问题2

可以直接运行该 notebook

In [1]:
```
# TODO import
import hmz
from hmz.math_model.evaluate import GRA
from hmz.math_model.predict import BP, predict_accuracy

import mitosheet
import numpy as np
import pandas as pd
import plotly as py
import cufflinks as cf
import plotly.express as px
import plotly.graph_objects as go
import plotly.figure_factory as ff

cf.set_config_file(
    offline=True,
    world_readable=True,
    theme='white',        # 设置绘图风格
)

import warnings
warnings.filterwarnings("ignore")

import sklearn
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
```

In [2]:
```
sheet1 = pd.read_excel(
    io="附件1(Attachment 1)2022-51MCM-Problem B.xlsx",
    index_col=None,
    sheet_name='温度(temperature)', )
sheet2 = pd.read_excel(
    io="附件1(Attachment 1)2022-51MCM-Problem B.xlsx",
```

```
    index_col=None,
    sheet_name='产品质量(quality of the products)', )
sheet3 = pd.read_excel(
    io="附件1(Attachment 1)2022-51MCM-Problem B.xlsx",
    index_col=None,
    sheet_name='原矿参数(mineral parameter)', )
```

In [3]:
```
mitosheet.sheet(sheet1, sheet2, sheet3, analysis_to_replay="id-cfmwwnbuec")
```

Out[3]: MitoWidget(analysis_data_json='{"analysisName": "id-cfmwwnbuec", "analysisToReplay": null, "code": [], "stepSu…

In [ ]:

准备数据

In [4]:
```
# todo 找到有效的温度数据
cond1= sheet1.iloc[:, 0].astype('string').apply(lambda x: x[14: 16]) == "50"
data_part1 = sheet1[cond1].iloc[:-2, :]
data_part1.index = [i for i in range(len(data_part1))]
print(data_part1.shape)
data_part1
```

(235, 3)

Out[4]:

	时间 (Time)	系统I温度 (Temperature of system I)	系统II温度 (Temperature of system II)
0	2022-01-13 00:50:00	1173.63	813.92
1	2022-01-13 01:50:00	854.55	767.64
2	2022-01-13 02:50:00	855.34	767.99
3	2022-01-13 03:50:00	853.57	766.20
4	2022-01-13 04:50:00	854.81	768.08
...	...	...	...
230	2022-01-22 17:50:00	1406.05	932.16
231	2022-01-22 18:50:00	1404.32	931.43
232	2022-01-22 19:50:00	1404.68	930.64
233	2022-01-22 20:50:00	1404.85	931.16
234	2022-01-22 21:50:00	1404.76	931.28

In [5]:
```
# todo 找到有效的产品质量数据
sheet2_time_string = sheet2.iloc[:, 0].astype('string').apply(lambda x: x)
exp_date = ["2022-01-20 08:50:00", "2022-01-20 09:50:00", "2022-01-20 10:50:00"]
cond2 = sheet2_time_string.apply(lambda x: x == exp_date[0] or x == exp_date[1] or x == exp_date[2])
data_part2 = sheet2[cond2.apply(lambda x: not x)].iloc[2:, :]
data_part2.index = [i for i in range(len(data_part2))]
print(data_part2.shape)
data_part2
```

```
(235, 5)
```

Out[5]:

	时间 (Time)	指标A (index A)	指标B (index B)	指标C (index C)	指标D (index D)
0	2022-01-13 02:50:00	78.15	26.21	12.93	14.59
1	2022-01-13 03:50:00	78.39	25.22	12.93	14.28
2	2022-01-13 04:50:00	79.22	24.60	12.41	13.70
3	2022-01-13 05:50:00	79.52	23.88	11.55	13.56
4	2022-01-13 06:50:00	80.04	23.48	11.55	13.47
...	...	...	...	...	...
230	2022-01-22 19:50:00	79.76	22.00	11.72	18.84
231	2022-01-22 20:49:00	80.51	22.00	11.37	18.53
232	2022-01-22 21:50:00	80.16	21.78	10.85	17.90
233	2022-01-22 22:50:00	79.79	22.58	11.20	17.05
234	2022-01-22 23:50:00	80.19	21.69	10.68	17.19

In [6]:
```
# todo 找到原矿参数数据
cnt = data_part1.iloc[:, 0].astype('string').apply(lambda x: x[8: 10])
time_cnt = []
for i in pd.DataFrame(cnt).groupby(by='时间 (Time)'):
    time_cnt.append(len(i[1]))
data_part3 = pd.DataFrame(np.repeat(sheet3.iloc[:-2, :].values, time_cnt, axis=0), columns=sheet3.columns)
print(data_part3.shape)
data_part3
```

```
(235, 5)
```

Out[6]:

	时间 (Time)	原矿参数1 (Mineral parameter 1)	原矿参数2 (Mineral parameter 2)	原矿参数3 (Mineral parameter 3)	原矿参数4 (Mineral parameter 4)
0	2022-01-13	49.24	90.38	46.13	28.16
1	2022-01-13	49.24	90.38	46.13	28.16
2	2022-01-13	49.24	90.38	46.13	28.16
3	2022-01-13	49.24	90.38	46.13	28.16
4	2022-01-13	49.24	90.38	46.13	28.16
...	...	...	...	...	...
230	2022-01-22	54.74	93.05	49.03	21.48
231	2022-01-22	54.74	93.05	49.03	21.48
232	2022-01-22	54.74	93.05	49.03	21.48
233	2022-01-22	54.74	93.05	49.03	21.48
234	2022-01-22	54.74	93.05	49.03	21.48

In [7]:

```
mitosheet.sheet(data_part1, data_part2, data_part3, analysis_to_replay="id-hwuxbrihpd")
```

Out[7]: MitoWidget(analysis_data_json='{"analysisName": "id-txxvqwavch", "analysisToReplay": {"analysisName": "id-hwux…

In [ ]:

温度与指标的关系——相关性分析

In [8]:

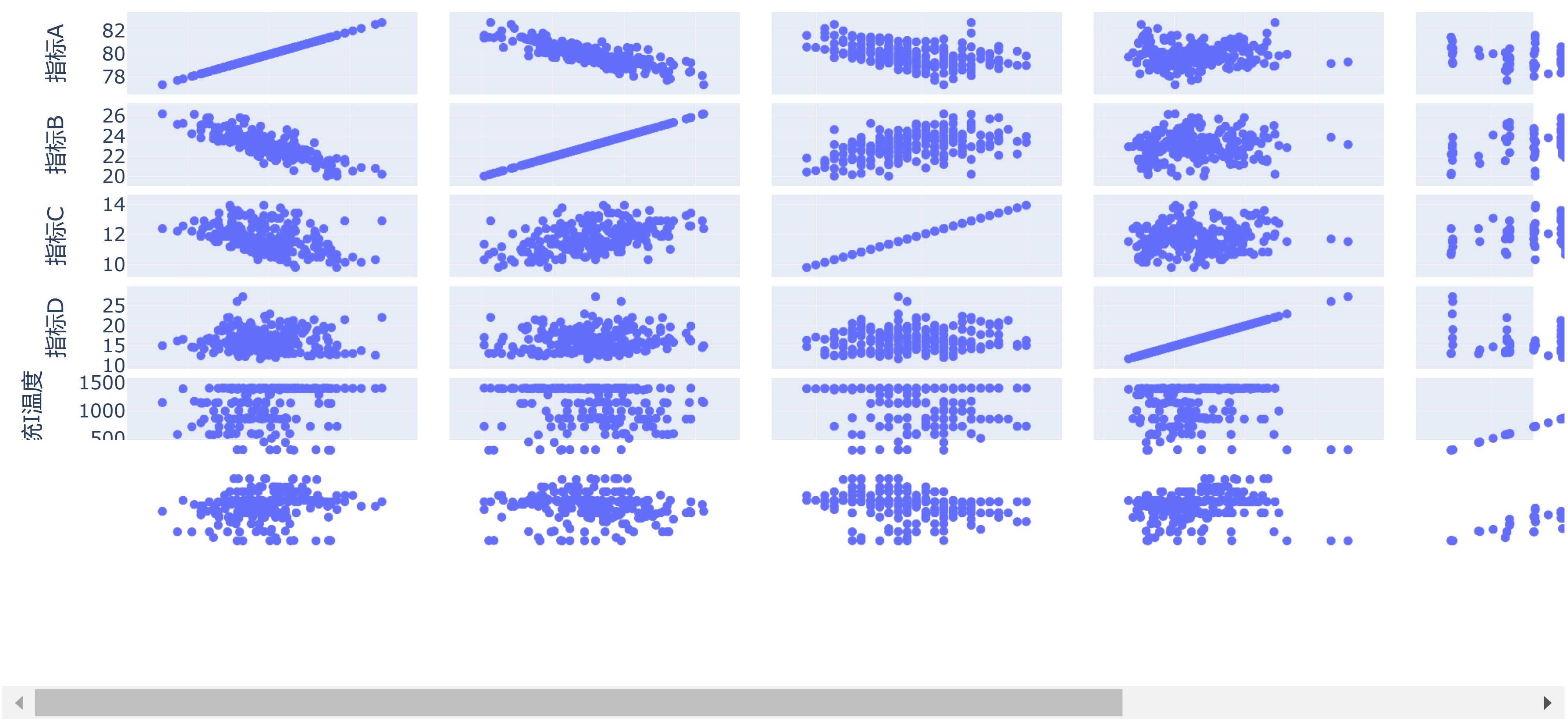
```
X = pd.concat([data_part2.iloc[:, 1:], data_part3.iloc[:, 1:]], axis=1)
Ys = data_part1.iloc[:, 1:]
```

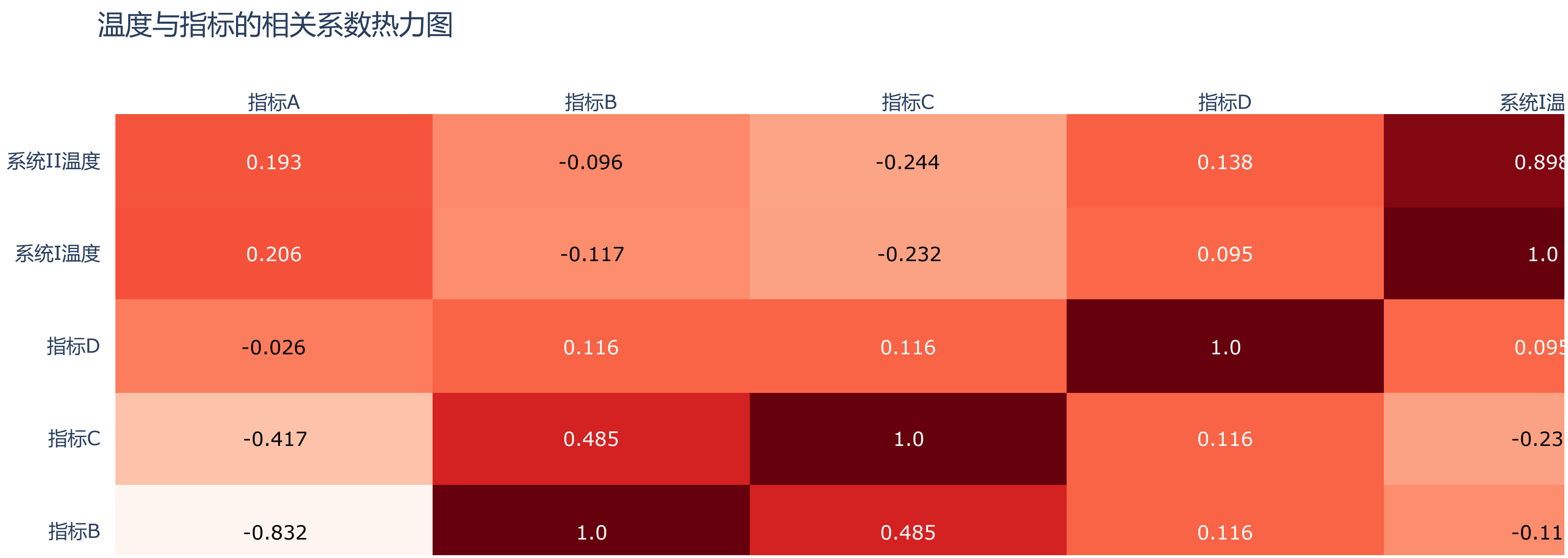
In [9]:

```
wendu_zhibiao = pd.concat([X.iloc[:, :4], Ys], axis=1)
wendu_zhibiao = pd.DataFrame(wendu_zhibiao, dtype=np.float)
wendu_zhibiao.columns = ['指标A', '指标B', '指标C', '指标D', '系统I温度', '系统II温度']

def plot_ScatterMatrix_Heatmap(data, fig_pre_name, save=True):
    # todo 矩阵散点图
    fig = px.scatter_matrix(
        data,
        title=fig_pre_name + '的矩阵散点图',
    )
    if save:
        fig.write_image('./img/问题2-' + fig_pre_name + '的矩阵散点图.svg', width=1100, height=800)
```

```python
    fig.show()

    # todo 相关系数热力图
    corrs = data.corr(method='pearson')  # 'pearson', 'kendall', 'spearman'
    figure = ff.create_annotated_heatmap(
        z=corrs.values,
        x=list(corrs.columns),
        y=list(corrs.index),
        annotation_text=corrs.round(3).values,
        showscale=True,
        colorscale='reds',
    )
    figure.update_layout(title=fig_pre_name + '的相关系数热力图')
    if save:
        figure.write_image('./img/问题2-' + fig_pre_name + '的相关系数热力图.svg')
    figure.show()
    return None

plot_ScatterMatrix_Heatmap(wendu_zhibiao, '温度与指标', True)
```

