问题4-思路1

不管是否可以达到，一律按可以达到处理（说好听点，就是假设可以 doge），然后直接预测温度

In [1]:

```
# TODO import
import hmz
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
import graphviz
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score

from colorama import Fore
def color(text):
    return Fore.RED + text + Fore.RESET
```

In [2]:

```
file_path = './附件2(Attachment 2)2022-51MCM-Problem B.xlsx'
sheet1 = pd.read_excel(
    io=file_path,
```

```python
    index_col=None,
    sheet_name='温度(temperature)', )
sheet2 = pd.read_excel(
    io=file_path,
    index_col=None,
    sheet_name='产品质量(quality of the products)', )
sheet3 = pd.read_excel(
    io=file_path,
    index_col=None,
    sheet_name='原矿参数(mineral parameter)', )
sheet4 = pd.read_excel(
    io=file_path,
    index_col=None,
    sheet_name='过程数据(process parameter)', )
```

In [3]:

```python
# todo 随便检查一下数据
sheet2
```

Out[3]:

	时间 (Time)	指标A (index A)	指标B (index B)	指标C (index C)	指标D (index D)
0	2022-01-25 00:50:00	78.31	23.66	12.24	17.81
1	2022-01-25 01:50:00	78.46	23.88	12.41	17.99
2	2022-01-25 02:50:00	79.08	23.52	12.41	17.86
3	2022-01-25 03:50:00	79.29	22.94	11.72	17.86
4	2022-01-25 04:50:00	79.95	21.42	10.68	17.63
...	...	...	...	...	...
1739	2022-04-07 19:50:00	79.82	23.84	11.03	13.52
1740	2022-04-07 20:50:00	78.98	25.36	11.37	12.85
1741	2022-04-07 21:50:00	78.86	25.40	11.37	11.42
1742	2022-04-07 22:50:00	79.10	25.58	11.37	11.55
1743	2022-04-07 23:50:00	79.32	24.82	11.03	11.55

In [4]:

```python
from sklearn.decomposition import PCA

pd.DataFrame(PCA().fit_transform(sheet4.iloc[:, 1:]))
```

Out[4]:

	0	1	2	3
0	-51.648701	16.946603	0.141087	-1.472603e-16
1	-31.237172	5.546222	0.053911	1.706418e-14
2	-31.745872	-13.135754	-0.046181	-2.872148e-17
3	-53.811520	-7.711616	0.010853	-7.644242e-17
4	-35.967364	-11.206386	-0.030462	-3.954490e-17
...	...	...	...	...
614	4.236387	4.045971	0.001150	-3.411051e-19
615	14.598511	14.406331	0.043964	-2.448952e-18
616	13.716216	10.905797	0.026200	2.704134e-18
617	-3.231871	9.877502	0.041998	-2.408235e-17
618	-27.163606	8.395425	0.064145	-6.184941e-17

准备数据

表1——温度(temperature)

In [5]:

```
# todo 1 温度
sheet1_copy = sheet1.copy()
sheet1_copy.iloc[:, 0] = sheet1_copy.iloc[:, 0].astype('string').apply(lambda x: x[:-9])
data_part1 = pd.DataFrame([], columns=list(sheet1_copy.columns[1:]), dtype=np.float64)
for d in sheet1_copy.groupby(by='时间 (Time)'):
    data_part1.loc[d[0], :] = d[1].mean().values
print(data_part1.shape)
data_part1
```

```
(73, 2)
```

Out[5]:

	系统I温度 (Temperature of system I)	系统II温度 (Temperature of system II)
2022-01-25	1378.853377	955.675052
2022-01-26	1404.817708	943.595729
2022-01-27	1016.714986	861.417375
2022-01-28	601.506435	779.863899
2022-01-29	1055.616887	782.035552
...	...	...
2022-04-03	814.740889	797.645847
2022-04-04	471.278949	656.072465
2022-04-05	764.303243	785.466132
2022-04-06	591.403870	641.759421
2022-04-07	507.772234	631.842053

表2——产品质量(quality of the products)

In [6]:

```
# todo 2 产品质量
sheet2_copy = sheet2.copy()
def is_qualified(x):
    return 77.78 < x[1] < 80.33 and x[2] < 24.15 and x[3] < 17.15 and x[4] < 15.62
data_part2_step1 = pd.DataFrame(sheet2_copy.apply(is_qualified, axis=1))
data_part2_step1.columns = ['是否合格']
data_part2_step1.insert(0, column='时间 (Time)', value=sheet2_copy.iloc[:, 0], )

data_part2 = pd.DataFrame([], columns=['合格率'], dtype=np.float64)
data_part2_step1.iloc[:, 0] = data_part2_step1.iloc[:, 0].astype('string').apply(lambda x: x[:-9])
for d in data_part2_step1.groupby(by='时间 (Time)'):
    data_part2.loc[d[0], :] = d[1].mean().values[0]
data_part2.astype('float')
print(data_part2.shape)
data_part2
```

```
(73, 1)
```

Out[6]:

	合格率
2022-01-25	0.166667
2022-01-26	0.208333
2022-01-27	0.000000
2022-01-28	0.000000
2022-01-29	0.000000
...	...
2022-04-03	0.333333
2022-04-04	0.227273
2022-04-05	0.666667
2022-04-06	0.291667
2022-04-07	0.083333

表3——原矿参数(mineral parameter)

In [7]:

```
# todo 3 原矿参数
data_part3 = sheet3.iloc[:-4, :]
data_part3.iloc[:, 0] = data_part3.iloc[:, 0].astype('string')
data_part3.index = data_part3.iloc[:, 0].values
print(data_part3.shape)
data_part3
```

```
(73, 5)
```

Out[7]:

	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)
2022-01-25	2022-01-25	55.26	108.03	43.29	20.92
2022-01-26	2022-01-26	55.28	102.38	46.13	20.10
2022-01-27	2022-01-27	54.04	102.21	47.94	21.30
2022-01-28	2022-01-28	56.43	112.74	43.54	20.14
2022-01-29	2022-01-29	54.89	109.21	43.60	21.64
...	...	...	...	...	...
2022-04-03	2022-04-03	54.30	101.07	47.10	20.65
2022-04-04	2022-04-04	53.98	89.87	51.14	20.67
2022-04-05	2022-04-05	52.95	88.09	52.89	21.59
2022-04-06	2022-04-06	56.13	103.83	46.67	20.35
2022-04-07	2022-04-07	54.40	105.14	49.03	20.82

表4——过程数据(process parameter)

In [8]:

```
# todo 4 过程数据
sheet4_copy = sheet4.copy()
sheet4_copy.iloc[:, 0] = sheet4_copy.iloc[:, 0].astype('string').apply(lambda x: x[:-9])
data_part4_step1 = sheet4_copy.groupby(by='时间 (Time)')

data_part4 = pd.DataFrame([], columns=list(sheet4_copy.columns)[1:], dtype=np.float64)
for d in data_part4_step1:
    data_part4.loc[d[0], :] = d[1].iloc[:, 1:].mean(axis=0).values

data_part4.astype(np.float64)
print(data_part4.iloc[:-4, :].shape)
data_part4
```

```
(73, 4)
```

Out[8]:

	过程数据1 (Process parameter 1)	过程数据2 (Process parameter 2)	过程数据3 (Process parameter 3)	过程数据4 (Process parameter 4)
2022-01-25	1.25	3.09	235.346250	157.388750
2022-01-26	1.25	3.09	240.648750	153.235000
2022-01-27	1.25	3.09	247.526250	155.666250
2022-01-28	1.25	3.09	257.313750	147.592500
2022-01-29	1.25	3.09	268.251250	149.742500
...	...	...	...	...
2022-04-07	1.25	3.09	296.355556	147.936667
2022-04-08	1.25	3.09	315.613750	141.091250
2022-04-09	1.25	3.09	281.596667	155.241111
2022-04-10	1.25	3.09	278.377500	155.655000
2022-04-11	1.25	3.09	269.025000	157.950000

