MachineLearning

June 4, 2025

1 MACHINE LEARNING

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math
import warnings
import seaborn as sns
warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

1.1 CARGA DE DATOS Y PRE-PROCESAMIENTO

[2]:
```
df_original = pd.read_excel(
    'bbdd/Base Datos con categorias agrupadas.xlsx',
    header=1,
    decimal=',',
    )
df = df_original.copy()
num_var = ['Edad',
           'IMC',
           'Htopre',
           'Leucograma',
           'Creatininapre',
           'Proteinasatotales',
           'PCR',
           'TestFuerzaAgarre',
           'Test5metros',
           'Estanciahospitalariacalculada']
cat_var = ['Género',
```

```python
            'DMdicotomica',
            'TabaquismoDicot',
            'HTA',
            'DL',
            'Obesidad',
            'EPOCdicot',
            'Insuficienciarenalpreoperatoria',
            'ComplicacionesTODAS',
            'ComplicacionesMACE',
            'FragilidadGrST',
            'FragilidadGST',
            'Barthel',
            'Katz',
            'Frail',
            'Edmonton'
            ]
cat_binary_var = ['Género',
                  'DMdicotomica',
                  'TabaquismoDicot',
                  'HTA',
                  'DL',
                  'Obesidad',
                  'EPOCdicot',
                  'Insuficienciarenalpreoperatoria',
                  'ComplicacionesTODAS',
                  'ComplicacionesMACE',
                  'FragilidadGrST',
                  'FragilidadGST',
                  ]
cat_points_var = ['Barthel',
                  'Katz',
                  'Frail',
                  'Edmonton'
                  ]
# Definir la relación entre valores numéricos y categorías
cat_classes = {
    "Género": {0: "Hombre", 1: "Mujer"},
    "DMdicotomica": {0: "No", 1: "Sí"},
    "TabaquismoDicot": {0: "No", 1: "Sí"},
    "HTA": {0: "No", 1: "Sí"},
    "DL": {0: "No", 1: "Sí"},
    "Obesidad": {0: "No", 1: "Sí"},
    "EPOCdicot": {0: "No", 1: "Sí"},
    "Insuficienciarenalpreoperatoria": {0: "No", 1: "Sí"},
    "ComplicacionesTODAS": {0: "No", 1: "Sí"},
    "ComplicacionesMACE": {0: "No", 1: "Sí"},
    "FragilidadGrST": {0: "No", 1: "Sí"},
```

```
    "FragilidadGST": {0: "No", 1: "Sí"}
}

X_num = df[num_var]
X_num = X_num.apply(pd.to_numeric, errors='coerce')
X_cat = df[cat_var]
X_cat = X_cat.apply(pd.to_numeric, errors='coerce')
X_cat_binary = X_cat[cat_binary_var]
X_cat_points = X_cat[cat_points_var]
X = pd.concat([X_num, X_cat], axis=1)
y = df['Mortalidad30']
```

1.2 SELECCIÓN DE CARACTERÍSTICAS

[3]:
```
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelBinarizer
from sklearn.preprocessing import OrdinalEncoder


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')) , # Primero rellenamos␣
↪los datos
    ('scalar', StandardScaler())  # Luego normalizamos
])

categorical_binary_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(sparse_output=False, handle_unknown='ignore'))  #␣
↪Luego codificamos
])

categorical_points_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    #('onehot', OneHotEncoder(sparse_output=False, handle_unknown='ignore'))  #␣
↪Luego codificamos
    ('ordinal', OrdinalEncoder(handle_unknown='use_encoded_value',␣
↪unknown_value=-1))  # Luego codificamos
])

preprocessor = ColumnTransformer(
```

```
    transformers=[
        ('num', numerical_transformer, X_num.columns),
        ('cat_bin', categorical_binary_transformer, X_cat_binary.columns),
        ('cat_points', categorical_points_transformer, X_cat_points.columns),
    ]
)
# Entrenar un premodelo
premodel = RandomForestClassifier(n_estimators=100,
                                  random_state=42,
                                  class_weight='balanced')

pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', premodel)
])

pipeline.fit(X_train, y_train)
```

[3]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('scalar',
StandardScaler())]),
                                                  Index(['Edad', 'IMC',
'Htopre', 'Leucograma', 'Creatininapre',
       'Proteinasatotales', 'PCR', 'TestFuerzaAgarre', 'Test5metros',
       'Estanciahospitalariacalculada'],
      dtype='object')),
                                                 ('cat_bin',
                                                  Pip…
       'ComplicacionesMACE', 'FragilidadGrST', 'FragilidadGST'],
      dtype='object')),
                                                 ('cat_points',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('ordinal',
OrdinalEncoder(handle_unknown='use_encoded_value',
  unknown_value=-1))]),
                                                  Index(['Barthel', 'Katz',
'Frail', 'Edmonton'], dtype='object'))])),
                ('classifier',
                 RandomForestClassifier(class_weight='balanced',
                                        random_state=42))])
```

[4]:
```
# Visualiación del árbol
from sklearn.tree import export_graphviz
```

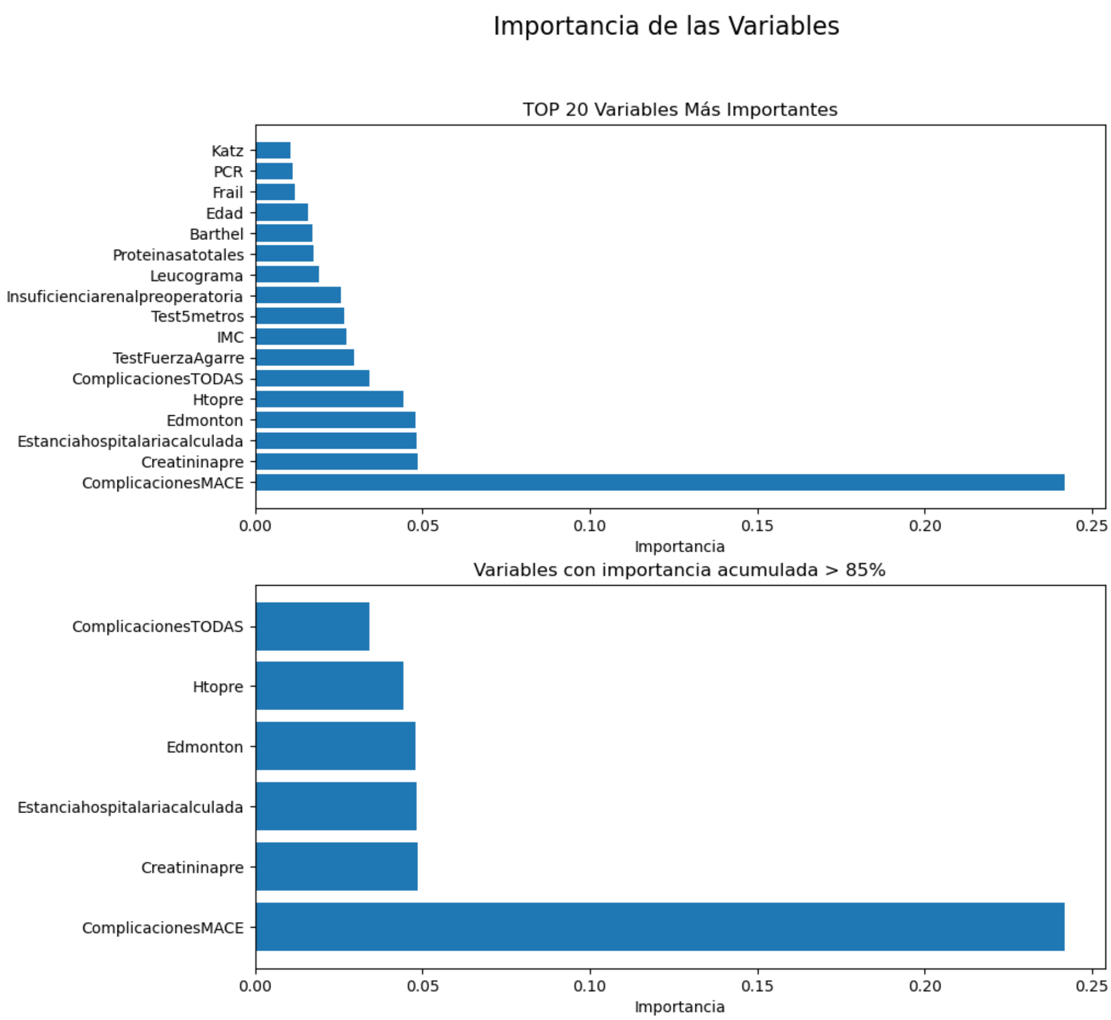
```python
from IPython.display import Image
import graphviz

rf_model = pipeline.named_steps['classifier']
feature_names = preprocessor.get_feature_names_out()

for i in range (1):
    tree = rf_model.estimators_[i]
    dot_data = export_graphviz(tree, out_file=None,
                               feature_names=feature_names,
                               class_names=['No', 'Sí'],
                               filled=True, rounded=True,
                               special_characters=True)
    graph = graphviz.Source(dot_data)
    graph.format = 'png'
    graph.render(f'arbol_{i}')
    graph.view()
```

[5]:

```python
# Extraer importancias
feature_names = preprocessor.get_feature_names_out()
#Unificar las clases de las variables categóricas
feature_names = [name.split('__')[1] if '__' in name else name for name in␣
 ↪feature_names]
feature_names = [cat_classes.get(name, name) for name in feature_names]
feature_names = [name[0] if isinstance(name, tuple) else name for name in␣
 ↪feature_names]
feature_names = [name.replace('num__', '') for name in feature_names]
feature_names = [name.replace('cat__', '') for name in feature_names]
feature_names = [name.replace(' ', '_') for name in feature_names]
feature_names = [name.replace('1_', '') for name in feature_names]
feature_names = [name.replace('0_', '') for name in feature_names]
feature_names = [name.replace('1', '') for name in feature_names]
feature_names = [name.replace('0', '') for name in feature_names]
feature_names = [name.replace('num_', '') for name in feature_names]
feature_names = [name.replace('cat_', '') for name in feature_names]
feature_names = [name.replace('__', '') for name in feature_names]
feature_names = [name.replace('_', '') for name in feature_names]

feature_importance = pd.DataFrame({'feature': feature_names, 'importance':␣
 ↪premodel.feature_importances_})
feature_importance = feature_importance.sort_values(by='importance',␣
 ↪ascending=False)


# Graficar la importancia de las variables
fig, ax = plt.subplots(2, 1, figsize=(10, 10))
# Definir título de la figura
```

```
fig.suptitle(f'Importancia de las Variables', fontsize=16)

#garfico de barras d eimportancia en ax[0[]
ax[0].barh(feature_importance['feature'][:20],␣
 ↪feature_importance['importance'][:20])
ax[0].set_xlabel('Importancia')
ax[0].set_title('TOP 20 Variables Más Importantes')

ax[1].barh(feature_importance['feature'][:int(len(feature_importance)*0.2)],␣
 ↪feature_importance['importance'][:int(len(feature_importance)*0.2)])
ax[1].set_xlabel('Importancia')
ax[1].set_title('Variables con importancia acumulada > 85%')
```

[5]: Text(0.5, 1.0, 'Variables con importancia acumulada > 85%')



[ ]:
```
ax[0].set_title('TOP 20 Feature Importance')
ax[1].set_xlabel('Importance')
fig.show()
```

[6]:
```
selected_features = feature_importance['feature'][:
↪int(len(feature_importance)*0.2)].tolist()
print(selected_features)
```

```
['ComplicacionesMACE', 'ComplicacionesMACE', 'Creatininapre',
'Estanciahospitalariacalculada', 'Edmonton', 'Htopre', 'ComplicacionesTODAS']
```

1.3 SELECCIÓN DEL MODELO

[7]:
```
from sklearn.model_selection import KFold

# Filtrar las columnas de X_num, X_cat_binary y X_cat_points según␣
↪selected_features
X_num = X_num[[col for col in X_num.columns if col in selected_features]]
X_cat_binary = X_cat_binary[[col for col in X_cat_binary.columns if col in␣
↪selected_features]]
X_cat_points = X_cat_points[[col for col in X_cat_points.columns if col in␣
↪selected_features]]
X = pd.concat([X_num, X_cat_binary, X_cat_points], axis=1)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
# Definir el número de splits de KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)  # 5-fold Cross Validation

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, X_num.columns),
        ('cat_bin', categorical_binary_transformer, X_cat_binary.columns),
        ('cat_points', categorical_points_transformer, X_cat_points.columns),
    ]
)
```

1.3.1 Funciones Auxiliares

[8]:
```
# Función evaluate_model
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import numpy as np
```

```python
from sklearn.metrics import roc_auc_score
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import roc_curve, auc, precision_recall_curve

def evaluate_model(pipeline:Pipeline):
    # Predicciones
    y_pred = pipeline.predict(X_test)
    y_pred_proba = pipeline.predict_proba(X_test)[:, 1]

    # Matriz de confusión
    cm = confusion_matrix(y_test, y_pred)

    # ROC Calculation
    fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
    roc_auc = auc(fpr, tpr)

    # Precision-Recall Calculation
    precision, recall, thresholds = precision_recall_curve(y_test, y_pred_proba)
    #pr_auc = auc(recall, precision)
    pr_auc = average_precision_score(y_test, y_pred_proba)


    #Graficar las metricas de evaluación
    fig, ax = plt.subplots(1, 3, figsize=(20, 6))

    #Matriz de consuson en ax1
    ax[0].set_title('Matriz de Confusión')
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=['No', 'Sí'], yticklabels=['No', 'Sí'], ax=ax[0])
    ax[0].set_xlabel('False Positive Rate')
    ax[0].set_ylabel('True Positive Rate')
    ax[0].set_title('Confusion Matrix')
    ax[0].legend()

    #Curva ROC en ax2
    ax[1].plot(fpr, tpr, color='blue', lw=2, label='AUC = %0.2f' % roc_auc)
    ax[1].plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
    ax[1].set_xlim([0.0, 1.0])
    ax[1].set_ylim([0.0, 1.05])
    ax[1].set_xlabel('False Positive Rate')
    ax[1].set_ylabel('True Positive Rate')
    ax[1].set_title('Receiver Operating Characteristic (ROC) Curve')
    ax[1].set_title(f'ROC Curve')
    ax[1].legend()

    #Precision recall en ax3
```

```python
    ax[2].plot(recall, precision, color='blue', lw=2, label='AUC = %0.2f' %␣
↪pr_auc)
    ax[2].set_xlabel('Recall')
    ax[2].set_ylabel('Precision')
    ax[2].set_title('Precision-Recall Curve')
    ax[2].legend()

    plt.tight_layout()
    plt.show()

    #Calcular F1-score
    from sklearn.metrics import f1_score
    f1 = f1_score(y_test, y_pred)
    print(f'F1 Score: {f1:.2f}')
    #Calcular el AUC
    from sklearn.metrics import roc_auc_score
    roc_auc = roc_auc_score(y_test, y_pred_proba)
    print(f'AUC: {roc_auc:.2f}')
    #Calcular el accuracy
    from sklearn.metrics import accuracy_score
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Accuracy: {accuracy:.2f}')
    #Calcular el recall
    from sklearn.metrics import recall_score
    recall = recall_score(y_test, y_pred)
    print(f'Recall: {recall:.2f}')
    #Calcular el precision
    from sklearn.metrics import precision_score
    precision = precision_score(y_test, y_pred)
    print(f'Precision: {precision:.2f}')
    #Calcular el MCC
    from sklearn.metrics import matthews_corrcoef
    mcc = matthews_corrcoef(y_test, y_pred)
    print(f'MCC: {mcc:.2f}')
    #Calcular el F-beta score
    from sklearn.metrics import fbeta_score
    beta = 0.5
    f_beta = fbeta_score(y_test, y_pred, beta=beta)
    print(f'F-beta Score: {f_beta:.2f}')
    #Calcular el ROC AUC
    from sklearn.metrics import roc_auc_score
    roc_auc = roc_auc_score(y_test, y_pred_proba)
    print(f'ROC AUC: {roc_auc:.2f}')
    #Calcular el PR AUC
    from sklearn.metrics import average_precision_score
    pr_auc = average_precision_score(y_test, y_pred_proba)
    print(f'PR AUC: {pr_auc:.2f}')
```

```python
    #Calcular el Brier score
    from sklearn.metrics import brier_score_loss
    brier_score = brier_score_loss(y_test, y_pred_proba)
    print(f'Brier Score: {brier_score:.2f}')
    #Calcular el log loss
    from sklearn.metrics import log_loss
    log_loss_value = log_loss(y_test, y_pred_proba)
    print(f'Log Loss: {log_loss_value:.2f}')
    #Calcular el Hamming loss
    from sklearn.metrics import hamming_loss
    hamming_loss_value = hamming_loss(y_test, y_pred)
    print(f'Hamming Loss: {hamming_loss_value:.2f}')
```

[9]:
```python
# Función evaluate_model_kfolds
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪roc_auc_score, f1_score, roc_curve, auc
from sklearn.metrics import precision_recall_curve, average_precision_score,␣
 ↪brier_score_loss, log_loss, hamming_loss
from sklearn.metrics import precision_score, recall_score, accuracy_score,␣
 ↪matthews_corrcoef, fbeta_score
from sklearn.model_selection import StratifiedKFold
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

def evaluate_model_kfolds(pipeline: ImbPipeline, X_train, y_train, kf):
    # Variables para almacenar las métricas promediadas
    f1_scores = []
    roc_aucs = []
    pr_aucs = []
    accuracies = []
    recalls = []
    precisions = []
    mccs = []
    f_beta_scores = []
    brier_scores = []
    log_losses = []
    hamming_losses = []
    # Variables para almacenar las métricas promediadas
    results = {
        "cm": [], "fpr": [], "tpr": [], "recall": [], "precision": [], "f1": []
    }
```

```python
    # Listas para acumular las predicciones y etiquetas reales para la matriz␣
↪de confusión y las curvas
    y_true_all = []
    y_pred_all = []
    y_pred_proba_all = []