温度与指标的矩阵散点图





温度与指标的灰色关联度

In [10]:
```
# 灰色关联度
data_temp = pd.concat([Ys, X], axis=1)
data_temp.columns = ['系统I温度', '系统II温度', '指标A', '指标B', '指标C', '指标D', '原矿参数1 (Mineral parameter 1)', '原矿参数2 (Mineral p

rga = GRA(data_temp[['系统I温度', '指标A', '指标B', '指标C', '指标D']])
print('系统I温度：')
rga.gamma()

rga = GRA(data_temp[['系统II温度', '指标A', '指标B', '指标C', '指标D']])
print('系统II温度：')
rga.gamma()
```

```
系统I温度:
灰色关联度:
[0.76119146 0.74778561 0.74120439 0.77103186]

系统II温度:
灰色关联度:
[0.86530354 0.83196642 0.80451489 0.7950681 ]
```

Out[10]:
```
array([0.86530354, 0.83196642, 0.80451489, 0.7950681 ])
```

In [11]:
```python
# 问题2数据
X.to_csv("quention2-X_data.csv")
Ys.to_csv("quention2-Y_data.csv")
mitosheet.sheet(X, Ys, analysis_to_replay="id-yrblhfrhjp")
```

Out[11]:
```
MitoWidget(analysis_data_json='{"analysisName": "id-ljuwcysdbe", "analysisToReplay": {"analysisName": "id-yrbl…
```

In [ ]:

预测温度 I II

In [12]:
```python
index_num = Ys.shape[1]
index_name = ["系统I设定温度", "系统 II 设定温度"]
index_colors = ["red", "lightpink", "blue", "lightblue"]

data_to_predict = np.array(
    [[79.17, 22.72, 10.51, 17.05, 57.5, 108.62, 44.5, 20.09, ],
     [80.10, 23.34, 11.03, 13.29, 57.5, 108.62, 44.5, 20.09, ],],
)
```

In [13]:
```python
from hmz.math_model.process import Dimensionalize

dim = Dimensionalize(X)
XX = dim.fit(method='standard')
XX = XX.fillna(0)
dim = Dimensionalize(Ys)
YYs = dim.fit(method='standard')

th = 0.5
con = 0.2
test_size = 0.3
random_state = 10
```

```python
def run_model(model_name, model, X=X, Ys=Ys, index_num=index_num, test_size=test_size, random_state=random_state):
    """
    :param :
    """
    data = []
    yhats = []
    print(model_name, ":\n")
    for i in range(index_num):
        Y = Ys.iloc[:, i]
        xtrain, xtest, ytrain, ytest = train_test_split(
            np.array(X, dtype=float), np.array(Y, dtype=float),
            test_size=test_size,
            random_state=random_state,
            shuffle=True,
        )  # object -> float

        try:
            model.fit(xtrain, ytrain)  # sklearn
        except:
            model.train(xtrain, ytrain, lr=1e-2, init_method='kmeans', train_method=1, epoch=1000)  # RBF
        yhat = model.predict(xtest)
        yhats.append(yhat)

        acc = predict_accuracy(ytest, yhat, type=1, th=th, con=con)  # todo 评价指标：回归
        print("accuracy:", acc)
        print("MSE:", MSE(yhat, ytest), "MAE:", MAE(yhat, ytest), end='')
        try:
            print(" R2:", model.score(xtest, ytest))
        except:
            pass

        print("预测结果：", model.predict(data_to_predict))

        # todo 画图
        Yhat = model.predict(np.array(X, dtype=float))
        data.append(go.Scatter(
            x=data_part1.index, y=Y,
            name=index_name[i] + "-真实值",
            line=dict(color=index_colors[i * 2 + 1], width=1.0)),
        )
        data.append(go.Scatter(
            x=data_part1.index, y=Yhat,
            name=index_name[i] + "-预测值",
            line=dict(color=index_colors[i * 2], width=1.0)),
        )
```

```python
    # todo 画图：点差图
    cols = str(Y.name)
    Yhat = pd.DataFrame(Yhat)
    Y.index = [i for i in range(len(Y))]
    Y_data = pd.concat([Y, Yhat], axis=1)
    Y_data.columns = ["真实值", "预测值"]

    Y_data.figure(
        kind='spread',
        color=[index_colors[i * 2 + 1], index_colors[i * 2]],
        title='基于' +  model_name + '的' + cols + '预测模型',
    ).write_image('./img/问题2-基于' +  model_name + '的' + cols + '预测模型.svg')
    Y_data.iplot(
        kind='spread',
        color=[index_colors[i * 2], index_colors[i * 2 + 1]],
        title='基于' +  model_name + '的' + cols + '预测模型',
    )
    print()
fig = go.Figure(data=data)

annotations = []
annotations.append(dict(
    x=0.5, y=-0.1,
    xref='paper', yref='paper',
    xanchor='center', yanchor='top',
    text='时间',
    font=dict(size=16),
    showarrow=False,
))
fig.update_layout(
    title='基于' +  model_name + '的系统温度预测模型',
    annotations=annotations,
)
fig.write_image('./img/问题2-基于' + model_name + '的系统温度预测模型.svg')
fig.show()

wendu_zhibiao_ = pd.concat(
    [pd.DataFrame(xtest[:, :4]), pd.DataFrame(yhats).T],
    axis=1,
    ignore_index=True,
).astype(float)  # 4个指标、2个温度
wendu_zhibiao_.columns = ['指标A', '指标B', '指标C', '指标D', '系统I温度', '系统II温度']

# 矩阵散点图、热力图
plot_ScatterMatrix_Heatmap(wendu_zhibiao_, model_name + '预测温度与指标', save=True)
```

```
    # 灰色关联度
    rga = GRA(wendu_zhibiao_[['系统I温度', '指标A', '指标B', '指标C', '指标D']])
    print('系统I温度：')
    rga.gamma()

    rga = GRA(wendu_zhibiao_[['系统II温度', '指标A', '指标B', '指标C', '指标D']])
    print('系统II温度：')
    rga.gamma()

    return wendu_zhibiao_
```

使用回归模型进行预测

多元线性回归、决策树、随机森林、XGBoost、RBF

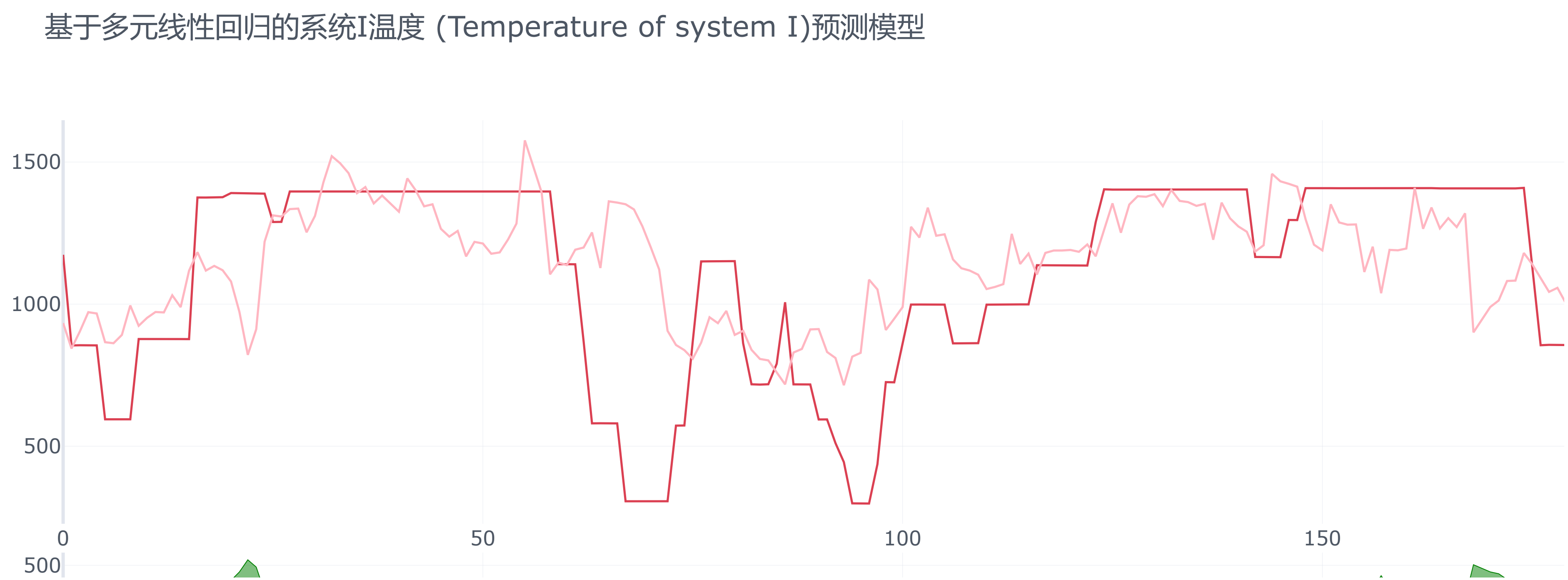
In [14]:

```
test_size_ = 0.3

# sklearn 库
from sklearn.linear_model import LinearRegression as LR
from sklearn.tree import ExtraTreeRegressor as ETR, DecisionTreeRegressor as DTR
from sklearn.ensemble import RandomForestRegressor as RFR
from xgboost import XGBRegressor, XGBRFRegressor


models_names = [
    "多元线性回归",
    "决策树",
    "随机森林",
    "XGBoost",
]

models = [
    LR(),
    ETR(),
    RFR(criterion='mae', n_estimators=100, random_state=24),  # mse, friedman_mse, mae
    XGBRFRegressor(n_estimators=100, random_state=0),
]

for name, model in zip(models_names, models):
    _ = run_model(name, model, test_size=test_size_)

# hmz 库
from hmz.math_model.predict import RBF

model = RBF(hidden_num=int((1 - test_size_) * len(X)), rbf_type=2)
```

```python
_ = run_model("RBF", model, test_size=test_size_)
```

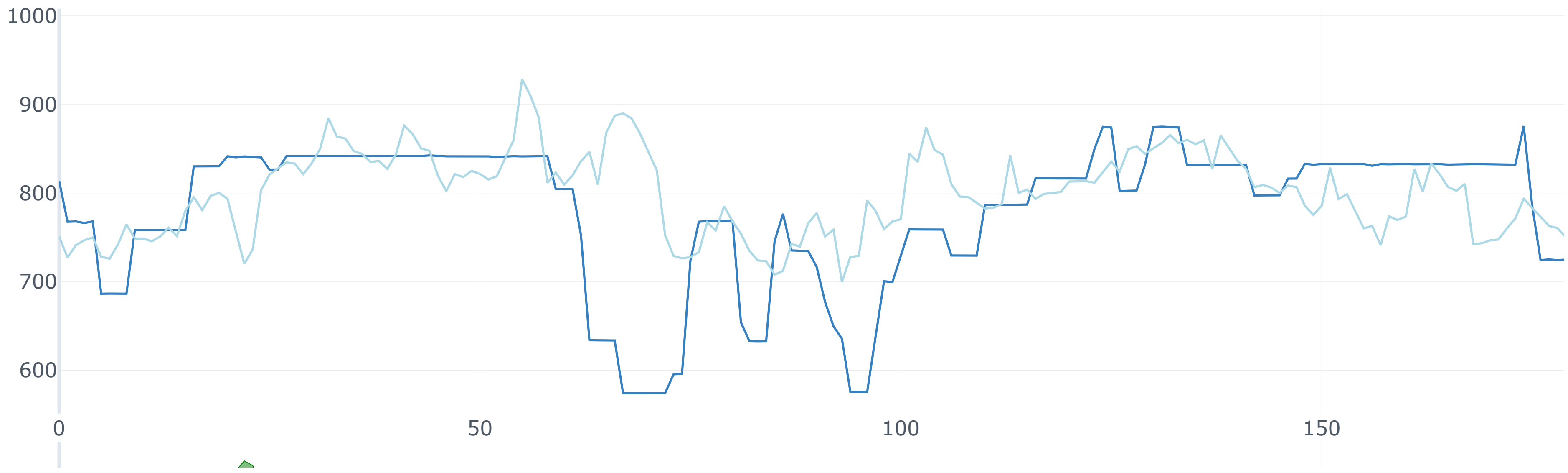
多元线性回归 :

accuracy: 0.7479576906028069
MSE: 85593.81999063717 MAE: 207.28683707736184 R2: 0.2015792522640284
预测结果: [943.87170068 935.68652472]

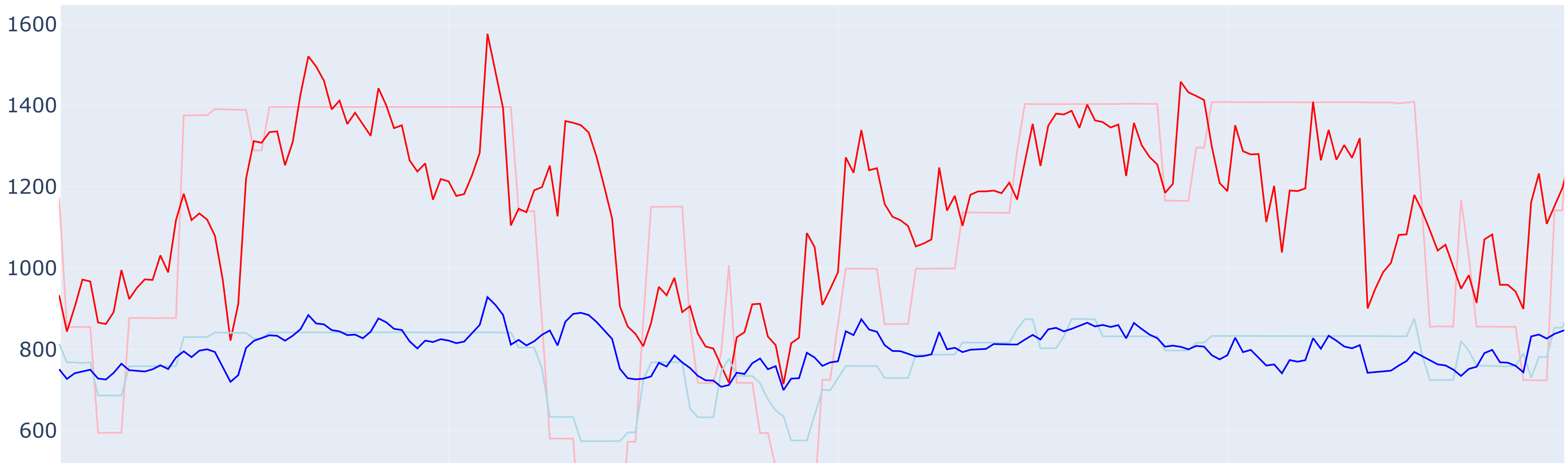


accuracy: 0.9328286273382573
MSE: 6981.525264683283 MAE: 56.635045033290325 R2: 0.07930916612915195
预测结果: [742.76419838 753.05783666]

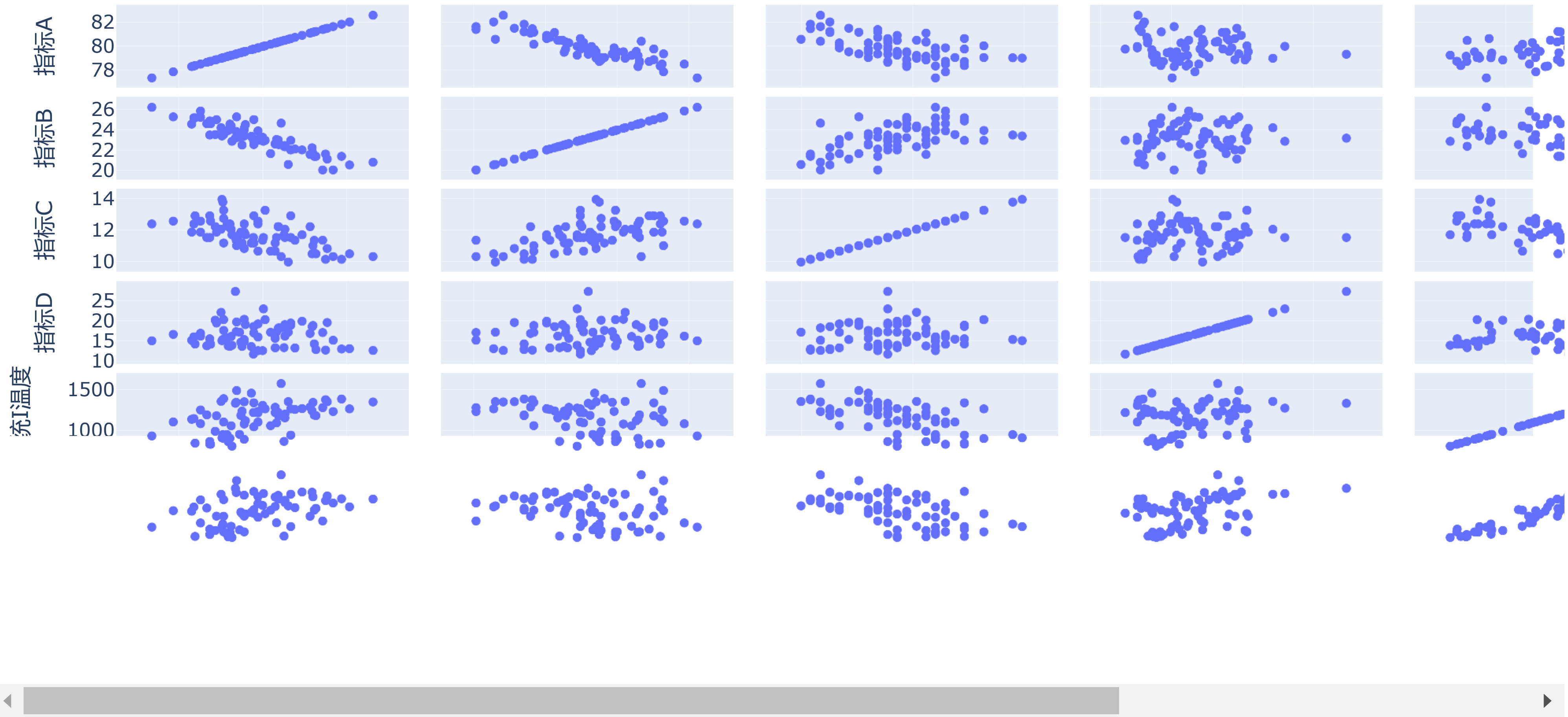
基于多元线性回归的系统II温度 (Temperature of system II)预测模型



