f"  {idx+1}. {row['Variable']} (|z| = {row['abs_z']:.4f}, p = {
        row['p_value']:.6f})")
```

Obtenemos:

```
    --- IV.4: Variables con mayor poder explicativo (según |z|) ---

El estadístico z (valor z) mide cuántas desviaciones estándar el
    coeficiente
está alejado de cero.

- |z| GRANDE: La variable tiene un efecto FUERTE y SIGNIFICATIVO
- |z| DIMINUTO: La variable tiene un efecto DEBIL o NO SIGNIFICATIVO

Valores de referencia:
  |z| > 2.576 - p < 0.01 (muy significativo)
  |z| > 1.960 - p < 0.05 (significativo)
  |z| < 1.960 - p ≥ 0.05 (no significativo)


Variables ordenadas por |z| (poder explicativo):
      Variable      coef         z    abs_z      p_value
      ST_Slope -1.713367 -7.610191 7.610191 2.736916e-14
           Sex  1.433095  5.010147 5.010147 5.438835e-07
 ChestPainType -0.590115 -4.864054 4.864054 1.150056e-06
ExerciseAngina  1.179007  4.615228 4.615228 3.926649e-06
     FastingBS  0.487935  4.076314 4.076314 4.575527e-05
       Oldpeak  0.440785  3.272266 3.272266 1.066891e-03
         MaxHR -0.301256 -2.272285 2.272285 2.306933e-02
           Age  0.149275  1.145851 1.145851 2.518568e-01
   Cholesterol  0.101749  0.790525 0.790525 4.292211e-01
    RestingECG -0.130590 -0.710227 0.710227 4.775633e-01

- Las 2 variables que MEJOR explican la variable dependiente son:
  12. ST_Slope (|z| = 7.6102, p = 0.000000)
  3. Sex (|z| = 5.0101, p = 0.000001)
```

3.4.5. Intervalo de confianza del 95 % para la mejor variable

```
    # IV.5 - Intervalo de confianza del 95% para la mejor variable
print(f"\n--- IV.5: Intervalo de Confianza 95% para la mejor variable ---"
    )

mejor_variable = top_variables.iloc[0]
var_name = mejor_variable['Variable']
coef_val = mejor_variable['coef']
```

```
ci_lower = mejor_variable['CI_lower']
ci_upper = mejor_variable['CI_upper']

print(f"\nMejor variable: {var_name}")
print(f"Coeficiente: {coef_val:.6f}")
print(f"Intervalo de Confianza 95%: [{ci_lower:.6f}, {ci_upper:.6f}]")

print("""
INTERPRETACION:
El intervalo de confianza del 95% indica que estamos 95% seguros de que el
VERDADERO valor del coeficiente poblacional está dentro de este rango.

Si el intervalo NO contiene el 0, la variable es significativa (p < 0.05).
Cuanto más ESTRECHO el intervalo, más PRECISA es la estimación.
""")

if ci_lower > 0 and ci_upper > 0:
    print(f"- El IC no contiene 0 y es completamente POSITIVO")
    print(f"   {var_name} tiene un efecto POSITIVO significativo y
       confiable")
elif ci_lower < 0 and ci_upper < 0:
    print(f"- El IC no contiene 0 y es completamente NEGATIVO")
    print(f"   {var_name} tiene un efecto NEGATIVO significativo y
       confiable")
else:
    print(f"- El IC contiene 0: la variable NO es significativa")
```

Obtenemos:

```
    --- IV.5: Intervalo de Confianza 95% para la mejor variable ---

Mejor variable: ST_Slope
Coeficiente: -1.713367
Intervalo de Confianza 95%: [-2.154635, -1.272098]

INTERPRETACION:
El intervalo de confianza del 95% indica que estamos 95% seguros de que el
VERDADERO valor del coeficiente poblacional está dentro de este rango.

Si el intervalo NO contiene el 0, la variable es significativa (p < 0.05).
Cuanto más ESTRECHO el intervalo, más PRECISA es la estimación.

