proyecto-final-m4

February 5, 2025

[51]:
```
# IMPORTACION DE LIBRERIAS
import numpy as np
import pandas as pd

# LIBRERIAS PARA HACER GRAFICOS
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as xp
from sklearn.svm import SVC
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

[2]:
```
# 2 - CARGA DEL FICHERO DE DATOS
file = '/content/ASI_casoPractico.csv'
data = pd.read_csv(file, sep = ';')
data.head()
```

[2]:

	ID	b	e	LBE	AC	FM	UC	ASTV	MSTV	ALTV	…	Min	Max	Nmax	\
0	1	240	357	120	0	0	0	73	0.5	43	…	62	126	2	
1	2	5	632	132	4	0	4	17	2.1	0	…	68	198	6	
2	3	177	779	133	2	0	5	16	2.1	0	…	68	198	5	
3	4	411	1192	134	2	0	6	16	2.4	0	…	53	170	11	
4	5	533	1147	132	4	0	5	16	2.4	0	…	53	170	9	

	Nzeros	Mode	Mean	Median	Variance	Tendency	Target
0	0	120	137	121	73	1	1
1	1	141	136	140	12	0	0
2	1	141	135	138	13	0	0
3	0	137	134	137	13	1	0
4	0	137	136	138	11	1	0

[5 rows x 26 columns]

1 1. Análisis descriptivo de las variables explicativas y el target.

[3]:
```
# 2 - INFORMACION DEL CONJUNTO DE DATOS
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 26 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   ID        2126 non-null   int64  
 1   b         2126 non-null   int64  
 2   e         2126 non-null   int64  
 3   LBE       2126 non-null   int64  
 4   AC        2126 non-null   int64  
 5   FM        2126 non-null   int64  
 6   UC        2126 non-null   int64  
 7   ASTV      2126 non-null   int64  
 8   MSTV      2126 non-null   float64
 9   ALTV      2126 non-null   int64  
 10  MLTV      2126 non-null   float64
 11  DL        2126 non-null   int64  
 12  DS        2126 non-null   int64  
 13  DP        2126 non-null   int64  
 14  DR        2126 non-null   int64  
 15  Width     2126 non-null   int64  
 16  Min       2126 non-null   int64  
 17  Max       2126 non-null   int64  
 18  Nmax      2126 non-null   int64  
 19  Nzeros    2126 non-null   int64  
 20  Mode      2126 non-null   int64  
 21  Mean      2126 non-null   int64  
 22  Median    2126 non-null   int64  
 23  Variance  2126 non-null   int64  
 24  Tendency  2126 non-null   int64  
 25  Target    2126 non-null   int64  
dtypes: float64(2), int64(24)
memory usage: 432.0 KB
```

[4]:
```
data.dtypes
```

[4]:
```
ID            int64
b             int64
e             int64
LBE           int64
AC            int64
FM            int64
```

```
UC            int64
ASTV          int64
MSTV        float64
ALTV          int64
MLTV        float64
DL            int64
DS            int64
DP            int64
DR            int64
Width         int64
Min           int64
Max           int64
Nmax          int64
Nzeros        int64
Mode          int64
Mean          int64
Median        int64
Variance      int64
Tendency      int64
Target        int64
dtype: object
```

- ID: Identificador único de cada registro.
- b, e, DR: No se especifica su descripción, pero suelen ser valores numéricos reservados para cálculos o identificaciones.
- LBE: Latido Basal Estimado.
- AC: Aceleraciones.
- FM: Movimientos Fetales.
- UC: Contracciones Uterinas.
- ASTV: Tiempo de Variabilidad a Corto Plazo.
- MSTV: Media de la Variabilidad a Corto Plazo.
- ALTV: Tiempo de Variabilidad a Largo Plazo.
- MLTV: Media de la Variabilidad a Largo Plazo.
- DL: No está especificado, podría ser alguna medida de deceleración.
- DS: Desviación Estándar.
- DP: No especificado, podría ser una medida de presión.
- Width: Ancho del vector de tiempo.
- Min: Valor mínimo.
- Max: Valor máximo.
- Nmax: Número de máximos.
- Nzeros: Número de ceros.
- Mode: Moda del ciclo cardíaco.
- Mean: Media del ciclo cardíaco.
- Median: Mediana del ciclo cardíaco.
- Variance: Varianza del ciclo cardíaco.
- Tendency: Tendencia del latido.
- Target: Variable objetivo que indica el estado del feto (normal o anormal).

[5]:
```
# NÚMERO DE VALORES ÚNICOS PARA CADA VARIABLE
print(data.agg(['nunique']).T)
```

```
          nunique
ID           2126
b             979
e            1064
LBE            48
AC             22
FM             96
UC             19
ASTV           75
MSTV           57
ALTV           87
MLTV          249
DL             15
DS              2
DP              5
DR              1
Width         154
Min           109
Max            86
Nmax           18
Nzeros          9
Mode           88
Mean          103
Median         95
Variance      133
Tendency        3
Target          2
```

Clasifique cada variable según su tipo, cualitativa o cuantitativa discreta o continúa. texto en negrita

- Cualitativa: Target (interpretando como una categoría) Aunque es numérica, representa una categoría que indica el estado (normal o anormal). Por eso, se considera cualitativa.
- Cuantitativa Discreta: ID, b, e, LBE, AC, FM, UC, ASTV, ALTV, DL, DS, DP, Width, Min, Max, Nmax, Nzeros, Mode, Median, Tendency
- Cuantitativa Continua: MSTV, MLTV, Mean, Variance

[6]:
```
# MEDIDAS DE CENTRALIZACION, LOCALIZACION Y DISPERION
data.describe().T
```

[6]:
```
          count         mean         std    min      25%     50%      75%  \
ID       2126.0  1063.500000  613.867657    1.0   532.25  1063.5  1594.75
b        2126.0   878.439793  894.084748    0.0    55.00   538.0  1521.00
e        2126.0  1702.877234  930.919143  287.0  1009.00  1241.0  2434.75
LBE      2126.0   133.303857    9.840844  106.0   126.00   133.0   140.00
```

AC	2126.0	2.722484	3.560850	0.0	0.00	1.0	4.00
FM	2126.0	7.241298	37.125309	0.0	0.00	0.0	2.00
UC	2126.0	3.659925	2.847094	0.0	1.00	3.0	5.00
ASTV	2126.0	46.990122	17.192814	12.0	32.00	49.0	61.00
MSTV	2126.0	1.332785	0.883241	0.2	0.70	1.2	1.70
ALTV	2126.0	9.846660	18.396880	0.0	0.00	0.0	11.00
MLTV	2126.0	8.187629	5.628247	0.0	4.60	7.4	10.80
DL	2126.0	1.570085	2.499229	0.0	0.00	0.0	3.00
DS	2126.0	0.003293	0.057300	0.0	0.00	0.0	0.00
DP	2126.0	0.126058	0.464361	0.0	0.00	0.0	0.00
DR	2126.0	0.000000	0.000000	0.0	0.00	0.0	0.00
Width	2126.0	70.445908	38.955693	3.0	37.00	67.5	100.00
Min	2126.0	93.579492	29.560212	50.0	67.00	93.0	120.00
Max	2126.0	164.025400	17.944183	122.0	152.00	162.0	174.00
Nmax	2126.0	4.068203	2.949386	0.0	2.00	3.0	6.00
Nzeros	2126.0	0.323612	0.706059	0.0	0.00	0.0	0.00
Mode	2126.0	137.452023	16.381289	60.0	129.00	139.0	148.00
Mean	2126.0	134.610536	15.593596	73.0	125.00	136.0	145.00
Median	2126.0	138.090310	14.466589	77.0	129.00	139.0	148.00
Variance	2126.0	18.808090	28.977636	0.0	2.00	7.0	24.00
Tendency	2126.0	0.320320	0.610829	-1.0	0.00	0.0	1.00
Target	2126.0	0.221543	0.415383	0.0	0.00	0.0	0.00

	max
ID	2126.0
b	3296.0
e	3599.0
LBE	160.0
AC	26.0
FM	564.0
UC	23.0
ASTV	87.0
MSTV	7.0
ALTV	91.0
MLTV	50.7
DL	16.0
DS	1.0
DP	4.0
DR	0.0
Width	180.0
Min	159.0
Max	238.0
Nmax	18.0
Nzeros	10.0
Mode	187.0
Mean	182.0
Median	186.0

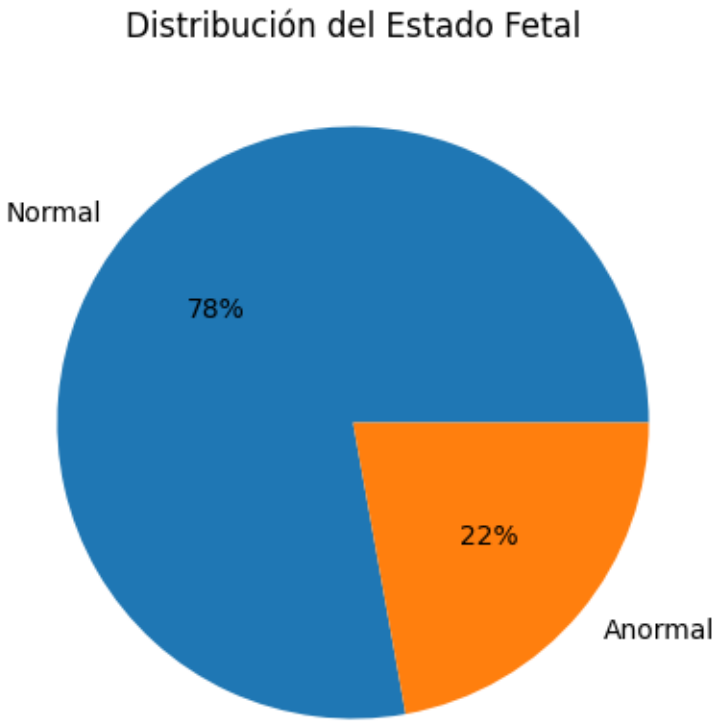
```
Variance   269.0
Tendency     1.0
Target       1.0
```

[7]:
```
# ANÁLISIS DESCRIPTIVO - NULOS
print(data.isnull().sum())
```

```
ID          0
b           0
e           0
LBE         0
AC          0
FM          0
UC          0
ASTV        0
MSTV        0
ALTV        0
MLTV        0
DL          0
DS          0
DP          0
DR          0
Width       0
Min         0
Max         0
Nmax        0
Nzeros      0
Mode        0
Mean        0
Median      0
Variance    0
Tendency    0
Target      0
dtype: int64
```

[8]:
```
# ELIMINAR COLUMNAS NO NECESARIAS
data = data.drop(["ID", "b", "e", "DR"], axis = 1)
```

[9]:
```
# DISTRIBUCIÓN DE FRECUENCIAS DEL TARGET
target_counts = data['Target'].value_counts()
target_labels = ["Normal", "Anormal"]
plt.figure(figsize=(8, 5))
plt.pie(target_counts, labels=target_labels, autopct="%1.0f%%")
plt.title("Distribución del Estado Fetal")
plt.show()
print("Conteo de Target:", target_counts)
```