    # KFold Cross Validation
    for train_index, test_index in kf.split(X_train, y_train):
        # Dividir el conjunto de datos en entrenamiento y prueba según el índice
        X_train_fold, X_test_fold = X_train.iloc[train_index], X_train.
↪iloc[test_index]
        y_train_fold, y_test_fold = y_train.iloc[train_index], y_train.
↪iloc[test_index]

        # Ajustar el pipeline
        pipeline.fit(X_train_fold, y_train_fold)

        # Predecir en el conjunto de prueba
        y_pred = pipeline.predict(X_test_fold)
        y_pred_proba = pipeline.predict_proba(X_test_fold)[:, 1]  # Obtener las␣
↪probabilidades

        # Si alguna de las clases está ausente, saltar ese fold
        if len(np.unique(y_test_fold)) == 1:
            print("Skipping fold with only one class in y_test_fold.")
            continue

        # Acumular las predicciones y etiquetas reales
        y_true_all.extend(y_test_fold)
        y_pred_all.extend(y_pred)
        y_pred_proba_all.extend(y_pred_proba)

        # Calcular las métricas de evaluación para cada fold
        f1 = f1_score(y_test_fold, y_pred)
        f1_scores.append(f1)

        accuracy = accuracy_score(y_test_fold, y_pred)
        accuracies.append(accuracy)

        recall = recall_score(y_test_fold, y_pred)
        recalls.append(recall)

        precision = precision_score(y_test_fold, y_pred)
        precisions.append(precision)

        mcc = matthews_corrcoef(y_test_fold, y_pred)
        mccs.append(mcc)
```

```
        f_beta = fbeta_score(y_test_fold, y_pred, beta=0.5)
        f_beta_scores.append(f_beta)

        # Calcular Brier Score y Log Loss
        brier_score = brier_score_loss(y_test_fold, y_pred_proba)
        brier_scores.append(brier_score)

        try:
            log_loss_value = log_loss(y_test_fold, y_pred_proba)
            log_losses.append(log_loss_value)
        except ValueError:
            print("Skipping log_loss due to issue with y_test_fold in fold.")
            continue

        hamming_loss_value = hamming_loss(y_test_fold, y_pred)
        hamming_losses.append(hamming_loss_value)

        # Calcular el AUC y PR AUC para cada fold
        roc_auc = roc_auc_score(y_test_fold, y_pred_proba)
        roc_aucs.append(roc_auc)

        pr_auc = average_precision_score(y_test_fold, y_pred_proba)
        pr_aucs.append(pr_auc)

    # Promediar las métricas
    print(f"F1 Score: {np.mean(f1_scores):.3f} ± {np.std(f1_scores):.3f}")
    print(f"ROC AUC: {np.mean(roc_aucs):.3f} ± {np.std(roc_aucs):.3f}")
    print(f"PR AUC: {pr_auc:.3f} ± {np.std(pr_aucs):.3f}")
    print(f"Accuracy: {np.mean(accuracies):.3f} ± {np.std(accuracies):.3f}")
    print(f"Recall: {np.mean(recalls):.3f} ± {np.std(recalls):.3f}")
    print(f"Precision: {np.mean(precisions):.3f} ± {np.std(precisions):.3f}")
    print(f"MCC: {np.mean(mccs):.3f} ± {np.std(mccs):.2f}")
    print(f"F-beta Score: {np.mean(f_beta_scores):.3f} ± {np.std(f_beta_scores):
↪.3f}")
    print(f"Brier Score: {np.mean(brier_scores):.3f} ± {np.std(brier_scores):.
↪3f}")
    print(f"Log Loss: {np.mean(log_losses):.3f} ± {np.std(log_losses):.3f}")
    print(f"Hamming Loss: {np.mean(hamming_losses):.3f} ± {np.
↪std(hamming_losses):.3f}")

    # Graficar las métricas de evaluación utilizando los datos combinados

    # Calcular la curva ROC y Precision-Recall utilizando los datos acumulados
    fpr, tpr, _ = roc_curve(y_true_all, y_pred_proba_all)
    roc_auc = auc(fpr, tpr)
```

```
    precision, recall, _ = precision_recall_curve(y_true_all, y_pred_proba_all)
    #pr_auc = auc(recall, precision)
    #pr_auc = average_precision_score(y_true_all, y_pred_proba_all)


    fig, ax = plt.subplots(1, 3, figsize=(20, 6))

    # Matriz de Confusión utilizando todos los datos
    cm = confusion_matrix(y_true_all, y_pred_all)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
                xticklabels=['No', 'Sí'], yticklabels=['No', 'Sí'], ax=ax[0])
    ax[0].set_xlabel('Predicted')
    ax[0].set_ylabel('True')
    ax[0].set_title('Confusion Matrix (Overall)')

    # Curva ROC utilizando todos los datos
    ax[1].plot(fpr, tpr, color='blue', lw=2, label='AUC = %0.2f' % roc_auc)
    ax[1].plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
    ax[1].set_xlim([0.0, 1.0])
    ax[1].set_ylim([0.0, 1.05])
    ax[1].set_xlabel('False Positive Rate')
    ax[1].set_ylabel('True Positive Rate')
    ax[1].set_title('ROC Curve')
    ax[1].legend()

    # Precision-Recall Curve utilizando todos los datos

    ax[2].set_xlim([0.0, 1.0])
    ax[2].set_ylim([0.0, 1.05])
    ax[2].plot(recall, precision, color='blue', lw=2, label='AUC = %0.2f' %␣
↪pr_auc)
    ax[2].set_xlabel('Recall')
    ax[2].set_ylabel('Precision')
    ax[2].set_title('Precision-Recall Curve')
    ax[2].legend()

    plt.tight_layout()
    plt.show()

    #Return
    results['roc-auc']= roc_auc
    results['pr-auc']=pr_auc
    results['cm'] = cm
    results['fpr'] = fpr
    results['tpr'] = tpr
    results['recall'] = recall
    results['precision'] = precision
```

```
    results['f1'] = np.mean(f1_scores)
    results['mcc'] = np.mean(mcc)
    results['f-beta'] = np.mean(f_beta_scores)
    results['brier'] = np.mean(brier_scores)
    results['log_loss'] = np.mean(log_losses)
    results['hamming_loss'] = np.mean(hamming_losses)
    results ['recall_score'] = np.mean(recalls)
    results ['precision_score'] = np.mean(precisions)
    return results
```

1.3.2 SVM

[10]:
```
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import KFold, cross_val_score
from sklearn.svm import SVC

#MODELO SVM
model = SVC(probability=True, random_state=42)

# Crear el pipeline estándar de scikit-learn
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('regressor', model)
])

# Evaluar el modelo usando KFold
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
 ↪scoring='accuracy')

# Entrenamiento del pipeline
pipeline.fit(X_train, y_train)
```

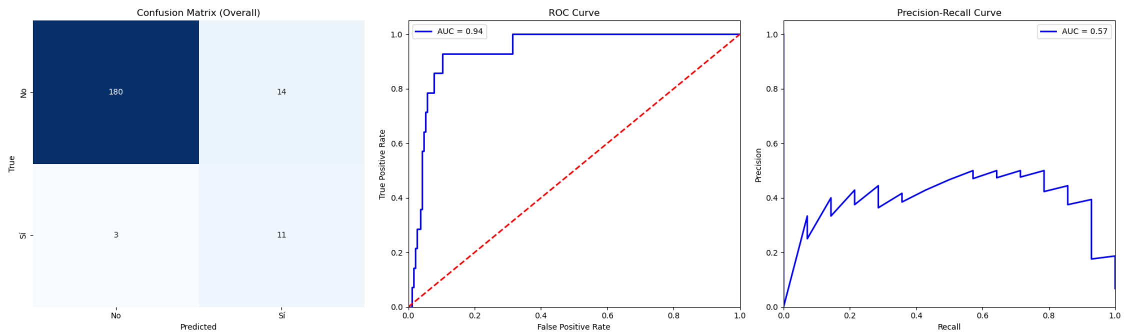
[10]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('scalar',
StandardScaler())]),
                                                  Index(['Htopre',
'Creatininapre', 'Estanciahospitalariacalculada'], dtype='object')),
                                                 ('cat_bin',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
```

```
                                                              OneHotEnco…
                                                   Index(['ComplicacionesTODAS',
'ComplicacionesMACE'], dtype='object')),
                                                  ('cat_points',
                                                   Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('ordinal',
OrdinalEncoder(handle_unknown='use_encoded_value',
  unknown_value=-1))]),
                                                   Index(['Edmonton'],
dtype='object'))])),
                ('smote', SMOTE(random_state=42)),
                ('regressor', SVC(probability=True, random_state=42))])
```

[11]:

```python
svc_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.523 ± 0.145
ROC AUC: 0.949 ± 0.015
PR AUC: 0.569 ± 0.148
Accuracy: 0.918 ± 0.008
Recall: 0.817 ± 0.185
Precision: 0.422 ± 0.163
MCC: 0.532 ± 0.12
F-beta Score: 0.455 ± 0.157
Brier Score: 0.057 ± 0.010
Log Loss: 0.222 ± 0.090
Hamming Loss: 0.082 ± 0.008
```



1.3.3 XGBOOST

[12]:

```python
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import KFold, cross_val_score
```

```
# Modelo XGBoost
model = xgb.XGBClassifier( )

# Crear el pipeline estándar de scikit-learn
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('regressor', model)
])

# Evaluar el modelo usando KFold
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
  ↪scoring='accuracy')

# Entrenamiento del pipeline
pipeline.fit(X_train, y_train)
```

[12]:

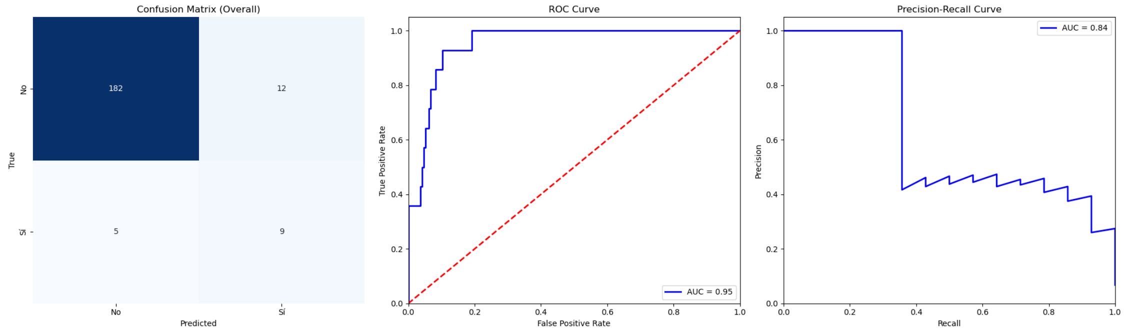
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('scalar',
StandardScaler())]),
                                                  Index(['Htopre',
'Creatininapre', 'Estanciahospitalariacalculada'], dtype='object')),
                                                 ('cat_bin',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
                                                                   OneHotEnco…
                               feature_types=None, feature_weights=None,
                               gamma=None, grow_policy=None,
                               importance_type=None,
                               interaction_constraints=None, learning_rate=None,
                               max_bin=None, max_cat_threshold=None,
                               max_cat_to_onehot=None, max_delta_step=None,
                               max_depth=None, max_leaves=None,
                               min_child_weight=None, missing=nan,
                               monotone_constraints=None, multi_strategy=None,
                               n_estimators=None, n_jobs=None,
                               num_parallel_tree=None, …))])
```

[13]:

```
xgboost_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.501 ± 0.070
ROC AUC: 0.956 ± 0.035
```

```
PR AUC: 0.844 ± 0.260
Accuracy: 0.918 ± 0.025
Recall: 0.717 ± 0.218
Precision: 0.417 ± 0.083
MCC: 0.493 ± 0.08
F-beta Score: 0.443 ± 0.069
Brier Score: 0.059 ± 0.018
Log Loss: 0.204 ± 0.066
Hamming Loss: 0.082 ± 0.025
```



1.3.4 RANDOM FOREST

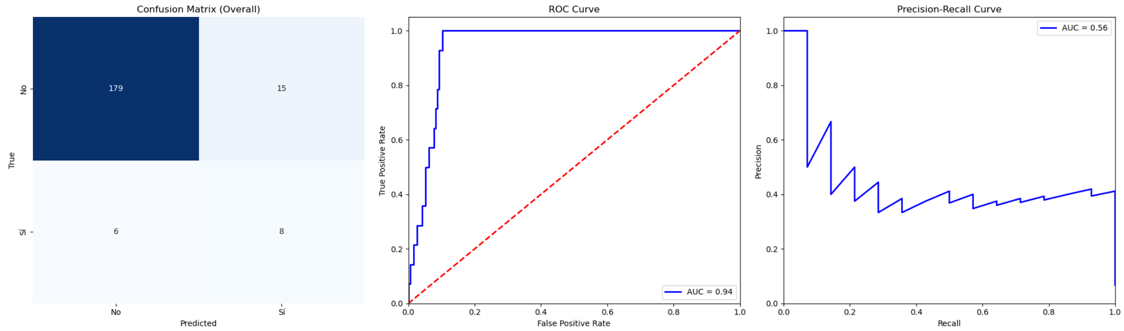
[14]:
```
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import KFold, cross_val_score

# Modelo Random Forest
model = RandomForestClassifier(n_estimators=100,
                               random_state=42,
                               class_weight='balanced',
                               max_depth=5,
                               min_samples_split=2,
                               min_samples_leaf=1,
                               max_features='sqrt',
                               bootstrap=True)

# Crear el pipeline estándar de scikit-learn
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('regressor', model)
])

# Evaluar el modelo usando KFold
```

```
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
↪scoring='accuracy')

# Entrenamiento del pipeline
pipeline.fit(X_train, y_train)
```

[14]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('scalar',
StandardScaler())]),
                                                  Index(['Htopre',
'Creatininapre', 'Estanciahospitalariacalculada'], dtype='object')),
                                                 ('cat_bin',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
                                                                   OneHotEnco…
                                                  Index(['ComplicacionesTODAS',
'ComplicacionesMACE'], dtype='object')),
                                                 ('cat_points',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('ordinal',
OrdinalEncoder(handle_unknown='use_encoded_value',
  unknown_value=-1))]),
                                                  Index(['Edmonton'],
dtype='object'))])),
                ('smote', SMOTE(random_state=42)),
                ('regressor',
                 RandomForestClassifier(class_weight='balanced', max_depth=5,
                                        random_state=42))])
```

[15]:
```
rf_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.425 ± 0.043
ROC AUC: 0.953 ± 0.030
PR AUC: 0.564 ± 0.240
Accuracy: 0.899 ± 0.025
Recall: 0.667 ± 0.216
Precision: 0.341 ± 0.075
MCC: 0.413 ± 0.06
F-beta Score: 0.368 ± 0.063
Brier Score: 0.061 ± 0.011
Log Loss: 0.178 ± 0.031
Hamming Loss: 0.101 ± 0.025
```



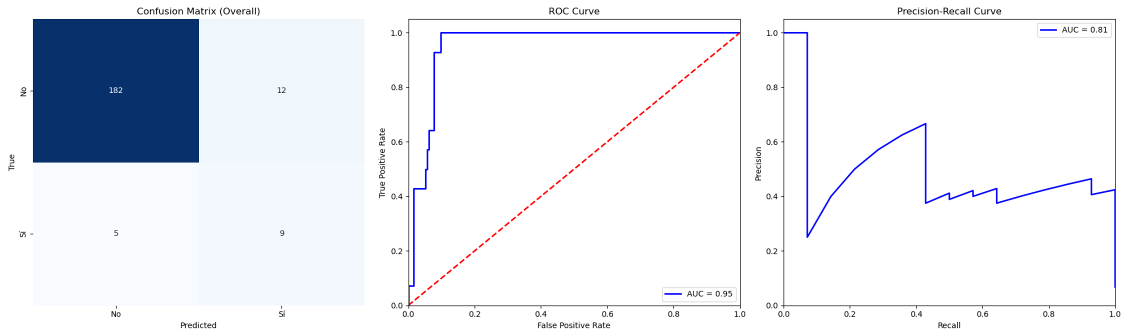
1.3.5 CATBOOST

[16]:
```
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import KFold, cross_val_score
from catboost import CatBoostClassifier

# Modelo Random Forest

model = CatBoostClassifier(iterations=1000,
                           learning_rate=0.1,
                           depth=6,
                           l2_leaf_reg=3,
                           loss_function='Logloss',
                           eval_metric='AUC',
                           random_seed=42,
                           verbose=0)

# Crear el pipeline estándar de scikit-learn
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('regressor', model)
])

# Evaluar el modelo usando KFold
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
 ↪scoring='accuracy')

# Entrenamiento del pipeline
pipeline.fit(X_train, y_train)
```

[16]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('scalar',
StandardScaler())]),
                                                  Index(['Htopre',
'Creatininapre', 'Estanciahospitalariacalculada'], dtype='object')),
                                                 ('cat_bin',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
                                                                   OneHotEnco…
                                                  Index(['ComplicacionesTODAS',
'ComplicacionesMACE'], dtype='object')),
                                                 ('cat_points',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('ordinal',
OrdinalEncoder(handle_unknown='use_encoded_value',
  unknown_value=-1))]),
                                                  Index(['Edmonton'],
dtype='object'))])),
                ('smote', SMOTE(random_state=42)),
                ('regressor',
                 <catboost.core.CatBoostClassifier object at 0x3091e9c90>)])
```

[17]:
```
cb_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.497 ± 0.079
ROC AUC: 0.965 ± 0.024
PR AUC: 0.811 ± 0.223
Accuracy: 0.918 ± 0.016
Recall: 0.717 ± 0.166
Precision: 0.421 ± 0.137
MCC: 0.491 ± 0.05
F-beta Score: 0.445 ± 0.120
Brier Score: 0.063 ± 0.015
Log Loss: 0.248 ± 0.052
Hamming Loss: 0.082 ± 0.016
```



1.3.6 KNN

[18]:
```
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import KFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=5,
                      weights='uniform',
                      algorithm='auto',
                      leaf_size=30,
                      metric='minkowski',
                      p=2,
                      metric_params=None,
                      n_jobs=-1)

# Crear el pipeline estándar de scikit-learn
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('regressor', model)
])

# Evaluar el modelo usando KFold
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
 ↪scoring='accuracy')

# Entrenamiento del pipeline
pipeline.fit(X_train, y_train)
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas requires version
'1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