- El IC no contiene 0 y es completamente NEGATIVO
   ST_Slope tiene un efecto NEGATIVO significativo y confiable
```

3.4.6. Visualización de los Intervalos de Confianza

La Figura 7 presenta los coeficientes estimados para las cinco variables más relevantes, junto con sus intervalos de confianza al 95 %. En donde Los puntos representan los coeficientes estimados ($\hat{\beta}$). Las barras horizontales indican el intervalo de confianza. La línea roja vertical ($x = 0$) representa el punto de "sin efecto"

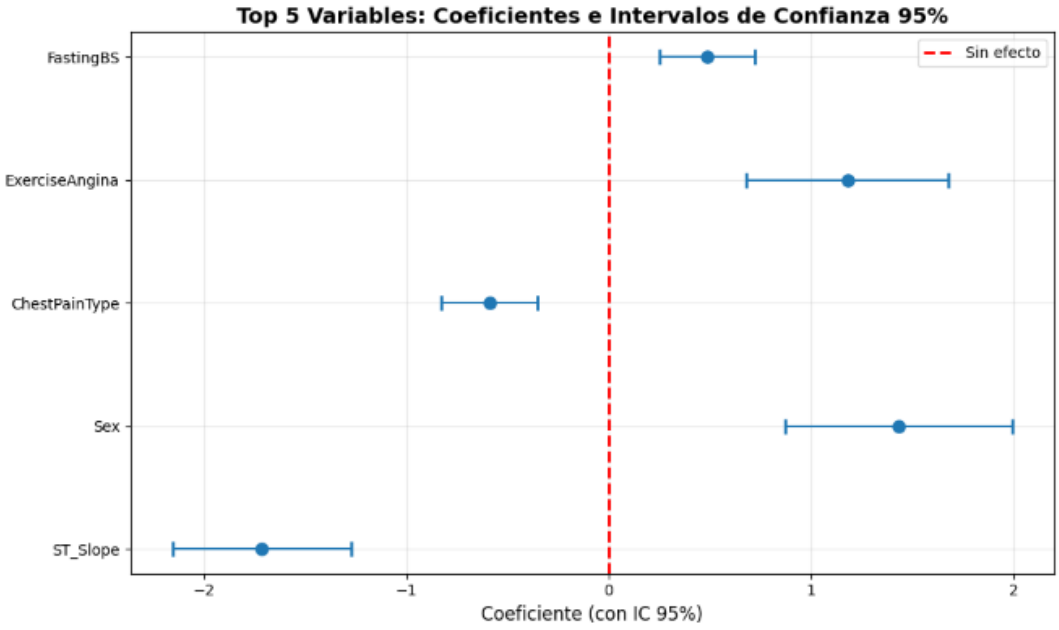

Figura 7: Coeficientes e intervalos de confianza al 95 % para las principales variables del modelo.

Interpretación: Ninguna de las variables seleccionadas cruza la línea de referencia, lo que confirma su significancia estadística. Las variables **ST_Slope** y **Sex** presentan los intervalos más alejados de cero, siendo las de mayor peso explicativo en la probabilidad de enfermedad cardíaca.

3.5. PARTE V

En esta sección se evalúa el desempeño del modelo de regresión logística mediante la construcción de la matriz de confusión. Esta matriz permite comparar las predicciones del modelo (*Predicho*) con los valores reales (*Actual*) de la variable dependiente (`HeartDisease`[5][1].

```
    # Predicciones en conjunto de prueba
y_pred = modelo.predict(X_test)

# Crear matriz de confusión
cm = confusion_matrix(y_test, y_pred)

print("\nMatriz de Confusión:")
print(cm)
print("""
Estructura:
                    Predicho
                    0       1
Actual    0     [TN]   [FP]
          1     [FN]   [TP]

TN (True Negative): Correctamente predicho como NO enfermedad
TP (True Positive): Correctamente predicho como SI enfermedad
FN (False Negative): Incorrectamente predicho como NO (era SI)
FP (False Positive): Incorrectamente predicho como SI (era NO)
```

```
""")

tn, fp, fn, tp = cm.ravel()
print(f"True Negatives (TN): {tn}")
print(f"False Positives (FP): {fp}")
print(f"False Negatives (FN): {fn}")
print(f"True Positives (TP): {tp}")

# Visualizar matriz de confusión
fig, ax = plt.subplots(figsize=(8, 6))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['No
    Enfermedad', 'Enfermedad'])
disp.plot(cmap='Blues', ax=ax, values_format='d')
plt.title('Matriz de Confusión', fontsize=14, fontweight='bold')
```

La matriz de confusión tiene la siguiente estructura:

```
    Matriz de Confusión:
[[68  9]
 [17 90]]

