融合模糊粒信息的医疗辅助诊断系统源代码

```
from flask import Flask
from app.extension import db, cors
from app.config import Config
from app.views import bp as api_bp
def create_app():
    app = Flask(__name__)
    app.config.from_object(Config)
    db.init_app(app)
    cors.init_app(app)
    app.register_blueprint(api_bp)
    @app.cli.command()
    def create():
        db.drop_all()
        db.create_all()
    return app
from app.__init__ import create_app
app = create_app()
class Config:
    HOSTNAME = "127.0.0.1"
    PORT = 3306
    USERNAME = "root"
    PASSWORD = "gyztt030607"
    DATABASE = "flasktest"
    SQLALCHEMY_DATABASE_URI =
f"mysql+pymysql://{USERNAME}:{PASSWORD}@{HOSTNAME}:{PORT}/{DATABASE}?charset=utf8mb4"
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    SQLALCHEMY_ECHO = False
    TEMPLATES_AUTO_RELOAD = True
    SEND_FILE_MAX_AGE_DEFAULT = 0
from flask_sqlalchemy import SQLAlchemy
from flask_cors import CORS
db = SQLAlchemy()
cors = CORS()
# -*- coding = utf-8 -*-
# @Time : 2023/7/24
# @Author : wuyan
# @File : KFRAD.py
# @Software : PyCharm
import numpy as np
import xlrd
from scipy.io import loadmat
from scipy.spatial.distance import pdist, squareform
# Kernelized fuzzy rough anomaly detection(KFRAD)
# 计算关系矩阵的函数：输入属性子集数据和核参数，输出模糊关系矩阵
# data 需是矩阵类型,多属性子集是矩阵类型
# 单属性子集可用 list 矩阵化后转置:temp=gaussian_matrix((np.matrix(a[:,0])).T,r)或者
temp=gaussian_matrix(a[:,0:1],r)
def gaussian_matrix(data, r):
    n = data.shape[0] # 获取矩阵行数，n 个样本
    m = data.shape[1] # 获取矩阵列数，m 个属性
    datatrans = np.zeros((n, m))
    datatrans[:, 0:m] = data
    temp = pdist(datatrans, 'euclidean') # 计算欧式距离
    temp = squareform(temp)
    temp = np.exp(-(temp ** 2) / r) # 用核函数求隶属度
    return temp
```

```
# input:
# data is data matrix without decisions, where rows for samples and columns for attributes.
# lammda is used to adjust the adaptive fuzzy radius.
# output:
# Ranking objects and fuzzy rough granules-based outlier factor (FRGOF)
def KFRAD(data, delta):
    n, m = data.shape
    # 计算第一个条件属性的邻域集合 ------------------------------
    LA = np.arange(0, m)  # 属性集合序号 0~m-1，对应 1~m
    weight1 = np.zeros((n, m))  # 单属性权重
    weight3 = np.zeros((n, m))  # 单属性权重
    Acc_A_a = np.zeros((n, m))  # 去掉一个属性之后的近似精度
    for l in range(0, m):
        lA_d = np.setdiff1d(LA, l)  # 在 LA 中但不在 l 中的已排序的唯一值
        # Acc_A_a_tem = np.zeros((n, m))
        # 求单属性子集的模糊关系矩阵
        NbrSet_tem = gaussian_matrix((np.matrix(data[:, l])).T, delta)
        NbrSet_temp, ia, ic = np.unique(NbrSet_tem, return_index=True, return_inverse=True, axis=0)
        # ia 为矩阵 NbrSet_temp 中的元素在矩阵 NbrSet_tem 中的位置
        # ic 为矩阵 NbrSet_tem 中的元素在矩阵 NbrSet_temp 中的位置
        # NbrSet_temp 去除了 NbrSet_tem 的重复元素并升序排列
        for i in range(0, NbrSet_temp.shape[0]):  # NbrSet_temp 的行数
            i_tem = np.where(ic == i)[0]
            data_tem = data[:, lA_d]
            NbrSet_tmp = gaussian_matrix(data_tem, delta)  # 多属性子集的模糊关系矩阵
            a = 1 - NbrSet_tmp
            b = np.tile(NbrSet_temp[i, :], (n, 1))
            Low_A = sum((np.minimum(a + b - np.multiply(a, b) + np.multiply(np.sqrt(2 * a - np.multiply(a, a)),
                                                                            np.sqrt(2 * b - np.multiply(b, b))),
                                    1)).min(-1))
            a = NbrSet_tmp
            Upp_A = sum((np.maximum(
                np.multiply(a, b) - np.multiply(np.sqrt(1 - np.multiply(a, a)), np.sqrt(1 - np.multiply(b, b))),
                0)).max(-1))
            Acc_A_a[i_tem, l] = Low_A / Upp_A  # 计算近似精度
            weight3[i_tem, l] = 1 - (sum(NbrSet_temp[i, :]) / n) ** (1 / 3)
            weight1[i_tem, l] = (sum(NbrSet_temp[i, :]) / n)
    GAL = np.zeros((n, m))
    for col in range(m):
        GAL[:, col] = 1 - (Acc_A_a[:, col]) * weight1[:, col]
    KFRAD = np.array(np.mean(GAL * weight3, axis=1))
    KFRAD = np.round(KFRAD, 3)
    return KFRAD
# min-max-normalize-----------------------------------------------
# start,end 是归一化的列范围
# 注意：此处 start,end 的取值在 1~m 之间
def normalize(data, start, end):
    n = data.shape[0]  # 获取矩阵行数
    m = data.shape[1]  # 获取矩阵列数
    trandata = np.zeros((n, m - 1))
    if start < 1 or end > m or start > end:
        print("范围出错")
    else:
        for i in range(start - 1, end):
            temp = data[:, i]
            max = np.max(temp)
```

```
            min = np.min(temp)
            if (max > 1):
                dis = max - min
                temp = (temp - min) / dis
            trandata[:, i] = temp
    return trandata
import numpy as np
import pandas as pd
from scipy.spatial.distance import pdist, squareform
# 计算模糊多粒度熵
def entropy(M):
    a = M.shape[0]
    K = sum(-(1 / a) * (np.log2(1 / (M.sum(axis=0)))))
    return K
# 计算关系矩阵
def rela_srr(data, delta):
    datatrans = np.zeros((len(data), 2))
    datatrans[:, 0] = data
    temp = pdist(datatrans, 'cityblock') # 计算曼哈顿距离，差的绝对值
    temp = squareform(temp) # 把得到的一维矩阵转化为二维矩阵
    # temp[temp > (1 - delta)] = 1
    temp[temp > delta] = 1
    r = 1 - temp
    return r
# 算法的主要实现
def MNIFS(data, lammda):
    # 获取 data 的行列数，row:对象数，attrinu:属性数
    row, attrinu = data.shape
    # delta 是属性的多邻域半径
    delta = np.zeros(attrinu)
    ID = (data <= 1).all(axis=0)
    delta[ID] = (np.std(data[:, ID], axis=0)) / lammda
    Select_Fea = [] # 已被选择的特征
    sig = [] # 用于记录每个特征被选择时的指标值
    E = np.zeros(attrinu) # E 是模糊多粒度熵
    Joint_E = np.zeros((attrinu, attrinu)) # Joint_E 是模糊多粒度联合熵
    MI = np.zeros((attrinu, attrinu)) # MI 是模糊互信息
    # 计算模糊多粒度熵
    for j in range(0, attrinu):
        r = rela_srr(data[:, j], delta[j]) # r 是属性 j 对应的模糊多邻域颗粒
        E[j] = entropy(r)
    # print("模糊多粒度熵：")
    # print(E)
    # 计算模糊多粒度联合熵和模糊互信息
    for i in range(0, attrinu):
        ri = rela_srr(data[:, i], delta[i]) # ri 是属性 i 对应的模糊集
        for j in range(0, i + 1):
            rj = rela_srr(data[:, j], delta[j]) # rj 是属性 j 对应的模糊集
            Joint_E[i, j] = entropy(np.minimum(ri, rj)) # 计算 i 和 j 的联合熵
            Joint_E[j, i] = Joint_E[i, j]
            # i 和 j 的模糊互信息等于 i 和 j 的模糊多粒度熵之和减去 i 和 j 的模糊联合熵
            MI[i, j] = E[i] + E[j] - Joint_E[i, j]
            MI[j, i] = MI[i, j]
    # print('模糊多粒度联合熵：')
    # print(Joint_E)
```

```
#
# print('模糊互信息 FMI：')
# print(MI)
# 计算模糊相关性，即模糊互信息的平均值
Ave_MI = np.mean(MI, axis=1)  # 计算每一行的均值
n1 = (np.argsort(Ave_MI)[::-1]).tolist()  # 排序后的索引值
x1 = (Ave_MI[np.argsort(Ave_MI)[::-1]]).tolist()  # 排序后的 Ave_MI
# print("模糊相关性:")
# print(Ave_MI)
sig.append(x1[0])
Select_Fea.append(n1[0])
unSelect_Fea = n1[1:]
# 计算未被选择的特征的冗余度
while unSelect_Fea:
    Red = np.zeros((len(unSelect_Fea), len(Select_Fea)))  # Red：冗余度
    # i 是未选择特征
    for i in range(0, len(unSelect_Fea)):
        # j 是已选择特征
        for j in range(0, len(Select_Fea)):
            # FE = Joint_E[Select_Fea[j], unSelect_Fea[i]] - E[unSelect_Fea[i]]
            FE = Joint_E[Select_Fea[j], unSelect_Fea[i]]
            Red[i, j] = Ave_MI[Select_Fea[j]] - FE / E[Select_Fea[j]] * Ave_MI[Select_Fea[j]]
    Ave_FRed = np.mean(Red, axis=1)  # 计算每个未选择特征对应的冗余度平均值
    # print('冗余度：')
    # print(Ave_FRed)
    # 计算交互性
    Itr = np.zeros((len(unSelect_Fea), len(unSelect_Fea)))  # Itr:交互性
    # 如果只剩一个未选择特征，则交互性为 0
    if len(unSelect_Fea) == 1:
        Ave_Itr = np.sum(Itr, axis=1)
    else:
        # 使用交点法计算关系矩阵
        srrcj = np.ones((row, row))
        for j in range(0, len(Select_Fea)):
            srr_Select_j = rela_srr(data[:, Select_Fea[j]], delta[Select_Fea[j]])
            srrcj = np.minimum(srrcj, srr_Select_j)
        # 遍历所有未选特征，计算当前候选特征 c 的交互性
        for c in range(0, len(unSelect_Fea)):
            # 遍历所有未选特征
            for i in range(0, len(unSelect_Fea)):
                if c == i:
                    continue
                # 计算交互性，使用公式 I[p;q|t]=Joint_E[p,t]+Joint_E[q,t]-Joint_E[p,q,t]-E[t]
                srr_UnSe_i = rela_srr(data[:, unSelect_Fea[i]], delta[unSelect_Fea[i]])
                srr_UnSe_c = rela_srr(data[:, unSelect_Fea[c]], delta[unSelect_Fea[c]])
                Joint_Three = entropy(np.minimum(np.minimum(srr_UnSe_i, srrcj), srr_UnSe_c))
                Joint_Two = entropy(np.minimum(srrcj, srr_UnSe_c))
                Itr[c, i] = Joint_E[unSelect_Fea[i], unSelect_Fea[c]] + Joint_Two - Joint_Three - E[unSelect_Fea[c]]
                Itr[c, i] = np.abs(Itr[c, i])
                # print(Joint_E[unSelect_Fea[i], unSelect_Fea[c]] ,Joint_Two , Joint_Three , E[unSelect_Fea[c]])
                # print(Itr[c, i],'c:',c,'i:',i)
        # print(Itr)
        Ave_Itr = np.sum(Itr, axis=1)
        Ave_Itr = Ave_Itr / (len(unSelect_Fea) - 1)
        # print('交互性：')
```

```
            # print(Ave_Itr)
        # 计算最大相关最小冗余最大交互
        UFmRMR = Ave_MI[unSelect_Fea] - Ave_FRed + Ave_Itr
        UFmRMR = UFmRMR.tolist()
        # print('评价指标：')
        # print(UFmRMR)
        max_sig = max(UFmRMR)
        max_tem = UFmRMR.index(max(UFmRMR))
        sig.append(max_sig)
        Select_Fea.append(unSelect_Fea[max_tem])
        unSelect_Fea.pop(max_tem)
    select_feature = Select_Fea
    # print('相关度是：', sig)
    # print('特征序列：', select_feature)
    return select_feature, sig
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
def kmeans_clustering(data, num_clusters):
    """
    使用 K-means 算法进行聚类
    参数：
    - data: 包含要进行聚类的数据的二维数组
    - num_clusters: 聚类簇的数量
    返回：
    - labels: 每个样本所属的聚类标签
    - centroids: 聚类中心点的坐标
    """
    # 数据归一化
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(data)
    # 创建 K-means 聚类器
    kmeans = KMeans(n_clusters=num_clusters)
    # 执行聚类
    kmeans.fit(scaled_data)
    # 获取聚类标签
    labels = kmeans.labels_
    # 将数字标签映射为文字标签
    label_map = {0: "高血压型心脏病", 1: "冠心病", 2: "心肌炎"}
    # 将数字标签转换为文字标签
    labeled_data = np.array([label_map[label] for label in labels])
    return labeled_data
from app.views.utils.file import file_view
from app.views.detection.detection import detection_view
from app.views.utils.image import image_view
from app.views.reduction.reduction import reduction_view
from app.views.classification.classification import classification_view
from flask import Blueprint
bp = Blueprint('api', __name__)
bp.add_url_rule('/file', view_func=file_view, methods=['POST'])
bp.add_url_rule('/detection', view_func=detection_view, methods=['POST'])
bp.add_url_rule('/reduction', view_func=reduction_view, methods=['POST'])
bp.add_url_rule('/classification', view_func=classification_view, methods=['POST'])
bp.add_url_rule('/image/<path:filename>', view_func=image_view, methods=['GET'])
import json
import os
```

```
import numpy as np
import pandas as pd
from flask import request
from flask.views import MethodView
from matplotlib import pyplot as plt
from app.functions.kmeans import kmeans_clustering
class ClassificationApi(MethodView):
    def __init__(self):
        self.col_count = None
        self.row_count = None
    def classification(self, data_matrix, k):
        output = kmeans_clustering(data_matrix, k)
        # print(self.result)
        # matrix = np.reshape(output, (-1, 1)) # 使用 -1 表示自动推断行数，1 表示只有一列
        final_matrix = np.concatenate((data_matrix, output[:, np.newaxis]), axis=1)
        # 创建一个包含适当 id 的数组
        num_rows = final_matrix.shape[0] # 确定矩阵的行数
        id_column = np.arange(1, num_rows + 1)[:, np.newaxis] # 创建一个列向量，从 1 开始的 id
        # 将 id 列插入到矩阵的最前面一列
        final_matrix = np.concatenate((id_column, final_matrix), axis=1)
        data_json = json.dumps(final_matrix.tolist())
        return data_json, final_matrix
    def echarts(self, matrix):
        # 提取最后一列数据
        last_column = matrix[:, -1]
        # 获取最后一列的唯一值及其对应的出现次数
        unique_values, counts = np.unique(last_column, return_counts=True)
