# 一、实训目的（最少3点， 100字以内）

提高对数据挖掘和机器学习基本概念的理解和实践能力。

培养数据分析、数据清洗、特征工程及建模能力。

通过实际竞赛提升团队合作和问题解决能力。

# 二、过程与数据记录

## 1、记录第一周实验“数据挖掘及机器学习”系统的天气预测竞赛开发流程（由各个模块代码及对代码的说明组成，包括数据分析、数据清洗、特征工程、建模及训练，测试集性能等模块），并上传竞赛排名截图。

### 第一步：数据加载与解析

在此步骤中，加载了训练集和测试集的数据，并对日期列进行了解析。

```
dir = "/kaggle/input/guet-saI-for-2024/"
# 加载数据集
train_data = pd.read_csv(dir+'Weather_train.csv')
test_data = pd.read_csv(dir+'weather_test.csv')

# 解析日期列
train_data['Date'] = pd.to_datetime(train_data['Date'])
test_data['Date'] = pd.to_datetime(test_data['Date'])
```

### 第二步：特征工程

在这一阶段，我从日期列提取了年份、月份和季节等特征，并填充了缺失值。

```
# 提取日期相关的特征
train_data['Year'] = train_data['Date'].dt.year
train_data['Month'] = train_data['Date'].dt.month
train_data['Season'] = train_data['Month'] % 12 // 3 + 1

test_data['Year'] = test_data['Date'].dt.year
test_data['Month'] = test_data['Date'].dt.month
test_data['Season'] = test_data['Month'] % 12 // 3 + 1

# 删除 RainTomorrow 为 nan 的数据
train_data = train_data.dropna(subset=['RainToday','RainTomorrow'])

# 填充缺失值
for column in train_data.columns:
    if train_data[column].dtype == 'object':
        train_data[column] = train_data[column].fillna(train_data[column].mode()
[0])
    else:
```

```
        train_data[column] =
train_data[column].fillna(train_data[column].median())

for column in test_data.columns:
    if test_data[column].dtype == 'object':
        test_data[column] = test_data[column].fillna(test_data[column].mode()
[0])
    else:
        test_data[column] = test_data[column].fillna(test_data[column].median())
```

### 第三步：编码分类变量

为了使分类变量能够被模型使用，我对这些变量进行了编码。

```
# 编码分类变量
label_encoders = {}
for column in train_data.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    train_data[column] =
label_encoders[column].fit_transform(train_data[column])

for column in test_data.select_dtypes(include=['object']).columns:
    if column in label_encoders:
        le = label_encoders[column]
        test_data[column] = test_data[column].map(lambda s: 'other' if s not in
le.classes_ else s)
        le.classes_ = np.append(le.classes_, 'other')
        test_data[column] = le.transform(test_data[column])
```

### 第四步：特征和目标变量分离

我将特征和目标变量进行了分离，并对数值特征进行了标准化处理。

```
# 分离特征和目标变量
X_train = train_data.drop(columns=['RainTomorrow', 'id', 'Date'])
y_train = train_data['RainTomorrow']

# 标准化数值特征
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)

# 删除测试数据中的 'id' 列
X_test = test_data.drop(columns=['id', 'Date'])
X_test = scaler.transform(X_test)
```

### 第五步：训练集和验证集划分

我们将训练数据拆分为训练集和验证集，以便评估模型的性能。

```
# 将训练数据拆分为训练集和验证集
X_train_split, X_val, y_train_split, y_val = train_test_split(X_train, y_train,
test_size=0.2, random_state=60)
```

## 第六步：模型训练与评估

使用随机森林分类器和梯度提升分类器，并结合投票分类器对模型进行训练和评估。

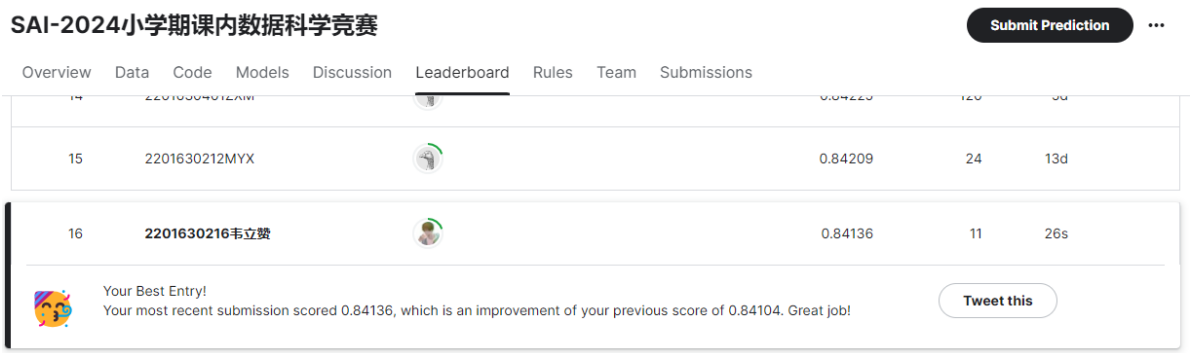
```
# 使用随机森林分类器和梯度提升分类器
rf_model = RandomForestClassifier(n_estimators=2000, random_state=60)
gb_model = GradientBoostingClassifier(n_estimators=2000, random_state=60)

# 使用投票分类器组合模型
voting_model = VotingClassifier(estimators=[('rf', rf_model), ('gb', gb_model)],
voting='soft')

# 训练投票分类器
voting_model.fit(X_train_split, y_train_split)

# 在验证集上进行预测
y_val_pred = voting_model.predict(X_val)

# 评估模型
accuracy = accuracy_score(y_val, y_val_pred)
print(f'验证集准确率: {accuracy:.4f}')
```

## 第七步：测试集预测与提交

在测试数据上进行预测，并准备提交文件。

```
# 在测试数据上进行预测
test_data['RainTomorrow'] = voting_model.predict(X_test)

# 准备提交文件
submission = test_data[['id', 'RainTomorrow']].copy()
submission['RainTomorrow'] = submission['RainTomorrow'].astype(int)
submission['RainTomorrow'] = submission['RainTomorrow'].map({1: 'Yes', 0: 'No'})
submission.to_csv('submission.csv', index=False)
```