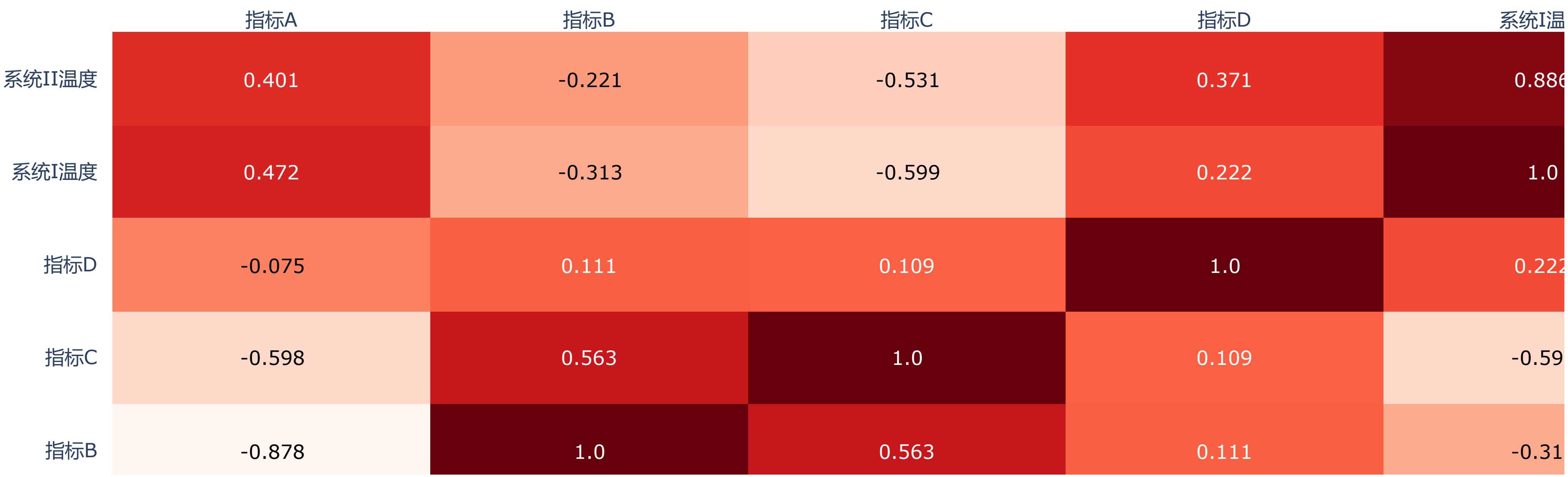
基于多元线性回归的系统温度预测模型



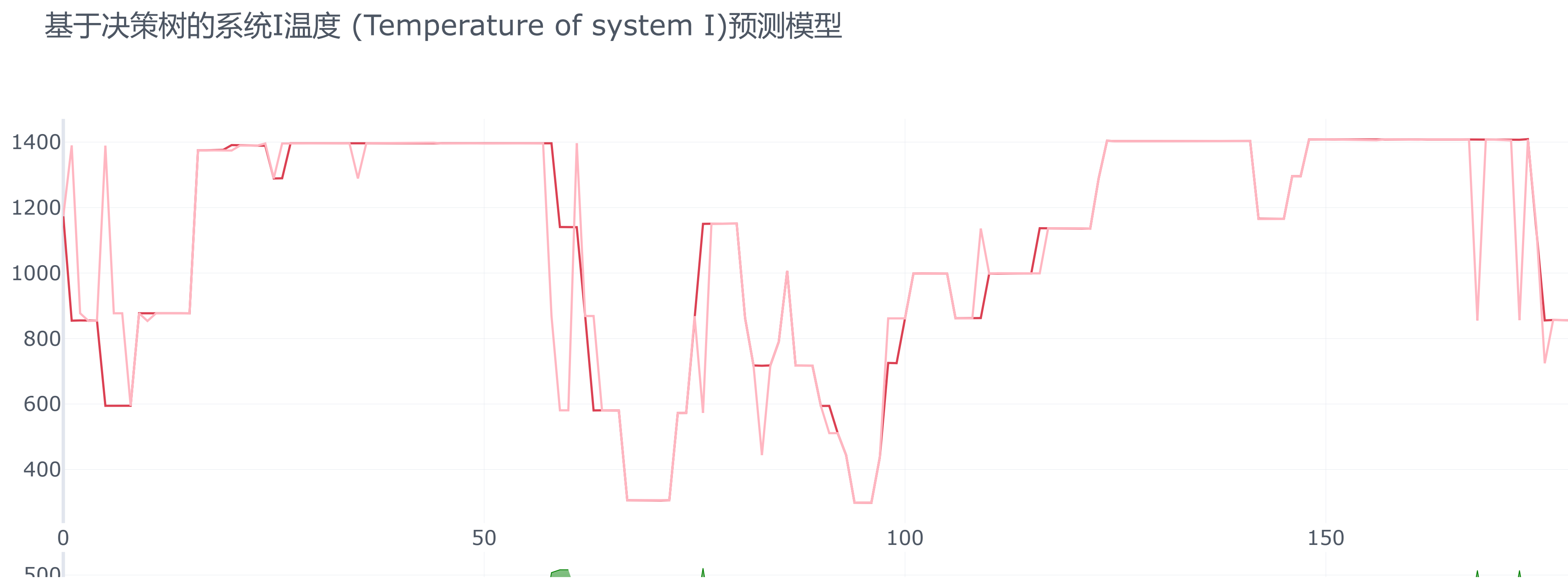
多元线性回归预测温度与指标的矩阵散点图



多元线性回归预测温度与指标的相关系数热力图



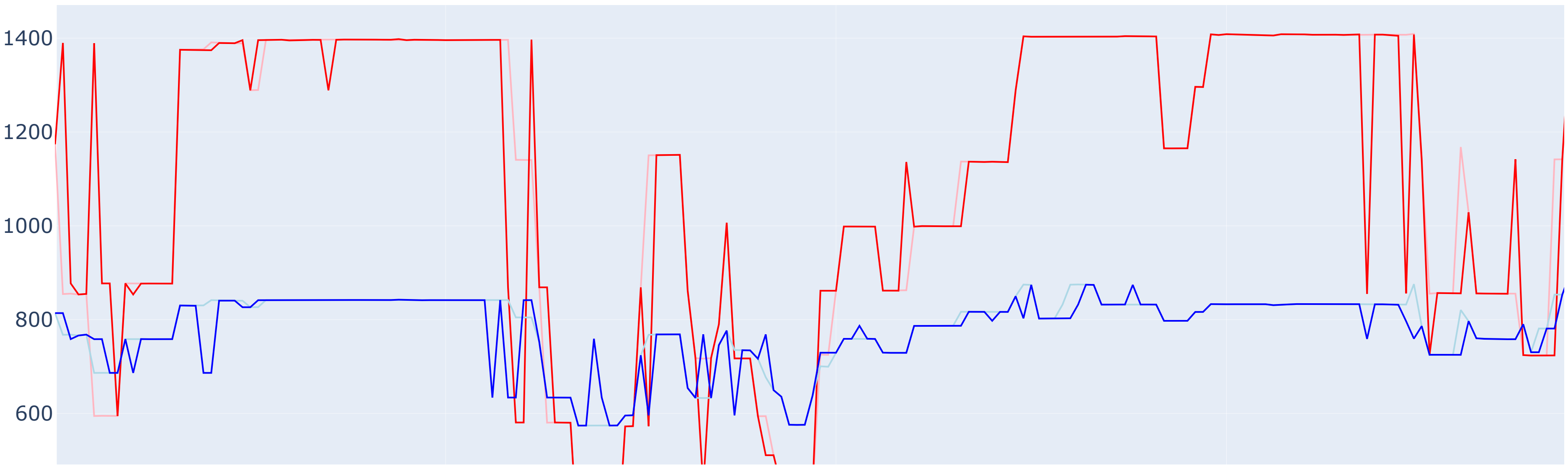

```
系统I温度:
灰色关联度:
[0.70941952 0.64707675 0.64223381 0.64418555]

系统II温度:
灰色关联度:
[0.86825905 0.78983481 0.75940625 0.73559382]

决策树 :

accuracy: 0.8848045843508983
MSE: 58348.09547323943 MAE: 125.91295774647887 R2: 0.45572787822988137
预测结果:  [1028.67 1028.67]
```



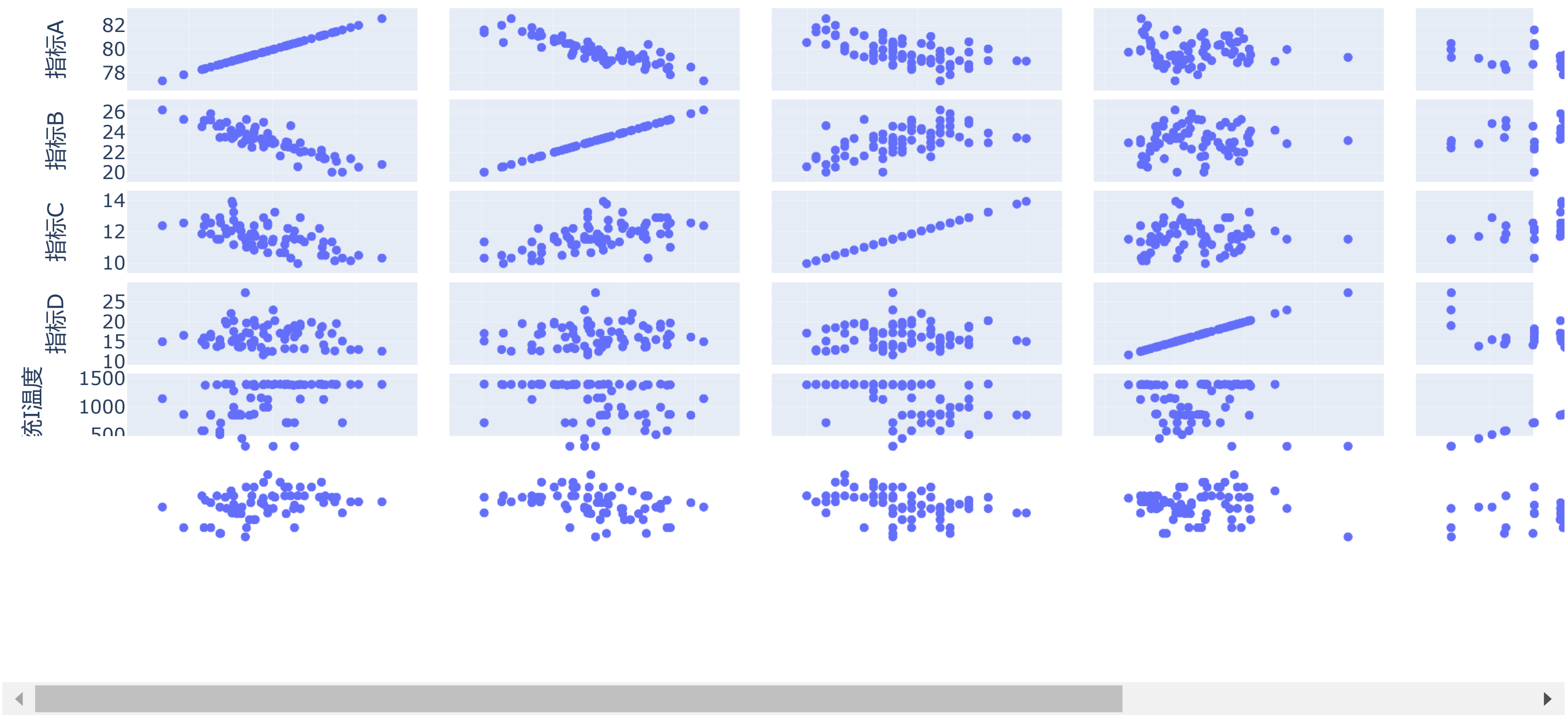

```
accuracy: 0.9568313827242567
MSE: 4963.522398591549 MAE: 41.45338028169012 R2: 0.34543392699400377
预测结果:  [595.57 574.42]
```

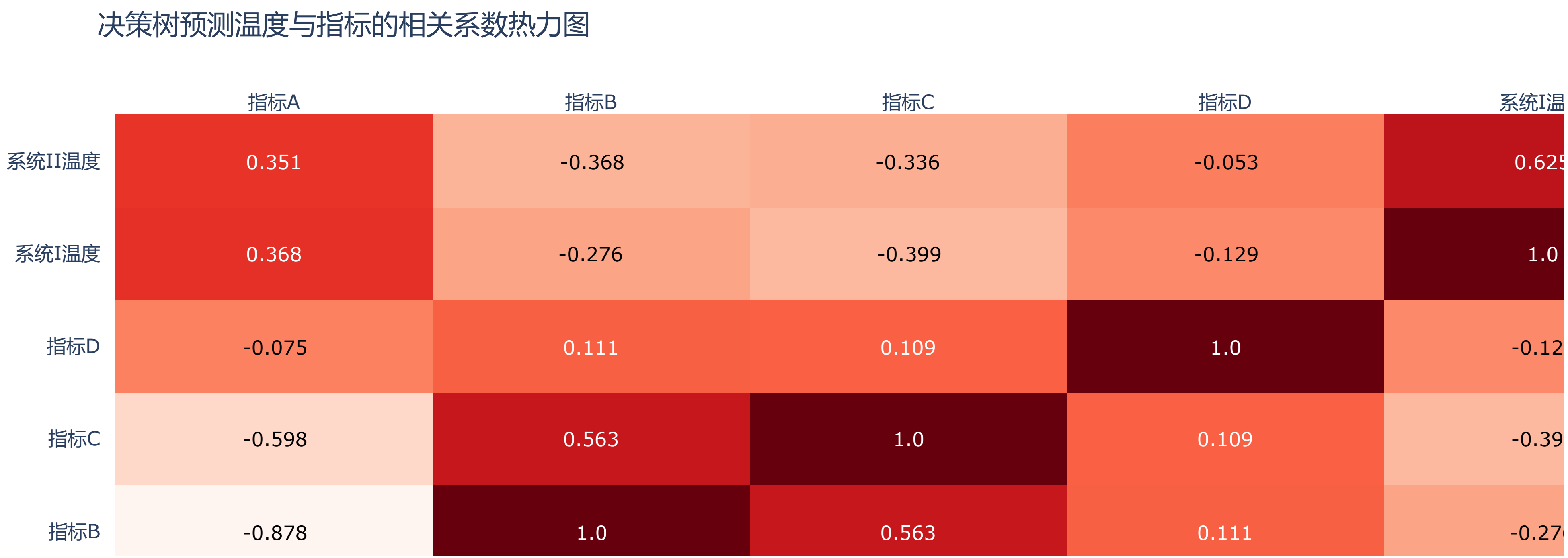
基于决策树的系统II温度 (Temperature of system II)预测模型

基于决策树的系统温度预测模型



决策树预测温度与指标的矩阵散点图





系统I温度:

灰色关联度:

[0.74050733 0.71993199 0.7070888 0.74321624]

系统II温度:

灰色关联度:

[0.85728006 0.80723298 0.78897056 0.77112458]

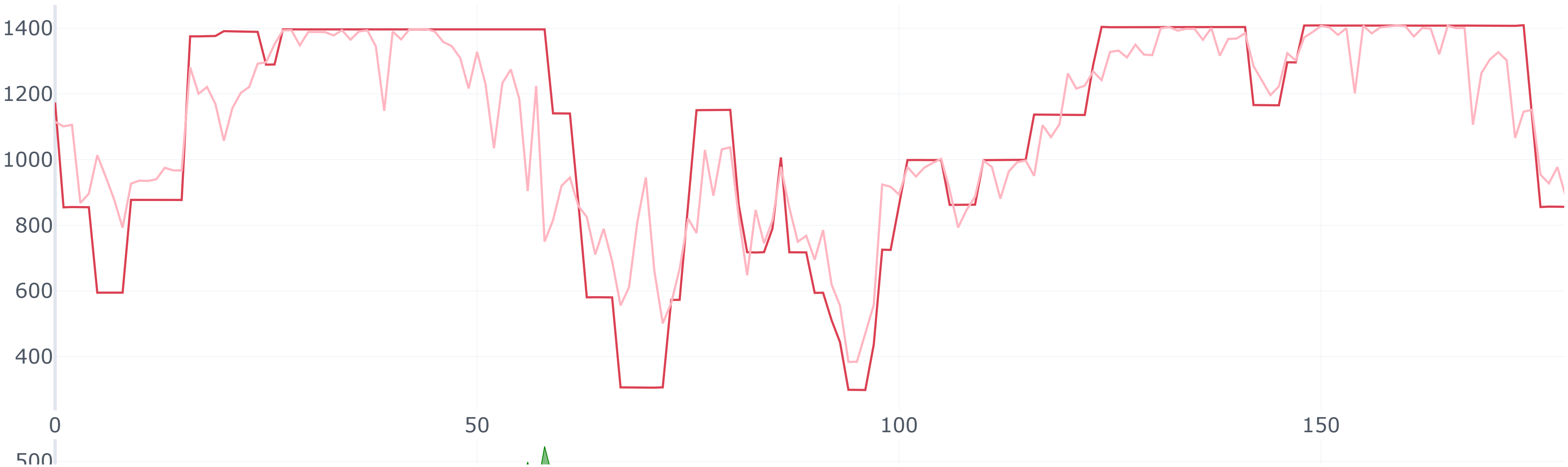
随机森林 :