```
Conteo de Target: Target
0    1655
1     471
Name: count, dtype: int64
```

Proporción de estados fetales normales y anormales La proporción de estados fetales es:

- Normal: 78%
- Anormal: 22%

[15]:
```
# MEDIDAS DE CENTRALIZACIÓN, LOCALIZACIÓN Y DISPERSIÓN
print(data[['FM', 'ALTV', 'Median', 'AC', 'FM', 'UC']].describe().T)
```

```
          count        mean        std   min    25%    50%    75%    max
FM       2126.0    7.241298  37.125309   0.0    0.0    0.0    2.0  564.0
ALTV     2126.0    9.846660  18.396880   0.0    0.0    0.0   11.0   91.0
Median   2126.0  138.090310  14.466589  77.0  129.0  139.0  148.0  186.0
AC       2126.0    2.722484   3.560850   0.0    0.0    1.0    4.0   26.0
FM       2126.0    7.241298  37.125309   0.0    0.0    0.0    2.0  564.0
UC       2126.0    3.659925   2.847094   0.0    1.0    3.0    5.0   23.0
```

Obtenga las medidas de centralización, localización y dispersión para las variables: FM, ALTV y Median.

1. FM

- Media (mean): 7.24
- Desviación estándar (std): 37.13
- Mínimo (min): 0.0
- 25% Cuartil (25%): 0.0
- Mediana (50%): 0.0
- 75% Cuartil (75%): 2.0
- Máximo (max): 564.0

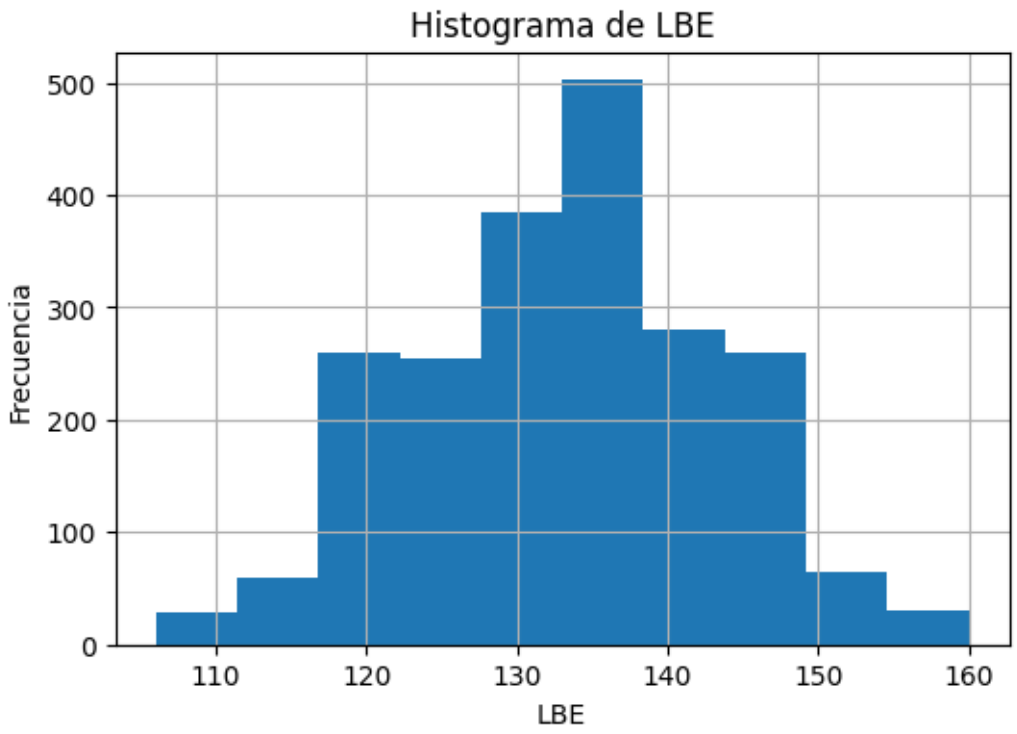
2. ALTV

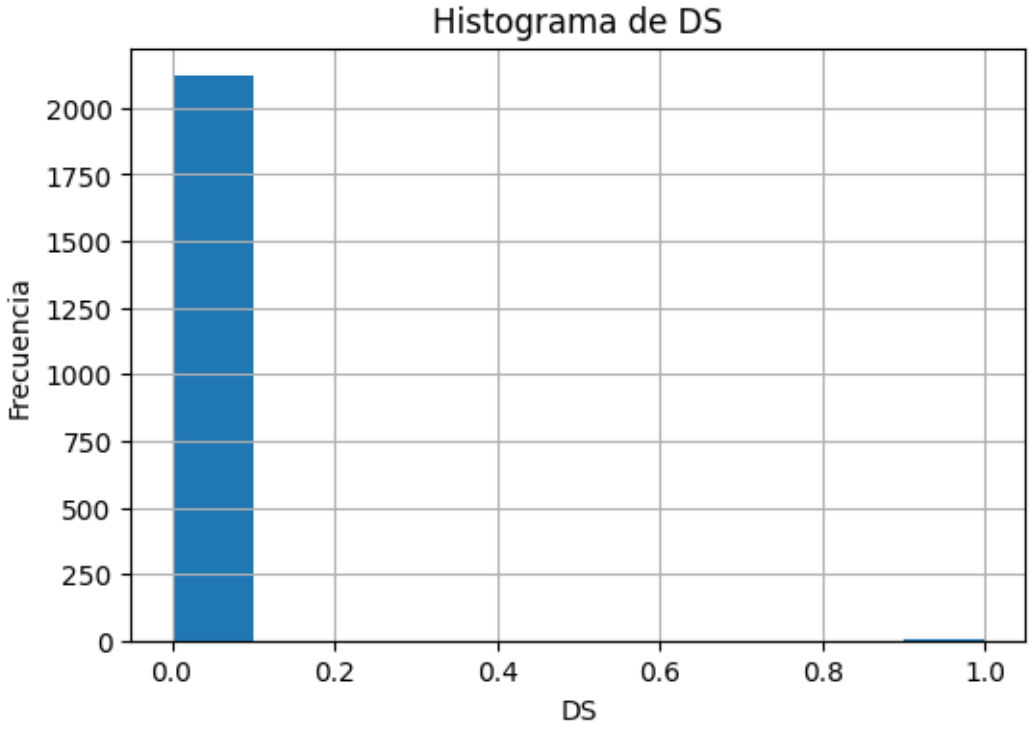
- Media (mean): 9.85
- Desviación estándar (std): 18.40
- Mínimo (min): 0.0
- 25% Cuartil (25%): 0.0
- Mediana (50%): 0.0
- 75% Cuartil (75%): 11.0
- Máximo (max): 91.0

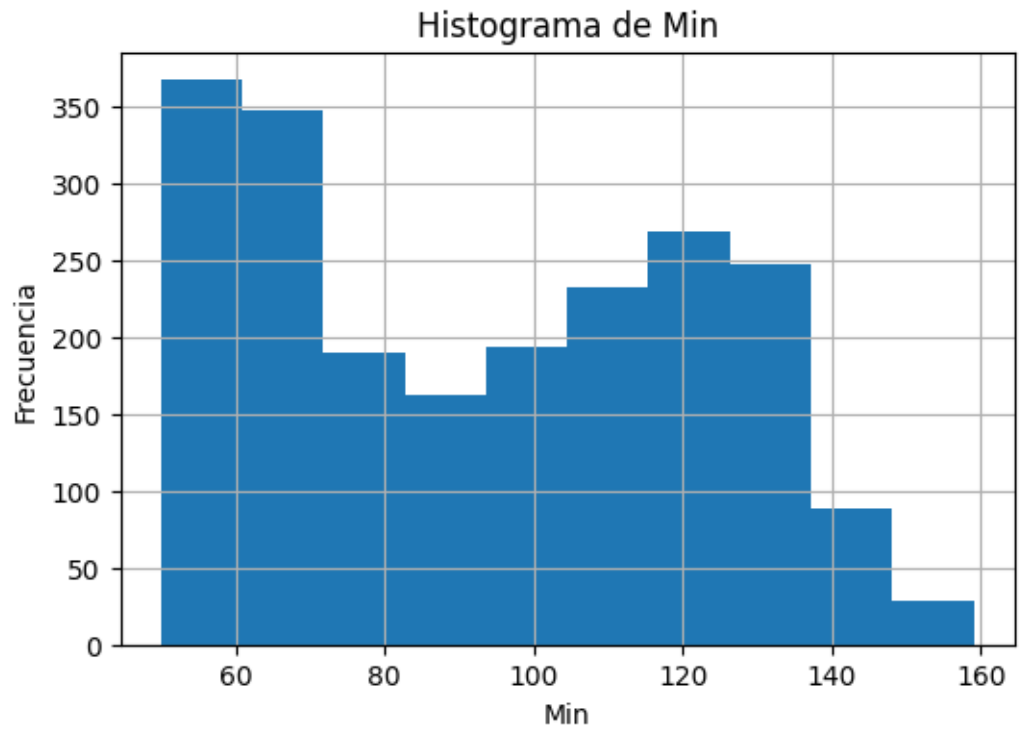
3. Median

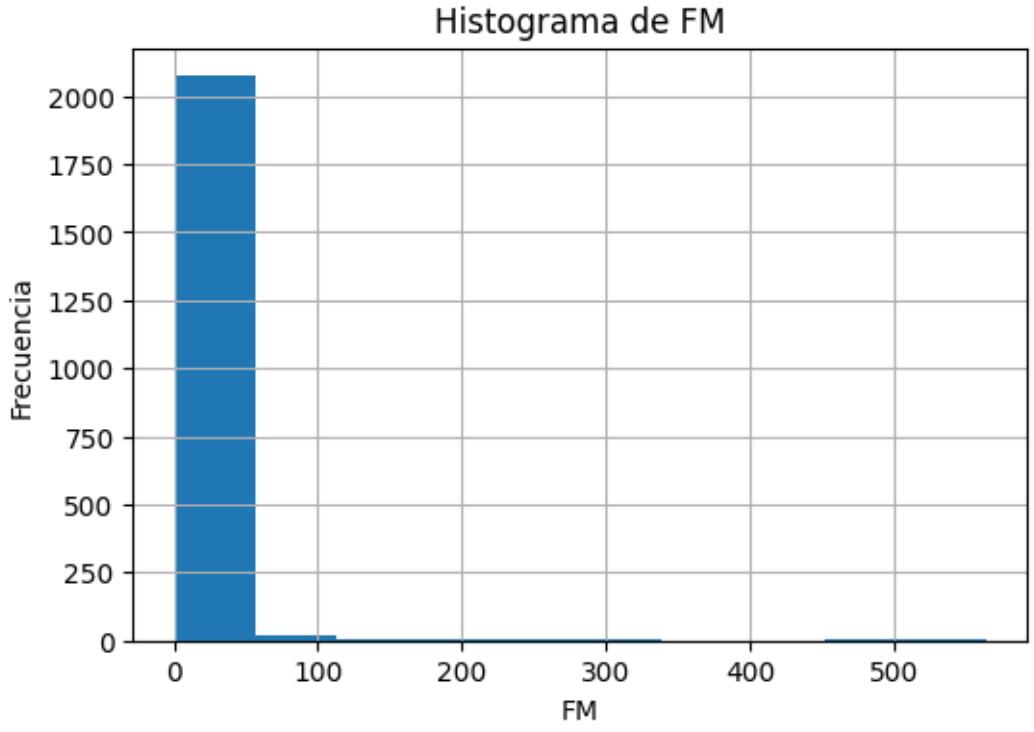
- Media (mean): 138.09
- Desviación estándar (std): 14.47
- Mínimo (min): 77.0
- 25% Cuartil (25%): 129.0
- Mediana (50%): 139.0
- 75% Cuartil (75%): 148.0
- Máximo (max): 186.0

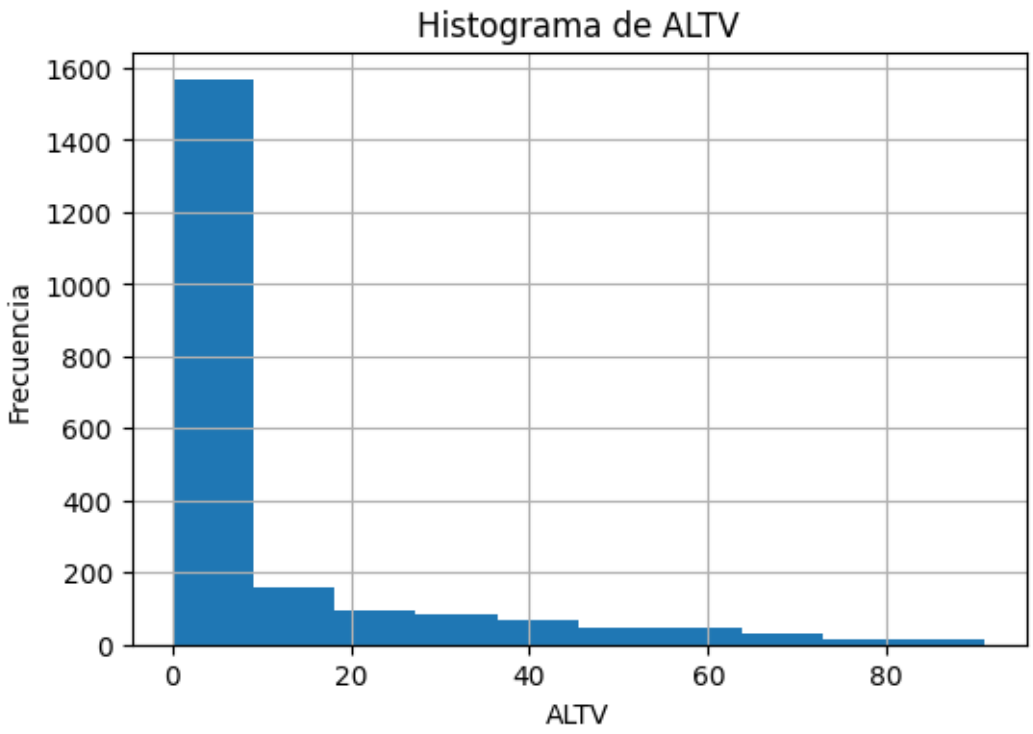
[18]:
```
# HISTOGRAMA PARA VARIABLES LBE, DS Y Min
variables_histograma = ['LBE', 'DS', 'Min', 'FM', 'ALTV', 'AC', 'FM', 'UC']
for variable in variables_histograma:
    plt.figure(figsize=(6, 4))
    data[variable].hist()
    plt.title(f'Histograma de {variable}')
    plt.xlabel(variable)
    plt.ylabel('Frecuencia')
    plt.show()
```

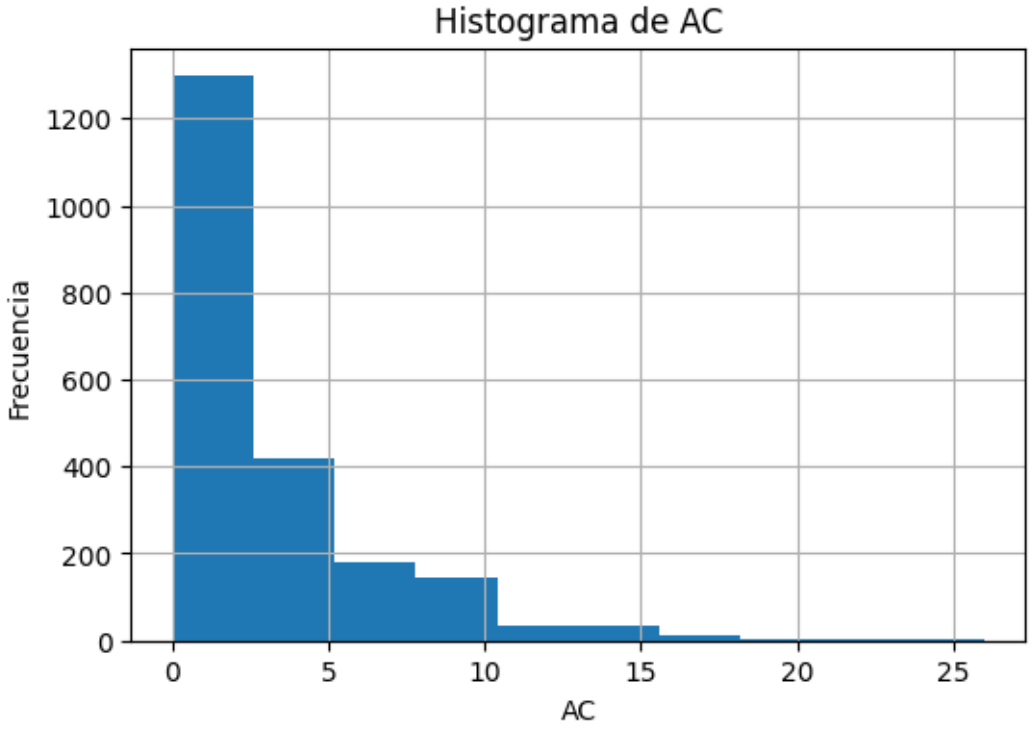

Histograma de LBE
Frecuencia
LBE
0
100
200
300
400
500
110
120
130
140
150
160

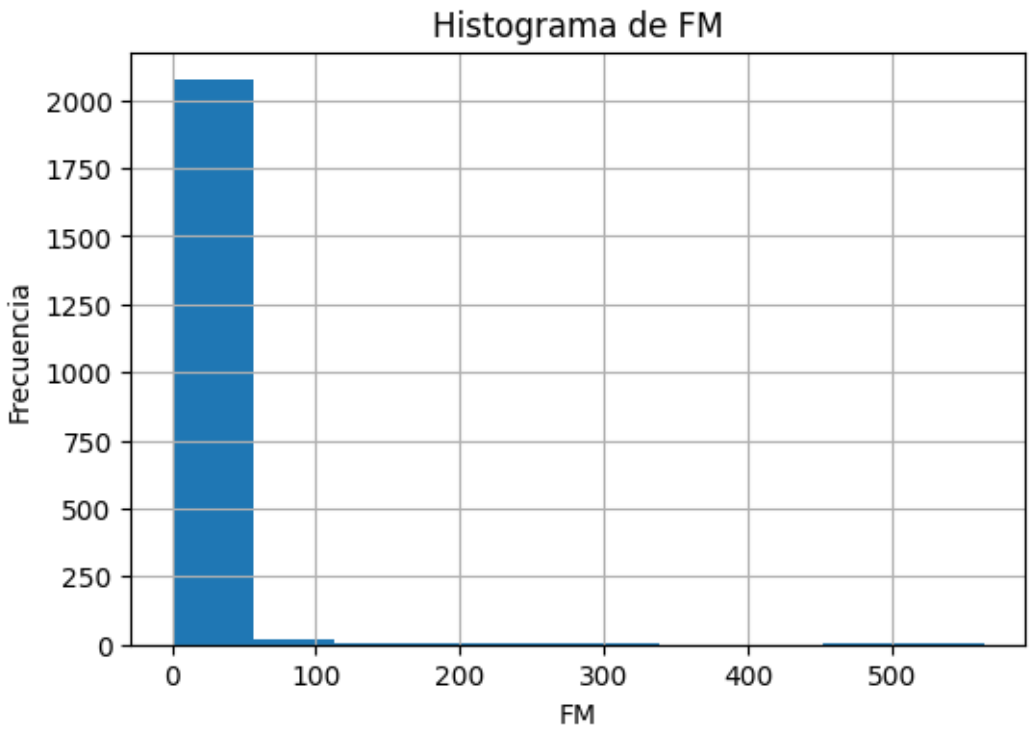

Histograma de DS
Frecuencia
DS
0
250
500
750
1000
1250
1500
1750
2000
0.0
0.2
0.4
0.6
0.8
1.0

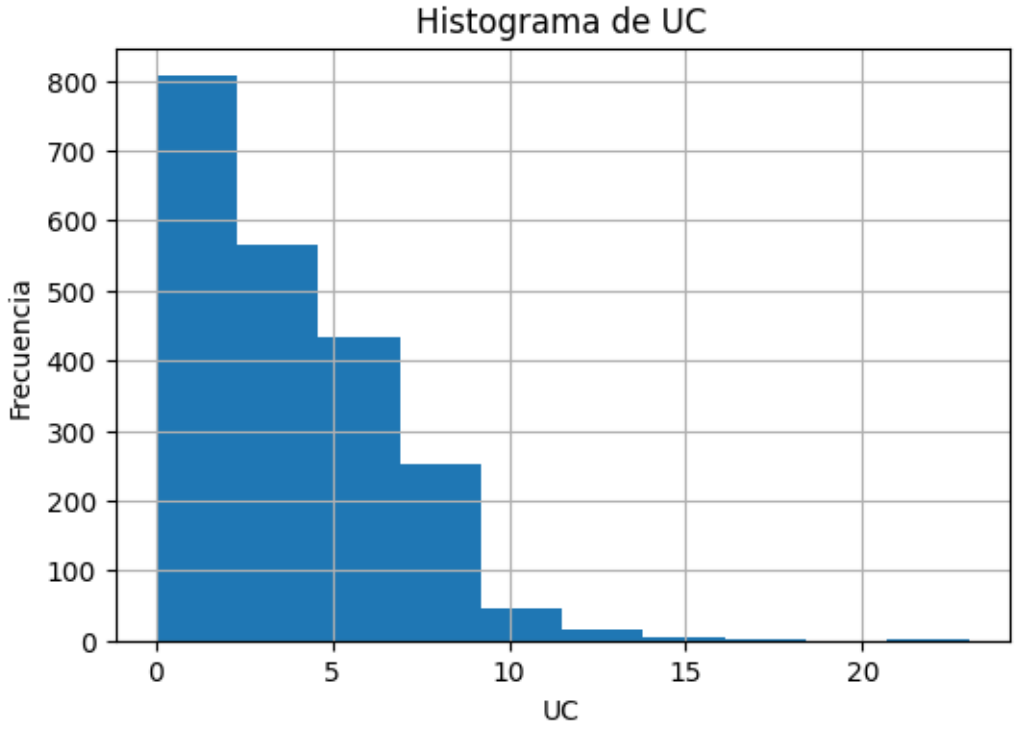

Histograma de Min
Frecuencia
Min
0
50
100
150
200
250
300
350
60
80
100
120
140
160


Histograma de FM
Frecuencia
FM
0
250
500
750
1000
1250
1500
1750
2000
0
100
200
300
400
500


Histograma de ALTV
Frecuencia
ALTV
0
200
400
600
800
1000
1200
1400
1600
0
20
40
60
80


Histograma de AC
Frecuencia
AC
0
200
400
600
800
1000
1200
0
5
10
15
20
25


Histograma de FM
Frecuencia
2000
1750
1500
1250
1000
750
500
250
0
0
100
200
300
400
500
FM


Histograma de UC
Frecuencia
800
700
600
500
400
300
200
100
0
0
5
10
15
20
UC

Obtenga el histograma para las variables: LBE, DS y Min.

1. Histograma de LBE

- Interpretación: Esto sugiere que LBE sigue una distribución bastante equilibrada alrededor de un punto central, con la mayoría de los datos agrupados cerca del promedio.

