

2025

Predicción de miopía con Python

Fundamentos para la Ciencia de Datos

Miranda, Brian

Contenido

1-Tratamiento de datos.....	3
2- Modelo de regresión logística	4
3- Ajuste de modelo	6
3.1- Alternativa 1 con RFE.....	6
3.2- Alternativa 2- Sin RFE.....	7
3.2.1-Genero todas las combinaciones posibles:.....	7
3.2.2- Selección hacia atrás:	7
3.3- Comparo métodos	8
4- Comparación y evolución de modelos	8
4.1- ANOVA.....	9
4.2- ODDS RATIO	10
4.3- MULTICOLINEALIDAD Y DISNTANCIA DE COOK	12
4.3.1- Modelo completo	13
4.3.2- Alternativa reducida de 4 variables	14
4.3.3- Modelo de 3 variables predictoras (utilizado en R)	15
4.4- AIC.....	15
4.5- Curva ROC y AUC	16
4.5.1- Modelo de 4 variables (spheq, sporthr, vcd, ma).....	16
4.5.2- Modelo de 3 variables (spheq, sporthr, ma).....	17
5- NAIVE BAYES	18
5.1- Modelo de 3 variables	18
5.2- Pumbral optimo segun modelos	20
5.3- Comparación de modelos según Pumbrals	22
5.4- Elección del mejor modelo de Bayes	23
5.4.1- Según F1-score	23
5.4.2- Según ROC y AUC	23
5.5- Matriz de confusión	24
6- Modelo de regresión logístico - Parte 2.....	25

1-Tratamiento de datos

Se importa y se elimina muestras con datos faltantes (NaN).

```
# Conversión de variables categóricas (como "Si" y "No") a valores numéricos
datos0['ma'] = datos0['ma'].map({'Si': 1, 'No': 0})
datos0['mio'] = datos0['mio'].map({'Si': 1, 'No': 0})

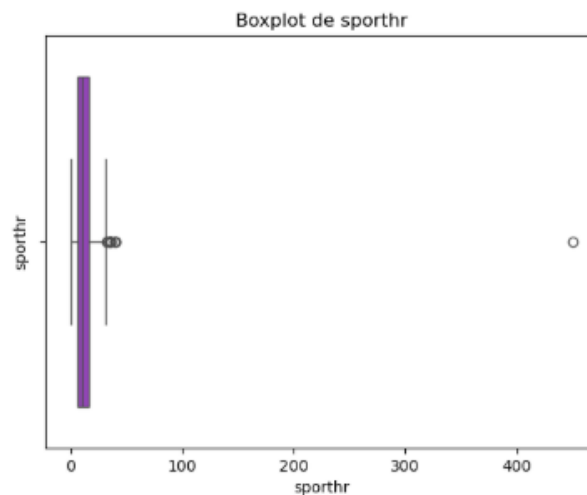
# Verificación de datos faltantes-conteo de los valores nulos para cada columna
print(datos0.isnull().sum())

# Eliminar filas con valores NaN
datos = datos0.dropna()

# Verificación de tipos de datos después de la conversión
print(datos.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 618 entries, 0 to 617
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  -
0    id          618 non-null    int64
1    mio          618 non-null    int64
2    spheq        618 non-null    float64
3    al           618 non-null    float64
4    acd          618 non-null    float64
5    vcd          618 non-null    float64
6    spothr       618 non-null    int64
7    tvhr         618 non-null    int64
8    ma           618 non-null    int64
```

Se analizan las variables numéricas y se observa un outlier para spothr que se elimina, al igual que la columna id.



Se obtiene la siguiente distribución de variables:

```

: datos = datos.drop(258)

: datos=datos.drop(columns=['id'])
print(datos.describe()) # se aprecia que se elimina el maxima

```

	mio	spheq	al	acd	vcd	spothr
count	617.000000	617.000000	617.000000	617.000000	617.000000	617.000000
mean	0.131280	0.801853	22.495802	3.578903	15.375559	11.833063
std	0.337981	0.626075	0.680259	0.230480	0.664028	7.780867
min	0.000000	-0.699000	19.900000	2.772000	13.380000	0.000000
25%	0.000000	0.460000	22.040000	3.424000	14.930000	6.000000
50%	0.000000	0.730000	22.460000	3.586000	15.360000	10.000000
75%	0.000000	1.034000	22.970000	3.730000	15.840000	16.000000
max	1.000000	4.372000	24.560000	4.250000	17.300000	40.000000

	tvhr	ma
count	617.000000	617.000000
mean	8.936791	0.505673
std	5.716593	0.500373
min	0.000000	0.000000
25%	4.000000	0.000000
50%	8.000000	1.000000
75%	12.000000	1.000000
max	31.000000	1.000000

Defino modelo para regresión logística binomial, donde las variables independientes son numéricas, excepto ma y la variable dependiente a predecir, mio, que son factores con 0= no es miope y 1= es miope. Se considera una partición de 70 % para entrenamiento y 30 % para prueba del modelo y se usa semilla 666.

Al parecer R y Python NO consideran las mismas muestras bajo el mismo número de semilla (random state), por lo que varían los resultados obtenidos.

```

Número de muestras de entrenamiento: 431
Número de muestras de prueba: 186
mio
0    164
1     22
Name: count, dtype: int64
mio
0    372
1     59
Name: count, dtype: int64

```

Observar que las muestras de 0 y 1 para mio esta desbalanceado, por lo que se deberá ajustar la Pumbal que definirá si mio es 0 o 1 (por defecto valor 0,5), como se efectúa más adelante.

2- MODELO DE REGRESIÓN LOGÍSTICA

Para el modelo completo y uno reducido con las variables más significativas sin considerar un Pumbal óptimo.

```

import statsmodels.api as sm
#ofrece una implementación de regresión logística binomial
import pandas as pd

# Función para obtener los valores estadísticos del modelo
def obtener_resumen_estadistico(X_train, y_train):
    # Añadir una constante para el intercepto (como en R)
    X_train_sm = sm.add_constant(X_train) # Esto añadirá

    # Ajustar el modelo de regresión logística
    modelo = sm.Logit(y_train, X_train_sm)
    resultado = modelo.fit()

    # Resumen del modelo (similar al summary en R)
    return resultado.summary()
print(y_train.value_counts())
# Obtener el resumen estadístico del modelo completo (usan
resumen = obtener_resumen_estadistico(X_train, y_train)
print(X)#VERIFICO VARIABLES EN JUEGO-parece algo simple pe
#de cuales eran las variables predictoras que incluía X
# Imprimir el resumen completo
print(resumen)

```

```

=====
Logit Regression Results
=====
Dep. Variable:          mio    No. Observations:          431
Model:                Logit    Df Residuals:              423
Method:               MLE      Df Model:                7
Date:                Tue, 22 Jul 2025    Pseudo R-squ.:          0.3900
Time:                18:17:28    Log-Likelihood:         -104.98
converged:            True      LL-Null:              -172.09
Covariance Type:      nonrobust    LLR p-value:           8.219e-26
=====
               coef    std err          z      P>|z|      [0.025    0.975]
-----
const         12.9310     9.168      1.410     0.158     -5.039     30.900
spheq         -4.5461     0.582     -7.817     0.000     -5.686    -3.406
al            -1.0544     1.436     -0.734     0.463     -3.868     1.759
acd           1.6314     1.411     1.156     0.248     -1.134     4.397
vcd           0.3804     1.367     0.278     0.781     -2.298     3.059
spothr        -0.0625     0.026     -2.376     0.017     -0.114     -0.011
tvhr          -0.0273     0.032     -0.853     0.393     -0.090     0.035
ma             0.7582     0.382     1.986     0.047     0.010     1.506
=====

```

Se observa que las variables más significativas son spheq, spothr y ma, las mismas utilizadas en código R (alternativa 3 - manual) y se considera un modelo solo con estas variables obteniéndose:

```

=====
Logit Regression Results
=====
Dep. Variable:          mio    No. Observations:          431
Model:                Logit    Df Residuals:              427
Method:               MLE      Df Model:                3
Date:                Tue, 22 Jul 2025    Pseudo R-squ.:          0.3700
Time:                17:28:07    Log-Likelihood:         -108.41
converged:            True      LL-Null:              -172.09
Covariance Type:      nonrobust    LLR p-value:           2.007e-27
=====
               coef    std err          z      P>|z|      [0.025    0.975]
-----
const         0.6035     0.464      1.301     0.193     -0.306     1.513
spheq         -4.3146     0.553     -7.804     0.000     -5.398    -3.231
spothr        -0.0634     0.025     -2.498     0.013     -0.113     -0.014
ma             0.7025     0.371     1.893     0.058     -0.025     1.430
=====

```

3- Ajuste de modelo

3.1- Alternativa 1 con RFE

RFE selecciona las mejores características de manera automática y recursiva en función de su impacto en el rendimiento del modelo:

- Reducción de la dimensionalidad: Al eliminar las características menos importantes, reduces la complejidad del modelo.
- Mejor interpretabilidad: Menos variables hacen que el modelo sea más fácil de interpretar.
- Evitar sobreajuste: Un modelo con muchas variables irrelevantes tiende a sobreajustarse (overfitting), lo que reduce su capacidad para generalizar.

