

上海理工大学光电信息与计算机工程学院

**《数据挖掘》实验报告**

****

**专　　业 智能科学与技术**

**姓 名　　高浩琦**

**学　 号 2035060413**

**年　　级 2020级**

**指导教师 孙占全**

**成 绩：**

**教师签字：**

# 实验一 数据预处理与可视化

一、实验目的和要求

1. 掌握数据预处理方法，如缺失值修复、数据标准化等。

2. 掌握数据挖掘中的分类聚类方法和分类性能的评价估计

3. 熟悉ROC和PR曲线评估结果的计算过程

4. 以鸢尾花数据集为例（'http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data），学会使用python做去除缺失数据记录或用均值补全缺失数据、做数据标准化、萼片和花瓣长宽属性10等分的直方图、萼片和花瓣长宽属性箱式图、萼片和花瓣长宽属性两两之间散点图

5. 基于鸢尾花数据集，利用决策数分类器进行分类，画出测试结果的ROC和PR曲线。

二、实验算法

Pd.read\_csv()

Data.describe()

Data.iloc()

Data.isnull()

Plt.hist()

Plt.boxplot()

Plt.scatter()

Plt.subplot()

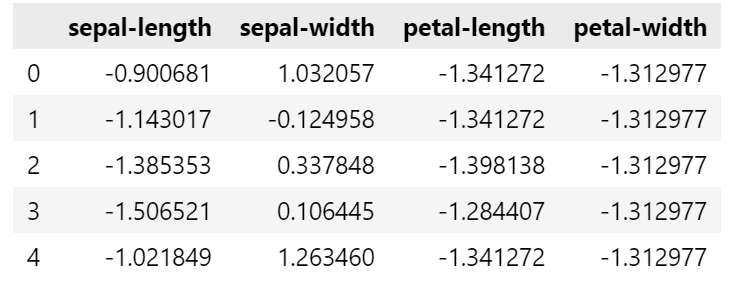
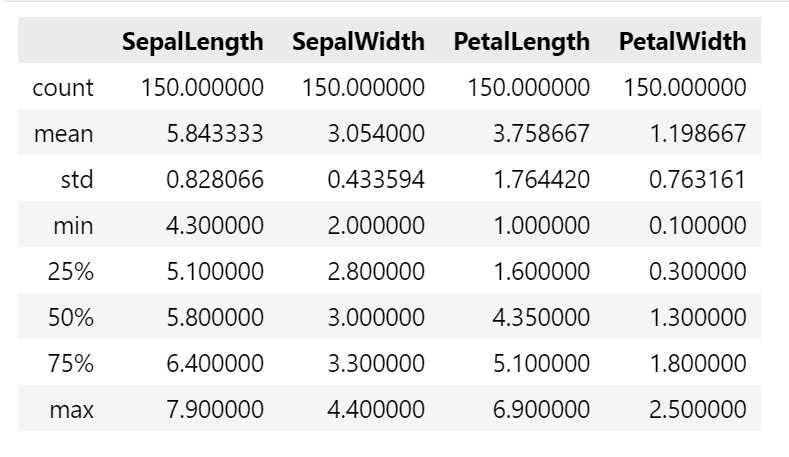
Plt.xlable()

Plt.ylable()

StandardScaler()