2. Histograma de DS

- Interpretación: Esto indica que hay poca variabilidad en esta variable, lo que podría significar que la condición medida por DS es consistente o uniforme en la mayoría de los casos.

3. Histograma de Min

- Interpretación: Los valores mínimos tienen una amplia variación y se distribuyen de manera más pareja, indicando diversidad en la medida más baja observada por esta variable.

4. Histograma de FM (Movimientos Fetales):

- Interpretación: Hay una gran cantidad de observaciones con bajos movimientos fetales, y algunos valores atípicos significativos.

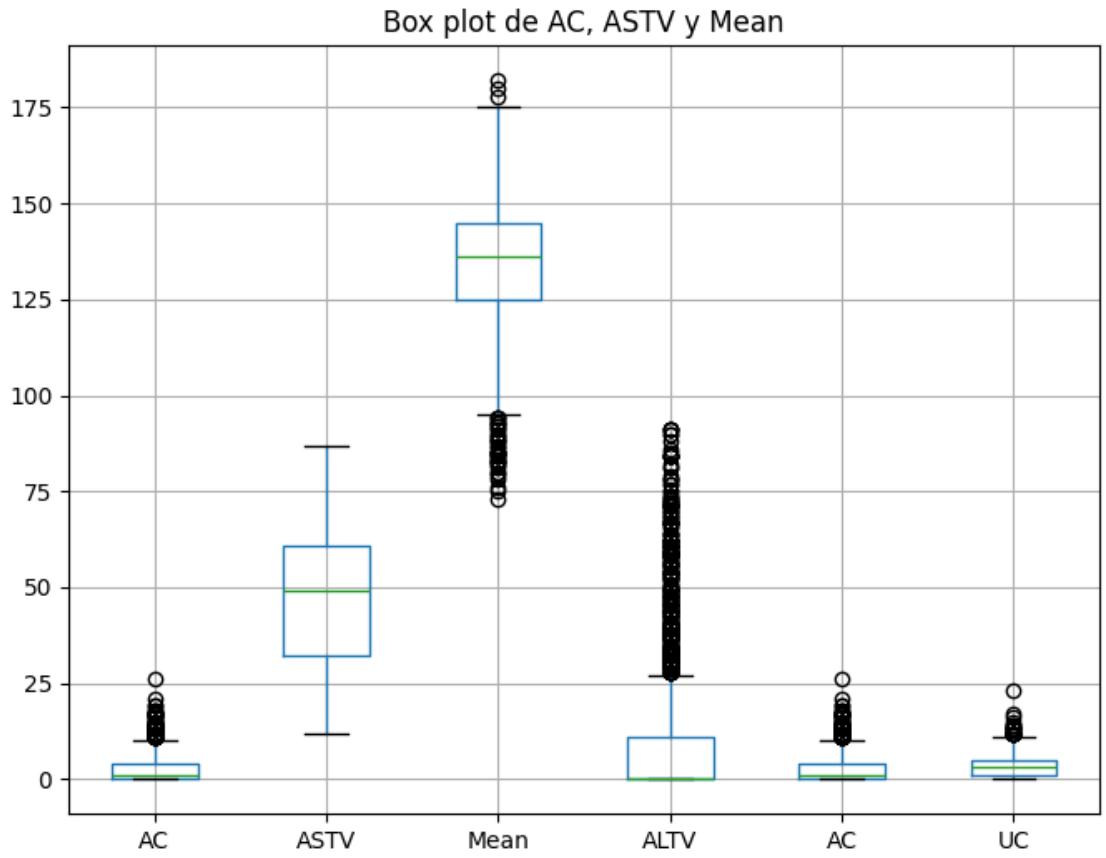
5. Histograma de ALTV (Tiempo de Variabilidad a Largo Plazo):

- Interpretación: La variabilidad a largo plazo es baja en la mayoría de los casos, pero algunos registros muestran valores más altos.

6. Histograma de AC (Aceleraciones): Interpretación: Las aceleraciones son generalmente bajas, con algunas observaciones fuera de lo común.

7. Histograma de UC (Contracciones Uterinas): Interpretación: La mayoría de las contracciones registradas son mínimas, pero hay casos de contracciones más altas.

[22]:
```
# BOX-PLOT PARA VARIABLES AC, ASTV Y Mean
variables_boxplot = ['AC', 'ASTV', 'Mean','ALTV', 'AC', 'UC']
plt.figure(figsize=(8, 6))
data[variables_boxplot].boxplot()
plt.title('Box plot de AC, ASTV y Mean')
plt.show()
```



Obtenga un gráfico box-plot para las variables: AC, ASTV y Mean.

1. AC (Aceleraciones)

- Distribución: La caja es pequeña, indicando poca variabilidad en los datos principales.
- Outliers (valores atípicos): Hay varios, lo que sugiere que aunque la mayoría de los datos están agrupados en un rango pequeño, existen algunos valores bastante altos.

2. ASTV (Tiempo de Variabilidad a Corto Plazo)

- Distribución: La caja es más grande, mostrando más variabilidad en la mitad de los datos.
- Outliers: Menos visibles, pero algunos fuera de los límites.

3. Mean (Promedio de Ciclo Cardiaco)

- Distribución: Amplia variabilidad dentro del rango intercuartílico.
- Mediana: Situada hacia el extremo inferior del rango.
- Outliers: Bastantes valores atípicos por encima del tercer cuartil, mostrando casos que se desvían del valor central esperado

4. AC (Aceleraciones):

- Distribución: Caja pequeña, indicando poca variabilidad en los datos centrales.
- Valores Atípicos: Presentes, sugieren existencia de valores atípicos altos.

5. ASTV (Tiempo de Variabilidad a Corto Plazo):

- Distribución: Caja más grande, mostrando más variabilidad en la mitad de los datos.
- Valores Atípicos: Menos visibles pero presentes fuera de los límites.

6. Mean (Promedio de Ciclo Cardiaco):

- Distribución: Amplia variabilidad dentro del rango intercuartílico.
- Mediana: Situada hacia el extremo inferior del rango.
- Valores Atípicos: Presentes, especialmente por encima del tercer cuartil, mostrando desviaciones del valor central.

[24]:
```
# Seleccionar las columnas donde deseas eliminar outliers
var_outliers = ['AC', 'ASTV', 'Mean', 'UC']