In [9]:

```
X = pd.concat([data_part2, data_part3.iloc[:, 1:], data_part4.iloc[: -4, :], ], axis=1)
Ys = data_part1
```

In [10]:

```
X
```

Out[10]:

	合格率	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)	过程数据1 (Process parameter 1)	过程数据2 (Process parameter 2)	过程数据3 (Process parameter 3)	过程数据4 (Process parameter 4)
2022-01-25	0.166667	55.26	108.03	43.29	20.92	1.25	3.09	235.346250	157.388750
2022-01-26	0.208333	55.28	102.38	46.13	20.10	1.25	3.09	240.648750	153.235000
2022-01-27	0.000000	54.04	102.21	47.94	21.30	1.25	3.09	247.526250	155.666250
2022-01-28	0.000000	56.43	112.74	43.54	20.14	1.25	3.09	257.313750	147.592500
2022-01-29	0.000000	54.89	109.21	43.60	21.64	1.25	3.09	268.251250	149.742500
...	...	...	...	...	...	...	...	...	...
2022-04-03	0.333333	54.30	101.07	47.10	20.65	1.25	3.09	276.777500	147.698750
2022-04-04	0.227273	53.98	89.87	51.14	20.67	1.25	3.09	281.201250	142.853750
2022-04-05	0.666667	52.95	88.09	52.89	21.59	1.25	3.09	288.371429	150.355714
2022-04-06	0.291667	56.13	103.83	46.67	20.35	1.25	3.09	296.696250	152.365000
2022-04-07	0.083333	54.40	105.14	49.03	20.82	1.25	3.09	296.355556	147.936667

In [11]:

```
X.to_csv("quention3-X_data.csv")
Ys.to_csv("quention3-Y_data.csv")
```

预测是否可以达到合格-思路1

不管是否可以达到，一律按可以达到处理（说好听点，就是假设可以 doge），然后直接预测温度

In [12]:

```
# 04-10 04-11
test_size = 0.3
index_num = 2
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(
    [[0.8, 56.27, 111.38, 47.52, 20.26, 1.25, 3.09, 278.377500, 155.65500, ],
     [0.9, 56.71, 111.46, 46.67, 18.48, 1.25, 3.09, 269.025000, 157.950000, ],],
)
```

In [ ]:

```
```

In [13]:

```
from sklearn.metrics import mean_absolute_error as MAE
from sklearn.metrics import mean_squared_error as MSE
```

In [14]:

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

In [15]:

```
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')

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

        try:
            model.fit(xtrain, ytrain)
        except:
            model.train(xtrain, ytrain, lr=1e-2, init_method='kmeans', train_method=0, epoch=1000)  # RBF
        yhat = model.predict(xtest)

        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))
    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=1.5)),
    )
    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=1.5)),
    )

    # 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4-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4-基于' + model_name + '的系统温度预测模型.svg')
    fig.show()
    return model.predict(data_to_predict)
```

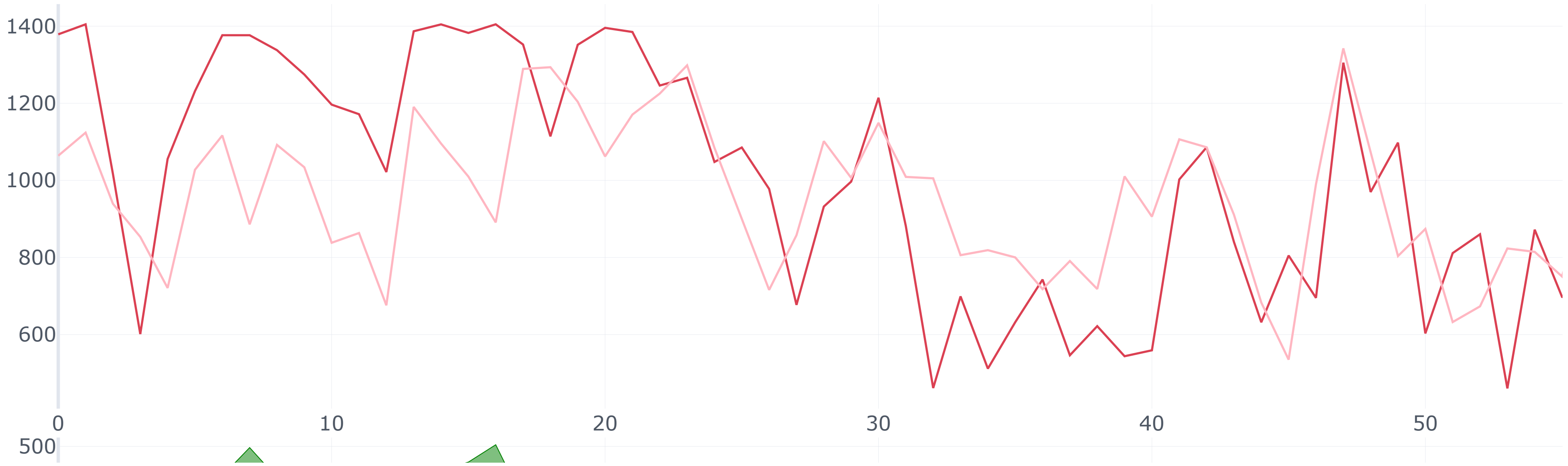
1. 使用逻辑回归

In [16]:

```
from sklearn.linear_model import Ridge, Lasso
from sklearn.linear_model import LinearRegression as LR
from sklearn.preprocessing import PolynomialFeatures as PF

models_lr = [
    LR(),
    Ridge(),
    Lasso(),
]

for model in models_lr[:1]:
    run_model("逻辑回归", model, )
```

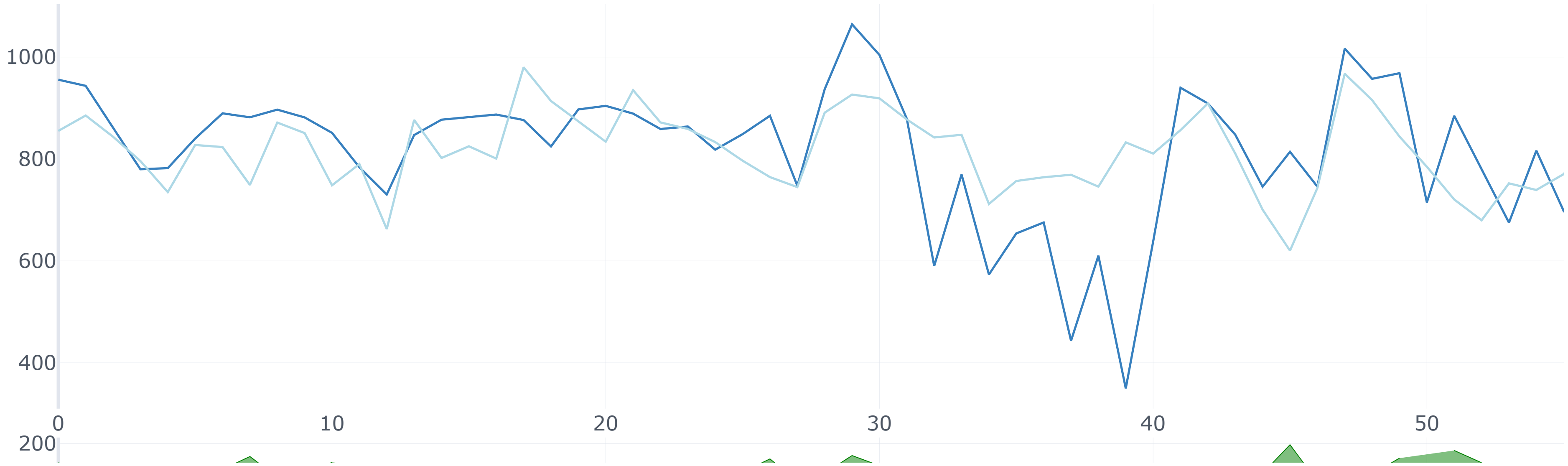
```
逻辑回归 ：

accuracy: 0.728910359813127
MSE: 79963.93139581602 MAE: 236.51616500792395 R2: 0.16568081243466226
预测结果:  [1074.80686405 1272.402887  ]
```

