Métricas de modelos PREDICTIVOS DISCRETOS

1. PASOS PREVIOS

#Librerías

In [163]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Módulo de separación de datos
from sklearn.model_selection import train_test_split
# Módulo de Matriz de confusión
from sklearn.metrics import confusion_matrix
from sklearn.neural_network import MLPClassifier
# Módulo de Reporte de métricas
from sklearn.metrics import classification_report
# Métricas
from sklearn import metrics
# Curva ROC
from sklearn.metrics import RocCurveDisplay

# muestra el gráfico en el cuaderno (no utiliza una ventana)
%matplotlib inline
```

Carga de datos

In [170]:
```
# El objetivo del dataset, es identificación de sexo en en base a las medidas del resto
# sexo = f(peso, altura, pie, hombros, brazos, caderas, ojos)
df = pd.read_csv("ds_persona2_ML.csv")
df.head()
```

Out[170]:

	Peso	Altura	Pie	Hombros	Brazos	Caderas	Ojos	Sexo
0	60	163	37	41	68	95	1	1
1	52	166	37	37	70	87	2	1
2	61	172	39	39	69	91	2	1
3	73	181	43	50	78	101	2	2
4	53	172	39	39	72	89	1	1

In [171]:
```
# diagrama de dispersión en 3D: edad vs. fracción de eyección vs. sodio sérico
import plotly.express as px
fig = px.scatter_3d(df, x='Peso', y='Altura',
                    z='Brazos', color='Sexo')
fig.update_traces(marker_size = 3)
fig.show()
```

2. APLICACION DEL MODELO PREDICTIVO

Separación de datos para entrenamiento y test

In [172]:
```
X = df.drop(['Sexo'], axis=1)
y = df['Sexo']
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, random_state=
```

In [173]:
```
X.head()
```

Out[173]:

	Peso	Altura	Pie	Hombros	Brazos	Caderas	Ojos
0	60	163	37	41	68	95	1
1	52	166	37	37	70	87	2
2	61	172	39	39	69	91	2
3	73	181	43	50	78	101	2
4	53	172	39	39	72	89	1

In [174]:
```
y.head()
```

Out[174]:
```
0    1
1    1
2    1
3    2
4    1
Name: Sexo, dtype: int64
```

Entrenamiento con árbol de decisión

In [175]:
```
# Configuración del modelo
modelo = MLPClassifier(max_iter=1000)

# Entrenamiento del modelo
modelo.fit(X_train, y_train)

# Evaluación del modelo
modelo.score(X_test, y_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.
py:686: ConvergenceWarning:

Stochastic Optimizer: Maximum iterations (1000) reached and the optimization hasn't co
nverged yet.
```

Out[175]:
```
0.92
```

3. PREDICCIÓN DEL MODELO

Predicción de datos de test

In [176]:
```
pred = modelo.predict(X_test)
pd.DataFrame([pred,y_test])
```

Out[176]:

	0	1	2	3	4	5	6	7	8	9	...	15	16	17	18	19	20	21	22	23	24
0	1	1	1	1	1	1	1	1	2	1	...	1	2	1	2	2	1	1	1	1	1
1	1	1	1	1	2	1	1	1	2	1	...	1	2	1	2	2	1	1	1	1	1

2 rows × 25 columns

4. CÁLCULO Y VISUALIZACIÓN DE MÉTRICAS

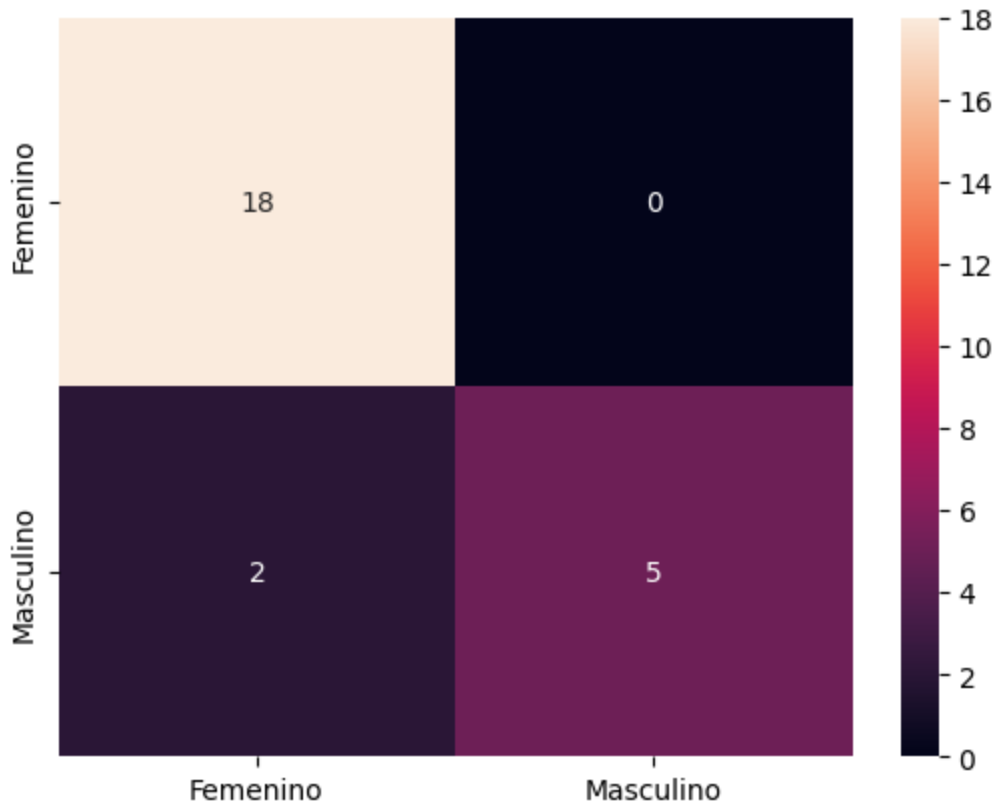
Matriz de confusión

In [179]:
```
MatrizConfusion = confusion_matrix(y_test,pred)
print('Matriz de Confusión:')
print(MatrizConfusion)
print('Exactitud = ', Exactitud)
```

```
Matriz de Confusión:
[[18  0]
 [ 2  5]]
Exactitud =  0.92
```

In [180]:
```
import seaborn as sns

categorias = ['Femenino', 'Masculino']
sns.heatmap(MatrizConfusion, yticklabels=categorias, xticklabels=categorias, annot=True)
print('Exactitud = ', Exactitud)
```

```
Exactitud =  0.92
```



Obtención de métricas

In [181]:
```
Exactitud = metrics.accuracy_score(y_test,pred)
Precision = metrics.precision_score(y_test,pred)
Sensibilidad = metrics.recall_score(y_test,pred)
PuntuacionF1 = metrics.f1_score(y_test,pred)
Reporte = classification_report(y_test, pred)

print('Exactitud    : ', Exactitud)
print('Precisión    : ', Precision)
print('Sensibilidad : ', Sensibilidad)
print('Puntuación F1: ', PuntuacionF1)
```