accuracy: 0.8151595491593185

MSE: 51581.94969618424 MAE: 173.18280774647883 R2: 0.5188426121113454

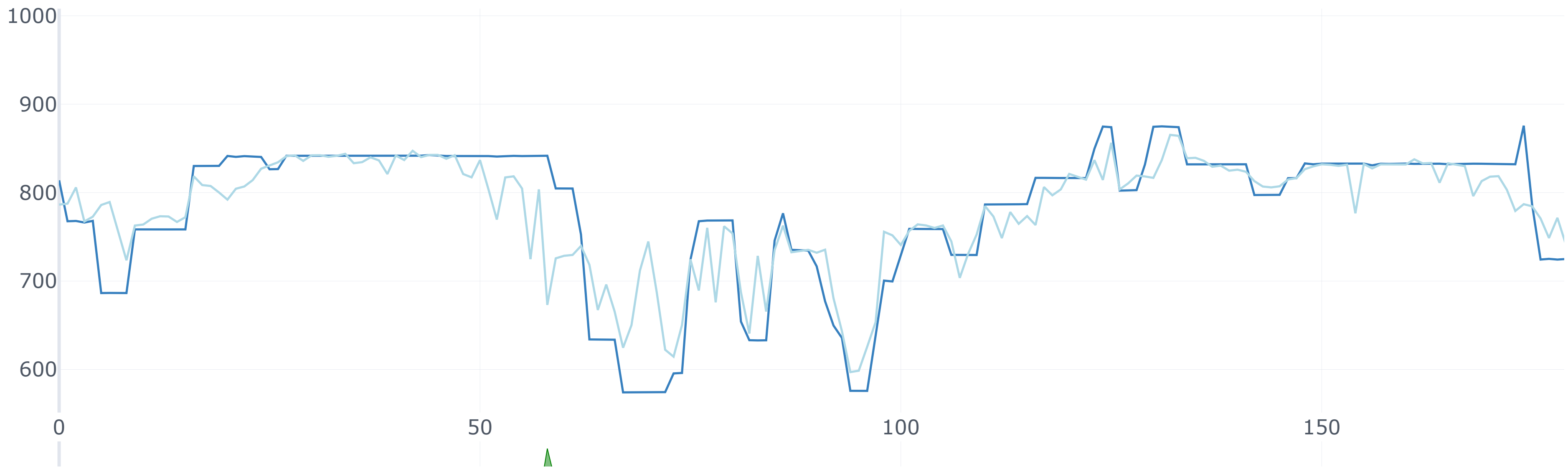
预测结果: [1251.7375 1152.7035]

基于随机森林的系统I温度 (Temperature of system I)预测模型

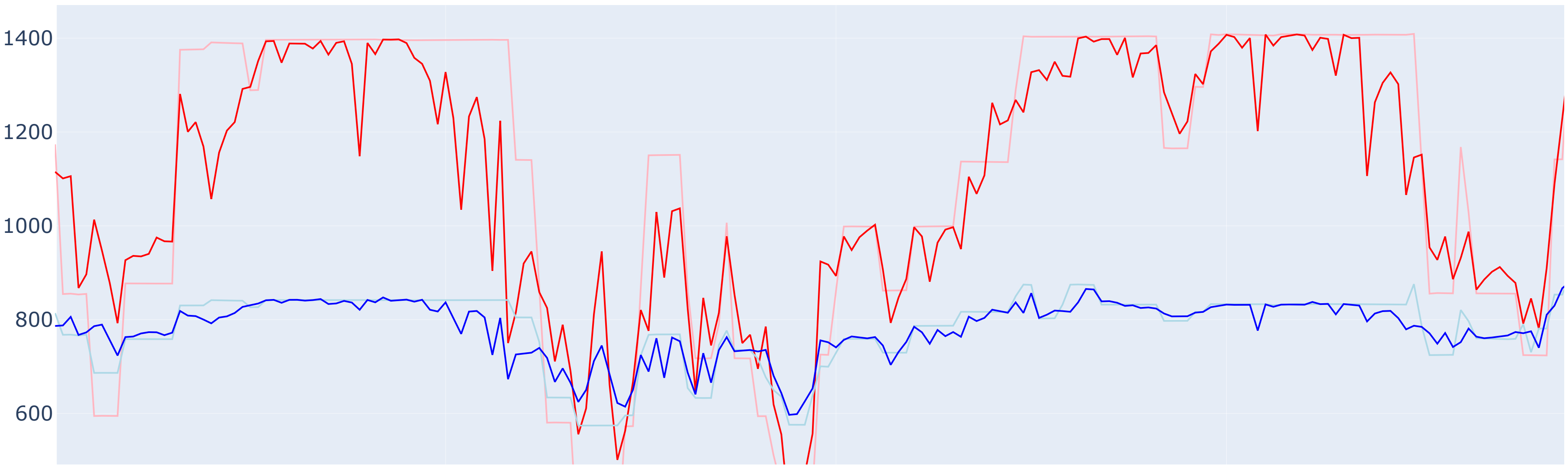


accuracy: 0.9539434035028242
MSE: 3388.2626171911224 MAE: 43.25783239436626 R2: 0.5531718047092631
预测结果： [829.3569 808.3618]

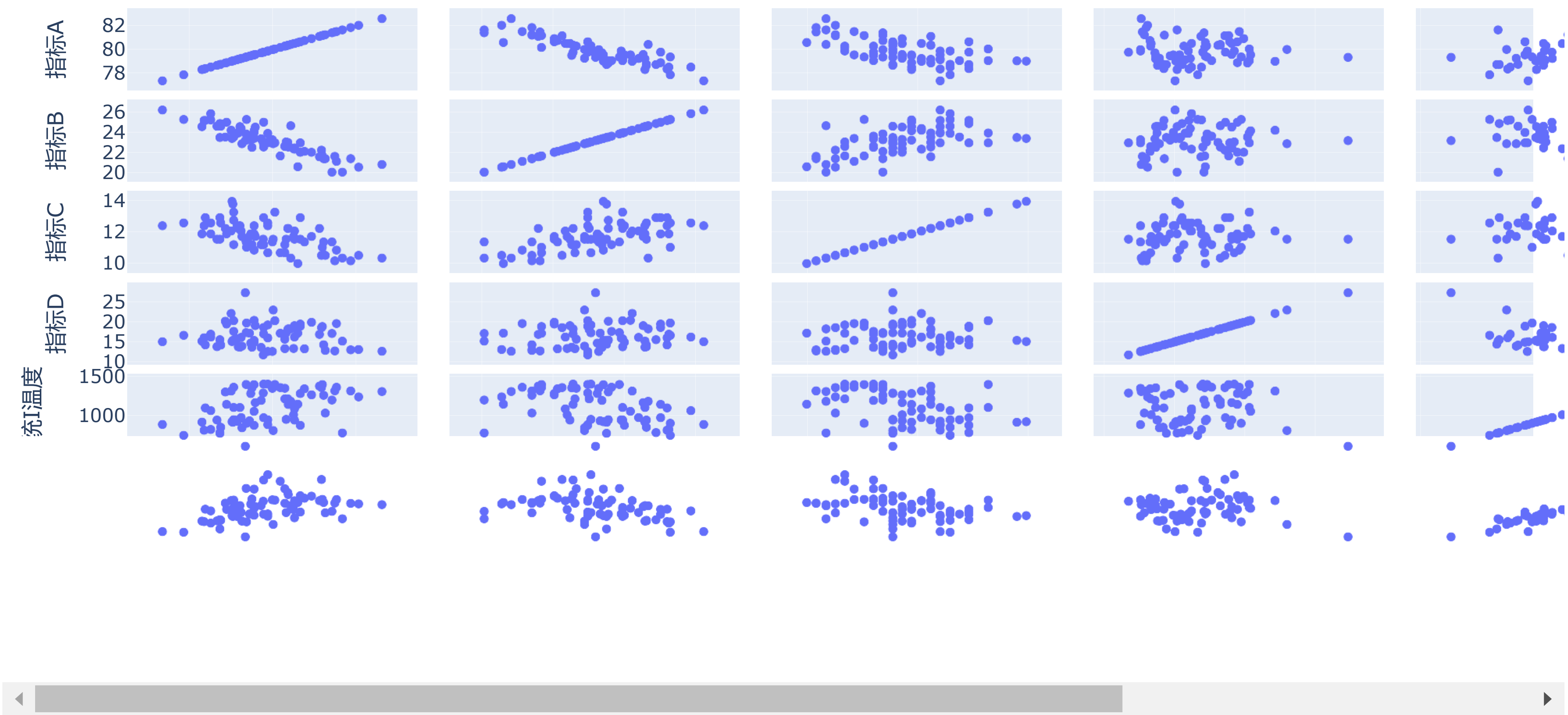
基于随机森林的系统II温度 (Temperature of system II)预测模型

