### 案例 9：基于小波分析 - LSTM 模型的人民币兑欧元汇率预测

* **问题背景**：汇率波动受宏观经济指标、国际贸易收支、利率政策、地缘政治等多种因素影响，短期汇率受投机交易等因素影响存在大量噪声，长期则呈现一定趋势。准确预测汇率对进出口企业规避汇率风险、制定经营策略具有重要意义。
* **问题描述**：某进出口企业需要对人民币兑欧元的未来 30 天的日汇率中间价进行预测。要求模型能够分离汇率序列中的高频噪声（如短期投机交易影响）和低频有效信号（如宏观经济趋势），提高预测精度，帮助企业提前锁定汇率成本。
* **数据情况**：提供过去 10 年的人民币兑欧元日汇率中间价数据，同时提供中国和欧元区的 GDP 增长率、通货膨胀率、利率、进出口贸易差额等宏观经济数据，以及重大国际事件（如贸易摩擦、政策调整）记录。数据存在个别交易日因节假日导致的缺失。

### 案例 9：小波分析 - LSTM 模型汇率预测代码

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import pywt  from sklearn.preprocessing import MinMaxScaler  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import LSTM, Dense  from sklearn.metrics import mean\_squared\_error  # 数据加载与预处理  data = pd.read\_csv('exchange\_rate.csv', parse\_dates=['date'], index\_col='date')  rate\_data = data['rate'].values.reshape(-1, 1)  # 数据归一化  scaler = MinMaxScaler(feature\_range=(0, 1))  rate\_scaled = scaler.fit\_transform(rate\_data)  # 小波分解  def wavelet\_decompose(data, wavelet='db4', level=3):  coeffs = pywt.wavedec(data, wavelet, level=level)  return coeffs # 返回近似系数和细节系数  # 小波重构  def wavelet\_reconstruct(coeffs, wavelet='db4'):  return pywt.waverec(coeffs, wavelet)  # 分解序列  coeffs = wavelet\_decompose(rate\_scaled.flatten(), level=3)  n\_levels = len(coeffs)  # 为每个分量构建LSTM模型并预测  predictions = []  for i in range(n\_levels):  component = coeffs[i].reshape(-1, 1)    # 构建序列数据  def create\_seq(data, look\_back=10):  X, y = [], []  for j in range(len(data) - look\_back - 1):  X.append(data[j:j+look\_back, 0])  y.append(data[j+look\_back, 0])  return np.array(X), np.array(y)    look\_back = 10  X, y = create\_seq(component, look\_back)  X = X.reshape(X.shape[0], X.shape[1], 1)    # 划分训练集和测试集  train\_size = int(len(X) \* 0.8)  X\_train, X\_test = X[:train\_size], X[train\_size:]  y\_train, y\_test = y[:train\_size], y[train\_size:]    # 训练LSTM模型  model = Sequential()  model.add(LSTM(32, input\_shape=(look\_back, 1)))  model.add(Dense(1))  model.compile(loss='mse', optimizer='adam')  model.fit(X\_train, y\_train, epochs=20, batch\_size=16, verbose=0)    # 预测  y\_pred = model.predict(X\_test)  predictions.append(y\_pred)  # 小波重构预测结果  # 对齐各分量长度  min\_len = min(len(p) for p in predictions)  aligned\_preds = [p[:min\_len] for p in predictions]  # 重构  pred\_coeffs = []  for i in range(n\_levels):  # 低通分量（近似系数）需要补全长度  if i == 0:  pred\_coeff = np.zeros\_like(coeffs[i])  pred\_coeff[-min\_len:] = aligned\_preds[i].flatten()  else:  pred\_coeff = np.zeros\_like(coeffs[i])  pred\_coeff[-min\_len:] = aligned\_preds[i].flatten()  pred\_coeffs.append(pred\_coeff)  # 重构序列  final\_pred = wavelet\_reconstruct(pred\_coeffs)  final\_pred = final\_pred[-min\_len:].reshape(-1, 1)  # 反归一化  final\_pred = scaler.inverse\_transform(final\_pred)  test\_actual = scaler.inverse\_transform(rate\_scaled[-min\_len:])  # 评估模型  mse = mean\_squared\_error(test\_actual, final\_pred)  print(f'小波-LSTM模型MSE: {mse}')  # 可视化结果  plt.figure(figsize=(12, 6))  plt.plot(test\_actual, label='实际汇率')  plt.plot(final\_pred, label='预测汇率')  plt.legend()  plt.savefig('exchange\_rate\_prediction.png')  plt.show() |