

```

1  演示应用软件 matlab 代码:
2  classdef wave_pre < matlab.apps.AppBase
3  % Properties that correspond to app components
4  properties (Access = public)
5  UIFigure matlab.ui.Figure
6  UIAxes matlab.ui.control.UIAxes
7  Button matlab.ui.control.StateButton
8  DATASlider matlab.ui.control.Slider
9  Label matlab.ui.control.Label
10 Button_2 matlab.ui.control.StateButton
11 Label_2 matlab.ui.control.Label
12 YearsLabel matlab.ui.control.Label
13 YearsSpinner matlab.ui.control.Spinner
14 end
15 % Callbacks that handle component events
16 methods (Access = private)
17 % Value changed function: Button
18 function ButtonValueChanged(app, event)
19     value = app.Button.Value;
20     years=2013:2020;
21     for year=years
22         h=load([num2str(year) 'pred.mat']);
23         hp=h.y_pred;
24         hp=hp';
25         filename=['D:\Test_counter\wave_0001' num2str(year) '.nc'];
26         ncdisp(filename, '/', 'full');
27         partition=ncread(filename, 'partition');
28         time=ncread(filename, 'time');
29         lon=ncread(filename, 'lon');
30         lat=ncread(filename, 'lat');
31         hs=ncread(filename, 'hs');
32         hs1=hs;
33         indices = 1:6:size(hs1, 2);
34         result_hs= hs1(:, indices);
35         cha=result_hs-hp;
36         meanmlp=mean(hp,2);
37         meanfvcom=mean(result_hs,2);
38         meancha=meanfvcom-meanmlp;
39         xmd=500;
40         ymd=500;
41         d=load('D:\Test_counter\bigmap.dat');
42         xoutline=[d(:,1);d(1,1)];
43         youtline=[d(:,2);d(1,2)];
44         [X,Y]=meshgrid(linspace(min(xoutline),max(xoutline),xmd),linspace(min(youtline),max(youtli
45 ne),ymd));
46         Inwater=double(inpolygon(X,Y,xoutline,youtline));
47         Inwater(Inwater==0)=NaN;
48         d=load('D:\Test_counter\hainandao.dat');

```

```

1  x1outline=[d(:,1);d(1,1)];
2  y1outline=[d(:,2);d(1,2)];
3  q=lon(:,1);
4  w=lat(:,1);
5  z=double(meanmlp);
6  Zh=griddata(double(q),double(w),double(z),X,Y,'V4');
7  ZhPlot=Zh.*Inwater;
8  f=contourf(app.UIAxes,X,Y,ZhPlot);
9  shading interp;
10 hold(app.UIAxes,"on");
11 plot(app.UIAxes,xoutline, youtline, 'LineWidth', 0.1, 'Color', 'k');
12 colormap (app.UIAxes,jet);
13 colormap(app.UIAxes,slanCM('coolwarm'));
14 hold(app.UIAxes,"on");
15 caxis(app.UIAxes,[0 2]);
16 axis(app.UIAxes,[98 123 0 25]);
17 box on
18 grid on;
19 set(gca,'GridLineStyle',':');
20 contourLevels = 1:1:8;
21 fill(app.UIAxes,x1outline,y1outline,[1 1 1]*0.7);
22 plot(app.UIAxes,x1outline, y1outline, 'LineWidth', 0.1, 'Color', 'k');
23 set(app.UIAxes,'color',[1 1 1]*0.7,'FontSize',15);
24 hc=colorbar(app.UIAxes,'EastOutside','FontSize',14);
25 set(get(hc,'title'),'string','Hs/m','FontName','Times New Roman','FontSize',15);
26 title(app.UIAxes,[num2str(year), 'yr'],'Fontname','Times New Roman')
27 set(gca,'xtick',[98:5:123],'xticklabel',{'98^oE','103^oE','108^oE','113^oE','118^oE','123^oE'},'lin
28 ewidth',1,'FontSize',21,'Fontname','Times New Roman');
29 set(gca,'ytick',[0:5:25],'yticklabel',{'0^oN','5^oN','10^oN','15^oN','20^oN','25^oN'},'linewidth',1,
30 'FontSize',21,'Fontname','Times New Roman');
