import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

# Embedded sample stock price data (closing prices only)

sample\_data = [

112.01, 113.05, 115.31, 114.42, 116.85, 117.29, 118.69, 119.03, 120.01, 121.03,

122.45, 123.12, 124.87, 123.56, 124.79, 126.52, 125.73, 127.82, 129.41, 130.12,

131.58, 132.97, 131.76, 130.15, 131.49, 132.23, 133.81, 135.12, 136.97, 138.03,

139.47, 140.92, 142.03, 143.12, 144.21, 145.64, 146.79, 147.01, 146.23, 147.56,

149.31, 148.62, 147.83, 149.94, 151.32, 152.04, 153.21, 154.18, 155.67, 156.32,

157.84, 158.67, 159.88, 161.23, 162.31, 163.78, 164.56, 165.41, 166.75, 168.42

]

# Convert to DataFrame

df = pd.DataFrame(sample\_data, columns=["Close"])

# Scale data

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(df)

# Prepare dataset for LSTM

window\_size = 10

X, y = [], []

for i in range(window\_size, len(scaled\_data)):

X.append(scaled\_data[i - window\_size:i, 0])

y.append(scaled\_data[i, 0])

X, y = np.array(X), np.array(y)

X = np.reshape(X, (X.shape[0], X.shape[1], 1))

# Build LSTM model

model = Sequential([

LSTM(50, return\_sequences=True, input\_shape=(X.shape[1], 1)),

LSTM(50),

Dense(1)

])

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(X, y, epochs=20, batch\_size=8, verbose=0)

# Predict the next value

last\_sequence = scaled\_data[-window\_size:]

X\_test = np.reshape(last\_sequence, (1, window\_size, 1))

predicted\_scaled = model.predict(X\_test)

predicted\_price = scaler.inverse\_transform(predicted\_scaled)

# Output the prediction

print(f"Predicted next closing price: ${predicted\_price[0][0]:.2f}")

# Plotting

plt.figure(figsize=(10, 5))

plt.plot(range(len(sample\_data)), sample\_data, label="Historical Prices")

plt.scatter(len(sample\_data), predicted\_price[0][0], color='red', label="Predicted Next Price")

plt.title("Stock Price Prediction with AI")

plt.xlabel("Days")

plt.ylabel("Price ($)")

plt.legend()

plt.grid(True)

plt.show()

OUTPUT:

