

2023— 2024学年（一）学期

大作业名称：基于决策树算法

对新冠病毒数据的分析

姓 名：

年 级:

专业班级：

学 号：

|  |  |  |  |
| --- | --- | --- | --- |
| **评分标准** | **各项标准满分** | **学生得分** | **总分** |
| 格式准确，语言通顺流畅，逻辑性强 | 10 |  |  |
| 设计目的明确 | 15 |  |
| 结果分析正确 | 15 |  |
| 对业务内容分析透彻合理、全面、详细 | 30 |  |
| 数据模型、算法模型运用得当，有实际价值 | 30 |  |
| **备注** | 如有大段抄袭或雷同文章均为不合格 | |

注:评分标准和各项标准满分仅供教师参考

目录

[一、总体介绍 1](#_Toc87903384)

[二、设计思路与方案 2](#_Toc87903385)

[三、模型说明 3](#_Toc87903386)

[四、实验运行结果及分析](#_Toc87903387) 5

[五、实验总结](#_Toc87903388) 12

[六、感想及体会 12](#_Toc87903389)

# 一．总体介绍

## 1.1实验背景

新冠肺炎，全称为新型冠状病毒肺炎（COVID-19），是由一种名为新型冠状病毒（SARS-CoV-2）引起的传染性疾病。这种病毒最初于2019年底在中国武汉被发现，并很快在全球范围内传播。COVID-19的症状多种多样，从轻微感冒症状到严重呼吸困难甚至死亡。常见症状包括发热、咳嗽、乏力、呼吸困难、味觉或嗅觉丧失等。

大多数感染COVID-19病毒的人会经历轻度到中度的呼吸道疾病，并且无需特殊治疗即可康复。年龄较大的人和患有心血管疾病、糖尿病、慢性呼吸道疾病和癌症等基础医学问题的人更容易出现严重疾病。

在整个疫情期间，医疗保健提供者面临的主要问题之一是医疗资源短缺和有效分配的合理计划。在这些艰难时刻，能够预测个人在被检测为阳性时甚至在此之前可能需要的资源类型，对当局将是巨大的帮助，因为他们可以采购和安排必要的资源来挽救患者的生命。

## 1.2实验目的

本次实验的主要目标是建立一个机器学习模型，根据新冠肺炎患者目前的症状、状态和病史，预测患者是否处于感染新冠肺炎的高风险状态。为了实现该目标，我们将运用决策树进行分类。

## 1.3数据集介绍

这个[数据集](https://datos.gob.mx/busca/dataset/informacion-referente-a-casos-covid-19-en-mexico)由墨西哥政府提供。该数据集包含大量的匿名患者相关信息，包括先决条件。原始数据集包括21个不同特征和1,048,576位独特的患者。在布尔型特征中，1表示“是”，2表示“否”。数值为97和99表示缺失数据。

分为以下几个标签：

* sex: 1代表女性，2代表男性。
* age: 患者的年龄。
* classification: COVID测试结果。数值1-3表示患者以不同程度被诊断为COVID。数值4或更高表示患者不携带COVID或测试结果不确定。
* patient type: 患者在单位中接受的护理类型。1表示回家，2表示住院。pneumonia: 患者是否已经有肺部感染。
* pregnancy: 患者是否怀孕。
* diabetes: 患者是否患有糖尿病。
* copd: 指示患者是否患有慢性阻塞性肺病。
* asthma: 患者是否患有哮喘。
* inmsupr: 患者是否免疫抑制。
* hypertension: 患者是否患有高血压。
* cardiovascular: 患者是否患有心脏或血管相关疾病。
* renal chronic: 患者是否患有慢性肾脏疾病。
* other disease: 患者是否患有其他疾病。
* obesity: 患者是否肥胖。
* tobacco: 患者是否吸烟。
* usmr: 指示患者是否接受了第一、第二或第三级医疗单位的治疗。
* medical unit提供护理的国家卫生系统机构类型。
* intubed: 患者是否连接到呼吸机。
* icu: 指示患者是否被送入重症监护室。
* date died: 如果患者已故，则指示死亡日期；否则为9999-99-99。