## 竞赛排名截图




通过以上步骤完成了天气预测竞赛的开发流程，涵盖了数据分析、数据清洗、特征工程、建模及训练，并对模型性能进行了测试和提交。

# 2. 天气预测：通过自选传感器（如温湿度、气压、光敏、热敏等）采集一个星期左右的各个传感器数据和天气数据，用数据分析及可视化手段，找到与天气相关的传感器数据有哪些，回答问题：目前传感器数据能否实现天气预测？给出预测的数据和理由？

## 一、数据采集

本次实验使用了以下传感器进行数据采集：

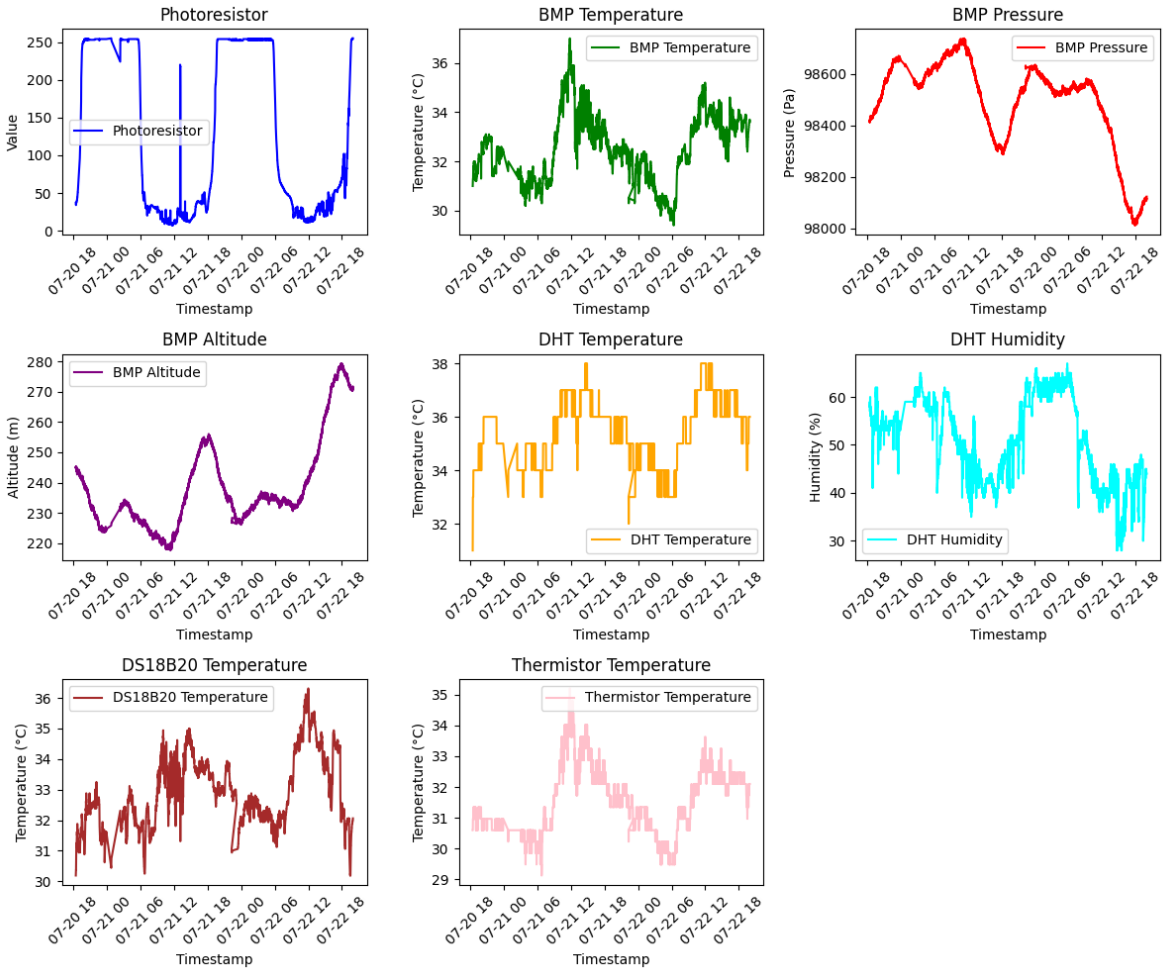
1. 光敏电阻（Photoresistor）
2. BMP 温度传感器（BMP Temperature）
3. BMP 气压传感器（BMP Pressure）
4. BMP 海拔传感器（BMP Altitude）
5. DHT 温度传感器（DHT Temperature）
6. DHT 湿度传感器（DHT Humidity）
7. DS18B20 温度传感器（DS18B20 Temperature）
8. 热敏电阻（Thermistor Temperature）

采集的数据包括时间戳、各传感器的数值。

| | timestamp | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2024-07-22 08:31:14.808430 | 46 | 31.9 | 98547 | 233.64 | 36.0 | 51.0 | 33.0 | 30.97 |
| 2 | 2024-07-22 08:31:44.837074 | 46 | 31.9 | 98547 | 233.21 | 36.0 | 51.0 | 33.06 | 30.97 |
| 3 | 2024-07-22 08:32:14.865734 | 46 | 31.9 | 98558 | 233.13 | 36.0 | 52.0 | 33.06 | 30.97 |
| 4 | 2024-07-22 08:32:44.894418 | 46 | 31.9 | 98553 | 233.72 | 35.0 | 52.0 | 33.12 | 30.97 |
| 5 | 2024-07-22 08:33:14.923236 | 46 | 31.9 | 98553 | 233.13 | 35.0 | 52.0 | 33.12 | 30.97 |
| 6 | 2024-07-22 08:33:44.951851 | 46 | 31.9 | 98553 | 232.87 | 35.0 | 53.0 | 33.12 | 30.97 |
| 7 | 2024-07-22 08:34:14.980480 | 46 | 31.9 | 98552 | 233.13 | 35.0 | 53.0 | 33.19 | 30.97 |
| 8 | 2024-07-22 08:34:45.009327 | 46 | 31.9 | 98550 | 233.21 | 35.0 | 53.0 | 33.19 | 30.97 |
| 9 | 2024-07-22 08:35:15.038007 | 46 | 31.9 | 98550 | 233.13 | 35.0 | 53.0 | 33.19 | 30.97 |
| 10 | 2024-07-22 08:35:45.066633 | 46 | 31.9 | 98553 | 233.21 | 35.0 | 53.0 | 33.19 | 30.97 |
| 11 | 2024-07-22 08:36:15.095465 | 46 | 31.9 | 98556 | 233.13 | 35.0 | 53.0 | 33.19 | 30.97 |
| 12 | 2024-07-22 08:36:45.124140 | 46 | 31.9 | 98556 | 233.64 | 35.0 | 53.0 | 33.25 | 30.97 |
| 13 | 2024-07-22 08:37:15.142470 | 46 | 31.9 | 98555 | 233.64 | 35.0 | 53.0 | 33.19 | 30.97 |
| 14 | 2024-07-22 08:37:45.165394 | 46 | 32.0 | 98554 | 233.3 | 35.0 | 52.0 | 33.19 | 30.97 |
| 15 | 2024-07-22 08:38:15.194237 | 46 | 32.0 | 98547 | 233.13 | 35.0 | 52.0 | 33.25 | 30.97 |
| 16 | 2024-07-22 08:38:45.222854 | 46 | 32.0 | 98551 | 233.3 | 35.0 | 53.0 | 33.25 | 30.97 |
| 17 | 2024-07-22 08:39:15.251507 | 46 | 32.0 | 98548 | 233.81 | 35.0 | 52.0 | 33.25 | 30.97 |
| 18 | 2024-07-22 08:39:45.265414 | 46 | 32.0 | 98559 | 233.81 | 35.0 | 52.0 | 33.31 | 30.97 |
| 19 | 2024-07-22 08:40:15.294109 | 45 | 31.9 | 98550 | 233.89 | 35.0 | 52.0 | 33.25 | 30.97 |