31 set(app.UIAxes,'Layer','top')
32 hold(app.UIAxes,"off")
33 xmd=500;
34 ymd=500;
35 figure('Units','pixels','Position',[100,100,700,500]);
36 d=load('D:\Test_counter\bigmap.dat');
37 xoutline=[d(:,1);d(1,1)];
38 youtline=[d(:,2);d(1,2)];
39 [X,Y]=meshgrid(linspace(min(xoutline),max(xoutline),xmd),linspace(min(youtline),max(youtli
40 ne),ymd));
41 Inwater=double(inpolygon(X,Y,xoutline,youtline));
42 Inwater(Inwater==0)=NaN;
43 d=load('D:\Test_counter\hainandao.dat');
44 x1outline=[d(:,1);d(1,1)];
45 y1outline=[d(:,2);d(1,2)];
46 q=lon(:,1);
47 w=lat(:,1);
48 z=double(meanmlp);
    
```

```

1     Zh=griddata(double(q),double(w),double(z),X,Y,'V4');
2     ZhPlot=Zh.*Inwater;
3     f=contourf(X,Y,ZhPlot);
4     shading interp;
5     hold on;
6     plot(xoutline, youtline, 'LineWidth', 0.1, 'Color', 'k');
7     colormap(slanCM('coolwarm'));
8     hold on
9     caxis([0 2]);
10    axis([98 123 0 25]);
11    box on
12    grid on;
13    set(gca,'GridLineStyle',':');
14    contourLevels = 1:1:8;
15    fill(x1outline,y1outline,[1 1 1]*0.7);
16    plot(x1outline, y1outline, 'LineWidth', 0.1, 'Color', 'k');
17    set(gca,'color',[1 1 1]*0.7,'FontSize',15);
18    hc=colorbar('EastOutside','FontSize',14);
19    set(get(hc,'title'),'string','Hs/m','FontName','Times New Roman','FontSize',15);
20    title([num2str(year), 'yr'],'Fontname','Times New Roman')
21    set(gca,'xtick',[98:5:123],'xticklabel',{'98^oE','103^oE','108^oE','113^oE','118^oE','123^oE'},'lin
22    ewidth',1,'FontSize',21,'Fontname','Times New Roman');
23    set(gca,'ytick',[0:5:25],'yticklabel',{'0^oN','5^oN','10^oN','15^oN','20^oN','25^oN'},'linewidth',1,
24    'FontSize',21,'Fontname','Times New Roman');
25    hold off
26    set(gca,'Layer','top')
27    hold on
28    saveas(gcf,num2str(year),'jpg');
29    close
30    end
31    end
32    % Value changed function: YearsSpinner
33    function YearsSpinnerValueChanged(app, event)
34    app.DATASlider.Value = app.YearsSpinner.Value;
35    end
36    % Value changed function: Button_2
37    function Button_2ValueChanged(app, event)
38    value = app.Button_2.Value;
39    year = app.YearsSpinner.Value;
40    figname = [num2str(year), '.jpg'];
41    A = imread(figname);
42    imshow(A);
43    end
44    % Callback function
45    function DATASliderValueChanging(app, event)
46    changingValue = event.Value;
47    app.Gauge.Value = changingValue;
48    figname = [num2str(changingValue), '.jpg'];
    
```

```

1  A = imread(figname);
2  imshow(A);
3  end
4  % Callback function
5  function YearsSpinnerValueChanged2(app, event)
6  %value = app.YearsSpinner.Value;
7  end
8  % Callback function
9  function YearsSpinnerValueChanged3(app, event)
10 %value = app.YearsSpinner.Value;
11 end
12 end
13 % Component initialization
14 methods (Access = private)
15 % Create UIFigure and components
16 function createComponents(app)
17 % Create UIFigure and hide until all components are created
18 app.UIFigure = uifigure('Visible', 'off');
19 app.UIFigure.Position = [100 100 640 480];
20 app.UIFigure.Name = 'UI Figure';
21 % Create UIAxes
22 app.UIAxes = uiaxes(app.UIFigure);