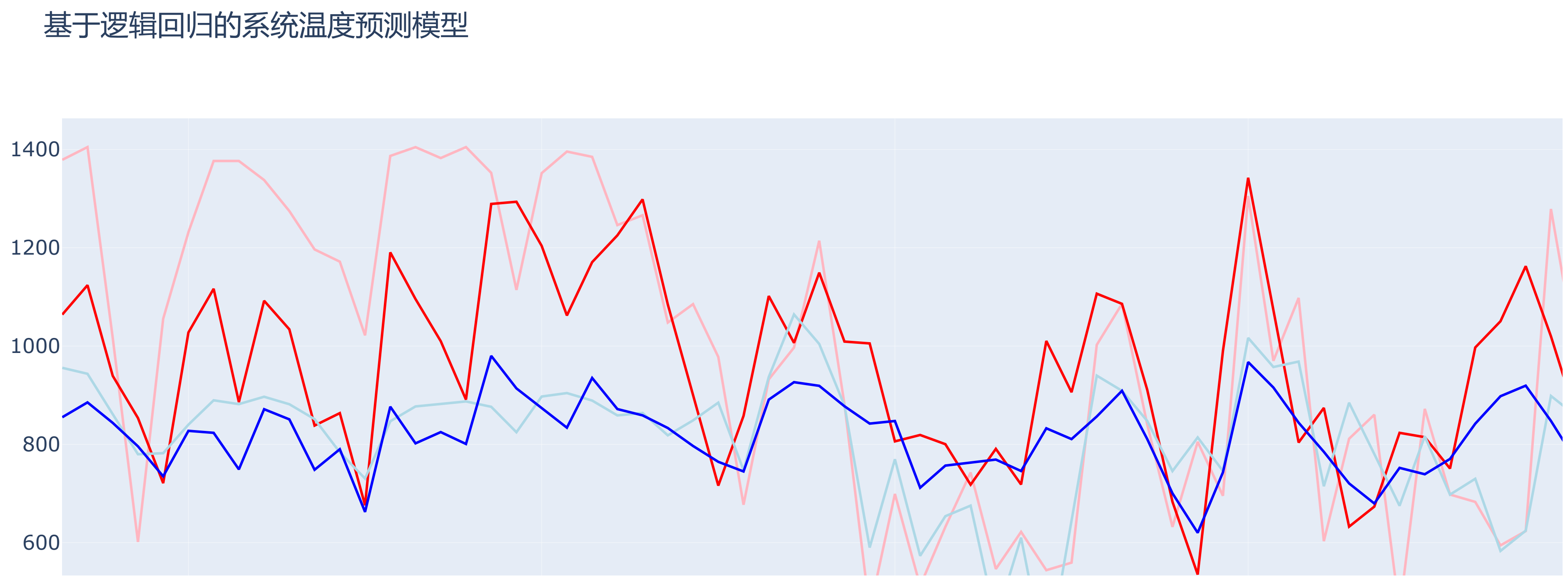
基于逻辑回归的系统I温度 (Temperature of system I)预测模型



```
accuracy: 0.8085497767417638
MSE: 28984.40505212021 MAE: 125.74285529371278 R2: -0.2329850436166907
预测结果:  [867.18235787 909.5239446 ]
```

基于逻辑回归的系统II温度 (Temperature of system II)预测模型



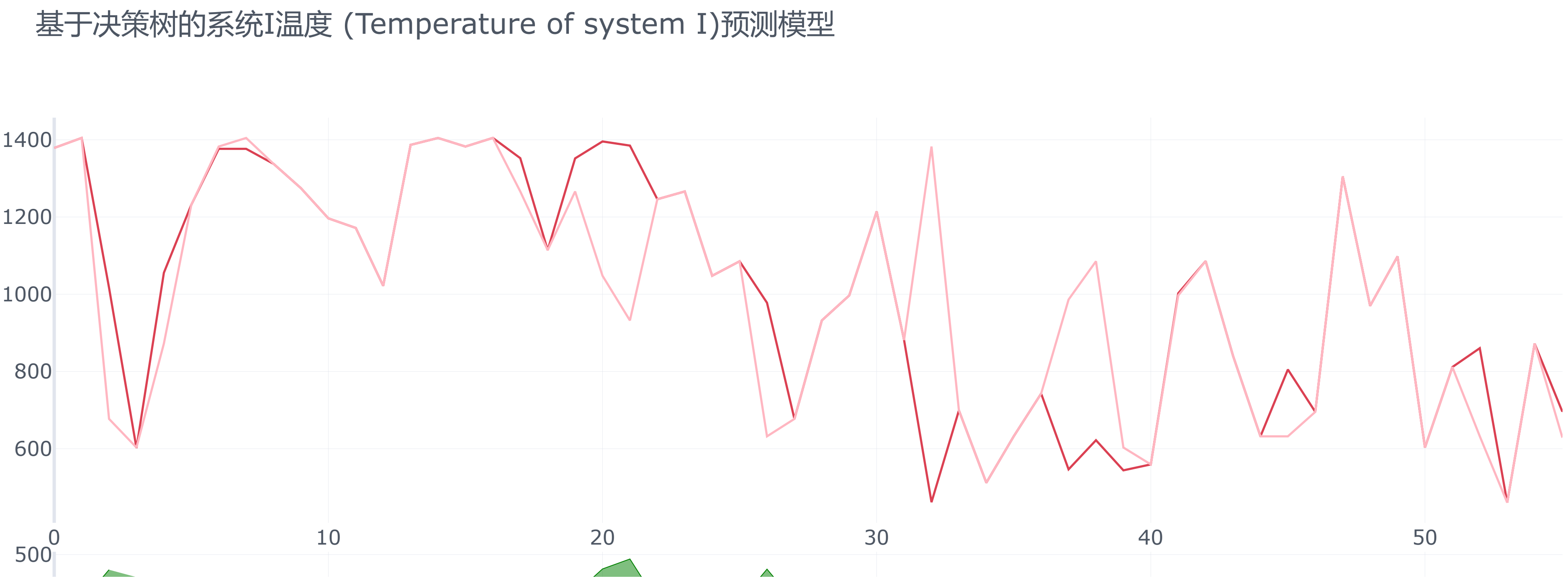

基于逻辑回归的系统温度预测模型

2. 决策树

In [17]:

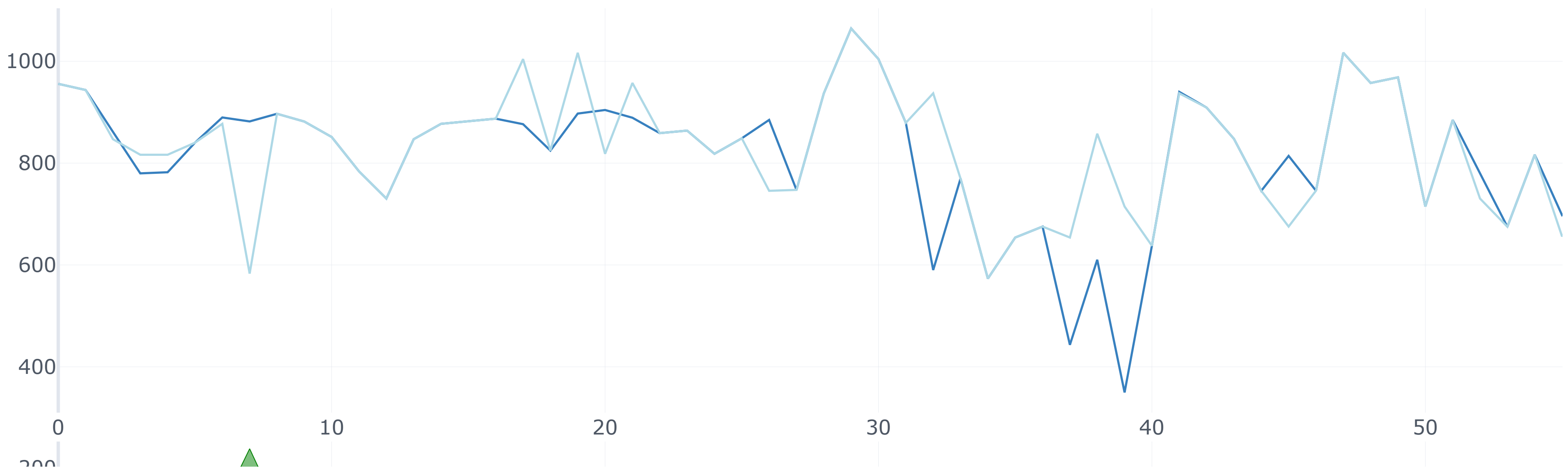
```
from sklearn.tree import DecisionTreeRegressor, ExtraTreeRegressor

model_dt = [DecisionTreeRegressor(), ExtraTreeRegressor()]  # mse, friedman_mse, mae

for model in model_dt[:1]:
    run_model("决策树", model, )
```

```
决策树 ：

accuracy: 0.7057510208815958
MSE: 110552.9356258625 MAE: 230.63225092364576 R2: -0.15347549606795363
预测结果：  [1047.83615278 1266.14945139]
```



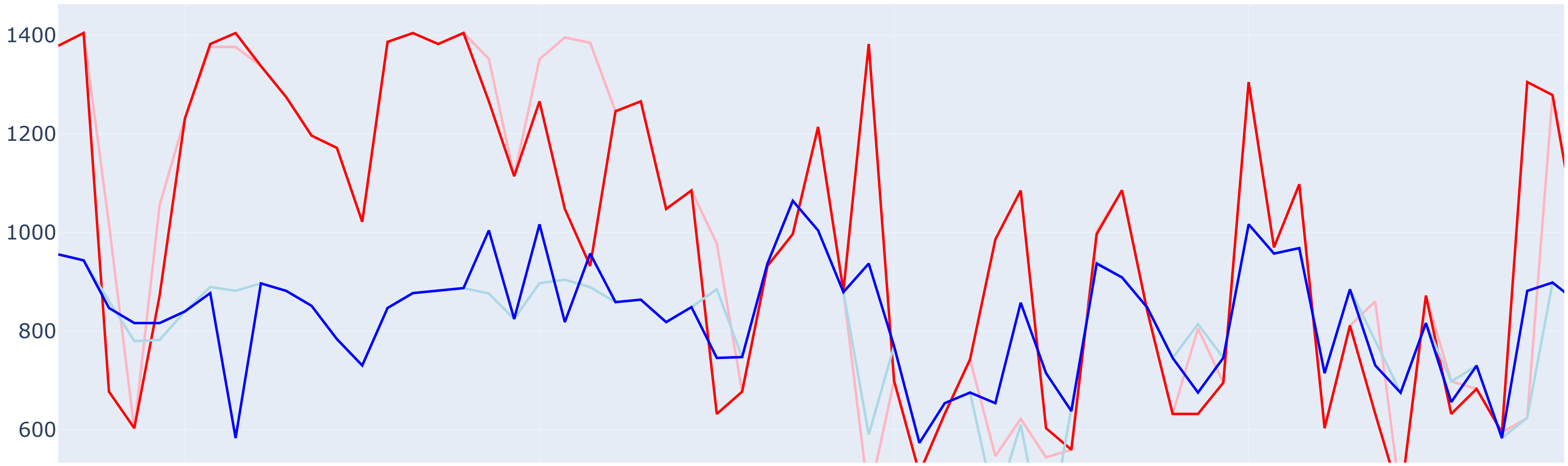
基于决策树的系统I温度 (Temperature of system I)预测模型

```
accuracy: 0.8132963315166466
MSE: 29734.821408529628 MAE: 129.64571964692576 R2: -0.2649074564547098
预测结果：  [818.26811111 968.42205289]
```



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

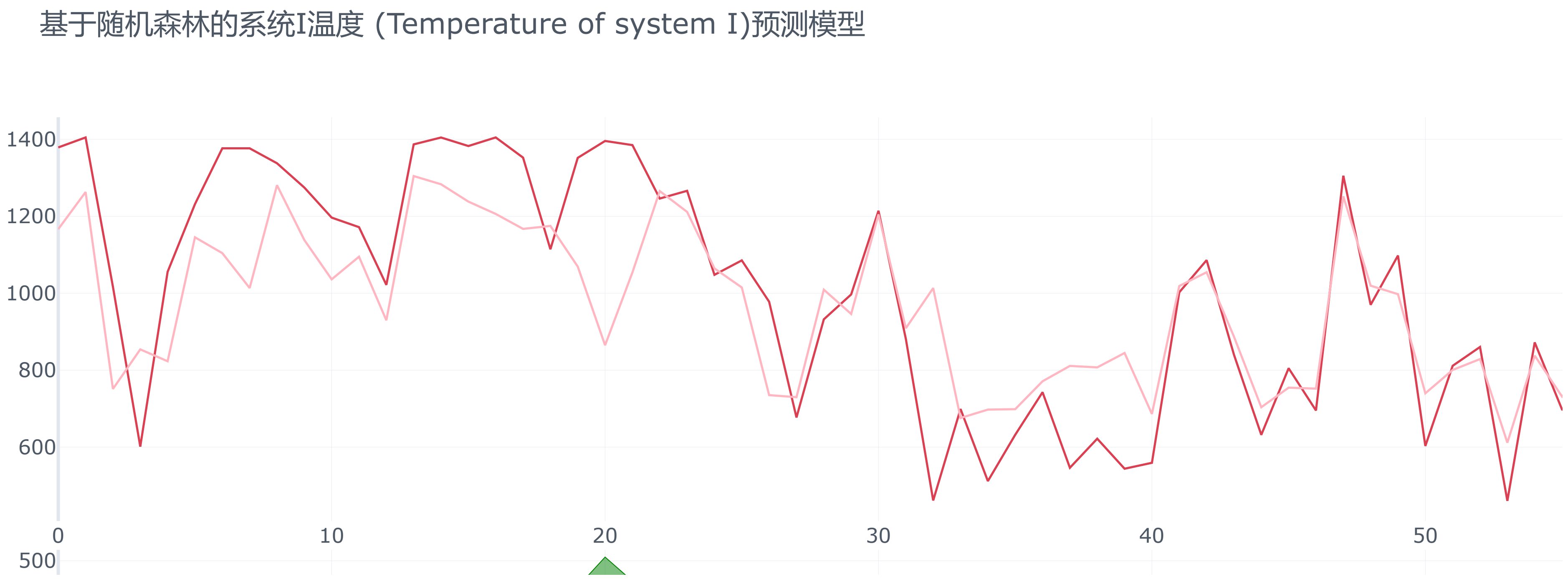


3. 使用随机森林

In [18]:

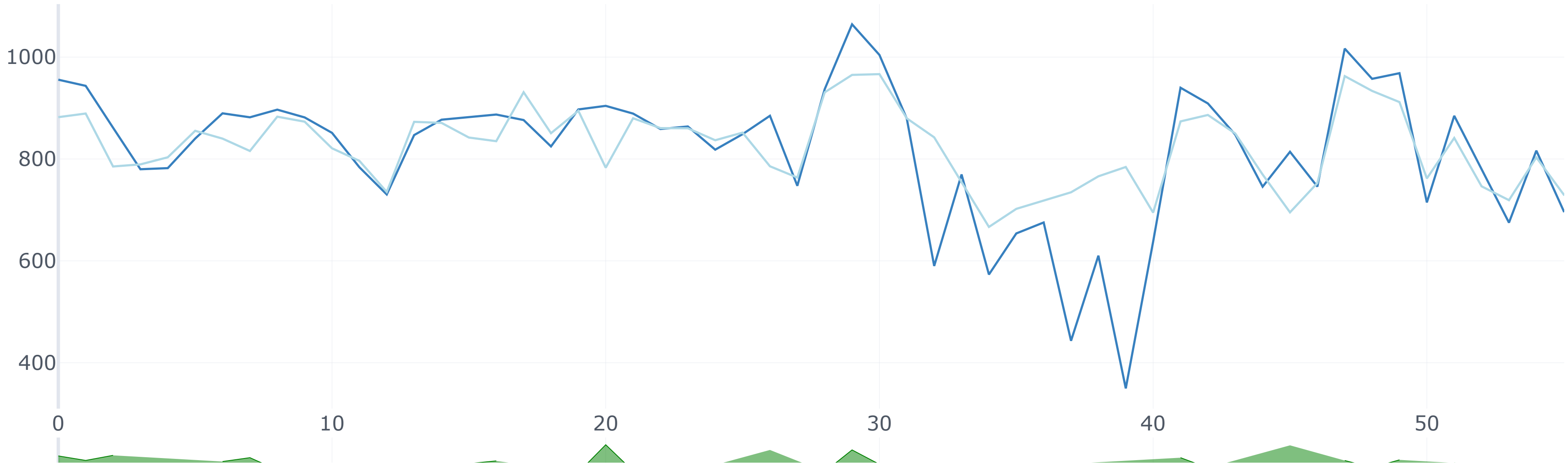
```python
from sklearn.ensemble import RandomForestRegressor as RFR

model_rf = [RFR(criterion='mae', n_estimators=100, random_state=0)]  # mse, friedman_mse, mae

for model in model_rf:
    run_model("随机森林", model, )
```

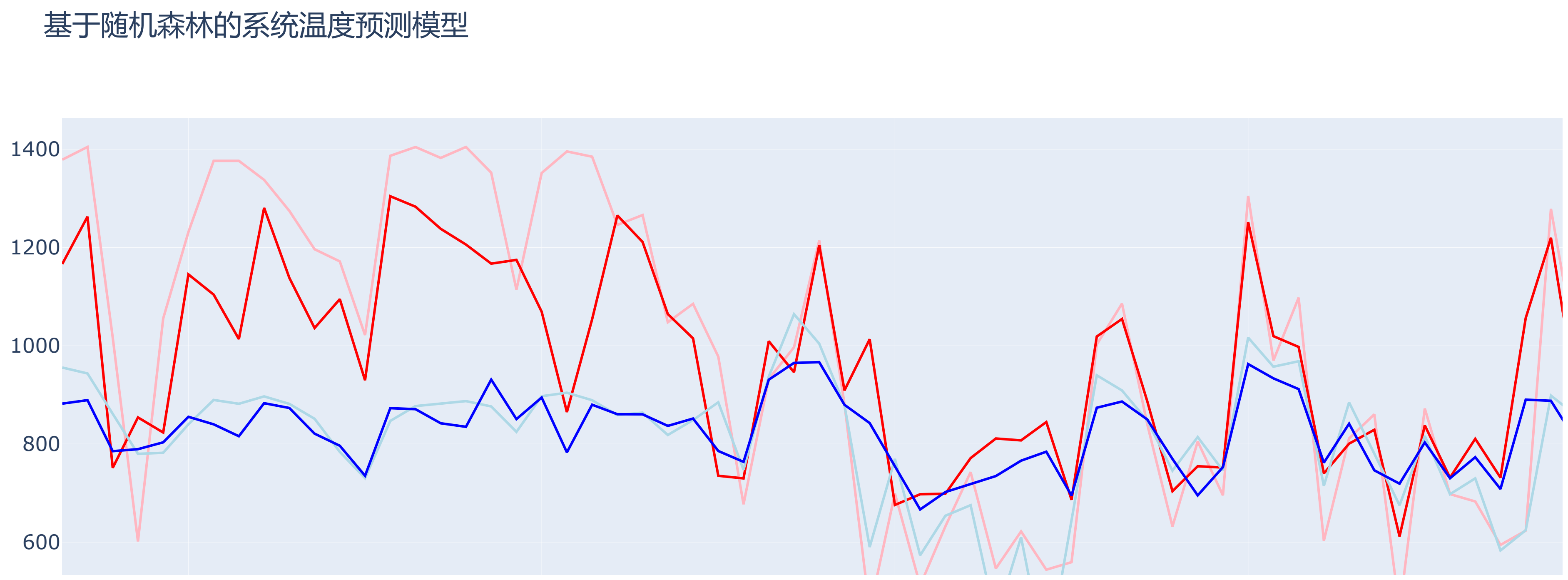
```
随机森林 :

accuracy: 0.7481190054840587
MSE: 75615.77232966159 MAE: 228.03770350796893 R2: 0.21104817339546467
预测结果:  [ 899.96611601 1065.29412275]
```



```
accuracy: 0.8326941802445973
MSE: 23268.624951783542 MAE: 105.13597069503925 R2: 0.010161964701891124
预测结果:  [784.75317061 871.85108096]
```

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



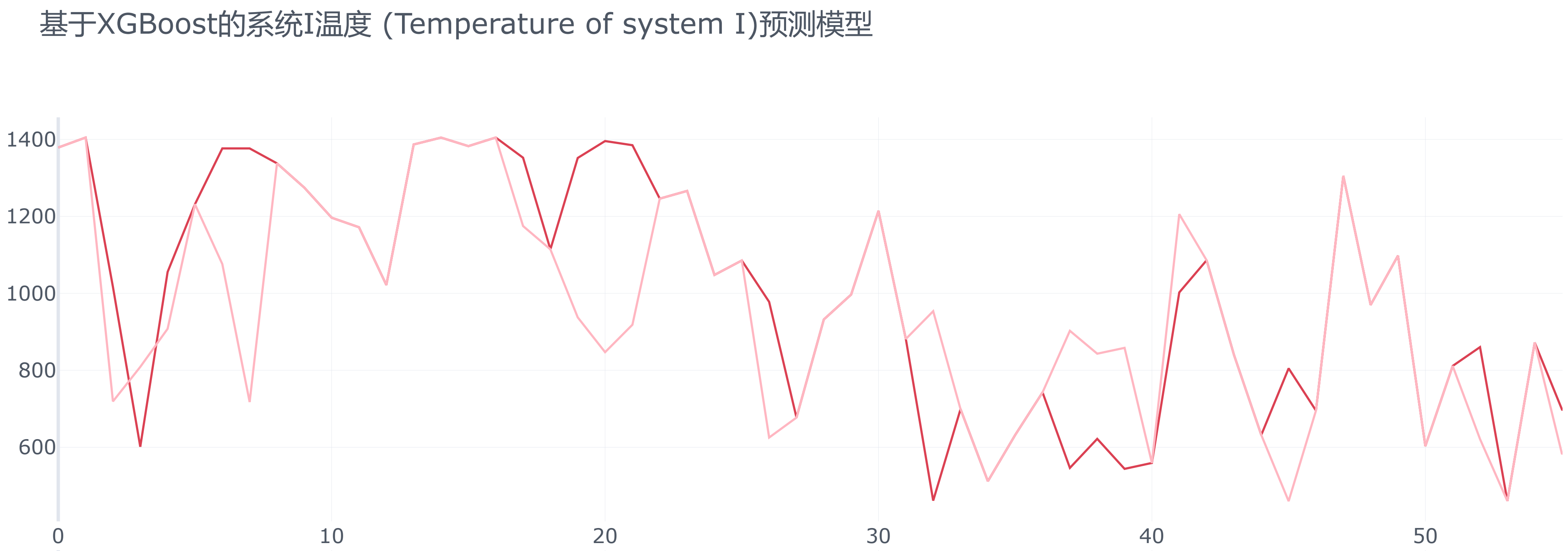


4. XGBoost

In [19]:

```
from xgboost import XGBRegressor, XGBRFRegressor

models_xgb = [
    XGBRegressor(n_estimators=100, random_state=0),
    XGBRFRegressor(n_estimators=100, random_state=0),
]
for model in models_xgb:
    run_model("XGBoost", model, )
#     print(dim.inverse(run_model("XGBoost", model, ), method='standard'))
```

