tree\model.py

```python
import pandas as pd
import numpy as np
import seaborn as sns
import pickle
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_curve, auc
from sklearn.preprocessing import LabelEncoder


df = pd.read_csv('datos_atletas_nan_outliers.csv')
print(df.head())
print(df.info())

#? Ahora detectamos y tratamos los datos atípicos, creando un histograma de estos para visualizar su distribución
# df.hist(figsize=(12, 8), bins=30)
# plt.tight_layout()
# plt.show()

#? Seleccionamos las variables que vamos a normalizar y creamos una lista
variables = ['Volumen Sistolico', 'Peso', 'VO2 Max']

#?Normalizamos las variables con normalización MinMax
scaler = MinMaxScaler()
df[variables] = scaler.fit_transform(df[variables])

#?Creamos histogramas de las variables normalizadas para comprobar que hemos eliminado los outliers
df[variables].hist(figsize=(12, 6), bins=30)
plt.tight_layout()
plt.show()

#? Creamos un diagrama boxplot para visualizar los outliers
# sns.boxplot(data=df[variables])
# plt.title("Boxplot de las variables seleccionadas")
# plt.show()

#? Filtramos los outliers:
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)]
```

```
49  print(df_limpio.shape)
50
51  #? Ahora volvemos a crear el diagrama boxplot para ver si los outliers se han eliminado
52  # sns.boxplot(data=df_limpio[variables])
53  # plt.title("Boxplot de las variables sin outliers")
54  # plt.show()
55
56  #? Declaramos las variables:
57  X = df_limpio[variables]
58  y = df_limpio['Tipo Atleta']
59
60  #? Convertir la variable 'Tipo de atleta' a valores binarios (0 y 1)
61  le = LabelEncoder()
62  df_limpio['Tipo Atleta'] = le.fit_transform(df_limpio['Tipo Atleta'])
63
64  #? Definimos X e y
65  X = df_limpio[variables]
66  y = df_limpio['Tipo Atleta']
67
68  #? Dividimos los datos en train, 80% y test 20%
69  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42,
    stratify=y)
70
71  #? Entrenamos el modelo
72  modelo = RandomForestClassifier(n_estimators=100, random_state=42)
73  modelo.fit(X_train, y_train)
74
75  #? Realizamos las predicciones, tanto de la curva ROC-AUC, como las del modelo
76  y_pred = modelo.predict(X_test)
77  y_pred_proba = modelo.predict_proba(X_test)[:, 1]
78
79  #? Realizamos la evaluación del modelo y pedimos las métricas de evaluación
80  fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
81  roc_auc = auc(fpr, tpr)
82
83  #? Representamos la Curva ROC
84  plt.figure(figsize=(8,6))
85  plt.plot(fpr, tpr, color='blue', lw=2, label=f'Curva ROC (AUC = {roc_auc:.2f})')
86  plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
87  plt.xlim([0.0, 1.0])
88  plt.ylim([0.0, 1.05])
89  plt.xlabel('Tasa de Falsos Positivos (FPR)')
90  plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
91  plt.title('Curva ROC')
92  plt.legend(loc='lower right')
93  plt.show()
94
95  #? Ahora imprimimos la matriz de confusión y las métricas de evaluación:
96  print("Matriz de Confusión:")
97  conf_matrix = confusion_matrix(y_test, y_pred)
98  sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
99  plt.xlabel("Predicción")
100 plt.ylabel("Real")
101 plt.title("Matriz de Confusión")
```

```
102 plt.show()
103
104 print("Reporte de Clasificación:")
105 print(classification_report(y_test, y_pred))
106
107 accuracy = accuracy_score(y_test, y_pred)
108 print(f"Accuracy: {accuracy:.4f}")
109
110 #? Guardamos el modelo y el scaler
111 with open('modelo.pkl', 'wb') as f:
112     pickle.dump((modelo, scaler, le), f)
113
114 print("Modelo entrenado y guardado como 'modelo.pkl'")
```