# Eliminar outliers
for var in var_outliers:
    Q1 = data[var].quantile(0.25)
    Q3 = data[var].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filtrar los datos
    data = data[(data[var] >= lower_bound) & (data[var] <= upper_bound)]

# Revisar los datos sin outliers
print(data.describe())
```

```
              LBE           AC           FM           UC          ASTV  \
count  1977.000000  1977.000000  1977.000000  1977.000000  1977.000000
mean    133.448154     2.305008     7.229641     3.471927    46.623166
std       9.908359     2.808774    37.936740     2.641606    17.391833
min     106.000000     0.000000     0.000000     0.000000    12.000000
25%     126.000000     0.000000     0.000000     1.000000    32.000000
50%     133.000000     1.000000     0.000000     3.000000    47.000000
75%     141.000000     4.000000     2.000000     5.000000    61.000000
max     160.000000    10.000000   564.000000    11.000000    87.000000

              MSTV         ALTV         MLTV           DL           DS  … \
count  1977.000000  1977.000000  1977.000000  1977.000000  1977.000000  …
mean      1.278300    10.491148     8.559585     1.445625     0.001012  …
std       0.855797    18.827043     5.497340     2.377186     0.031798  …
min       0.200000     0.000000     0.000000     0.000000     0.000000  …
25%       0.700000     0.000000     5.000000     0.000000     0.000000  …
50%       1.100000     0.000000     7.800000     0.000000     0.000000  …
75%       1.700000    12.000000    11.100000     2.000000     0.000000  …
max       7.000000    91.000000    50.700000    16.000000     1.000000  …
```

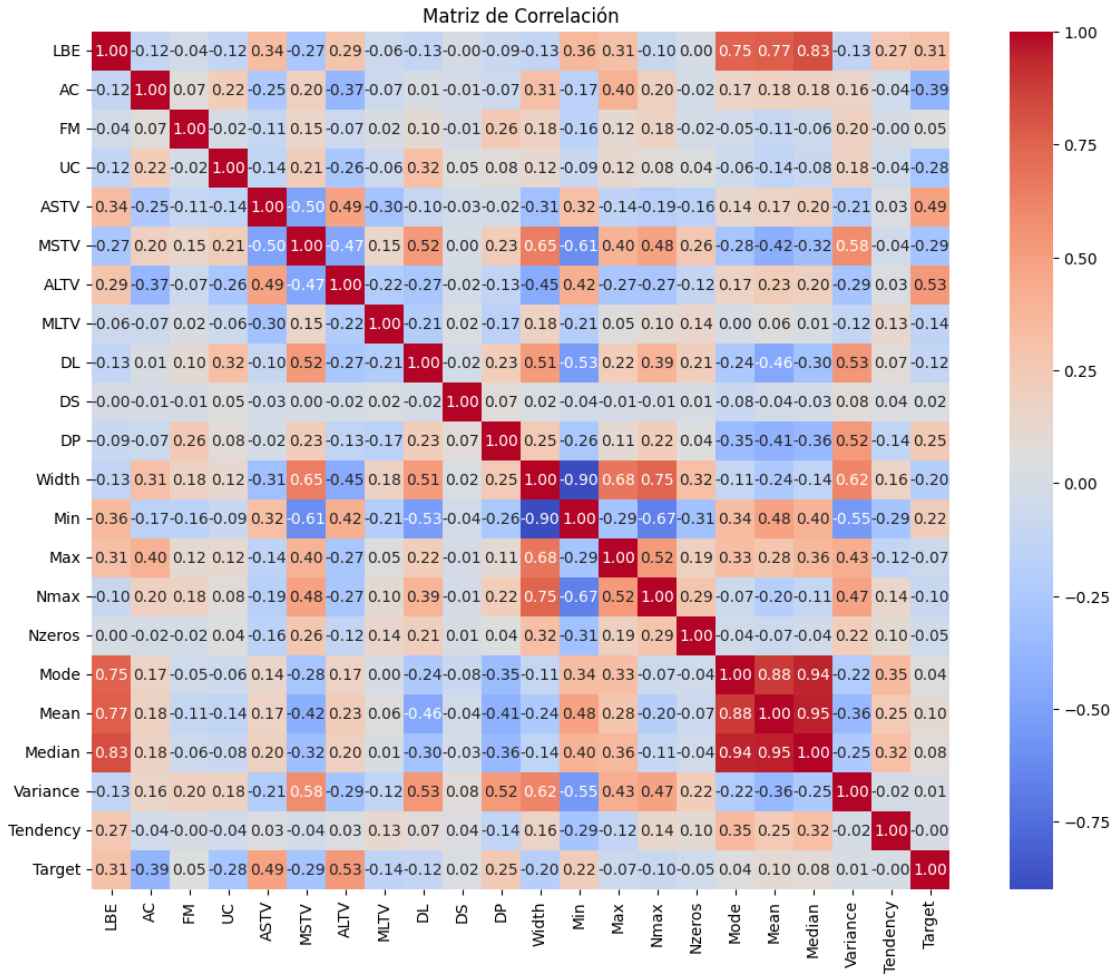
```
                Min          Max         Nmax       Nzeros         Mode  \
count  1977.000000  1977.000000  1977.000000  1977.000000  1977.000000
mean     94.903895   163.022256     3.945878     0.316641   138.024279
std      29.361937    17.452988     2.929434     0.705596    14.736739
min      50.000000   122.000000     0.000000     0.000000    60.000000
25%      68.000000   151.000000     2.000000     0.000000   129.000000
50%      96.000000   161.000000     3.000000     0.000000   139.000000
75%     120.000000   174.000000     6.000000     0.000000   148.000000
max     159.000000   238.000000    18.000000    10.000000   187.000000