        # 将唯一值和对应的出现次数存储到两个列表中
        x_data = unique_values.tolist()
        y_data = counts.tolist()
        # 构建 option 对象
        option = {
            "xAxis": {
                "type": 'category',
                "data": x_data
            },
            "yAxis": {
                "type": 'value'
            },
            "series": [
                {
                    "data": y_data,
                    "type": 'bar'
                }
            ]
        }
        # 输出 JSON 字符串
        return option
    def post(self):
        # 检查是否有文件上传
        if 'file' not in request.files:
            return {
                'status': 'error',
                'message': 'No file provided',
                'code': 1
            }
```

```
# 获取上传的文件
file = request.files['file']
is_classification = request.form.get('is_classification')
k = request.form.get('k')
if k is not None:
    k = int(k)
else:
    k = 3
# 检查文件是否存在
if file.filename == '':
    return {
        'status': 'error',
        'message': 'No file selected',
        'code': 2
    }
# 指定相对路径保存文件，相对于当前工作目录
upload_folder = 'file'
# 获取当前工作目录
current_directory = os.getcwd()
# 构建保存文件的完整路径
save_path = os.path.join(current_directory, upload_folder, file.filename)
# 保存文件
file.save(save_path)
# 从 CSV 文件中加载数据到 DataFrame
data_frame = pd.read_csv(save_path, encoding='utf-8', header=None)
self.row_count = data_frame.shape[0]
self.col_count = data_frame.shape[1] - 1
data_matrix = data_frame.values
data_matrix = np.delete(data_matrix, data_matrix.shape[1] - 1, axis=1)  # 删除最后一列
# 获取列的数量
num_columns = len(data_frame.columns) - 1
# 生成对应的列名
header = [f'指标{i}' for i in range(1, num_columns + 1)]
# 提取 columns 信息
detail_columns_json = {str(index): value for index, value in enumerate(header)}
# 添加 result 元素
header.append("分类结果")
header.insert(0, "id")
columns_json = {str(index): value for index, value in enumerate(header)}
detail_data = json.dumps(data_matrix.tolist())
if is_classification:
    data, matrix = self.classification(data_matrix, k)
    option = self.echarts(matrix)
    # 返回成功消息
    return {
        'status': 'success',
        'message': '分类完毕',
        'data': data,
        'header': columns_json,
        'option': option,
    }
else:
    # 返回成功消息
    return {
        'status': 'success',
        'message': '上传完成',
```

```
            'row_count': self.row_count,
            'col_count': self.col_count,
            'file': file.filename,
            'detail_data': detail_data,
            'detail_header': detail_columns_json
        }
classification_view = ClassificationApi.as_view('classification_api')
import json
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from flask import request
from flask.views import MethodView
from app.functions.KFRAD import KFRAD
class DetectionApi(MethodView):
    def __init__(self):
        self.col_count = None
        self.row_count = None
        self.result = None
        self.n = ''
    def detection(self, data_matrix, delta):
        output = KFRAD(data_matrix, delta)
        self.result = output
        final_matrix = np.concatenate((data_matrix, output[:, np.newaxis]), axis=1)
        # 创建一个包含适当 id 的数组
        num_rows = final_matrix.shape[0]  # 确定矩阵的行数
        id_column = np.arange(1, num_rows + 1)[:, np.newaxis]  # 创建一个列向量，从 1 开始的 id
        # 将 id 列插入到矩阵的最前面一列
        final_matrix = np.concatenate((id_column, final_matrix), axis=1)
        data_json = json.dumps(final_matrix.tolist())
        return data_json
    def echarts(self):
        # 使用列表推导式生成 scatter_data 列表
        scatter_data = [[i + 1, value] for i, value in enumerate(self.result)]
        if self.n != '':
            linevalue = self.n
        else:
            linevalue = 0
        # 构建 option 对象
        option = {
            "xAxis": {},
            "yAxis": {},
            "series": [
                {
                    "symbolSize": 10,
                    "data": scatter_data,
                    "type": "scatter",
                    "markLine": {
                        "symbol": ["none"],
                        "silent": True,
                        "lineStyle": {
                            "type": "solid",
                            "color": "red"
                        },
                        "label": {
                            "position": "middle"
```

```
                },
                "data": [
                    {
                        "yAxis": linevalue
                    }
                ]
            }
        }
    ]
}
# 将 Python 字典转换为 JSON 字符串
# json_string = json.dumps(option, indent=2)
# 输出 JSON 字符串
return option
def post(self):
    # 检查是否有文件上传
    if 'file' not in request.files:
        return {
            'status': 'error',
            'message': 'No file provided',
            'code': 1
        }
    # 获取上传的文件
    file = request.files['file']
    self.n = request.form.get('n')
    is_detection = request.form.get('is_detection')
    delta = request.form.get('delta')
    if delta is not None:
        delta = float(delta)
    # 检查文件是否存在
    if file.filename == '':
        return {
            'status': 'error',
            'message': 'No file selected',
            'code': 2
        }
    # 指定相对路径保存文件，相对于当前工作目录
    upload_folder = 'file'
    # 获取当前工作目录
    current_directory = os.getcwd()
    # 构建保存文件的完整路径
    save_path = os.path.join(current_directory, upload_folder, file.filename)
    # 保存文件
    file.save(save_path)
    # 从 CSV 文件中加载数据到 DataFrame
    data_frame = pd.read_csv(save_path, encoding='utf-8', header=None)
    self.row_count = data_frame.shape[0]
    self.col_count = data_frame.shape[1] - 1
    data_matrix = data_frame.values
    data_matrix = np.delete(data_matrix, data_matrix.shape[1] - 1, axis=1)  # 删除最后一列
    # 获取列的数量
    num_columns = len(data_frame.columns) - 1
    # 生成对应的列名
    header = [f'指标{i}' for i in range(1, num_columns + 1)]
    # 提取 columns 信息
    detail_columns_json = {str(index): value for index, value in enumerate(header)}
```

```
        # 添加 result 元素
        header.append("异常分数")
        header.insert(0, "id")
        columns_json = {str(index): value for index, value in enumerate(header)}
        detail_data = json.dumps(data_matrix.tolist())
        if is_detection:
            data = self.detection(data_matrix, delta)
            option = self.echarts()
            # 返回成功消息
            return {
                'status': 'success',
                'message': '检测完毕',
                'data': data,
                'header': columns_json,
                'option': option,
            }
        else:
            # 返回成功消息
            return {
                'status': 'success',
                'message': '上传完成',
                'row_count': self.row_count,
                'col_count': self.col_count,
                'file': file.filename,
                'detail_data': detail_data,
                'detail_header': detail_columns_json
            }
detection_view = DetectionApi.as_view('detection_api')
import json
import os
import numpy as np
import pandas as pd
from flask import request
from flask.views import MethodView
from app.functions.demo_new_ltr import MNIFS
class ReductionApi(MethodView):
    def __init__(self):
        self.col_count = None
        self.row_count = None
    def reduction(self, data_matrix, lammda):
        select_feature, sig = MNIFS(data_matrix, lammda)
        # 将两个列表组合成一个矩阵
        matrix = np.column_stack((np.array(select_feature) + 1, np.array(sig)))
        # 生成排名列数据
        new_column = np.arange(1, len(matrix) + 1)
        # 将排名列与原始矩阵连接起来
        matrix = np.column_stack((matrix, new_column))
        # 按矩阵的第一列数值从小到大排序
        sorted_matrix = matrix[matrix[:, 0].argsort()]
        data_json = json.dumps(sorted_matrix.tolist())
        return data_json, sorted_matrix
    def echarts(self, matrix):
        x_data = matrix[:, 0].tolist()
        y_data = [len(matrix[:, 2].tolist()) + 1 - data for data in matrix[:, 2].tolist()]
        # 构建 option 对象
        option = {
            "tooltip": {
```

```
            "trigger": 'axis',
            "axisPointer": {
                "type": 'shadow'
            }
        },
        "grid": {
            "left": '3%',
            "right": '4%',
            "bottom": '3%',
            "containLabel": "true"
        },
        "xAxis": {
            "type": 'category',
            "data": x_data,
        },
        "yAxis": {
            "type": 'value',
            "name": '指标重要程度',
        },
        "series": [
            {
                "data": y_data,
                "type": 'bar'
            }
        ]
    }
    # 输出 JSON 字符串
    return option
def post(self):
    # 检查是否有文件上传
    if 'file' not in request.files:
        return {
            'status': 'error',
            'message': 'No file provided',
            'code': 1
        }
    # 获取上传的文件
    file = request.files['file']
    is_reduction = request.form.get('is_reduction')
    lammda = request.form.get('lammda')
    if lammda is not None:
        lammda = float(lammda)
    else:
        lammda = 0.4
    # 检查文件是否存在
    if file.filename == '':
        return {
            'status': 'error',
            'message': 'No file selected',
            'code': 2
        }
    # 指定相对路径保存文件，相对于当前工作目录
    upload_folder = 'file'
    # 获取当前工作目录
    current_directory = os.getcwd()
    # 构建保存文件的完整路径
    save_path = os.path.join(current_directory, upload_folder, file.filename)
```

```
        # 保存文件
        file.save(save_path)
        # 从 CSV 文件中加载数据到 DataFrame
        data_frame = pd.read_csv(save_path, encoding='utf-8', header=None)
        self.row_count = data_frame.shape[0]
        self.col_count = data_frame.shape[1] - 1
        data_matrix = data_frame.values
        data_matrix = np.delete(data_matrix, data_matrix.shape[1] - 1, axis=1) # 删除最后一列
        # 获取列的数量
        num_columns = len(data_frame.columns) - 1
        # 生成对应的列名
        header_detail = [f'指标{i}' for i in range(1, num_columns + 1)]
        # 提取 columns 信息
        detail_columns_json = {str(index): value for index, value in enumerate(header_detail)}
        detail_data = json.dumps(data_matrix.tolist())
        if is_reduction:
            data, matrix = self.reduction(data_matrix, lammda)
            # 生成对应的列名
            header = ["指标编号", "指标相关度", "指标重要性排名（值越小排名越高）"]
            columns_json = {str(index): value for index, value in enumerate(header)}
            option = self.echarts(matrix)
            # 返回成功消息
            return {
                'status': 'success',
                'message': '分类完毕',
                'data': data,
                'header': columns_json,
                'option': option,
            }
        else:
            # 返回成功消息
            return {
                'status': 'success',
                'message': '上传完成',
                'row_count': self.row_count,
                'col_count': self.col_count,
                'file': file.filename,
                'detail_data': detail_data,
                'detail_header': detail_columns_json
            }
reduction_view = ReductionApi.as_view('reduction_api')
import os
import pandas as pd
from flask import request
from flask.views import MethodView
class FileApi(MethodView):
    def sum(self, file):
        try:
            # 使用 Pandas 读取 excel 文件
            df = pd.read_excel(file)
            # 提取第一列数据并求和
            result = df.iloc[:, 1 - 1].sum()
            return result
        except Exception as e:
            print("An error occurred:", str(e))
            return None
```

```
    def post(self):
        # 检查是否有文件上传
        if 'file' not in request.files:
            return {
