### 案例 8：基于马尔可夫链 - ETS 模型的连锁超市生鲜产品短期需求预测

* **问题背景**：生鲜产品具有易腐烂、保质期短的特点，连锁超市需要精准预测其短期需求，以合理制定采购计划，减少损耗。生鲜产品需求受日常消费习惯影响呈现一定趋势和季节性，同时受周末、节假日、促销活动等因素影响产生随机波动。
* **问题描述**：某连锁超市需要对旗下 10 家门店的 15 种生鲜产品（如蔬菜、水果、肉类等）未来 7 天的日需求量进行预测。要求模型能够兼顾产品的日常消费趋势、周度季节性，同时捕捉促销活动或节假日导致的需求突发波动，提高预测的抗干扰能力。
* **数据情况**：提供各门店过去 2 年的生鲜产品日销量数据，同时提供促销活动记录（促销时间、促销方式、折扣幅度等）、节假日信息、天气数据（影响顾客到店频率）等。数据存在因缺货导致的部分零销量记录。

### 案例 8：马尔可夫链 - ETS 模型超市生鲜需求预测代码

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from statsmodels.tsa.holtwinters import ExponentialSmoothing  from sklearn.preprocessing import MinMaxScaler  from sklearn.metrics import mean\_absolute\_error  from collections import Counter  # 数据加载与预处理  data = pd.read\_csv('fresh\_food\_demand.csv', parse\_dates=['date'], index\_col='date')  demand\_data = data['demand'].values  # 划分训练集和测试集  train\_size = int(len(demand\_data) \* 0.8)  train, test = demand\_data[:train\_size], demand\_data[train\_size:]  # ETS模型训练与预测  ets\_model = ExponentialSmoothing(train, trend='add', seasonal='add', seasonal\_periods=7)  ets\_result = ets\_model.fit()  ets\_pred = ets\_result.forecast(len(test))  # 马尔可夫链状态划分与转移矩阵构建  def markov\_chain(residuals, n\_states=5):  # 残差状态划分  scaler = MinMaxScaler(feature\_range=(0, n\_states-1))  scaled\_res = scaler.fit\_transform(residuals.reshape(-1, 1)).astype(int).flatten()    # 计算转移矩阵  transition = np.zeros((n\_states, n\_states))  for i in range(len(scaled\_res)-1):  transition[scaled\_res[i], scaled\_res[i+1]] += 1  transition = transition / transition.sum(axis=1, keepdims=True) # 归一化  return transition, scaler  # 计算训练集残差并构建马尔可夫链  train\_residuals = train - ets\_result.fittedvalues  transition\_matrix, scaler = markov\_chain(train\_residuals)  # 马尔可夫链预测残差  def predict\_residuals(ets\_pred, test\_actual, transition\_matrix, scaler, n\_states=5):  residuals = test\_actual - ets\_pred[:len(test\_actual)]  scaled\_res = scaler.transform(residuals.reshape(-1, 1)).astype(int).flatten()    # 基于最后一个状态预测下一状态  last\_state = scaled\_res[-1]  next\_state\_probs = transition\_matrix[last\_state]  pred\_state = np.argmax(next\_state\_probs)    # 状态反归一化得到残差预测  pred\_residual = scaler.inverse\_transform(np.array([[pred\_state]]))[0][0]  return pred\_residual  # 组合预测（滚动预测）  final\_pred = []  for i in range(len(test)):  # 每步更新ETS预测  current\_ets\_pred = ets\_result.forecast(i+1)[-1]  # 计算当前残差并更新马尔可夫预测  if i == 0:  prev\_residual = test[i] - current\_ets\_pred  pred\_res = prev\_residual # 初始残差  else:  pred\_res = predict\_residuals(ets\_pred[:i], test[:i], transition\_matrix, scaler)  final\_pred.append(current\_ets\_pred + pred\_res)  # 评估模型  mae = mean\_absolute\_error(test, final\_pred)  print(f'组合模型MAE: {mae}')  # 可视化结果  plt.figure(figsize=(12, 6))  plt.plot(test, label='实际需求')  plt.plot(final\_pred, label='预测需求')  plt.legend()  plt.savefig('fresh\_demand\_prediction.png')  plt.show() |