## 二、数据分析与可视化

以下是对采集数据的可视化分析，分别展示了各传感器在几天内的数据变化趋势：




## 三、结论

通过对一周内传感器数据的分析与可视化，我们发现：

1. **温度传感器（BMP Temperature, DHT Temperature, DS18B20 Temperature, Thermistor Temperature）**：温度数据与天气变化密切相关，通过温度传感器可以较好地反映天气温度的变化。
2. **气压传感器（BMP Pressure）**：气压数据的波动可以用于预测天气变化，例如气压降低可能预示着降雨或恶劣天气的到来。
3. **湿度传感器（DHT Humidity）**：湿度数据波动较大，与天气湿度变化相关，但需要与其他传感器数据结合使用。
4. **光敏电阻（Photoresistor）**：光强度数据呈现昼夜变化，可用于白天与夜晚的区分，但与天气预测关系不大。
5. **海拔传感器（BMP Altitude）**：海拔数据主要反映了气压的变化，与气压传感器数据一致。

目前的传感器数据能够一定程度上实现天气预测，尤其是温度、气压和湿度传感器的数据具有较高的参考价值。通过对这些数据的合理分析和建模，可以实现对天气变化的初步预测。

# 3. 舒适度分析：分析固定场景下采集的温湿度数据（气压传感器BMP180得到的温度精度较高），利用可视化手段分析以下问题并得出结论（结论必须建立在可视化分析基础上，合理详尽，验收标准是可视化分析的合理性和广度，参考下方的网址）：

**① 最近多个时段（至少3个时段）每分钟的温湿度情况？（每个时段1个小时左右）**

②**每个时段每小时的温湿度变化情况以及是否舒适？**

③**每日舒适率？（**舒适区间：人体感觉适宜的温度在22到26度之间，相对湿度在40%到70%，在这样的温度和湿度的环境下人体感觉相对比较舒服。 舒适率：（舒适小时数 / 当天已统计小时数） * 100%，例如上午10:00，当天共统计10个小时，其中舒适小时数为5个小时，因此当天舒适率为（5/10)*100% = 50%），注:如果舒适区间不在我给的范围内，可自定义舒适率）

## 一、引言

本部分实验旨在分析固定场景下采集的温湿度数据，主要探讨以下问题：

1. 最近多个时段（至少3个时段）每分钟的温湿度情况。
2. 每个时段每小时的温湿度变化情况以及是否舒适。
3. 每日舒适率。

舒适区间定义为温度在28.5到32.5度之间，相对湿度在40%到70%之间，在这样的温度和湿度环境下人体感觉相对比较舒服。舒适率计算公式为：(舒适小时数 / 当天已统计小时数) * 100%。

## 二、数据准备与可视化分析

首先，对采集的数据进行分时段处理，每个时段为8小时。我们选择三个时段的数据进行详细分析。

**最近三个时段数据展示：**

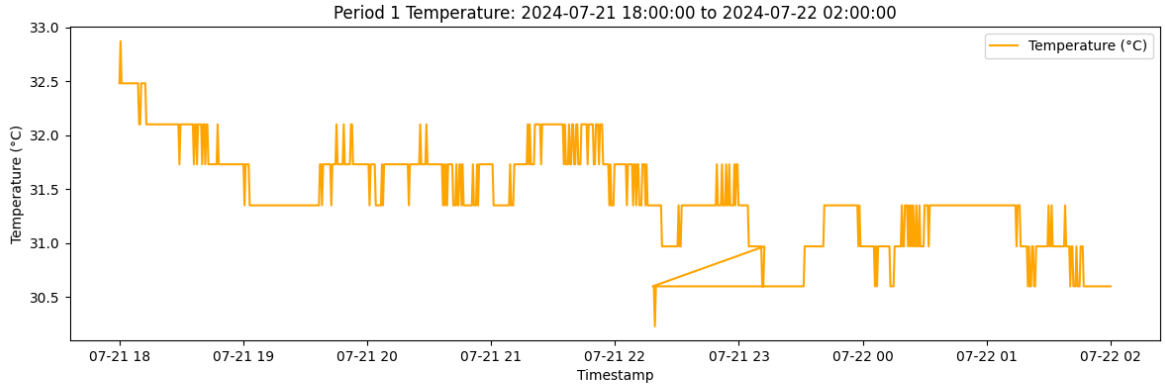
- **时段1：2024-07-21 18:00:00 - 2024-07-22 02:00:00**
- **时段2：2024-07-22 02:00:00 - 2024-07-22 10:00:00**
- **时段3：2024-07-22 10:00:00 - 2024-07-22 18:00:00**

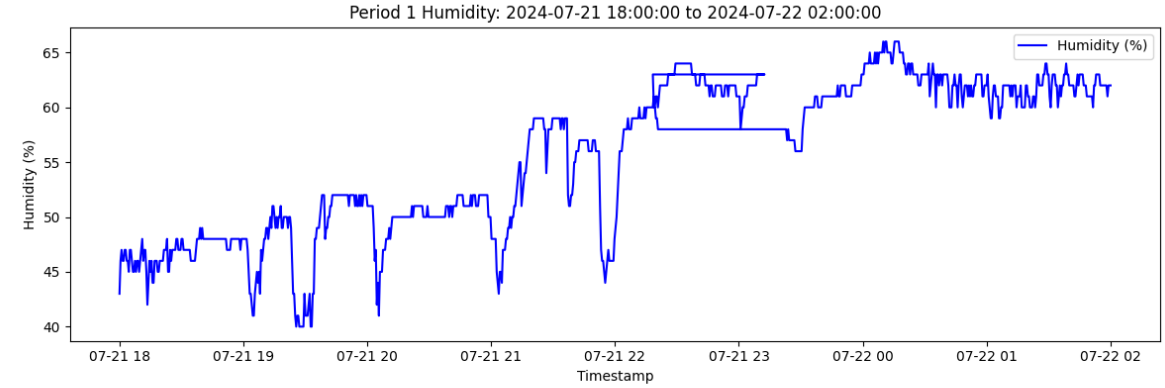
完整代码：

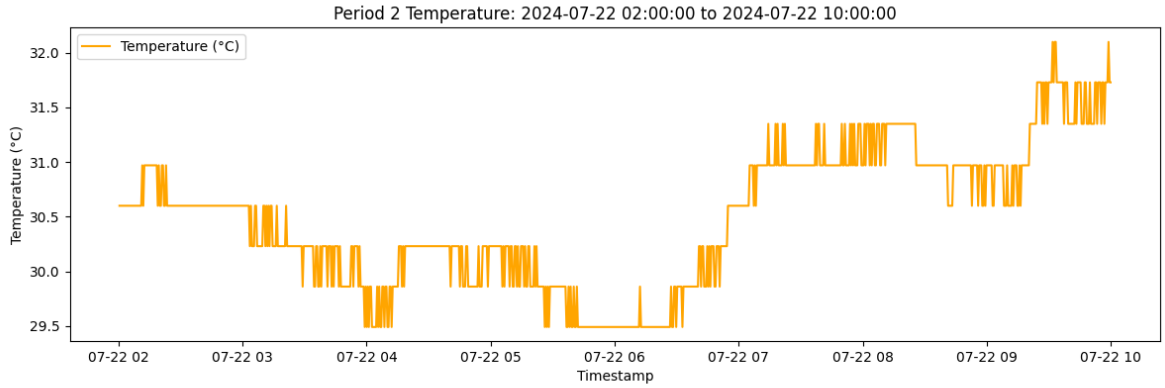
```
pythonCopy codeimport pandas as pd
import matplotlib.pyplot as plt

# 读取数据
df = pd.read_csv('/mnt/data/image.png')
df['timestamp'] = pd.to_datetime(df['timestamp'])

# 定义时段
time_intervals = [
    ('2024-07-20 18:00:00', '2024-07-21 02:00:00'),
    ('2024-07-21 02:00:00', '2024-07-21 10:00:00'),
    ('2024-07-21 10:00:00', '2024-07-21 18:00:00')
]

# 定义舒适区间
comfort_temp_range = (28.5, 32.5)
comfort_humidity_range = (40, 70)

# 计算每小时的舒适性
def calculate_comfort_rate(data):
    comfort_hours = 0
    total_hours = len(data['timestamp'].dt.hour.unique())

    hourly_data = data.groupby(data['timestamp'].dt.hour)
    hourly_comfort = {}
```