```
Exactitud    :  0.92
Precisión    :  0.9
Sensibilidad :  1.0
Puntuación F1:  0.9473684210526316
```

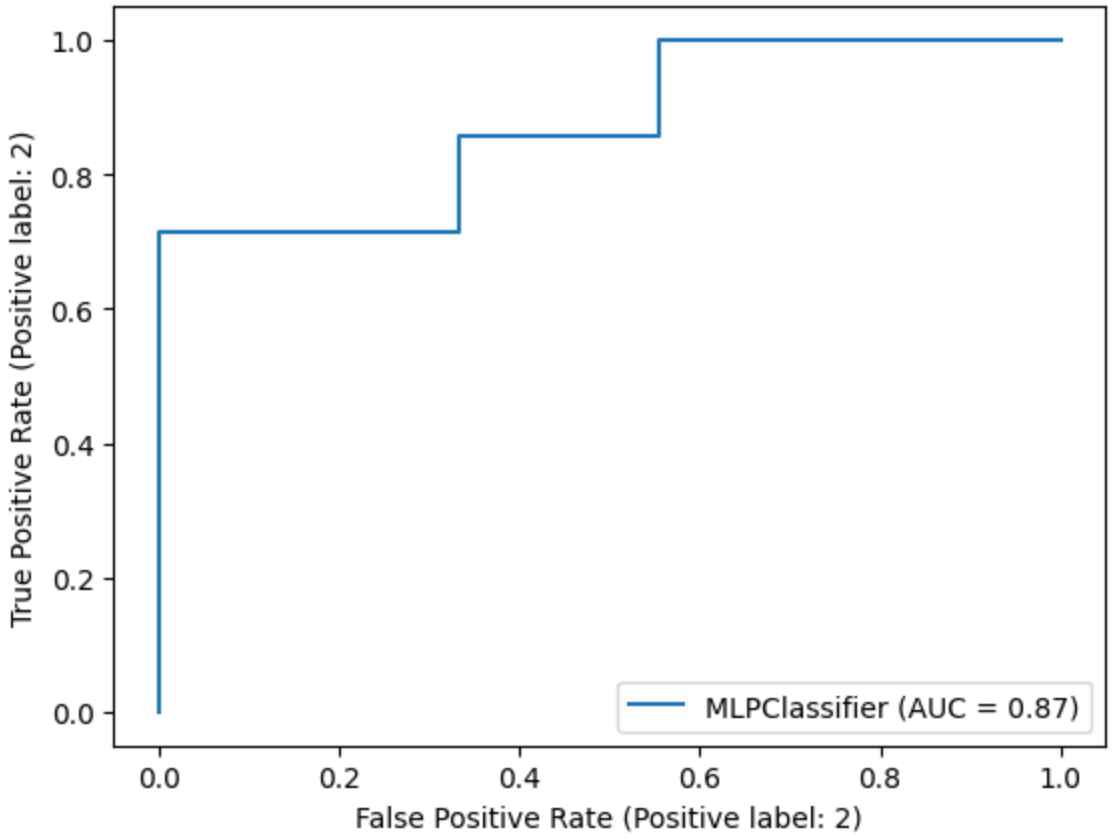
In [182]:
```
print('Reporte de métricas: ')
print(Reporte)
```

```
Reporte de métricas:
              precision    recall  f1-score   support

           1       0.90      1.00      0.95        18
           2       1.00      0.71      0.83         7