23 title(app.UIAxes, '')
24 xlabel(app.UIAxes, 'Longitude (°E) ');
25 ylabel(app.UIAxes, 'Latitude (°N) ');
26 app.UIAxes.FontName = 'Times New Roman';
27 app.UIAxes.FontSize = 14;
28 app.UIAxes.TitleFontWeight = 'bold';
29 app.UIAxes.Position = [22 96 578 332];
30 % Create Button
31 app.Button = uibutton(app.UIFigure, 'state');
32 app.Button.ValueChangedFcn = createCallbackFcn(app, @ButtonValueChanged, true);
33 app.Button.Text = '预测绘图';
34 app.Button.BackgroundColor = [0 0.4471 0.7412];
35 app.Button.FontName = '微软雅黑';
36 app.Button.FontSize = 16;
37 app.Button.FontColor = [1 1 1];
38 app.Button.Position = [51 9 97 41];
39 % Create DATASlider
40 app.DATASlider = uislidder(app.UIFigure);
41 app.DATASlider.Limits = [2013 2020];
42 app.DATASlider.MinorTicks = [];
43 app.DATASlider.FontName = 'Times New Roman';
44 app.DATASlider.FontSize = 16;
45 app.DATASlider.Position = [295 47 305 3];
46 app.DATASlider.Value = 2013;
47 % Create Label
48 app.Label = uilabel(app.UIFigure);
    
```

```

1  app.Label.BackgroundColor = [0 0.4471 0.7412];
2  app.Label.HorizontalAlignment = 'center';
3  app.Label.FontName = '微软雅黑';
4  app.Label.FontSize = 20;
5  app.Label.FontColor = [1 1 1];
6  app.Label.Position = [1 439 640 42];
7  app.Label.Text = '海浪智能快速预报演示软件';
8  % Create Button_2
9  app.Button_2 = uibutton(app.UIFigure, 'state');
10 app.Button_2.ValueChangedFcn = createCallbackFcn(app, @Button_2ValueChanged, true);
11 app.Button_2.Text = '数据查看';
12 app.Button_2.BackgroundColor = [0.9294 0.6941 0.1255];
13 app.Button_2.FontName = '微软雅黑';
14 app.Button_2.FontSize = 16;
15 app.Button_2.FontColor = [1 1 1];
16 app.Button_2.Position = [163 8 98 42];
17 % Create Label_2
18 app.Label_2 = uilabel(app.UIFigure);
19 app.Label_2.FontName = '微软雅黑';
20 app.Label_2.FontSize = 16;
21 app.Label_2.Position = [418 60 60 23];
22 app.Label_2.Text = '时间/年';
23 % Create YearsLabel
24 app.YearsLabel = uilabel(app.UIFigure);
25 app.YearsLabel.HorizontalAlignment = 'right';
26 app.YearsLabel.FontName = 'Times New Roman';
27 app.YearsLabel.FontSize = 18;
28 app.YearsLabel.Position = [81 67 46 22];
29 app.YearsLabel.Text = 'Years';
30 % Create YearsSpinner
31 app.YearsSpinner = uispinner(app.UIFigure);
32 app.YearsSpinner.Limits = [2013 2020];
33 app.YearsSpinner.ValueChangedFcn = createCallbackFcn(app, @YearsSpinnerValueChanged,
34 true);
35 app.YearsSpinner.FontName = 'Times New Roman';
36 app.YearsSpinner.FontSize = 18;
37 app.YearsSpinner.Position = [142 66 82 22.8000011444092];
38 app.YearsSpinner.Value = 2013;
39 % Show the figure after all components are created
40 app.UIFigure.Visible = 'on';
41 end
42 end
43 % App creation and deletion
44 methods (Access = public)
45 % Construct app
46 function app = wave_pre
47 % Create UIFigure and components
48 createComponents(app)

```

```
1
2      % Register the app with App Designer
3      registerApp(app, app.UIFigure)
4      if nargin == 0
5          clear app
6      end
7      end
8      % Code that executes before app deletion
9      function delete(app)
10     % Delete UIFigure when app is deleted
11     delete(app.UIFigure)
12     end
