1 实验目的

（1）掌握线性SVM和基于核的SVM的调用方法

（2）掌握不平衡数据集分类的代价敏感方法

（3）能够恰当调节支持向量机的关键参数

（4）掌握GridSearch调参方法

2 任务描述

2.1 基于支持向量机的系统故障识别

大数据时代，分布式系统成为信息存储和处理的主流系统。相对于传统系统而言，分布式系统更为庞大和复杂，故障发生的平均几率比较高，其运维的难度和复杂度大大提高。在分布式计算环境中，系统的一个或多个节点或组件发生故障或异常，导致系统的正常运行受到影响或出现性能下降。由于分布式系统由多个节点和组件构成，并且节点之间相互依赖，一个节点的故障可能会对整个系统产生连锁反应，引发级联故障。因此，分布式系统故障检测是确保系统正常运行的关键环节。通过有效的故障检测，可以提高分布式系统的可用性和性能，保障用户体验和业务连续性，降低故障带来的损失和影响。本次实验在数据预处理试验的基础上，进一步实施，使用支持向量机做系统故障识别任务。

2.2 数据集

本次实验选用系统故障诊断数据集，该数据集中包含107个系统KPI指标（已做脱敏处理，由feature0、feature1 ...feature106表示）以及故障类型（共6个类别，用0、1、2、3、4、5表示，0类为正常），实验者需要对提供的数据集进行自行划分，并经过模型训练获得性能较好的系统故障分类结果。KPI（关键绩效指标）是用于衡量和评估组织、系统或流程绩效的量化指标。在信息技术和系统管理领域，KPI被广泛应用于衡量系统的运行状态、性能和健康状况。当系统发生故障时，会引起相关的KPI指标发生异常。通过对KPI指标的监测可以度量系统的健康状况和性能，并在故障诊断和性能调优中起着重要作用。

常见的KPI指标如下：

吞吐量（Throughput）：表示系统在单位时间内处理的请求或事务数量。

延迟（Latency）：表示系统响应请求所需的平均时间。

错误率（Error Rate）：表示系统在处理请求时发生错误的频率。

可用性（Availability）：表示系统处于可运行状态的时间比例。

响应时间（Response Time）：指从用户发起请求到系统返回响应的时间。

系统负载（System Load）：表示系统当前的负荷水平。

资源利用率（Resource Utilization）：表示系统资源（如CPU、内存、磁盘）的使用率。

故障率（Failure Rate）：表示单位时间内系统发生故障的频率。

系统可扩展性（System Scalability）：表示系统能在负载增加时保持性能水平。

故障恢复时间（Fault Recovery Time）：指系统从故障中恢复正常运行所需的时间。

3 相关知识

3.1支持向量机

支持向量机（Support Vector Machine，SVM）是一种常用的监督学习算法，用于二分类和多分类任务。其主要原理是将数据点映射到高维空间中，找到一个最优的超平面来将不同类别的数据点分开，使得不同类别的数据尽可能远离超平面。

常用的核函数有线性核（Linear Kernel）、多项式核（Polynomial Kernel）、高斯径向基函数核（Radial Basis Function Kernel，RBF Kernel）、sigmoid核等。

3.2 GridSearch主要原理

GridSearch（网格搜索）是一种用于超参数优化的技术，常用于机器学习算法的调参过程。其主要原理是通过在预定义的参数空间中穷举搜索，尝试不同的参数组合，以找到最优的参数配置，从而优化模型的性能。GridSearch的优点是简单且直观，可以帮助寻找较好的超参数组合。然而，由于它的穷举搜索方式，当参数空间较大时，可能会产生大量的计算开销。因此，在实践中，可以结合其他技术如随机搜索（RandomizedSearch）或优化算法（如贝叶斯优化）来加速调参过程并找到更优的参数配置。