               Mean       Median     Variance     Tendency       Target
count  1977.000000  1977.000000  1977.000000  1977.000000  1977.000000
mean    135.302984   138.426404    17.229135     0.332828     0.214466
std      13.924600    13.369241    26.444668     0.598158     0.410555
min      95.000000    86.000000     0.000000    -1.000000     0.000000
25%     125.000000   129.000000     2.000000     0.000000     0.000000
50%     136.000000   139.000000     6.000000     0.000000     0.000000
75%     145.000000   148.000000    22.000000     1.000000     0.000000
max     175.000000   178.000000   269.000000     1.000000     1.000000

[8 rows x 22 columns]
```

- Identificar las variables explicativas con mayor correlación con el target

[25]:
```
# Calcular la matriz de correlación
correlation_matrix = data.corr()

# Imprimir un resumen de las correlaciones del target
print(correlation_matrix['Target'].sort_values(ascending=False))

# Visualización de la matriz de correlación completa
plt.figure(figsize=(14, 10))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm',␣
 ↪square=True)
plt.title('Matriz de Correlación')
plt.show()
```

```
Target      1.000000
ALTV        0.530901
ASTV        0.487891
LBE         0.306657
DP          0.252243
Min         0.220350
Mean        0.095918
Median      0.083922
FM          0.045802
Mode        0.040627
DS          0.022137
```

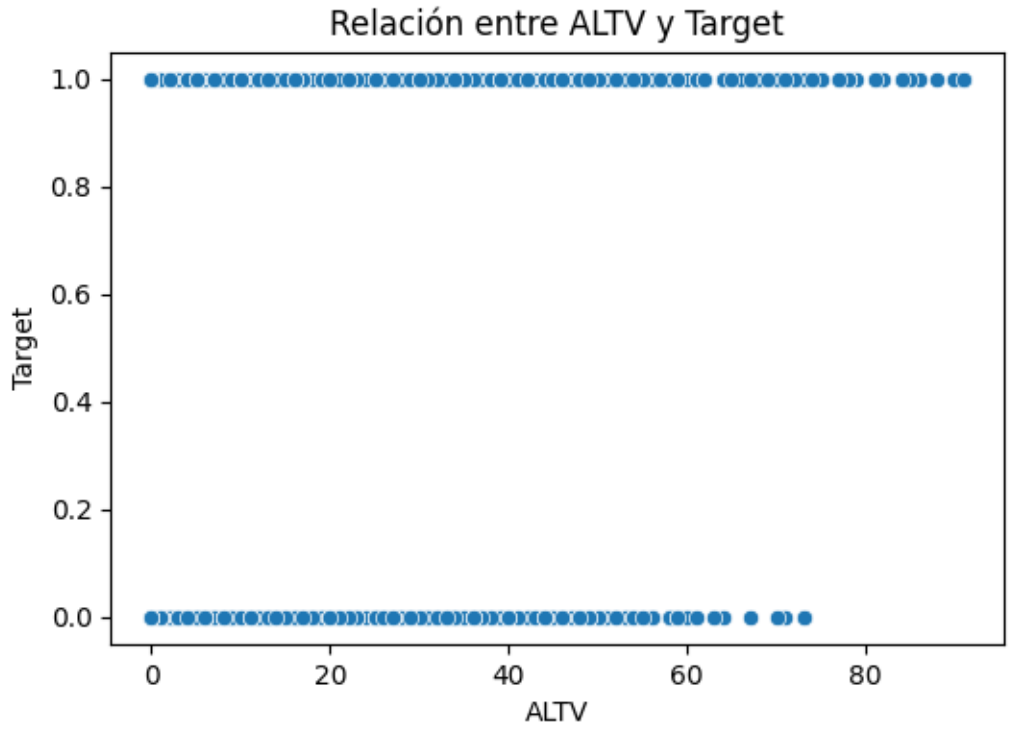
```
Variance     0.011413
Tendency    -0.000245
Nzeros      -0.045868
Max         -0.065432
Nmax        -0.095960
DL          -0.117679
MLTV        -0.143565
Width       -0.199198
UC          -0.279557
MSTV        -0.287639
AC          -0.388532
Name: Target, dtype: float64
```

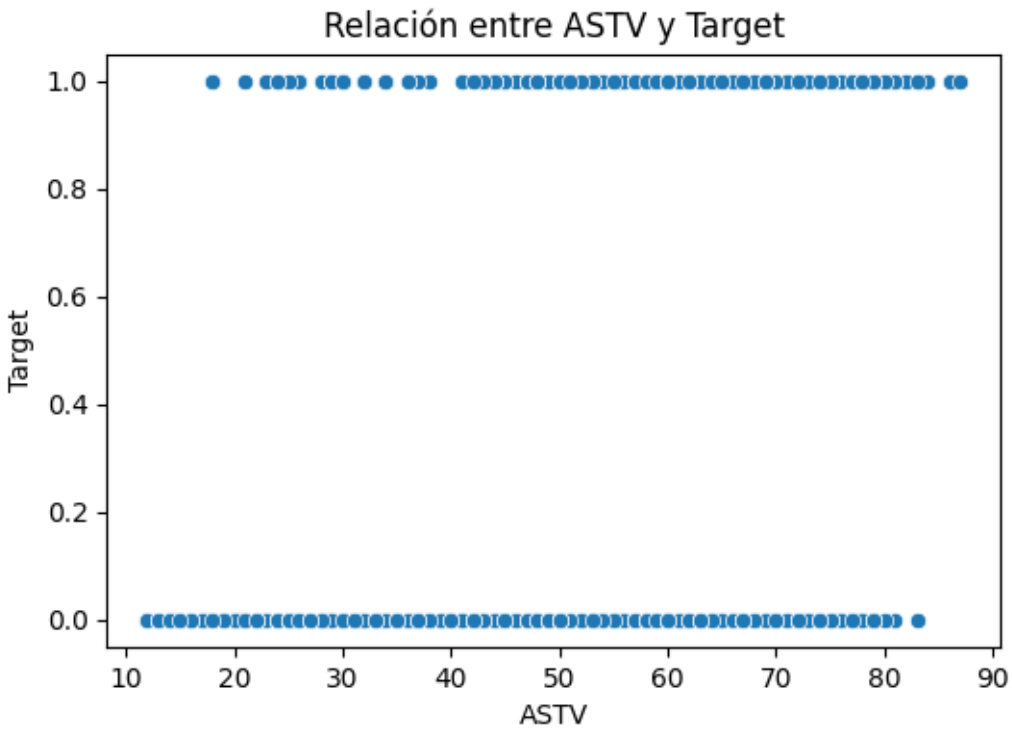


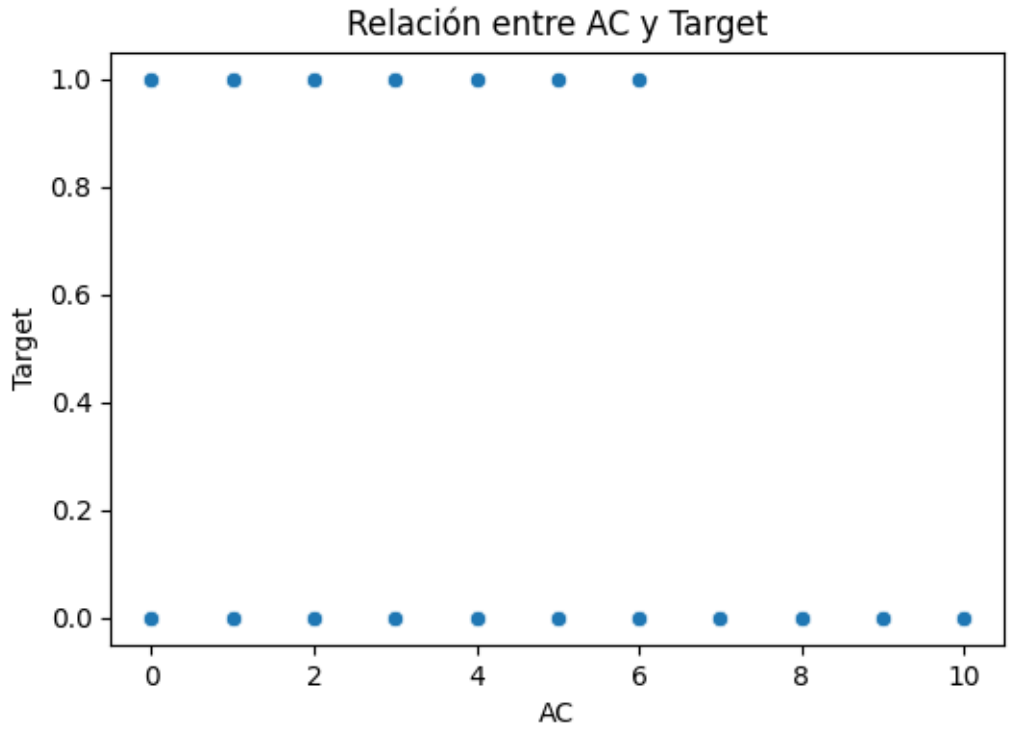
[26]:
```
# Seleccionar las tres variables con mayor correlación positiva
target_correlations = correlation_matrix['Target'].abs().
  ↪sort_values(ascending=False)
top_three_vars = target_correlations.index[1:4]
```

```
print("Las tres variables con mayor correlación con el target son:",␣
 ↪top_three_vars.tolist())

# Graficar la relación de estas variables con el target
for var in top_three_vars:
    plt.figure(figsize=(6, 4))
    sns.scatterplot(x=data[var], y=data['Target'])
    plt.title(f'Relación entre {var} y Target')
    plt.xlabel(var)
    plt.ylabel('Target')
    plt.show()
```

```
Las tres variables con mayor correlación con el target son: ['ALTV', 'ASTV',
'AC']
```




Relación entre ASTV y Target
Target
ASTV
1.0
0.8
0.6
0.4
0.2
0.0
10
20
30
40
50
60
70
80
90


Relación entre AC y Target
Target
AC
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10

[27]:
```
# Calcular las correlaciones con el target
correlations = data.corr()