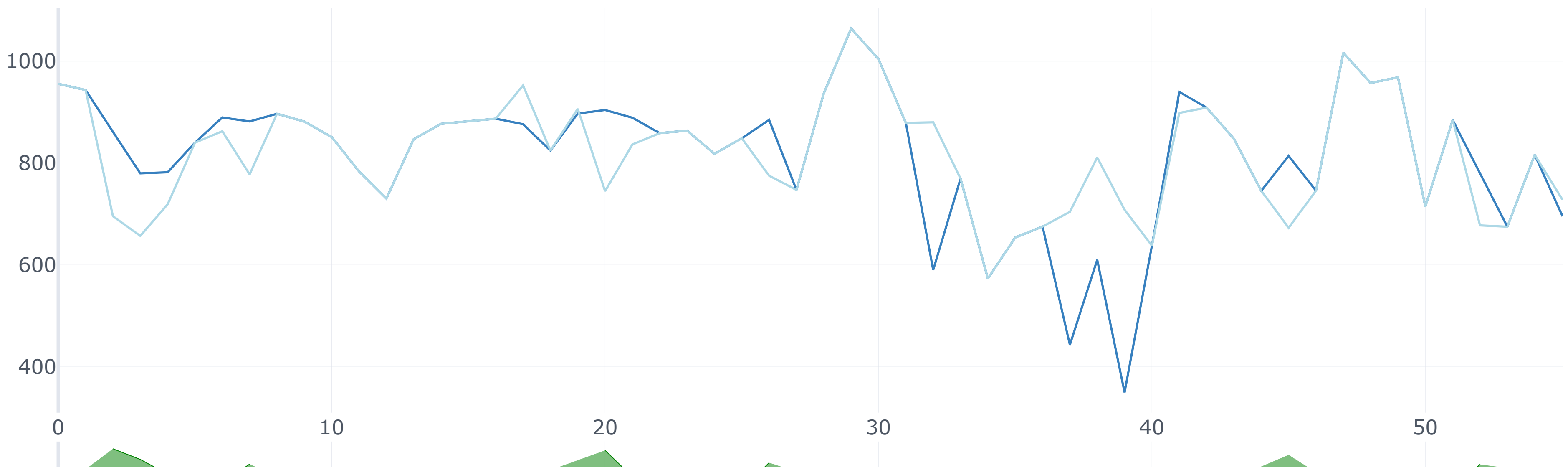
```
XGBoost :

accuracy: 0.6651749975400643
MSE: 125433.73793470411 MAE: 313.9292203794766 R2: -0.3087372331526155
预测结果:  [1006.54425  995.7379 ]
```

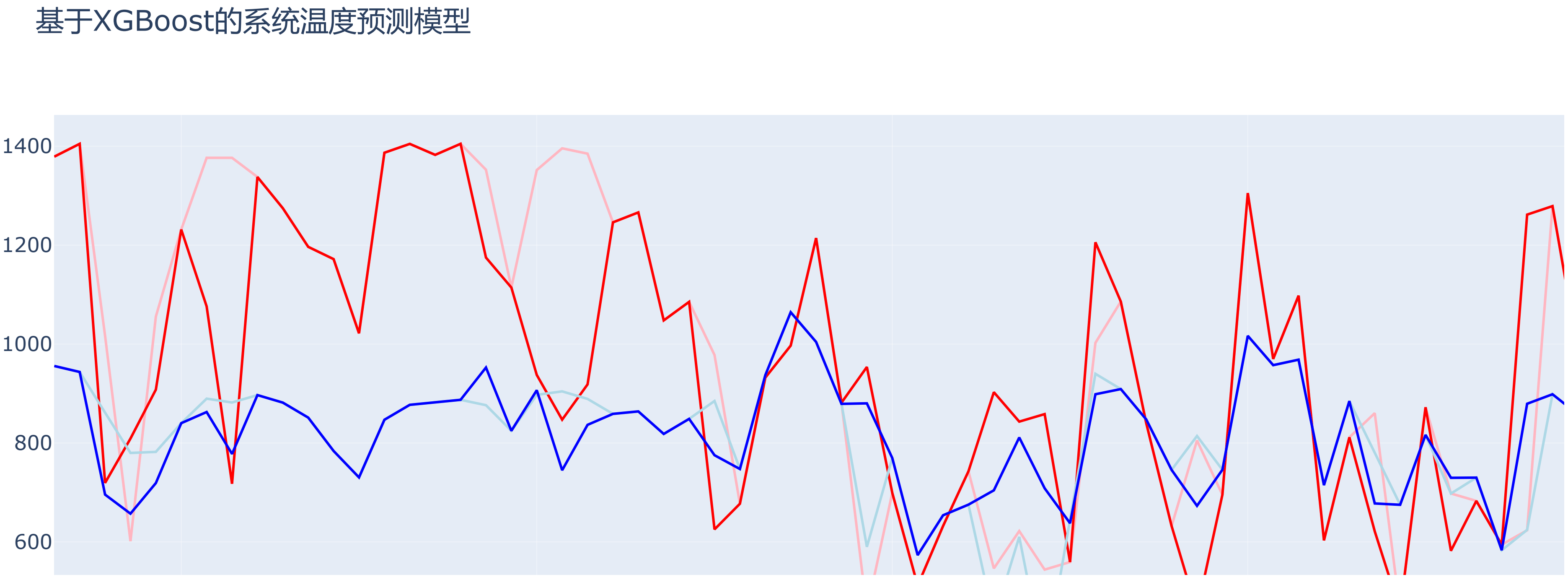


基于XGBoost的系统I温度 (Temperature of system I)预测模型

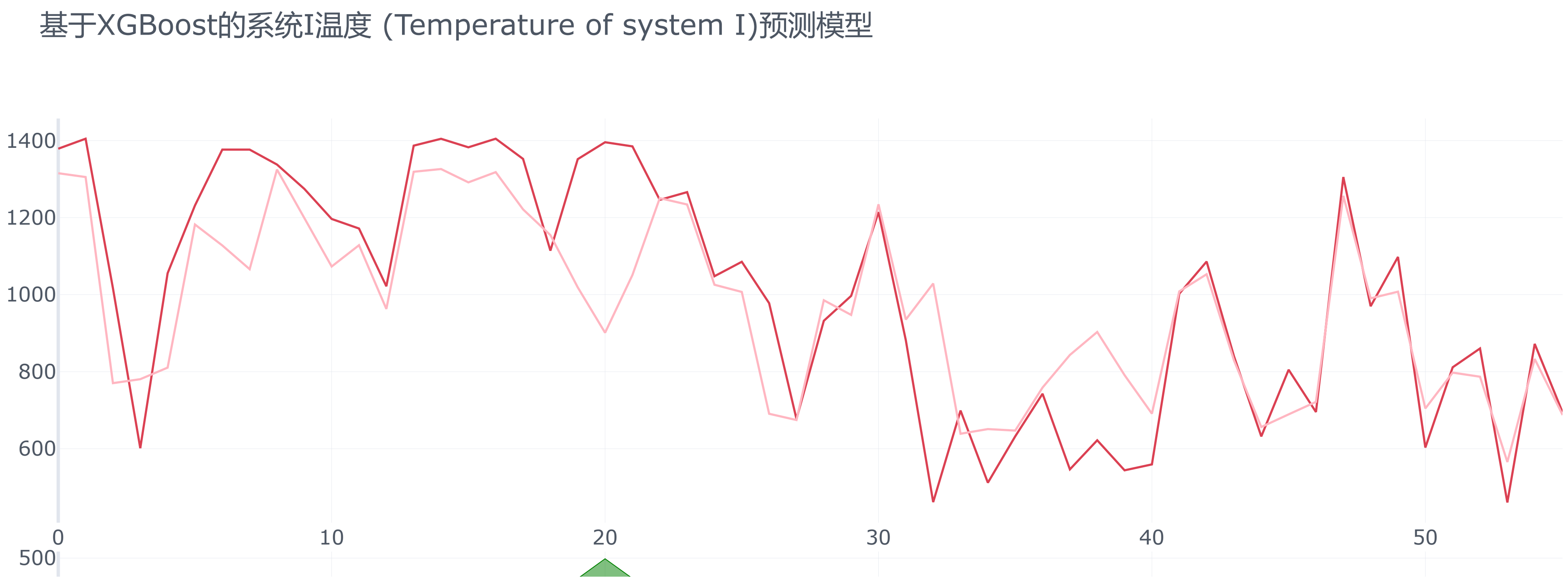
```
accuracy: 0.8190435796189237
MSE: 24599.212189948597 MAE: 123.72797372451205 R2: -0.04644068630766429
预测结果:  [773.37823 856.7975 ]
```