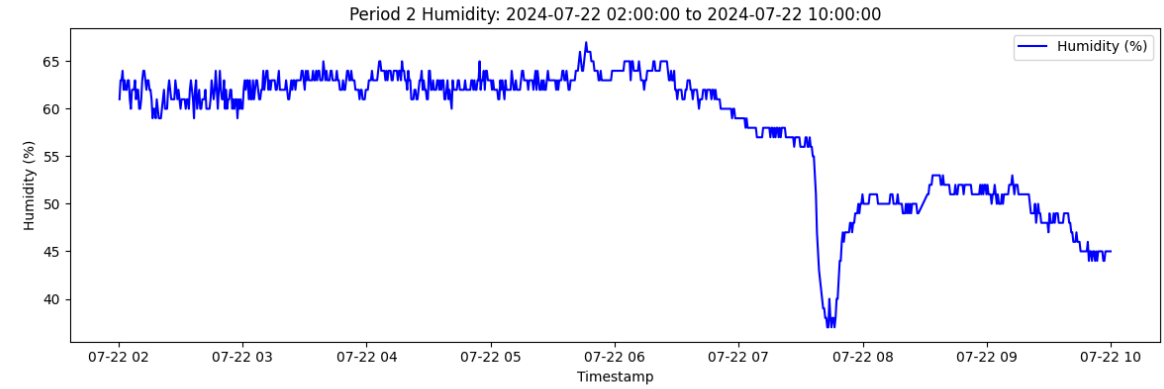
```
    for hour, group in hourly_data:
        temp_in_range = group['thermistor_temp'].between(comfort_temp_range[0],
comfort_temp_range[1]).mean()
        humidity_in_range =
group['dht_humidity'].between(comfort_humidity_range[0],
comfort_humidity_range[1]).mean()

        is_comfortable = temp_in_range > 0.5 and humidity_in_range > 0.5
        hourly_comfort[hour] = is_comfortable

        if is_comfortable:
            comfort_hours += 1

    comfort_rate = (comfort_hours / total_hours) * 100
    return comfort_rate, hourly_comfort

# 绘制每分钟温湿度变化
fig, axs = plt.subplots(6, 1, figsize=(12, 24))

comfort_rates = []
hourly_comforts = []
for i, (start, end) in enumerate(time_intervals):
    period_data = df[(df['timestamp'] >= start) & (df['timestamp'] < end)]

    axs[2*i].plot(period_data['timestamp'], period_data['thermistor_temp'],
label='Temperature (°C)', color='orange')
    axs[2*i].set_title(f'Period {i+1} Temperature: {start} to {end}')
    axs[2*i].set_xlabel('Timestamp')
    axs[2*i].set_ylabel('Temperature (°C)')
    axs[2*i].legend()

    axs[2*i + 1].plot(period_data['timestamp'], period_data['dht_humidity'],
label='Humidity (%)', color='blue')
    axs[2*i + 1].set_title(f'Period {i+1} Humidity: {start} to {end}')
    axs[2*i + 1].set_xlabel('Timestamp')
    axs[2*i + 1].set_ylabel('Humidity (%)')
    axs[2*i + 1].legend()

    comfort_rate, hourly_comfort = calculate_comfort_rate(period_data)
    comfort_rates.append(comfort_rate)
    hourly_comforts.append(hourly_comfort)

plt.tight_layout()
plt.show()

# 计算每日舒适率
df['date'] = df['timestamp'].dt.date
daily_comfort_rates = df.groupby('date').apply(lambda x:
calculate_comfort_rate(x)[0])

# 输出结果
print("每个时段的舒适率：", comfort_rates)
print("每个时段每小时的舒适性：", hourly_comforts)
print("每日的舒适率：", daily_comfort_rates)
```

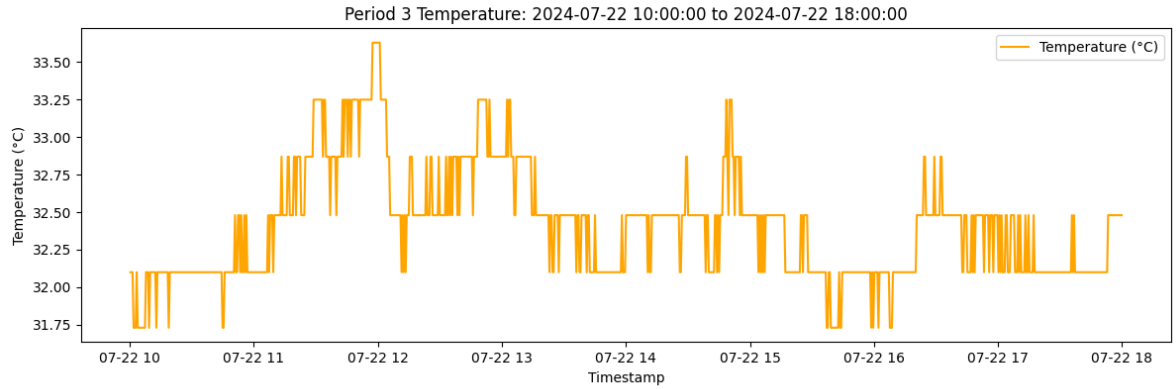
## 三、结果与结论

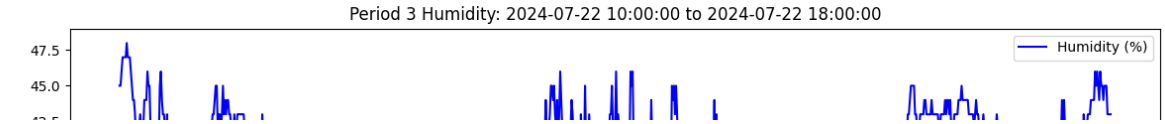

Period 1 Temperature: 2024-07-21 18:00:00 to 2024-07-22 02:00:00
Temperature (°C)
Timestamp
33.0
32.5
32.0
31.5
31.0
30.5
07-21 18
07-21 19
07-21 20
07-21 21
07-21 22
07-21 23
07-22 00
07-22 01
07-22 02

