#### 一、数据预处理操作，包括数据清洗、数据分析以及数据可视化。时间、水位、流量三列数据都没有空缺值，年月日按照前一个非空值填充，筛选出所有含沙量不为空白的行，即完成数据预处理。

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# 设置中文和负号显示  
plt.rcParams['font.sans-serif'] = ['SimHei']  
plt.rcParams['axes.unicode\_minus'] = False  
  
# 先导入数据，时间、水位、流量三列数据都没有空缺值。年月日按照前一个非空值填充，筛选出含沙量不为空白的行  
data2016 = pd.read\_excel('附件1.xlsx', sheet\_name='2016')  
data2017 = pd.read\_excel('附件1.xlsx', sheet\_name='2017')  
data2018 = pd.read\_excel('附件1.xlsx', sheet\_name='2018')  
data2019 = pd.read\_excel('附件1.xlsx', sheet\_name='2019')  
data2020 = pd.read\_excel('附件1.xlsx', sheet\_name='2020')  
data2021 = pd.read\_excel('附件1.xlsx', sheet\_name='2021')  
  
# 合并六个表格的数据  
dataframeList = [data2016, data2017, data2018, data2019, data2020, data2021]  
years = [2016, 2017, 2018, 2019, 2020, 2021]  
  
# 填充年月日三列  
for df in dataframeList:  
 df[['年', '月', '日']] = df[['年', '月', '日']].ffill()  
  
# 删除含有缺失值的行  
for i in range(len(dataframeList)):  
 dataframeList[i] = dataframeList[i].dropna()  
  
  
# 检查和处理列名  
for df in dataframeList:  
 df.columns = df.columns.str.strip() # 去除列名的空格  
  
#显示筛选后的数据  
#print(dataframeList)  
  
# 绘图  
for i, df in enumerate(dataframeList):  
 plt.figure(figsize=(10, 6))  
 ax = plt.subplot()  
 n = df.shape[0]  
  
 if '含沙量(kg/m3)' in df.columns:  
 ax.plot(np.arange(1, n + 1), df['含沙量(kg/m3)'], marker='o', linestyle='-', color='b',  
 markersize=4, alpha=0.7, linewidth=1) # 调整点的大小、透明度和线条粗细  
 else:  
 raise KeyError("列 '含沙量(kg/m3)' 不存在，请检查列名。")  
  
 num\_ticks = min(n, 12)  
 tick\_positions = np.linspace(1, n, num=num\_ticks, dtype=int)  
 tick\_labels = [str(i) for i in range(1, num\_ticks + 1)]  
  
 ax.set\_xticks(tick\_positions)  
 ax.set\_xticklabels(tick\_labels, fontsize=12)  
 ax.set\_xlabel('月份', fontsize=14)  
 ax.set\_ylabel('含沙量 (kg/m3)', fontsize=14)  
 ax.set\_title(f'{years[i]} 年含沙量随时间变化图', fontsize=16)  
  
 ax.grid(True, linestyle='--', alpha=0.5) # 添加网格  
 file\_path = f'{years[i]}时间-含沙量关系图.png'  
 plt.savefig(file\_path, dpi=300)  
 plt.close()

#### 第一问，此时采用多元回归分析，最终预测出含沙量，并计算误差（没有对数据集分类，因为不需要），根据预测结果计算每年的总含沙量、总水流量，预测结果与误差分析在表格中。

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# 设置中文和负号显示  
plt.rcParams['font.sans-serif'] = ['SimHei']  
plt.rcParams['axes.unicode\_minus'] = False  
  
# 先导入数据，时间、水位、流量三列数据都没有空缺值。年月日按照前一个非空值填充，筛选出含沙量不为空白的行  
data2016 = pd.read\_excel('附件1.xlsx', sheet\_name='2016')  
data2017 = pd.read\_excel('附件1.xlsx', sheet\_name='2017')  
data2018 = pd.read\_excel('附件1.xlsx', sheet\_name='2018')  
data2019 = pd.read\_excel('附件1.xlsx', sheet\_name='2019')  
data2020 = pd.read\_excel('附件1.xlsx', sheet\_name='2020')  
data2021 = pd.read\_excel('附件1.xlsx', sheet\_name='2021')  
  