# Obtener las tres variables con mayor correlación absoluta con el target
target_correlations = correlations['Target'].abs().sort_values(ascending=False)
top_three_vars = target_correlations.index[1:4]
print("Las tres variables con mayor correlación con el target son:",␣
 ↪top_three_vars.tolist())
```

```
Las tres variables con mayor correlación con el target son: ['ALTV', 'ASTV',
'AC']
```

2 2. Modelizacion: conjunto de datos en los conjuntos de entrenamiento y test

[28]:
```
from sklearn.model_selection import train_test_split
```

[31]:
```
# Definir X e y
X = data.loc[:, data.columns != "Target"]
y = data.loc[:, data.columns == "Target"]

# Separar los datos en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40,␣
 ↪random_state=0)

# Verificar las dimensiones
print("Tamaño del conjunto de entrenamiento:", X_train.shape)
print("Tamaño del conjunto de entrenamiento (y):", y_train.shape)
print("Tamaño del conjunto de prueba:", X_test.shape)
print("Tamaño del conjunto de prueba (y):", y_test.shape)

# Modelo de Naive Bayes
gnb = GaussianNB()
modelNB = gnb.fit(X_train, y_train.values.ravel())
y_pred_train_nb = modelNB.predict_proba(X_train)[:, 1]
y_pred_test_nb = modelNB.predict_proba(X_test)[:, 1]

# Modelo SVM
svm_model = SVC(probability=True, random_state=0)
svm_model.fit(X_train, y_train.values.ravel())
y_pred_test_svm = svm_model.predict_proba(X_test)[:, 1]
```

```
Tamaño del conjunto de entrenamiento: (1186, 21)
Tamaño del conjunto de entrenamiento (y): (1186, 1)
Tamaño del conjunto de prueba: (791, 21)
Tamaño del conjunto de prueba (y): (791, 1)
```

3 4. Modelizacion: Ajustar el algoritmo de Naive Bayes.

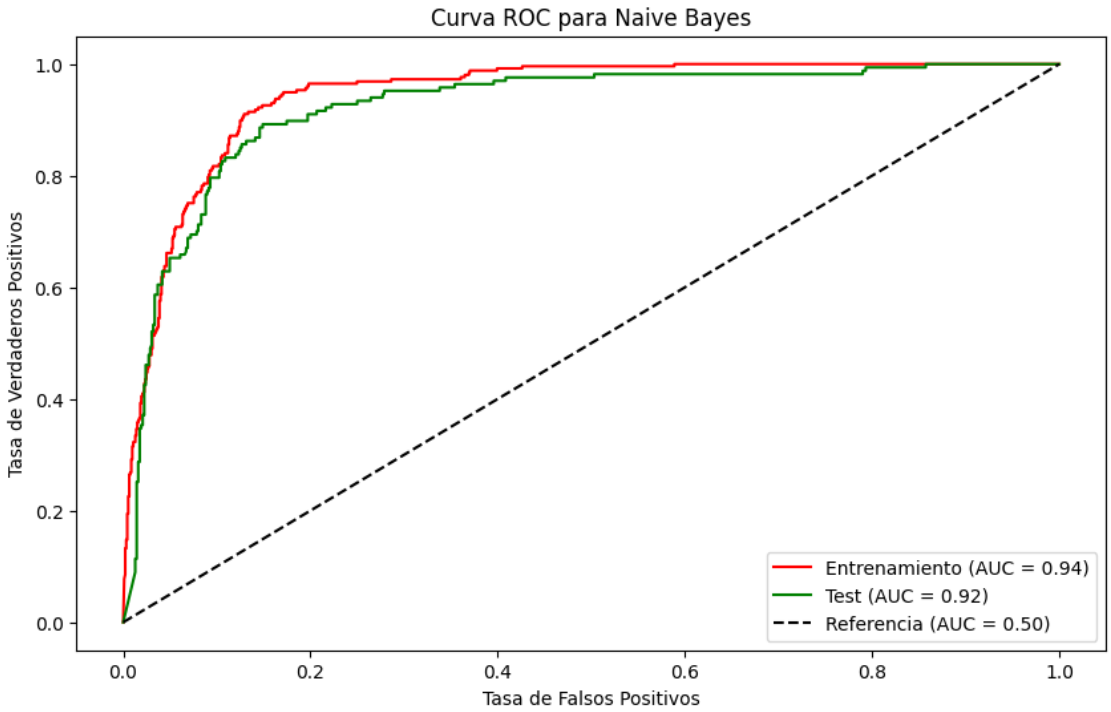
[32]:
```
gnb = GaussianNB()

# Ajustar el modelo con los datos de entrenamiento
modelNB = gnb.fit(X_train, y_train.values.ravel())

# Predicción de probabilidades
y_pred_train = modelNB.predict_proba(X_train)[:, 1]
y_pred_test = modelNB.predict_proba(X_test)[:, 1]

print(y_pred_train)
#print(y_pred_test)
```

```
[1.51451575e-05 9.98919769e-01 7.60431591e-06 … 1.91682570e-19
 4.50254828e-05 1.02398992e-12]
```

[ ]:
```
```

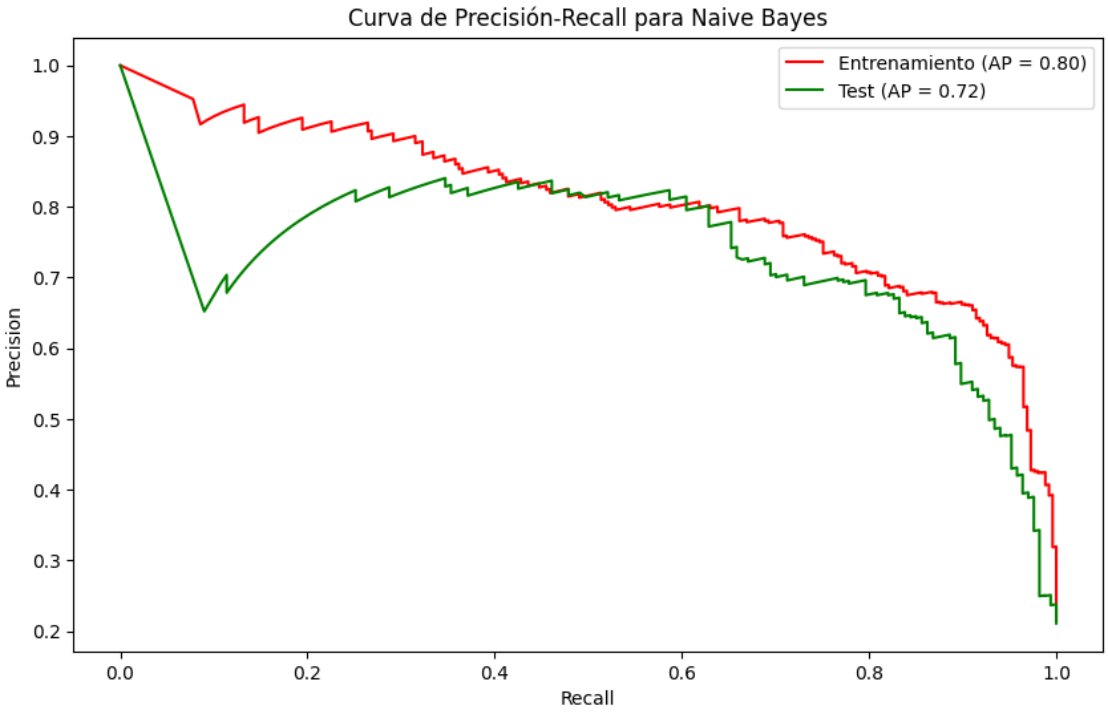
[33]:
```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt
```

[34]:
```
# Calcular la curva ROC y el área bajo la curva (AUC) para entrenamiento y test
fpr_train, tpr_train, _ = roc_curve(y_train, y_pred_train)
fpr_test, tpr_test, _ = roc_curve(y_test, y_pred_test)

auc_train = roc_auc_score(y_train, y_pred_train)
auc_test = roc_auc_score(y_test, y_pred_test)

# Graficar la curva ROC
plt.figure(figsize=(10, 6))
plt.plot(fpr_train, tpr_train, label=f'Entrenamiento (AUC = {auc_train:.2f})',␣
 ↪color='red')
plt.plot(fpr_test, tpr_test, label=f'Test (AUC = {auc_test:.2f})',␣
 ↪color='green')
plt.plot([0, 1], [0, 1], 'k--', label='Referencia (AUC = 0.50)')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC para Naive Bayes')
plt.legend(loc='best')
plt.show()
```

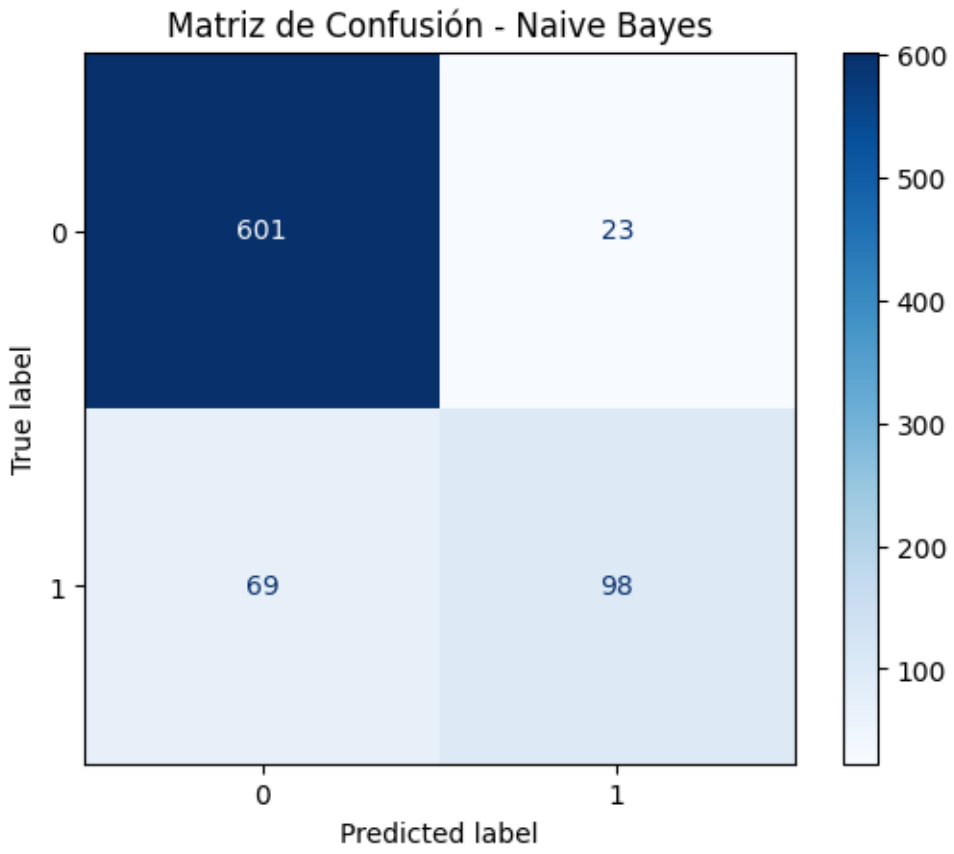


[35]:
```
from sklearn.metrics import precision_recall_curve, average_precision_score
```

[36]:
```
# Calcular la curva de Precisión-Recall y el área bajo la curva (AP) para␣
 ↪entrenamiento y test
precision_train, recall_train, _ = precision_recall_curve(y_train, y_pred_train)
precision_test, recall_test, _ = precision_recall_curve(y_test, y_pred_test)

ap_train = average_precision_score(y_train, y_pred_train)
ap_test = average_precision_score(y_test, y_pred_test)