在使用GridSearchCV时，需要指定一些参数来配置搜索的方式和评估：

sklearn.model\_selection.GridSearchCV(estimator, param\_grid, \*, scoring=None, n\_jobs=None, iid='deprecated', refit=True, cv=None, verbose=0, pre\_dispatch='2\*n\_jobs', error\_score=nan, return\_train\_score=False)

estimator：需要调优的模型，例如一个分类器或回归器对象。

param\_grid：一个字典或列表，其中的键是模型的超参数名称，值是对应的取值列表。

scoring：要使用的评估指标，可以是字符串（如'accuracy'、'f1'等）或一个可调用的评估函数。

cv：交叉验证的折数，用于计算每个参数组合的性能。

verbose：控制输出的详细程度，通常设置为一个非负整数来显示更多信息。

1. 实验要求与步骤

实验准备：本次实验应用Python包有pandas、seaborn、matplotlib、joblib、sklearn、numpy。如下命令导入：

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import accuracy\_score, classification\_report, make\_scorer

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.preprocessing import MinMaxScaler

from sklearn.feature\_selection import mutual\_info\_classif

import numpy as np

from sklearn.svm import SVC

from sklearn.metrics import f1\_score

import joblib

实验步骤：

（1）自行判断，进行必要的数据预处理。

def data\_preprocess(data):

    processed\_data = data

    # ---------Begin-----------#

    # 请进行合理的预处理,处理后的数据命名为processed\_data

    means = processed\_data.mean()

    processed\_data.fillna(means, inplace=True)

    scalar = MinMaxScaler()

    processed\_data = pd.DataFrame(scalar.fit\_transform(processed\_data.iloc[:, :-1]))

    processed\_data['label'] = data.iloc[:, -1]

    print(processed\_data)

    # ---------End-------------#

    X = processed\_data.iloc[:, :-1]  #拟输入模型的数据

    y = processed\_data.iloc[:, -1]   #分类标签

    return X, y

（2）自主选择，进行必要的特征工程。

def feature\_selected():

    data\_frame = load\_data()

    X\_train, y\_train = data\_preprocess(data\_frame)

    # 使用全部特征

    feature\_set = X\_train.columns

    feature\_set = list(X\_train.columns)

    # 特征选择，返回保留的特征集合feature\_set

    # ---------Begin-----------#

    # 特征选择

    from sklearn.ensemble import RandomForestClassifier

    # 创建随机森林分类器,进行特征选择。参数设置：n\_estimators=100, random\_state=42

    rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

    # 训练模型

    rf\_classifier.fit(X\_train, y\_train)

    # 获取特征重要性

    feature\_importance = rf\_classifier.feature\_importances\_

    # 对特征重要性进行排序

    sorted\_indices = np.argsort(feature\_importance)[::-1]

    # 指定阈值来保留特定数量的特征,命名为feature\_set

    num\_top\_features = 80

    feature\_set = sorted\_indices[:num\_top\_features]

     # ---------End-------------#

    return feature\_set

1. 基于GridSearchCV，首先在param\_grid中定义参数范围，组合多种参数；创建GridSearchCV对象，使用5折交叉验证，f1-score作为优化标准。由于数据集为多分类，需要设置f1\_score中的average参数为'macro'。尝试找出最佳参数组合，获得最高f1-score。

代码示例

# 定义要调优的参数范围

param\_grid = {

'C': [5, 10, 15, 20, 25, 30], # 惩罚系数

'gamma': [0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5] # 核函数参数

}

# 设置average参数为'macro'

f1 = make\_scorer(f1\_score, average='macro') # 设置average参数为'macro'

# 创建GridSearchCV对象

grid\_search= GridSearchCV(SVC(kernel="rbf", class\_weight="balanced", probability=True), param\_grid=param\_grid, cv=5,scoring=f1)

# 进行参数优化

results = grid\_search.fit(train\_xdata[feature\_set], train\_ydata)

# 输出最优的参数组合

print("最优参数：", grid\_search.best\_params\_)

1. 测试说明

1. 分析与讨论
2. 本实验任务，对数据做预处理，如缺失值填充、归一化，是否能改善分类性能？

可以

1. 在特征筛选过程中，如何确定选用特征数量？

测试

1. 总结归纳本次调参经验

在结果较好区间继续gridsearch