                'status': 'error',
                'message': 'No file provided',
                'code': 1
            }
        # 获取上传的文件
        file = request.files['file']
        # 检查文件是否存在
        if file.filename == '':
            return {
                'status': 'error',
                'message': 'No file selected',
                'code': 2
            }
        total_sum = self.sum(file)
        # 指定相对路径保存文件，相对于当前工作目录
        upload_folder = 'file'
        # 获取当前工作目录
        current_directory = os.getcwd()
        print(upload_folder, "+", current_directory, "+", total_sum)
        # 构建保存文件的完整路径
        save_path = os.path.join(current_directory, upload_folder, file.filename)
        # 保存文件
        file.save(save_path)
        total_sum = int(total_sum)
        # 返回保存成功的消息
        return {
            'status': 'success',
            'message': 'Sum calculated successfully',
            'sum': total_sum
        }
file_view = FileApi.as_view('file_api')
import os
from flask import send_file
from flask.views import MethodView
class ImageApi(MethodView):
    def get(self, filename):
        # 获取图片所在的目录
        current_dir = os.path.dirname(os.path.dirname(os.path.dirname(os.path.dirname(os.path.realpath(__file__)))))
        # 指定资源目录的相对路径
        images_dir = os.path.join(current_dir, 'image')
        # 使用 send_file 函数发送资源
        return send_file(os.path.join(images_dir, filename))
image_view = ImageApi.as_view('image_api')
function createUserList() {
  return [
    {
      userId: 1,
      avatar:
        'https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif',
      username: 'admin',
      password: '111111',
      desc: '平台管理员',
```

```
        roles: ['平台管理员'],
        buttons: ['cuser.detail'],
        routes: ['home'],
        token: 'Admin Token',
    },
    {
        userId: 2,
        avatar:
            'https://wpimg.wallstcn.com/f778738c-e4f8-4870-b634-56703b4acafe.gif',
        username: 'system',
        password: '111111',
        desc: '系统管理员',
        roles: ['系统管理员'],
        buttons: ['cuser.detail', 'cuser.user'],
        routes: ['home'],
        token: 'System Token',
    },
 ]
}
export default [
 {
    url: '/api/user/login',//请求地址
    method: 'post',//请求方式
    response: ({ body }) => {
        const { username, password } = body;
        const checkUser = createUserList().find(
            (item) => item.username === username && item.password === password,
        )
        if (!checkUser) {
            return { code: 201, data: { message: '账号或者密码不正确' } }
        }
        const { token } = checkUser
        return { code: 200, data: { token } }
    },
 },
 {
    url: '/api/user/info',
    method: 'get',
    response: (request) => {
        const token = request.headers.token;
        const checkUser = createUserList().find((item) => item.token === token)
        if (!checkUser) {
            return { code: 201, data: { message: '获取用户信息失败' } }
        }
        return { code: 200, data: {checkUser} }
    },
 },
]
<template>
  <div>
    <router-view></router-view>
  </div>
</template>
<script setup lang="ts">
</script>
<style scoped>
</style>
```

```
import request from '@/utils/request'
import type { loginForm, loginResponseData } from './type';
enum API{
  LOGIN_URL='/user/login',
  USERINFO_URL='/user/info'
}
export const reqLogin=(data: loginForm)=>request.post<any,loginResponseData>(API.LOGIN_URL,data);
export const reqUserInfo=()=>request.get<any>(API.USERINFO_URL);
export interface loginForm {
  username: string;
  password: string;
}
interface dataType {
  token: string;
}
export interface loginResponseData {
  code: number;
  data: dataType;
}
<template>
  <svg :style="{ width, height }">
    <use :xlink:href="prefix + name" :fill="color"></use>
  </svg>
</template>
<script setup lang="ts">
defineProps({
  prefix:{
    type:String,
    default:"#icon-"
  },
  name:String,
  color:{
    type:String,
    default:''
  },
  width:{
    type:String,
    default:'16px'
  },
  height:{
    type:String,
    default:'16px'
  }
})
</script>
<style scoped></style>
import SvgIcon from "./SvgIcon/index.vue";
import * as ElementPlusIconsVue from '@element-plus/icons-vue'
const allGlobalComponent:any = { SvgIcon };
export default {
  install(app:any) {
    Object.keys(allGlobalComponent).forEach(key => {
      app.component(key, allGlobalComponent[key]);
    });
    for (const[key, component] of Object.entries(ElementPlusIconsVue)){
      app.component(key, component)
    }
  },
```

```
};
<template>
  <div class="layout_container">
    <!--左侧菜单-->
    <div class="layout_slider">
      <div class="logo" alt="">
        <img src="../../../icon.svg" />
        <p style="font-size: 20px">医疗辅助诊断平台</p>
      </div>
      <!--滚动条展示菜单-->
      <el-scrollbar class="scrollbar">
        <el-menu background-color="#343a3f" text-color="#bbc1c6" :default-active="$route.path">
          <!--根据路由动态生成菜单-->
          <Menu :menuList="userStore.menuRoutes"></Menu>
        </el-menu>
      </el-scrollbar>
    </div>
    <!--顶部导航-->
    <div class="layout_tabbar">
      <Tabbar></Tabbar>
    </div>
    <!--内容展示区-->
    <div class="layout_main">
      <Main></Main>
    </div>
  </div>
</template>
<script setup lang="ts">
import { useRoute } from 'vue-router';
import Menu from './menu/index.vue'
import Main from './main/index.vue'
import Tabbar from './tabbar/index.vue'
import useUserStore from '@/store/modules/user'
let userStore=useUserStore()
let $route=useRoute()
</script>
<script lang="ts">
export default {
  name:"Layout"
}
</script>
<style scoped lang="scss">
.layout_container {
  width: 100%;
  height: 100vh;
  background: white;
  .layout_slider {
    width: $base-menu-width;
    height: 100vh;
    background: $base-menu-background;
    box-shadow: 0px 0px 10px $base-menu-background;
    z-index: 1;
    position: relative;
    .logo {
      width: 100%;
      height: 50px;
      padding: 0px 5px;
```

```
      margin: 0px 0px 10px 0px;
      color: #bbc1c6;
      display: flex;
      align-items: center;
      img {
        width: 40px;
        height: auto;
        margin: 5px 15px;
      }
    }
    .scrollbar {
      width: 100%;
      height: calc(100vh - 60px);
      color: #bbc1c6;
      .el-menu{
        border-right: none;
      }
    }
  }
  .layout_tabbar {
    width: calc(100% - $base-menu-width);
    height: $base-tabbar-height;
    position: fixed;
    top: 0px;
    left: $base-menu-width;
    background: white;
    color: black;
  }
  .layout_main {
    position: absolute;
    width: calc(100% - $base-menu-width);
    height: calc(100vh - $base-tabbar-height);
    background: white;
    top: $base-tabbar-height ;
    left: calc($base-menu-width);
    padding: 20px;
    overflow: auto;
    background-color: #f5f7fb;
  }
}
</style>
<template>
  <div>
    <router-view v-slot="{Component}">
      <transition name="fade">
        <!--渲染 layout 一级路由组件-->
        <component :is="Component" v-if="flag"/>
      </transition>
    </router-view>
  </div>
</template>
<script setup lang="ts">
import {watch, ref, nextTick} from 'vue'
import useLayoutSettingStore from '@/store/modules/setting';
let layoutSettingStore=useLayoutSettingStore()
let flag=ref(true)
watch(()=>layoutSettingStore.refsh,()=>{
  flag.value=false;
```

```
  nextTick(()=>{
    flag.value=true
  })
})
</script>
<script lang="ts">
export default {
  name:"Main"
}
</script>
<style scoped>
/* .fade-enter-from{
  opacity:0
}
.fade-enter-active{
  transition: all 1s
}
.fade-enter-to{
  opacity:1
} */
</style>
<template>
  <!--没有子路由-->
  <template v-for="item in menuList" :key="item.path">
    <template v-if="!item.children">
      <el-menu-item
        v-if="!item.meta.hidden"
        :index="item.path"
        @click="goRoute"
      >
        <el-icon class="icon_class">
          <component :is="item.meta.icon"></component>
        </el-icon>
        <template #title>
          <span class="title_class">{{ item.meta.title }}</span>
        </template>
      </el-menu-item>
    </template>
    <!--有一个子路由-->
    <template v-if="item.children && item.children.length == 1">
      <el-menu-item
        v-if="!item.children[0].meta.hidden"
        :index="item.children[0].path"
        @click="goRoute"
      >
        <el-icon class="icon_class">
          <component :is="item.children[0].meta.icon"></component>
        </el-icon>
        <template #title>
          <span class="title_class">{{ item.children[0].meta.title }}</span>
        </template>
      </el-menu-item>
    </template>
    <!--有多个子路由-->
    <el-sub-menu
      v-if="item.children && item.children.length > 1"
      :index="item.path"
    >
```

```
      <template #title>
        <el-icon class="icon_class">
          <component :is="item.meta.icon"></component>
        </el-icon>
        <span class="title_class">{{ item.meta.title }}</span>
      </template>
      <Menu :menuList="item.children"></Menu>
    </el-sub-menu>
  </template>
</template>
<script setup lang="ts">
import { useRouter } from "vue-router";
defineProps(["menuList"]);
let $router = useRouter();
const goRoute = (vc: any) => {
  $router.push(vc.index);
};
</script>
<script lang="ts">
export default {
  name: "Menu",
};
</script>
<style scoped>
.title_class {
  padding: 0 0 0 15px;
  font-size: 16px;
}
.icon_class {
  padding: 0 0 0 5px;
}
</style>
<template>
  <div class="tabbar">
    <!--面包屑-->
    <div class="tabbar_left">
      <el-icon style="margin-right: 10px" @click="changeIcon">
        <component :is="layoutSettingStore.fold?'Fold':'Expand'"></component>
      </el-icon>
      <!--左侧面包屑-->
      <el-breadcrumb separator-icon="ArrowRight">
        <!--面包屑动态展示路由-->
        <el-breadcrumb-item v-for="(item, index) in $route.matched" :key="index" v-
show="item.meta.title" :to="item.path">
          <el-icon style="margin: 0px 3px">
            <component :is="item.meta.icon"></component>
          </el-icon>
          <span>{{ item.meta.title }}</span>
        </el-breadcrumb-item>
      </el-breadcrumb>
    </div>
    <!--右侧设置-->
    <div class="tabbar_right">
      <el-button size="default" icon="Refresh" circle @click="updateRefsh"></el-button>
      <el-button
        size="default"
        icon="FullScreen"
```

```
      circle
    ></el-button>
    <el-button size="default" icon="Setting" circle></el-button>
    <img src="@/assets/icons/user.svg" style="width: 30px; height: 30px; margin: 0px 10px 0px 30px" />
    <el-dropdown>
     <span class="el-dropdown-link">
      admin
      <el-icon class="el-icon--right">
       <arrow-down />
      </el-icon>
     </span>
     <template #dropdown>
      <el-dropdown-menu>
       <el-dropdown-item @click="logout">退出登录</el-dropdown-item>
      </el-dropdown-menu>
     </template>
    </el-dropdown>
   </div>
  </div>
</template>
<script setup lang="ts">
import useLayoutSettingStore from '@/store/modules/setting';
import { useRouter } from "vue-router";
let $router=useRouter()
let layoutSettingStore=useLayoutSettingStore();
const changeIcon=()=>{
  layoutSettingStore.fold=!layoutSettingStore.fold
}
const updateRefsh=()=>{
  layoutSettingStore.refsh=!layoutSettingStore.refsh
}
const logout=()=>{
  $router.push('/login')
}
</script>
<script lang="ts">
export default {
  name:"Tabbar",
}
</script>
<style scoped lang="scss">
.tabbar {
  width: 100%;
  height: 100%;
  display: flex;
  justify-content: space-between;
  .tabbar_left {
    display: flex;
    align-items: center;
    margin-left: 20px;
  }
  .tabbar_right {
    display: flex;
    align-items: center;
    margin-right: 20px;
  }
}
</style>
```

```