 True Negatives (TN): 68
False Positives (FP): 9
False Negatives (FN): 17
True Positives (TP): 90
Text(0.5, 1.0, 'Matriz de Confusión')
```

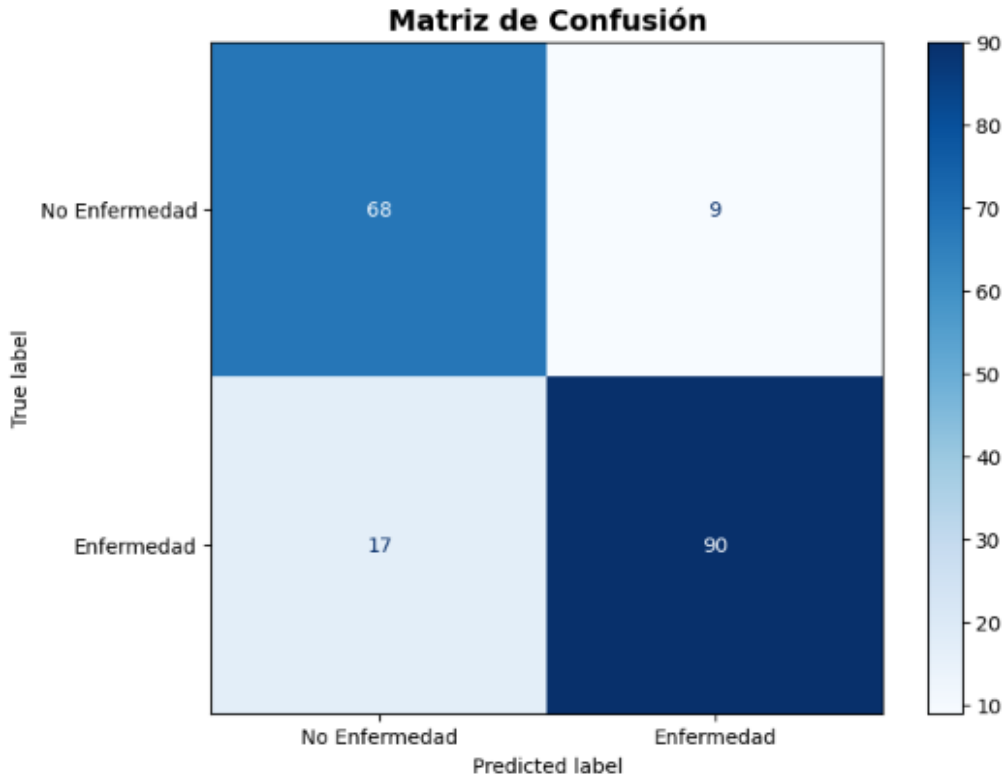

Figura 8: Matriz de confusión

3.5.1. Metricas derivadas

```
    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
recall = tp / (tp + fn) if (tp + fn) > 0 else 0
f1_score = 2 * (precision * recall) / (precision + recall) if (precision +
    recall) > 0 else 0
specificity = tn / (tn + fp) if (tn + fp) > 0 else 0

print(f"\nMétricas de Evaluación:")
print(f"  Precision (Precisión): {precision:.4f}")
print(f"  Recall (Sensibilidad): {recall:.4f}")
print(f"  F1-Score: {f1_score:.4f}")
print(f"  Specificity (Especificidad): {specificity:.4f}")
```

En donde sus resultados de evaluación son:

```
    Métricas de Evaluación:
  Precision (Precisión): 0.9091
  Recall (Sensibilidad): 0.8411
  F1-Score: 0.8738
  Specificity (Especificidad): 0.8831
```

3.5.2. Mejor modelo con máximo 4 variables (criterio LLR p-value)

En esta etapa se busca identificar las variables más significativas estadísticamente dentro del modelo de Regresión Logística. Para ello, se utiliza el criterio del *p*-value obtenido del test de razón de verosimilitudes (**Likelihood Ratio Test, LLR**). Las variables con menor *p*-value son aquellas que tienen una mayor evidencia de influencia sobre la variable dependiente (`HeartDisease`), es decir, son las más relevantes para el modelo.

A continuación se presenta el fragmento de código utilizado para seleccionar las variables más significativas:

```
    # Seleccionar top 4 variables por p-value (excluyendo intercepto)
top_4_pvalue = resultados_df[resultados_df['Variable'] != 'Intercepto'].
    nsmallest(4, 'p_value')
top_4_vars = top_4_pvalue['Variable'].tolist()

print(f"\nTop 4 variables seleccionadas (menor p-value):")
for i, row in top_4_pvalue.iterrows():
    print(f"  {row['Variable']}: p-value = {row['p_value']:.6f}")
```

Los resultados obtenidos fueron los siguientes:

```
    Top 4 variables seleccionadas (menor p-value):
  ST_Slope: p-value = 0.000000
  Sex: p-value = 0.000001
  ChestPainType: p-value = 0.000001
  ExerciseAngina: p-value = 0.000004
```

Las variables con menor *p*-value son las que más contribuyen a explicar la probabilidad de padecer una enfermedad cardíaca.En este caso,`ST_Slope`, `Sex`, `ChestPainType` y `ExerciseAngina` presentan una alta significancia estadística ($p < 0.001$), lo que indica que tienen un fuerte efecto

sobre el modelo. Estos resultados son consistentes con los obtenidos previamente mediante el criterio de Pseudo R^2, lo que refuerza la confiabilidad de la selección.

3.5.3. Entrenar modelo reducido

Una vez identificadas las cuatro variables más significativas según el criterio del *p*-value (`ST_Slope`, `Sex`, `ChestPainType` y `ExerciseAngina`), se procede a entrenar un modelo reducido de Regresión Logística. El objetivo es comparar su desempeño con el modelo completo (que incluye todas las variables) mediante métricas como el **Pseudo R^2**, el **p-value del LLR (Likelihood Ratio Test)** y el **Log-Likelihood**.

El siguiente bloque de código muestra el proceso de entrenamiento y comparación de ambos modelos:

```
    X_train_reduced = X_train[top_4_vars]
X_test_reduced = X_test[top_4_vars]

X_train_reduced_sm = sm.add_constant(X_train_reduced)
modelo_reducido = sm.Logit(y_train, X_train_reduced_sm).fit(disp=0)

print(f"\n--- Comparación de Modelos ---")
print(f"Modelo Completo ({len(feature_names)} variables):")
print(f"  Pseudo R_dos: {pseudo_r2:.4f}")
print(f"  LLR p-value: {llr_pvalue:.2e}")
print(f"  Log-Likelihood: {log_likelihood:.4f}")

print(f"\nModelo Reducido (4 variables):")
print(f"  Pseudo R_dos: {modelo_reducido.prsquared:.4f}")
print(f"  LLR p-value: {modelo_reducido.llr_pvalue:.2e}")
print(f"  Log-Likelihood: {modelo_reducido.llf:.4f}")

if modelo_reducido.llr_pvalue < llr_pvalue:
    print("\n El modelo reducido tiene un LLR p-value MEJOR (más pequeño)"
        )
else:
    print("\n El modelo completo mantiene un LLR p-value mejor")
```

Los resultados obtenidos fueron los siguientes:

```

--- Comparación de Modelos ---
Modelo Completo (11 variables):
  Pseudo R_dos: 0.4732
  LLR p-value: 1.25e-95
  Log-Likelihood: -266.3836

Modelo Reducido (4 variables):
  Pseudo R_dos: 0.4272
  LLR p-value: 3.38e-92
  Log-Likelihood: -289.6169

El modelo completo mantiene un LLR p-value mejor
```

El modelo reducido, con solo cuatro variables, mantiene un desempeño razonablemente alto ($R^2 = 0.4272$) y un *p*-value del LLR aún muy significativo (3.38×10^{-92}). No obstante, el modelo completo presenta un ajuste ligeramente mejor ($R^2 = 0.4732$) y un menor *p*-value, indicando que explica una mayor proporción de la variabilidad en los datos. Aun así, el modelo reducido es preferible en contextos donde se busca un balance entre simplicidad e interpretabilidad, sacrificando solo una pequeña parte del poder explicativo.

3.5.4. Creación de gráfica logística y diagrama de dispersión

Para visualizar la relación entre la variable más influyente del modelo y la probabilidad predicha de padecer enfermedad cardíaca, se genera una gráfica logística acompañada de un diagrama de dispersión. Este gráfico permite observar cómo el modelo ajusta la función sigmoide (*curva logística*) sobre los datos observados, y cómo la probabilidad estimada cambia a medida que la variable independiente varía.

