### 案例 5：基于 GM (1,1)-BP 模型的稀有金属月度价格预测

* **问题背景**：稀有金属（如稀土、钨、钼等）在新能源、高端制造等领域应用广泛，其价格受资源储量、市场需求、政策调控、国际形势等因素影响，且由于交易不频繁，历史数据样本量较少，价格波动呈现非线性特征。
* **问题描述**：某稀有金属贸易公司需要对某类重点稀有金属未来 6 个月的月度价格进行预测。要求模型能够利用有限的历史数据，兼顾价格的长期线性趋势（如资源稀缺导致的价格上涨）和短期非线性波动（如突发政策影响），为公司的采购和销售策略提供指导。
* **数据情况**：提供该稀有金属过去 3 年的月度价格数据（单位：元 / 千克），同时提供相关的行业政策文件、全球产量及需求量统计数据、主要贸易国的进出口情况等。数据共 36 条，存在个别月份因特殊事件导致的价格异常波动。

### 案例 5：GM (1,1)-BP 模型稀有金属月度价格预测代码

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.preprocessing import MinMaxScaler  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense  from sklearn.metrics import mean\_absolute\_percentage\_error  # 灰色预测GM(1,1)模型  class GM11:  def \_\_init\_\_(self):  self.a = None  self.b = None    def fit(self, x0):  x1 = np.cumsum(x0) # 累加生成  z1 = (x1[:-1] + x1[1:]) / 2.0 # 紧邻均值生成  B = np.vstack([-z1, np.ones\_like(z1)]).T  Y = x0[1:].reshape(-1, 1)    # 最小二乘法求解参数  params = np.dot(np.dot(np.linalg.inv(np.dot(B.T, B)), B.T), Y)  self.a, self.b = params[0][0], params[1][0]    def predict(self, x0, n\_pred):  x1\_initial = np.cumsum(x0)[-1]  predictions = []  for k in range(1, n\_pred + 1):  x1\_pred = (x0[0] - self.b / self.a) \* np.exp(-self.a \* k) + self.b / self.a  x0\_pred = x1\_pred - x1\_initial  predictions.append(x0\_pred)  x1\_initial = x1\_pred  return np.array(predictions)  # 数据加载与预处理  data = pd.read\_csv('rare\_metal\_price.csv', parse\_dates=['month'], index\_col='month')  price\_data = data['price'].values  # 划分训练集和预测期  train\_data = price\_data[:-6] # 留6个月作为测试  n\_pred = 6  # GM(1,1)模型训练与预测  gm = GM11()  gm.fit(train\_data)  gm\_pred = gm.predict(train\_data, n\_pred)  # 计算GM(1,1)残差  actual = price\_data[-6:]  gm\_residual = actual - gm\_pred  # 准备BP神经网络数据（用历史价格作为特征预测残差）  def create\_bp\_data(price, residual, look\_back=3):  X, y = [], []  for i in range(len(residual)):  X.append(price[-(look\_back + len(residual)) + i : -(len(residual) - i)])  y.append(residual[i])  return np.array(X), np.array(y)  look\_back = 3  X\_bp, y\_bp = create\_bp\_data(price\_data, gm\_residual, look\_back)  # 数据归一化  scaler\_X = MinMaxScaler()  scaler\_y = MinMaxScaler()  X\_bp\_scaled = scaler\_X.fit\_transform(X\_bp)  y\_bp\_scaled = scaler\_y.fit\_transform(y\_bp.reshape(-1, 1))  # BP神经网络构建与训练  bp\_model = Sequential()  bp\_model.add(Dense(16, input\_dim=look\_back, activation='relu'))  bp\_model.add(Dense(8, activation='relu'))  bp\_model.add(Dense(1))  bp\_model.compile(loss='mse', optimizer='adam')  bp\_model.fit(X\_bp\_scaled, y\_bp\_scaled, epochs=50, batch\_size=2, verbose=1)  # BP预测残差  residual\_pred\_scaled = bp\_model.predict(X\_bp\_scaled)  residual\_pred = scaler\_y.inverse\_transform(residual\_pred\_scaled).flatten()  # 组合预测结果  final\_pred = gm\_pred + residual\_pred  # 评估模型  mape = mean\_</doubaocanvas> |