tree\app.py

```python
import streamlit as st
import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_curve, auc
from sklearn.tree import plot_tree

# Configuración de la app
st.set_page_config(page_title='Clasificación de Atletas', layout='wide')

# Cargar modelo y datos
@st.cache_resource
def cargar_modelo():
    with open('modelo.pkl', 'rb') as f:
        modelo, scaler, label_encoder = pickle.load(f)
    return modelo, scaler, label_encoder

modelo, scaler, label_encoder = cargar_modelo()

@st.cache_data
def cargar_datos():
    df = pd.read_csv('datos_atletas_nan_outliers.csv')
    variables = ['Volumen Sistolico', 'Peso', 'VO2 Max']
    df[variables] = scaler.transform(df[variables])
    return df, variables

def eliminar_outliers(df, variables):
    df_limpio = df.copy()
    for var in variables:
        Q1 = df[var].quantile(0.25)
        Q3 = df[var].quantile(0.75)
        IQR = Q3 - Q1
        limite_inferior = Q1 - 1.5 * IQR
        limite_superior = Q3 + 1.5 * IQR
        df_limpio = df_limpio[(df_limpio[var] >= limite_inferior) & (df_limpio[var] <= limite_superior)]
    return df_limpio

# Main y subpáginas
pagina = st.sidebar.selectbox("Selecciona una página:", ["Predicción", "Preprocesamiento y Métricas"])

df, variables = cargar_datos()
df_limpio = eliminar_outliers(df, variables)
X = df_limpio[variables]
y = label_encoder.transform(df_limpio['Tipo Atleta'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

```
49
50  if pagina == "Predicción":
51      st.sidebar.title('Ajustes del Modelo')
52
53      volumen_sistolico = st.sidebar.slider('Volumen Sistolico', 40, 150, 75)
54      peso = st.sidebar.slider('Peso', 40, 120, 70)
55      vo2_max = st.sidebar.slider('VO2 Max', 30, 80, 50)
56      criterion = st.sidebar.radio('Criterio', ['gini', 'entropy'])
57      max_depth = st.sidebar.slider('Profundidad del Árbol', 2, 4, 3)
58
59      entrada = np.array([[volumen_sistolico, peso, vo2_max]])
60      entrada = scaler.transform(entrada)
61
62      modelo.set_params(criterion=criterion, max_depth=max_depth)
63      modelo.fit(X_train, y_train)
64
65      prediccion = modelo.predict(entrada)[0]
66      prediccion_label = label_encoder.inverse_transform([prediccion])[0]
67
68      st.title('Clasificación de Atletas')
69      st.write('Esta aplicación predice el tipo de atleta según sus características
    fisiológicas.')
70
71      st.subheader('Predicción:')
72      st.write(f'El atleta pertenece a la categoría: **{prediccion_label}**')
73
74      st.subheader('Visualización del Árbol de Decisión')
75      fig, ax = plt.subplots(figsize=(10, 5))
76      plot_tree(modelo.estimators_[0], feature_names=variables,
    class_names=label_encoder.classes_, filled=True, rounded=True, fontsize=8)
77      st.pyplot(fig)
78
79  elif pagina == "Preprocesamiento y Métricas":
80      st.title('Análisis de Preprocesamiento y Evaluación del Modelo')
81
82      st.subheader('Datos Preprocesados')
83      st.write('Visualización de los datos después del preprocesamiento:')
84      st.dataframe(df_limpio)
85
86      st.subheader('Histogramas de Distribución')
87      fig, ax = plt.subplots(1, 3, figsize=(15, 4))
88      for i, var in enumerate(variables):
89          sns.histplot(df_limpio[var], kde=True, ax=ax[i])
90          ax[i].set_title(f'Distribución de {var}')
91      st.pyplot(fig)
92
93      st.subheader('Diagramas Boxplot (Sin Outliers)')
94      fig2, ax2 = plt.subplots(figsize=(8, 4))
95      sns.boxplot(data=df_limpio[variables], ax=ax2)
96      st.pyplot(fig2)
97
98      st.subheader('Balance de Clases')
99      fig4, ax4 = plt.subplots()
100     clase_counts = pd.Series(y).value_counts().sort_index()
```

```
    ax4.bar(label_encoder.classes_, clase_counts, color='skyblue')
    ax4.set_title("Distribución de Clases")
    ax4.set_ylabel("Número de Instancias")
    st.pyplot(fig4)

    st.subheader('Evaluación del Modelo')
    y_pred = modelo.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, output_dict=True)
    st.write(f"**Accuracy:** {acc:.4f}")
    st.json(report)

    st.subheader('Curva ROC-AUC')
    y_pred_proba = modelo.predict_proba(X_test)[:, 1]
    fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
    roc_auc = auc(fpr, tpr)
    fig3, ax3 = plt.subplots()
    ax3.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    ax3.plot([0, 1], [0, 1], color='gray', linestyle='--')
    ax3.set_xlabel('Tasa de Falsos Positivos')
    ax3.set_ylabel('Tasa de Verdaderos Positivos')
    ax3.set_title('Curva ROC-AUC')
    ax3.legend(loc='lower right')
    st.pyplot(fig3)

```