三、实验过程记录

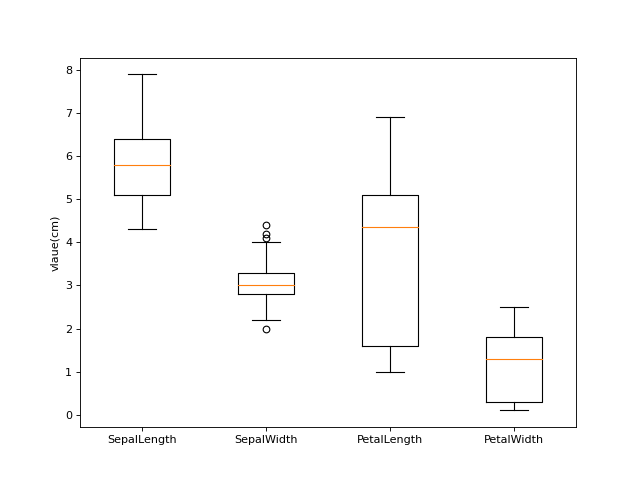
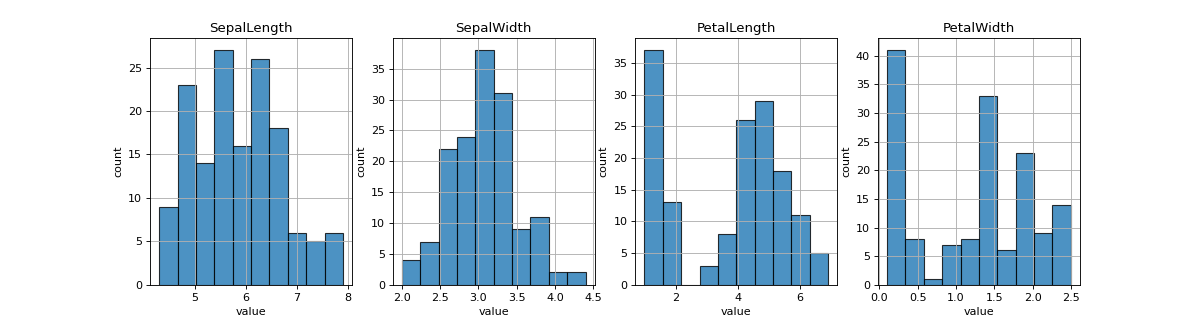
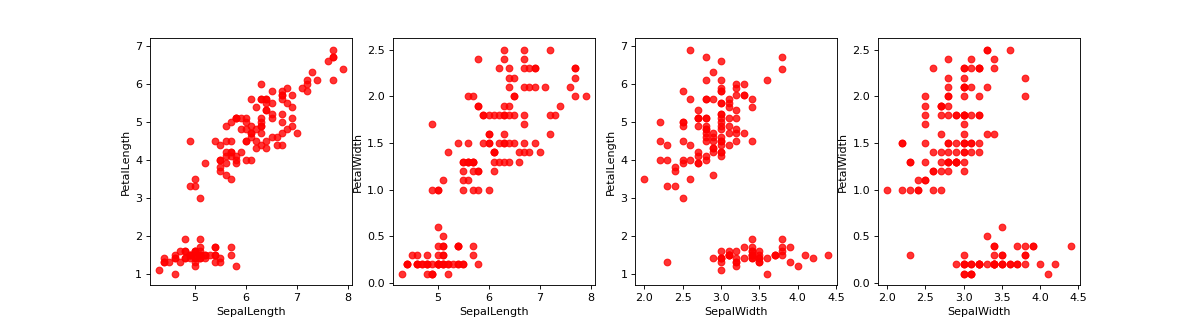
1.导入鸢尾花数据集（'http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data），对数据进行标准化处理

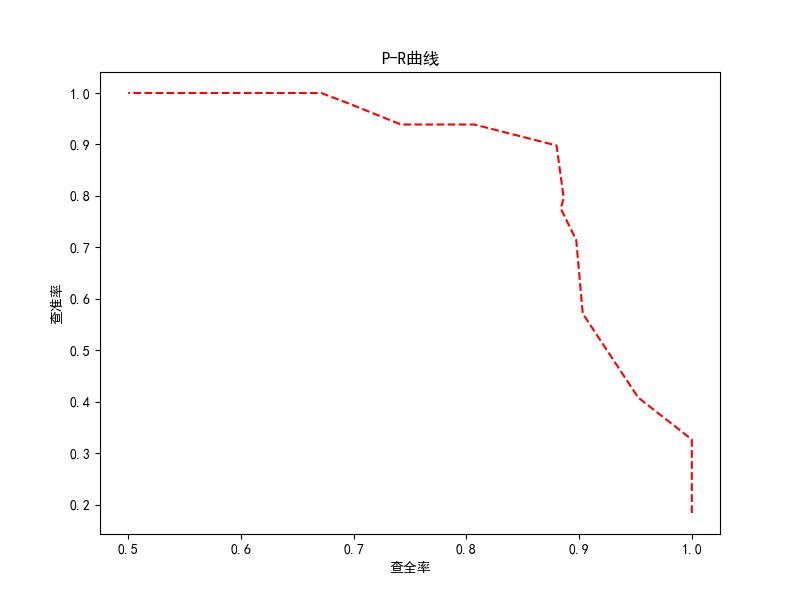
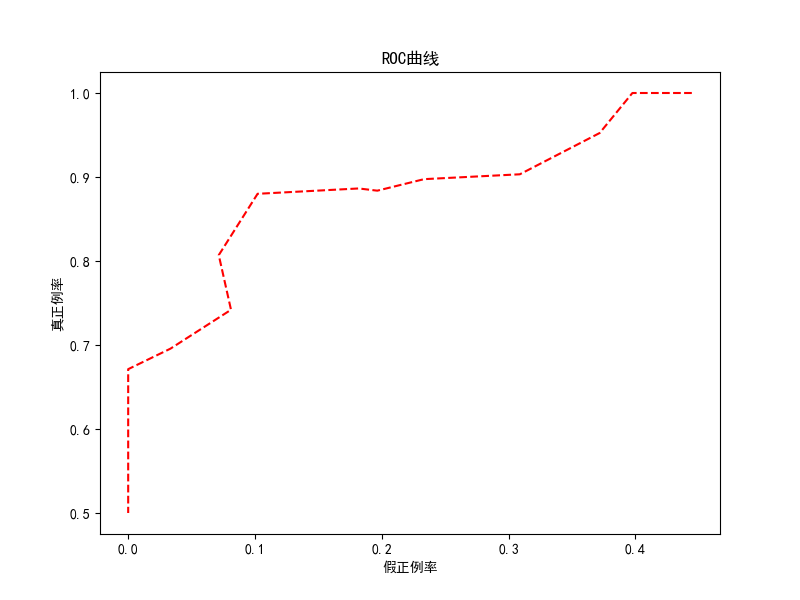