13     end
14     end
15
```

附录 1：训练模型 Python 代码

```

1  #这里是调用 Scikit-Learn
2  import numpy as np
3  from tensorflow.keras.models import Sequential
4  from tensorflow.keras.layers import LSTM
5  from tensorflow.keras.layers import Dense, Dropout
6  import pandas as pd
7  from matplotlib import pyplot as plt
8  from sklearn.preprocessing import MinMaxScaler
9  from keras.wrappers.scikit_learn import KerasRegressor
10 from keras import regularizers
11 import matplotlib.pyplot as plt
12 import matplotlib
13 from sklearn.model_selection import train_test_split
14 from sklearn.metrics import make_scorer
15 from sklearn.metrics import mean_squared_error
16 from sklearn.model_selection import GridSearchCV
17 from tensorflow.keras.optimizers import Adam
18 from scipy.io import loadmat
19 #%% 导入训练数据
20 mat_data = loadmat('D:\python\CNN-LSTM\风场数据提取/uv2002-20110.5.mat')
21 inputuv= mat_data['uv']
22 h_data=loadmat('D:\python\CNN-LSTM\波浪数据提取/oh2002-2011.mat')
23 ouths=h_data['all_hs']
24 #%% 导入测试数据
25 x_data= loadmat('D:\python\CNN-LSTM\风场数据提取/uv20120.5')
26 xinput=x_data['uv']
27 #%% 数据归一化处理
28 from sklearn.preprocessing import StandardScaler
29 scaler = StandardScaler()
30 X_train_scaled = scaler.fit_transform(inputuv)
31 X_test_scaled = scaler.transform(xinput)
32 X_test=X_test_scaled
33 #%%
34 X_train1=X_train_scaled
35 y_train1=ouths
36 X_train, X_val, y_train, y_val = train_test_split(X_train1, y_train1, test_size=0.2,
37 random_state=42)
38 from sklearn.neural_network import MLPRegressor
39 from sklearn.model_selection import GridSearchCV
40 from sklearn.model_selection import LeaveOneOut
41 # 定义 MLPRegressor 模型
42 mlp = MLPRegressor()
43 # 定义参数网格
44 param_grid = {
45     'solver': ['adam'],
46     #'hidden_layer_sizes': [(256,256)],

```

```

1      'hidden_layer_sizes': [(32), (64), (128), (256), (32, 32), (64, 64), (128, 128), (256, 256), (32,
2      32, 32), (64, 64, 64), (128, 128, 128), (256, 256, 256)],
3      'activation': ['relu'],
4      'alpha': [0.0001],
5      'beta_1': [0.9],
6      'beta_2': [0.999],
7      'batch_size': [200],
8      'early_stopping': [False],
9      'validation_fraction': [0.1],
10     'learning_rate': ['adaptive'],
11     'max_iter': [200],
12     'learning_rate_init': [0.001],
13     'epsilon': [1e-08],
14     'shuffle': [True],
15     'random_state': [None],
16     'warm_start': [False],
17     'n_iter_no_change': [10],
18     'verbose': [True],
19     'tol': [0.0001]}
20 # 初始化 GridSearchCV
21 grid_search = GridSearchCV(estimator=mlp, param_grid=param_grid, scoring='r2', cv=5,
22 verbose=2)
23 # 在训练数据上直接拟合 GridSearchCV neg_mean_absolute_error r2
24 grid_result = grid_search.fit(X_train, y_train)
25 print("Best Parameters:", grid_result.best_params_)
26 # 输出最佳模型的测试分数
27 print("Best Score:", grid_result.best_score_)
28 #%% 获取网格搜索结果
29 from matplotlib.font_manager import FontProperties
30 font = FontProperties(fname=r"C:\WINDOWS\Fonts\simSun.TTC", size=16.5)
31 results = grid_search.cv_results_
32 import matplotlib.pyplot as plt
33 # 获取每次迭代的损失值和 MAE
34 loss_values = grid_result.best_estimator_.loss_curve_