import { createApp } from "vue";
import App from "@/App.vue";
import ElementPlus from "element-plus";
import "element-plus/dist/index.css";
import zhCn from "element-plus/dist/locale/zh-cn.mjs";
import "virtual:svg-icons-register";
import globalComponent from "@/components";
import "@/styles/index.scss";
import router from "./router";
import pinia from "./store";
import * as echarts from 'echarts';
const app = createApp(App);
app.use(ElementPlus, {
  locale: zhCn,
});
app.use(globalComponent);
app.use(router);
app.use(pinia);
app.config.globalProperties.$echarts = echarts
app.mount("#app");
import {createRouter, createWebHashHistory} from 'vue-router'
import { constantRoute } from './routes';
let router = createRouter({
  history: createWebHashHistory(),
  routes: constantRoute,
  scrollBehavior(){
    return{
      left:0,
      top:0
    }
  }
});
export default router;
export const constantRoute = [
  {
    path: "/login",
    component: () => import("@/views/login/index.vue"),
    name: "login",
    meta: {
      title: "登录",
      hidden: true,
    },
  },
  {
    path: "/",
    component: () => import("@/layout/index.vue"),
    name: "layout",
    meta: {
      title: "",
      hidden: false,
      icon: "",
    },
    redirect: "/home",
    children: [
      {
        path: "/home",
        component: () => import("@/views/home/index.vue"),
        meta: {
```

```
        title: "首页",
        hidden: false,
        icon: "HomeFilled",
      },
    },
  ],
},
{
  path: "/screen",
  component: () => import("@/views/screen/index.vue"),
  name: "Screen",
  meta: {
    title: "数据大屏",
    hidden: false,
    icon: "Platform",
  },
},
{
  path: "/analyse",
  component: () => import("@/layout/index.vue"),
  name: "analyse",
  meta: {
    title: "数据分析",
    hidden: false,
    icon: "DataLine",
  },
  redirect:'/analyse/reduce',
  children: [
    {
      path: "/analyse/reduce",
      component: () => import("@/views/analyse/reduce/index.vue"),
      meta: {
        title: "特征筛选",
        hidden: false,
        icon: "Filter",
      },
    },
    {
      path: "/analyse/detection",
      component: () => import("@/views/analyse/detection/index.vue"),
      meta: {
        title: "异常检测",
        hidden: false,
        icon: "Aim",
      },
    },
    {
      path: "/analyse/classification",
      component: () => import("@/views/analyse/classification/index.vue"),
      meta: {
        title: "辅助识别",
        hidden: false,
        icon: "View",
      },
    },
  ],
},
```

```
  {
    path: "/404",
    component: () => import("@/views/404/index.vue"),
    name: "404",
    meta: {
      title: "404",
      hidden: true,
    },
  },
  {
    path: "/:pathMatch(.*)*",
    redirect: "/404",
    name: "Any",
    meta: {
      title: "Any",
      hidden: true,
    },
  },
];
import { createPinia } from "pinia";
let pinia = createPinia();
export default pinia;
import {defineStore} from "pinia";
let useLayoutSettingStore=defineStore('SettingStore',{
  state:()=>{
    return{
      fold:false,
      refsh:false,
    }
  }
})
export default useLayoutSettingStore
import { defineStore } from "pinia";
import { reqLogin,reqUserInfo } from "@/api/user";
import type { loginForm } from "@/api/user/type";
import { constantRoute } from "@/router/routes";
let useUserStore = defineStore("User", {
  state: () => {
    return {
      token: localStorage.getItem("TOKEN"),
      menuRoutes:constantRoute
    };
  },
  actions: {
    async userLogin(data: loginForm) {
      console.log(data);
      let result: any = await reqLogin(data);
      if (result.code == 200) {
        this.token = result.data.token;
        localStorage.setItem("TOKEN", result.data.token);
        return "ok";
      } else {
        return Promise.reject(new Error(result.data.message));
      }
    },
  },
  getters: {},
});
```

```
export default useUserStore;
import axios from "axios";
import { ElMessage } from "element-plus";
let request = axios.create({
  baseURL: import.meta.env.VITE_APP_BASE_API, //基础路径上会携带/api
  timeout: 5000,
});
request.interceptors.request.use((config) => {
  return config;
});
request.interceptors.response.use(
  (response) => {
    return response.data;
  },
  (error) => {
    let message = "";
    let status = error.response.status;
    switch (status) {
      case 401:
        message = "TOKEN 过期";
        break;
      case 403:
        message = "无权访问";
        break;
      case 404:
        message = "请求地址错误";
        break;
      case 500:
        message = "服务器出现问题";
        break;
      default:
        message = "网络出现其他问题";
        break;
    }
    ElMessage({
      type: "error",
      message,
    });
    return Promise.reject(error);
  }
);
export default request;
export const getTime = () => {
  let message = "";
  let hours = new Date().getHours();
  if (hours >= 6 && hours <= 9) {
    message = "早上好";
  } else if (hours <= 12) {
    message = "上午好";
  } else if (hours <= 18) {
    message = "下午好";
  } else if (hours <= 24) {
    message = "晚上好";
  } else {
    message = "夜深了";
  }
  return message;
```

```
};
<template>
  <div>
    <h1>我是一级路由 404</h1>
  </div>
</template>
<script setup lang="ts">
</script>
<style scoped>
</style>
<template>
  <div>
    <div>
      <el-upload
        ref="upload"
        class="upload-demo"
        drag
        action="https://run.mocky.io/v3/9d059bf9-4660-45f2-925d-ce80ad6c4d15"
        :limit="1"
        :on-exceed="handleExceed"
        :auto-upload="false"
        :http-request="classification"
      >
        <el-icon class="el-icon--upload"><upload-filled /></el-icon>
        <div class="el-upload__text">
          Drop file here or <em>click to upload</em>
        </div>
        <div class="el-upload__tip">
          limit 1 .csv file, new file will cover the old file
        </div>
      </el-upload>
    </div>
    <div>
      <el-button class="ml-3" type="primary" plain @click="submitDetail">
        点击上传
      </el-button>
      <el-button class="ml-3" style="margin-left: 10px" plain @click="detail">
        文件详情
      </el-button>
      <el-button
        class="ml-3"
        type="success"
        plain
        @click="dialogFormVisible = true"
      >
        立即检测
      </el-button>
      <el-button
        class="ml-3"
        style="margin-left: 10px"
        plain
        @click="dialogImageVisible = true"
      >
        可视化结果
      </el-button>
    </div>
    <el-alert
```

```
    style="background-color: #f5f7fb; margin: 5px 0; padding: 0"
    title="上传后可查看文件详情"
    type="info"
    :closable="false"
    show-icon
  />
  <div style="margin-top: 10px">
    <el-table
      :default-sort="{ prop: 'date', order: 'descending' }"
      v-loading="loading"
      :data="tableData"
      height="400"
      border
      style="width: 100%"
    >
      <!-- <el-table-column v-if="form.anomalyNum !== ""type="index" :index="indexMethod" /> -->
      <el-table-column
        v-for="(value, key) in header"
        :key="key"
        :prop="value"
        :label="value"
        sortable
      />
    </el-table>
  </div>
  <el-dialog
    v-model="dialogFormVisible"
    title="参数设置"
    width="500"
    text-align:
    center
  >
    <el-form :model="form">
      <el-form-item label="算法选择" text-align: left>
        <el-select
          v-model="form.algorithm"
          placeholder="请选择聚类算法(默认 Kmeans 算法)"
        >
          <!-- <el-option label="Mean Shift" value="ms" /> -->
          <el-option label="KMEANS" value="kmeans" />
          <!-- <el-option label="GMM" value="gmm" /> -->
        </el-select>
      </el-form-item>
    </el-form>
    <template #footer>
      <div class="dialog-footer">
        <el-button @click="dialogFormVisible = false">取消</el-button>
        <el-button type="primary" @click="submitclassification"> 确认 </el-button>
      </div>
    </template>
  </el-dialog>
  <el-dialog v-model="dialogTableVisible" title="文件详情" width="800">
    <el-table :data="detailData">
      <el-table-column property="name" label="文件名" width="350" />
      <el-table-column property="sampleSize" label="样本个数" />
      <el-table-column property="attributeNum" label="指标个数" />
    </el-table>
```

```
    <el-table
      :data="detailTableData"
      height="300"
      stripe
      border
      style="width: 100%; margin-top: 20px"
    >
      <el-table-column
        type="index"
        :index="indexMethod"
        :label="'id'"
        width="60"
      />
      <el-table-column
        v-for="(value, key) in detailHeader"
        :key="key"
        :prop="value"
        :label="value"
      />
    </el-table>
  </el-dialog>
  <el-dialog v-model="dialogImageVisible" title="可视化结果" width="600">
    <div class="demo-image">
      <div>
        <v-chart class="chart" :option="option" autoresize />
      </div>
    </div>
  </el-dialog>
 </div>
</template>
<script setup lang="ts">
import { ref, reactive, Ref } from "vue";
import axios, { AxiosResponse } from "axios";
import { ElMessage } from "element-plus";
import { ElUpload } from "element-plus";
import { genFileId } from "element-plus";
import { UploadFilled } from "@element-plus/icons-vue";
import type { UploadInstance, UploadProps, UploadRawFile } from "element-plus";
import { use } from "echarts/core";
import { BarChart } from 'echarts/charts'
import { GridComponent } from 'echarts/components'
import { CanvasRenderer } from 'echarts/renderers'
import VChart from "vue-echarts";
use([GridComponent, BarChart, CanvasRenderer])
const loading = ref(false);
const detailLock = ref(false);
const classificationLock = ref(false);
const indexMethod = (index: number) => {
 return index + 1;
};
interface FormDataFile {
 file: File;
}
interface DetailData {
 name: string;
 attributeNum: string;
 sampleSize: string;
}
```

```
interface RowData {
  [key: string]: string;
}
const dialogTableVisible = ref(false) as Ref<boolean>;
const dialogFormVisible = ref(false) as Ref<boolean>;
const dialogImageVisible = ref(false) as Ref<boolean>;
const form = reactive({
  algorithm: "请选择分类算法(默认 Kmeans 算法)",
  isFeatures: false,
  k: 3,
});
const detailData: DetailData[] = reactive([
  {
    name: "",
    sampleSize: "",
    attributeNum: "",
  },
]);
const matrixData: any[] = [];
const tableData = ref<RowData[]>([]);
const detailMatrixData: any[] = [];
const detailTableData = ref<RowData[]>([]);
const header = ref({}) as Ref<Record<string, string>>;
const detailHeader = ref({}) as Ref<Record<string, string>>;
const upload = ref<UploadInstance>();
const option = ref({});
const classification = (file: FormDataFile): any => {
  if (classificationLock.value) {
    let formData = new FormData();
    const isclassification = true;
    formData.append("file", file.file);
    formData.append("is_classification", isclassification.toString()); // 添加额外的参数
    formData.append("k", form.k.toString());
    formData.append("algorithm", form.algorithm);
    dialogFormVisible.value = false;
    loading.value = true;
    postclassification(formData).then((res: AxiosResponse<any>) => {
      if (res.data.status == "success") {
        console.log(res.data);
        matrixData.splice(0);
        matrixData.push(...JSON.parse(res.data.data));
        tableData.value.splice(0);
        matrixData.forEach((item: any) => {
          let rowData: RowData = {};
          Object.keys(res.data.header).forEach((index: string) => {
            const key = res.data.header[index];
            let value = item[index];
            rowData[key] = value;
          });
          tableData.value.push(rowData);
        });
        loading.value = false;
        header.value = res.data.header;
        classificationLock.value = false;
        option.value = res.data.option;
      }
    });
  }
```

```
  if (detailLock.value) {
    let formData = new FormData();
    formData.append("file", file.file);
    postclassification(formData).then((res: AxiosResponse<any>) => {
      if (res.data.status == "success") {
        detailMatrixData.splice(0);
        detailMatrixData.push(...JSON.parse(res.data.detail_data));
        detailTableData.value.splice(0);
        detailMatrixData.forEach((item: any) => {
          let rowData: RowData = {};
          Object.keys(res.data.detail_header).forEach((index: string) => {
            const key = res.data.detail_header[index];
            let value = item[index];
            rowData[key] = value;
          });
          detailTableData.value.push(rowData);
        });
        ElMessage({
          message: "文件已成功上传",
          type: "success",
        });
        detailData[0].name = res.data.file;
        detailData[0].attributeNum = res.data.col_count;
        detailData[0].sampleSize = res.data.row_count;
        detailHeader.value = res.data.detail_header;
        tableData.value.splice(0);
      }
    });
    detailLock.value = false;
  }
};
const postclassification = (file: FormData) => {
  return axios({
    url: "/api/classification",
    method: "post",
    data: file,
    headers: {
      "Content-Type": "multipart/form-data",
    },
  });
};
const detail = (file: File | null) => {
  if (file) {
    dialogTableVisible.value = true;
  } else {
    ElMessage.error("未上传文件");
  }
};
const handleExceed: UploadProps["onExceed"] = (files) => {
  upload.value!.clearFiles();
  const file = files[0] as UploadRawFile;
  file.uid = genFileId();
  upload.value!.handleStart(file);
};
const submitDetail = () => {
  upload.value!.submit();
  detailLock.value = true;
};
```

```
const submitclassification = () => {
  upload.value!.submit();
  classificationLock.value = true;
};
</script>
<style>
.el-table .error-row {
  --el-table-tr-bg-color: var(--el-color-error-light-9);
}
.el-table .warning-row {
  --el-table-tr-bg-color: var(--el-color-warning-light-9);
}
.el-table .success-row {
  --el-table-tr-bg-color: var(--el-color-success-light-9);
}
.div_scatter {
  height: 50vh;
  width: 50vw;
}
.chart {
  height: 65vh;
}
</style>
<template>
  <div>
    <div>
      <el-upload
        ref="upload"
        class="upload-demo"
        drag
        action="https://run.mocky.io/v3/9d059bf9-4660-45f2-925d-ce80ad6c4d15"
        :limit="1"
        :on-exceed="handleExceed"
        :auto-upload="false"
        :http-request="detection"
      >
        <el-icon class="el-icon--upload"><upload-filled /></el-icon>
        <div class="el-upload__text">
          Drop file here or <em>click to upload</em>
        </div>
        <div class="el-upload__tip">
          limit 1 .csv file, new file will cover the old file
        </div>
      </el-upload>
    </div>
    <div>
      <el-button class="ml-3" type="primary" plain @click="submitDetail">
        点击上传
      </el-button>
      <el-button class="ml-3" style="margin-left: 10px" plain @click="detail">
        文件详情
      </el-button>
      <el-button
        class="ml-3"
        type="success"
        plain
        @click="dialogFormVisible = true"
      >
```

```
    立即检测
  </el-button>
  <el-button
    class="ml-3"
    style="margin-left: 10px"
    plain
    @click="dialogImageVisible = true"
  >
    可视化结果
  </el-button>
</div>
<el-alert
  style="background-color: #f5f7fb; margin: 5px 0; padding: 0"
  title="上传后可查看文件详情"
  type="info"
  :closable="false"
  show-icon
/>
<div style="margin-top: 10px">
  <el-table
    :default-sort="{ prop: 'date', order: 'descending' }"
    v-loading="loading"
    :row-class-name="tableRowClassName"
    :data="tableData"
    height="400"
    border
    style="width: 100%"
  >
    <!-- <el-table-column v-if="form.anomalyNum !== ""type="index" :index="indexMethod" /> -->
    <el-table-column
      v-for="(value, key) in header"
      :key="key"
      :prop="value"
      :label="value"
      sortable
    />
  </el-table>
</div>
<el-dialog
  v-model="dialogFormVisible"
  title="参数设置"
  width="500"
  text-align:
  center
>
  <el-form :model="form">
    <el-form-item label="算法选择" text-align: left>
      <el-select
        v-model="form.algorithm"
        placeholder="请选择异常检测算法(默认 KFRAD 算法)"
      >
        <el-option label="KNN(基于距离)" value="knn" />
        <el-option label="LOF(基于密度)" value="lof" />
        <el-option label="HBOS(基于密度)" value="hbos" />
        <el-option label="KFRAD(基于核模糊粗糙集)" value="kfrad" />
      </el-select>
    </el-form-item>
```

```
      <!-- <el-form-item label="特征筛选" text-align: left>
        <el-switch v-model="form.isFeatures" />
      </el-form-item> -->
      <el-form-item label="异常阈值" text-align: left>
        <el-input
          v-model="form.anomalyNum"
          placeholder="请输入异常阈值"
          clearable
        />
      </el-form-item>
      <el-form-item label="KFRAD 核参数选择" text-align: left  v-if="form.algorithm === 'kfrad'">
        <el-input
          v-model="form.delta"
          placeholder="请输入核参数"
          clearable
        />
      </el-form-item>
      <el-alert
        style="background-color: #ffffff; margin: 5px 0; padding: 0"
        title="KFRAD 异常阈值(核参数)参考：心脏病(Heart)-0.35(0.04)；乳腺癌(Wbc)-0.35(0.48)；糖尿病
(Diab)-0.6(0.02)；肝炎(Hepa)-0.27(0.06)；淋巴造影(Lym)-0.25(0.02)."
        type="info"
        :closable="false"
        show-icon
      />
    </el-form>
    <template #footer>
      <div class="dialog-footer">
        <el-button @click="dialogFormVisible = false">取消</el-button>
        <el-button type="primary" @click="submitDetection"> 确认 </el-button>
      </div>
    </template>
  </el-dialog>
  <el-dialog v-model="dialogTableVisible" title="文件详情" width="800">
    <el-table :data="detailData">
      <el-table-column property="name" label="文件名" width="350" />
      <el-table-column property="sampleSize" label="样本个数" />
      <el-table-column property="attributeNum" label="指标个数" />
    </el-table>
    <el-table
      :data="detailTableData"
      height="300"
      stripe
      border
      style="width: 100%; margin-top: 20px"
    >
      <el-table-column
        type="index"
        :index="indexMethod"
        :label="'id'"
        width="60"
      />
      <el-table-column
        v-for="(value, key) in detailHeader"
        :key="key"
        :prop="value"
```

```
        :label="value"
      />
    </el-table>
  </el-dialog>
  <el-dialog v-model="dialogImageVisible" title="可视化结果" width="600">
    <div class="demo-image">
      <!-- <el-image style="width: 600px; height: 600px" :src="url" :fit="fit" /> -->
      <div>
        <v-chart class="chart" :option="option" autoresize />
      </div>
    </div>
  </el-dialog>
 </div>
</template>
<script setup lang="ts">
import { ref, reactive, Ref } from "vue";
import axios, { AxiosResponse } from "axios";
import { ElMessage } from "element-plus";
import { ElUpload } from "element-plus";
import { genFileId } from "element-plus";
import { UploadFilled } from "@element-plus/icons-vue";
import type { UploadInstance, UploadProps, UploadRawFile } from "element-plus";
import { use } from "echarts/core";
import { ScatterChart } from "echarts/charts";
import { GridComponent, MarkLineComponent } from "echarts/components";
import { CanvasRenderer } from "echarts/renderers";
import VChart from "vue-echarts";
use([GridComponent, MarkLineComponent, ScatterChart, CanvasRenderer]);
const loading = ref(false);
const detailLock = ref(false);
const detectionLock = ref(false);
const indexMethod = (index: number) => {
 return index + 1;
};
interface FormDataFile {
 file: File;
}
interface DetailData {
 name: string;
 attributeNum: string;
 sampleSize: string;
}
interface RowData {
 [key: string]: string;
}
const dialogTableVisible = ref(false) as Ref<boolean>;
const dialogFormVisible = ref(false) as Ref<boolean>;
const dialogImageVisible = ref(false) as Ref<boolean>;
const form = reactive({
 algorithm: "请选择异常检测算法(默认 KFRAD 算法)",
 isFeatures: false,
 anomalyNum: 0.6,
 delta: 0.02,
});
const detailData: DetailData[] = reactive([
 {
  name: "",
  sampleSize: "",
```

```
    attributeNum: "",
  },
]);
const matrixData: any[] = [];
const tableData = ref<RowData[]>([]);
const detailMatrixData: any[] = [];
const detailTableData = ref<RowData[]>([]);
const header = ref({}) as Ref<Record<string, string>>;
const detailHeader = ref({}) as Ref<Record<string, string>>;
const tableRowClassName = ({ row }: { row: RowData; rowIndex: number }) => {
  if (Number(row.异常分数) >= form.anomalyNum) {
    return "error-row";
  }
};
const upload = ref<UploadInstance>();
const option = ref({});
const detection = (file: FormDataFile): any => {
  if (detectionLock.value) {
    let formData = new FormData();
    const n = form.anomalyNum;
    const isDetection = true;
    formData.append("file", file.file);
    formData.append("n", n.toString()); // 添加额外的参数 n
    formData.append("is_detection", isDetection.toString()); // 添加额外的参数
    formData.append("delta", form.delta.toString());
    formData.append("algorithm", form.algorithm);
    dialogFormVisible.value = false;
    loading.value = true;
    postDetection(formData).then((res: AxiosResponse<any>) => {
      if (res.data.status == "success") {
        console.log(res.data);
        matrixData.splice(0);
        matrixData.push(...JSON.parse(res.data.data));
        tableData.value.splice(0);
        matrixData.forEach((item: any) => {
          let rowData: RowData = {};
          Object.keys(res.data.header).forEach((index: string) => {
            const key = res.data.header[index];
            let value = item[index];
            rowData[key] = value;
          });
          tableData.value.push(rowData);
        });
        console.log(tableData.value);
        loading.value = false;
        header.value = res.data.header;
        detectionLock.value = false;
        option.value = res.data.option;
        console.log(option);
      }
    });
  }
  if (detailLock.value) {
    let formData = new FormData();
    formData.append("file", file.file);
    postDetection(formData).then((res: AxiosResponse<any>) => {
      if (res.data.status == "success") {
        detailMatrixData.splice(0);
```

```
        detailMatrixData.push(...JSON.parse(res.data.detail_data));
        detailTableData.value.splice(0);
        detailMatrixData.forEach((item: any) => {
          let rowData: RowData = {};
          Object.keys(res.data.detail_header).forEach((index: string) => {
            const key = res.data.detail_header[index];
            let value = item[index];
            rowData[key] = value;
          });
          detailTableData.value.push(rowData);
        });
        ElMessage({
          message: "文件已成功上传",
          type: "success",
        });
        detailData[0].name = res.data.file;
        detailData[0].attributeNum = res.data.col_count;
        detailData[0].sampleSize = res.data.row_count;
        detailHeader.value = res.data.detail_header;
        tableData.value.splice(0);
      }
    });
    detailLock.value = false;
  }
};
const postDetection = (file: FormData) => {
  return axios({
    url: "/api/detection",
    method: "post",
    data: file,
    headers: {
      "Content-Type": "multipart/form-data",
    },
  });
};
const detail = (file: File | null) => {
  if (file) {
    dialogTableVisible.value = true;
  } else {
    ElMessage.error("未上传文件");
  }
};
const handleExceed: UploadProps["onExceed"] = (files) => {
  upload.value!.clearFiles();
  const file = files[0] as UploadRawFile;