# Graficar la curva de Precisión-Recall
plt.figure(figsize=(10, 6))
plt.plot(recall_train, precision_train, label=f'Entrenamiento (AP = {ap_train:.
 ↪2f})', color='red')
plt.plot(recall_test, precision_test, label=f'Test (AP = {ap_test:.2f})',␣
 ↪color='green')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Curva de Precisión-Recall para Naive Bayes')
plt.legend(loc='best')
plt.show()
```



[52]:
```
# Matriz de confusión para Naive Bayes
cm_nb = confusion_matrix(y_test, y_pred_nb)

# Graficar matriz de confusión para Naive Bayes
disp_nb = ConfusionMatrixDisplay(confusion_matrix=cm_nb)
disp_nb.plot(cmap='Blues')
plt.title("Matriz de Confusión - Naive Bayes")
plt.show()
```



Conclusión: Análisis de Naive Bayes

1. Curva ROC:

- Interpretación: La pequeña diferencia entre las AUC de entrenamiento y prueba es positiva, mostrando que el modelo es estable y efectivo.

2. Curva de Precisión-Recall:

- Interpretación: La caída en el área de precisión-recall de entrenamiento a prueba es moderada, lo que sugiere que hay espacio para mejorar en la captura de positivos reales, pero en general el modelo es robusto en ambas métricas.

3. Martiz de confusion:

- Buena precisión: El modelo es bastante correcto en sus predicciones.
- Mejor sensibilidad: El modelo identifica mejor los casos positivos en comparación con el modelo SVM analizado anteriormente.
- Precisión positiva sólida: Indica que las predicciones positivas generalmente son correctas con mayor frecuencia.

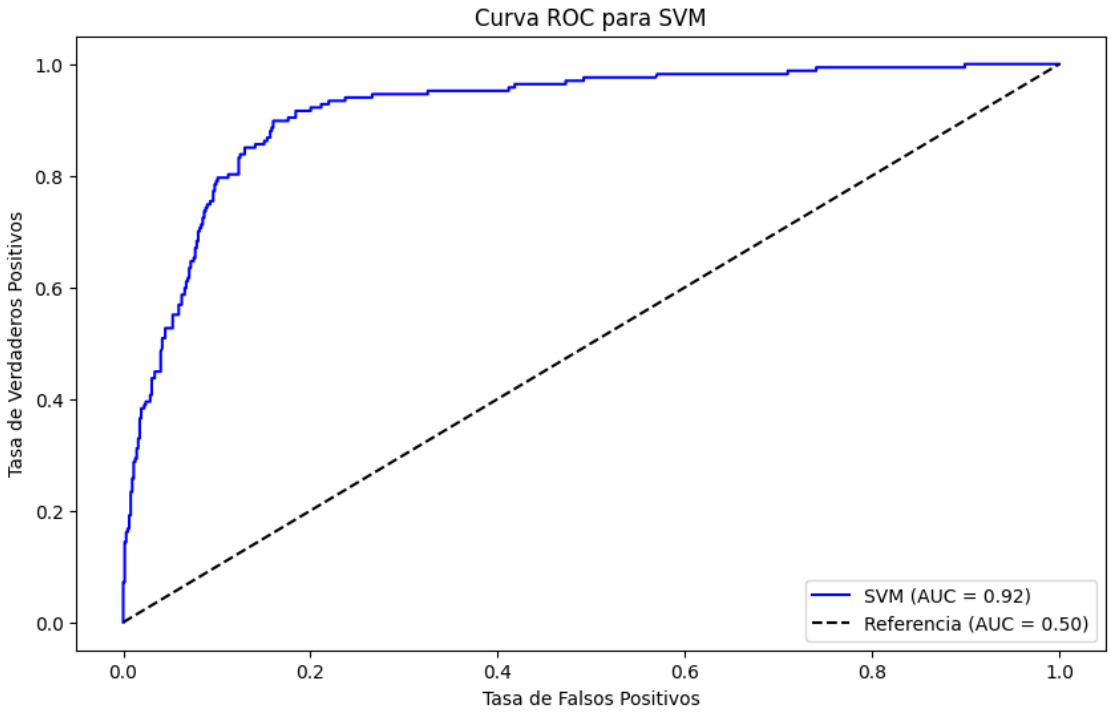
El modelo Naive Bayes demuestra un buen desempeño tanto en entrenamiento como en prueba, con alta AUC y un desempeño sólido de precisión-recall. Sigue existiendo un balance razonable

entre precisión y recall, asegurando que el modelo es confiable y efectivo para la clasificación. Esto lo hace adecuado para aplicaciones donde la diferenciación de clases es crucial.

4 5. Clasificación con Support Vector Machine (SVM)

[37]:
```
# Ajustar un modelo SVM
svm_model = SVC(probability=True, random_state=0)
svm_model.fit(X_train, y_train.values.ravel())

# Predicciones de probabilidad con SVM
y_pred_test_svm = svm_model.predict_proba(X_test)[:, 1]
```

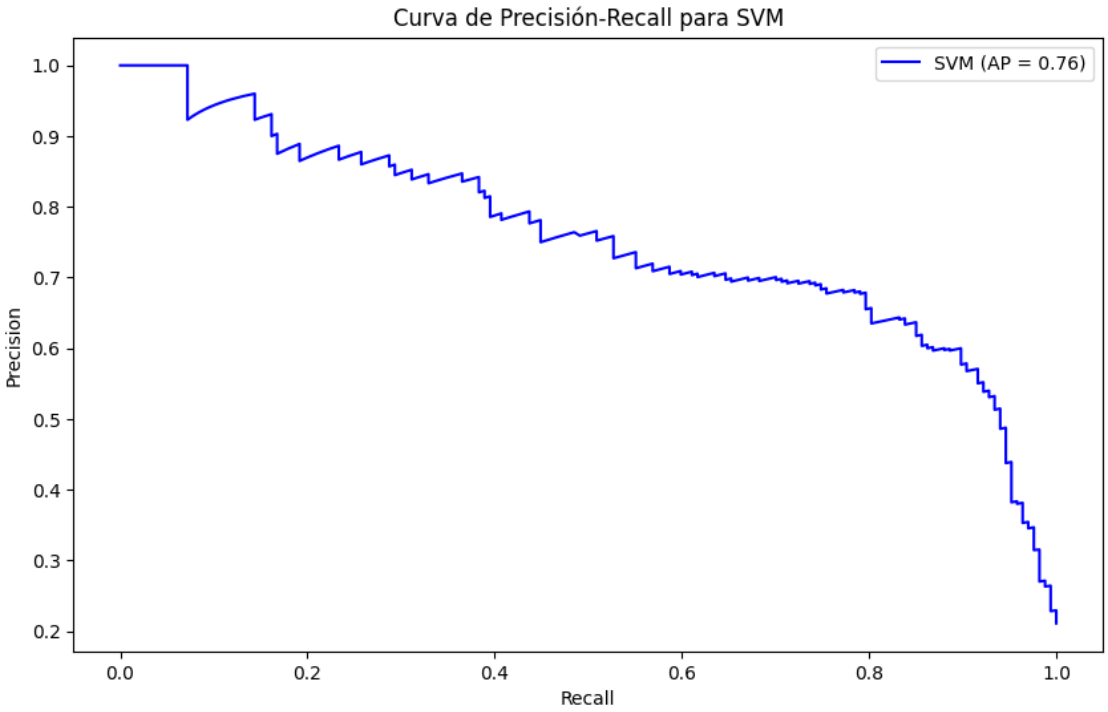
[38]:
```
# Obtener la curva ROC y el área bajo la curva (AUC) para SVM en el conjunto de␣
↪test
fpr_svm, tpr_svm, _ = roc_curve(y_test, y_pred_test_svm)
auc_svm = roc_auc_score(y_test, y_pred_test_svm)

# Graficar la curva ROC
plt.figure(figsize=(10, 6))
plt.plot(fpr_svm, tpr_svm, label=f'SVM (AUC = {auc_svm:.2f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--', label='Referencia (AUC = 0.50)')
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curva ROC para SVM')
plt.legend(loc='best')
plt.show()

# Imprimir resultados
print(f"AUC del modelo SVM en test: {auc_svm:.2f}")
```




```
AUC del modelo SVM en test: 0.92
```

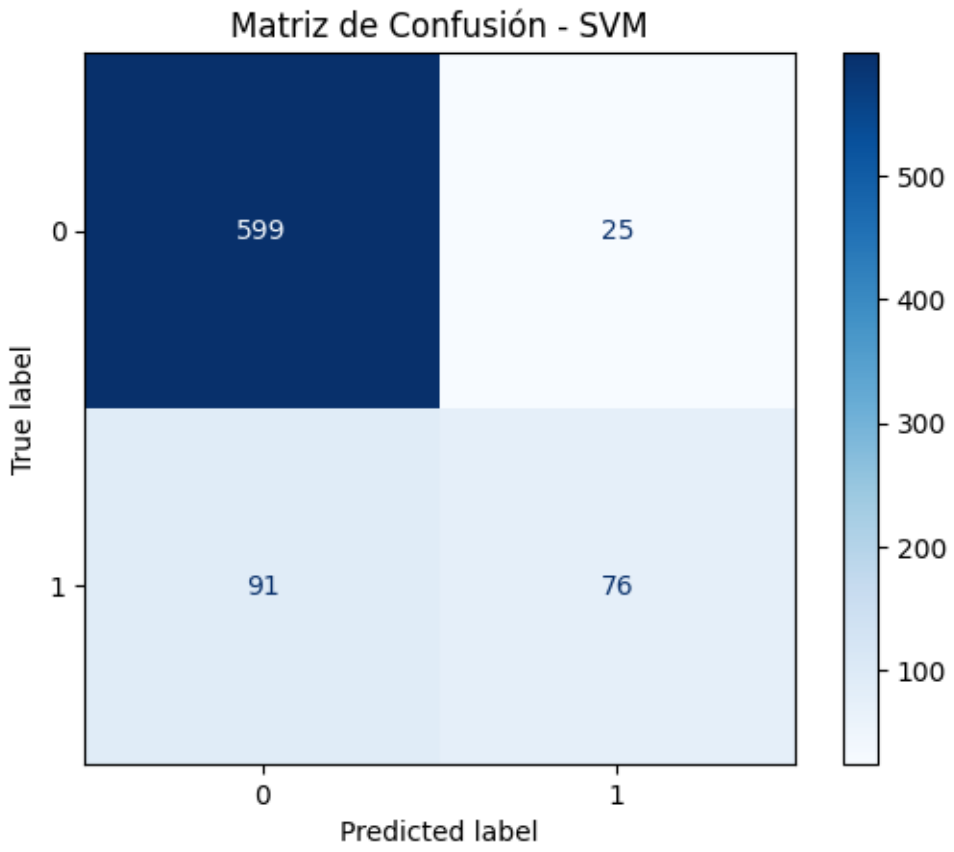
[43]:
```
# Predicciones de probabilidad para el conjunto de prueba
y_pred_test_svm = svm_model.predict_proba(X_test)[:, 1]