35 mae_values = -grid_result.cv_results_['mean_test_score']
36 # 创建一个新的图形
37 plt.figure(figsize=(10, 5))
38 # 绘制损失值和 MAE 随迭代次数的变化图
39 plt.plot(loss_values, color='royalblue')
40 #plt.plot(mae_values, label='MAE', color='red')
41 # 添加图例和标题
42 plt.legend()
43 plt.xlabel('Epochs', fontproperties=font)
44 plt.ylabel('Loss', fontproperties=font)
45 # 调整坐标轴粗细
46 plt.gca().spines['bottom'].set_linewidth(1.5)
47 plt.gca().spines['left'].set_linewidth(1.5)
48 plt.gca().spines['top'].set_linewidth(1.5)

```



```

1 plt.gca().spines['right'].set_linewidth(1.5)
2 # 调整坐标轴刻度文字大小
3 plt.tick_params(axis='both', which='major', labelsz=14)
4 # 调整坐标轴刻度向内移动
5 plt.tick_params(axis='both', which='major', direction='in', labelsz=14)
6 # 显示图形
7 #plt.show()
8 #g.ax_joint.legend()
9 # 显示图表
10 #plt.show()
11 # 调整布局以防止标签被裁剪
12 plt.tight_layout()
13 plt.savefig('损失系数 5.28.png', dpi=300, bbox_inches='tight') # 保存为 PNG 格式, 分辨
14 率为 300 dpi
15 #%%
16 import joblib
17 # 保存最佳模型
18 best_model = grid_result.best_estimator_
19 joblib.dump(best_model, 'best_model256321.pkl')
20 #%% 加载最佳模型
21 import joblib
22 loaded_model = joblib.load('best_model256321.pkl')
23 #%%在验证集上评估最佳模型性能
24 best_model = grid_search.best_estimator_
25 validation_score = best_model.score(X_val, y_val)
26 print("Validation R^2 Score:", validation_score)
27 #%%
28 # 使用最佳模型 best_model 对验证集 X_val 进行预测
29 # 使用加载的模型进行预测
30 import seaborn as sns
31 import matplotlib.pyplot as plt
32 import numpy as np
33 from sklearn.metrics import mean_squared_error, r2_score
34 val_predict = loaded_model.predict(X_val)
35 # 评估预测性能
36 validation_score1 = r2_score(y_val, val_predict)
37 #%% 绘制模型训练的误差图
38 import seaborn as sns
39 import matplotlib.pyplot as plt
40 import numpy as np
41 from sklearn.metrics import mean_squared_error, r2_score
42 # 假设这里 y_true 和 y_pred 是您的实际数据
43 y_pred = val_predict
44 y_true = y_val
45 #y_true_flat = y_true.flatten()
46 #y_pred_flat = y_pred.flatten()
47 #r2_total = r2_score(y_true_flat, y_pred_flat)
48 #rmse_total = np.sqrt(mean_squared_error(y_true_flat, y_pred_flat))

```

```

1  #bias_total = np.mean(y_pred_flat - y_true_flat)
2  #%%
3  from matplotlib.font_manager import FontProperties
4  font = FontProperties(fname=r"C:\WINDOWS\Fonts\simSun.TTC",size=16.5)
5  mean_y_pred = np.mean(y_pred, axis=1)
6  mean_y_true = np.mean(y_true, axis=1)
7  # 计算 R^2, RMSE 和 Bias
8  r2 = r2_score(mean_y_true, mean_y_pred)
9  rmse = np.sqrt(mean_squared_error(mean_y_true, mean_y_pred))
10 bias = np.mean(mean_y_pred - mean_y_true)
11 # 创建 JointGrid 对象
12 g = sns.JointGrid(x=mean_y_true, y=mean_y_pred, space=0, ratio=4)
13 # 绘制散点图和直方图
14 g = g.plot_joint(plt.scatter, color="royalblue", edgecolor="white", s=50)
15 # 绘制直方图
16 g = g.plot_marginals(sns.histplot, kde=True, color="royalblue")
17 # 计算最佳拟合线的斜率和截距
18 slope, intercept = np.polyfit(mean_y_true, mean_y_pred, 1)
19 # 创建 x 轴的值范围
20 line = np.linspace(min(np.min(mean_y_true), np.min(mean_y_pred)),
21                    max(np.max(mean_y_true), np.max(mean_y_pred)), 30000)
22 # 计算最佳拟合线的 y 值
23 best_fit_line = slope * line + intercept
24 # 绘制最佳拟合线
25 g.ax_joint.plot(line, best_fit_line, color='royalblue')