# 合并六个表格的数据  
dataframeList = [data2016, data2017, data2018, data2019, data2020, data2021]  
years = [2016, 2017, 2018, 2019, 2020, 2021]  
  
# 填充年月日三列  
for df in dataframeList:  
 df[['年', '月', '日']] = df[['年', '月', '日']].ffill()  
  
# 删除含有缺失值的行  
for i in range(len(dataframeList)):  
 dataframeList[i] = dataframeList[i].dropna()  
  
# 检查和处理列名  
for df in dataframeList:  
 df.columns = df.columns.str.strip() # 去除列名的空格  
  
  
  
  
import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.linear\_model import LinearRegression  
  
for i in range(6):  
 print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  
 print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  
 # 假设 merged\_df 是已按列合并的数据框  
 y = dataframeList[i].iloc[:, 6].values # 因变量为第 7 列数据  
 x = dataframeList[i].iloc[:, 4:6].values # 自变量为第 5 列到第 6 列数据  
  
 # 构建线性模型  
 model = LinearRegression()  
 model.fit(x, y) # 自变量在前，因变量在后  
  
 # 预测值  
 predicts = model.predict(x)  
  
 # 拟合程度 R^2  
 R2 = model.score(x, y)  
 print('R2 = %.3f' % R2) # 输出 R^2  
  
 # 输出斜率和截距  
 coef = model.coef\_ # 斜率  
 intercept = model.intercept\_ # 截距  
 print('斜率:', coef)  
 print('截距:', intercept)  
  
 # 绘图  
 plt.figure(figsize=(10, 6))  
 plt.plot(np.arange(len(y)), y, label='实际值', marker='o', linestyle='-', color='b', alpha=0.6)  
 plt.plot(np.arange(len(predicts)), predicts, label='预测值', marker='x', linestyle='--', color='r', alpha=0.7)  
 plt.xlabel('样本索引')  
 plt.ylabel('含沙量')  
 plt.legend()  
 plt.title(f"{i + 2016} 年预测")  
 plt.show()  
  
 from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error  
  
 # 计算均方误差（MSE）  
 mse = mean\_squared\_error(y, predicts)  
  
 # 计算均方根误差（RMSE）  
 rmse = np.sqrt(mse)  
  
 # 计算平均绝对误差（MAE）  
 mae = mean\_absolute\_error(y, predicts)  
  
 # 计算决定系数（R²），已经计算过  
 r2 = model.score(x, y)  
  
 # 输出结果  
 print(f'均方误差 (MSE): {mse:.3f}')  
 print(f'均方根误差 (RMSE): {rmse:.3f}')  
 print(f'平均绝对误差 (MAE): {mae:.3f}')  
 print(f'决定系数 (R²): {r2:.3f}')  
  
 import pandas as pd  
 import numpy as np  
  
 mean\_values = dataframeList[i][['流量(m3/s)']].sum()  
  
 # 计算每年的总水流量  
 annual\_flow = mean\_values \* 24 \* 60 \* 60 # 每天流量 \* 一年的秒数  
  
 # 输出年总水流量  
 print(f"{i + 2016} 年总水流量: {annual\_flow}")  
  
  
  
 dataframeList[i]['含沙量(kg/m3)'] = predicts # 确保 predicts 是与 data 长度相同的数组或 Series  
  
 # 计算每天的排沙量（流量 \* 含沙量）  
 daily\_sediment = dataframeList[i]['含沙量(kg/m3)'] \* dataframeList[i]['流量(m3/s)']  
 annual\_sediment = daily\_sediment.sum() \* 24 \* 60 \* 60 # 每天流量 \* 一年的秒数  
  
 # 输出年总排沙量  
 print(f"{i + 2016} 年总含沙量: {annual\_sediment}")  
df\_eval = pd.DataFrame(columns=['flow','sand'])  
Y = 2016  
for df in dataframeList:  
 x = sum(df.groupby(['年','月'])['含沙量(kg/m3)'].mean()\* df.groupby(['月','日'])['流量(m3/s)'].mean())  
 x = x\*24\*60\*60  
 y = sum(df.groupby(['月','日'])['流量(m3/s)'].mean())  
 y = y\*24\*60\*60  
 df\_eval.loc[str(Y), :] = [y, x]  
 Y += 1  
print(df\_eval )