# 二．设计思路与方案

## 2.1整体思路

对下载数据集进行预处理，去除缺失数据，之后使用One-hot编码，将离散特征的取值扩展到欧式空间。

同时将“死亡”“插管”“ICU”三列数据整合作为高风险数据，也就是我们预测的对象。这样问题就转化为了一个二分类问题。

之后确定最适合的决策树算法进行分类，根据信息增益和基尼指数选择最好的分裂策略。

最后使用多种指标查看分类模型的好坏同时绘制混淆矩阵。

## 2.2数据预处理

#去掉无效数据

covid = covid.loc[(covid.CLASIFFICATION\_FINAL < 4)]

covid = covid.loc[(covid.SEX == 1) | (covid.SEX == 2)]

covid = covid.loc[(covid.USMER == 1) | (covid.USMER == 2)]

covid = covid.loc[(covid.PATIENT\_TYPE == 1) | (covid.PATIENT\_TYPE == 2)]

covid = covid.loc[(covid.PNEUMONIA == 1) | (covid.PNEUMONIA == 2)]

covid = covid.loc[(covid.DIABETES == 1) | (covid.DIABETES == 2)]

covid = covid.loc[(covid.COPD == 1) | (covid.COPD == 2)]

covid = covid.loc[(covid.ASTHMA == 1) | (covid.ASTHMA == 2)]

covid = covid.loc[(covid.INMSUPR == 1) | (covid.INMSUPR == 2)]

covid = covid.loc[(covid.HIPERTENSION == 1) | (covid.HIPERTENSION == 2)]

covid = covid.loc[(covid.OTHER\_DISEASE == 1) | (covid.OTHER\_DISEASE == 2)]

covid = covid.loc[(covid.CARDIOVASCULAR == 1) | (covid.CARDIOVASCULAR == 2)]

covid = covid.loc[(covid.OBESITY == 1) | (covid.OBESITY == 2)]

covid = covid.loc[(covid.RENAL\_CHRONIC == 1) | (covid.RENAL\_CHRONIC == 2)]

covid = covid.loc[(covid.TOBACCO == 1) | (covid.TOBACCO == 2)]

#one-hot编码

covid.SEX = covid.SEX.apply(lambda x: x if x == 1 else 0)

covid.USMER = covid.USMER.apply(lambda x: x if x == 1 else 0) # no = 0, yes = 1

covid.PATIENT\_TYPE = covid.PATIENT\_TYPE.apply(lambda x: 0 if x == 1 else 1)

covid.PNEUMONIA = covid.PNEUMONIA.apply(lambda x: x if x == 1 else 0)

covid.DIABETES = covid.DIABETES.apply(lambda x: x if x == 1 else 0)

covid.COPD = covid.COPD.apply(lambda x: x if x == 1 else 0)

covid.ASTHMA = covid.ASTHMA.apply(lambda x: x if x == 1 else 0)

covid.INMSUPR = covid.INMSUPR.apply(lambda x: x if x == 1 else 0)

covid.HIPERTENSION = covid.HIPERTENSION.apply(lambda x: x if x == 1 else 0)

covid.OTHER\_DISEASE = covid.OTHER\_DISEASE.apply(lambda x: x if x == 1 else -0)

covid.CARDIOVASCULAR = covid.CARDIOVASCULAR.apply(lambda x: x if x == 1 else 0)

covid.OBESITY = covid.OBESITY.apply(lambda x: x if x == 1 else 0)

covid.RENAL\_CHRONIC = covid.RENAL\_CHRONIC.apply(lambda x: x if x == 1 else 0)

covid.TOBACCO = covid.TOBACCO.apply(lambda x: x if x == 1 else 0)

covid.DATE\_DIED = covid.DATE\_DIED.apply(lambda x: 0 if x == "9999-99-99" else 1)

covid.PREGNANT = covid.PREGNANT.apply(lambda x: x if x == 1 else 0)

covid.INTUBED = covid.INTUBED.apply(lambda x: x if x == 1 else 0)

covid.ICU = covid.ICU.apply(lambda x: x if x == 1 else 0)