```
    # Seleccionar la variable más importante
mejor_var_idx = feature_names.index(var_name)

# Obtener valores de esa variable
X_plot = X_test.iloc[:, mejor_var_idx].values if isinstance(X_test, pd.DataFrame) else X_test[:, mejor_var_idx]
y_true = y_test.values if isinstance(y_test, pd.Series) else y_test

# Calcular probabilidades predichas
y_proba = modelo.predict_proba(X_test)[:, 1]

# Ordenar para la curva logística
idx_sort = np.argsort(X_plot)
X_sorted = X_plot[idx_sort]
y_proba_sorted = y_proba[idx_sort]

# Crear gráfico
fig, ax = plt.subplots(figsize=(12, 7))

# Diagrama de dispersión (con jitter para visibilidad)
jitter = 0.05
y_jitter = y_true + np.random.normal(0, jitter, size=len(y_true))
ax.scatter(X_plot, y_jitter, alpha=0.3, s=30, label='Datos observados', color='steelblue')

# Curva logística
ax.plot(X_sorted, y_proba_sorted, color='red', linewidth=3, label='Curva Logística')

# Línea de decisión
ax.axhline(y=0.5, color='green', linestyle='--', linewidth=2, label='Umbral de decisión (0.5)')

ax.set_xlabel(f'{var_name} (estandarizado)', fontsize=12)
ax.set_ylabel('Probabilidad de HeartDisease=1', fontsize=12)
ax.set_title(f'Regresión Logística: {var_name} vs Probabilidad de Enfermedad Cardíaca',
```

```
            fontsize=14, fontweight='bold')
ax.legend(fontsize=10)
ax.grid(True, alpha=0.3)
ax.set_ylim(-0.1, 1.1)
plt.tight_layout()
```

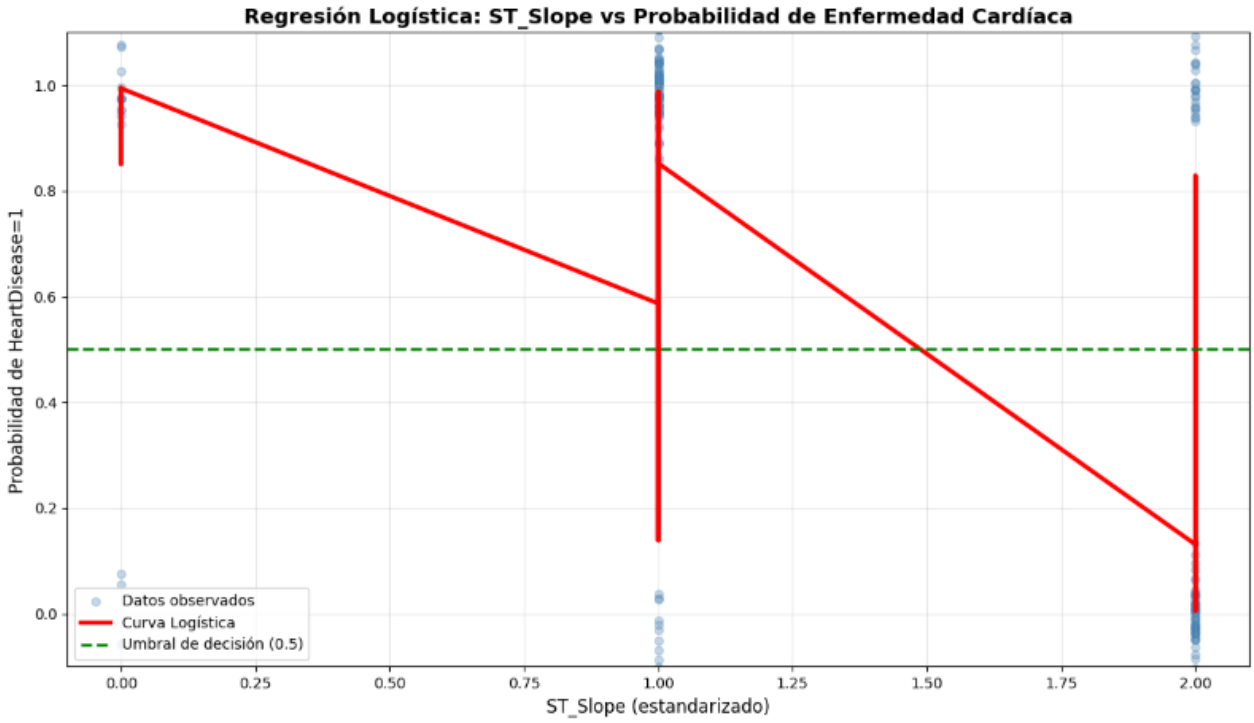

Figura 9: Gráfica logística y diagrama de dispersión

3.5.5. Precisión del Modelo

La precisión (*accuracy*) es una métrica fundamental para evaluar el rendimiento de un modelo de clasificación. Indica el porcentaje de predicciones correctas realizadas por el modelo sobre el total de observaciones, considerando tanto los aciertos en la clase positiva como en la negativa. Matemáticamente se expresa como:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

donde:

- TP: verdaderos positivos
- TN: verdaderos negativos
- FP: falsos positivos
- FN: falsos negativos

A continuación se muestra el código empleado para calcular la precisión y generar un reporte completo de clasificación:

```
    accuracy = accuracy_score(y_test, y_pred)
print(f"\nPrecisión (Accuracy): {accuracy:.4f} ({accuracy*100:.2f}%)")

print("""
INTERPRETACI N:
La precisión (accuracy) es el porcentaje de predicciones correctas sobre
    el total.

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Valores de referencia:
  > 0.90: EXCELENTE
  0.80 - 0.90: MUY BUENO
  0.70 - 0.80: BUENO
  0.60 - 0.70: ACEPTABLE
  < 0.60: POBRE

IMPORTANTE: En datasets desbalanceados, la accuracy puede ser engañosa.
Siempre revisar precision, recall y F1-score.
""")

if accuracy >= 0.90:
    print(f"    Con {accuracy:.2%}, tu modelo tiene precisión EXCELENTE")
elif accuracy >= 0.80:
    print(f"    Con {accuracy:.2%}, tu modelo tiene precisión MUY BUENA")
elif accuracy >= 0.70:
    print(f"    Con {accuracy:.2%}, tu modelo tiene precisión BUENA")
elif accuracy >= 0.60:
    print(f"    Con {accuracy:.2%}, tu modelo tiene precisión ACEPTABLE")
else:
    print(f"    Con {accuracy:.2%}, tu modelo tiene precisión POBRE")

# Reporte completo de clasificación
print("\n--- Reporte Completo de Clasificación ---")
print(classification_report(y_test, y_pred, target_names=['No Enfermedad',
    'Enfermedad']))
```

```
--- Reporte Completo de Clasificación ---
              precision    recall  f1-score   support

No Enfermedad       0.80      0.88      0.84        77
   Enfermedad       0.91      0.84      0.87       107

     accuracy                           0.86       184
    macro avg       0.85      0.86      0.86       184
 weighted avg       0.86      0.86      0.86       184
```

Figura 10: Reporte Completo de Clasificación

Interpretación: El modelo logra clasificar correctamente aproximadamente el 85.87 % de las observaciones, lo que se considera un desempeño **muy bueno** según las referencias comunes ($0.80 \leq$ Accuracy < 0.90).

Dado que la precisión puede ser engañosa en conjuntos de datos desbalanceados, se complementa este análisis con otras métricas como la *precisión por clase (precision)*, la *sensibilidad (recall)* y el *F1-score*, las cuales permiten evaluar de forma más completa el comportamiento del modelo en cada categoría.

3.6. PARTE VI

En esta sección se realiza la selección del modelo de regresión logística reducido, considerando un máximo de 4 variables independientes. El criterio de selección utilizado es el **Pseudo R^2**, que mide la proporción de variabilidad explicada por el modelo.

Dado que el número de variables es relativamente grande, se implementa un enfoque de **forward selection** para agregar iterativamente la variable que más incrementa el Pseudo R^2 en cada paso. Este método es computacionalmente más eficiente que la búsqueda exhaustiva de todas las combinaciones posibles.

El procedimiento consiste en:

1. Comenzar con ninguna variable seleccionada.
2. En cada paso, agregar la variable que más incrementa el Pseudo R^2.
3. Repetir hasta completar 4 variables.

A continuación se presenta el bloque de código utilizado para este análisis.

```
    print("""
ESTRATEGIA:
Probaremos todas las combinaciones posibles de 4 variables y seleccionaremos
la combinacion que maximice el Pseudo R_dos.

NOTA: Con muchas variables, esto puede ser computacionalmente costoso.
Usaremos un enfoque iterativo (forward selection) como alternativa eficiente.
""")


# Si tienes muchas variables, esto puede tardar. Limitar a forward selection
if len(feature_names) > 10:
    print("\n Numero grande de variables detectado.")
    print("   Usando FORWARD SELECTION en lugar de búsqueda exhaustiva.")