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



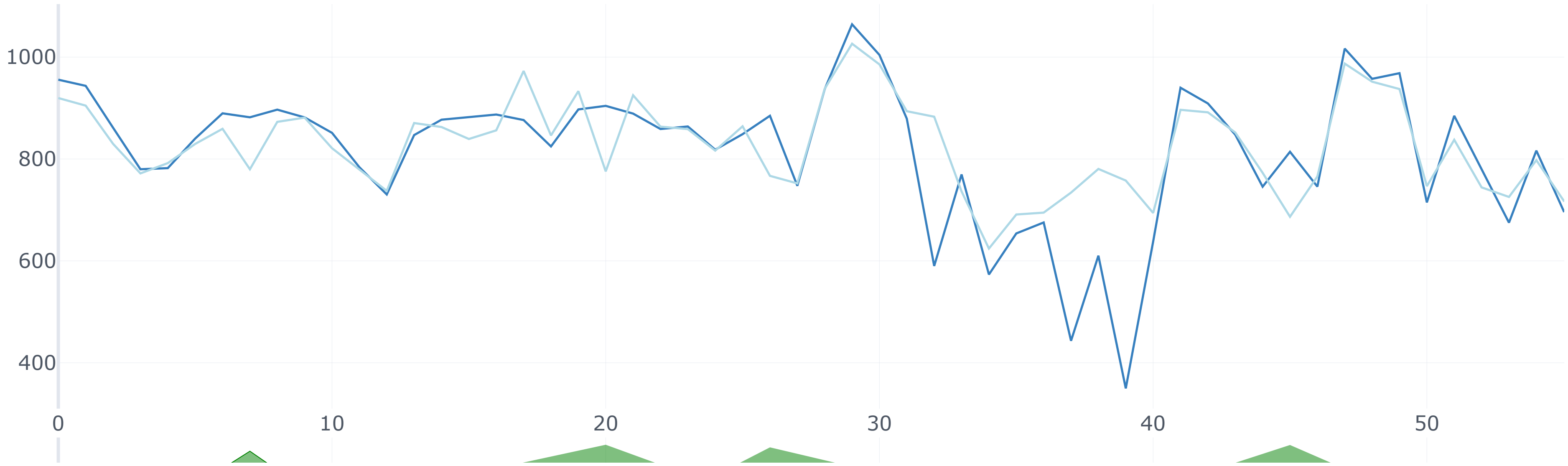


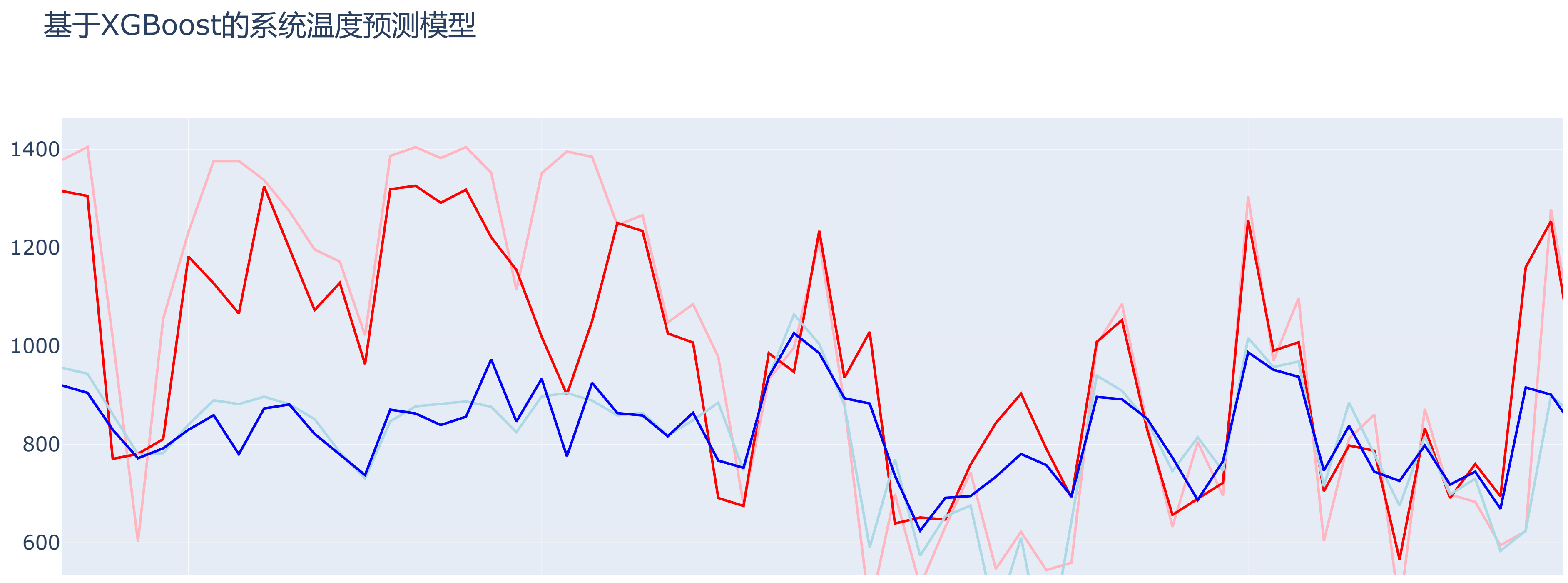
```
XGBoost :

accuracy: 0.7364865997767273
MSE: 80242.0395607846 MAE: 237.27895571624265 R2: 0.16277911695519243
预测结果:  [ 920.7894 1041.9387]
```



```
accuracy: 0.8286147648500213
MSE: 24504.20600093235 MAE: 110.11093508364803 R2: -0.04239916087709528
预测结果:  [766.02435 909.35596]
```