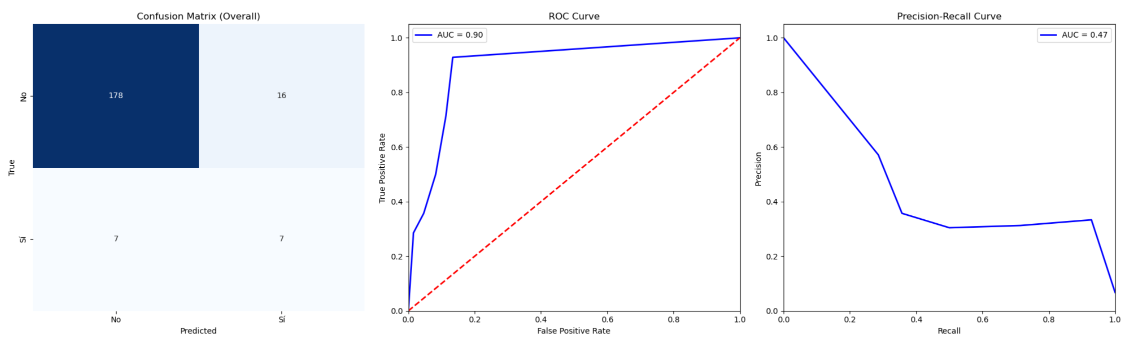
[18]:

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('scalar',
StandardScaler())]),
                                                  Index(['Htopre',
'Creatininapre', 'Estanciahospitalariacalculada'], dtype='object')),
                                                 ('cat_bin',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
                                                                   OneHotEnco…
                                                  Index(['ComplicacionesTODAS',
'ComplicacionesMACE'], dtype='object')),
                                                 ('cat_points',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
```

```
                                                                  ('ordinal',
OrdinalEncoder(handle_unknown='use_encoded_value',
  unknown_value=-1))]),
                                                  Index(['Edmonton'],
dtype='object'))])),
                ('smote', SMOTE(random_state=42)),
                ('regressor', KNeighborsClassifier(n_jobs=-1))])
```

[19]:
```
knn_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.371 ± 0.089
ROC AUC: 0.909 ± 0.060
PR AUC: 0.474 ± 0.195
Accuracy: 0.889 ± 0.032
Recall: 0.583 ± 0.260
Precision: 0.298 ± 0.095
MCC: 0.348 ± 0.12
F-beta Score: 0.321 ± 0.090
Brier Score: 0.071 ± 0.015
Log Loss: 0.832 ± 0.461
Hamming Loss: 0.111 ± 0.032
```



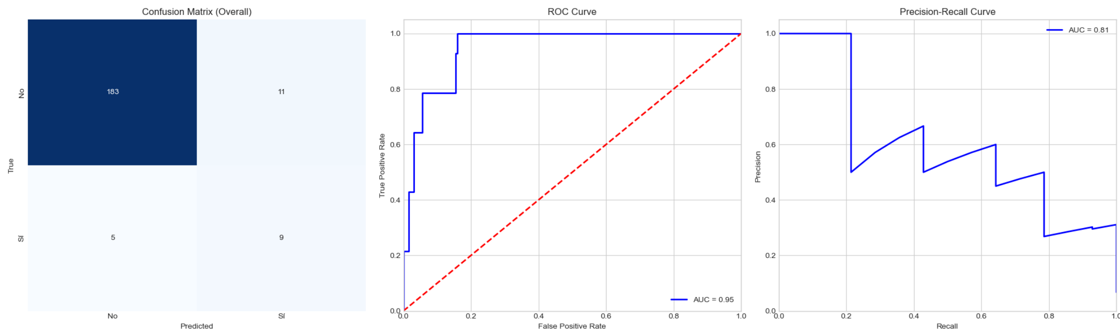
1.3.7 MLP

[ ]:
```
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import KFold, cross_val_score
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(hidden_layer_sizes=(100, 50),
                        activation='relu',
                        solver='adam',
                        alpha=0.0001,
```

```
                              batch_size='auto',
                              learning_rate='constant',
                              learning_rate_init=0.001,
                              power_t=0.5,
                              max_iter=200,
                              shuffle=True,
                              random_state=42,
                              tol=0.0001,
                              verbose=False,
                              warm_start=False,
                              momentum=0.9,
                              nesterovs_momentum=True,
                              early_stopping=False,
                              validation_fraction=0.1,
                              beta_1=0.9,
                              beta_2=0.999,
                              epsilon=1e-08)

# Crear el pipeline estándar de scikit-learn
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('regressor', model)
])

# Evaluar el modelo usando KFold
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
 ↪scoring='accuracy')

# Entrenamiento del pipeline
pipeline.fit(X_train, y_train)
```

[ ]:
```
mlp_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.491 ± 0.203
ROC AUC: 0.949 ± 0.055
PR AUC: 0.810 ± 0.259
Accuracy: 0.923 ± 0.024
Recall: 0.683 ± 0.247
Precision: 0.450 ± 0.234
MCC: 0.488 ± 0.18
F-beta Score: 0.463 ± 0.225
Brier Score: 0.055 ± 0.012
Log Loss: 0.286 ± 0.117
Hamming Loss: 0.077 ± 0.024
```



[ ]:
```python
import joblib
joblib.dump(pipeline, 'model.joblib')

#open model
model = joblib.load('model.joblib')
#show model variables
print(model)
```

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('scalar',
StandardScaler())]),
                                                  Index(['Htopre',
'Creatininapre', 'Estanciahospitalariacalculada'], dtype='object')),
                                                 ('cat_bin',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
                                                                   OneHotEnco…
                                                  Index(['ComplicacionesTODAS',
'ComplicacionesMACE'], dtype='object')),
                                                 ('cat_points',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('ordinal',
OrdinalEncoder(handle_unknown='use_encoded_value',
  unknown_value=-1))]),
                                                  Index(['Edmonton'],
dtype='object'))])),
                ('smote', SMOTE(random_state=42)),
                ('regressor',
                 MLPClassifier(hidden_layer_sizes=(100, 50), random_state=42))])
```

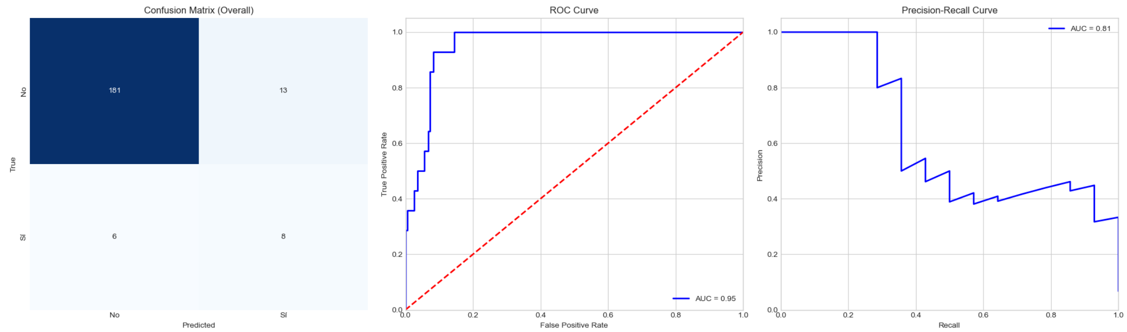
1.3.8 XGB + CATBOOST

```
[ ]: from sklearn.ensemble import StackingClassifier, VotingClassifier
     from sklearn.linear_model import LogisticRegression

     # Definir el metamodelo
     meta_model = LogisticRegression()

     # Definir los modelos
     model_xgb = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss')
     model_cb = CatBoostClassifier(iterations=1000, learning_rate=0.1, depth=6,␣
      ↪loss_function='Logloss', eval_metric='AUC', random_seed=42, verbose=0)

     # Crear un ensemble con Stacking
     ensemble_model = VotingClassifier(
         estimators=[('xgb', model_xgb), ('cat', model_cb)],
         voting='soft',
     )

     # Crear el pipeline con el ensemble
     pipeline = ImbPipeline(steps=[
         ('preprocessor', preprocessor),
         ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
         ('classifier', ensemble_model)
     ])

     # Evaluar el modelo con KFold Cross-Validation
     cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
      ↪scoring='accuracy')

     # Entrenar el pipeline
     pipeline.fit(X_train, y_train)

     # Predicciones
     y_pred = pipeline.predict(X_test)
```

```
[ ]: xbg_catboost_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.447 ± 0.035
ROC AUC: 0.965 ± 0.024
PR AUC: 0.811 ± 0.230
Accuracy: 0.909 ± 0.021
Recall: 0.667 ± 0.216
Precision: 0.378 ± 0.095
MCC: 0.438 ± 0.03
F-beta Score: 0.399 ± 0.072
Brier Score: 0.059 ± 0.015
```

```
Log Loss: 0.196 ± 0.047
Hamming Loss: 0.091 ± 0.021
```