26 # 绘制 1:1 线
27 g.ax_joint.plot(line, line, color='black', linestyle='--')
28 # 添加注释
29 g.ax_joint.text(0.05, 0.95, f"$R^2$={r2:.2f}\nRMSE={rmse:.2f}m\nBias={bias:.2f}m",
30                verticalalignment='top', horizontalalignment='left',
31                transform=g.ax_joint.transAxes, fontsize=12)
32 # 设置轴标签
33 g.ax_joint.set_xlabel('FVCOM-SWAVE 空间平均有效波高 (m)', fontproperties=font)
34 g.ax_joint.set_ylabel('MLP 空间平均有效波高 (m)', fontproperties=font)
35 # 设置 x 轴和 y 轴的范围
36 max_limit = max(np.max(mean_y_true), np.max(mean_y_pred))
37 min_limit = -0.5;
38 g.ax_joint.set_xlim(min_limit, max_limit+0.5)
39 g.ax_joint.set_ylim(min_limit, max_limit+0.5)
40 # 设置轴的比例为 1:1
41 g.ax_joint.set_aspect('equal')
42 # 添加图例
43 g.ax_joint.legend()
44 # 调整坐标轴刻度向内移动
45 plt.tick_params(axis='both', which='major', direction='in', labelsiz=14)
46 # 显示图表
47 #plt.show()
48 # 调整布局以防止标签被裁剪

```

```

1 plt.tight_layout()
2 # 保存图片
3 plt.savefig('验证 0604.png', dpi=300, bbox_inches='tight') # 保存为 PNG 格式, 分辨率为
4 300 dpi
5 plt.savefig('验证 0604.jpg', dpi=300, bbox_inches='tight') # 保存为 JPG 格式, 分辨率为
6 300 dpi
7 #%%
8 # 使用最佳模型进行预测
9 y_pred_normalized = best_model.predict(X_test)
10 #%%
11 import matplotlib.pyplot as plt
12 import matplotlib
13 original=testh
14 df = pd.DataFrame(original)
15 yoriginal = df.values
16 y1_pred = y_pred_normalized[:,500]
17 y1_original=testh[:, 500]
18 # 设置全局字体为中文字体, 这里以 SimHei 为例
19 matplotlib.rcParams['font.family'] = 'SimHei'
20 # 设置图形的长宽和分辨率
21 plt.figure(figsize=(15, 7), dpi=300)
22 # 绘制曲线, 增加线的粗细
23 plt.plot(y1_original, color='red', label='模拟值', linewidth=2)
24 plt.plot(y1_pred, color='blue', label='预测值', linewidth=2)
25 # 添加图例, 修改字体大小
26 plt.legend(fontsize=14)
27 # 添加标题, 修改字体大小
28 plt.title('预测与模拟波高对比', fontsize=16, pad=10)
29 # 添加坐标轴标签, 修改字体大小
30 plt.xlabel('时间', fontsize=16)
31 plt.ylabel('波高/m', fontsize=16)
32 # 修改坐标轴刻度的字体大小
33 plt.xticks(fontsize=15)
34 plt.yticks(fontsize=15)
35 # 修改坐标轴粗细
36 plt.gca().spines['bottom'].set_linewidth(2)
37 plt.gca().spines['left'].set_linewidth(2)
38 plt.gca().spines['right'].set_linewidth(2)
39 plt.gca().spines['top'].set_linewidth(2)
40 #%% 保存为 mat
41 import scipy.io
42 # 假设您有一些数据
43 data = {'y_pred': y_pred}
44 # 保存为 .mat 文件
45 scipy.io.savemat('predmin.mat', data)
46
47
48

```

附录 2：预测模型 Python 代码

```

1  #这里是调用 Scikit-Learn
2  import numpy as np
3  from tensorflow.keras.models import Sequential
4  from tensorflow.keras.layers import LSTM
5  from tensorflow.keras.layers import Dense, Dropout
6  import pandas as pd
7  from matplotlib import pyplot as plt
8  from sklearn.preprocessing import MinMaxScaler
9  from keras.wrappers.scikit_learn import KerasRegressor
10 from keras import regularizers
11 import matplotlib.pyplot as plt
12 import matplotlib
13 from sklearn.model_selection import train_test_split
14 from sklearn.metrics import make_scorer
15 from sklearn.metrics import mean_squared_error
16 from sklearn.model_selection import GridSearchCV
17 from tensorflow.keras.optimizers import Adam
18 from scipy.io import loadmat
19 #%% 导入应用数据
20 x_data= loadmat('D:\python\CNN-LSTM\风场数据提取/uv20120.5')