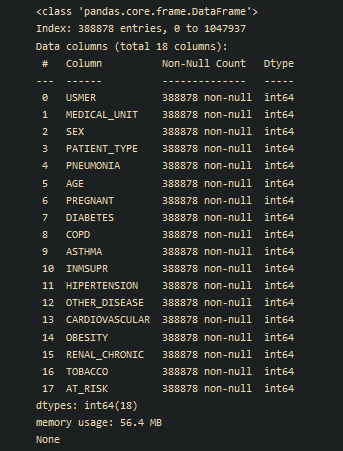
#合并高风险数据

covid['AT\_RISK'] = covid['DATE\_DIED'] + covid['INTUBED'] + covid['ICU']

covid.AT\_RISK = covid.AT\_RISK.apply(lambda x: 1 if x > 0 else 0)

#查看数据

print(covid.info())



## 2.3决策树算法原理

决策树的基本原理：

决策树的主要思想是通过一系列的“问题”来划分数据集。每个问题都是关于特征的，它将数据集划分成更纯的子集。这个过程会一直持续，直到所有的数据点都被正确地分类或者到达了树的深度限制。

主要概念：

**节点（Node）：** 决策树的每个分割点，代表了数据集中的一个特征及其分割标准。

**分支（Branch）：** 从一个节点到另一个节点的路径，代表了针对特定特征的不同取值。

**叶子节点（Leaf Node）：** 决策树的末端节点，代表了最终的输出类别或值。

**根节点（Root Node）：** 决策树的初始节点，整个树的起始点。

分裂准则：

决策树在构建过程中使用分裂准则来确定如何选择最佳的特征进行数据划分。这些准则通常包括：

**信息增益（Information Gain）：** 以熵的减小量来衡量划分前后数据集的不确定性减少程度。熵是数据集中信息的度量，信息增益计算划分前后熵的差值。

**基尼不纯度（Gini Impurity）：** 基尼不纯度是指数据集中一个元素被错误分类的概率。基尼系数越小表示数据集纯度越高，用于度量划分的纯度。

# 三．模型说明

## 3.1 决策树模型

直接使用sklearn中的决策树模型并进行调参。

在Scikit-Learn的决策树模型中，有一些关键参数可以控制模型的行为，如max\_depth（树的最大深度）、criterion（切分标准）、splitter（分裂策略）等。

优点和缺点：

**优点：** 决策树易于理解和解释，对数据预处理的要求较低，能够处理多种数据类型。

**缺点：** 可能会产生过拟合，对噪声和离群值较敏感。

通过调整参数和优化分割策略，可以改善决策树模型的性能，并根据数据的特点得到更好的预测结果。

在实际调参时，heuristics 和 splitters 是用于构建决策树的参数选项。

heuristics 参数指定了用于计算节点分裂质量的指标。在决策树的节点分裂过程中，根据这些指标来选择最佳的分裂方式。通常有两种常用的指标：信息增益和基尼不纯度。

"entropy" 使用信息增益作为分裂标准，尝试最大化每次分裂中的信息增益，使得子节点的不确定性（熵）降低。

"gini" 使用基尼不纯度作为分裂标准，尝试最小化每次分裂中的基尼不纯度，使得子节点的不纯度降低。

splitters 参数表示节点分裂时考虑的策略。

"best" 尝试在所有特征的所有可能切分点中找到最佳的切分点。

"random" 在随机的一部分特征子集中找到最佳切分点，这有助于减少过拟合并增加模型的多样性。

代码如下：

for d in max\_depth:

dt\_fmeasures[d] = np.zeros(len(heuristics)\*len(splitters))

i=0

for heuristic, splitter in [(heuristic, splitter) for heuristic in heuristics for splitter in splitters]:

# create a Decision Tree classifier instance and compute the prediction

dt\_classifier = DecisionTreeClassifier(criterion=heuristic, splitter=splitter, max\_depth=d, class\_weight='balanced')

dt\_classifier.fit(train\_x, train\_y)

y\_pred = dt\_classifier.predict(validation\_x)

# calculate F-measures and save best model

dt\_fmeasures[d][i] = get\_Fmeasure(y\_pred, validation\_y)

if dt\_fmeasures[d][i] > dt\_best\_result:

dt\_best\_result = dt\_fmeasures[d][i]

dt\_best\_model = dt\_classifier

i=i+1

后续根据F1指标的好坏选择效果最好的决策树模型。

即为：（splitter = random, max depth = 7）

## 3.2 评价指标

1·**精确度（Precision）：** 精确度是指真正例（True Positives）的比例，即预测为正例中有多少是真正的正例。它关注于模型在预测为正例时的准确性。