  file.uid = genFileId();
  upload.value!.handleStart(file);
};
const submitDetail = () => {
  upload.value!.submit();
  detailLock.value = true;
};
const submitDetection = () => {
  upload.value!.submit();
  detectionLock.value = true;
};
</script>
<style>
```

```
.el-table .error-row {
  --el-table-tr-bg-color: var(--el-color-error-light-9);
}
.el-table .success-row {
  --el-table-tr-bg-color: var(--el-color-success-light-9);
}
.div_scatter {
  height: 50vh;
  width: 50vw;
}
.chart {
  height: 65vh;
}
</style>
<template>
  <div>
    <div>
      <el-upload
        ref="upload"
        class="upload-demo"
        drag
        action="https://run.mocky.io/v3/9d059bf9-4660-45f2-925d-ce80ad6c4d15"
        :limit="1"
        :on-exceed="handleExceed"
        :auto-upload="false"
        :http-request="reduction"
      >
        <el-icon class="el-icon--upload"><upload-filled /></el-icon>
        <div class="el-upload__text">
          Drop file here or <em>click to upload</em>
        </div>
        <div class="el-upload__tip">
          limit 1 .csv file, new file will cover the old file
        </div>
      </el-upload>
    </div>
    <div>
      <el-button class="ml-3" type="primary" plain @click="submitDetail">
        点击上传
      </el-button>
      <el-button class="ml-3" style="margin-left: 10px" plain @click="detail">
        文件详情
      </el-button>
      <el-button
        class="ml-3"
        type="success"
        plain
        @click="dialogFormVisible = true"
      >
        立即筛选
      </el-button>
      <el-button
        class="ml-3"
        style="margin-left: 10px"
        plain
        @click="dialogImageVisible = true"
      >
```

```
    可视化结果
  </el-button>
 </div>
 <el-alert
  style="background-color: #f5f7fb; margin: 5px 0; padding: 0"
  title="上传后可查看文件详情"
  type="info"
  :closable="false"
  show-icon
 />
 <div style="margin-top: 10px">
  <el-table
   :default-sort="{ prop: 'date', order: 'descending' }"
   v-loading="loading"
   :data="tableData"
   height="400"
   border
   style="width: 100%"
  >
   <!-- <el-table-column v-if="form.anomalyNum !== ""type="index" :index="indexMethod" /> -->
   <el-table-column
    v-for="(value, key) in header"
    :key="key"
    :prop="value"
    :label="value"
    sortable
   />
  </el-table>
 </div>
 <el-dialog
  v-model="dialogFormVisible"
  title="参数设置"
  width="500"
  text-align:
  center
 >
  <el-form :model="form">
   <el-form-item label="算法选择" text-align: left>
    <el-select
     v-model="form.algorithm"
     placeholder="请选择特征筛选算法(默认 MNIFS 算法)"
    >
     <el-option label="MNIFS(基于模糊邻域的属性约简)" value="mnifs" />
     <!-- <el-option label="KMEANS" value="kmeans" />
     <el-option label="GMM" value="gmm" /> -->
    </el-select>
   </el-form-item>
   <el-form-item label="MNIFS 参数选择" text-align: left  v-if="form.algorithm === 'mnifs'">
    <el-input
     v-model="form.lammda"
     placeholder="请输入 MNIFS 参数"
     clearable
    />
   </el-form-item>
  </el-form>
  <template #footer>
   <div class="dialog-footer">
```

```
        <el-button @click="dialogFormVisible = false">取消</el-button>
        <el-button type="primary" @click="submitreduction"> 确认 </el-button>
      </div>
    </template>
  </el-dialog>
  <el-dialog v-model="dialogTableVisible" title="文件详情" width="800">
    <el-table :data="detailData">
      <el-table-column property="name" label="文件名" width="350" />
      <el-table-column property="sampleSize" label="样本个数" />
      <el-table-column property="attributeNum" label="指标个数" />
    </el-table>
    <el-table
      :data="detailTableData"
      height="300"
      stripe
      border
      style="width: 100%; margin-top: 20px"
    >
      <el-table-column
        type="index"
        :index="indexMethod"
        :label="'id'"
        width="60"
      />
      <el-table-column
        v-for="(value, key) in detailHeader"
        :key="key"
        :prop="value"
        :label="value"
      />
    </el-table>
  </el-dialog>
  <el-dialog v-model="dialogImageVisible" title="可视化结果" width="700">
    <div class="demo-image">
      <div>
        <v-chart class="chart" :option="option" autoresize />
      </div>
    </div>
  </el-dialog>
 </div>
</template>
<script setup lang="ts">
import { ref, reactive, Ref } from "vue";
import axios, { AxiosResponse } from "axios";
import { ElMessage } from "element-plus";
import { ElUpload } from "element-plus";
import { genFileId } from "element-plus";
import { UploadFilled } from "@element-plus/icons-vue";
import type { UploadInstance, UploadProps, UploadRawFile } from "element-plus";
import { use } from "echarts/core";
import { BarChart } from 'echarts/charts'
import { GridComponent, TooltipComponent } from 'echarts/components'
import { CanvasRenderer } from 'echarts/renderers'
import VChart from "vue-echarts";
use([GridComponent, BarChart, CanvasRenderer, TooltipComponent])
const loading = ref(false);
const detailLock = ref(false);
```

```
const reductionLock = ref(false);
const indexMethod = (index: number) => {
  return index + 1;
};
interface FormDataFile {
  file: File;
}
interface DetailData {
  name: string;
  attributeNum: string;
  sampleSize: string;
}
interface RowData {
  [key: string]: string;
}
const dialogTableVisible = ref(false) as Ref<boolean>;
const dialogFormVisible = ref(false) as Ref<boolean>;
const dialogImageVisible = ref(false) as Ref<boolean>;
const form = reactive({
  algorithm: "请选择特征筛选算法(默认 MNIFS 算法)",
  isFeatures: false,
  lammda: 0.4
});
const detailData: DetailData[] = reactive([
  {
    name: "",
    sampleSize: "",
    attributeNum: "",
  },
]);
const matrixData: any[] = [];
const tableData = ref<RowData[]>([]);
const detailMatrixData: any[] = [];
const detailTableData = ref<RowData[]>([]);
const header = ref({}) as Ref<Record<string, string>>;
const detailHeader = ref({}) as Ref<Record<string, string>>;
const upload = ref<UploadInstance>();
const option = ref({});
const reduction = (file: FormDataFile): any => {
  if (reductionLock.value) {
    let formData = new FormData();
    const isreduction = true;
    formData.append("file", file.file);
    formData.append("is_reduction", isreduction.toString()); // 添加额外的参数
    formData.append("algorithm", form.algorithm);
    formData.append("lammda", form.lammda.toString());
    dialogFormVisible.value = false;
    loading.value = true;
    postreduction(formData).then((res: AxiosResponse<any>) => {
      if (res.data.status == "success") {
        console.log(res.data);
        matrixData.splice(0);
        matrixData.push(...JSON.parse(res.data.data));
        tableData.value.splice(0);
        matrixData.forEach((item: any) => {
          let rowData: RowData = {};
          Object.keys(res.data.header).forEach((index: string) => {
            const key = res.data.header[index];
```

```
          let value = item[index];
          rowData[key] = value;
        });
        tableData.value.push(rowData);
      });
      console.log(tableData.value);
      loading.value = false;
      header.value = res.data.header;
      reductionLock.value = false;
      option.value = res.data.option;
    }
  });
 }
 if (detailLock.value) {
  let formData = new FormData();
  formData.append("file", file.file);
  postreduction(formData).then((res: AxiosResponse<any>) => {
   if (res.data.status == "success") {
    detailMatrixData.splice(0);
    detailMatrixData.push(...JSON.parse(res.data.detail_data));
    detailTableData.value.splice(0);
    detailMatrixData.forEach((item: any) => {
     let rowData: RowData = {};
     Object.keys(res.data.detail_header).forEach((index: string) => {
      const key = res.data.detail_header[index];
      let value = item[index];
      rowData[key] = value;
     });
     detailTableData.value.push(rowData);
    });
    ElMessage({
     message: "文件已成功上传",
     type: "success",
    });
    detailData[0].name = res.data.file;
    detailData[0].attributeNum = res.data.col_count;
    detailData[0].sampleSize = res.data.row_count;
    detailHeader.value = res.data.detail_header;
    tableData.value.splice(0);
   }
  });
  detailLock.value = false;
 }
};
const postreduction = (file: FormData) => {
 return axios({
  url: "/api/reduction",
  method: "post",
  data: file,
  headers: {
   "Content-Type": "multipart/form-data",
  },
 });
};
const detail = (file: File | null) => {
 if (file) {
  dialogTableVisible.value = true;
 } else {
```

```
    ElMessage.error("未上传文件");
  }
};
const handleExceed: UploadProps["onExceed"] = (files) => {
  upload.value!.clearFiles();
  const file = files[0] as UploadRawFile;
  file.uid = genFileId();
  upload.value!.handleStart(file);
};
const submitDetail = () => {
  upload.value!.submit();
  detailLock.value = true;
};
const submitreduction = () => {
  upload.value!.submit();
  reductionLock.value = true;
};
</script>
<style>
.el-table .error-row {
  --el-table-tr-bg-color: var(--el-color-error-light-9);
}
.el-table .warning-row {
  --el-table-tr-bg-color: var(--el-color-warning-light-9);
}
.el-table .success-row {
  --el-table-tr-bg-color: var(--el-color-success-light-9);
}
.div_scatter {
  height: 50vh;
  width: 50vw;
}
.chart {
  height: 65vh;
}
</style>
<template>
  <el-card>
    <div class="box">
      <div class="user-circle">
        <img src="@/assets/icons/user.svg" style="height: 150px" />
      </div>
      <div class="text">
        <h3 class="title">欢迎回来，吴医生！</h3>
        <p class="subtitle">智融医诊平台</p>
        <p class="subtitle">融合模糊粒信息(Grc)的医疗辅助诊断系统</p>
        <div class="buttonbox">
          <el-button type="primary" :icon="Platform" size="large" @click="gotoPlat">数据大屏</el-button><br><br><br>
          <el-button type="primary" :icon="Filter" size="large" @click="gotoFilter">特征筛选</el-button>
          <el-button type="primary" :icon="Aim" size="large" @click="gotoAim">异常检测</el-button>
          <el-button type="primary" :icon="View" size="large" @click="gotoView">辅助识别</el-button>
        </div>
      </div>
      <!--患者信息列表-->
      <div style="margin: 0px 0px 0px 0px">
        <div class="things">患者信息评估列表</div>
        <el-table
```

```
        :data="tableData"
        border
        stripe
        style="width: 100%"
        height="300"
      >
        <el-table-column prop="num" label="序号" width="100" />
        <el-table-column prop="card" label="卡号" width="100" />
        <el-table-column prop="name" label="姓名" width="100" />
        <el-table-column prop="room" label="科室" width="100" />
        <el-table-column prop="index" label="指标" width="100" />
        <el-table-column prop="risk" label="风险系数" width="100" />
      </el-table>
    </div>
    <!--代办事务-->
    <div style="margin: 0px 0px 30px 60px">
      <div class="things">待办事务</div>
      <el-timeline style="max-width: 600px">
        <el-timeline-item
          v-for="(activity, index) in activities"
          :key="index"
          :icon="activity.icon"
          :type="activity.type"
          :color="activity.color"
          :size="activity.size"
          :hollow="activity.hollow"
          :timestamp="activity.timestamp"
        >
          {{ activity.content }}
        </el-timeline-item>
      </el-timeline>
    </div>
  </div>
 </el-card>
 <!--走马灯-->
 <div style="margin: 60px 0 0 0">
   <el-carousel :interval="4000" type="card" height="250px">
     <el-carousel-item v-for="item in imglist" :key="item.id">
       <img
         style="object-fit: scale-down; width: 100%; height: 100%"
         :src="item.url"
       />
     </el-carousel-item>
   </el-carousel>
 </div>
</template>
<script setup lang="ts">
import image1 from "@/assets/images/image1.png";