#### 第二问，具体看图片分析规律

##### 突变性，为了方便，设置5%为界限

import pymannkendall as mk  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# 设置中文和负号显示  
plt.rcParams['font.sans-serif'] = ['SimHei']  
plt.rcParams['axes.unicode\_minus'] = False  
  
# 先导入数据，时间、水位、流量三列数据都没有空缺值。年月日按照前一个非空值填充，筛选出含沙量不为空白的行  
data2016 = pd.read\_excel('附件1.xlsx', sheet\_name='2016')  
data2017 = pd.read\_excel('附件1.xlsx', sheet\_name='2017')  
data2018 = pd.read\_excel('附件1.xlsx', sheet\_name='2018')  
data2019 = pd.read\_excel('附件1.xlsx', sheet\_name='2019')  
data2020 = pd.read\_excel('附件1.xlsx', sheet\_name='2020')  
data2021 = pd.read\_excel('附件1.xlsx', sheet\_name='2021')  
  
# 合并六个表格的数据  
dataframeList = [data2016, data2017, data2018, data2019, data2020, data2021]  
years = [2016, 2017, 2018, 2019, 2020, 2021]  
  
# 填充年月日三列  
for df in dataframeList:  
 df[['年', '月', '日']] = df[['年', '月', '日']].ffill()  
  
# 删除含有缺失值的行  
for i in range(len(dataframeList)):  
 dataframeList[i] = dataframeList[i].dropna()  
  
  
# 检查和处理列名  
for df in dataframeList:  
 df.columns = df.columns.str.strip() # 去除列名的空格  
  
  
  
def MK(x, y, st):  
 # st:检验的变量名称  
 n = len(y)  
 # 正序计算  
 # 定义累计量序列Sk，长度n，初始值为0  
 Sk = np.zeros(n)  
 UFk = np.zeros(n)  
  
 # 定义Sk序列元素s  
 s = 0  
  
 for i in range(1, n):  
 for j in range(0,i):  
 if y.iloc[i] > y.iloc[j]:  
 s += 1  
 Sk[i] = s  
 E = (i+1)\*(i/4)  
 Var = (i+1)\*i\*(2\*(i+1)+5)/72  
 UFk[i] = (Sk[i] - E)/np.sqrt(Var)  
  
 # 逆序计算  
 # 定义逆累计量序列Sk2  
 # 定义逆统计量序列Sk2  
 y2 = np.zeros(n)  
 Sk2 = np.zeros(n)  
 UBk = np.zeros(n)  
  
 s = 0  
 y2 = y[::-1]  
  
 for i in range(1, n):  
 for j in range(0,i):  
 if y2.iloc[i] > y2.iloc[j]:  
 s += 1  
 Sk2[i] = s  
 E = (i+1)\*(i/4)  
 Var = (i+1)\*i\*(2\*(i+1)+5)/72  
 UBk[i] = -(Sk2[i] - E)/np.sqrt(Var)  
  
 UBk2 = UBk[::-1]  
  
  
 # 画图  
 plt.figure()  
 plt.plot(range(6),UFk, label='UF', color='b',marker='s')  
 plt.plot(range(6), UBk2, label='UB',color='g', linestyle='--', marker='o')  
 plt.ylabel('Mann-Kendall检验值')  
 plt.xlabel('年份')  
  
 # 添加辅助线  
 x\_lim = plt.xlim()  
 # 添加显著水平线和y=0  
 plt.plot(x\_lim,[-1.96,-1.96],':',color='r',label='5%显著水平')  
 plt.plot(x\_lim, [0,0],'--',color='r')  
 plt.plot(x\_lim,[1.96,1.96],':',color='r')  
 plt.xticks(range(6), x.tolist())  
 # plt.legend(loc='upper right', bbox\_to\_anchor=(0.9,0.95),ncol=3,fancybox=True)  
  
 # 设置图例位置，第一个参数调整左右位置，第二个参数调整上下位置  
 plt.legend(bbox\_to\_anchor=(0.82,0.11), facecolor='w',frameon=False)  
 # 添加文本注释  
 plt.text(0,-1.6,'突变点检验')  
 plt.savefig('MK检验'+st+'.png',dpi=350)  
 plt.show()  
  