Se consideran las 5 mejores variables predictoras.

```
# Modelo con selección hacia atrás (backward elimination)
# Se utiliza RFE para selección de características
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

# Inicializar el modelo de regresión logística
modelo_completo_sklearn = LogisticRegression(max_iter=1000)

# Aplicar RFE para seleccionar características
selector = RFE(modelo_completo_sklearn, n_features_to_select=5)
selector = selector.fit(X_train, y_train)

# Mostrar las características seleccionadas
print("Características seleccionadas por RFE:")
print(X.columns[selector.support_])

Características seleccionadas por RFE:
Index(['spheq', 'al', 'acd', 'vcd', 'ma'], dtype='object')
```

Se obtienen las mismas variables significativas utilizadas en el modelo manual definido anteriormente excepto por sporthr.

Respecto al modelo automático realizado en R, se obtienen las mismas variables excepto vcd, que es reemplazada por sporthr.

Si bien utilizamos la misma semilla (666) R y Python lo consideran diferente.

3.2- Alternativa 2- Sin RFE

3.2.1-Genero todas las combinaciones posibles:

```
# Iterar sobre todas las combinaciones posibles de 1 a todas las variables
for i in range(1, len(variables) + 1): # Recorrer desde 1 hasta el número total
    # Generar todas las combinaciones de variables de tamaño i
    for comb in itertools.combinations(variables, i): #itertools.comb... toma i e
        #/al haber un for repite luego tomando de a 2 variables difreentes y asi
        comb_list = list(comb)
        #Si no uso list(comb), simplemente estaría trabajando con una tupla ( lis
        aic = obtener_aic(X_train, y_train, comb_list) #calculo AIC solo con cier
        precision = obtener_precision(X_train, y_train, X_test, y_test, comb_list)

    # Almacenar los resultados
    resultados_aic.append(aic)
    resultados_precision.append(precision)
    num_variables.append(len(comb_list))
    combinaciones.append(comb_list) # Almacenar la combinación de variables
```

Se obtienen los siguientes resultados:

```
Mejor modelo basado en AIC:
Num_Variables          4
AIC                    222.602342
Precisión              0.870968
Combinación            [spheq, vcd, sporthr, ma]
Name: 80, dtype: object
```

```
Variables del mejor modelo basado en AIC:
['spheq', 'vcd', 'sporthr', 'ma']
```

```
Mejor modelo basado en precisión:
Num_Variables          2
AIC                    229.897533
Precisión              0.887097
Combinación            [spheq, vcd]
Name: 9, dtype: object
```

```
Variables del mejor modelo basado en precisión:
['spheq', 'vcd']
```

3.2.2- Selección hacia atrás:

Se inicia con todas las variables predictoras, se calcula su AIC y se eliminan variables que mejoren el AIC, reduciéndolo hasta alcanzar el menor AIC posible.

```
Index(['spheq', 'al', 'acd', 'vcd', 'sporthr', 'tvhr', 'ma'], dtype='object')
Optimization terminated successfully.
Current function value: 0.252871
Iterations 8
Optimization terminated successfully.
Current function value: 0.249158
Iterations 8
Variables seleccionadas: ['spheq', 'al', 'acd', 'sporthr', 'ma']
AIC final: 222.8354994454113
Precisión del modelo final: 0.8548387096774194
```

3.3- Comparo métodos

```
--Considerando todas las variante:--
Num_Variables      4
AIC                 222.602342
Precisión          0.870968
Combinación        [spheq, vcd, sporthr, ma]
Name: 80, dtype: object

--Selección hacia atrás:--
Variables seleccionadas: ['spheq', 'al', 'acd', 'sporthr', 'ma']
AIC final: 222.8354994454113
Precisión del modelo final: 0.8548387096774194
```

Ambas contienen las 3 variables que consideramos en el modelo manual realizado en R (las más significativas, $p < 0,05$).





La alternativa automática con step realizada en R también concluye a las mismas 5 variables de selección hacia atrás

Se aprecia que AIC y precisión son similares.

Recordemos que AIC penaliza la complejidad del modelo y ayuda a seleccionar el modelo más adecuado entre varios modelos posibles, mientras que ANOVA no penaliza la complejidad, pero da información sobre la significancia de las variables. ¡Y ambos se calculan sobre las muestras de entrenamiento!!

4- Comparación y evolución de modelos

En primera instancia se analiza la partición de train del dataset bajo la semilla 666 analizando:

Paso	¿Usar train?	¿Por qué?
1. ANOVA	 Sí	Estás explorando relaciones entre predictores y la variable objetivo para seleccionar variables.
2. ODDS RATIO	 Sí	Se calculan con los coeficientes del modelo ajustado solo sobre train. No debes mirar test todavía.
3. Multicolinealidad y Cook	 Sí	Analizás la estructura interna del modelo y detectás problemas en train. Aún no evaluás rendimiento.
4. AIC	 Sí	Es un criterio de ajuste interno del modelo . Sirve para comparar modelos sobre los datos de entrenamiento .

Luego se utiliza la parte de test para obtener predicciones y calcular métricas como AUC, precisión, sensibilidad, etc. No se ajusta ningún modelo nuevo: simplemente aplicás el modelo ya entrenado.

4.1- ANOVA

En modelos lineales, ANOVA compara modelos mediante la Suma de cuadrados de la varianza explicada → ¿Cuánto mejora el modelo al agregar variables?

En modelos logísticos (donde no hay varianza en el sentido clásico), se usa Devianza o log-verosimilitud → ¿Cuánto mejora la capacidad predictiva del modelo (log-likelihood) al agregar variables?

Uso muestras de train.

```
X_train_sm = sm.add_constant(X_train)

# Ajustar el modelo de regresión logística binomial (regresión logística)
modelo_completo_logit = sm.Logit(y_train, X_train_sm).fit()

# Mostrar el resumen del modelo completo: coeficientes, p-valores, etc.
print("Resumen del Modelo Completo de Regresión Logística:")
print(modelo_completo_logit.summary())

# Ajustar el modelo completo utilizando Logit con fórmula
formula_completa = 'mio ~ spheq + al + acd + vcd + sporthr + tvhr + ma'
modelo_completo_logit_ols = logit(formula_completa, data=pd.concat([y_train, X_train], axis=1)).fit()

# Ajustar el modelo reducido (con un subconjunto de 4 variables seleccionadas)
modelo_reducido_logit = logit('mio ~ spheq + vcd + sporthr + ma', data=pd.concat([y_train, X_train], axis=1)).fit()

# Ajustar otro modelo reducido (con 3 variables seleccionadas)
modelo_reducido2_logit = logit('mio ~ spheq + sporthr + ma', data=pd.concat([y_train, X_train], axis=1)).fit()

# Comparación entre el modelo completo y el modelo reducido usando la razón de verosimilitud (likelihood ratio test)
dev_completo_logit = modelo_completo_logit.llf # Log-likelihood del modelo completo
dev_reducido_logit = modelo_reducido_logit.llf # Log-likelihood del modelo reducido
dev_reducido2_logit = modelo_reducido2_logit.llf # Log-likelihood del modelo reducido 2

# Mostrar Los Log-Likelihoods para Los tres modelos
print(f'\nLog-likelihood del Modelo Completo: {dev_completo_logit}')
print(f'Log-likelihood del Modelo Reducido: {dev_reducido_logit}')
print(f'Log-likelihood del Modelo Reducido 2: {dev_reducido2_logit}')
```

Respecto a log-verosimilitud (log-likelihood) más grande o menor devianza, indica mejor ajuste, siendo:

$$\text{Devianza} = -2 \times \text{Log-Likelihood}$$

Recordemos que el Pvalor es la probabilidad de obtener los datos observados (o más extremos), si la hipótesis nula “El modelo reducido se ajusta tan bien como el modelo completo” fuera cierta.