    # Forward Selection
    selected_vars = []
    remaining_vars = feature_names.copy()
    best_r2_history = []

    for step in range(4):
```

```
        best_r2 = -np.inf
        best_var = None

        for var in remaining_vars:
            test_vars = selected_vars + [var]
            X_temp = sm.add_constant(X_train[test_vars])
            try:
                modelo_temp = sm.Logit(y_train, X_temp).fit(disp=0)
                if modelo_temp.prsquared > best_r2:
                    best_r2 = modelo_temp.prsquared
                    best_var = var
            except:
                continue

        if best_var:
            selected_vars.append(best_var)
            remaining_vars.remove(best_var)
            best_r2_history.append(best_r2)
            print(f"\nPaso {step+1}: Agregada '{best_var}' Pseudo R_dos =
                {best_r2:.4f}")

    top_4_r2_vars = selected_vars

else:
    # Búsqueda exhaustiva (solo si pocas variables)
    print("\nBuscando la mejor combinación de 4 variables...")
    best_r2 = -np.inf
    best_combination = None

    for combo in combinations(feature_names, 4):
        X_temp = sm.add_constant(X_train[list(combo)])
        try:
            modelo_temp = sm.Logit(y_train, X_temp).fit(disp=0)
            if modelo_temp.prsquared > best_r2:
                best_r2 = modelo_temp.prsquared
                best_combination = combo
        except:
            continue

    top_4_r2_vars = list(best_combination)

print(f"\n--- Mejor Combinacion de 4 Variables (Pseudo R_dos) ---")
for i, var in enumerate(top_4_r2_vars, 1):
    print(f"  {i}. {var}")
```

Los resultados de la selección muestran que las cuatro variables que mejor explican la presencia de enfermedad cardíaca según el criterio de Pseudo R^2 son:

```
    --- Mejor Combinacion de 4 Variables (Pseudo R_dos) ---
  1. ST_Slope
  2. ExerciseAngina
  3. Sex
  4. ChestPainType
```

3.6.1. Entrenamiento del modelo final con estas 4 variables

```
    X_train_r2 = sm.add_constant(X_train[top_4_r2_vars])
X_test_r2 = sm.add_constant(X_test[top_4_r2_vars])
modelo_final_r2 = sm.Logit(y_train, X_train_r2).fit(disp=0)

print(modelo_final_r2.summary())

print(f"\n--- Comparación Final ---")
print(f"\nModelo Completo ({len(feature_names)} variables):")
print(f"  Pseudo R_dos: {pseudo_r2:.4f}")
print(f"  AIC: {modelo_statsmodels.aic:.4f}")
print(f"  BIC: {modelo_statsmodels.bic:.4f}")

print(f"\nModelo con 4 vars (criterio p-value):")
print(f"  Pseudo R_dosR_dosR_dos: {modelo_reducido.prsquared:.4f}")
print(f"  AIC: {modelo_reducido.aic:.4f}")
print(f"  BIC: {modelo_reducido.bic:.4f}")
print(f"  Variables: {top_4_vars}")

print(f"\nModelo con 4 vars (criterio Pseudo R_dosR_dos):")
print(f"  Pseudo R_dos: {modelo_final_r2.prsquared:.4f}")
print(f"  AIC: {modelo_final_r2.aic:.4f}")
print(f"  BIC: {modelo_final_r2.bic:.4f}")
print(f"  Variables: {top_4_r2_vars}")

print("""
CRITERIOS DE SELECCION:

1. Pseudo R_dos: Mayor es mejor (más varianza explicada)
2. AIC (Akaike Information Criterion): Menor es mejor
3. BIC (Bayesian Information Criterion): Menor es mejor

AIC y BIC penalizan modelos con más parámetros. Son útiles para evitar
   overfitting.
BIC penaliza más fuertemente que AIC.