 # 创建数据列表  
  
  
flow\_data = [  
 14344560000.0, 15365721600.0, 38998828800.0,  
 38750544000.0, 37287907200.0, 35621863200.0  
]  
  
sand\_data = [  
 16926941213.318216, 18308654134.734844, 313839743902.640869,  
 282082366078.467407, 321765354042.700134, 236622317946.540375  
]  
  
# 将数据列表合并为一个 DataFrame  
df = pd.DataFrame({  
 'flow': flow\_data,  
 'sand': sand\_data  
})  
  
  
x = np.arange(2016, 2022)  
# 检验水流量  
MK(x, df['flow'], 'flow')  
  
# 检验排沙量  
MK(x, df['sand'], 'sand')

##### 季节性

import pymannkendall as mk  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# 设置中文和负号显示  
plt.rcParams['font.sans-serif'] = ['SimHei']  
plt.rcParams['axes.unicode\_minus'] = False  
  
# 先导入数据，时间、水位、流量三列数据都没有空缺值。年月日按照前一个非空值填充，筛选出含沙量不为空白的行  
data2016 = pd.read\_excel('附件1.xlsx', sheet\_name='2016')  
data2017 = pd.read\_excel('附件1.xlsx', sheet\_name='2017')  
data2018 = pd.read\_excel('附件1.xlsx', sheet\_name='2018')  
data2019 = pd.read\_excel('附件1.xlsx', sheet\_name='2019')  
data2020 = pd.read\_excel('附件1.xlsx', sheet\_name='2020')  
data2021 = pd.read\_excel('附件1.xlsx', sheet\_name='2021')  
  
# 合并六个表格的数据  
dataframeList = [data2016, data2017, data2018, data2019, data2020, data2021]  
years = [2016, 2017, 2018, 2019, 2020, 2021]  
  
for df in dataframeList:  
 df.rename(columns={'年': 'year', '月':'month', '日':'day', '时间':'time',  
 '水位(m)':'waterLevel','流量(m3/s)':'flow','含沙量(kg/m3) ': 'sand'}, inplace = True)  
# 填充年月日三列  
  
print(dataframeList)  
for df in dataframeList:  
 df[['year', 'month', 'day']] = df[['year', 'month', 'day']].ffill()  
  
# 删除含有缺失值的行  
for i in range(len(dataframeList)):  
 dataframeList[i] = dataframeList[i].dropna()  
print(dataframeList)  
  
# 检查和处理列名  
for df in dataframeList:  
 df.columns = df.columns.str.strip() # 去除列名的空格  
  
  
  
dataframeList = [data2016, data2017, data2018, data2019, data2020, data2021]  
Df1 = pd.DataFrame()  
  
  
  
for df in dataframeList:  
 x = pd.DataFrame(df.groupby(['month'])['sand'].mean()\*df.groupby(['month'])['flow'].mean())  
 Df1 = pd.concat([Df1, x])  
Df1.index = pd.Series(pd.date\_range('2016-01', periods=12\*6, freq='M'))  
Df2 = pd.DataFrame(columns=['flow'])  
for df in dataframeList:  
 x = pd.DataFrame(df.groupby(['month'])['flow'].mean())  
 Df2 = pd.concat([Df2, x])  
Df2.index = pd.Series(pd.date\_range('2016-01', periods=12\*6, freq='M'))  
Df1.columns=['sand']  
def seasonal\_mul(data, st):  
 # st：变量名称，字符串类型  
 from statsmodels.tsa.seasonal import seasonal\_decompose  
  
 result\_mul = seasonal\_decompose(data, model='multiplicative', extrapolate\_trend='freq')  
  
 plt.rcParams.update({'figure.figsize': (10, 10)})  
 result\_mul.plot()  
 plt.savefig('E题'+st+'\_multiplicative.png')  
 plt.show()  
 # 获取趋势、季节、残差值  
 trend = result\_mul.trend  
 seasonal = result\_mul.seasonal  
 residual = result\_mul.resid  
def seasonal\_add(data, st):  
 # st：变量名称，字符串类型  
 from statsmodels.tsa.seasonal import seasonal\_decompose  
  
 result\_add = seasonal\_decompose(data, model='additive', extrapolate\_trend='freq')  
 plt.rcParams.update({'figure.figsize': (10, 10)})  
 result\_add.plot()  
 plt.savefig('E题'+st+'\_additive.png')  
 plt.show()  
  
seasonal\_add(Df1['sand'], 'sand')