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



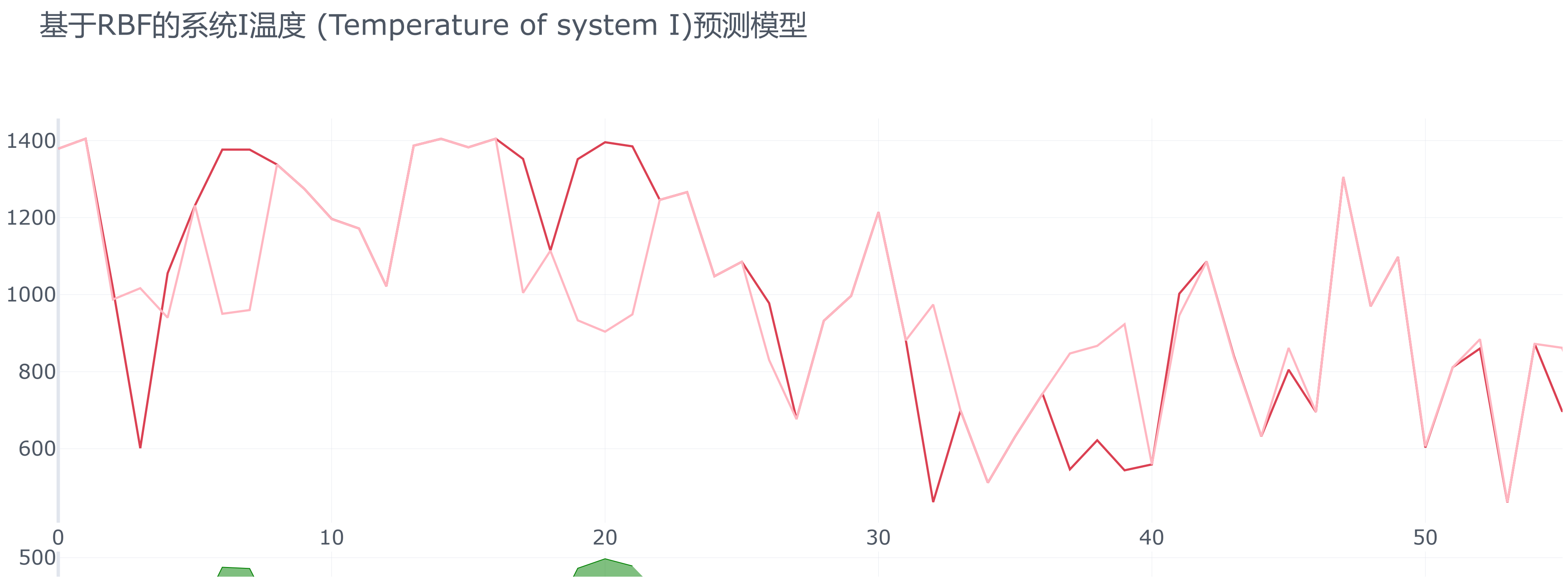


5. RBF

```
In [20]: from hmz.math_model.predict import RBF
test_size_ = 0.3
model = RBF(hidden_num=int((1-test_size_)*len(X)), rbf_type=2)
run_model("RBF", model, test_size=test_size_, )
```

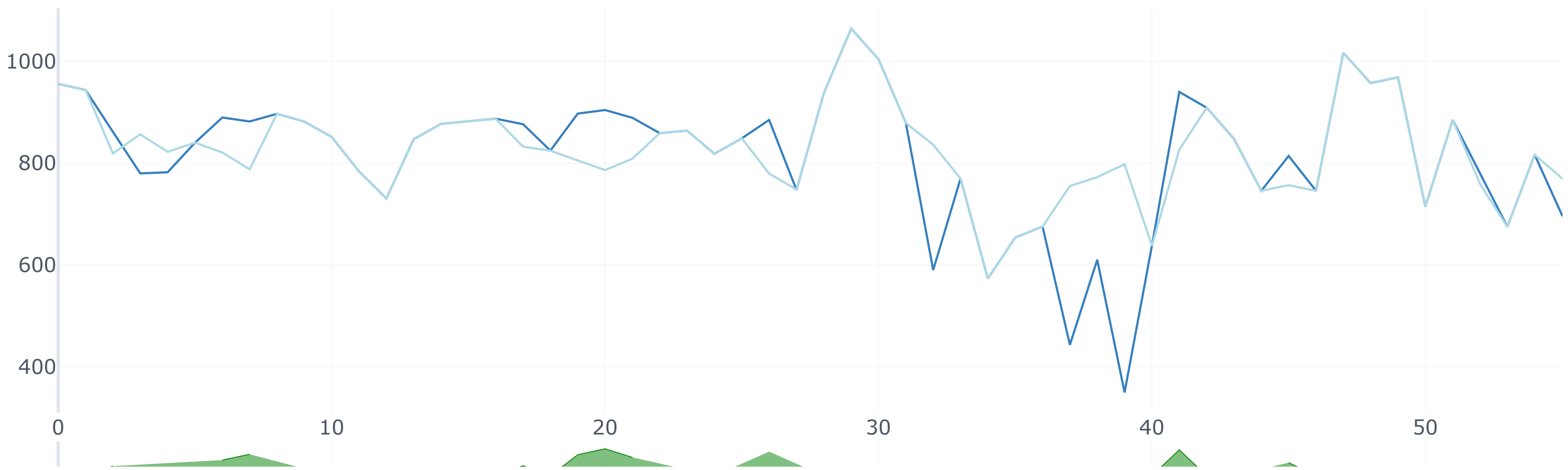
```
RBF :
```

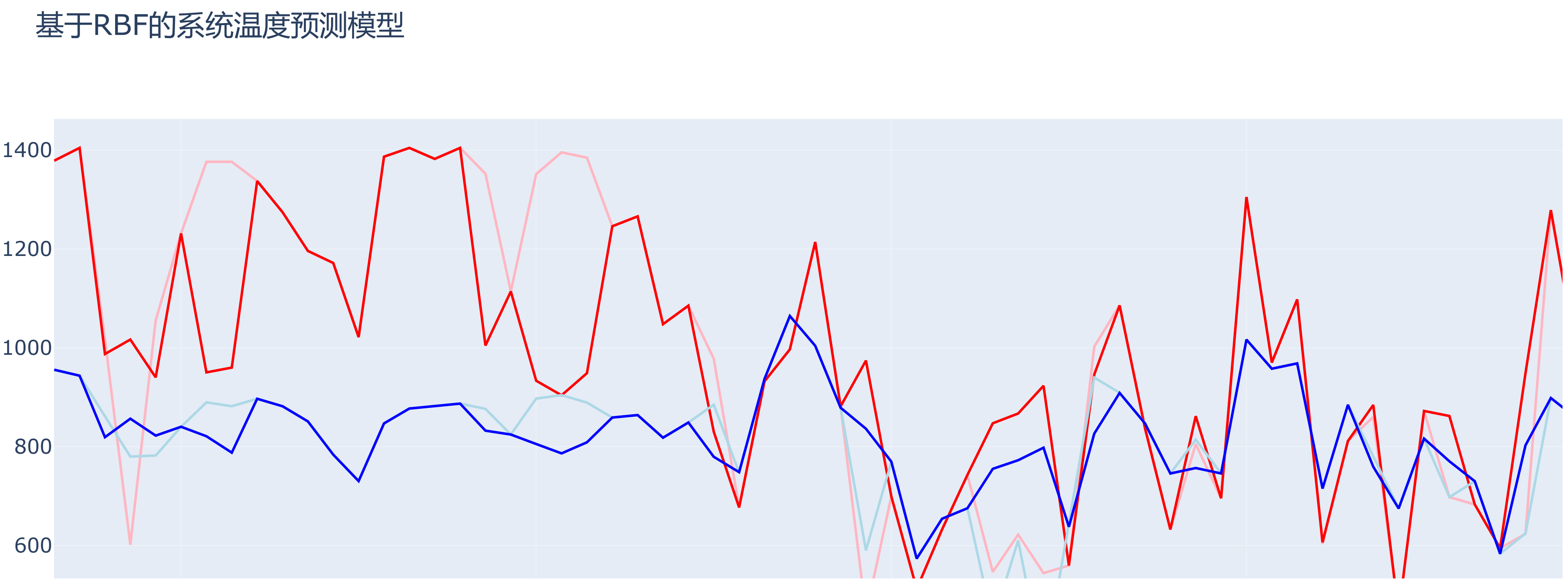
```
epoch: 1000, train loss: 22.617414160740005: 100%|████████████████████| 1000/1000 [00:02<00:00, 375.59it/s]
accuracy: 0.7107989723765779
MSE: 91190.04995764988 MAE: 250.0400608856144预测结果:  [ 953.60340444 1004.87951751]
```


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

```
epoch: 1000, train loss: 2.9663763778266605: 100%|████████████████████████████████| 1000/1000 [00:02<00:00, 374.89it/s]
accuracy: 0.8229874507141477
MSE: 23393.87545373886 MAE: 115.49403994877721预测结果:  [798.57456106 821.54029028]
```

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





```
Out[20]: array([798.57456106, 821.54029028])
```

```
In [ ]:
```