Entonces:

- Si el valor p es muy pequeño (por ejemplo, 0.01), hay muy poca probabilidad de ver esos datos si H_0 fuera cierta, así que la rechazamos.
- Si el valor p es grande (por ejemplo, 0.23), entonces es perfectamente razonable ver esos datos si H_0 fuera cierta, y no hay evidencia para rechazarla.

Log-likelihood del Modelo Completo: -104.97630544265996
Log-likelihood del Modelo Reducido: -106.30117087915347
Log-likelihood del Modelo Reducido 2: -108.41054249154536

Comparación entre Modelo Completo y Modelo Reducido:
Log-likelihood del Modelo Completo: -104.97630544265996
Log-likelihood del Modelo Reducido: -106.30117087915347
Diferencia de Log-likelihoods: 2.6497308729870213
Grados de Libertad de la Diferencia: 3
P-valor de la prueba de chi-cuadrado: 0.44883770800257794

Comparación entre Modelo Completo y Modelo Reducido 2:
Log-likelihood del Modelo Completo: -104.97630544265996
Log-likelihood del Modelo Reducido 2: -108.41054249154536
Diferencia de Log-likelihoods: 6.868474097770786
Grados de Libertad de la Diferencia: 4
P-valor de la prueba de chi-cuadrado: 0.1430041715475906

COMPARAR

- El Modelo 2 tiene un *Log-Likelihood* más alto (menos negativo) y un Pseudo R^2 ligeramente mayor (0.3874 vs 0.3823), lo que indica que ajusta un poco mejor los datos.
- el Modelo 2 ajusta un poco mejor globalmente, introduce una variable (acd) que no es significativa (su), lo cual puede no justificar su inclusión.

4.2- ODDS RATIO

Siendo que la regresión logística binaria modela la $\text{logit}(p)$, con $\text{odds} = p / (1 - p)$, la razón de probabilidades se tiene que:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots$$

Si se exponencia a ambos lados se tienen los odds ratios (OR). Por eso, los coeficientes se interpretan en términos de cuánto se multiplican las odds por cada unidad adicional de la variable, y en nuestro caso consideramos que se eleva en una unidad una determinada variable.

Para el modelo de 4 variables obtenido se tiene las siguientes métricas y ODDS correspondientes:

```

# Supongamos que df es tu DataFrame con Las variables
X = datos[["spheq", "vcd", "sporthr", "ma"]] # Variables del modelo reducido
y = datos['mio'] # 'mio' variable binaria (0 o 1)

# Dividir Los datos en conjunto de entrenamiento y conjunto de prueba
random_state = 666
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=random_state)

# Agregar constante a Las variables independientes (intercepto)
X_train_sm = sm.add_constant(X_train)

# Ajustar el modelo de regresión Logística (modelo Logit)
modelo_logit = sm.Logit(y_train, X_train_sm).fit()

# Ver el resumen del modelo
print("Resumen del Modelo Logístico:")
print(modelo_logit.summary())

# Calculo el Odds Ratio
# Los coeficientes estimados del modelo Logit se transforman a Odds Ratios aplicando La función exponencial
odds_ratios = np.exp(modelo_logit.params)

# Odds Ratios para Las variables del modelo
print("\nOdds Ratios para el Modelo de 4 Variables:")
print(odds_ratios)

```

```

Resumen del Modelo Logístico:
Logit Regression Results
=====
Dep. Variable:      mio    No. Observations:      431
Model:              Logit    Df Residuals:          426
Method:              MLE    Df Model:              4
Date:               Wed, 23 Jul 2025    Pseudo R-squ.:      0.3823
Time:               06:57:08    Log-Likelihood:     -106.30
converged:          True    LL-Null:           -172.09
Covariance Type:    nonrobust    LLR p-value:        1.792e-27
=====

```

	coef	std err	z	P> z	[0.025	0.975]
const	9.4518	4.441	2.128	0.033	0.747	18.156
spheq	-4.4639	0.566	-7.890	0.000	-5.573	-3.355
vcd	-0.5729	0.286	-2.005	0.045	-1.133	-0.013
sporthr	-0.0616	0.026	-2.391	0.017	-0.112	-0.011
ma	0.7693	0.378	2.036	0.042	0.029	1.510

```

=====
Odds Ratios para el Modelo de 4 Variables:
const      12731.408163
spheq       0.011517
vcd         0.563886
sporthr     0.940285
ma          2.158229
dtype: float64
Distribución de clases en el conjunto total:
mio
0    0.869
1    0.131
Name: proportion, dtype: float64

```

Y para el modelo de 3 variables se obtiene:

```

Logit Regression Results
=====
Dep. Variable:          mio      No. Observations:          431
Model:                  Logit      Df Residuals:            427
Method:                  MLE      Df Model:                3
Date:                   Wed, 23 Jul 2025      Pseudo R-squ.:          0.3700
Time:                   07:03:38      Log-Likelihood:         -108.41
converged:              True      LL-Null:                -172.09
Covariance Type:        nonrobust      LLR p-value:            2.007e-27
=====
               coef      std err          z      P>|z|      [0.025      0.975]
-----
const          0.6035          0.464          1.301      0.193      -0.306          1.513
spheq         -4.3146          0.553         -7.804      0.000      -5.398         -3.231
sporthr       -0.0634          0.025         -2.498      0.013      -0.113         -0.014
ma              0.7025          0.371          1.893      0.058      -0.025          1.430
=====

Odds Ratios para el Modelo de 4 Variables:
const          1.828522
spheq          0.013373
sporthr        0.938544
ma             2.018821
dtype: float64

```

Ambos modelos alcanzan OR similares, a continuación se detalla para el último caso una interpretación.

El coeficiente de ma sugiere que cuando la variable ma está presente, es decir $ma = 1$ (con miopía materna), se duplica las Odds de que el individuo sea miope, comparado con quienes no tienen madre miope.

Dado que un OR menor a 1 implica menores odds del evento, por cada aumento unitario de spheq, el odds de que sea miope se reduce en un factor de 0,013, una fuerte asociación negativa.

4.3- MULTICOLINEALIDAD Y DISTANCIA DE COOK

La multicolinealidad afecta la precisión y estabilidad de los coeficientes del modelo y es detectada mediante el VIF o la matriz de correlación.

Por otra parte, la distancia de Cook detecta observaciones influyentes que pueden tener un gran impacto en el modelo (outlier), y está más relacionada con la influencia individual de cada observación en lugar de las relaciones entre variables. Mide cuanto cambia el modelo completo al eliminar esa observación.

Consideraciones:

VIF: Si alguna variable tiene un VIF superior a 10, indica que esa variable está altamente correlacionada con las otras y podría estar afectando la estabilidad del modelo. Si es mayor a 5 prestar atención.

Distancia de Cook: Si alguna observación tiene una distancia de Cook mayor a 1 o 4/número muestras (para más de 100 muestras), esa observación está teniendo un gran impacto en los resultados del modelo, podrían ser valores atípicos (outlier) o errores de datos.

4.3.1- Modelo completo

Considerando todas las muestras se obtienen los siguientes valores:

```
#Cálculo inicial considerando todas las variables iniciales - NO es necesario definir como modelo (regresión logística binomial) - NO DEPENDE DEL MODELO.
import pandas as pd
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Definir las variables independientes
X = datos[["spheq", "al", "acd", "vcd", "sporthr", "tvhr", "ma"]] # Variables independientes

# 1. Matriz de Correlación
correlation_matrix = X.corr()
print("Matriz de Correlación:")
print(correlation_matrix)

# 2. Calcular el VIF para cada variable
# Agregar una constante para la regresión
X_const = sm.add_constant(X)

# Calcular el VIF para cada variable
vif = pd.DataFrame()
vif['Variable'] = X_const.columns
vif['VIF'] = [variance_inflation_factor(X_const.values, i) for i in range(X_const.shape[1])]

print("VIF para cada variable:")
print(vif)
```

```
Matriz de Correlación:
           spheq      al      acd      vcd      sporthr      tvhr      ma
spheq    1.000000 -0.304681 -0.240010 -0.245995 -0.013886 -0.079413 -0.130706
al        -0.304681  1.000000  0.457841  0.941867  0.096623  0.075537  0.045593
acd        -0.240010  0.457841  1.000000  0.201092  0.078938 -0.041365  0.117289
vcd        -0.245995  0.941867  0.201092  1.000000  0.078780  0.080492  0.008186
sporthr    -0.013886  0.096623  0.078938  0.078780  1.000000  0.167319 -0.082106
tvhr       -0.079413  0.075537 -0.041365  0.080492  0.167319  1.000000  0.007787
ma         -0.130706  0.045593  0.117289  0.008186 -0.082106  0.007787  1.000000

VIF para cada variable:
  Variable      VIF
0    const  2663.225863
1    spheq    1.141119
2      al    26.930260
3      acd    3.160308
4      vcd   21.953118
5  sporthr    1.049529
6      tvhr    1.051127
7      ma     1.036296
```

De la matriz de correlación se busca correlaciones fuertes (mayores a 0,9 o menores a -0,9) entre las variables independientes que pueden estar causando multicolinealidad.