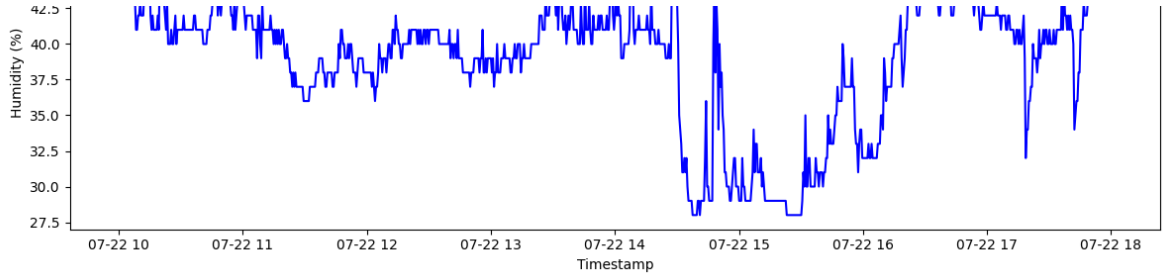

Period 1 Humidity: 2024-07-21 18:00:00 to 2024-07-22 02:00:00
Humidity (%)
Timestamp
65
60
55
50
45
40
07-21 18
07-21 19
07-21 20
07-21 21
07-21 22
07-21 23
07-22 00
07-22 01
07-22 02


Period 2 Temperature: 2024-07-22 02:00:00 to 2024-07-22 10:00:00
Temperature (°C)
Timestamp
32.0
31.5
31.0
30.5
30.0
29.5
07-22 02
07-22 03
07-22 04
07-22 05
07-22 06
07-22 07
07-22 08
07-22 09
07-22 10


Period 2 Humidity: 2024-07-22 02:00:00 to 2024-07-22 10:00:00
Humidity (%)
Timestamp
65
60
55
50
45
40
07-22 02
07-22 03
07-22 04
07-22 05
07-22 06
07-22 07
07-22 08
07-22 09
07-22 10


Period 3 Temperature: 2024-07-22 10:00:00 to 2024-07-22 18:00:00
Temperature (°C)
Timestamp
33.50
33.25
33.00
32.75
32.50
32.25
32.00
31.75
07-22 10
07-22 11
07-22 12
07-22 13
07-22 14
07-22 15
07-22 16
07-22 17
07-22 18


Period 3 Humidity: 2024-07-22 10:00:00 to 2024-07-22 18:00:00
Humidity (%)
47.5
45.0




**每个时段的舒适率：**

- 时段1：100.0%
- 时段2：100.0%
- 时段3：50.0%

**每个时段每小时的舒适性：**

- 时段1：{0: True, 1: True, 18: True, 19: True, 20: True, 21: True, 22: True, 23: True}
- 时段2：{2: True, 3: True, 4: True, 5: True, 6: True, 7: True, 8: True, 9: True}
- 时段3：{10: True, 11: False, 12: False, 13: True, 14: False, 15: False, 16: True, 17: True}

**每日舒适率：**

- 2024-07-21：78.26%
- 2024-07-22：80.00%

**结论：** 通过对采集的温湿度数据进行分析，我们发现：

- 在多个时段内温湿度均有较大波动。
- 在第一个和第二个时段内，温度和湿度均在舒适区间内，舒适率为100%。
- 在第三个时段内，部分时间温度和湿度不在舒适区间内，舒适率为50%。
- 每日的舒适率也有所波动，但总体在70%以上。

当前环境在大部分时间内是舒适的，但在个别时段内，温湿度可能超出舒适区间，需要进一步调控以提高整体舒适度。

# 4. 自定义数据分析（组内个人项目，组内每一个人自定义一个数据分析问题，以下两点二选一即可）：

## ① 个人设计简洁的UI界面或者自定义网页服务器，实现三种以上的传感器数据实时（或定时）显示及数据可视化分析，

# 实现步骤

## 1. 传感器数据读取

编写Python脚本，实现对各个传感器的数据读取。

```
pythonCopy codeimport PCF8591 as ADC
import RPi.GPIO as GPIO
import Adafruit_DHT
import Adafruit_BMP.BMP085 as BMP085
import os
import math
```

```
import smbus

DHT_PIN = 17
BH1750_DEVICE = 0x5c
ONE_TIME_HIGH_RES_MODE_1 = 0x20

def setup():
    ADC.setup(0x48)
    global bmp_sensor, dht_sensor, ds18b20_device, bus
    bmp_sensor = BMP085.BMP085()
    dht_sensor = Adafruit_DHT.DHT11
    ds18b20_device = setup_ds18b20()
    if (GPIO.RPI_REVISION == 1):
        bus = smbus.SMBus(0)
    else:
        bus = smbus.SMBus(1)

def setup_ds18b20():
    for i in os.listdir('/sys/bus/w1/devices'):
        if i != 'w1_bus_master1':
            return i

def read_ds18b20():
    location = f'/sys/bus/w1/devices/{ds18b20_device}/w1_slave'
    with open(location) as tfile:
        text = tfile.read()
    temperature_data = text.split("\n")[1].split(" ")[9]
    temperature = float(temperature_data[2:]) / 1000
    return round(temperature, 2)

def read_thermistor():
    analog_val = ADC.read(2)
    vr = 5 * float(analog_val) / 255
    rt = 10000 * vr / (5 - vr)
    temperature = 1 / ((math.log(rt / 10000) / 3950) + (1 / (273.15 + 25))) -
273.15
    return round(temperature, 2)

def convertToNumber(data):
    result = (data[1] + (256 * data[0])) / 1.2
    return result

def read_bh1750(addr=BH1750_DEVICE):
    data = bus.read_i2c_block_data(addr, ONE_TIME_HIGH_RES_MODE_1)
    return convertToNumber(data)

def read_sensors():
    photoresistor = ADC.read(1)
    bmp_temp = round(bmp_sensor.read_temperature(), 4)
    bmp_pressure = bmp_sensor.read_pressure()
    bmp_altitude = round(bmp_sensor.read_altitude(), 4)
    dht_humidity, dht_temp = Adafruit_DHT.read_retry(dht_sensor, DHT_PIN)
    if dht_humidity is not None and dht_temp is not None:
        dht_humidity = round(dht_humidity, 4)
        dht_temp = round(dht_temp, 4)
```

```
    ds18b20_temp = read_ds18b20()
    thermistor_temp = read_thermistor()
    bh1750_light = round(read_bh1750(), 4)
    return {
        'photoresistor': photoresistor,
        'bmp_temp': bmp_temp,
        'bmp_pressure': bmp_pressure,
        'bmp_altitude': bmp_altitude,
        'dht_temp': dht_temp,
        'dht_humidity': dht_humidity,
        'ds18b20_temp': ds18b20_temp,
        'thermistor_temp': thermistor_temp,
        'bh1750_light': bh1750_light
    }
```