    accuracy                           0.92        25
   macro avg       0.95      0.86      0.89        25
weighted avg       0.93      0.92      0.92        25
```

Curva ROC

In [183]:
```
RocCurveDisplay.from_estimator(modelo, X_test, y_test)
```

Out[183]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7df62b373820>



CASO 2 CLASIFIACIÓN MULTICLASE (digitos manuscritos)

Importación de datos

In [184]:
```
datos = pd.read_csv("digito16x16.csv", sep=',', header=0)
datos.head()
```

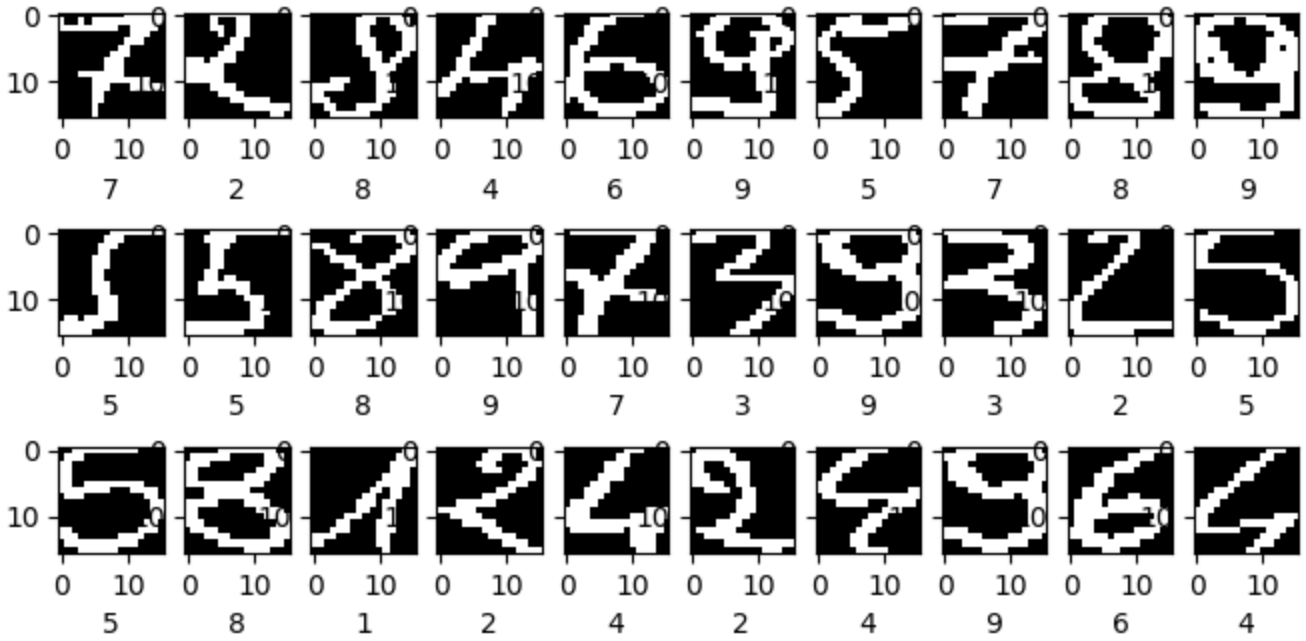
Out[184]:

	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	...	p248	p249	p250	p251	p252	p253	p254	p255	p256	d
0	0	0	0	0	0	0	1	1	1	1	...	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	1	1	1	1	1	...	1	1	1	1	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	1	1	1	1	...	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	

5 rows × 257 columns

Visialización de digitos

In [189]:
```
import random

XX = datos.drop(['digito'], axis=1)
yy = datos['digito']