Notar que ya tengo mis modelos reducidos y el modelo de 4 variables contiene a vcd de VIF=21,95 pero para definir el problema se debe repetir al análisis anterior con las muestras de train y con cada modelo reducido y no con el total de muestras.

Se repite el análisis considerando las muestras de train para el modelo completo.

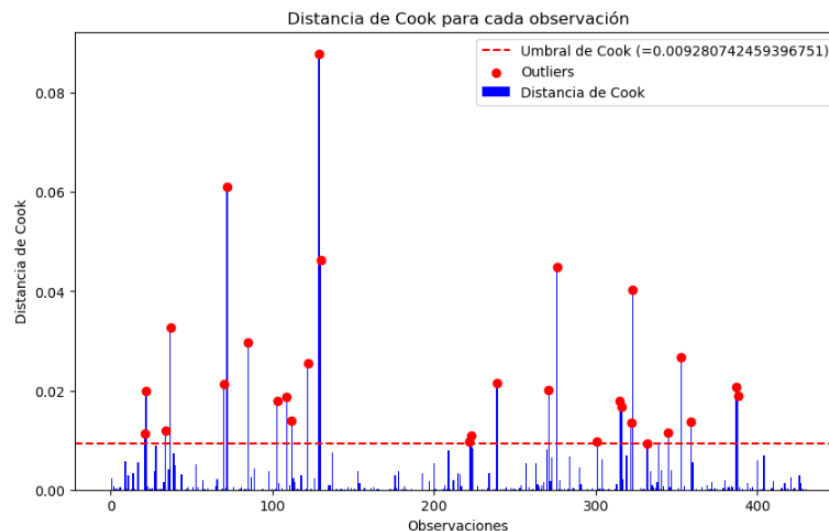
```
VIF para cada variable (entrenamiento):
Variable      VIF
0      const  2529.115001
1      spheq   1.154946
2         al  28.667254
3         acd   3.276910
4         vcd  23.436072
5      spothr   1.060459
6         tvhr   1.065887
7         ma    1.033893
influence:
<statsmodels.stats.outliers_influence.MLEInfluence object at 0x00000127173AF080>
Observaciones influyentes (distancia de Cook > umbral):
[0.01862471 0.01040607 0.01418864 0.0118486  0.03147801 0.01758098
 0.05186289 0.02613447 0.01016242 0.01387137 0.01240157 0.02477276
 0.05931108 0.05844168 0.01215789 0.01130926 0.01668481 0.03063918
 0.01476836 0.01731272 0.03760782 0.01443152 0.013154  0.02808666
 0.01282232 0.01420027 0.0277565  0.02905982 0.01132283 0.02135918
 0.01278203]
```

Se aprecia una leve diferencia entre los resultados obtenidos.

4.3.2- Alternativa reducida de 4 variables

Se repite el análisis considerando spheq, spothr, vcd y ma.

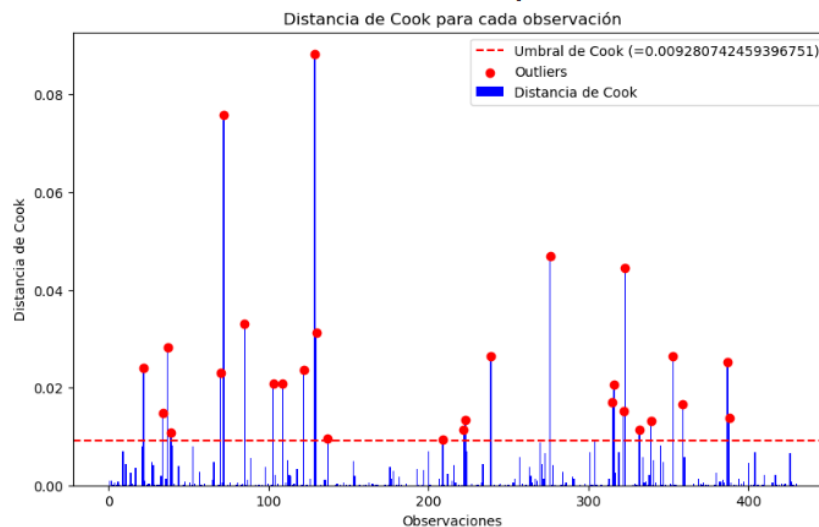
```
Matriz de Correlación (entrenamiento):
           spheq  spothr  vcd  ma
spheq  1.000000 -0.077577 -0.251440 -0.139422
spothr -0.077577 1.000000  0.117866 -0.063410
vcd    -0.251440  0.117866  1.000000  0.023871
ma     -0.139422 -0.063410  0.023871  1.000000
VIF para cada variable (entrenamiento):
Variable      VIF
0      const  574.615753
1      spheq   1.092081
2      spothr   1.022274
3         vcd   1.078714
4         ma    1.025638
Observaciones influyentes (distancia de Cook > umbral):
[0.01123819 0.01991481 0.01193188 0.03268279 0.02126853 0.06102444
 0.02972207 0.01796726 0.01879001 0.01384134 0.02544561 0.08788319
 0.04630912 0.0096802  0.01084598 0.02142648 0.02009173 0.04481493
 0.00973317 0.01792314 0.01664697 0.01342863 0.04038004 0.00936926
 0.01162865 0.026753  0.01366174 0.02075965 0.01896935]
```



4.3.3- Modelo de 3 variables predictoras (utilizado en R)

Se consideran spheq, sporthr y ma en la misma muestra de entrenamiento, manteniéndose el umbral de 4/nro de muestras.

```
Matriz de Correlación (entrenamiento):
      spheq  sporthr  ma
spheq  1.000000 -0.077577 -0.139422
sporthr -0.077577  1.000000 -0.063410
ma      -0.139422 -0.063410  1.000000
VIF para cada variable (entrenamiento):
Variable  VIF
0  const  7.027484
1  spheq  1.027682
2  sporthr 1.011774
3  ma      1.025621
Observaciones influyentes (distancia de Cook > umbral):
[0.02400587 0.01485871 0.02834967 0.01069001 0.0230508 0.07585376
 0.03302724 0.02080011 0.0209147 0.02373231 0.08835127 0.03135515
 0.00955899 0.00931728 0.01142232 0.01349065 0.02641461 0.0469489
 0.01697531 0.02056198 0.01523664 0.0444999 0.01144072 0.01323311
 0.02644094 0.0167063 0.02523549 0.01389091]
```



Se aprecia prácticamente los mismos valores de distancia de Cook para ambos modelos, aunque en el reducido de 3 variables se halla un outlier menos.

Los modelos reducidos presentan sus variables independientes sin problemas de correlación entre ellas (elementos de matriz correspondiente) y VIF menores a 5.

4.4- AIC

A partir de los cálculos realizados y variables predictoras del modelo, incluyendo la constante, se tiene los siguientes valores para cada uno de ellos.

```

import numpy as np

# Función para calcular AIC
def calcular_AIC(log_verosimilitud, k):
    AIC = 2 * k - 2 * log_verosimilitud
    return AIC

# Datos
# Devianzas de los modelos
log_verosimilitud_completo = -104.976
log_verosimilitud_reducido_1 = -106.301
log_verosimilitud_reducido_2 = -108.411

# Número de parámetros en cada modelo
# k = cantidad de variables + 1 intercepto

# Calcular AIC para cada modelo
AIC_completo = calcular_AIC(log_verosimilitud_completo, 8)
AIC_reducido_1 = calcular_AIC(log_verosimilitud_reducido_1, 5)
AIC_reducido_2 = calcular_AIC(log_verosimilitud_reducido_2, 4)

AIC del Modelo Completo: 225.952
AIC del Modelo Reducido 1: 222.602
AIC del Modelo Reducido 2: 224.822

```

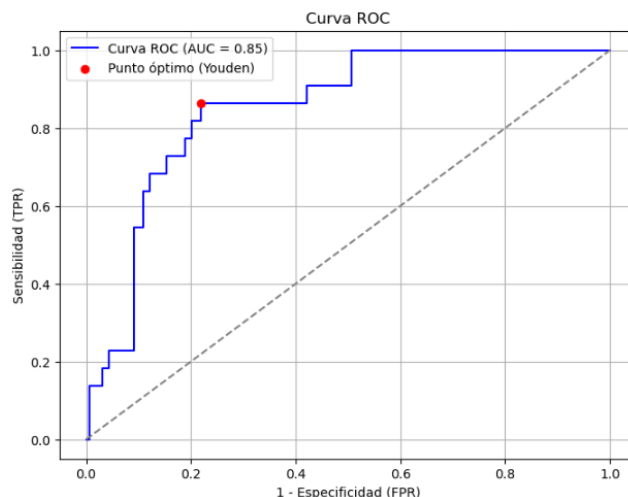
El modelo con menor AIC (mejor) es el reducido 1 (4 variables predictoras) con mejor balance entre ajuste y complejidad respecto al reducido 2. Se opta por el modelo reducido 1, ya que tiene una penalización mínima en términos de ajuste (AIC) y es más simple que el modelo completo. El modelo reducido 2, con un AIC levemente más alto, podría no ser tan adecuado debido a mayor pérdida de variables en principio.