2·**召回率（Recall）：** 召回率是指被正确预测为正例的样本占实际正例的比例，也称为灵敏度。它关注于模型对正例样本的识别能力。

3·**F1 值（F1-Score）：** F1 值是精确度和召回率的调和平均值，它综合了这两个指标的性能。F1 值越高，模型在精确度和召回率上的表现越好。

4·**支持度（Support）**：每个类别的样本数量，表示数据集中属于该类别的样本个数。

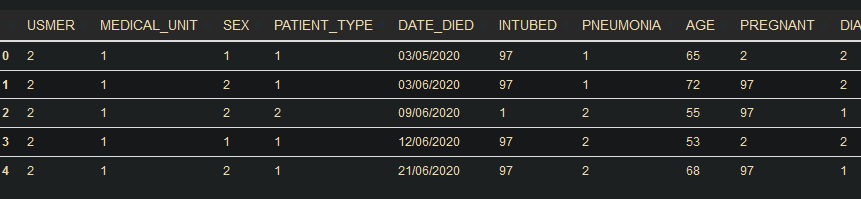
5·**准确率（Accuracy）：** 准确率是指分类正确的样本数占总样本数的比例。这是最常见的评估指标，但在不平衡数据集中，可能不是唯一的选择，因为它不能反映在不同类别中的错误情况。

6·**混淆矩阵（Confusion Matrix）：** 混淆矩阵是一种以表格形式呈现模型预测结果的可视化工具。它展示了模型预测的真正例、假正例、真负例和假负例的数量。

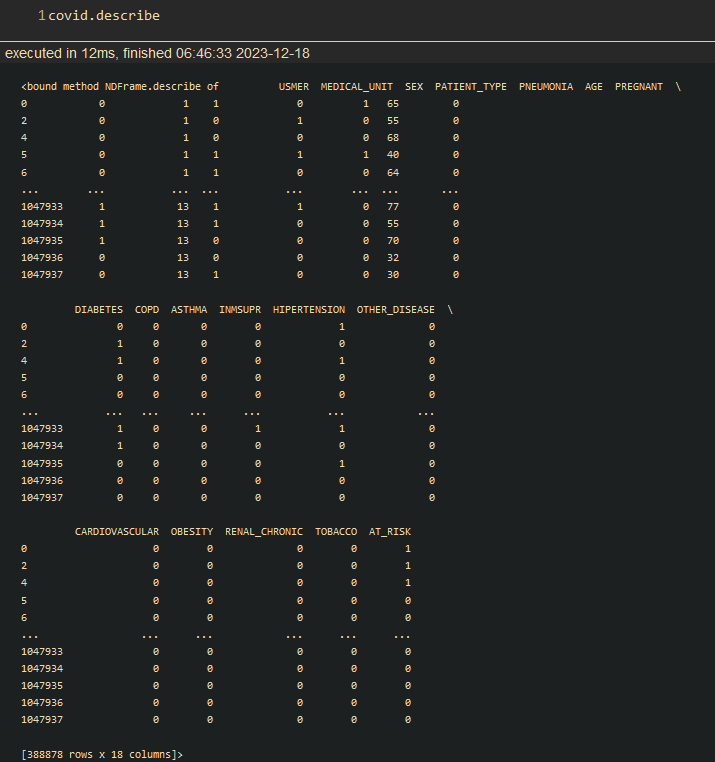
# 四．实验运行结果及分析

## 4.1 数据预处理

去掉空数据后如图



数据整合后如图：



按照9:1划分训练集，再按照1：1划分验证集，测试集。

#train(90%), validation(5%) and test(5%)

train, validation = train\_test\_split(covid, test\_size=0.1, shuffle=True)

validation, test = train\_test\_split(validation, test\_size=0.5, shuffle=True)

validation\_y = validation.AT\_RISK.to\_numpy()

validation\_x = validation.drop(columns = ['AT\_RISK']).to\_numpy()

test\_y = test.AT\_RISK.to\_numpy()

test\_x = test.drop(columns = ['AT\_RISK']).to\_numpy()

## 4.2 选择合适的决策树模型

因为转化为了二分类问题，故先自主定义F1指标，用于评价不同决策树预测效果的好坏

def get\_Fmeasure(prediction, labels):