ran = datos.shape[0]-1

fig, ax = plt.subplots(3, 10, figsize=(8,4))
for k in range(3):
    for j in range(10):
        azar = random.sample(range(ran),1)
        img = np.array(XX.iloc[azar].values)
        img = np.array(img).reshape(16,16)
        ax[k,j].imshow(img, cmap='gray')
        ax[k,j].set_xlabel(yy[azar[0]])
plt.show()
```



Separación de datos para test y trainig

In [190]:
```
X = datos.drop(['digito'], axis=1)
y = datos['digito']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=5, test_size=0.3)
```

Entrenamiento del modelo

In [192]:
```
# Configuración del modelo
modelo = MLPClassifier(max_iter=1000)
# Entrenamiento
modelo.fit(X_train, y_train)
# Evaluación
modelo.score(X_test, y_test)
```

Out[192]:
```
0.9163179916317992
```

Predicción del modelo

In [193]:
```
pred = modelo.predict(X_test)
pd.DataFrame([pred,y_test])
```

Out[193]:

	0	1	2	3	4	5	6	7	8	9	...	468	469	470	471	472	473	474	475	476	477
0	3	4	0	7	6	9	2	7	0	9	...	3	3	6	4	1	0	7	7	6	6
1	3	4	0	7	6	9	2	7	0	9	...	3	3	6	4	4	0	7	7	6	6

2 rows × 478 columns

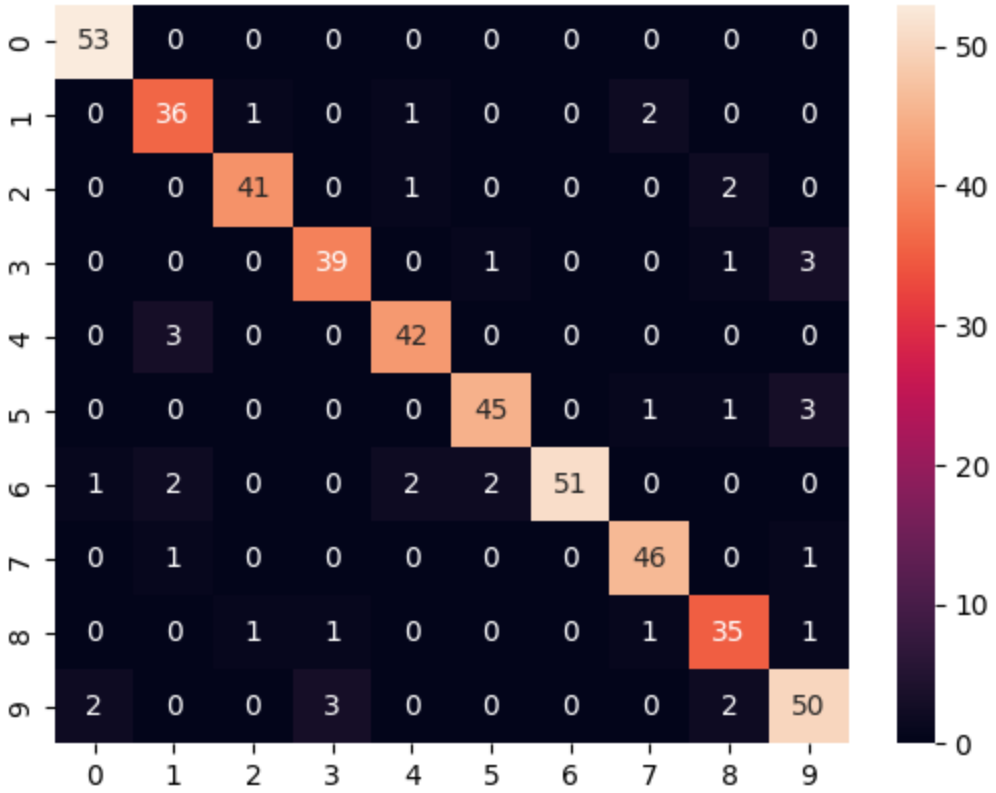
Matriz de Confusión y Exactitud

In [195]:
```
MatrizConfusion = confusion_matrix(y_test, pred)
Exactitud = metrics.accuracy_score(y_test, pred)

print('Matriz de Confusión')
print(MatrizConfusion)
print('Exactitud = ', Exactitud)
```

```
Matriz de Confusión
[[53  0  0  0  0  0  0  0  0  0]
 [ 0 36  1  0  1  0  0  2  0  0]
 [ 0  0 41  0  1  0  0  0  2  0]
 [ 0  0  0 39  0  1  0  0  1  3]
 [ 0  3  0  0 42  0  0  0  0  0]
 [ 0  0  0  0  0 45  0  1  1  3]
 [ 1  2  0  0  2  2 51  0  0  0]
 [ 0  1  0  0  0  0  0 46  0  1]
 [ 0  0  1  1  0  0  0  1 35  1]
 [ 2  0  0  3  0  0  0  0  2 50]]
Exactitud =  0.9163179916317992
```

In [196]:
```
import seaborn as sns

categorias = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
sns.heatmap(MatrizConfusion, yticklabels=categorias, xticklabels=categorias, annot=True)
print('Exactitud = ', Exactitud)
```

```
Exactitud =  0.9163179916317992
```



TAREA 1:

Obtenga las métricas: precision, sensibilidad, especificidad, puntuacionF1, y grafique la curva ROC para el modelo de clasificación multiclase (Recomendación: revisar micropromedios o macropromedios en las diapositivas del aula virtual)

5. TAREA 2

Repetir el procedimiento para los dataset: **voz.csv**, **expresión facial.csv** al final obtener las métricas: exactitud, precisión, sensibilidad, especificidad, puntuación F1 y grafiqwue la curva ROC.