### 案例 14：基于 LSTM-SVR 模型的风电场风速短期预测

* **问题背景**：风能是一种清洁可再生能源，风电场的发电量取决于风速，而风速具有强烈的非线性和随机性，受地形、气候、湍流等因素影响，短期风速预测难度较大。准确预测风速对风电场的功率调度、电网稳定运行具有重要意义。
* **问题描述**：某风电场需要对场内 10 台风机未来 48 小时的每 10 分钟风速进行预测。要求模型能够先捕捉风速的整体非线性变化趋势，再对预测残差进行修正，以降低过拟合风险，提高预测精度，为风电场的发电计划制定提供支持。
* **数据情况**：提供该风电场过去 3 年的每 10 分钟风速数据，同时提供风机的位置信息、地形数据、气象数据（温度、湿度、气压、风向等）。数据存在因设备维护导致的部分时段缺失，且风速受湍流影响存在高频波动。

### 案例 14：LSTM-SVR 模型风电场风速短期预测代码

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.preprocessing import MinMaxScaler  from sklearn.svm import SVR  from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import LSTM, Dense  import joblib  # 数据加载与预处理  data = pd.read\_csv('wind\_farm\_data.csv', parse\_dates=['time'], index\_col='time')  wind\_speed = data['wind\_speed'].values.reshape(-1, 1)  # 数据归一化  scaler = MinMaxScaler(feature\_range=(0, 1))  wind\_scaled = scaler.fit\_transform(wind\_speed)  # 构建序列数据  def create\_sequences(data, look\_back=12):  X, y = [], []  for i in range(len(data) - look\_back - 1):  X.append(data[i:i+look\_back, 0])  y.append(data[i+look\_back, 0])  return np.array(X), np.array(y)  look\_back = 12 # 用过去12个10分钟数据预测下一个10分钟  X, y = create\_sequences(wind\_scaled, look\_back)  # 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:]  # 重塑LSTM输入形状 [samples, time steps, features]  X\_train\_lstm = X\_train.reshape(X\_train.shape[0], X\_train.shape[1], 1)  X\_test\_lstm = X\_test.reshape(X\_test.shape[0], X\_test.shape[1], 1)  # 第一步：LSTM模型预测  lstm\_model = Sequential()  lstm\_model.add(LSTM(64, input\_shape=(look\_back, 1), return\_sequences=False))  lstm\_model.add(Dense(32, activation='relu'))  lstm\_model.add(Dense(1))  lstm\_model.compile(loss='mse', optimizer='adam')  lstm\_model.fit(X\_train\_lstm, y\_train, epochs=30, batch\_size=32, verbose=1)  # LSTM预测及残差计算  lstm\_pred\_train = lstm\_model.predict(X\_train\_lstm).flatten()  lstm\_pred\_test = lstm\_model.predict(X\_test\_lstm).flatten()  train\_residual = y\_train - lstm\_pred\_train # 训练残差  test\_residual = y\_test - lstm\_pred\_test # 测试残差  # 第二步：SVR模型拟合残差  svr\_model = SVR(kernel='rbf', C=10, gamma=0.1)  svr\_model.fit(X\_train, train\_residual) # 用原始特征预测残差  # SVR残差预测  svr\_residual\_pred = svr\_model.predict(X\_test)  # 组合预测：LSTM预测 + SVR残差修正  final\_pred = lstm\_pred\_test + svr\_residual\_pred  # 反归一化  final\_pred\_actual = scaler.inverse\_transform(final\_pred.reshape(-1, 1))  y\_test\_actual = scaler.inverse\_transform(y\_test.reshape(-1, 1))  lstm\_pred\_actual = scaler.inverse\_transform(lstm\_pred\_test.reshape(-1, 1))  # 评估模型  print(f'LSTM模型MAE: {mean\_absolute\_error(y\_test\_actual, lstm\_pred\_actual)}')  print(f'LSTM-SVR组合模型MAE: {mean\_absolute\_error(y\_test\_actual, final\_pred\_actual)}')  # 可视化结果  plt.figure(figsize=(12, 6))  plt.plot(y\_test\_actual[:100], label='实际风速')  plt.plot(lstm\_pred\_actual[:100], label='LSTM预测', alpha=0.7)  plt.plot(final\_pred\_actual[:100], label='组合模型预测', alpha=0.7)  plt.legend()  plt.savefig('wind\_speed\_prediction.png')  plt.show()  # 保存模型  lstm\_model.save('lstm\_wind\_speed.h5')  joblib.dump(svr\_model, 'svr\_residual\_wind.pkl')  joblib.dump(scaler, 'scaler\_wind.pkl') |