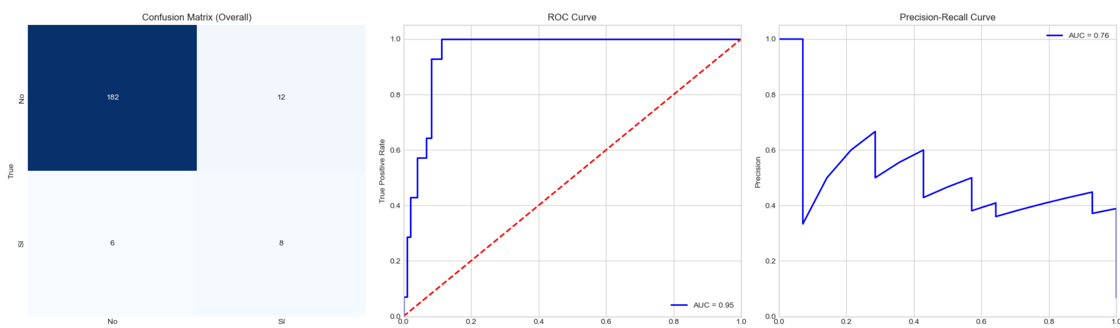
1.3.9 RF + CATBOOST

```
[ ]: from sklearn.ensemble import StackingClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression

# Definir el metamodelo
meta_model = LogisticRegression()

# Definir los modelos
model_rf = RandomForestClassifier(n_estimators=500, random_state=42)
model_cb = CatBoostClassifier(iterations=1000, learning_rate=0.1, depth=6,␣
↪loss_function='Logloss', eval_metric='AUC', random_seed=42, verbose=0)

# Crear un ensemble con Stacking
ensemble_model = VotingClassifier(
    estimators=[('RF', model_rf), ('cat', model_cb)],
    voting='soft',  # Usar soft voting
)

# Crear el pipeline con el ensemble
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('classifier', ensemble_model)
])

# Evaluar el modelo con KFold Cross-Validation
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
↪scoring='accuracy')

# Entrenar el pipeline
pipeline.fit(X_train, y_train)
```

```
# Predicciones
y_pred = pipeline.predict(X_test)
```

[ ]:
```
rf_catboost_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.459 ± 0.053
ROC AUC: 0.965 ± 0.021
PR AUC: 0.761 ± 0.196
Accuracy: 0.913 ± 0.017
Recall: 0.667 ± 0.216
Precision: 0.396 ± 0.108
MCC: 0.451 ± 0.04
F-beta Score: 0.414 ± 0.088
Brier Score: 0.055 ± 0.012
Log Loss: 0.172 ± 0.041
Hamming Loss: 0.087 ± 0.017
```

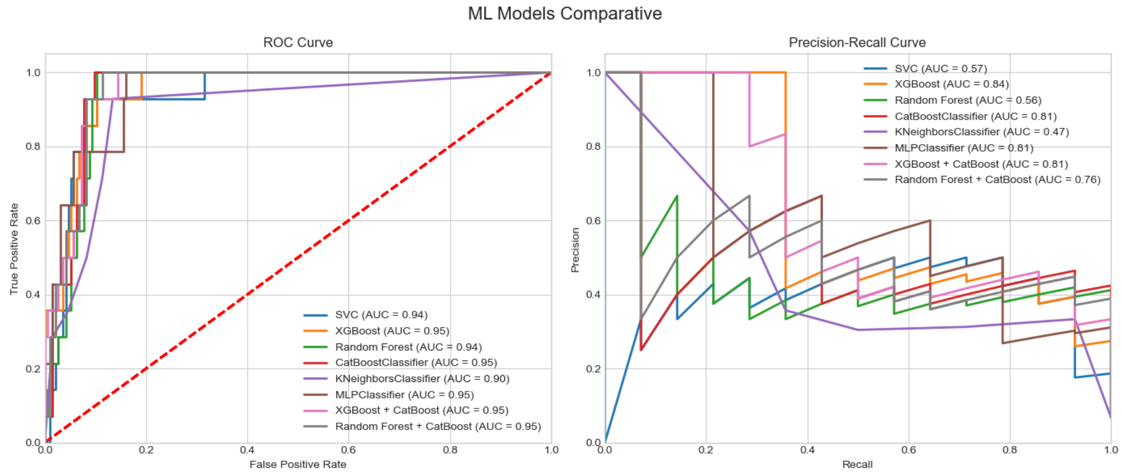


1.3.10 COMPARATIVA

[ ]:
```
# Comparativa ROC y Precision-Recall
import matplotlib.pyplot as plt
from sklearn.metrics import auc

# Crear la figura y los subgráficos
fig, ax = plt.subplots(1, 2, figsize=(14, 6))
fig.suptitle('ML Models Comparative', fontsize=16)

# Diccionario con los nombres de los modelos y sus métricas
scores = {
    'SVC': svc_scores,
    'XGBoost': xgboost_scores,
    'Random Forest': rf_scores,
    'CatBoostClassifier': cb_scores,
```

```python
    'KNeighborsClassifier': knn_scores,
    'MLPClassifier': mlp_scores,
    'XGBoost + CatBoost': xbg_catboost_scores,
    'Random Forest + CatBoost': rf_catboost_scores

}

# Graficar las curvas ROC (Izquierda)
ax[0].set_title("ROC Curve")
for model_name, metrics in scores.items():
    #print(model_name)
    mean_fpr = metrics['fpr']
    mean_tpr = metrics['tpr']
    #print(mean_tpr)
    #print(mean_fpr)
    roc_auc = auc(mean_fpr, mean_tpr)

    ax[0].plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')  # Línea␣
 ↪diagonal
    ax[0].plot(mean_fpr, mean_tpr, lw=2, label=f'{model_name} (AUC = {roc_auc:.
 ↪2f})')

ax[0].set_xlim([0.0, 1.0])
ax[0].set_ylim([0.0, 1.05])
ax[0].set_xlabel('False Positive Rate')
ax[0].set_ylabel('True Positive Rate')
ax[0].legend(loc='lower right')

# Graficar las curvas Precision-Recall (Derecha)
ax[1].set_title("Precision-Recall Curve")
for model_name, metrics in scores.items():
    mean_recall = metrics['recall']
    mean_precision = metrics['precision']
    pr_auc = auc(mean_recall, mean_precision)
    pr_auc = metrics['pr-auc']

    ax[1].plot(mean_recall, mean_precision, lw=2, label=f'{model_name} (AUC =␣
 ↪{pr_auc:.2f})')

ax[1].set_xlim([0.0, 1.0])
ax[1].set_ylim([0.0, 1.05])
ax[1].set_xlabel('Recall')
ax[1].set_ylabel('Precision')
ax[1].legend(loc='upper right')

# Mostrar la figura con ambas curvas
plt.tight_layout()
```