## 2. 创建网页服务器

使用Flask框架创建一个简单的网页服务器，并实现传感器数据的实时传输。

```
#!/usr/bin/python3
import csv
import json
import PCF8591 as ADC
import RPi.GPIO as GPIO
import time
import datetime
import Adafruit_DHT
import Adafruit_BMP.BMP085 as BMP085
import os
import math
import smbus
from flask import Flask, Response, request, jsonify, render_template, send_file

# Flask app
app = Flask(__name__)

# DHT11 温湿度传感器管脚定义
DHT_PIN = 17

# 默认设备I2C地址
BH1750_DEVICE = 0x5c  # BH1750光传感器默认设备I2C地址

# 设置BH1750光传感器测量模式
ONE_TIME_HIGH_RES_MODE_1 = 0x20

# 用于保存传感器数据的列表
sensor_data_list = []

# 初始化各个传感器
def setup():
    ADC.setup(0x48)  # 设置PCF8591模块地址
    global bmp_sensor, dht_sensor, ds18b20_device, bus
    bmp_sensor = BMP085.BMP085()  # 气压传感器
    dht_sensor = Adafruit_DHT.DHT11  # 温湿度传感器
    ds18b20_device = setup_ds18b20()  # DS18B20温度传感器
```

```
    # 初始化I2C总线
    if (GPIO.RPI_REVISION == 1):
        bus = smbus.SMBus(0)
    else:
        bus = smbus.SMBus(1)


# 设置DS18B20温度传感器
def setup_ds18b20():
    for i in os.listdir('/sys/bus/w1/devices'):
        if i != 'w1_bus_master1':
            return i


# 读取DS18B20温度值
def read_ds18b20():
    location = f'/sys/bus/w1/devices/{ds18b20_device}/w1_slave'
    with open(location) as tfile:
        text = tfile.read()
    temperature_data = text.split("\n")[1].split(" ")[9]
    temperature = float(temperature_data[2:]) / 1000
    return round(temperature, 2)


# 读取热敏电阻温度值
def read_thermistor():
    analog_val = ADC.read(2)
    vr = 5 * float(analog_val) / 255
    rt = 10000 * vr / (5 - vr)
    temperature = 1 / ((math.log(rt / 10000) / 3950) + (1 / (273.15 + 25))) -
273.15
    return round(temperature, 2)


# 转换BH1750数据
def convertToNumber(data):
    result = (data[1] + (256 * data[0])) / 1.2
    return result


# 读取BH1750光传感器数据
def read_bh1750(addr=BH1750_DEVICE):
    data = bus.read_i2c_block_data(addr, ONE_TIME_HIGH_RES_MODE_1)
    return convertToNumber(data)


# 读取所有传感器数据
def read_sensors():
    photoresistor = ADC.read(1)
    bmp_temp = round(bmp_sensor.read_temperature(), 4)
    bmp_pressure = bmp_sensor.read_pressure()
    bmp_altitude = round(bmp_sensor.read_altitude(), 4)
    dht_humidity, dht_temp = Adafruit_DHT.read_retry(dht_sensor, DHT_PIN)
    if dht_humidity is not None and dht_temp is not None:
```

```
        dht_humidity = round(dht_humidity, 4)
        dht_temp = round(dht_temp, 4)
    ds18b20_temp = read_ds18b20()
    thermistor_temp = read_thermistor()
    bh1750_light = round(read_bh1750(), 4)

    data = {
        'time': datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
        'photoresistor': photoresistor,
        'bmp_temp': bmp_temp,
        'bmp_pressure': bmp_pressure,
        'bmp_altitude': bmp_altitude,
        'dht_temp': dht_temp,
        'dht_humidity': dht_humidity,
        'ds18b20_temp': ds18b20_temp,
        'thermistor_temp': thermistor_temp,
        'bh1750_light': bh1750_light
    }
    sensor_data_list.append(data)
    return data


# http://192.168.137.69:5000/sensor_data?interval=1
# SSE endpoint
@app.route('/sensor_data')
def sensor_data():
    def generate(interval):
        while True:
            start_time = time.time()
            data = read_sensors()
            yield f"data: {json.dumps(data)}\n\n"
            elapsed_time = time.time() - start_time
            sleep_time = max(0, interval - elapsed_time)
            time.sleep(sleep_time)

    interval = request.args.get('interval', default=1, type=int)
    return Response(generate(interval), mimetype='text/event-stream')


# 导出传感器数据为CSV
@app.route('/export')
def export():
    keys = sensor_data_list[0].keys()
    with open('/tmp/sensor_data.csv', 'w', newline='') as output_file:
        dict_writer = csv.DictWriter(output_file, fieldnames=keys)
        dict_writer.writeheader()
        dict_writer.writerows(sensor_data_list)
    return send_file('/tmp/sensor_data.csv', as_attachment=True)


# Web UI
@app.route('/')
def index():
    return render_template('index.html')
```

```
# Main entry point
if __name__ == '__main__':
    setup()
    app.run(host='0.0.0.0', port=5000)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sensor Data</title>
    <style>
        body { font-family: Arial, sans-serif; }
        .container { width: 80%; margin: 0 auto; padding: 20px; }
        h1 { text-align: center; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { padding: 10px; border: 1px solid #ddd; text-align: center; }
        th { background-color: #f4f4f4; }
        .interval-input { margin-top: 20px; text-align: center; }
        .chart-container { width: 100%; height: 500px; margin-top: 20px; }
        .download-btn { text-align: center; margin-top: 20px; }
    </style>
    <!-- 引入 ECharts 文件 -->
    <script src="https://cdn.jsdelivr.net/npm/echarts/dist/echarts.min.js">
</script>
</head>
<body>
    <div class="container">
        <h1>实时传感器数据</h1>
        <div class="interval-input">
            <label for="interval">刷新间隔（秒）：</label>
            <input type="number" id="interval" value="1" min="1">
            <button onclick="updateInterval()">更新间隔</button>
        </div>
        <table id="sensorData">
            <tr>
                <th>传感器</th>
                <th>数据</th>
            </tr>
            <tr><td>光敏电阻</td><td id="photoresistor">--</td></tr>
            <tr><td>BMP 温度 (°C)</td><td id="bmp_temp">--</td></tr>
            <tr><td>BMP 气压 (Pa)</td><td id="bmp_pressure">--</td></tr>
            <tr><td>BMP 海拔 (m)</td><td id="bmp_altitude">--</td></tr>
            <tr><td>DHT 温度 (°C)</td><td id="dht_temp">--</td></tr>
            <tr><td>DHT 湿度 (%)</td><td id="dht_humidity">--</td></tr>
            <tr><td>DS18B20 温度 (°C)</td><td id="ds18b20_temp">--</td></tr>
            <tr><td>热敏电阻 温度 (°C)</td><td id="thermistor_temp">--</td></tr>
            <tr><td>BH1750 光强 (lx)</td><td id="bh1750_light">--</td></tr>
        </table>
        <div class="download-btn">
            <a href="/export" download>导出数据为CSV</a>
        </div>
        <!-- 创建用于显示图表的 DOM 节点 -->
        <div class="chart-container" id="temperatureChart"></div>
```

```
        <div class="chart-container" id="pressureChart"></div>
        <div class="chart-container" id="lightChart"></div>
        <div class="chart-container" id="photoresistorChart"></div>
    </div>
    <script>
        var interval = document.getElementById('interval').value;
        var source = new EventSource('/sensor_data?interval=' + interval);

        // 初始化 ECharts 实例
        var temperatureChart =
echarts.init(document.getElementById('temperatureChart'));
        var pressureChart =
echarts.init(document.getElementById('pressureChart'));
        var lightChart = echarts.init(document.getElementById('lightChart'));
        var photoresistorChart =
echarts.init(document.getElementById('photoresistorChart'));

        // 温度图表配置项
        var temperatureOption = {
            title: {
                text: '温度传感器数据趋势'
            },
            tooltip: {
                trigger: 'axis'
            },
            legend: {
                data: ['BMP 温度', 'DHT 温度', 'DS18B20 温度', '热敏电阻 温度']
            },
            xAxis: {
                type: 'category',
                boundaryGap: false,
                data: []
            },
            yAxis: {
                type: 'value'
            },
            series: [
                {name: 'BMP 温度', type: 'line', data: []},
                {name: 'DHT 温度', type: 'line', data: []},
                {name: 'DS18B20 温度', type: 'line', data: []},
                {name: '热敏电阻 温度', type: 'line', data: []}
            ],
            dataZoom: [
                {
                    type: 'slider',
                    start: 0,
                    end: 100
                },