4.5- Curva ROC y AUC

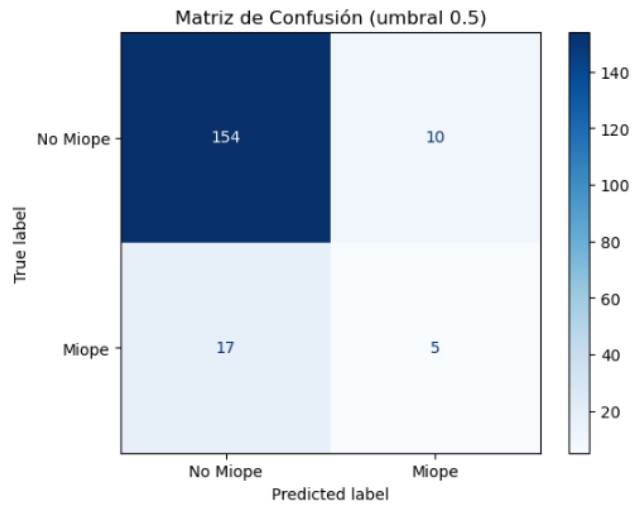
Se utiliza las muestras de test y se sigue considerando la p umbral de **0,5** para definir si mio es 0 o 1. Mas adelante se evalúa un Pumbtral optimo.

4.5.1- Modelo de 4 variables (spheq, sporthr, vcd, ma)

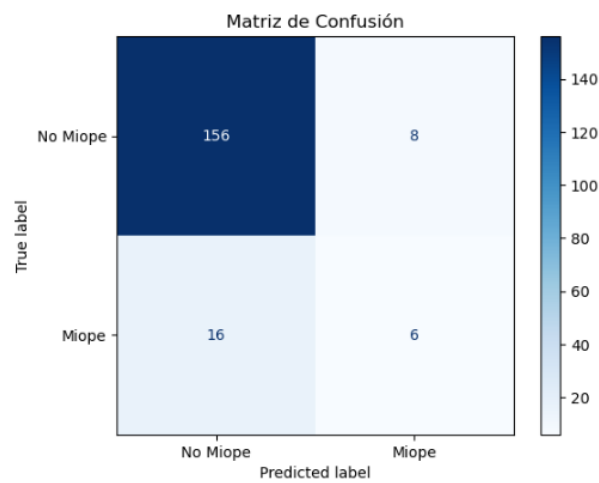
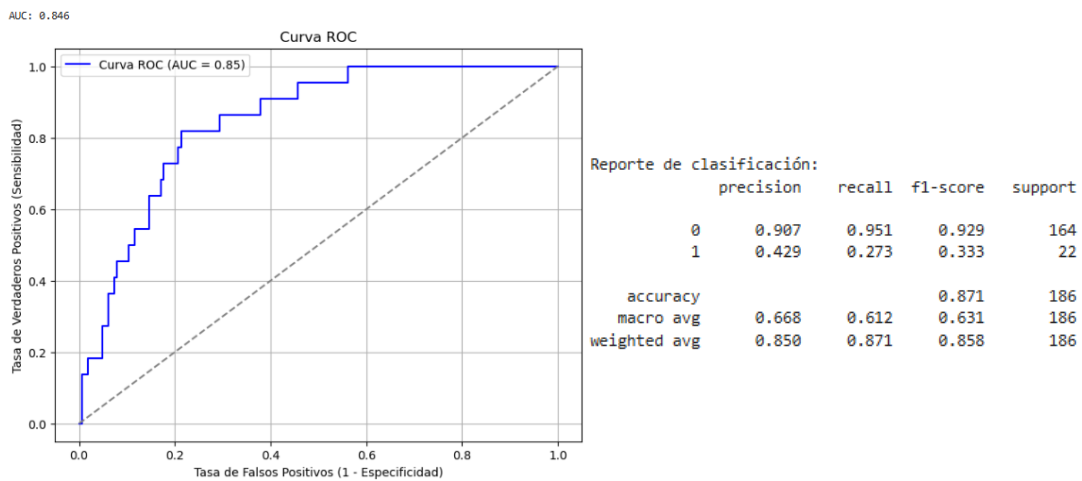
AUC: 0.851
Umbral óptimo según índice de Youden: 0.166



Reporte de clasificación:				
	precision	recall	f1-score	support
0	0.901	0.939	0.919	164
1	0.333	0.227	0.270	22
accuracy			0.855	186
macro avg	0.617	0.583	0.595	186
weighted avg	0.833	0.855	0.843	186



4.5.2- Modelo de 3 variables (spheq, sporthr, ma)



¿Será que el modelo aun con los problemas observados presenta un ajuste aceptable?

En la sección 5 se aprecia la importancia del Pumbrol que define si mio es 0 o 1 al ser menor o mayor a 0,5 (por defecto) respectivamente en un modelo de Bayes, y en la sección 6 se reconsidera este modelo de regresión logística variando la Pumbrol y logrando una mejor configuración de la matriz de confusión y mejores métricas.

5- NAIVE BAYES

Se considera un modelo de Bayes para predecir la miopía y en principio se consideran las mismas variables significativas obtenidas anteriormente según el modelo de regresión logística binomial. Luego se busca las variables más significativas que

5.1- Modelo de 3 variables

Inicialmente se define modelo considerando las mismas 3 variables utilizadas en el modelo de regresión logística binomial. Luego se busca las variables más significativas para este modelo de Bayes.

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
# Defino variables independientes y dependientes
# Creo el modelo de Naive Bayes (con Laplace=1)
# GaussianNB no tiene directamente un parámetro Laplace, pero se puede usar el parámetro 'var_smoothing' para regularizar el modelo.
print(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=666) # Semilla igual al anterior
mod1 = GaussianNB(var_smoothing=1.0) # Smoothing similar a Laplace
mod1.fit(X_train, y_train)

# Ver el modelo ajustado
print(mod1)

print(X_train.shape) #tamaño de muestra train
print(X_test.shape) #tamaño de muestra test
print(y_test)#veo que hay muestras con 0 y 1

print("----")
print("balance de clases y_test")
# Ver el balance de clases en y_test
print(y_test.value_counts())
print("porcentaje de cada clase en y_test")
# Ver el porcentaje de cada clase en y_test
print(y_test.value_counts(normalize=True))

# Predicciones de probabilidades (tipo "raw")-me da probabilidad de que sea 0 o 1
proba_1 = mod1.predict_proba(X_test)
print("Probabilidades predichas:")
print(proba_1[:5]) #[:5] para ver solo las primeras 5 predicciones, sino vere todas