import image2 from "@/assets/images/image2.png";
import image3 from "@/assets/images/image3.png";
import image4 from "@/assets/images/image4.png";
import { Aim, Filter, MoreFilled, Platform, View } from "@element-plus/icons-vue";
import { useRouter } from "vue-router";
let $router=useRouter()
const gotoPlat=()=>{
  $router.push('/screen')
}
```

```
const gotoFilter=()=>{
  $router.push('/analyse/reduce')
}
const gotoAim=()=>{
  $router.push('/analyse/detection')
}
const gotoView=()=>{
  $router.push('/analyse/classification')
}
const tableData = [
  {
    num: "01",
    card: "240807",
    name: "王某某",
    room: "眼科",
    index: "8",
    risk: "75%",
  },
  {
    num: "02",
    card: "240808",
    name: "赵某某",
    room: "内科",
    index: "10",
    risk: "90%",
  },
  {
    num: "03",
    card: "240809",
    name: "李某某",
    room: "外科",
    index: "12",
    risk: "88%",
  },
  {
    num: "04",
    card: "240810",
    name: "张某某",
    room: "骨科",
    index: "6",
    risk: "68%",
  },
  {
    num: "05",
    card: "240811",
    name: "陈某某",
    room: "内科",
    index: "6",
    risk: "25%",
  },
  {
    num: "06",
    card: "240812",
    name: "郑某某",
    room: "口腔科",
    index: "4",
```

```
    risk: "36%",
  },
  {
    num: "07",
    card: "240813",
    name: "汪某某",
    room: "儿科",
    index: "5",
    risk: "67%",
  },
  {
    num: "08",
    card: "240814",
    name: "李某某",
    room: "内科",
    index: "9",
    risk: "82%",
  },
  {
    num: "09",
    card: "240815",
    name: "王某某",
    room: "内科",
    index: "6",
    risk: "45%",
  },
  {
    num: "10",
    card: "240816",
    name: "朱某某",
    room: "儿科",
    index: "5",
    risk: "93%",
  },
];
const activities = [
  {
    content: "Custom icon",
    timestamp: "2024-04-12",
    size: "large",
    type: "primary",
    icon: MoreFilled,
  },
  {
    content: "Custom color",
    timestamp: "2024-04-13",
    color: "#0bbd87",
  },
  {
    content: "Custom size",
    timestamp: "2024-04-15",
    size: "large",
  },
  {
    content: "Custom hollow",
    timestamp: "2024-04-17",
    type: "primary",
```

```
    hollow: true,
  },
  {
    content: "Default node",
    timestamp: "2024-04-20",
  },
];
const imglist = [
  { id: 0, url: image1 },
  { id: 1, url: image2 },
  { id: 3, url: image3 },
  { id: 4, url: image4 },
];
</script>
<style scoped>
.box {
  display: flex;
  .user-circle {
    height: 150px;
    width: 150px;
    border-radius: 75px;
    overflow: hidden;
    margin-left: 10px;
  }
  .text {
    height: 150px;
    .buttonbox{
      height:200px;
      margin:20px 0px;
      padding: 20px 0px;
      transform: translateX(-140px);
    }
  }
  .title {
    font-size: 30px;
    margin: 20px 30px;
    font-weight: 600;
  }
  .subtitle {
    font-size: 16px;
    padding: 10px 30px;
    color: gray;
  }
  .el-carousel__item h3 {
    color: #475669;
    opacity: 0.75;
    line-height: 200px;
    margin: 0;
    text-align: center;
  }
  .things {
    margin: 0 0 20px 0;
    color: grey;
  }
}
</style>
<template>
  <div class="login_container">
```

```
    <el-row>
      <el-col :span="12" :xs="0"></el-col>
      <el-col :span="12" :xs="24">
        <el-form class="login_form">
          <h1>Hello!</h1>
          <h2>欢迎来到智融医诊平台</h2>
          <h3>

                  &n
bsp    
            ——融合模糊粒信息(GrC)的新一代医疗辅助诊断系统
          </h3>
          <el-form-item class="login_item">
            <h4>账号</h4>
            <el-input
              class="login_input"
              :prefix-icon="User"
              v-model="loginForm.username"
            ></el-input>
          </el-form-item>
          <el-form-item>
            <h4>密码</h4>
            <el-input
              class="login_input"
              type="password"
              :prefix-icon="Lock"
              v-model="loginForm.password"
              show-password
            ></el-input>
          </el-form-item>
          <el-form-item>
            <el-button class="login_btn" type="primary" size="default" @click="login()"
              >登录</el-button
            >
          </el-form-item>
        </el-form>
      </el-col>
    </el-row>
  </div>
</template>
<script setup lang="ts">
import { User, Lock } from "@element-plus/icons-vue";
import { reactive } from "vue";
import { useRouter } from "vue-router";
import useUserStore from "@/store/modules/user"
import { ElNotification } from "element-plus";
import {getTime} from '@/utils/time'
let $router=useRouter()
let userStore=useUserStore();
let loginForm = reactive({
  username: "admin",
  password: "111111",
});
const login= async ()=>{
  try {
    await userStore.userLogin(loginForm);
    $router.push('/');
```

```
    ElNotification({
      type:'success',
      message:'欢迎回来！',
      title:getTime()
    })
  } catch (error) {
    ElNotification({
      type:'error',
      message:(error as Error).message
    })
  }
}
</script>
<style scoped lang="scss">
.login_container {
  width: 100%;
  height: 100vh;
  background: url("@/assets/images/login.jpg") no-repeat;
  background-size: cover;
}
.login_form {
  position: relative;
  width: 80%;
  top: 18vh;
  left: 3vw;
  background: #4884ff;
  padding: 30px;
  border-radius: 20px;
  box-shadow: 0px 0px 40px grey;
  h1 {
    color: white;
    font-size: 60px;
    margin: 30px 0px;
    font-family: 'Times New Roman', Times, serif;
  }
  h2 {
    color: white;
    font-size: 30px;
    margin: 30px 0px;
    font-family:'Times New Roman', Times, serif
  }
  h3 {
    color: white;
    font-size: 20px;
    margin: 30px 0px;
    font-family: 'Times New Roman', Times, serif;
  }
  h4{
    color: white;
    font-size: 18px;
    margin: 0 0 0 15px;
  }
  .login_input{
    width: 70%;
    height: 40px;
    margin: 5px 42px;
    font-size: 18px;
  }
```

```
  .login_btn {
    width: 70%;
    margin: 20px auto;
    height: 40px;
    font-size: 18px;
    background: #96cffa
  }
  .login_item{
    display: flex;
  }
}
</style>
<template>
  <div class="background">
    <div class="top">
      <h1 class="title">医疗数据可视化大屏</h1>
      <div style="transform: translateX(32vw); display: flex; margin: 1vh">
        <el-button
          @click="gotoHome"
          color="#ff7070"
          :icon="SwitchButton"
          circle
        />
        <div style="color: #f56c6c; margin: 1vh 0.5vw">退出</div>
      </div>
    </div>
    <div class="content">
      <div class="left">
        <div id="showleft1" class="left1"></div>
        <div id="showleft3" class="left3"></div>
      </div>
      <div class="center">
        <div class="center1">
          <div class="center11">
            <div class="zhongzuo">
              <div class="text1">主任医师门诊人次</div>
              <div style="display: flex">
                <div class="text2">230</div>
                <div class="text3increase">+15%</div>
              </div>
            </div>
            <div class="zhongzuo">
              <div class="text1">副主任医师门诊人次</div>
              <div style="display: flex">
                <div class="text2">450</div>
                <div class="text3decrease">-10%</div>
              </div>
            </div>
            <div class="zhongzuo">
              <div class="text1">主治医师门诊人次</div>
              <div style="display: flex">
                <div class="text2">366</div>
                <div class="text3increase">+22%</div>
              </div>
            </div>
            <div class="zhongzuo">
              <div class="text1">住院医师门诊人次</div>
```

```
            <div style="display: flex">
              <div class="text2">140</div>
              <div class="text3decrease">-8%</div>
            </div>
          </div>
        </div>
        <div id="showcenter12" class="center12"></div>
      </div>
      <div id="showcenter2" class="center2"></div>
    </div>
    <div class="right">
      <div class="right1">
        <div class="things">科室就诊量一览表</div>
        <el-table
          :data="tableData2"
          border
          stripe
          size="small"
          style="width: 100%"
          height="280"
        >
          <el-table-column prop="num" label="序号" width="50" />
          <el-table-column prop="room" label="科室" width="90" />
          <el-table-column prop="people" label="门诊人次" width="90" />
          <el-table-column prop="rate" label="同比增幅" width="90" />
          <el-table-column prop="satisfy" label="综合满意度" width="90" />
        </el-table>
      </div>
      <div class="right2">
        <div class="things">患者信息评估列表</div>
        <el-table :data="tableData1" border stripe size="small" height="340">
          <el-table-column prop="num" label="序号" width="50" />
          <el-table-column prop="card" label="卡号" width="80" />
          <el-table-column prop="name" label="姓名" width="80" />
          <el-table-column prop="room" label="科室" width="80" />
          <el-table-column prop="index" label="指标" width="50" />
          <el-table-column prop="risk" label="风险系数" width="70" />
        </el-table>
      </div>
    </div>
  </div>
 </div>
</template>
<script setup lang="ts">
import * as echarts from "echarts";
import { SwitchButton } from "@element-plus/icons-vue";
import { onMounted } from "vue";
import { useRouter } from "vue-router";
let $router = useRouter();
const gotoHome = () => {
 $router.push("/home");
};
const tableData2 = [
 {
   num: "01",
   room: "外科",
```

```
    people: "70",
    rate: "7.2%",
    satisfy: "85%",
  },
  {
    num: "02",
    room: "内科",
    people: "62",
    rate: "16.4%",
    satisfy: "83%",
  },
  {
    num: "03",
    room: "儿科",
    people: "45",
    rate: "4.8%",
    satisfy: "77%",
  },
  {
    num: "04",
    room: "眼科",
    people: "30",
    rate: "7%",
    satisfy: "89%",
  },
  {
    num: "05",
    room: "耳鼻喉科",
    people: "28",
    rate: "25%",
    satisfy: "82%",
  },
  {
    num: "06",
    room: "妇科",
    people: "20",
    rate: "27%",
    satisfy: "88%",
  },
  {
    num: "07",
    room: "产科",
    people: "18",
    rate: "-17%",
    satisfy: "85%",
  },
  {
    num: "08",
    room: "肿瘤科",
    people: "10",
    rate: "-33%",
    satisfy: "86%",
  },
];
const tableData1 = [
  {
    num: "01",
```

card: "240807",
name: "王某某",
room: "眼科",
index: "8",
risk: "75%",
},
{
num: "02",
card: "240808",
name: "赵某某",
room: "内科",
index: "10",
risk: "90%",
},
{
num: "03",
card: "240809",
name: "李某某",
room: "外科",
index: "12",
risk: "88%",
},
{
num: "04",
card: "240810",
name: "张某某",
room: "骨科",
index: "6",
risk: "68%",
},
{
num: "05",
card: "240811",
name: "陈某某",
room: "内科",
index: "6",
risk: "25%",
},
{
num: "06",
card: "240812",
name: "郑某某",
room: "口腔科",
index: "4",
risk: "36%",
},
{
num: "07",
card: "240813",
name: "汪某某",
room: "儿科",
index: "5",
risk: "67%",
},
{
num: "08",

card: "240814",
name: "李某某",
room: "内科",
index: "9",
risk: "82%",
},
{
num: "09",
card: "240815",
name: "王某某",
room: "内科",
index: "6",
risk: "45%",
},
{
num: "10",
card: "240816",
name: "朱某某",
room: "儿科",
index: "5",
risk: "93%",
},
{
num: "11",
card: "240807",
name: "王某某",
room: "眼科",
index: "8",
risk: "75%",
},
{
num: "12",
card: "240808",
name: "赵某某",
room: "内科",
index: "10",
risk: "90%",
},
{
num: "13",
card: "240809",
name: "李某某",
room: "外科",
index: "12",
risk: "88%",
},
{
num: "14",
card: "240810",
name: "张某某",
room: "骨科",
index: "6",
risk: "68%",
},
{
num: "15",

```
    card: "240811",
    name: "陈某某",
    room: "内科",
    index: "6",
    risk: "25%",
  },
  {
    num: "16",
    card: "240812",
    name: "郑某某",
    room: "口腔科",
    index: "4",
    risk: "36%",
  },
  {
    num: "17",
    card: "240813",
    name: "汪某某",
    room: "儿科",
    index: "5",
    risk: "67%",
  },
  {
    num: "18",
