### 案例 1：基于 ARIMA-LSTM 模型的城市短期电力负荷预测

* **问题背景**：随着城市化进程加快和新能源接入电网比例提升，城市电力负荷波动日益复杂。短期电力负荷既受经济发展、居民生活习惯等因素影响呈现线性增长趋势，又会因极端天气（如高温、寒潮）、大型文体活动等突发情况产生非线性剧烈波动。精准预测短期电力负荷，对电力调度部门制定发电计划、保障电网稳定运行具有重要意义。
* **问题描述**：某省级电力公司需要对其管辖的核心城市进行未来 72 小时的电力负荷预测。要求预测模型能够同时捕捉电力负荷的线性趋势（如工作日与周末的常规负荷差异）和非线性波动（如夏季高温导致的空调负荷激增），并给出每小时的负荷预测值，为电网调度提供科学依据。
* **数据情况**：提供该城市过去 5 年的小时级电力负荷数据，包括负荷值（单位：MW）；同时提供对应的气象数据（温度、湿度、降雨量等）、节假日信息、大型活动举办记录等辅助数据。数据存在部分缺失值和异常值，需要进行预处理。

### 案例 1：ARIMA-LSTM 模型城市短期电力负荷预测代码

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from statsmodels.tsa.arima.model import ARIMA  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('electricity\_load.csv', parse\_dates=['time'], index\_col='time')  load\_data = data['load'].values.reshape(-1, 1)  # 数据归一化  scaler = MinMaxScaler(feature\_range=(0, 1))  load\_scaled = scaler.fit\_transform(load\_data)  # 划分训练集和测试集  train\_size = int(len(load\_scaled) \* 0.8)  train, test = load\_scaled[0:train\_size, :], load\_scaled[train\_size:len(load\_scaled), :]  # ARIMA模型训练与预测  arima\_model = ARIMA(train, order=(5, 1, 0))  arima\_result = arima\_model.fit()  arima\_pred = arima\_result.predict(start=len(train), end=len(train)+len(test)-1, typ='levels')  # 计算ARIMA残差  arima\_residual = test - arima\_pred.reshape(-1, 1)  # 准备LSTM数据  def create\_dataset(dataset, look\_back=24):  dataX, dataY = [], []  for i in range(len(dataset) - look\_back - 1):  a = dataset[i:(i + look\_back), 0]  dataX.append(a)  dataY.append(dataset[i + look\_back, 0])  return np.array(dataX), np.array(dataY)  look\_back = 24  trainX, trainY = create\_dataset(arima\_residual[:-72], look\_back)  testX, testY = create\_dataset(arima\_residual, look\_back)  # 重塑数据格式 [samples, time steps, features]  trainX = np.reshape(trainX, (trainX.shape[0], trainX.shape[1], 1))  testX = np.reshape(testX, (testX.shape[0], testX.shape[1], 1))  # LSTM模型构建与训练  lstm\_model = Sequential()  lstm\_model.add(LSTM(50, input\_shape=(look\_back, 1)))  lstm\_model.add(Dense(1))  lstm\_model.compile(loss='mean\_squared\_error', optimizer='adam')  lstm\_model.fit(trainX, trainY, epochs=20, batch\_size=32, verbose=1)  # LSTM预测残差  lstm\_pred = lstm\_model.predict(testX)  # 组合预测结果  final\_pred = arima\_pred[look\_back+1:] + lstm\_pred.flatten()  # 反归一化  final\_pred = scaler.inverse\_transform(final\_pred.reshape(-1, 1))  test\_actual = scaler.inverse\_transform(test[look\_back+1:])  # 评估模型  mse = mean\_squared\_error(test\_actual, final\_pred)  print(f'组合模型MSE: {mse}')  # 可视化结果  plt.figure(figsize=(12, 6))  plt.plot(test\_actual, label='实际负荷')  plt.plot(final\_pred, label='预测负荷')  plt.legend()  plt.savefig('electricity\_prediction.png')  plt.show() |