# Predicciones de clase (tipo "class")-me dice de una si es 0 o 1
predi_1 = mod1.predict(X_test)
print("Clases predichas:")#veo todas
print(predi_1)
```

[illegible]

Se observa que todas las predicciones son mio=0 para el Pvalor predeterminado de clasificación de 0,5, que decide si la clase es 0 o 1, debiendo ajustarlo para que el modelo sea más sensible a la clase minoritaria al predecir.

A priori se considera un P umbral en función de la cantidad de muestras, resultando un $p=22/186=0,12$, por lo que se opta por un umbral de 0,15.

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
print(X)
# Creo el modelo de Naive Bayes (con Laplace=1)
mod1 = GaussianNB(var_smoothing=1.0) # Smoothing similar a Laplace

# Ajusto el modelo
mod1.fit(X_train, y_train)

# modelo ajustado
print(mod1)#instancia de GaussianNB.

# tamaño de las muestras de entrenamiento y prueba
print(X_train.shape) # Tamaño de muestra train
print(X_test.shape) # Tamaño de muestra test

# clases en y_test (para verificar que hay clases 0 y 1)
print(y_test)

# balance de clases en y_test
print(y_test.value_counts())

# porcentaje de cada clase en y_test
print(y_test.value_counts(normalize=True))

# Predicciones de probabilidades (tipo "raw") - me da la probabilidad de que sea 0 o 1
proba_1 = mod1.predict_proba(X_test)
print("Probabilidades predichas:")
print(proba_1[:10]) # Solo mostrar las primeras 10 probabilidades

# Ajusto el umbral para la clase 1 (decisión con probabilidad mayor que 0.3 en lugar de 0.5)
umbral = 0.15 # Ajustar el umbral aquí

# Predicciones de clase usando el nuevo umbral
predicciones_ajustadas = (proba_1[:, 1] > umbral).astype(int) # predice 1 si la probabilidad de la clase 1 es mayor que el umbral
print("Clases predichas con umbral ajustado:")
print(predicciones_ajustadas) # Muestra todas las clases predichas ajustadas
```

```

      sphes  sporth  ma
0    -0.052    40    1
1     0.608     4    1
2     1.179    14    0
3     0.525    18    0
4     0.697    14    1
...      ...      ...
613  0.678     2    1
614  0.665     6    1
615  1.834     8    1
616  0.665    12    0
617  0.802    25    1

[617 rows x 3 columns]
GaussianNB(var_smoothing=1.0)
(431, 3)
(186, 3)
134 1
580 0
96 0
187 0
190 0
...
239 0
409 0
314 0
313 1
192 1
Name: mio, Length: 186, dtype: int64
mio
0    164
1     22
Name: count, dtype: int64
mio
0    0.88172
1    0.11828
Name: proportion, dtype: float64
...

Name: proportion, dtype: float64
Probabilidades predichas:
[[0.87663039 0.12336961]
 [0.85965629 0.14034371]
 [0.88768885 0.11231115]
 [0.89889575 0.10110425]
 [0.86988491 0.13011509]
 [0.8570048  0.1429952 ]
 [0.85770686 0.14229314]
 [0.8370058  0.1629942 ]
 [0.87420093 0.12579907]
 [0.8502813  0.1497187 ]]

Clases predichas con umbral ajustado:
[[0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1
 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0
 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0
 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 1 0
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1
 0]]

```

A partir de las muestras de test se calcula la matriz de confusión y métricas correspondiente.

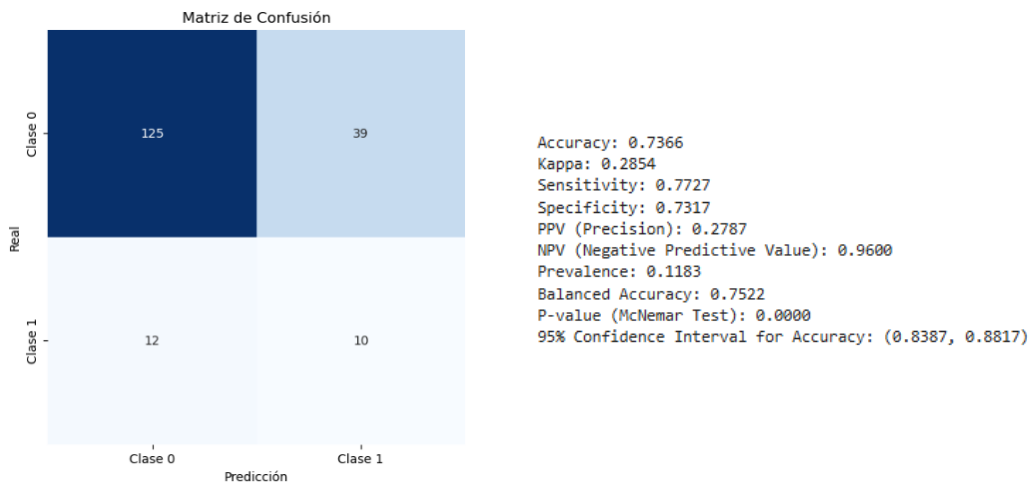
```

# Genero la matriz de confusión
cm = confusion_matrix(y_test, predicciones_ajustadas)

# Visualizo la matriz de confusión
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['Clase 0', 'Clase 1'], yticklabels=['Clase 0', 'Clase 1'])
#cm:matriz de datos a mostrar;fmt=d: formato de nros (d=enteros);annot=true da valores dentro de heatmap;cmap paleta de colores;
#cbar=False: no muestra barra lateral de colores; xtick.. etiqueta para columnas (eje x), típico de clases predichas;simil con ytick..
plt.xlabel('Predicción')
plt.ylabel('Real')
plt.title('Matriz de Confusión')
plt.show()

# Imprimir la matriz de confusión
print("Matriz de Confusión:")
print(cm)

```

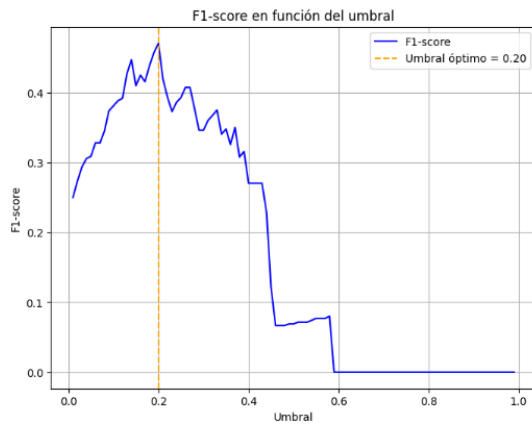


Recordar que el test tiene 22 muestras miopes (mio=1) y para la fila de la matriz de Real clase 1 alcanzo esa misma cantidad; compruebo que no hay error en la disposición de elementos de la matriz de confusión 😊.

5.2- Puntualidad óptima según modelos

Modelo de 3 variables:

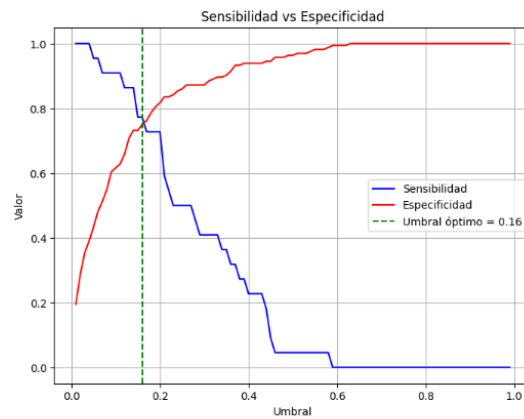
- Según F1score:



[F1-score] Umbral óptimo = 0.20 | Valor = 0.4786

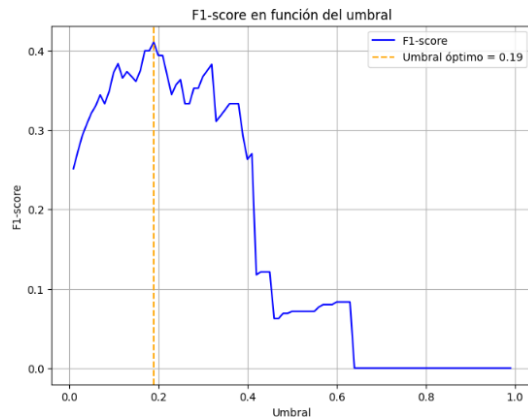


[Youden] Umbral óptimo = 0.14 | Valor = 0.5953

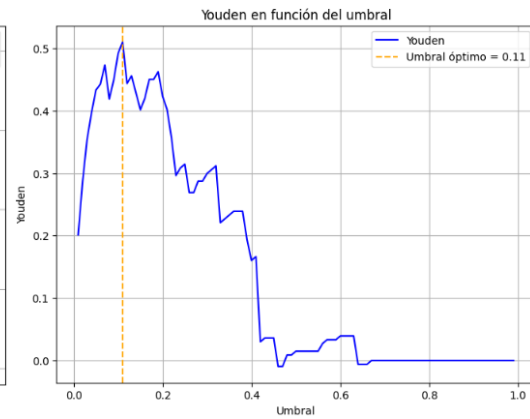


[Equilibrio] Umbral óptimo = 0.16 | Sensibilidad = 0.7727 | Especificidad = 0.7580