# Calcular la curva de Precisión-Recall y el área bajo la curva (AP) para SVM
precision_test_svm, recall_test_svm, _ = precision_recall_curve(y_test,␣
↪y_pred_test_svm)
ap_test_svm = average_precision_score(y_test, y_pred_test_svm)
```

[44]:
```
# Graficar la curva de Precisión-Recall para SVM
plt.figure(figsize=(10, 6))
plt.plot(recall_test_svm, precision_test_svm, label=f'SVM (AP = {ap_test_svm:.
↪2f})', color='blue')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Curva de Precisión-Recall para SVM')
plt.legend(loc='best')
plt.show()

# Imprimir el área promedio bajo la curva
print(f"Área Promedio bajo la curva (AP) para SVM en test: {ap_test_svm:.2f}")
```




```
Área Promedio bajo la curva (AP) para SVM en test: 0.76
```

[61]:
```
# Predicciones con SVM
y_pred_svm = svm_model.predict(X_test)

# Matriz de confusión para SVM
cm_svm = confusion_matrix(y_test, y_pred_svm)

# Graficar matriz de confusión para SVM
disp_svm = ConfusionMatrixDisplay(confusion_matrix=cm_svm)
disp_svm.plot(cmap='Blues')
plt.title("Matriz de Confusión - SVM")
plt.show()
```



Conclucion: Análisis de SVM 1. Curva ROC: AUC (0.92): Una AUC cercana a 1 significa que el modelo es eficaz en predecir correctamente tanto los falsos positivos como los verdaderos positivos.

2. Curva de Precisión-Recall: AP (0.76): Indica un buen equilibrio entre precisión y recall, aunque hay margen para mejorar.

3. Matriz de confusión

- Alta precisión: El modelo es generalmente correcto en sus predicciones.
- Baja sensibilidad: El modelo no logra identificar correctamente muchos de los casos positivos.
- Precisión positiva razonable: Indica que cuando el modelo predice positivo, suele tener razón con buena frecuencia.

Conclusión El modelo SVM tiene un rendimiento sólido, con una buena capacidad de generalización reflejada en las métricas AUC y AP. Es fiable para tareas de clasificación en este conjunto de datos, aunque podría ser optimizado para mejorar su precisión.

5 5. Mejoramos modelo SVM con GridSearchCV

[62]:
```
# Definición del espacio de hiperparámetros para GridSearchCV
param_grid = [
    {"kernel": ["rbf"], "gamma": [1e-3, 1e-4], "C": [0.1, 1, 10]},  # Kernel RBF
    {"kernel": ["linear"], "C": [0.1, 1, 10]},                      # Kernel␣
 ↪Linear
    {"kernel": ["poly"], "C": [0.1, 1, 10], "degree": [2, 3]},      # Kernel␣
 ↪Polynomial
]

# Configuración y ajuste del GridSearchCV
grid = GridSearchCV(
    estimator=SVC(probability=True),  # Activamos probability=True para curvas␣
 ↪ROC
    param_grid=param_grid,
    scoring='roc_auc',  # Métrica: AUC
    n_jobs=-1,
    cv=3,
    verbose=0,
    return_train_score=True
)

# Ajustar el modelo en el conjunto de entrenamiento
grid.fit(X_train, y_train.values.ravel())

# Mostrar los mejores parámetros encontrados
best_params = grid.best_params_
print("Mejores parámetros encontrados:", best_params)
```

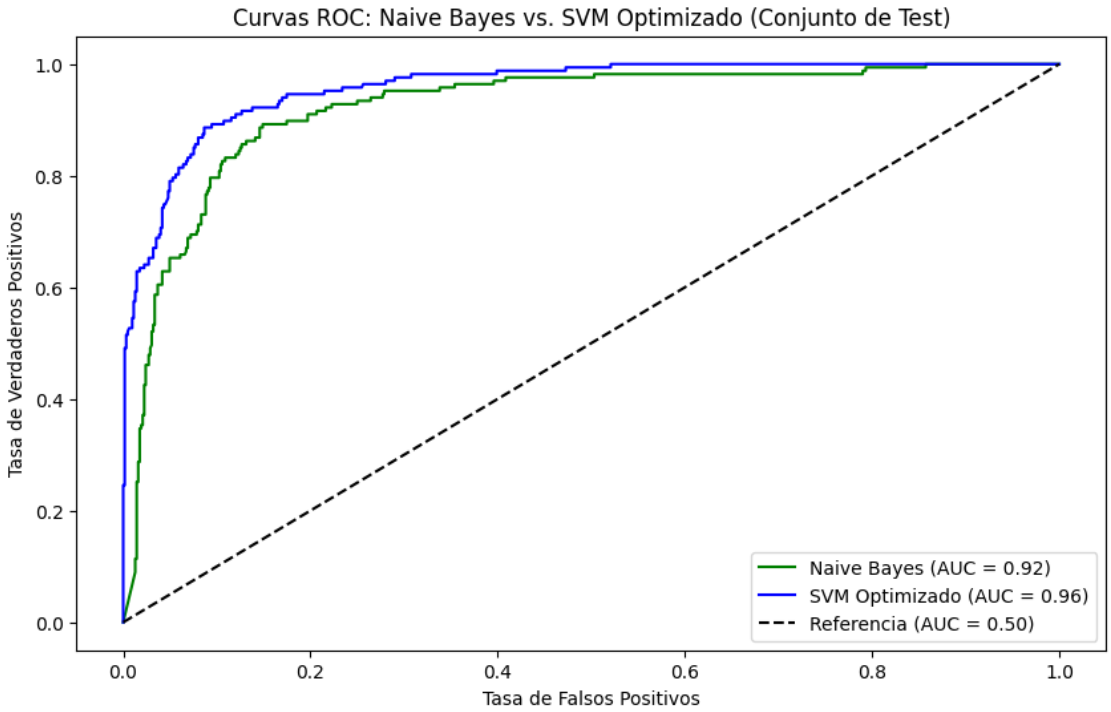
```
Mejores parámetros encontrados: {'C': 1, 'kernel': 'linear'}
```

[63]:
```
# Reentrenar el modelo SVM con los mejores parámetros
best_svm_model = SVC(probability=True, **best_params)
best_svm_model.fit(X_train, y_train.values.ravel())
```

[63]:
```
SVC(C=1, kernel='linear', probability=True)
```

[64]:
```
# Predicciones de probabilidad en el conjunto de test
y_pred_test_svm_optimized = best_svm_model.predict_proba(X_test)[:, 1]

# Obtener la curva ROC y el AUC para el modelo SVM optimizado en el conjunto de␣
 ↪test
fpr_svm_optimized, tpr_svm_optimized, _ = roc_curve(y_test,␣
 ↪y_pred_test_svm_optimized)
auc_svm_optimized = roc_auc_score(y_test, y_pred_test_svm_optimized)
```

```
# Graficar la comparación de curvas ROC (Naive Bayes vs SVM optimizado)
plt.figure(figsize=(10, 6))
plt.plot(fpr_test, tpr_test, label=f'Naive Bayes (AUC = {auc_test:.2f})',␣
 ↪color='green')  # Naive Bayes
plt.plot(fpr_svm_optimized, tpr_svm_optimized, label=f'SVM Optimizado (AUC =␣
 ↪{auc_svm_optimized:.2f})', color='blue')  # SVM optimizado
plt.plot([0, 1], [0, 1], 'k--', label='Referencia (AUC = 0.50)')  # Línea de␣
 ↪referencia
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Curvas ROC: Naive Bayes vs. SVM Optimizado (Conjunto de Test)')
plt.legend(loc='best')
plt.show()

# Imprimir resultados finales
print(f"AUC del modelo Naive Bayes en test: {auc_test:.2f}")
print(f"AUC del modelo SVM optimizado en test: {auc_svm_optimized:.2f}")
```

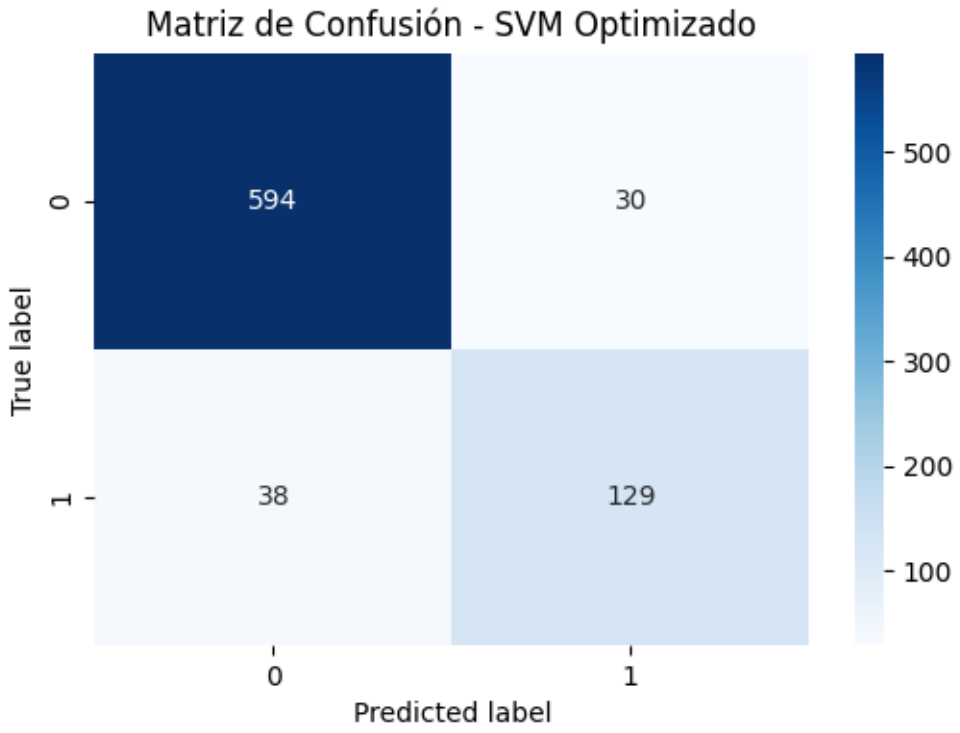



```
AUC del modelo Naive Bayes en test: 0.92
AUC del modelo SVM optimizado en test: 0.96
```

[67]:
```
# Matriz de confusión del SVM optimizado muestra
y_pred_svm_optimized = best_svm_model.predict(X_test)
```

```
conf_matrix = confusion_matrix(y_test, y_pred_svm_optimized)
print(conf_matrix)

plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Matriz de Confusión - SVM Optimizado')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()
```

```
[[594  30]
 [ 38 129]]
```



6 6. Conclusiones Generales:

- SVM Optimizado supera a Naive Bayes con mejor AUC (0.96 vs 0.92).
- Logra un mejor equilibrio entre precisión y sensibilidad.
- Menos falsos negativos y positivos en la matriz de confusión.
- Mejor discriminación en la curva ROC.
- Más adecuado para tareas de clasificación complejas