import streamlit as st  
import yfinance as yf  
import pandas as pd  
import numpy as np  
from ta.momentum import RSIIndicator  
from xgboost import XGBClassifier  
from sklearn.model\_selection import train\_test\_split  
import matplotlib.pyplot as plt  
  
st.set\_page\_config(layout="wide")  
st.title("Algoritmo di Trading RSI + XGBoost")  
  
# Sidebar inputs  
ticker = st.sidebar.text\_input("Ticker", value="NVDA")  
period = st.sidebar.selectbox("Periodo", ["1y", "2y", "3y", "5y"], index=2)  
profit\_threshold = st.sidebar.slider("Soglia profitto target (%)", 0.5, 5.0, 1.0) / 100  
  
# Carica dati  
@st.cache\_data  
def load\_data(ticker, period):  
    return yf.download(ticker, period=period, interval="1d")  
  
data = load\_data(ticker, period)  
  
# Calcoli tecnici  
data['RSI'] = RSIIndicator(close=data['Close'], window=14).rsi()  
data['Return'] = data['Close'].pct\_change().shift(-1)  
data['Target'] = (data['Return'] > profit\_threshold).astype(int)  
data.dropna(inplace=True)  
  
# Features  
X = data[['RSI']]  
y = data['Target']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=False)  
  
# Modello  
model = XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss')  
model.fit(X\_train, y\_train)  
y\_pred = model.predict(X\_test)  
  
# Segnali e backtest  
signals = pd.DataFrame(index=data.index[-len(y\_test):])  
signals['Prediction'] = y\_pred  
signals['RSI'] = X\_test['RSI']  
signals['Price'] = data.loc[signals.index, 'Close']  
signals['Buy\_Signal'] = (signals['Prediction'] == 1) & (signals['RSI'] < 30)  
signals['Sell\_Signal'] = (signals['Prediction'] == 0) & (signals['RSI'] > 70)  
signals['Position'] = 0  
signals.loc[signals['Buy\_Signal'], 'Position'] = 1  
signals['Daily\_Return'] = data.loc[signals.index, 'Close'].pct\_change().fillna(0)  
signals['Strategy\_Return'] = signals['Position'].shift(1) \* signals['Daily\_Return']  
signals['Cumulative\_Strategy\_Return'] = (1 + signals['Strategy\_Return']).cumprod()  
  
# ROI  
initial\_investment = 1000  
final\_value = initial\_investment \* signals['Cumulative\_Strategy\_Return'].iloc[-1]  
roi\_percentage = (final\_value - initial\_investment) / initial\_investment \* 100  
  
st.metric("Valore finale", f"{final\_value:.2f} €")  
st.metric("ROI stimato", f"{roi\_percentage:.2f} %")  
  
# Grafico operazioni  
fig, ax = plt.subplots(figsize=(12, 6))  
ax.plot(signals['Price'], label='Prezzo', alpha=0.6)  
ax.scatter(signals.index[signals['Buy\_Signal']], signals['Price'][signals['Buy\_Signal']], marker='^', color='green', label='Buy')  
ax.scatter(signals.index[signals['Sell\_Signal']], signals['Price'][signals['Sell\_Signal']], marker='v', color='red', label='Sell')  
ax.set\_title(f"Operazioni Buy/Sell - {ticker}")  
ax.legend()  
ax.grid(True)  
st.pyplot(fig)  
  
# Download CSV  
csv = signals[['Price', 'RSI', 'Buy\_Signal', 'Sell\_Signal', 'Cumulative\_Strategy\_Return']].to\_csv().encode('utf-8')  
st.download\_button("Scarica segnali (CSV)", csv, file\_name=f"{ticker}\_signals.csv", mime="text/csv")