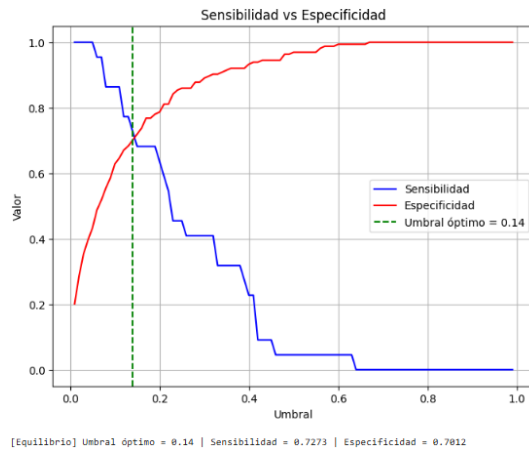
Modelo de 4 variables:



[F1-score] Umbral óptimo = 0.19 | Valor = 0.4118

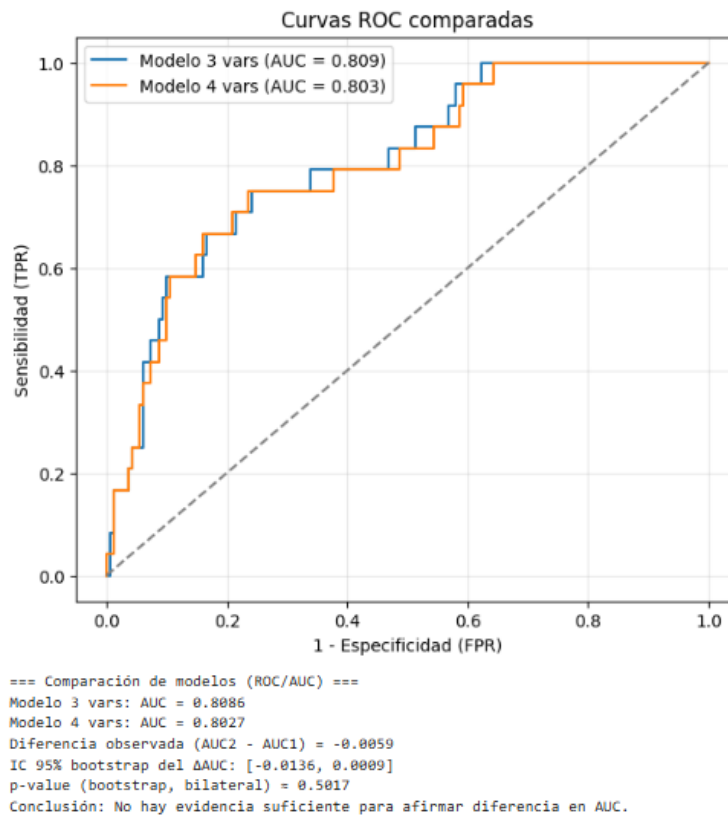


[Youden] Umbral óptimo = 0.11 | Valor = 0.5188

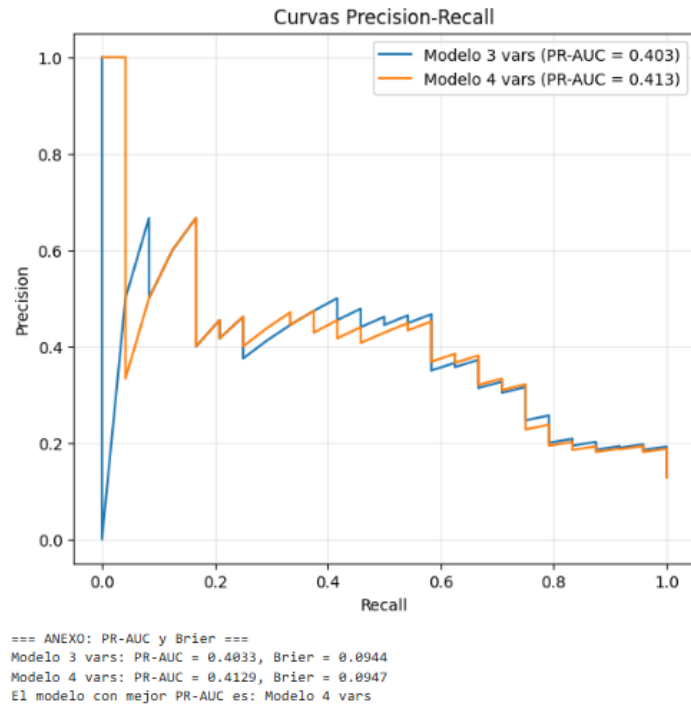


5.3- Comparación de modelos según Pumbrales

Mejor modelo:



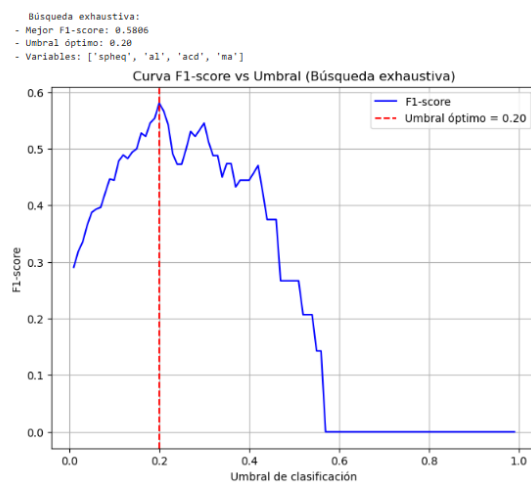
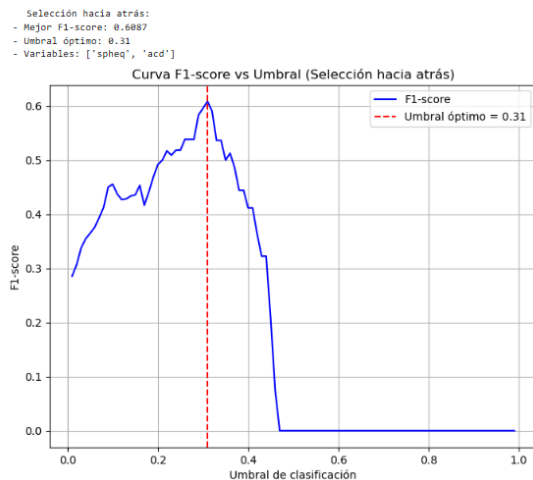
Además:



5.4- Elección del mejor modelo de Bayes

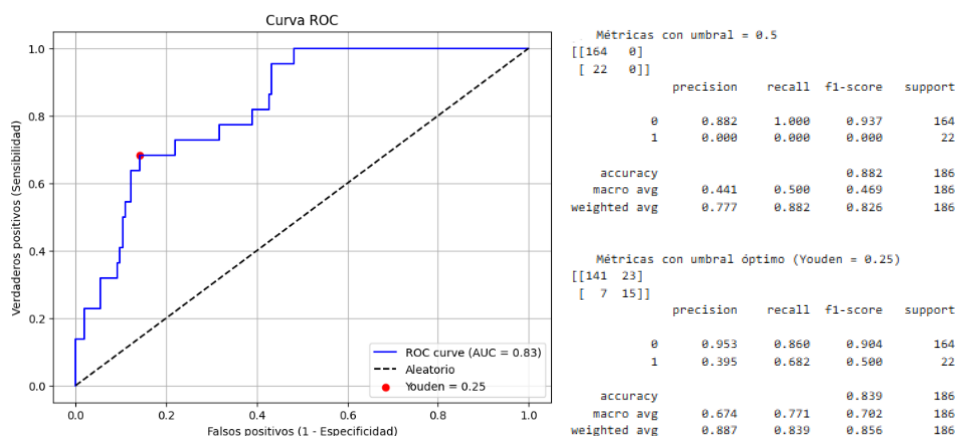
5.4.1- Según F1-score

Se considera los métodos de selección hacia atrás (BACKWARD SELECTION) y de búsqueda exhaustiva (BEST SUBSET), resultando el método de selección hacia atrás como mejor alternativa con mayor F1-score.