# Compares the prediction of each object to its label

true\_positive = np.sum((prediction + labels) == 2)

false\_positive = np.sum((labels - prediction) == -1)

false\_negative = np.sum((prediction - labels) == -1)

positive = np.count\_nonzero(labels == 1)

negative = len(labels)-positive

# Calculate F-measure

accuracy = 100\*np.sum(prediction == labels)/len(labels)

recall = true\_positive / (true\_positive + false\_negative)

precision = true\_positive / (true\_positive + false\_positive)

eval = 2\*(precision \* recall) / (precision + recall)

return eval

之后：

heuristics = ["entropy", "gini"]

splitters = ["best", "random"]

设置四个模型（l1,l2,l3,l4）

for k in dt\_fmeasures.keys():

fmeasures = dt\_fmeasures[k]

plt.scatter([k] \* len(fmeasures), fmeasures)

l1.append(dt\_fmeasures[k][0])

l2.append(dt\_fmeasures[k][1])

l3.append(dt\_fmeasures[k][2])

l4.append(dt\_fmeasures[k][3])

# create line for each of the four models

plt.errorbar(max\_depth, l1, label ='id3 best')

plt.errorbar(max\_depth, l2, label ='id3 random')

plt.errorbar(max\_depth, l3, label ='gini best')

plt.errorbar(max\_depth, l4, label ='gini random')

plt.legend(loc ='lower right')

plt.title('Decision Trees values comparison')

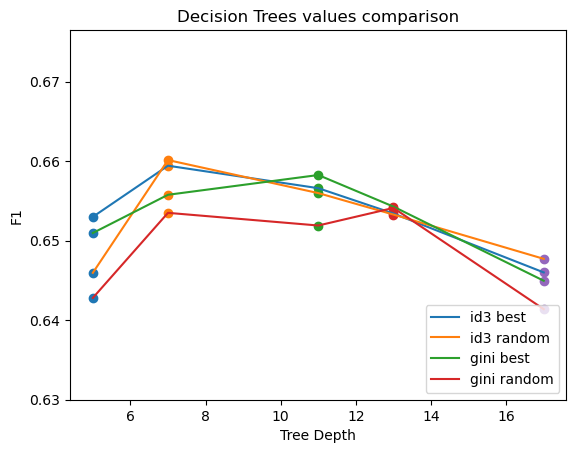
plt.xlabel('Tree Depth')

plt.ylabel('F1')

plt.ylim(0.630, 0.6765)

plt.show()

模型结果如图所示



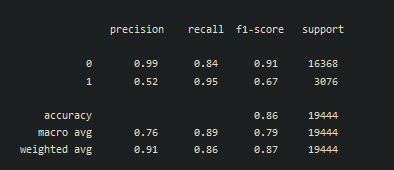


可知，当heuristic = entropy, splitter = random时，模型效果最好，此时F1指标为0.66，考虑了precision和recall两方面的因素。

## 4.3 定义并查看不同指标的评价

使用sklearn库函数classification\_report：

print('\n', classification\_report(test\_y, dt\_pred))



混淆矩阵：

test\_y = np.array(test\_y)

dt\_pred = np.array(dt\_pred)

y\_test = test\_y.copy()

y\_pred = dt\_pred.copy()

c\_matrix\_tree = confusion\_matrix(y\_test, y\_pred)

print(c\_matrix\_tree) #打印出来看看



from sklearn.metrics import confusion\_matrix # 生成混淆矩阵函数

import matplotlib.pyplot as plt # 绘图库

import numpy as np

import tensorflow as tf

def plot\_confusion\_matrix(cm, labels\_name, title):