##### 周期性

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from scipy.fftpack import fft, fftfreq  
  
from problem2\_2 import Df1, Df2  
  
  
def plot\_frequency\_spectrum(data, title='Frequency Spectrum'):  
 # 计算傅里叶变换  
 fft\_series = fft(data.values)  
 power = np.abs(fft\_series) # 幅度  
 sample\_freq = fftfreq(fft\_series.size, d=1) # 频率，假设数据点之间的间隔为1  
  
 # 仅考虑正频率  
 pos\_mask = np.where(sample\_freq > 0)  
 freqs = sample\_freq[pos\_mask]  
 powers = power[pos\_mask]  
  
 # 绘制频谱图  
 plt.figure(figsize=(12, 6))  
 plt.plot(freqs, powers, label='Magnitude Spectrum')  
 plt.xlabel('Frequency')  
 plt.ylabel('Amplitude')  
 plt.title(f'{title} - Frequency Spectrum')  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
def plot\_power\_spectrum(data, title='Power Spectrum'):  
 # 计算傅里叶变换  
 fft\_series = fft(data.values)  
 power = np.abs(fft\_series)\*\*2 # 功率谱  
 sample\_freq = fftfreq(fft\_series.size, d=1) # 频率，假设数据点之间的间隔为1  
  
 # 仅考虑正频率  
 pos\_mask = np.where(sample\_freq > 0)  
 freqs = sample\_freq[pos\_mask]  
 powers = power[pos\_mask]  
  
 # 绘制功率谱图  
 plt.figure(figsize=(12, 6))  
 plt.plot(freqs, powers, label='Power Spectrum')  
 plt.xlabel('Frequency')  
 plt.ylabel('Power')  
 plt.title(f'{title} - Power Spectrum')  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
# 使用傅里叶变换绘制水沙通量和流量的频谱图和功率谱图  
plot\_frequency\_spectrum(Df1['sand'], 'Sand Flux')  
plot\_power\_spectrum(Df1['sand'], 'Sand Flux')  
  
plot\_frequency\_spectrum(Df2['flow'], 'Flow Rate')  
plot\_power\_spectrum(Df2['flow'], 'Flow Rate')

#### 第三题预测结果有问题，结果太平了

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import MinMaxScaler  
from tensorflow.keras import Sequential  
from tensorflow.keras.layers import LSTM, Dense  
from tensorflow.keras.optimizers import Adam  
  
# 从 problem2\_2 导入数据  
from problem2\_2 import Df1, Df2  
  
# 计算水沙通量  
data = Df1['sand'] \* Df2['flow']  
flux = data.values.reshape(-1, 1)  
  
# 归一化数据  
scaler = MinMaxScaler(feature\_range=(0, 1))  
flux\_scaled = scaler.fit\_transform(flux)  
  
# 创建 LSTM 模型需要的序列数据  
def create\_sequences(data, seq\_length):  
 x, y = [], []  
 for i in range(len(data) - seq\_length):  
 x.append(data[i:i + seq\_length])  
 y.append(data[i + seq\_length])  
 return np.array(x), np.array(y)  
  
seq\_length = 30 # 使用过去 30 天的数据预测下一天  
x, y = create\_sequences(flux\_scaled, seq\_length)  
  
# 划分训练集和测试集  
train\_size = int(len(x) \* 0.8)  
x\_train, y\_train = x[:train\_size], y[:train\_size]  
x\_test, y\_test = x[train\_size:], y[train\_size:]  
  
# 构建 LSTM 模型  
model = Sequential()  
model.add(LSTM(50, return\_sequences=True, input\_shape=(x\_train.shape[1], 1)))  
model.add(LSTM(50))  
model.add(Dense(1))  
model.compile(optimizer=Adam(learning\_rate=0.001), loss='mse')  
  