                {
                    type: 'inside',
                    start: 0,
                    end: 100
                }
            ]
        };
```

```
        // 气压图表配置项
        var pressureOption = {
            title: {
                text: '气压传感器数据趋势'
            },
            tooltip: {
                trigger: 'axis'
            },
            legend: {
                data: ['BMP 气压']
            },
            xAxis: {
                type: 'category',
                boundaryGap: false,
                data: []
            },
            yAxis: {
                type: 'value'
            },
            series: [
                {name: 'BMP 气压', type: 'line', data: []}
            ],
            dataZoom: [
                {
                    type: 'slider',
                    start: 0,
                    end: 100
                },
                {
                    type: 'inside',
                    start: 0,
                    end: 100
                }
            ]
        };

        // 光强图表配置项
        var lightOption = {
            title: {
                text: '光强传感器数据趋势'
            },
            tooltip: {
                trigger: 'axis'
            },
            legend: {
                data: ['BH1750 光强']
            },
            xAxis: {
                type: 'category',
                boundaryGap: false,
                data: []
            },
            yAxis: {
                type: 'value'
```

```
        },
        series: [
            {name: 'BH1750 光强', type: 'line', data: []}
        ],
        dataZoom: [
            {
                type: 'slider',
                start: 0,
                end: 100
            },
            {
                type: 'inside',
                start: 0,
                end: 100
            }
        ]
    };

    // 光敏电阻图表配置项
    var photoresistorOption = {
        title: {
            text: '光敏电阻数据趋势'
        },
        tooltip: {
            trigger: 'axis'
        },
        legend: {
            data: ['光敏电阻']
        },
        xAxis: {
            type: 'category',
            boundaryGap: false,
            data: []
        },
        yAxis: {
            type: 'value'
        },
        series: [
            {name: '光敏电阻', type: 'line', data: []}
        ],
        dataZoom: [
            {
                type: 'slider',
                start: 0,
                end: 100
            },
            {
                type: 'inside',
                start: 0,
                end: 100
            }
        ]
    };

    // 使用配置项生成图表
```

```
        temperatureChart.setOption(temperatureOption);
        pressureChart.setOption(pressureOption);
        lightChart.setOption(lightOption);
        photoresistorChart.setOption(photoresistorOption);

        source.addEventListener('message', function(e) {
            var data = JSON.parse(e.data);
            var currentTime = new Date().toLocaleTimeString();

            // 更新 x 轴数据
            temperatureOption.xAxis.data.push(currentTime);
            pressureOption.xAxis.data.push(currentTime);
            lightOption.xAxis.data.push(currentTime);
            photoresistorOption.xAxis.data.push(currentTime);
            if (temperatureOption.xAxis.data.length > 20) {
                temperatureOption.xAxis.data.shift();
                pressureOption.xAxis.data.shift();
                lightOption.xAxis.data.shift();
                photoresistorOption.xAxis.data.shift();
            }

            // 更新系列数据
            temperatureOption.series[0].data.push(data.bmp_temp);
            temperatureOption.series[1].data.push(data.dht_temp);
            temperatureOption.series[2].data.push(data.ds18b20_temp);
            temperatureOption.series[3].data.push(data.thermistor_temp);

            pressureOption.series[0].data.push(data.bmp_pressure);

            lightOption.series[0].data.push(data.bh1750_light);

            photoresistorOption.series[0].data.push(data.photoresistor);

            temperatureChart.setOption(temperatureOption);
            pressureChart.setOption(pressureOption);
            lightChart.setOption(lightOption);
            photoresistorChart.setOption(photoresistorOption);

            document.getElementById('photoresistor').innerText =
data.photoresistor + ' units';
            document.getElementById('bmp_temp').innerText = data.bmp_temp + '
°C';
            document.getElementById('bmp_pressure').innerText =
data.bmp_pressure + ' Pa';
            document.getElementById('bmp_altitude').innerText =
data.bmp_altitude + ' m';
            document.getElementById('dht_temp').innerText = data.dht_temp + '
°C';
            document.getElementById('dht_humidity').innerText =
data.dht_humidity + ' %';
            document.getElementById('ds18b20_temp').innerText =
data.ds18b20_temp + ' °C';
            document.getElementById('thermistor_temp').innerText =
data.thermistor_temp + ' °C';
```

```
            document.getElementById('bh1750_light').innerText =
data.bh1750_light + ' lx';
        }, false);

        function updateInterval() {
            interval = document.getElementById('interval').value;
            source.close();
            source = new EventSource('/sensor_data?interval=' + interval);
            source.addEventListener('message', function(e) {
                var data = JSON.parse(e.data);
                var currentTime = new Date().toLocaleTimeString();

                // 更新 X 轴数据
                temperatureOption.xAxis.data.push(currentTime);
                pressureOption.xAxis.data.push(currentTime);
                lightOption.xAxis.data.push(currentTime);
                photoresistorOption.xAxis.data.push(currentTime);
                if (temperatureOption.xAxis.data.length > 20) {
                    temperatureOption.xAxis.data.shift();
                    pressureOption.xAxis.data.shift();
                    lightOption.xAxis.data.shift();
                    photoresistorOption.xAxis.data.shift();
                }

                // 更新系列数据
                temperatureOption.series[0].data.push(data.bmp_temp);
                temperatureOption.series[1].data.push(data.dht_temp);
                temperatureOption.series[2].data.push(data.ds18b20_temp);
                temperatureOption.series[3].data.push(data.thermistor_temp);

                pressureOption.series[0].data.push(data.bmp_pressure);

                lightOption.series[0].data.push(data.bh1750_light);

                photoresistorOption.series[0].data.push(data.photoresistor);

                temperatureChart.setOption(temperatureOption);
                pressureChart.setOption(pressureOption);
                lightChart.setOption(lightOption);
                photoresistorChart.setOption(photoresistorOption);

                document.getElementById('photoresistor').innerText =
data.photoresistor + ' units';
                document.getElementById('bmp_temp').innerText = data.bmp_temp +
' °C';
                document.getElementById('bmp_pressure').innerText =
data.bmp_pressure + ' Pa';
                document.getElementById('bmp_altitude').innerText =
data.bmp_altitude + ' m';
                document.getElementById('dht_temp').innerText = data.dht_temp +
' °C';
                document.getElementById('dht_humidity').innerText =
data.dht_humidity + ' %';
                document.getElementById('ds18b20_temp').innerText =
data.ds18b20_temp + ' °C';
```

```
                document.getElementById('thermistor_temp').innerText =
data.thermistor_temp + ' °C';
                document.getElementById('bh1750_light').innerText =
data.bh1750_light + ' lx';
            }, false);
        }
    </script>
</body>
</html>
```