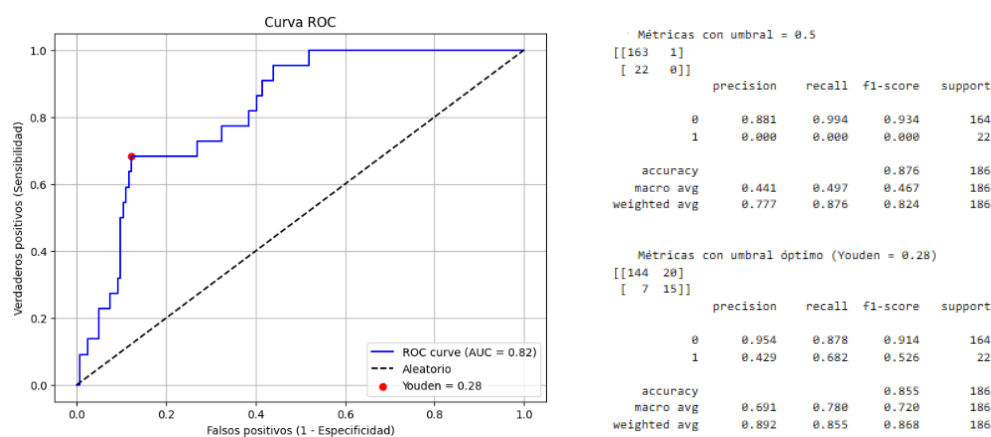


5.4.2- Según ROC y AUC

Se consideran las variables predictoras definidas con el método de F1-score, resultando para el modelo de dos variables obtenido por selección hacia atrás:



Para el modelo de 4 variables obtenidos por el metodo exhaustivo se tiene los siguientes resultados:

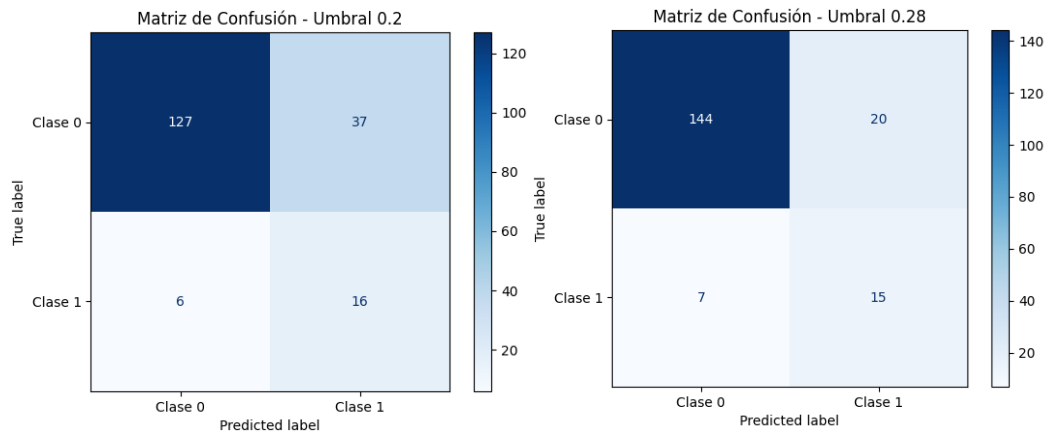


Se aprecia que el modelo de 4 variables presenta un AUC similar al de 2 variables predictoras, pero ofrece un mejor balance, con mejor accuracy, F1-score, macro avg F1 y Weighted avg F1.

Solo las variables spheq y ma coinciden con las variables del modelo obtenido anteriormente, demostrando que no siempre un modelo que contemple solo variables significativas alcance mejores métricas.

5.5- Matriz de confusión

Se comparan las matrices de confusión según los métodos anteriores para el modelo de 4 variables.



Las métricas resultantes para cada una de ellas es:

	Umbral	Accuracy	Precision	Recall (Sensibilidad)	Especificidad	F1	Balanced Acc.	AUC	Youden	TP	FP	FN	TN
Modelo_1 (0.20)	0.20	0.768817	0.301887	0.727273	0.774390	0.426667	0.750831	0.821785	0.501663	16	37	6	127
Modelo_2 (0.25)	0.25	0.822581	0.365854	0.681818	0.841463	0.476190	0.761641	0.823171	0.523282	15	26	7	138

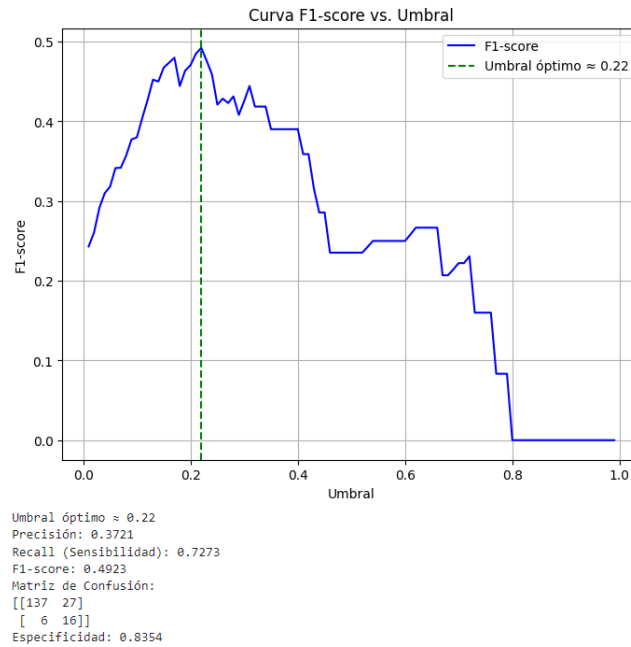
Conclusión:

Si el objetivo es un modelo más equilibrado (según F1 y Youden), el modelo 2 (umbral 0,25) es mejor, aunque si el objetivo es maximizar la detección de positivos (recall), el modelo 1 (umbral 0,20) es mejor modelo.

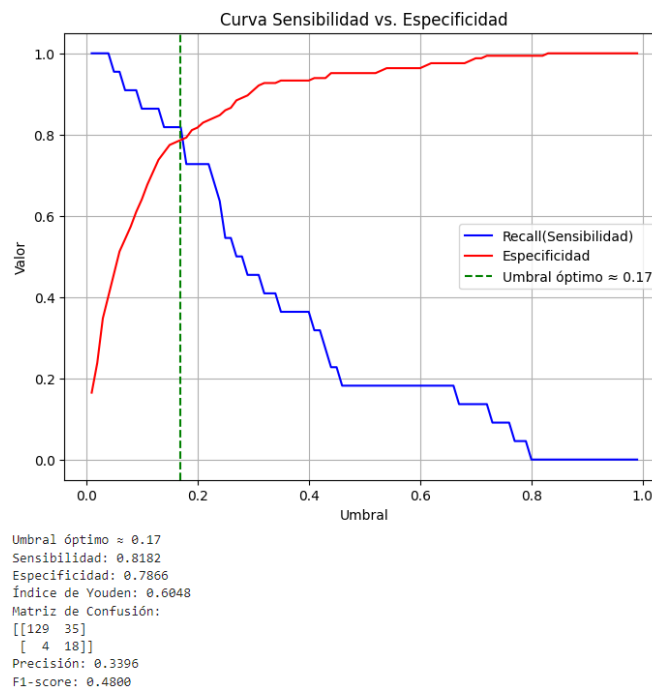
6- Modelo de regresión logístico - Parte 2

Se considera las mismas 4 variables spheq, sporthr, ma y vcd para el modelo logístico, pero se varia la Pumbral a fin de maximizar el F1-score y alcanzar un equilibrio entre las curvas de sensibilidad y especificidad para el modelo de ajuste.

Según el método F1-score se tiene (modelo 1):



Y para equilibrio entre Sensibilidad y especificidad (modelo 2) se tiene:



El modelo 1 destaca por una mayor especificidad, cometiendo menores errores al clasificar negativos como positivos; un leve mayor F1-score, que indica un mejor balance general entre recall (sensibilidad) y precisión; y una leve mejor precisión respecto al modelo 2. Por su parte el modelo2 alcanza una mayor sensibilidad (recall), identificando mejor los casos positivos.

Si se desea identificar mayor cantidad de casos positivos (recall), el Modelo 2 sería preferible, pero si el objetivo es evitar los falsos positivos (especificidad) y obtener un mejor balance entre precisión y recall, el Modelo 1 es la mejor alternativa.

Conclusión

Tras comparar las métricas y matrices de confusión de los mejores modelos de Bayes con los de regresión logística se concluye que el modelo de regresión logística con umbral de Youden de 0,28 es la mejor alternativa en términos de alta sensibilidad y balance general, por su alto F1-score (equilibrio entre sensibilidad y precisión), balance de precisión y recall e índice de Youden superior al de los modelos de Bayes (equilibrio entre sensibilidad y especificidad).

En caso de priorizar únicamente el balance entre precisión y recall el modelo 1 es la mejor opción por presentar un F1-score mayor.

Y si se desea evitar los falsos positivos el modelo de Bayes con Pumbal 0,25 es la mejor opción al presentar un mayor nivel de especificidad.