21 xinput=x_data['uv']
22 #%% 数据归一化处理
23 from sklearn.preprocessing import StandardScaler
24 scaler = StandardScaler()
25 X_test_scaled = scaler.fit_transform(xinput)
26 X_test=X_test_scaled
27 #%% 加载最佳模型
28 import joblib
29 loaded_model = joblib.load('best_model256321.pkl')
30 # 使用最佳模型进行预测
31 import time
32 start_time = time.time()
33 y_pred= loaded_model.predict(X_test)
34 prediction_time = time.time() - start_time
35 print("模型预测所花费的时间： ", prediction_time, "秒")
36 #%%
37 # 记录预测开始时间
38 # 使用最佳模型进行预测
39 y_pred = loaded_model.predict(X_test)# 计算预测所花费的时间
40 #%% 绘制模型应用的误差图
41 import seaborn as sns
42 import matplotlib.pyplot as plt
43 import numpy as np
44 from sklearn.metrics import mean_squared_error, r2_score
45 # 假设这里 y_true 和 y_pred 是您的实际数据
46 h_data=loadmat('D:\python\CNN-LSTM\波浪数据提取/TQh2012.mat')
47 hs2012=h_data['hs']

```

```

1  y_true = hs2012
2  from matplotlib.font_manager import FontProperties
3  font = FontProperties(fname=r"C:\WINDOWS\Fonts\simSun.TTC",size=16.5)
4  mean_y_pred = np.mean(y_pred, axis=1)
5  mean_y_true = np.mean(y_true, axis=1)
6  # 计算 R^2, RMSE 和 Bias
7  r2 = r2_score(mean_y_true, mean_y_pred)
8  rmse = np.sqrt(mean_squared_error(mean_y_true, mean_y_pred))
9  bias = np.mean(mean_y_pred - mean_y_true)
10 # 创建 JointGrid 对象
11 g = sns.JointGrid(x=mean_y_true, y=mean_y_pred, space=0, ratio=4)
12 # 绘制散点图和直方图
13 g = g.plot_joint(plt.scatter, color="royalblue", edgecolor="white", s=50)
14 # 绘制直方图
15 g = g.plot_marginals(sns.histplot, kde=True, color="royalblue")
16 # 计算最佳拟合线的斜率和截距
17 slope, intercept = np.polyfit(mean_y_true, mean_y_pred, 1)
18 # 创建 x 轴的值范围
19 line = np.linspace(min(np.min(mean_y_true), np.min(mean_y_pred)),
20                    max(np.max(mean_y_true), np.max(mean_y_pred)), 30000)
21 # 计算最佳拟合线的 y 值
22 best_fit_line = slope * line + intercept
23 # 绘制最佳拟合线
24 g.ax_joint.plot(line, best_fit_line, color='royalblue')
25 # 绘制 1:1 线
26 g.ax_joint.plot(line, line, color='black', linestyle='--')
27 # 添加注释
28 g.ax_joint.text(0.05, 0.95, f"$R^2$={r2:.2f}\nRMSE={rmse:.2f}m",
29                verticalalignment='top', horizontalalignment='left',
30                transform=g.ax_joint.transAxes, fontsize=12)
31 # 设置轴标签
32 g.ax_joint.set_xlabel('FVCOM-SWAVE 空间平均有效波高 (m)', fontproperties=font)
33 g.ax_joint.set_ylabel('MLP 空间平均有效波高 (m)', fontproperties=font)
34 # 设置 x 轴和 y 轴的范围
35 max_limit = max(np.max(mean_y_true), np.max(mean_y_pred))
36 min_limit = -0.5;
37 g.ax_joint.set_xlim(min_limit, max_limit+0.5)
38 g.ax_joint.set_ylim(min_limit, max_limit+0.5)
39 # 设置轴的比例为 1:1
40 g.ax_joint.set_aspect('equal')
41 # 添加图例
42 g.ax_joint.legend()
43 # 调整坐标轴刻度向内移动
44 plt.tick_params(axis='both', which='major', direction='in', labelsz=14)
45 # 显示图表
46 #plt.show()
47 # 调整布局以防止标签被裁剪
48 plt.tight_layout()
    
```

```
1  # 保存图片
2  plt.savefig('2012 应用.png', dpi=300, bbox_inches='tight') # 保存为 PNG 格式，分辨率为
3  300 dpi
4  plt.savefig('2012 应用.jpg', dpi=300, bbox_inches='tight') # 保存为 JPG 格式，分辨率为
5  300 dpi
6  #%% 保存为 mat
7  import scipy.io
8  # 假设您有一些数据
9  data = {'y_pred': y_pred}
10 # 保存为 .mat 文件
11 scipy.io.savemat('2012pred.mat', data)
```