    card: "240814",
    name: "李某某",
    room: "内科",
    index: "9",
    risk: "82%",
  },
  {
    num: "19",
    card: "240815",
    name: "王某某",
    room: "内科",
    index: "6",
    risk: "45%",
  },
  {
    num: "20",
    card: "240816",
    name: "朱某某",
    room: "儿科",
    index: "5",
    risk: "93%",
  },
];
type EChartsOption = echarts.EChartsOption;
const showcenter2 = () => {
  var app: any = {};
  var chartDom: any = document.getElementById("showcenter2");
  chartDom.removeAttribute("_echarts_instance_");
  var myChart = echarts.init(chartDom);
  var option;
  const categories = (function () {
    let now = new Date();
```

```
  let res = [];
  let len = 10;
  while (len--) {
    res.unshift(now.toLocaleTimeString().replace(/^\D*/, ""));
    now = new Date(+now - 2000);
  }
  return res;
})();
const categories2 = (function () {
  let res = [];
  let len = 10;
  while (len--) {
    res.push(10 - len - 1);
  }
  return res;
})();
const data = (function () {
  let res = [];
  let len = 10;
  while (len--) {
    res.push(Math.round(Math.random() * 100+20));
  }
  return res;
})();
const data2 = (function () {
  let res = [];
  let len = 0;
  while (len < 10) {
    res.push(+(Math.random() * 10 + 5).toFixed(1));
    len++;
  }
  return res;
})();
option = {
  title: {
    text: "今日就诊量动态图",
  },
  tooltip: {
    trigger: "axis",
    axisPointer: {
      type: "cross",
      label: {
        backgroundColor: "#283b56",
      },
    },
  },
  grid: {
    bottom: 40,
  },
  legend: {},
  toolbox: {
    show: false,
    feature: {
      dataView: { readOnly: false },
      restore: {},
      saveAsImage: {},
    },
  },
```

```
  dataZoom: {
    show: false,
    start: 0,
    end: 100,
  },
  xAxis: [
    {
      type: "category",
      boundaryGap: true,
      data: categories,
    },
    {
      type: "category",
      boundaryGap: true,
      data: categories2,
    },
  ],
  yAxis: [
    {
      type: "value",
      scale: true,
      name: "急诊流量",
      max: 30,
      min: 0,
      boundaryGap: [0.2, 0.2],
    },
    {
      type: "value",
      scale: true,
      name: "门诊流量",
      max: 120,
      min: 0,
      boundaryGap: [0.2, 0.2],
    },
  ],
  series: [
    {
      name: "门诊流量",
      type: "bar",
      xAxisIndex: 1,
      yAxisIndex: 1,
      data: data,
    },
    {
      name: "急诊流量",
      type: "line",
      data: data2,
    },
  ],
};
app.count = 11;
setInterval(function () {
  let axisData = new Date().toLocaleTimeString().replace(/^\D*/, "");
  data.shift();
  data.push(Math.round(Math.random() * 100 + 20));
  data2.shift();
  data2.push(+(Math.random() * 10 + 5).toFixed(0));
```

```
    categories.shift();
    categories.push(axisData);
    categories2.shift();
    categories2.push(app.count++);
    myChart.setOption({
      xAxis: [
        {
          data: categories,
        },
        {
          data: categories2,
        },
      ],
      series: [
        {
          data: data,
        },
        {
          data: data2,
        },
      ],
    });
  }, 2100);
  option && myChart.setOption(option);
};
const showcenter12 = () => {
  var chartDom: any = document.getElementById("showcenter12");
  chartDom.removeAttribute("_echarts_instance_");
  var myChart = echarts.init(chartDom);
  var option: EChartsOption;
  option = {
    title: {
      text: "科室就诊量分布图",
      left: "left",
    },
    tooltip: {
      trigger: "item",
    },
    legend: {
      orient: "vertical",
      left: 350,
      top: 60,
    },
    series: [
      {
        name: "科室就诊量",
        type: "pie",
        radius: [70, 140],
        center: ["37%", "53%"],
        avoidLabelOverlap: false,
        padAngle: 2,
        itemStyle: {
          borderRadius: 6,
        },
        label: {
          show: false,
          position: "center",
        },
```

```
      emphasis: {
        label: {
          show: true,
          fontSize: 40,
          fontWeight: "bold",
        },
      },
      labelLine: {
        show: false,
      },
      data: [
        { value: 70, name: "外科" },
        { value: 62, name: "内科" },
        { value: 45, name: "儿科" },
        { value: 30, name: "眼科" },
        { value: 28, name: "耳鼻喉科" },
        { value: 20, name: "妇科" },
        { value: 18, name: "产科" },
        { value: 10, name: "肿瘤科" },
      ],
    },
  ],
 };
 option && myChart.setOption(option);
};
const showleft3 = () => {
 var chartDom: any = document.getElementById("showleft3");
 chartDom.removeAttribute("_echarts_instance_");
 var myChart = echarts.init(chartDom);
 var option;
 option = {
  title: {
    text: "手术情况统计表",
    left: "left",
  },
  legend: { orient: "vertical", left: 300, top: 50 },
  tooltip: {},
  grid: {
    bottom: "10%",
  },
  dataset: {
    source: [
      ["product", "去年同期", "本月", "上月"],
      ["一级手术", 300, 260, 280],
      ["二级手术", 207, 176, 184],
      ["三级手术", 157, 170, 130],
      ["四级手术", 80, 60, 40],
    ],
  },
  xAxis: { type: "category" },
  yAxis: { max: 300 },
  series: [{ type: "bar" }, { type: "bar" }, { type: "bar" }],
 };
 option && myChart.setOption(option);
};
const showleft1 = () => {
```

```
var chartDom: any = document.getElementById("showleft1");
chartDom.removeAttribute("_echarts_instance_");
var myChart = echarts.init(chartDom);
var option: EChartsOption;
option = {
  title: {
    text: "今日就诊患者性别年龄分布图",
    left: "left",
  },
  tooltip: {
    trigger: "item",
    formatter: "{a} <br/>{b}: {c} ({d}%)",
  },
  legend: {
    orient: "vertical",
    left: 300,
    bottom: 50,
    data: [
      "80 岁以上",
      "70-80 岁",
      "60-70 岁",
      "50-60 岁",
      "40-50 岁",
      "30-40 岁",
      "20-30 岁",
      "10-20 岁",
      "0-10 岁",
    ],
  },
  series: [
    {
      name: "性别",
      type: "pie",
      selectedMode: "single",
      center: ["35%", "55%"],
      radius: [0, "30%"],
      data: [
        { value: 1048, name: "女" },
        { value: 775, name: "男" },
      ],
      label: {
        position: "inner",
      },
      labelLine: {
        show: false,
      },
    },
    {
      name: "年龄范围",
      type: "pie",
      center: ["35%", "55%"],
      radius: ["40%", "85%"],
      data: [
        { value: 153, name: "80 岁以上" },
        { value: 256, name: "70-80 岁" },
        { value: 310, name: "60-70 岁" },
```

```
          { value: 251, name: "50-60 岁" },
          { value: 180, name: "40-50 岁" },
          { value: 120, name: "30-40 岁" },
          { value: 50, name: "20-30 岁" },
          { value: 23, name: "10-20 岁" },
          { value: 24, name: "0-10 岁" },
        ],
        label: {
          position: "inner",
        },
        labelLine: {
          show: false,
        },
      },
    ],
  };
  option && myChart.setOption(option);
};
onMounted(() => {
  showcenter2();
  showcenter12();
  showleft3();
  showleft1();
});
</script>
<style scoped>
.background {
  background-color: #343a3f;
  overflow: hidden;
}
.top {
  width: 100vw;
  height: 9vh;
  display: flex;
  justify-content: center;
  padding: 2vh;
  .title {
    font-size: 30px;
    font-weight: 600;
    /* background: linear-gradient(to bottom, #00ffff, #13215c); */
    background: linear-gradient(to bottom, #5c7bd9, white);
    background-clip: text;
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
  }
}
.content {
  width: 100vw;
  height: 91vh;
  padding: 0 30px;
  display: flex;
  .left {
    width: 25vw;
    .left1 {
      height: 40vh;
      margin: 0.5vh;
      box-shadow: 0 0 5px black;
```

```
    padding: 0.5vh;
    background-color: #f5f7fb;
    border-radius: 5px;
  }
  .left1:hover {
    background-color: #f0f0f5;
  }
  .left3 {
    height: 47.5vh;
    margin: 1vh 0.5vh 1vh 0.5vh;
    box-shadow: 0 0 5px black;
    padding: 0.5vh;
    background-color: #f5f7fb;
    border-radius: 5px;
  }
  .left3:hover {
    background-color: #f0f0f5;
  }
 }
 .center {
  width: 46vw;
  padding: 0.5vh;
  .center1 {
    height: 42vh;
    display: flex;
    .center11 {
      width: 19vw;
      margin: 0 0.5vh 0.5vh 0;
      box-shadow: 0 0 5px black;
      padding: 0.5vh;
      background-color: #f5f7fb;
      border-radius: 5px;
      .zhongzuo {
        height: 9vh;
        margin: 1vh;
        border-radius: 1vh;
        box-shadow: 0 0 10px grey;
        .text1 {
          padding: 1vh;
          text-align: center;
          font-size: 20px;
          font-weight: 600;
        }
        .text2 {
          padding: 0 0 0 2.5vw;
          width: 60%;
          text-align: center;
          font-size: 30px;
          font-weight: 1000;
          background: linear-gradient(to bottom, #ff9800, #eae60e);
          background-clip: text;
          -webkit-background-clip: text;
          -webkit-text-fill-color: transparent;
        }
        .text3increase {
          padding: 0.5vh 5vw 0 0;
          width: 40%;
          text-align: center;
```

```
        font-size: 20px;
        font-weight: 600;
        background: linear-gradient(to bottom, #c6a236, #e1483c);
        background-clip: text;
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
      }
      .text3decrease {
        padding: 0.5vh 5vw 0 0;
        width: 40%;
        text-align: center;
        font-size: 20px;
        font-weight: 600;
        background: linear-gradient(to bottom, #36dfe4, #d5f9c9);
        background-clip: text;
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
      }
    }
    .zhongzuo:hover {
      background-color: #f0f0f5;
    }
  }
  .center12 {
    width: 29vw;
    margin: 0 0 0.5vh 0.5vh;
    box-shadow: 0 0 5px black;
    padding: 0.5vh;
    background-color: #f5f7fb;
    border-radius: 5px;
  }
  .center12:hover {
    background-color: #f0f0f5;
  }
 }
 .center2 {
  height: 46vh;
  margin: 0.5vh 0 0.5vh 0;
  box-shadow: 0 0 5px black;
  padding: 0.5vh;
  background-color: #f5f7fb;
  border-radius: 5px;
 }
 .center2:hover {
  background-color: #f0f0f5;
 }
}
.right {
 width: 25vw;
 .right1 {
  height: 40vh;
  margin: 0.5vh;
  box-shadow: 0 0 5px black;
  padding: 0.5vh;
  background-color: #f5f7fb;
  border-radius: 5px;
 }
 .right1:hover {
```

```
      background-color: #f0f0f5;
    }
    .right2 {
      height: 47.5vh;
      margin: 1vh 0.5vh 1vh 0.5vh;
      box-shadow: 0 0 5px black;
      padding: 0.5vh;
      background-color: #f5f7fb;
      border-radius: 5px;
    }
    .right2:hover {
      background-color: #f0f0f5;
    }
  }
}
.things {
  margin: 0 0 1vh 0;
  color: #464646;
  font-weight: 600;
  font-size: 16px;
}
</style>
import { defineConfig } from "vite";
import vue from "@vitejs/plugin-vue";
import path from "path";
import { createSvgIconsPlugin } from "vite-plugin-svg-icons";
import { viteMockServe } from "vite-plugin-mock";
export default defineConfig(({ command }) => {
  return {
    plugins: [
      vue(),
      createSvgIconsPlugin({
        iconDirs: [path.resolve(process.cwd(), "src/assets/icons")],
        symbolId: "icon-[dir]-[name]",
      }),
      viteMockServe({
        enable: command === "serve", //保证开发阶段可以使用 mock 接口
      }),
    ],
    resolve: {
      alias: {
        "@": path.resolve("./src"),
      },
    },
    css: {
      preprocessorOptions: {
        scss: {
          javascriptEnable: true,
          additionalData: "@import './src/styles/variable.scss';",
        },
      },
    },
    server: {
      proxy: {
        "/api": {
          target: "http://127.0.0.1:5000", // 访问数据的计算机域名
          ws: true, // 是否启用 websockets
```

```
        changeOrigin: true, //开启代理,
        rewrite: (path) => path.replace(/^\/api/, ''), // 重写代理规则，/api 开头，代理到/
      }
    }
  },
 };
});
```

注： 本软件使用 Visual Studio Code 与 Pycharm 开发,程序量为 3542 行。