RECOMENDACION:
- Si buscas INTERPRETABILIDAD: modelo con 4 variables
- Si buscas MAXIMA PRECISION: modelo completo (si no hay overfitting)
- Usa BIC si tienes muchos datos y quieres un modelo parsimonioso
""")

# Determinar mejor modelo
if modelo_final_r2.bic < modelo_reducido.bic and modelo_final_r2.bic <
   modelo_statsmodels.bic:
    print("\n MEJOR MODELO: 4 variables seleccionadas por Pseudo R_dos (
       menor BIC)")
elif modelo_reducido.bic < modelo_statsmodels.bic:
    print("\n MEJOR MODELO: 4 variables seleccionadas por p-value (menor
       BIC)")
else:
    print("\n MEJOR MODELO: Modelo completo (menor BIC)")
```

Los resultados del modelo de la regresión logística es:

Logit Regression Results

Dep. Variable:	HeartDisease	No. Observations:	734
Model:	Logit	Df Residuals:	729
Method:	MLE	Df Model:	4
Date:	Tue, 21 Oct 2025	Pseudo R-squ.:	0.4272
Time:	05:01:39	Log-Likelihood:	-289.62
converged:	True	LL-Null:	-505.62
Covariance Type:	nonrobust	LLR p-value:	3.384e-92

	coef	std err	z	P>|z|	[0.025	0.975]
const	1.9466	0.385	5.057	0.000	1.192	2.701
ST_Slope	-2.1776	0.206	-10.585	0.000	-2.581	-1.774
ExerciseAngina	1.4802	0.232	6.392	0.000	1.026	1.934
Sex	1.5483	0.271	5.721	0.000	1.018	2.079
ChestPainType	-0.6133	0.114	-5.395	0.000	-0.836	-0.390

Figura 11: Resultados de la regresión

```
    --- Comparación Final ---

Modelo Completo (11 variables):
  Pseudo  R_doS: 0.4732
  AIC: 556.7672
  BIC: 611.9493

Modelo con 4 vars (criterio p-value):
  Pseudo  R_doS: 0.4272
  AIC: 589.2337
  BIC: 612.2263
  Variables: ['ST_Slope', 'Sex', 'ChestPainType', 'ExerciseAngina']

Modelo con 4 vars (criterio Pseudo  R_do):
  Pseudo  R_doS: 0.4272
  AIC: 589.2337
  BIC: 612.2263
  Variables: ['ST_Slope', 'ExerciseAngina', 'Sex', 'ChestPainType']

CRITERIOS DE SELECCION:

1. Pseudo R_dos: Mayor es mejor (más varianza explicada)
2. AIC (Akaike Information Criterion): Menor es mejor
3. BIC (Bayesian Information Criterion): Menor es mejor

AIC y BIC penalizan modelos con más parámetros. Son útiles para evitar
   overfitting.
BIC penaliza más fuertemente que AIC.

RECOMENDACION:
```

```
- Si buscas INTERPRETABILIDAD: modelo con 4 variables
- Si buscas MAXIMA PRECISION: modelo completo (si no hay overfitting)
- Usa BIC si tienes muchos datos y quieres un modelo parsimonioso


 MEJOR MODELO: Modelo completo (menor BIC)
```

4. CONCLUSIÓN

El presente trabajo permitió aplicar la **regresión logística** para modelar una variable binaria a partir de múltiples variables numéricas y categóricas. Mediante la interpretación de los coeficientes, errores estándar, valores z y p-values, se identificaron las variables independientes que ejercen mayor influencia sobre la probabilidad de presentar enfermedad cardíaca, destacando especialmente **ST_Slope**, **Sex** y **ExerciseAngina** como las más significativas.

La construcción de la matriz de confusión y la evaluación del modelo mediante métricas como **pseudo R-cuadrada**, **Log-Likelihood** y **LLR p-value** permitió comprobar que el modelo presenta un buen ajuste y una precisión (*accuracy*) del 85.87 %, considerada **muy buena**. Asimismo, la selección de modelos optimizados con un máximo de cuatro variables independientes evidenció la importancia de elegir las variables más explicativas, logrando modelos más eficientes sin sacrificar la capacidad predictiva.

Este ejercicio resalta cómo los datasets disponibles en plataformas como Kaggle[1, 3] pueden ser utilizados de manera práctica para la formación de modelos predictivos confiables, proporcionando información valiosa para la toma de decisiones basada en datos y reforzando la comprensión de técnicas estadísticas avanzadas aplicadas en problemas reales de salud.

Referencias

[1] Kaggle. (n.d.). *Heart Disease UCI dataset*. https://www.kaggle.com/ronitf/heart-disease-uci

[2] Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd ed.). Wiley.

[3] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R* (2nd ed.). Springer.

[4] Kuhn, M., & Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. CRC Press.

[5] McFadden, D. (1974). Conditional logit analysis of qualitative choice behavior. In P. Zarembka (Ed.), *Frontiers in Econometrics* (pp. 105–142). Academic Press.