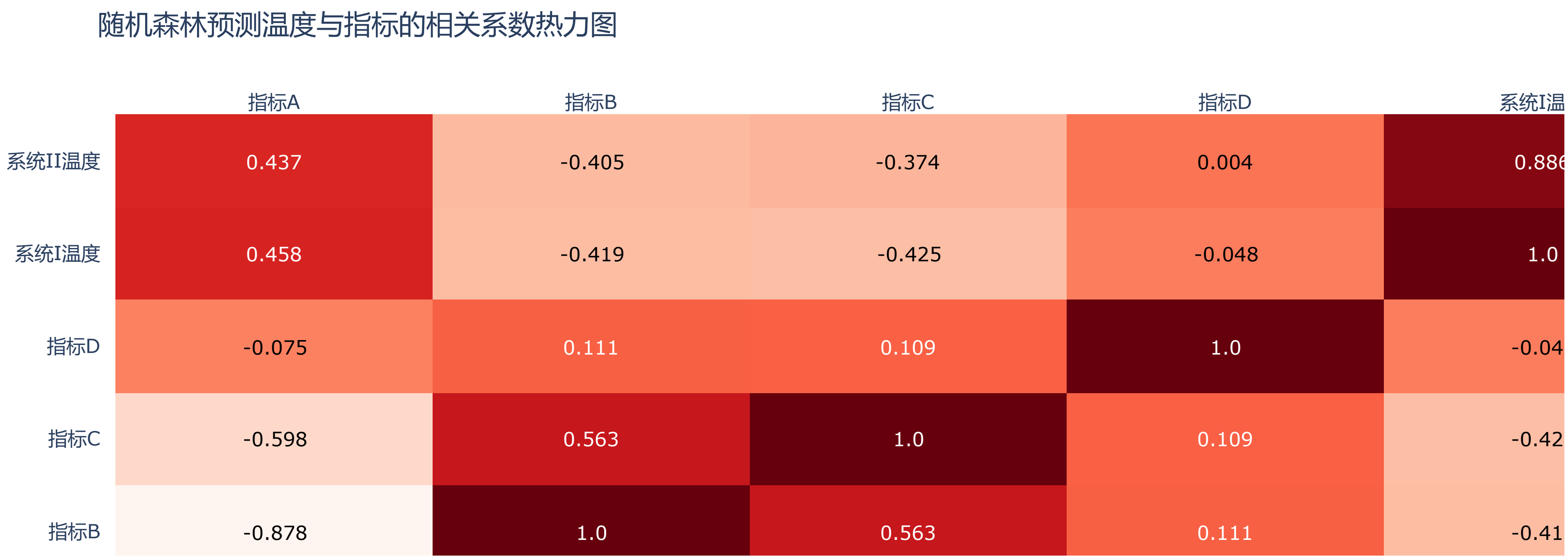


基于随机森林的系统温度预测模型



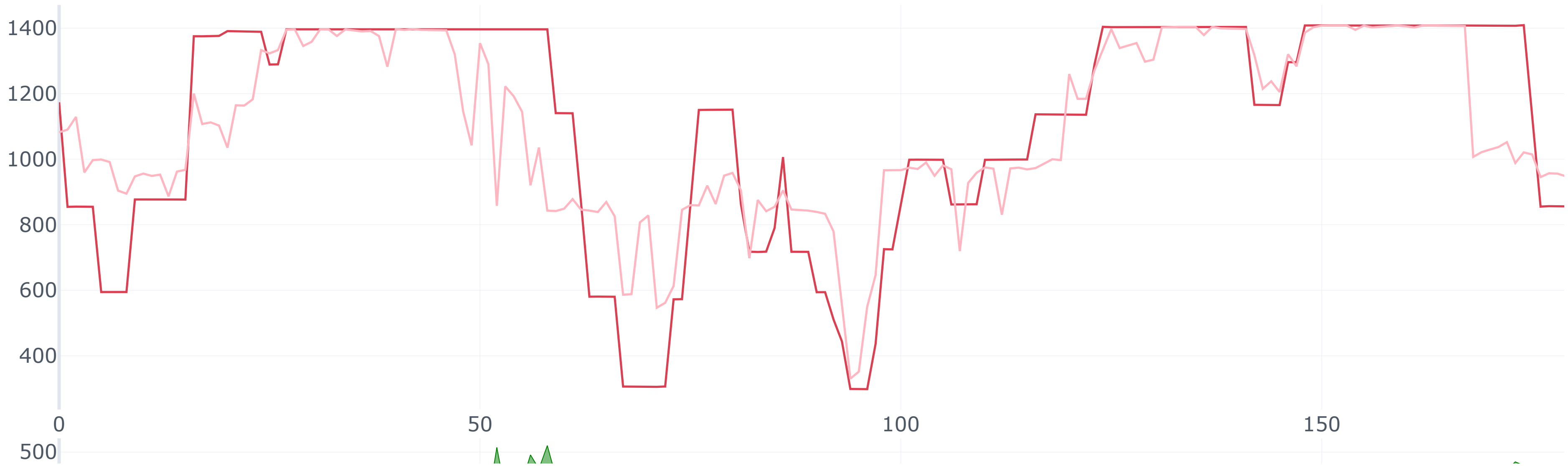
随机森林预测温度与指标的矩阵散点图



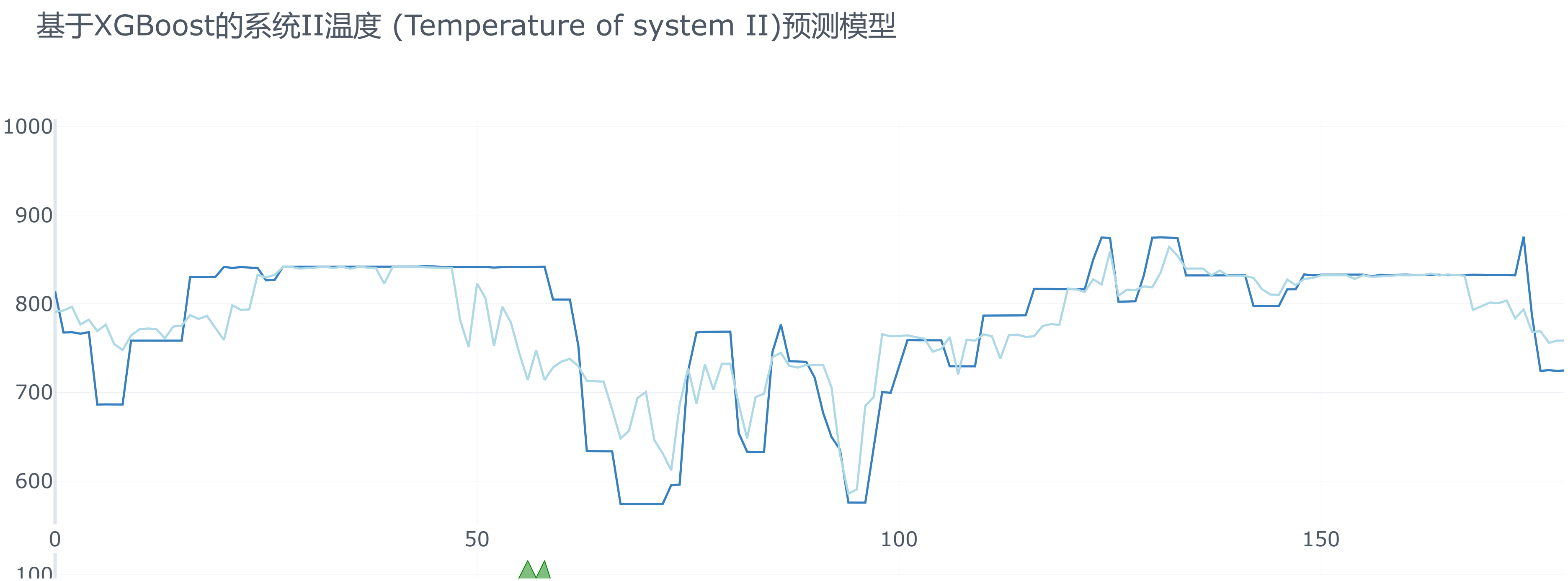



```
系统I温度:
灰色关联度:
[0.78209704 0.74668038 0.7369901  0.76563996]

系统II温度:
灰色关联度:
[0.88908324 0.81973345 0.81252091 0.77987629]

XGBoost :

accuracy: 0.8077794637943121
MSE: 58744.46985501162 MAE: 182.625312018596 R2: 0.4520304906110959
预测结果:  [1379.749  1251.7885]
```

基于XGBoost的系统I温度 (Temperature of system I)预测模型

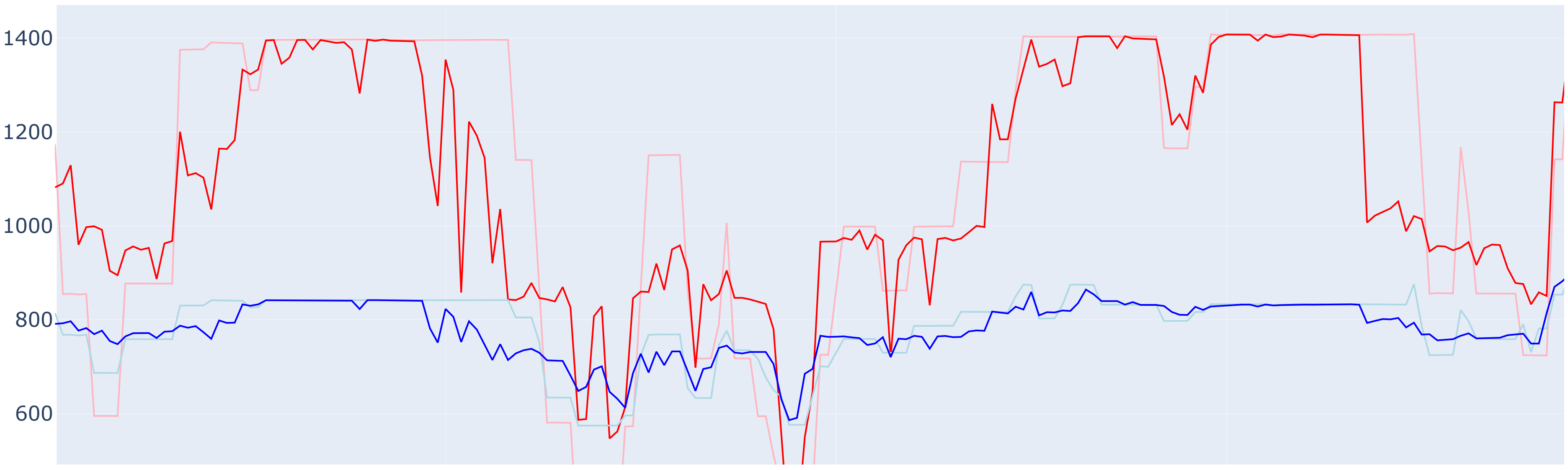


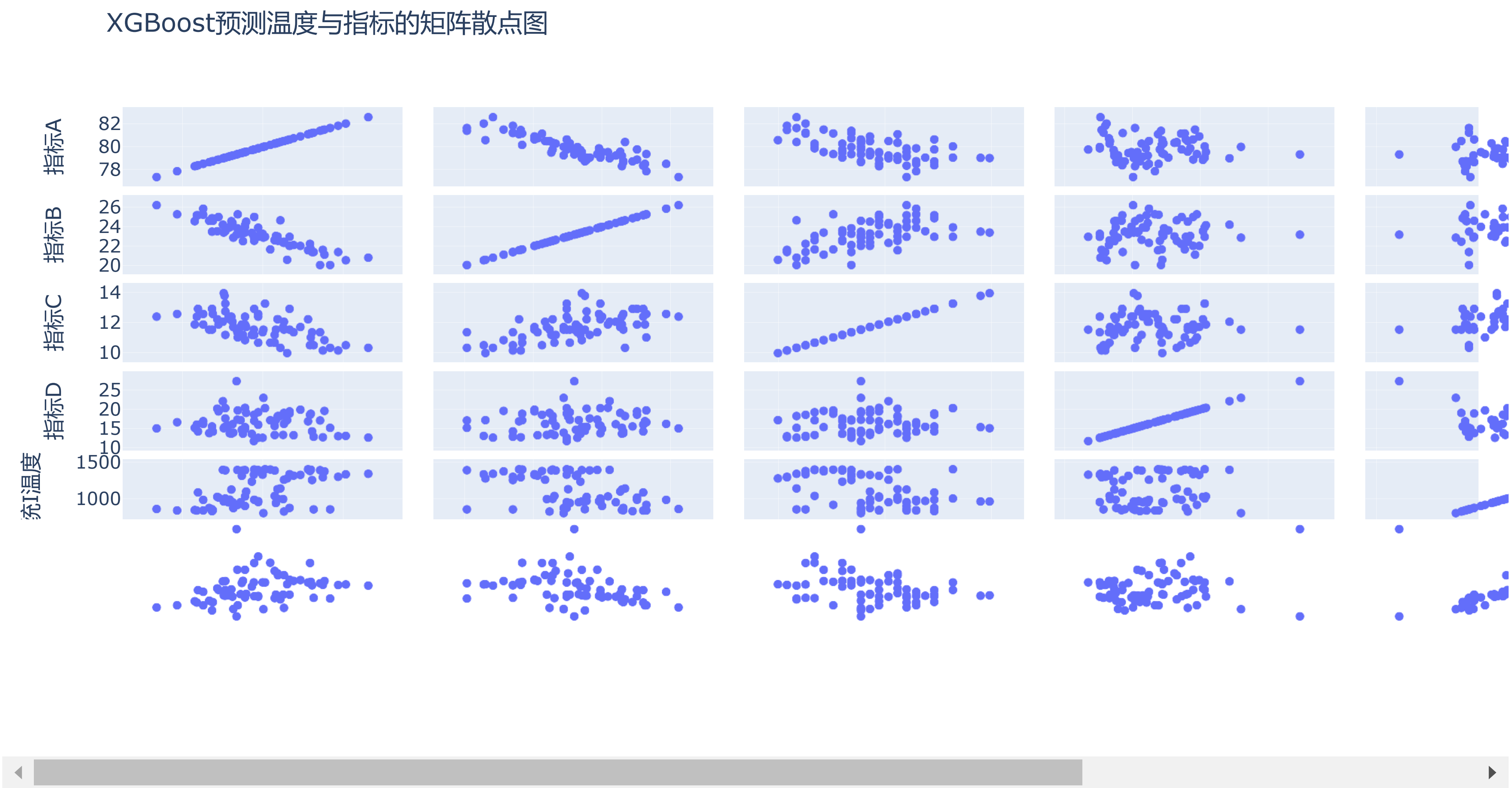

```
accuracy: 0.9556860830523095
MSE: 3036.700655104858 MAE: 42.053812651298415 R2: 0.5995341487185069
预测结果:  [823.028 794.621]
```