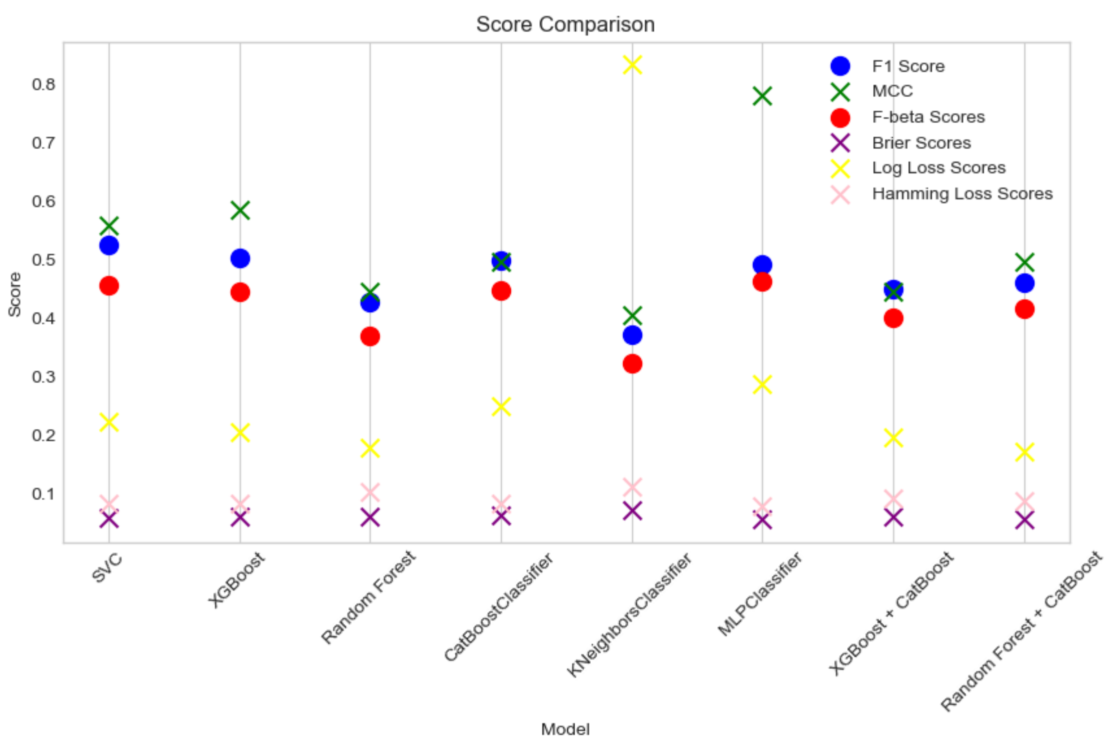
```
plt.show()


print ()
```



[ ]:

```
plt.figure(figsize=(10, 5))
plt.title('Score Comparison')
plt.xlabel('Model')
plt.ylabel('Score')
plt.xticks(rotation=45)
plt.grid(axis='y')

models = list(scores.keys())
f1_scores = [metrics['f1'] for metrics in scores.values()]
mcc_scores = [metrics['mcc'] for metrics in scores.values()]
f_beta_scores = [metrics['f-beta'] for metrics in scores.values()]
brier_scores = [metrics['brier'] for metrics in scores.values()]
log_losses = [metrics['log_loss'] for metrics in scores.values()]
hamming_losses = [metrics['hamming_loss'] for metrics in scores.values()]
roc_auc = [metrics['roc-auc'] for metrics in scores.values()]
pr_auc = [metrics['pr-auc'] for metrics in scores.values()]

plt.scatter(models, f1_scores, color='blue', s=100, label='F1 Score',
  ↪marker='o')
plt.scatter(models, mcc_scores, color='green', s=100, label='MCC', marker='x')
plt.scatter(models, f_beta_scores, color='red', s=100, label='F-beta Scores',
  ↪marker='o')
plt.scatter(models, brier_scores, color='purple', s=100, label='Brier Scores',
  ↪marker='x')
```

```
plt.scatter(models, log_losses, color='yellow', s=100, label='Log Loss Scores',␣
  ↪marker='x')
plt.scatter(models, hamming_losses, color='pink', s=100, label='Hamming Loss␣
  ↪Scores', marker='x')

plt.legend()
plt.show()

print(models)
print('F1 Score')
print(f1_scores)
print('MCC Score')
print(mcc_scores)
print('F-beta Score')
print(f_beta_scores)
print('Brier Score')
print(brier_scores)
print('Log Loss Score')
print(log_losses)
print('Hamming Loss Score')
print(hamming_losses)
print('ROC-AUC)')
print(roc_auc)
print('PR-AUC)')
print(pr_auc)
```



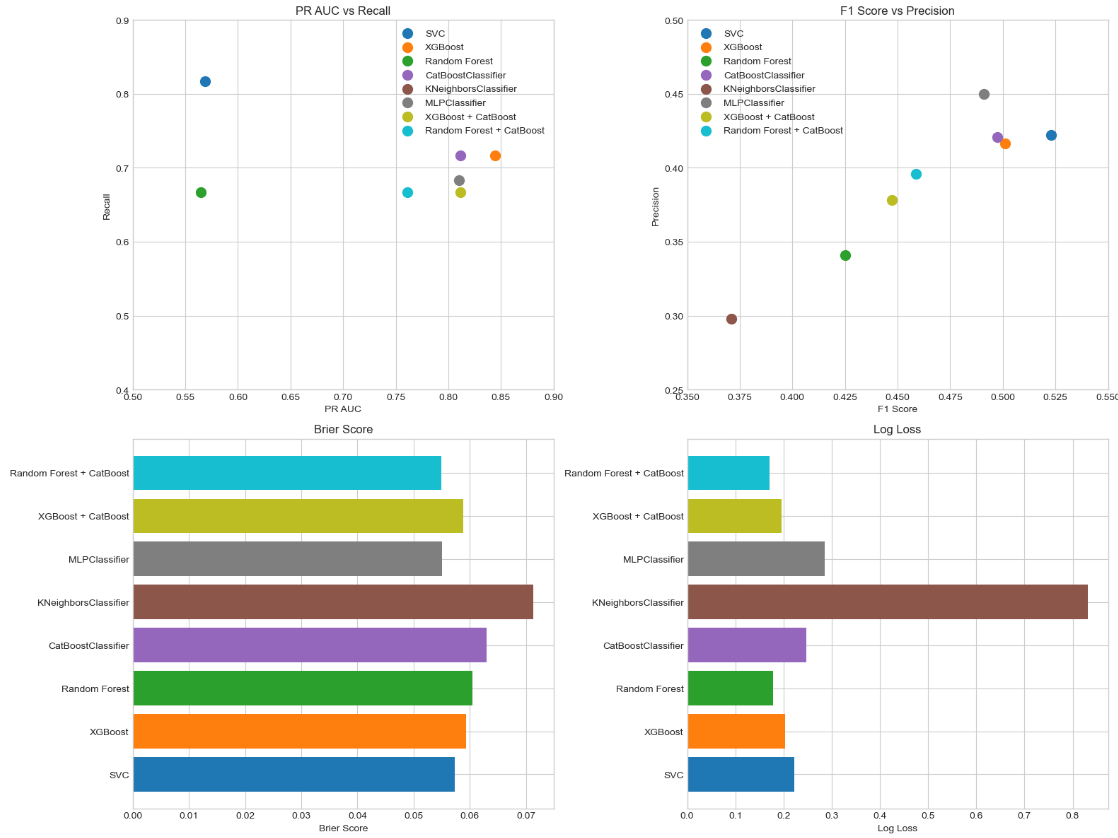
```
['SVC', 'XGBoost', 'Random Forest', 'CatBoostClassifier',
'KNeighborsClassifier', 'MLPClassifier', 'XGBoost + CatBoost', 'Random Forest +
CatBoost']
F1 Score
[0.523015873015873, 0.5010683760683761, 0.42500000000000004,
0.49747474747474746, 0.37094017094017095, 0.49112554112554113,
0.4472222222222222, 0.45858585858585854]
MCC Score
[0.5574468085106383, 0.5841226123902492, 0.4447297294454435,
0.49474575086790806, 0.4033227561742197, 0.7787234042553192, 0.4447297294454435,
0.49474575086790806]
F-beta Score
[0.45455542650664604, 0.4429291929291929, 0.3678109366913921,
0.4454323491848846, 0.3211770594123535, 0.46264367816091956, 0.3988059870412811,
0.41447996823250366]
Brier Score
[0.057357945921455405, 0.05936533091493442, 0.06050987248843567,
0.06301688554717158, 0.07134615384615385, 0.05499736328262671,
0.058871469679505226, 0.05497626683425379]
Log Loss Score
[0.2217554895318748, 0.20377143650502527, 0.17822798885378416,
0.2478069944106551, 0.8318499270856542, 0.2860649001750124, 0.1955278117353063,
0.17152870005305684]
Hamming Loss Score
[0.08173076923076923, 0.08173076923076923, 0.10096153846153846,
0.08173076923076923, 0.11057692307692307, 0.07692307692307693,
0.09134615384615385, 0.08653846153846154]
ROC-AUC)
[0.937039764359352, 0.9513991163475699, 0.9436671575846833, 0.9532400589101621,
0.898379970544919, 0.9484536082474226, 0.9547128129602357, 0.9525036818851251]
PR-AUC)
[0.569047619047619, 0.8444444444444443, 0.5644444444444444, 0.8111111111111111,
0.4742307692307693, 0.81, 0.8111111111111111, 0.7611111111111111]
```

Con esto, observamos que la mejor opción es el XGBoost, debido a que: • PR-AUC más alto, de 0,84 • Buen recall (0,71), por encima de la media de los modelos analizados y un valor suficientemente alto para el modelo • Brier Score y Log Loss bajos, lo que indica una buena calibración de probabilidad • No tiene el F1 más alto, pero en este caso se prioriza el recall

[ ]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Estilo general
plt.style.use('seaborn-whitegrid')
fig, axs = plt.subplots(2, 2, figsize=(16, 12))
```

```python
# Función para extraer valores escalares
def extract_scalar(value):
    if isinstance(value, np.ndarray):
        return np.mean(value)
    return value

# Extraer métricas
recall = [metrics['recall_score'] for metrics in scores.values()]
precision = [metrics['precision_score'] for metrics in scores.values()]

# Colores distintos para las barras
bar_colors = plt.cm.tab10(np.linspace(0, 1, len(models)))

# PR AUC vs Recall
for i, model in enumerate(models):
    axs[0, 0].scatter(pr_auc[i], recall[i], color=bar_colors[i], s=100,␣
 ↪label=model)
axs[0, 0].set_title('PR AUC vs Recall')
axs[0, 0].set_xlim([0.5, 0.9])
axs[0, 0].set_ylim([0.4, 0.9])
axs[0, 0].set_xlabel('PR AUC')
axs[0, 0].set_ylabel('Recall')
axs[0, 0].legend()

# F1 Score vs Precision
for i, model in enumerate(models):
    axs[0, 1].scatter(f1_scores[i], precision[i], color=bar_colors[i], s=100,␣
 ↪label=model)
axs[0, 1].set_title('F1 Score vs Precision')
axs[0, 1].set_xlim([0.35, 0.55])
axs[0, 1].set_ylim([0.25, 0.5])
axs[0, 1].set_xlabel('F1 Score')
axs[0, 1].set_ylabel('Precision')
axs[0, 1].legend()

# Brier Score (barras con colores distintos)
axs[1, 0].barh(models, brier_scores, color=bar_colors)
axs[1, 0].set_title('Brier Score') #Menor es mejor
axs[1, 0].set_xlabel('Brier Score')

# Log Loss (barras con colores distintos)
axs[1, 1].barh(models, log_losses, color=bar_colors)
axs[1, 1].set_title('Log Loss') #Menor es mejor
axs[1, 1].set_xlabel('Log Loss')

plt.tight_layout()
```

```
plt.show()
```



1.4 OPTIMIZACIÓN DE HIPERPARÁMETROS DEL MODELO SELCCIONADO (XGBOOST)

[ ]:
```
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import KFold, cross_val_score