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis] # 归一化

plt.imshow(cm, interpolation='nearest') # 在特定的窗口上显示图像

plt.title(title) # 图像标题

plt.colorbar()

num\_local = np.array(range(len(labels\_name)))

plt.xticks(num\_local, labels\_name, rotation=90) # 将标签印在x轴坐标上

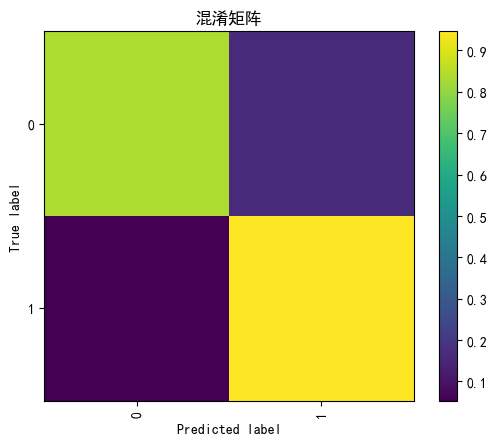
plt.yticks(num\_local, labels\_name) # 将标签印在y轴坐标上

plt.ylabel('True label')

plt.xlabel('Predicted label')

plot\_confusion\_matrix(c\_matrix\_tree, [0,1], "混淆矩阵")

plt.show()



可以看出该模型的效果属于比较好的。

# 五．实验总结

通过本次实验，学到了决策树处理分类问题，以及分类问题需要使用哪些指标。

**使用决策树进行分类：**

**直观易懂**：决策树是一种直观的模型，类似于流程图，易于理解和解释。它们展示了特征之间的关系以及导致特定结果的路径。

**特征选择的重要性**：决策树在每个节点选择最佳特征进行分割，因此特征选择对模型性能至关重要。了解如何选择最相关的特征是关键。

**过拟合风险**：决策树容易过拟合训练数据，特别是在树很深或没有限制树的生长时。剪枝和设置树的最大深度等技术可以减少过拟合的风险。

**集成方法的优势**：决策树可以作为集成方法的基础，例如随机森林和梯度提升树。这些集成方法能够综合多个决策树的预测结果，提高模型的性能和稳定性。

**参数调优的重要性**：调整决策树的参数对模型性能有很大影响。选择正确的分裂标准、树的深度、叶子节点的最小样本数等参数是优化模型的关键。

**处理类别不平衡**：当类别不平衡时，决策树可能偏向预测样本量较大的类别。采用平衡样本权重或使用其他技术来处理类别不平衡是重要的。

**分类问题的评价指标：**

**多样性的指标**：分类问题的评价指标有很多种，每种指标都关注模型性能的不同方面。例如准确率、召回率、精确度、F1 分数等。理解每个指标的特点以及何时使用它们是很重要的。

**平衡性与权衡**：有些指标关注某个方面的性能，但在实际问题中可能需要平衡多个指标。例如，在处理不平衡数据时，仅仅依赖准确率可能会产生误导，需要结合其他指标如召回率或 F1 分数来全面评估模型。

**业务需求决定指标选择**：评价指标的选择应该基于具体的业务需求和问题背景。比如在医疗领域，假阴性（False Negatives）可能比假阳性（False Positives）更严重，这将影响召回率的重要性。

**可解释性和清晰度**：评价指标应该能够清晰地传达模型性能。有些指标更易于解释和沟通，能够更好地向非技术人员解释模型的表现。

**实验和迭代**：在尝试不同模型和参数设置时，评价指标是优化模型的关键。通过观察指标的变化情况，可以了解模型的表现，并进行针对性的调整。

了解不同的评价指标及其应用场景，有助于更全面地评估模型的性能，选择适合特定问题的模型，从而更好地解决实际挑战。

# 六．感想及体会

**首先数据质量至关重要**：新冠肺炎数据的质量对于分析和挖掘的结果至关重要。首先尽量找官方数据集，确保数据准确、完整，并且针对特定目的进行了适当的清洗和预处理。此外政府为了保证隐私，隐藏了真实信息用编号代替姓名，不会对数据的有效性造成影响。

此外对二分类问题有了更深一层的了解，也更深一步学习了决策树算法和sklearn库。选择合适的算法和模型是关键。不同的问题需要不同的方法，需要不断尝试和比较，以找到最佳的解决方案。

数据挖掘领域在不断发展，新的算法和工具层出不穷。很多以前需要手动实现的函数现在都整合成了很方便的包，增加了数据分析的便利程度，因此保持学习的状态，跟上技术的发展是非常重要的。

最后数据挖掘不仅是技术层面的工作，更是发现数据背后价值的过程。从数据中挖掘出有意义的见解对业务决策至关重要。通过这次实验我明白了数据挖掘在现实生活中的实际价值，和生活密不可分，来自现实并能造福现实。