2.分别进行直方图、散点图、箱式图的绘制

3.从4.5的标准开始，通过for语句和if语句的循环嵌套，绘制pr和roc曲线

四、结果与分析

 1. 直方图、散点图、箱式图

 2. PR and ROC

五、自评

经历几天几夜的耕耘后，数据预处理方法、分类聚类方法、分类性能的评价估计、ROC和PR曲线评估结果的计算过程、决策数分类器等等均已经大致掌握。

但是由于自身的python 编程能力还不是很高，在编程方面还是有些许吃力，还是希望自己之后再接再厉，争取更上一层楼。

上海理工大学光电信息与计算机工程学院

**《数据挖掘》实验报告**

****

**专　　业 智能科学与技术**

**姓 名　　 高浩琦**

**学　 号 2035060413**

**年　　级 2020级**

**指导教师 孙占全**

**成 绩：**

**教师签字：**

# 实验二 KNN、Bayes和SVM分类

1. 实验目的和要求

1、掌握kNN分类器的基本原理，利用kNN分类器实现wine.csv数据分类，40%做测试；

2、在SVM的分类算法理论学习基础上，通过实验了解SVM分类的应用；

1. 掌握用Python使用SVM算法对数字图像识别digit数据集进行分类，测试分类准确率，并画出学习曲线；

4、学会使用Python训练和测试Bayes分类器，利用Bayes分类器对mashroom.csv数据进行分类，并对结果进行分A析，画ROC曲线。

二、实验算法

KNN

from sklearn.datasets import load\_wine

from sklearn.neighbors import KNeighborsClassifier

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

import pandas as pd

import matplotlib.pyplot as plt

data = load\_wine()

data\_x = data.data

data\_y = data.target

xtrain,xtest,ytrain,ytest = train\_test\_split(data\_x,data\_y,test\_size = 0.4 ,random\_state = 1 )

# print(xtest.shape)

score = []

k = [3,5,7]

i = 0

for x in k:

    knn = KNeighborsClassifier(n\_neighbors = x)

    clf = knn.fit(xtrain,ytrain)

    score.append(clf.score(xtest,ytest))

    i = i + 1

print("k = 3:",score[0])

print("k = 5:",score[1])

print("k = 7",score[2])

fig = plt.figure(alpha = 0.5)

plt.bar(k,score,color = 'b')

plt.xlabel = "k"

plt.ylabel = "ppp"

# plt.legend = "ss"

# fig.align\_ylabels = "p"

plt.show()

BAYES

from sklearn.preprocessing import LabelEncoder

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.naive\_bayes import GaussianNB

from sklearn.naive\_bayes import BernoulliNB

data = pd.read\_csv("mushroom.csv",header= None)

## 字母转化为数字 ##

labelincoder = LabelEncoder()

for col in data.columns:

    data[col] = labelincoder.fit\_transform(data[col])

