### 案例 12：基于 SARIMA-GARCH 模型的黄金期货价格波动率预测

* **问题背景**：黄金期货是重要的投资和避险工具，其价格波动率受全球经济形势、地缘政治风险、通货膨胀预期、美元指数等因素影响，呈现出明显的聚类特性（即高波动率之后往往跟随高波动率，低波动率之后往往跟随低波动率），且具有季节性特征。
* **问题描述**：某金融投资机构需要对黄金期货未来 3 个月的日价格波动率进行预测。要求模型能够捕捉价格的季节性趋势、线性自相关以及波动率聚类特性，为投资组合风险管理和交易策略制定提供依据。
* **数据情况**：提供过去 10 年的黄金期货日收盘价数据，同时提供美元指数、全球主要经济体的利率、通货膨胀率、地缘政治事件记录等。数据量约 2500 条，存在部分交易日因节假日缺失的情况。

### 案例 12：SARIMA-GARCH 模型黄金期货价格波动率预测代码

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from statsmodels.tsa.statespace.sarimax import SARIMAX  from arch import arch\_model  from sklearn.metrics import mean\_squared\_error  import statsmodels.api as sm  # 数据加载与预处理  data = pd.read\_csv('gold\_futures.csv', parse\_dates=['date'], index\_col='date')  price\_data = data['price'].dropna()  # 计算收益率（波动率基础）  returns = 100 \* price\_data.pct\_change().dropna() # 百分比收益率  # 划分训练集和测试集  train\_size = int(len(returns) \* 0.8)  train\_returns, test\_returns = returns[:train\_size], returns[train\_size:]  # SARIMA模型拟合均值方程  # 选择最优参数（此处简化为手动指定，实际可通过AIC筛选）  sarima\_order = (1, 0, 1)  seasonal\_order = (1, 0, 1, 22) # 22个交易日为月度周期  sarima\_model = SARIMAX(train\_returns, order=sarima\_order, seasonal\_order=seasonal\_order)  sarima\_result = sarima\_model.fit(disp=False)  # 获取SARIMA残差，用于GARCH模型  sarima\_residuals = sarima\_result.resid  # GARCH模型拟合波动率  garch\_model = arch\_model(sarima\_residuals, vol='GARCH', p=1, q=1)  garch\_result = garch\_model.fit(disp='off')  # 组合预测：先预测均值，再预测波动率  # SARIMA预测均值  sarima\_pred = sarima\_result.get\_forecast(steps=len(test\_returns))  mean\_forecast = sarima\_pred.predicted\_mean  # GARCH预测波动率（条件方差）  garch\_forecast = garch\_result.forecast(horizon=len(test\_returns))  volatility\_forecast = np.sqrt(garch\_forecast.variance.values[-1, :])  # 最终预测（均值±1.96\*波动率，95%置信区间）  upper\_bound = mean\_forecast + 1.96 \* volatility\_forecast  lower\_bound = mean\_forecast - 1.96 \* volatility\_forecast  # 评估波动率预测效果（使用实际收益率的绝对值作为真实波动率近似）  actual\_volatility = np.abs(test\_returns)  mse\_vol = mean\_squared\_error(actual\_volatility, volatility\_forecast)  print(f'波动率预测MSE: {mse\_vol}')  # 可视化结果  plt.figure(figsize=(12, 6))  plt.plot(test\_returns.index, test\_returns, label='实际收益率')  plt.plot(test\_returns.index, mean\_forecast, label='均值预测', color='red')  plt.fill\_between(test\_returns.index, lower\_bound, upper\_bound, color='pink', alpha=0.3, label='95%置信区间')  plt.legend()  plt.savefig('gold\_volatility\_prediction.png')  plt.show() |