# Modelo XGBoost
model = xgb.XGBClassifier( )

# Crear el pipeline estándar de scikit-learn
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('regressor', model)
])
```

```
# Evaluar el modelo usando KFold
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
↪scoring='accuracy')

# Entrenamiento del pipeline
pipeline.fit(X_train, y_train)
```

[ ]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('scalar',
StandardScaler())]),
                                                  Index(['Htopre',
'Creatininapre', 'Estanciahospitalariacalculada'], dtype='object')),
                                                 ('cat_bin',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                  ('onehot',
                                                                   OneHotEnco…
                               feature_types=None, feature_weights=None,
                               gamma=None, grow_policy=None,
                               importance_type=None,
                               interaction_constraints=None, learning_rate=None,
                               max_bin=None, max_cat_threshold=None,
                               max_cat_to_onehot=None, max_delta_step=None,
                               max_depth=None, max_leaves=None,
                               min_child_weight=None, missing=nan,
                               monotone_constraints=None, multi_strategy=None,
                               n_estimators=None, n_jobs=None,
                               num_parallel_tree=None, …))])
```

[ ]:
```
from sklearn.metrics import recall_score, average_precision_score, make_scorer
from sklearn.model_selection import GridSearchCV

# Custom scoring function compatible con GridSearchCV
def combined_scorer_func(estimator, X, y):
    y_pred = estimator.predict(X)
    y_prob = estimator.predict_proba(X)[:, 1]
    return 0.6 * recall_score(y, y_pred) + 0.4 * average_precision_score(y,␣
↪y_prob)

# No necesitas make_scorer en este caso, pasa directamente la función como␣
↪callable
grid_search = GridSearchCV(
    estimator=pipeline,
    param_grid=param_grid,
```

```python
    scoring=combined_scorer_func,  # Aquí va directamente la función, NO un␣
↪string
    cv=kf,
    n_jobs=-1,
    verbose=2
)

# Ejecutar búsqueda
grid_search.fit(X_train, y_train)

# Mostrar los mejores hiperparámetros
print("Mejores hiperparámetros:", grid_search.best_params_)

# Guardar el mejor modelo encontrado
best_model = grid_search.best_estimator_

# Evaluar usando tu función de evaluación K-Fold
xgboost_scores = evaluate_model_kfolds(best_model, X_train, y_train, kf)
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
```

```
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
```

```
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.2s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.2s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.3s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.3s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.2s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.3s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
```

```
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.3s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.3s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
```

```
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
```

```
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(

[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
```

```
total time=   0.1s
[CV] END regressor__colsample_bytree=0.8, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
```

```
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
```

```
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
```

```
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.01,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
```

```
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
```

```
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
```

```
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
```

```
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.1,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
```

```
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
```

```
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.1s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=3, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
```

```
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=5, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=100, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
```

```
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
```

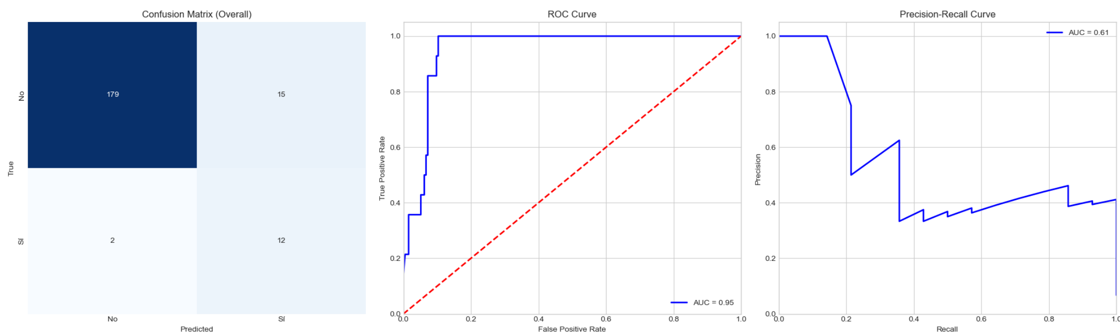
```
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=0.8;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
[CV] END regressor__colsample_bytree=1.0, regressor__learning_rate=0.2,
regressor__max_depth=7, regressor__n_estimators=200, regressor__subsample=1.0;
total time=   0.0s
Mejores hiperparámetros: {'regressor__colsample_bytree': 1.0,
'regressor__learning_rate': 0.01, 'regressor__max_depth': 3,
'regressor__n_estimators': 100, 'regressor__subsample': 0.8}
```

```
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall
is ill-defined and being set to 0.0 due to no true samples. Use `zero_division`
parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/Users/telmomm/anaconda3/lib/python3.11/site-
packages/sklearn/metrics/_ranking.py:1033: UserWarning: No positive class found
in y_true, recall is set to one for all thresholds.
  warnings.warn(
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.548 ± 0.138
ROC AUC: 0.963 ± 0.022
PR AUC: 0.611 ± 0.199
Accuracy: 0.918 ± 0.008
```

```
Recall: 0.900 ± 0.173
Precision: 0.421 ± 0.135
MCC: 0.566 ± 0.11
F-beta Score: 0.462 ± 0.137
Brier Score: 0.082 ± 0.007
Log Loss: 0.314 ± 0.015
Hamming Loss: 0.082 ± 0.008
```



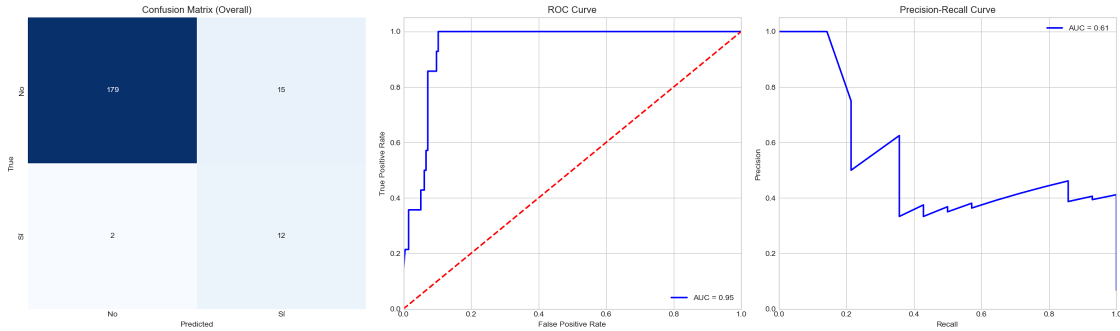
[ ]:
```
from imblearn.pipeline import Pipeline as ImbPipeline
import xgboost as xgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import KFold, cross_val_score

# Modelo XGBoost
model = xgb.XGBClassifier(
    colsample_bytree=1.0,
    learning_rate=0.01,
    max_depth=3,
    n_estimators=100,
    subsample=0.8,

)



# Crear el pipeline estándar de scikit-learn
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42, sampling_strategy='auto')),
    ('regressor', model)
])

# Evaluar el modelo usando KFold
cv_scores = cross_val_score(pipeline, X_train, y_train, cv=kf,␣
 ↪scoring='accuracy')
```

```
# Entrenamiento del pipeline
pipeline.fit(X_train, y_train)
```

```
[ ]: Pipeline(steps=[('preprocessor',
                  ColumnTransformer(transformers=[('num',
                                                   Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                   ('scalar',
StandardScaler())]),
                                                   Index(['Htopre',
'Creatininapre', 'Estanciahospitalariacalculada'], dtype='object')),
                                                  ('cat_bin',
                                                   Pipeline(steps=[('imputer',
SimpleImputer(strategy='most_frequent')),
                                                                   ('onehot',
                                                                    OneHotEnco…
                              feature_types=None, feature_weights=None,
                              gamma=None, grow_policy=None,
                              importance_type=None,
                              interaction_constraints=None, learning_rate=0.01,
                              max_bin=None, max_cat_threshold=None,
                              max_cat_to_onehot=None, max_delta_step=None,
                              max_depth=3, max_leaves=None,
                              min_child_weight=None, missing=nan,
                              monotone_constraints=None, multi_strategy=None,
                              n_estimators=100, n_jobs=None,
                              num_parallel_tree=None, …))])
```

```
[ ]: xgboost_scores = evaluate_model_kfolds(pipeline, X_train, y_train, kf)
```

```
Skipping fold with only one class in y_test_fold.
F1 Score: 0.548 ± 0.138
ROC AUC: 0.963 ± 0.022
PR AUC: 0.611 ± 0.199
Accuracy: 0.918 ± 0.008
Recall: 0.900 ± 0.173
Precision: 0.421 ± 0.135
MCC: 0.566 ± 0.11
F-beta Score: 0.462 ± 0.137
Brier Score: 0.082 ± 0.007
Log Loss: 0.314 ± 0.015
Hamming Loss: 0.082 ± 0.008
```



[ ]:
```python
#exportar en formato joblib el modelo catboostclassfier
import joblib
# Guardar el modelo
joblib.dump(pipeline, 'model.joblib')
```

[ ]:
```
['model.joblib']
```