    # print(col)

## 提取 类标签和变量属性

y = data[0]

x = data.drop(0, axis = 1)

xtrain,xtest,ytrain,ytest = train\_test\_split(x,y,test\_size = 0.3 ,random\_state = 1 )

## 标准化

ss\_x = StandardScaler()

xtrain = ss\_x.fit\_transform(xtrain)

xtest = ss\_x.fit\_transform(xtest)

## 计算模型

clf = GaussianNB()

model\_tree = clf.fit(xtrain,ytrain)

score = []

score.append(model\_tree.score(xtest,ytest))

# clf = MultinomialNB()

# model\_tree = clf.fit(xtrain,ytrain)

# score = model\_tree.score(xtest,ytest)

# print(score)

clf = BernoulliNB()

model\_tree = clf.fit(xtrain,ytrain)

score.append(model\_tree.score(xtest,ytest))

print("GaussianNB:",score[0])

print("BernoulliNB:",score[0])

SVM

import matplotlib.pylab as plt

import pandas as pd

from sklearn.datasets import load\_digits

digits = load\_digits()

y = digits.target

# %%

df = pd.DataFrame(data = digits.data)

df.head()

# %%

from sklearn.model\_selection import KFold

kf = KFold(n\_splits= 6, shuffle= True)

# %%

df.loc[1:6]

# %%

# for xtrain,xtest in kf.split():

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler

score = []

for train\_index, test\_index in kf.split(df):

    # print("TRAIN:", train\_index.size, "\nTEST:", test\_index.size)

    X\_train, X\_test = df.loc[train\_index], df.loc[test\_index]

    Y\_train, Y\_test = y[train\_index], y[test\_index]

    # ss\_x = StandardScaler()

    # X\_train = ss\_x.fit\_transform(X\_train)

    # X\_test = ss\_x.fit\_transform(X\_test)

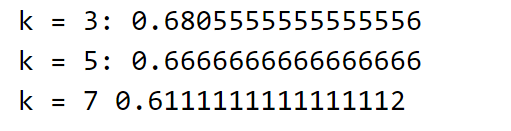
    svm = SVC()

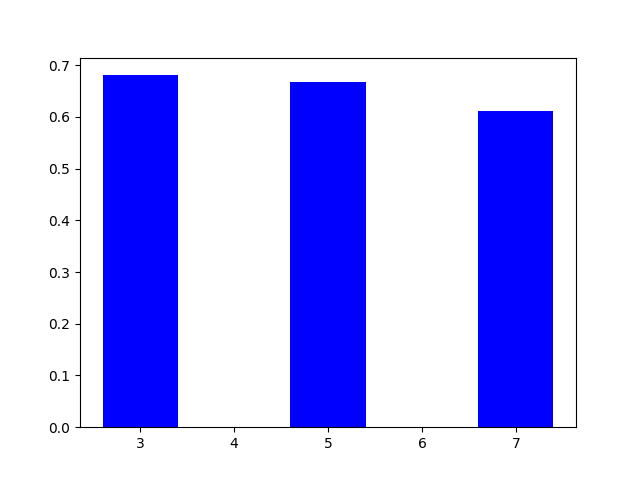
    dlf = svm.fit(X\_train,Y\_train)

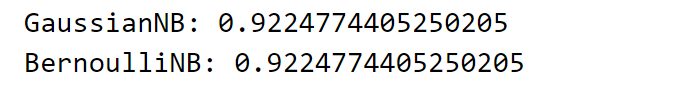
    score.append(dlf.score(X\_test,Y\_test))

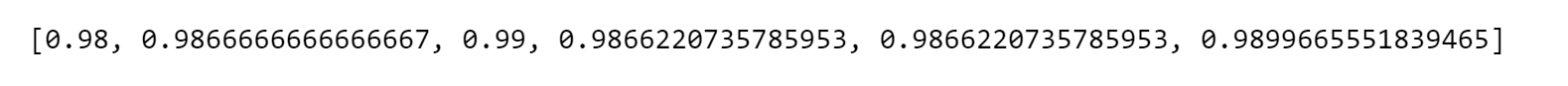
print(score)