# 训练模型  
history = model.fit(x\_train, y\_train, epochs=50, batch\_size=16, validation\_data=(x\_test, y\_test))  
  
# 预测未来的数值  
future\_steps = 365 \* 2 # 预测未来 2 年的数据（假设每日数据）  
input\_seq = flux\_scaled[-seq\_length:]  
predictions = []  
  
for \_ in range(future\_steps):  
 pred = model.predict(input\_seq.reshape(1, seq\_length, 1))  
 predictions.append(pred[0, 0])  
 input\_seq = np.append(input\_seq[1:], pred)  
  
# 反归一化预测值  
predictions = scaler.inverse\_transform(np.array(predictions).reshape(-1, 1))  
  
# 创建预测结果的数据框  
future\_dates = pd.date\_range(start=data.index[-1] + pd.Timedelta(days=1), periods=future\_steps, freq='D')  
prediction\_df = pd.DataFrame(predictions, index=future\_dates, columns=['Predicted Flux'])  
  
# 合并历史数据和预测数据以便绘图  
full\_df = pd.concat([data, prediction\_df])  
  
# 绘制结果图  
plt.figure(figsize=(14, 7))  
plt.plot(data.index, flux, label='历史数据')  
plt.plot(future\_dates, predictions, label='预测数据', linestyle='--')  
plt.title('水沙通量预测')  
plt.xlabel('日期')  
plt.ylabel('通量')  
plt.legend()  
plt.show()  
  
# 分析季节性并创建采样计划  
# 下面是一个示例：根据季节趋势调整采样频率  
# 注意：下面的代码仅为示例，具体调整可能需要根据实际数据和季节性分析来制定  
def determine\_sampling\_plan(df):  
 sampling\_plan = {}  
 for month in range(1, 13):  
 month\_data = df[df.index.month == month]  
 avg\_flux = month\_data['Predicted Flux'].mean()  
 if avg\_flux > 0.8: # 假设通量大于 0.8 为雨季  
 sampling\_plan[month] = '每天检测一次'  
 elif avg\_flux > 0.5: # 假设通量在 0.5 到 0.8 为常规时期  
 sampling\_plan[month] = '每三天检测一次'  
 else: # 否则为稳定期  
 sampling\_plan[month] = '每周检测一次'  
 return sampling\_plan  
  
sampling\_plan = determine\_sampling\_plan(prediction\_df)  
print("未来两年的采样监测方案：")  
for month, plan in sampling\_plan.items():  
 print(f"月份 {month}: {plan}")

#### 第四题，困死了，实在写不下去了，gpt的代码。。。

import numpy as np  
import pandas as pd  
from sklearn.preprocessing import MinMaxScaler  
from tensorflow.keras import Sequential  
from tensorflow.keras.layers import LSTM, Dense  
  
# 读取数据  
data = pd.read\_excel('附件2.xlsx')  
# 假设数据包含时间、河底高程和水沙通量  
  
# 数据预处理  
data = data[['时间', '河底高程', '水沙通量']]  
data.set\_index('时间', inplace=True)  
  
# 标准化  
scaler = MinMaxScaler()  
scaled\_data = scaler.fit\_transform(data)  
  
# 创建训练和测试数据  
def create\_dataset(data, time\_step=1):  
 X, y = [], []  
 for i in range(len(data) - time\_step - 1):  
 X.append(data[i:(i + time\_step), :])  
 y.append(data[i + time\_step, 0]) # 预测河底高程  
 return np.array(X), np.array(y)  
  
time\_step = 10  
X, y = create\_dataset(scaled\_data, time\_step)  
  
# 构建LSTM模型  
model = Sequential()  
model.add(LSTM(50, return\_sequences=True, input\_shape=(time\_step, scaled\_data.shape[1])))  
model.add(LSTM(50, return\_sequences=False))  
model.add(Dense(1))  
model.compile(optimizer='adam', loss='mean\_squared\_error')  
  
# 训练模型  
model.fit(X, y, epochs=100, batch\_size=32, verbose=1)  
  
# 预测  
predictions = model.predict(X)  
  
# 反标准化  
predictions = scaler.inverse\_transform(predictions)