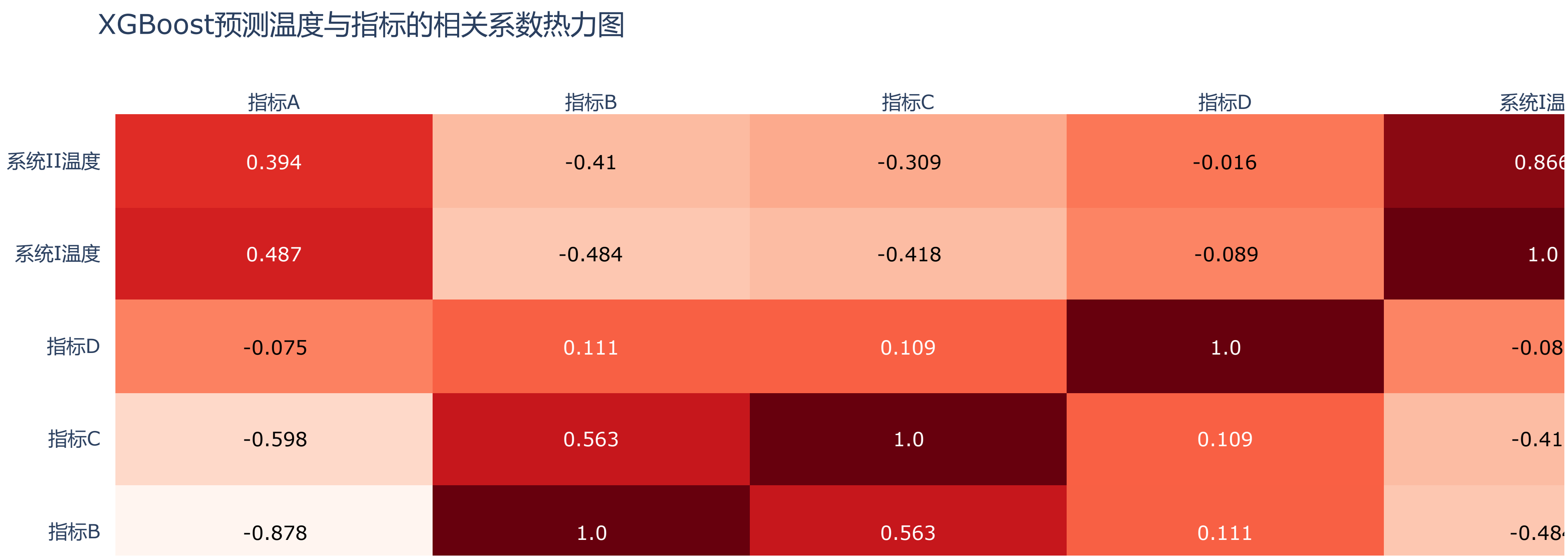


基于XGBoost的系统II温度 (Temperature of system II)预测模型
1000
900
800
700
600
0
50
100
150
100

基于XGBoost的系统温度预测模型




XGBoost预测温度与指标的矩阵散点图
指标A
82
80
78
指标B
26
24
22
20
指标C
14
12
10
指标D
25
20
15
10
预测温度
1500
1000



系统I温度:
灰色关联度:
[0.77401751 0.73659772 0.72989512 0.75327141]

系统II温度:
灰色关联度:
[0.88451616 0.81453299 0.80912308 0.78086379]

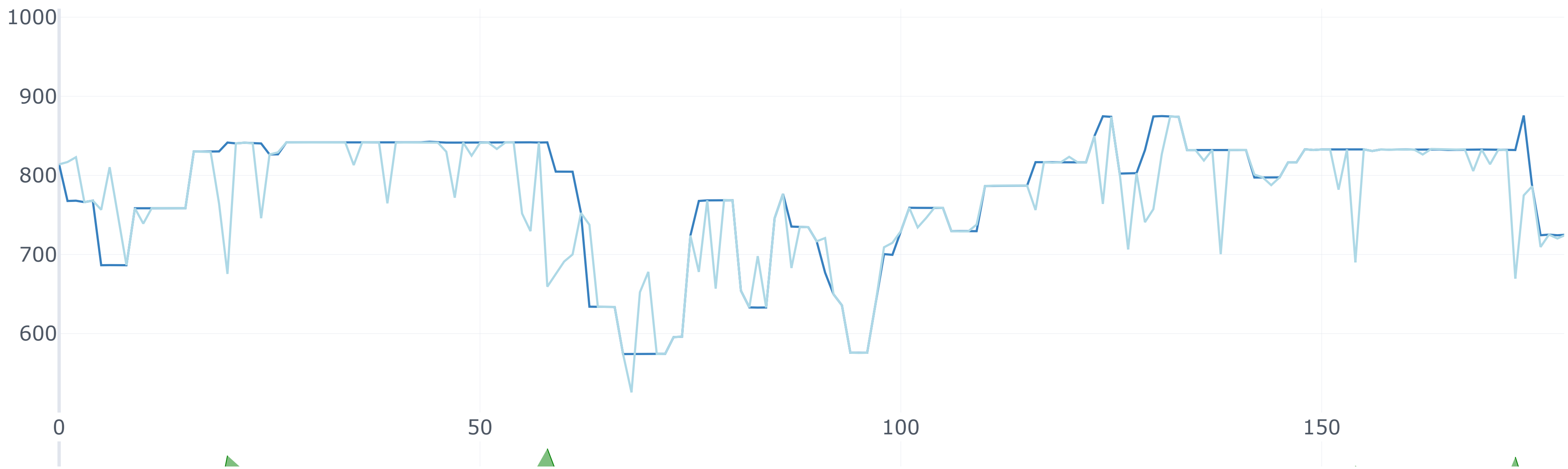
RBF :

accuracy: 0.8133333743708631
MSE: 62441.28197591328 MAE: 184.14521386725897预测结果: [846.07515545 767.0030533]

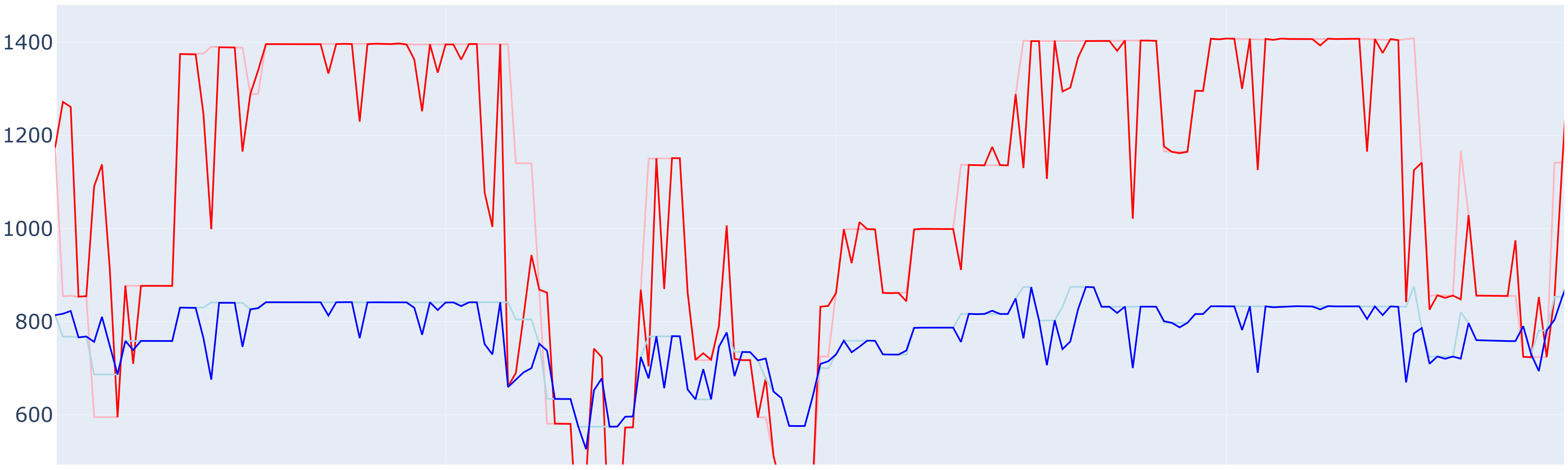
基于RBF的系统I温度 (Temperature of system I)预测模型

accuracy: 0.939522486615392
MSE: 5713.793359435118 MAE: 59.75171078232522预测结果: [612.18208285 566.27583365]

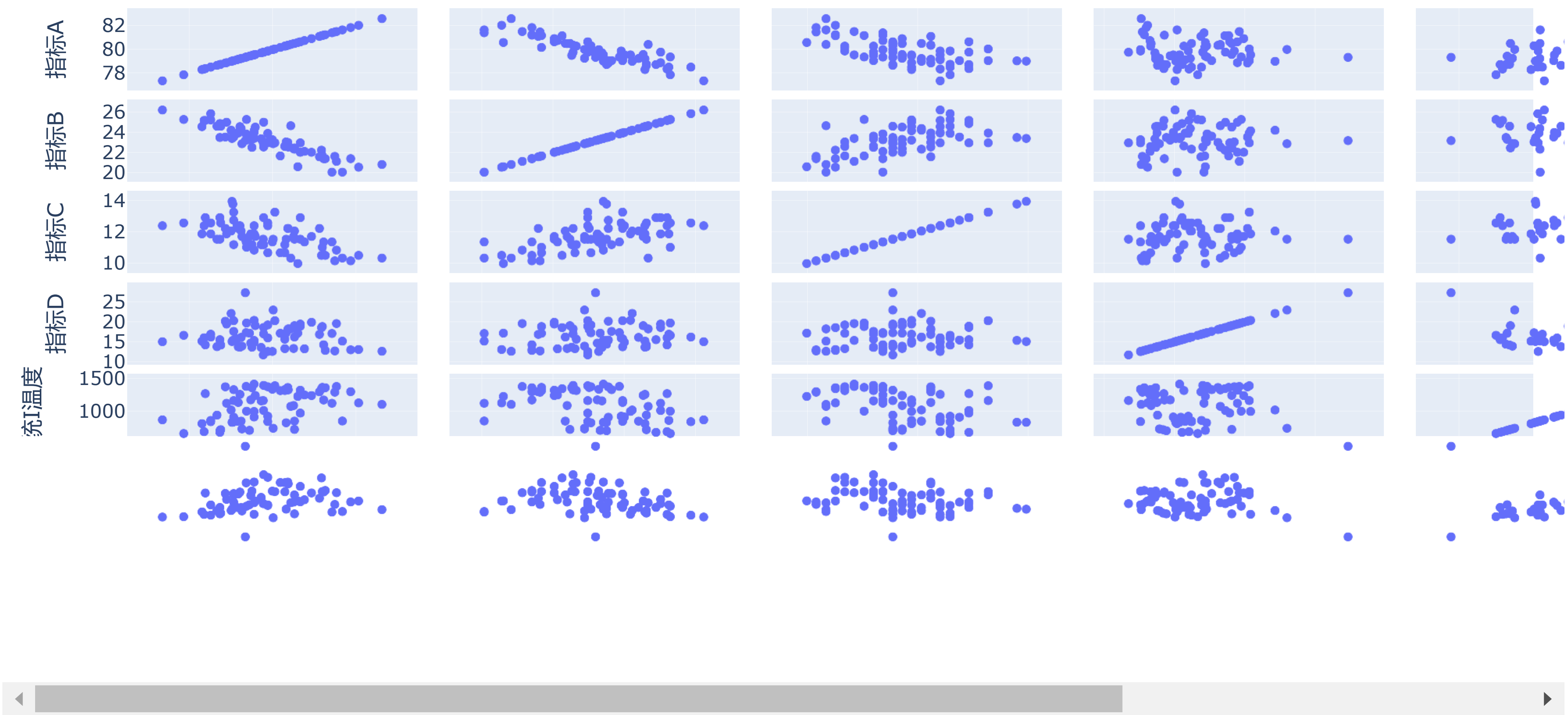
基于RBF的系统II温度 (Temperature of system II)预测模型