三、实验过程记录

1. KNN



2. Bayes

3. SVM

四、结果与分析

1. 由结果可知，KNN算法对于wine数据来说并不是一个很好的拟合算法，在k = 3,5,7 拟合概率均在0.6上下，由此可见KNN算法对训练数据依赖度特别大，对训练数据的容错性太差。如果训练数据集中，有一两个数据是错误的，刚刚好又在需要分类的数值的旁边，这样就会直接导致预测的数据的不准确。所以在预测该类型数据的时候我们要尽量避免使用这种算法。

2. 贝叶斯分类器的分类原理是通过某对象的先验概率，从几个利用贝叶斯公式计算出其后验概率，即该对象属于某一类的概率，选择具有最大后验概率的类作为该对象所属的类。也就是说，贝叶斯分类器是最小错误率意义上的优化。目前研究较多的贝叶斯分类器主要有四种，分别是：Naive Bayes、TAN、BAN和GBN。实验中采用了GBN和BAN分类器，结果均超过了90%，非常不错。

3．SVM分类思想很简单，就是将样本与决策面的间隔最大化，实验中采用了K-折交叉验证，结果产生了很好的拟合优度，分类效果较好。

五、自评

通过三次的实验课后，基本明白了数据挖掘对数据的处理与相关模型的拟合，掌握了python-sklearn库的使用，收获满满。

上海理工大学光电信息与计算机工程学院

**《数据挖掘》实验报告**

****

**专　　业 智能科学与技术**

**姓 名　　 高浩琦**

**学　 号 2035060413**

**年　　级 2020级**

**指导教师 孙占全**

**成 绩：**

**教师签字：**

# 实验三 Apriori关联分析算法

1. 实验目的和要求
2. 学会在具体应用上使用Apriori算法
3. 运用Python编程实现Apriori算法计算过程，并对数据进行分析。
4. 通过改变支持度阈值，得到不同的频繁项集和关联规则。
5. 实例分析，读取story.txt文件，每行为一个事件，事件包含多个项，项之间用逗号分割，最小支持度0.2，最小置信度0.7。可自己尝试不同的阈值。

二、实验算法

from apyori import apriori

import numpy as np

def info(results):

    for result in results:

        # for i in result.ordered\_statistics:

        # print(result.items)

        for i in result.ordered\_statistics:

            if i.items\_base and i.items\_add:

                print(list(i.items\_base) , "====>" , list(i.items\_add), "confidence:",i.confidence,"lift:",i.lift)

            else:

                print("skip")

f = open("story.txt",'r',encoding="utf-8")

orders = []

for i in f:

    orders.append(i.strip("\n").split(","))

print(orders)

results = apriori(transactions=orders, min\_support = 0.2 ,min\_confidence = 0.7)

support = np.arange(1,10,1)/10

confidence = np.arange(1,10,1)/10

for i in support:

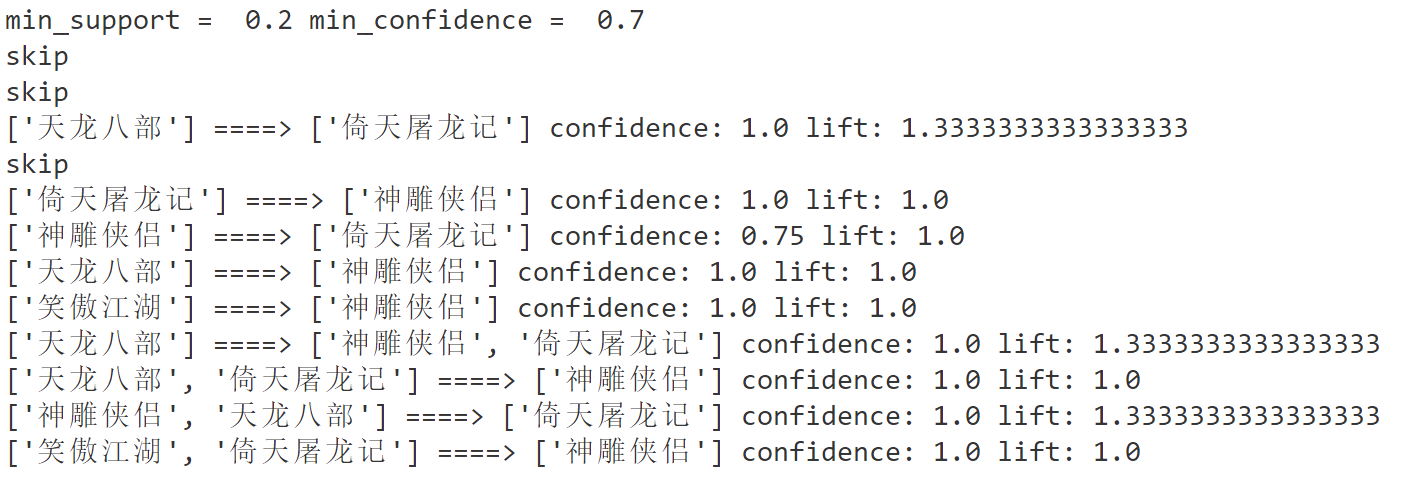
    for j in confidence:

        results = apriori(transactions=orders, min\_support = i ,min\_confidence = j)

        print("min\_support = ",i,"min\_confidence = ",j)

        info(results)

        print()

三、实验过程记录

四、结果与分析

从是按结果来看, Apriori 算法能够广泛应用于各种领域，通过对数据的关联性进行了分析和挖掘，挖掘出的这些信息在决策制定过程中具有重要的参考价值。

五、自评

Apriori 算法的基本思想是：首先找出所有的频集，这些项集出现的频繁性至少和预定义的最小支持度一样。然后由频集产生强关联规则，这些规则必须满足最小支持度和最小可信度。然后使用第1步找到的频集产生期望的规则，产生只包含集合的项的所有规则，其中每一条规则的右部只有一项，这里采用的是中规则的定义。一旦这些规则被生成，那么只有那些大于用户给定的最小可信度的规则才被留下来。为了生成所有频集，还使用了递归的方法。

上海理工大学光电信息与计算机工程学院

**《数据挖掘》实验报告**

****

**专　　业 智能科学与技术**

**姓 名　　高浩琦**

**学　 号 2035060413**

**年　　级 2020级**

**指导教师**

**成 绩：**

**教师签字：**

# 实验四 K-means聚类

一、实验目的和要求

1，掌握基于距离的K-means聚类的计算过程

2，通过Python编程还原计算过程，对数据进行聚类

3，通过运行程序更改不同K值对同一数据进行聚类，理解K-means聚类中K均值初始值选择对聚类造成的差异

二、实验算法

import numpy as np

import PIL.Image as image

from sklearn.cluster import KMeans

def loadData(filePath):

    data = []

    img = image.open(filePath)

    m,n = img.size

    for i in range(m):

        for j in range(n):

            x,y,z = img.getpixel((i,j))

            data.append([x/256.0,y/256.0,z/256.0])

    return  np.mat(data),m,n

imgData,row,col = loadData('block.jpg')

label = KMeans(n\_clusters=4).fit\_predict(imgData)

label = label.reshape([row,col])

pic\_new = image.new("L",(row,col))

for i in range(row):

    for j in range(col):

        pic\_new.putpixel((i,j),int(256/(label[i][j]+1)))

pic\_new.save("result-bull-4.jpg","JPEG")

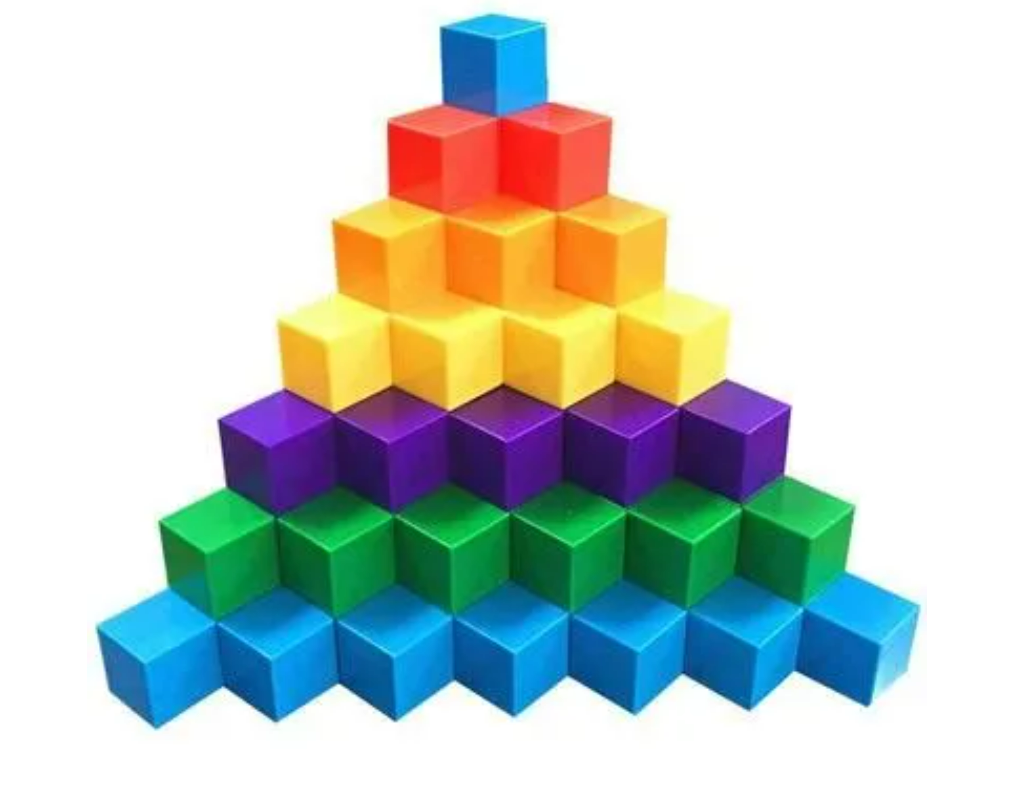
三、实验过程记录

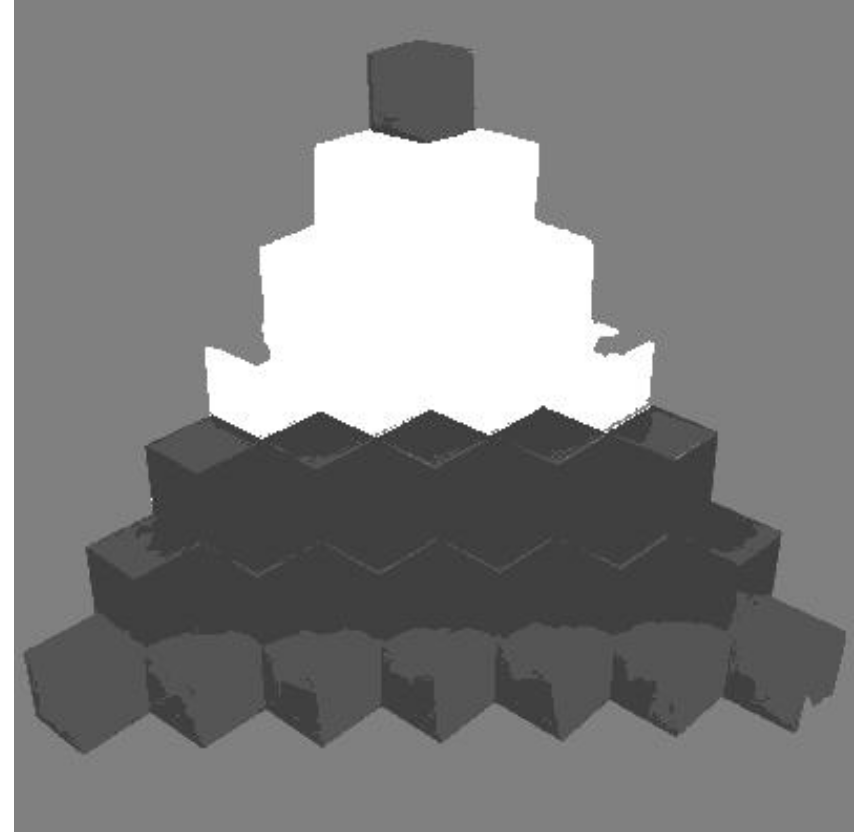
1．读取图片数据

2．利用kmeans算法计算图片

3. 输出图片

四、结果与分析





五、自评

Kmeans算法是最常用的聚类算法，主要思想是:在给定K值和K个初始类簇中心点的情况下，把每个点(亦即数据记录)分到离其最近的类簇中心点所代表的类簇中，所有点分配完毕之后，根据一个类簇内的所有点重新计算该类簇的中心点(取平均值)，然后再迭代的进行分配点和更新类簇中心点的步骤，直至类簇中心点的变化很小，或者达到指定的迭代次数。

通过本次实验 ，对kmeans算法的理解大大加深了。