## 结果分析

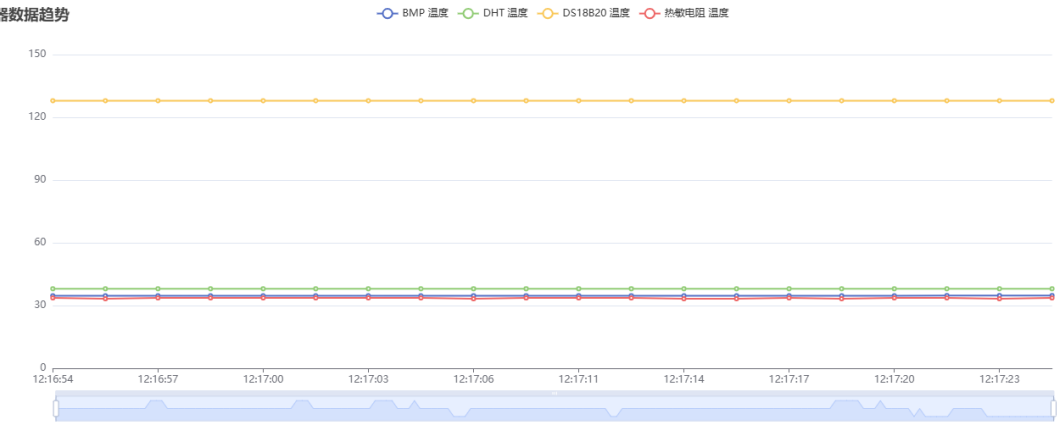
# 实时传感器数据

刷新间隔（秒）： 1 更新间隔

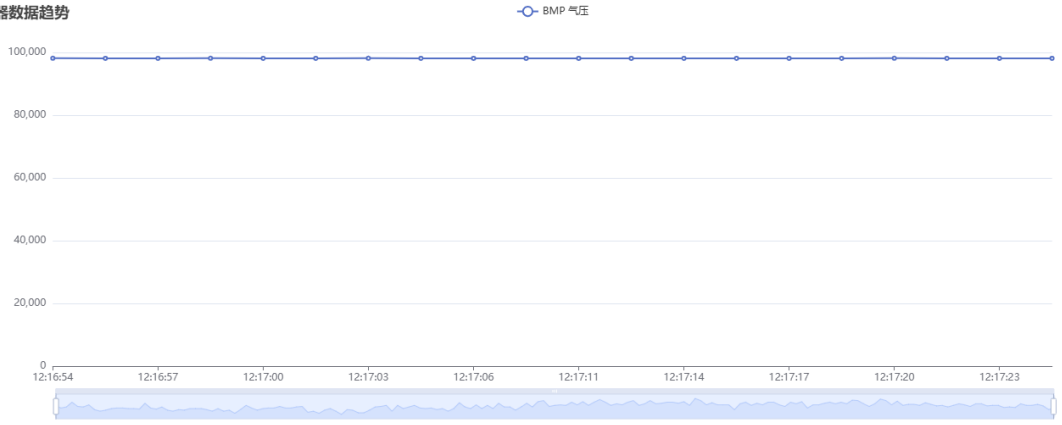
| 传感器 | 数据 |
| --- | --- |
| 光敏电阻 | 24 units |
| BMP 温度 (°C) | 34.6 °C |
| BMP 气压 (Pa) | 98139 Pa |
| BMP 海拔 (m) | 267.837 m |
| DHT 温度 (°C) | 38 °C |
| DHT 湿度 (%) | 35 % |
| DS18B20 温度 (°C) | 127.94 °C |
| 热敏电阻 温度 (°C) | 33.25 °C |
| BH1750 光强 (lx) | 1972.5 lx |

导出数据为CSV

温度传感器数据趋势

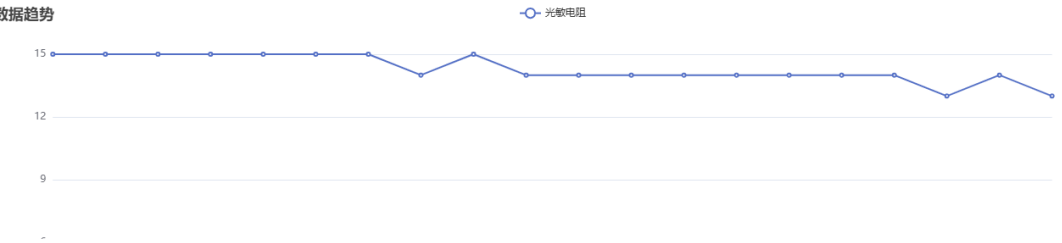


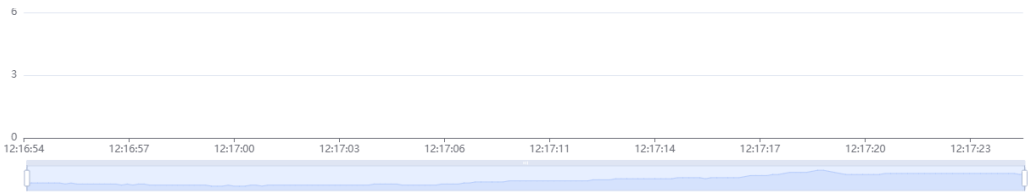

气压传感器数据趋势




光强传感器数据趋势


光敏电阻数据趋势






# 三、结果评价及思考

1、简述你对举办课内竞赛的看法，并叙述如何能够进一步提升“数据挖掘及机器学习”系统中天气预测竞赛的成绩？

举办课内竞赛激发了学生的学习兴趣和实践热情。为进一步提升“数据挖掘及机器学习”系统中天气预测竞赛的成绩，可以考虑引入更多样化的数据源、优化特征工程及模型选择，并加强团队合作和知识共享。

2、思考简述“数据传感器采集、分析及可视化”的更多应用场景（2-3个），并选择其中一个新的应用场景谈一谈你如何实施该场景下的人工智能项目实践？

**智能家居**：通过传感器数据分析实现家居环境自动调节，如智能温控、照明控制等。

**健康监测**：利用传感器数据进行个人健康指标监测和预警，如心率、血压等。

**选择的应用场景及实施方案**： 在智能农业中，利用传感器采集土壤湿度、温度、光照等数据，通过数据分析和可视化，实现精准灌溉、施肥等智能化农业管理，提高农业生产效率和资源利用率。