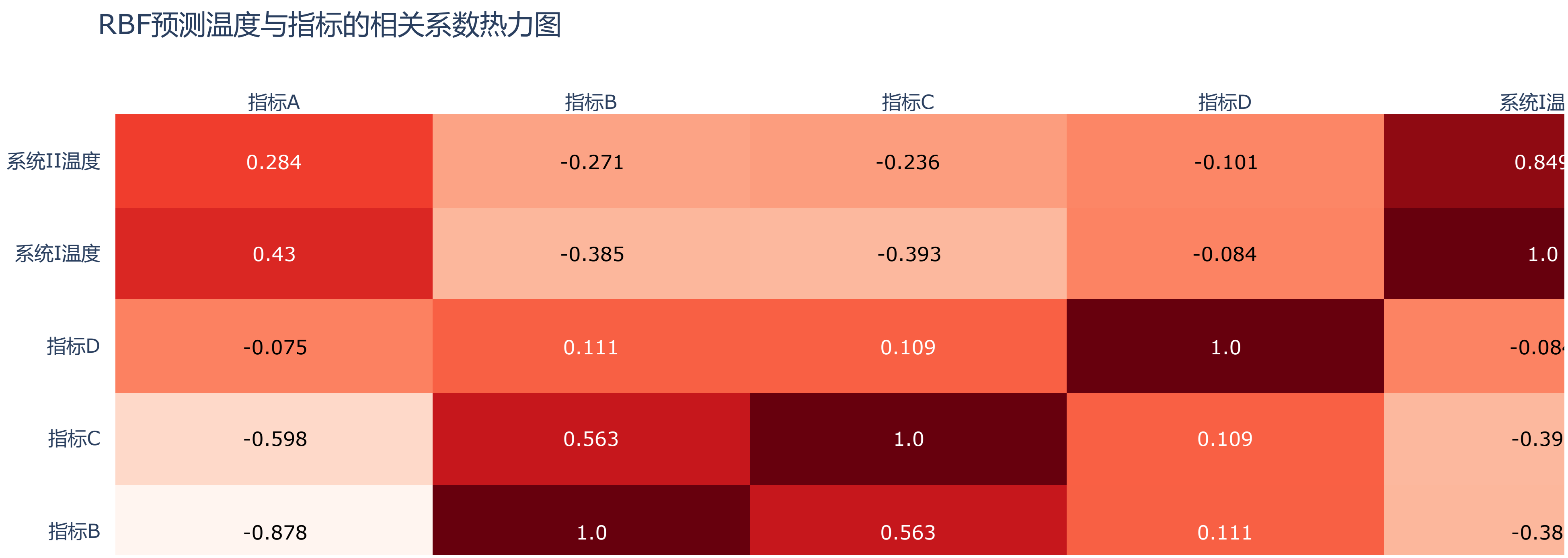


基于RBF的系统温度预测模型



RBF预测温度与指标的矩阵散点图






```
系统I温度:
灰色关联度:
[0.76966827 0.73766116 0.73568668 0.74454208]

系统II温度:
灰色关联度:
[0.86783797 0.82764907 0.82136425 0.78910302]
```

8行，废废了

BP

效果不好，不建议运行

In [15]:
```
hidden_num = [12, 20, 8]
lr = 0.01
epoch = 1000
optimizer = 'adam'
normalization = False

th = 0.5
con = 0.2
test_size = 0.3
random_state = 10

def run_BP(X=X, Ys=Ys, index_num=index_num):
    data_to_predict = np.array([
        [79.17, 22.72, 10.51, 17.05, 57.5, 108.62, 44.5, 20.09, ],
        [80.10, 23.34, 11.03, 13.29, 57.5, 108.62, 44.5, 20.09, ],
    ])

    data = []
    print("BP神经网络：")
    for i in range(index_num):
        Y = Ys.iloc[:, i]
        xtrain, xtest, ytrain, ytest = train_test_split(
            np.array(X, dtype=float), np.array(Y, dtype=float),
            test_size=0.3,
            random_state=10,
            shuffle=True,
        )

        bp = BP(
            X.shape[1], hidden_num, 1,
            lr=lr,
            epoch=epoch,
            optimizer=optimizer,
            normalization=normalization,
        )
        bp.train(xtrain, ytrain)
        y_pre = bp.predict(xtest).cpu().detach().numpy()

        acc = predict_accuracy(ytest[:, None], y_pre, type=1, th=th, con=con)  # 回归
        print("accuracy:", acc)
        print("预测结果：", bp.predict(data_to_predict))
#         print(mean_squared_error(y_true=ytest[:, None], y_pred=y_pre))
```

```
        # todo 画图
        Yhat = bp.predict(
            np.array(X, dtype=float)).cpu().detach().numpy()
        data.append(go.Scatter(
            x=data_part1.iloc[:, 0], y=Y,
            name=index_name[i] + "-真实值",
            line=dict(color=index_colors[i * 2]),
        )
        data.append(go.Scatter(
            x=data_part1.iloc[:, 0], y=np.squeeze(Yhat),
            name=index_name[i] + "-预测值",
            line=dict(color=index_colors[i * 2 + 1]),
        )

        # todo 画图：点差图
        cols = str(Y.name)
        Yhat = pd.DataFrame(Yhat)
        Y.index = [i for i in range(len(Y))]
        Y_data = pd.concat([Y, Yhat], axis=1)
        Y_data.columns = ["真实值", "预测值"]
        Y_data.figure(
            kind='spread',
            color=[index_colors[i * 2 + 1], index_colors[i * 2]],
            title='基于BP神经网络的指标预测模型—' + cols,
        ).write_image('./img/问题2-基于BP神经网络的' + cols + '预测模型.svg')
        Y_data.iplot(
            kind='spread',
            color=[index_colors[i * 2 + 1], index_colors[i * 2]],
            title='基于BP神经网络的指标预测模型—' + cols,
        )
    fig = go.Figure(data=data)

    annotations = []
    annotations.append(dict(
        x=0.5, y=-0.1,
        xref='paper', yref='paper',
        xanchor='center', yanchor='top',
        text='时间',
        font=dict(size=16),
        showarrow=False,
    ))
    fig.update_layout(
        title='基于BP神经网络的系统温度预测模型',
        annotations=annotations,
    )
    fig.write_image('./img/问题2-基于BP神经网络的系统温度预测模型.svg')
```

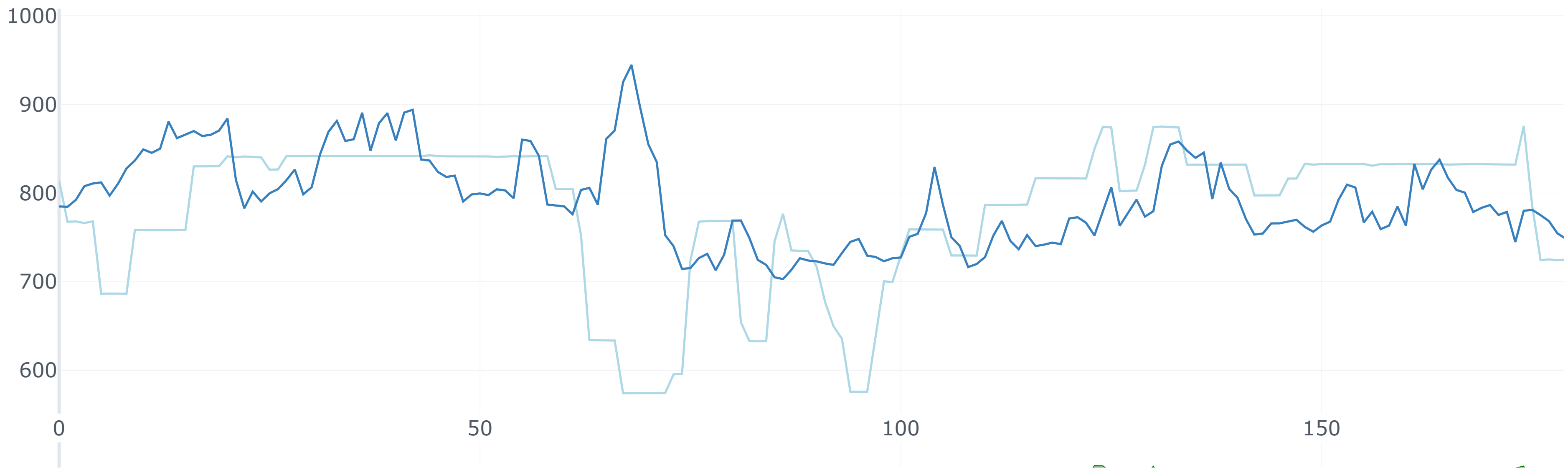
```python
    fig.show()
    return None

run_BP()
```

```
BP神经网络:
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                   [64, 12]             108
            Linear-2                   [64, 12]             108
              ReLU-3                   [64, 12]               0
              ReLU-4                   [64, 12]               0
            Linear-5                   [64, 20]             260
              ReLU-6                   [64, 20]               0
              ReLU-7                   [64, 20]               0
            Linear-8                    [64, 8]             168
              ReLU-9                    [64, 8]               0
             ReLU-10                    [64, 8]               0
           Linear-11                    [64, 1]               9
           Linear-12                    [64, 1]               9
================================================================
Total params: 662
Trainable params: 662
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.07
Params size (MB): 0.00
Estimated Total Size (MB): 0.07
----------------------------------------------------------------
epoch: 999, train loss: 146838.32, eval loss: 42733.27: 100%|██████████| 1000/1000 [00:13<00:00, 73.74it/s]
accuracy: 0.7388629357580284
预测结果:  tensor([[1057.2500],
        [1037.6388]], device='cuda:0', grad_fn=<AddmmBackward0>)
```

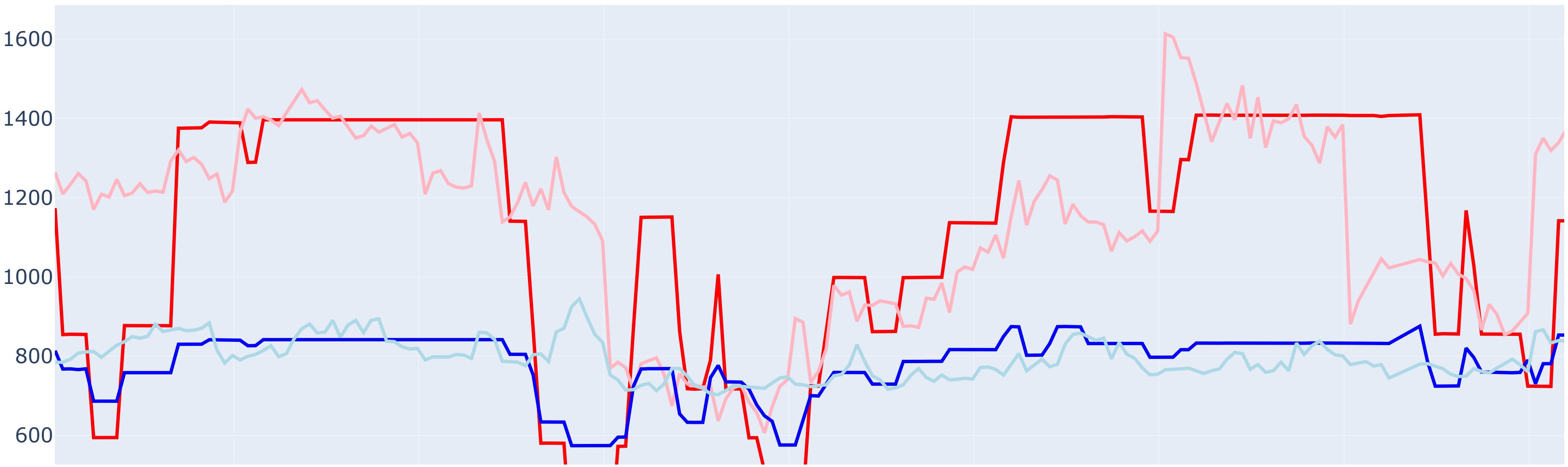
基于BP神经网络的指标预测模型——系统I温度 (Temperature of system I)

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                   [64, 12]             108
            Linear-2                   [64, 12]             108
              ReLU-3                   [64, 12]               0
              ReLU-4                   [64, 12]               0
            Linear-5                   [64, 20]             260
              ReLU-6                   [64, 20]               0
              ReLU-7                   [64, 20]               0
            Linear-8                    [64, 8]             168
              ReLU-9                    [64, 8]               0
             ReLU-10                    [64, 8]               0
           Linear-11                    [64, 1]               9
           Linear-12                    [64, 1]               9
================================================================
Total params: 662
Trainable params: 662
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.07
Params size (MB): 0.00
Estimated Total Size (MB): 0.07
----------------------------------------------------------------
epoch: 999, train loss: 15489.72, eval loss: 5071.24: 100%|████████████████████| 1000/1000 [00:12<00:00, 78.65it/s]
accuracy: 0.9279832851911516
预测结果： tensor([[780.8580],
        [748.0912]], device='cuda:0', grad_fn=<AddmmBackward0>)
```

基于BP神经网络的指标预测模型——系统II温度 (Temperature of system II)



基于BP神经网络的系统温度预测模型



In [ ]: