重庆邮电大学

学生实验实习报告册

学年学期：2022-2023 学年(秋)学期

课程名称：数据工程综合实践

学生学院：计算机学院/人工智能学院

专业班级：数据科学与大数据技术

学生学号：2022211631

学生姓名：万厚源

联系电话：16746870502

重庆邮电大学教务处印制

目　录

教师评阅记录表

【重要说明】

- 学生提交报告册最终版时，**必须包含此页**，否则不予成绩评定。
- 本报告册模板内容格式除确实因为填写内容改变了布局外，**不得变更其余部分的格式**，否则不予成绩评定。

报告是否符合考核规范	☑ 符合　☐ 不符合
报告格式是否符合标准	☑ 符合　☐ 不符合
报告是否完成要求内容	☑ 是　☐ 否
报告评语：	
报告成绩：	
评阅人签名（签章） 2023 年 1 月 10 日	

实验或实习报告

课程名称	数据工程综合实践	课程编号	A2041050
开课学院	计算机科学与技术学院/人工智能学院		
指导教师	XX		
实验实习地点	综合实验大楼 BXXXX		

学号	姓名
2022211631	万厚源

实验一

一、实验题目及内容

课后作业（二）3：随机生成一个五列十行的 dataframe 的数据类型，行列索引自定义，绘制出对应的柱状图、散点图，以及在查询官网学习绘制一个课程未讲解（即除柱状图、饼图、散点图、箱线图以外的图形）的数据分析的图形

二、实验过程步骤（注意是主要关键步骤，适当文字+截图说明）、实验结果及分析

(一)关键代码及其描述

1.利用 numpy 库的 random 软件包里的 rand 函数生成一个十行五列的数组，再用 pandas 库的 DataFrame 函数改成 DateFrame 类型，每列分别为 col1，col2，col3，col4，col5，每行分别为为从 1-10.代码见图一：

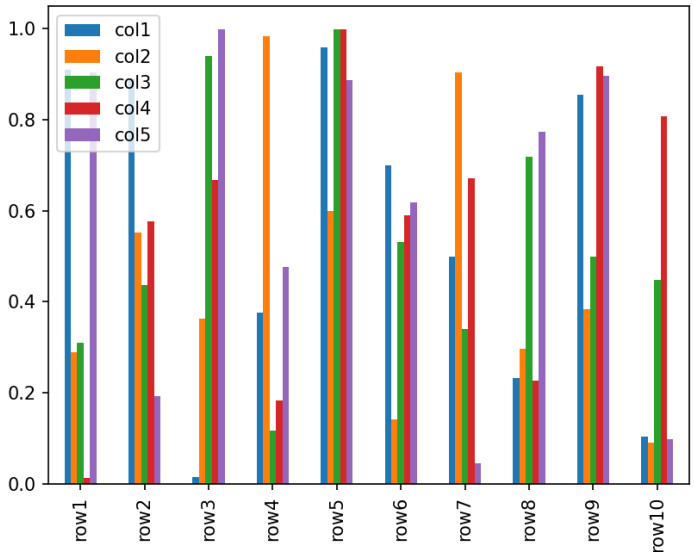
```
# 随机生成一个五列十行的Dataframe
df = pd.DataFrame(np.random.rand(10, 5),
                  columns=['col1', 'col2', 'col3', 'col4', 'col5'], index=['row'+str(i) for i in range(1, 11)])
```

图一.

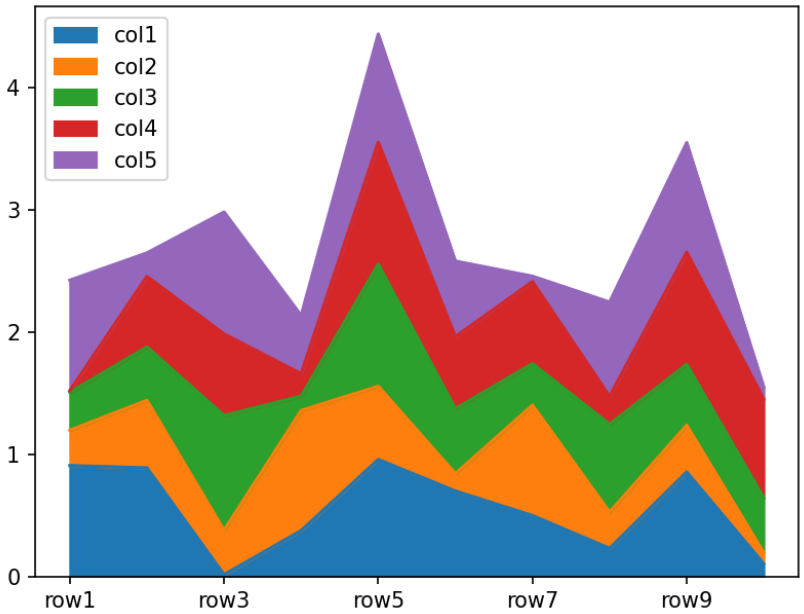
2.利用 pandas 库中 plot 类来划分类别，简单绘图，并用 matplotlib 库里的 pyplot 来展示所绘制画布和图表，关键代码如图二：

```python
# 绘制柱状图
df.plot(kind='bar')
plt.show()

# 绘制散点图
df.plot(kind='scatter', x='col1', y='col2')
plt.show()


#绘制面积图
df.plot(kind='area')
plt.show()

# 绘制除了柱状图、饼图、散点图、箱线图以外的图形的数据分析图形
# 一个常用的数据分析图形是折线图，可以用来展示数据随时间的变化趋势
df.plot(kind='line')
plt.show()
```

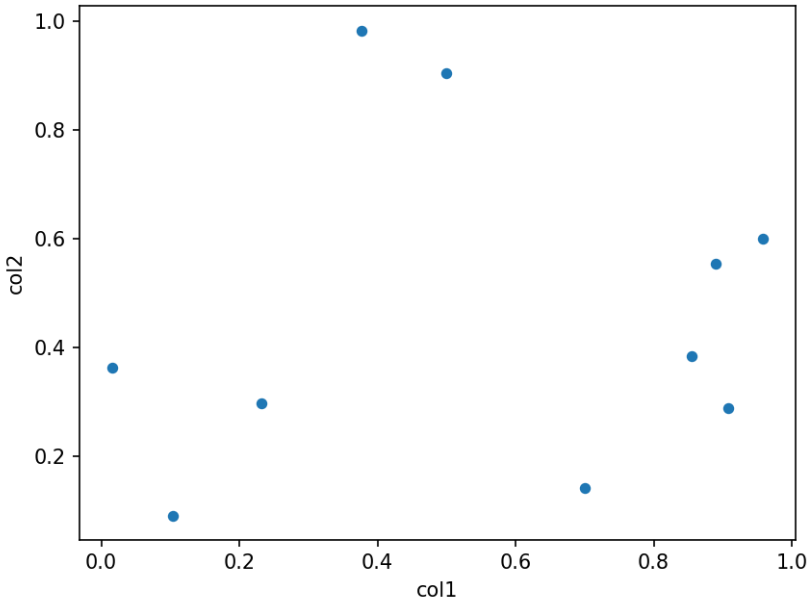
图二、用 plot 类来生成相应的可视化图表



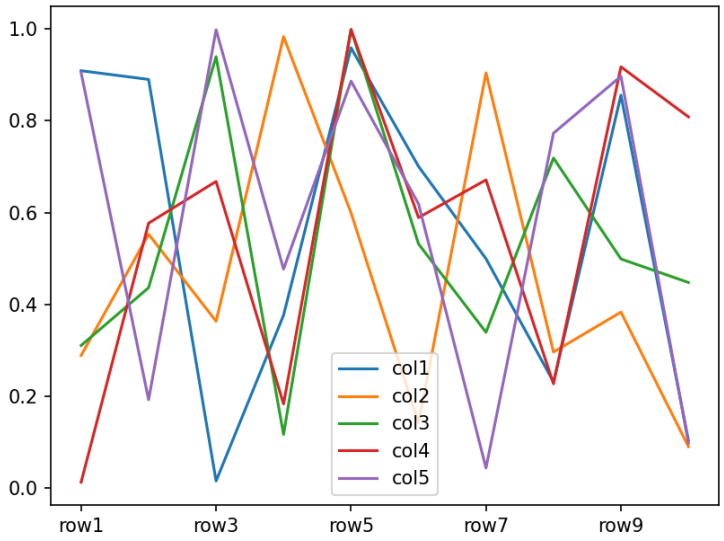
图三、同一组随机数据生成柱状图



图四、同一组随机数据生成面积图



图五、同一组随机数据生成散点图



图六、同一组随机数据生成折线图

（二）实验结果的分析

在 python 中，pyecharts 和 matplotlib 库可绘制较为复杂的代码，而采用 pandas 自带的 plot 类直接对 dataframe 型数据进行简单数据的可视乎

（三）完整代码

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# 随机生成一个五列十行的 Dataframe
df = pd.DataFrame(np.random.rand(10, 5),
                  columns=['col1', 'col2', 'col3', 'col4', 'col5'],
index=['row'+str(i) for i in range(1, 11)])
# 绘制柱状图
df.plot(kind='bar')
```

```
plt.show()
# 绘制散点图
df.plot(kind='scatter', x='col1', y='col2')
plt.show()
#绘制面积图
df.plot(kind='area')
plt.show()
# 绘制除了柱状图、饼图、散点图、箱线图以外的图形的数据分析图形
# 一个常用的数据分析图形是折线图，可以用来展示数据随时间的变化趋势
df.plot(kind='line')
plt.show()
```

实验二

一、实验题目及内容

完成 K-means 算法的代码实现（同时提交源代码）及数据（至少测试 5 个数据集，数据集来源建议采用 UCI 数据集）测试结果

二、实验过程步骤（注意是主要关键步骤，适当文字+截图说明）、实验结果及分析

（一）实验步骤

1.Kmeans 算法介绍以及原理

Kmeans 是经典的和较为简单的基于距离划分的聚类算法。算法的思想史：1.先随机在空间选取 k 个初始簇心，并以这 k 个簇中心为中心来计算各点到其的欧式距离，将每个点归类到距离最近的初始簇中心。2.计算得到簇的维度平均值，依次平均值作为新的簇中心。3.以新的簇中心为中心点，计算各点到其的欧式距离，并将每个点归类到距离最近的初始簇中心。4.依次循环往复迭代，知道不在变化。

2.Kmeans 算法分析

k-means 算法是一种基于距离的聚类算法，它的优点是简单、快速，容易解释和实现。但是，它的缺点也很明显，如对初始值敏感、对噪声和异常值敏感、对簇的形状、大小和密度的假设过于简单等。

3.核心代码编写

（1）计算两点之间的距离，为后续的运算做准备

```
# 定义两点之间的欧式距离
def x_y_distance(arr_x, arr_y):
    dis = arr_x - arr_y
    return np.sqrt(np.sum(np.power(dis, 2)))
```

（2）随机生成 k 个初始的簇中心

```
# 随机生成k个质心
def random_k_centroid(dataset, k):
    data_center = []
    # 在原有列表的基础上选则k个簇心

    for i in range(k):
        data_append = list(dataset.iloc[np.random.randint(0, dataset.shape[0])]
        data_center.append(data_append)
    data_return = np.mat(data_center)
```

（3）kmeans 主要函数：传入参数 data，k，分别表示数据集，和迭代次数，同时也初始化数据归属的列表和初始簇中心。根据距离进行循环迭代，迭代完进行数据划分，最后返回数据的中心和存放数据的列表，一列放索引，一列放距离。

```python
def judgeofthecentor(dataset_dimension,m,n,k):
    centroid = random_k_centroid(dataset_dimension, k)
    # 创建两列，一列放类别，一列放距离
    clusterAssment = np.zeros((m, 2))
    clusterchang = True
    # 判断质心变化，某点是都归属另一个簇了
    while clusterchang:
        clusterchang = False
        for i in range(m):
            min_dist = 100000000
            min_index = -1
            x = centroid[m, :]
            y = dataset_dimension.iloc[i, :].values
            for m in range(k):
                way = x_y_distance(x, y)
                judgeofkmeans(way, min_dist, min_index)

            if clusterAssment[i, 0] != min_index:
                clusterchang = True
                clusterAssment[i, :] = min_index, min_dist ** 2
        for j in range(k):
            put_cluster = dataset_dimension[clusterAssment[:, 0] == j]  # 将属于簇心的j拿
            centroid[j, :] = np.mean(put_cluster, axis=0)
    print("get over,please to continue")
    return centroid, clusterAssment

def KMeans(dataset_dimension, k):
    m = dataset_dimension.shape[0]
    n = dataset_dimension.shape[1]  # 行和列
    centroid,clusterAssment=judgeofthecentor(dataset_dimension,m,n,k)
    return centroid,clusterAssment
```

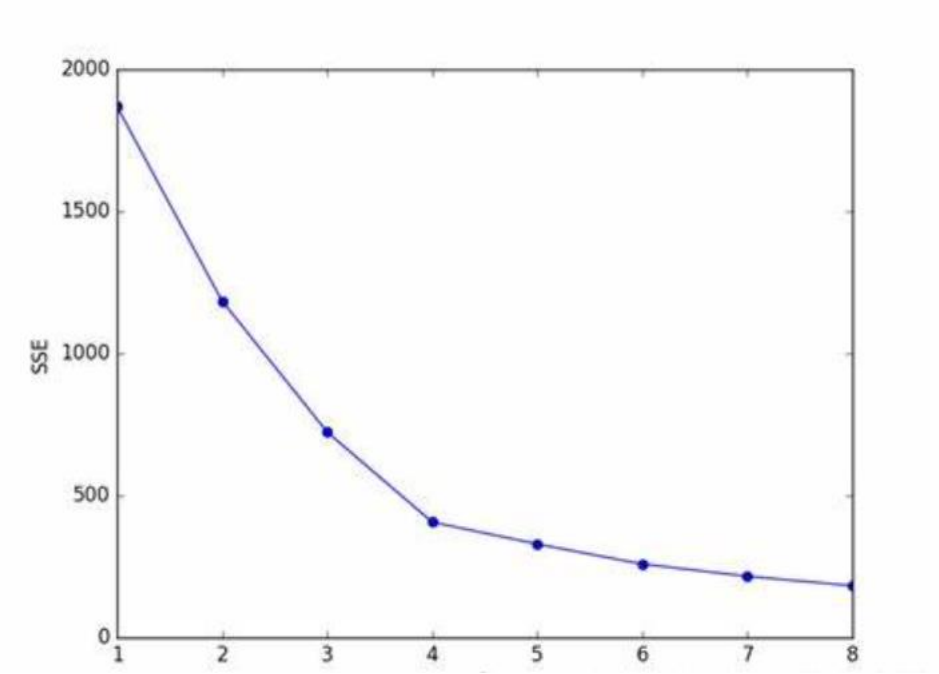
（4）利用手肘法，来找出最好的 k 值，根据具体的图像来选择向下有手肘的值

```python
# 误差平方和sse
def sse(dataset_dimension):
    sse = []
    for m in range(1, 5):
        for i in range(1, 12):
            centroid, result_set = KMeans(dataset_dimension, i)
            sse.append(np.sum(result_set[:, 1]))
            print('k = ' + str(i) + " sse = " + str(sum(result_set[:, 1])))

        plt.plot(range(1, 12), sse, '--o')
        plt.show()
```

4. 实验分析

4.1 手肘法

当选择的 K 值小于真实合适的 K 值时，K 值每增加 1，cost 的值就会大幅度变小，而当 K 值大于真实合适的 K 值时，K 值每增加 1，cost 的值变化很小，如果将 cost 的变化情况绘制成折线图，真实 K 值就会在一个很明显的转折点出现，类似一个肘部，如下图所示



如上图所示，当 K 值取到 4 后，出现了明显的转折点，那 K 值取 4 就比较合适。

手肘法是以成本函数最小化为目标，成本函数为各个类畸变程度之和，每个类的畸变程度为该类簇心与其内部成员位置距离的平方和。

具体 cost 公式为 $D_k = \sum_{i-1}^{n} \sum dist(x, c_i)^2$，其中 x 为第 i 类的样本点，ci为簇中心。

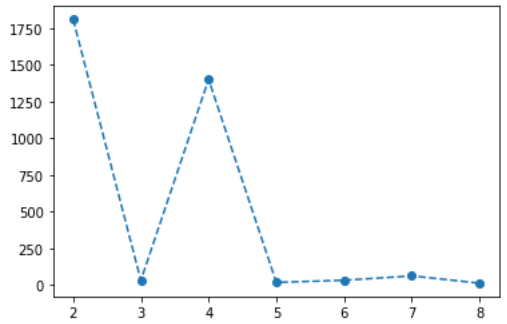
5.数据集测试

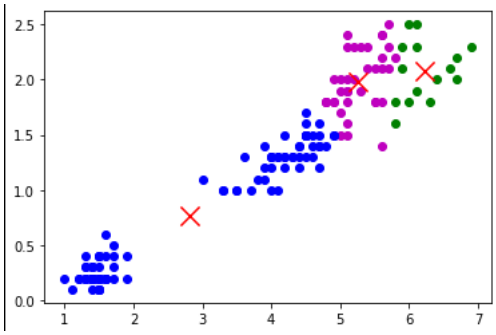
1.鸢尾花数据集（iris）

由于具体的数据不规范，对数据进行初步的压缩处理，因为kmenas

的效果是根据欧式距离来评定的，所以要规范化数据。根据手肘确定k值进行迭代，绘制折线图，通过对数据进行降维，绘制散点聚类情况。如下是代码以及效果图：

```
def pre(data,m,n,v):
    dataset = data.iloc[:, m:n]
    transfer = VarianceThreshold(threshold=v)
    data_dimension = pd.DataFrame(transfer.fit_transform(dataset))
    print(data_dimension)
    # 手肘法来获得合适的k
    sse(data_dimension)
    print_means(data_dimension,3)


#iris 数据集
data = pd.read_csv("D:\桌面\数据集\数据集\iris.csv")
dataset = data.copy()
dataset.drop('label',axis=1,inplace=True)
print(dataset)
```





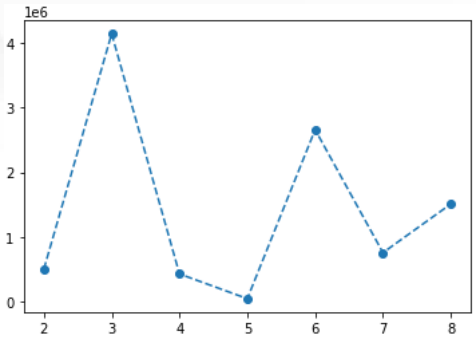
由绘制出来的散点图可以看到，kmenas 的聚类效果还是较为明显的，可以看到集中较为集中。

2. 山火数据集（forestfire）

（1）数据集导入，并设置相应的参数

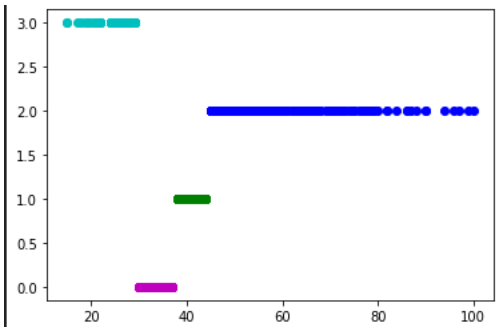
```
#forestfire数据集
data = pd.read_csv("D:\桌面\数据集\数据集\\forestfires.csv")
m=9
n=10
v=0.9
k=5
pre(data,m,n,v)
```

（2）然后用 sse 来进行手肘的绘制和 k 值的确定，



由图可以看出 k 选择 5 比较合适

（3）最后绘制出相应的散点图。



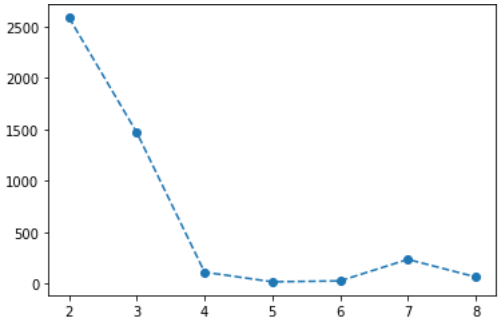
由散点图可以看出数据较为集中。

3. abalone 数据集

（1）导入数据集并设置相应的参数，要进行绘制的相关区域

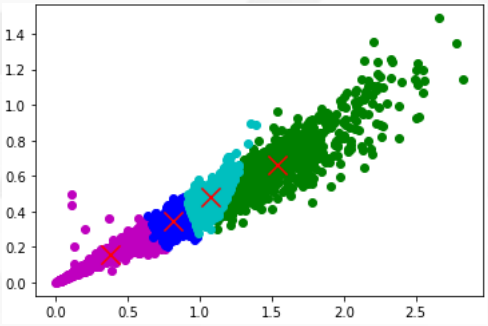
```
#abalone数据集
data =pd.read_csv("D:\桌面\数据集\数据集\\abalone.csv")
m=4
n=8
v=0.03
pre(data,m,n,v)
```

（2）绘制相关 sse 的图像，并考虑合适的 k 值。



由图可知 k 选择 4 比较合适

（3）绘制相关散点图



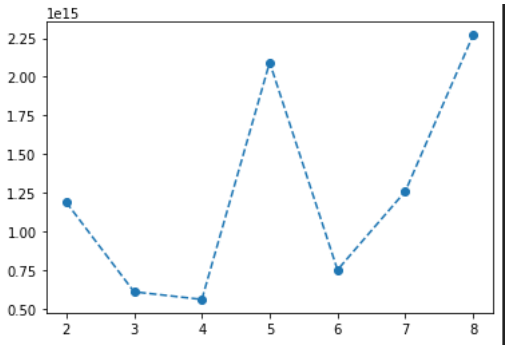
由图像可得，每个簇分类区域较为鲜明，分类效果较好，数据量小，运行时间短

4.顾客数据集（custom）

(1)导入数据，将相关数据集的分类区域进行降维压缩，

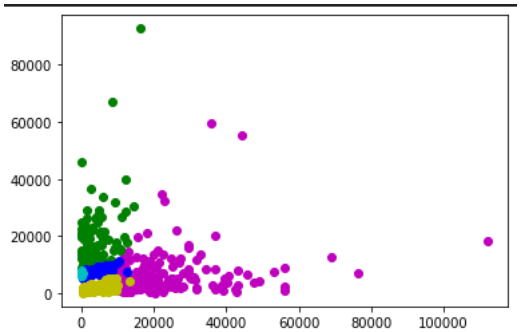
```
#custom数据集
data = pd.read_csv("D:\桌面\数据集\数据集\custom.csv")
m=2
n=7
v=70000000
pre(data,m,n,v)
```

（2）设置手肘，绘制折线图，考虑合适的 k 值



由图像可知，k 最合适为 6

（3）绘制相关散点图



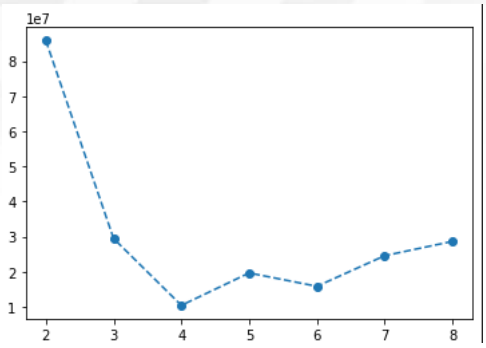
由图像可知，分类效果并不是很好，每个种类差不多都聚集在一起。

5.假期数据集（hoilday）

（1）导入数据，并对相关要绘制的区域进行压缩预处理

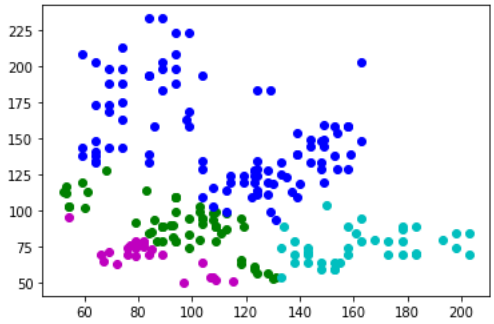
```
data = pd.read_csv("D:\桌面\数据集\数据集\holiday.csv")
m=2
n=7
v=1500
pre(data,m,n,v)
```

（2）绘制 sse 散点图，寻找合适的 k 值



由图像可知合适的 k 值取 4

（3）根据 k 值绘制相关的散点图



由图像可知，分类比较明显，运行时间比较短。

（二）实验分析

在实现 Kmeans 代码及采取手肘法后对五个不同的数据集做了测试，由实验结果得到了 Kmeans 的一系列优缺点。

优点：1.Kmeans代码实现较为简单，只需要公式和算法思想即可，通俗易懂。

2.Kmeans 的聚类效果比较好，能够借助手肘法确定 K 值，且能在有限次迭代内快速收敛，达到局部最优解。

3.针对不同的数据集使用 Kmeans 仅需要调整 K 值即可，操作起来非常方便。

4.当数据分离程度非常大时，Kmeans 能够飞速收敛并得到优秀的聚类效果

缺点：1.Kmeans 的时间复杂度非常高，

2.当数据分离程度不够时，Kmeans 的聚类效果就显得比较一般

3.由于 Kmeans 的聚类效果是由欧式距离来评定的，所以数据量纲的差距就显得非常重要，如果一个特征之间相差较远但该特征量纲较小，在计算欧氏距离的过程中就会被大量纲掩盖。所以在进行 Kmeans 前必须进行数据标准化或归一化。

（三）完整代码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyecharts
import math
import sklearn
from sklearn.feature_selection import VarianceThreshold
# 定义两点之间的欧式距离
def x_y_distance(arr_x, arr_y):
    dis = arr_x - arr_y
    return np.sqrt(np.sum(np.power(dis, 2)))
# 随机生成 k 个质心
def random_k_centroid(dataset, k):
    data_center = []
    # 在原有列表的基础上选则 k 个簇心
    for i in range(k):
```

```
            data_append = list(dataset.iloc[np.random.randint(0, dataset.shape[0])])
            data_center.append(data_append)
        data_return = np.mat(data_center)
        return data_return
def judgeofkmeans(way,min_dist,minin_dex):
        if way < min_dist:
            min_dist = way
            minin_dex = m
def judgeofthecentor(dataset_dimension,m,n,k):
        centroid = random_k_centroid(dataset_dimension, k)
        # 创建两列，一列放类别，一列放距离
        clusterAssment = np.zeros((m, 2))
        clusterchang = True
        # 判断质心变化，某点是都归属另一个簇了
        while clusterchang:
            clusterchang = False
            for i in range(m):
                min_dist = 100000000
                min_index = -1
                x = centroid[m, :]
                y = dataset_dimension.iloc[i, :].values
                for m in range(k):
                    way = x_y_distance(x, y)
```

```
                judgeofkmeans(way, min_dist, min_index)


            if clusterAssment[i, 0] != min_index:
                clusterchang = True
                clusterAssment[i, :] = min_index, min_dist ** 2
        for j in range(k):
            put_cluster = dataset_dimension[clusterAssment[:, 0] == j]  # 将属于簇心的 j 拿出来
            centroid[j, :] = np.mean(put_cluster, axis=0)
    print("get over,please to continue")
    return centroid, clusterAssment


def KMeans(dataset_dimension, k):
    m = dataset_dimension.shape[0]
    n = dataset_dimension.shape[1]  # 行和列
    centroid,clusterAssment=judgeofthecentor(dataset_dimension,m,n,k)
    return centroid,clusterAssment


def preprint(dataset,k,clusterassment):
    dataset['label'] = clusterassment[:, 0]
    color = ['m', 'g', 'b', 'c', 'y']
    for p in range(k):
        x_line = []
```

```
            y_line = []

            for q in range(dataset.shape[0]):

                if (dataset.iloc[q, dataset.shape[1] - 1] == p):
                    x_line.append(dataset.iloc[q, 0])
                    y_line.append(dataset.iloc[q, 1])
            plt.scatter(x_line, y_line, c=color[p])

def print_means(dataset, k):
    centroids, clusterassment = KMeans(dataset, k)
    preprint(dataset,k,clusterassment)
    plt.scatter(list(centroids[:, 0]), list(centroids[:, 1]), color='red', marker='*',
s=200)
    plt.show()

# 误差平方和 sse
def sse(dataset_dimension):
    sse = []
    for m in range(1, 5):
        for i in range(1, 12):
            centroid, result_set = KMeans(dataset_dimension, i)
            sse.append(np.sum(result_set[:, 1]))
```

```
            print('k = ' + str(i) + " sse = " + str(sum(result_set[:, 1])))


        plt.plot(range(1, 12), sse, '--o')
        plt.show()


# 模型收敛稳定性
def stability(dataset, k):
    np.random.seed(123)
    for i in range(1, 5):
        plt.subplot(2, 2, i)
        centroid, cluster = KMeans(dataset, k)
        plt.scatter(cluster.iloc[:, 0], cluster.iloc[:, 1], c=cluster.iloc[:, -1], )
        plt.plot(centroid[:, 0], centroid[:, 1], 'o', color='red')
        print(cluster.iloc[:, 3].sum())



# 初始质心的随机选取会影响聚类的最终结果
# 质心数量在某种程度上影响随机的程度，如果质心数量选取和数据空间分布相似，那么对最终聚类的结果影响比较小
# 解决方案，在设置质心随机生成的过程中，尽可能使质心更加分散，或者手动初始质心，降低随机性影响，挥着增量更新质心
# 在点到簇的每次指派后，增量地更新质心，而不是在所有的点都指派到簇
```

```
之后再更新，每步需要零次或者两次质心的更新，移到别的簇或者留到当前簇
# 使用增量更新确保不会产生空簇，所有簇都从单点开始，并且如果一个簇只有一个单点，那么总是被放到相同的簇
# 此外，如果使用增量更新，可以调整点的相对权重，点的权值随聚类进行而减小，可能产生更好的准确率和更快的收敛性
# 数据预处理
def pre(data, m, n, v):
    dataset = data.iloc[:, m:n]
    transfer = VarianceThreshold(threshold=v)
    data_dimension = pd.DataFrame(transfer.fit_transform(dataset))
    print(data_dimension)
    # 手肘法来获得合适的 k
    sse(data_dimension)
    print_means(data_dimension, 3)



# iris 数据集
data = pd.read_csv(r'D:\桌面\数据集\数据集\iris.csv')
dataset = data.copy()
dataset.drop('label', axis=1, inplace=True)
print(dataset)
```

```
m = 1

n = 4

v = 0.4

pre(data, m, n, v)


# forestfire 数据集

data = pd.read_csv("D:\桌面\数据集\数据集\forestfires.csv")

m = 9

n = 10

v = 0.8

pre(data, m, n, v)


# abalone 数据集

data = pd.read_csv("D:\桌面\数据集\数据集\abalone.csv")

m = 4

n = 8

v = 0.03

pre(data, m, n, v)


# custom 数据集

data = pd.read_csv("D:\桌面\数据集\数据集\custom.csv")

m = 2

n = 7
```

```
v = 70000000
pre(data, m, n, v)


# hoilday


data = pd.read_csv("D:\桌面\数据集\数据集\holiday.csv")
m = 2
n = 7
v = 1500
pre(data, m, n, v)
```

实验三

一、实验题目及内容

课后作业（五）2：对测试集 ccf_offline_stage1_test_revised 做分析与数据观察

二、实验过程步骤（注意是主要关键步骤，适当文字+截图说明）、实验结果及分析

（一）数据观察

1、首先进行数据的观察，分析特征的不同，依次为用户名，商家 id，优惠券 id，折扣率，同时折扣率分为满减和直接折扣，用户居住地与店家的距离、收到优惠券的日期。

2、知晓数据的详细情况

```
record_count 113640
received_count 113640
coupon_count 2050
user_count 76309
merchant_count 1559
min_received 20160701
max_received 20160731
```

图一、数据大体情况

由图可知，记录较为完善，没有缺陷项，各个指标合理，时间段处于 2016/7/1 到 2016/7/31，所以可以对 7 月份整个月的时间划分为周，并进而进行分析。

（二）数据分析，绘制各种可视化图表，观察特征数据详细分布情况

① 分析情况，绘制可视化图表，观察特征数据情况

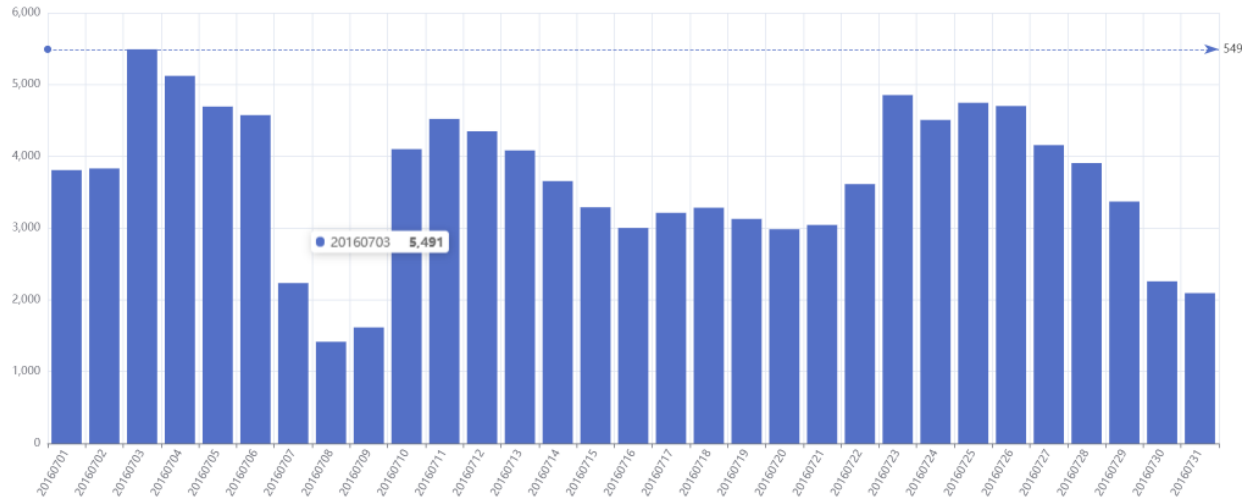
（1）用户每条领券的数量

先将每日领券数量提取出来，借助 pandas 中的 groupby 函数得到各个日期领券的具体情况，代码如下：

```
#选取领券日期不为空的数据
df_1 = offline[offline['Date_received'].notna()]
# 以Date_received为分组目标并统计优惠券的数量
tmp = df_1.groupby('Date_received', as_index=False)['Coupon_id'].count()
```

图二、主要代码

借助 pyecharts 库绘制对应的柱状图如下：



图三、处理效果图

由上图可以清楚的看到，领券数量在不同的时间有着巨大的差异，在月初的领取数量比月中和月底较少，可能是工资发放或者月初活动对消费产生影响，因此可以对不同的时段的数据分别进行分析，将每日的数据集整合，更改为一周的数据集可能会更有普适性。

也便于对于整个月进行深入调查。

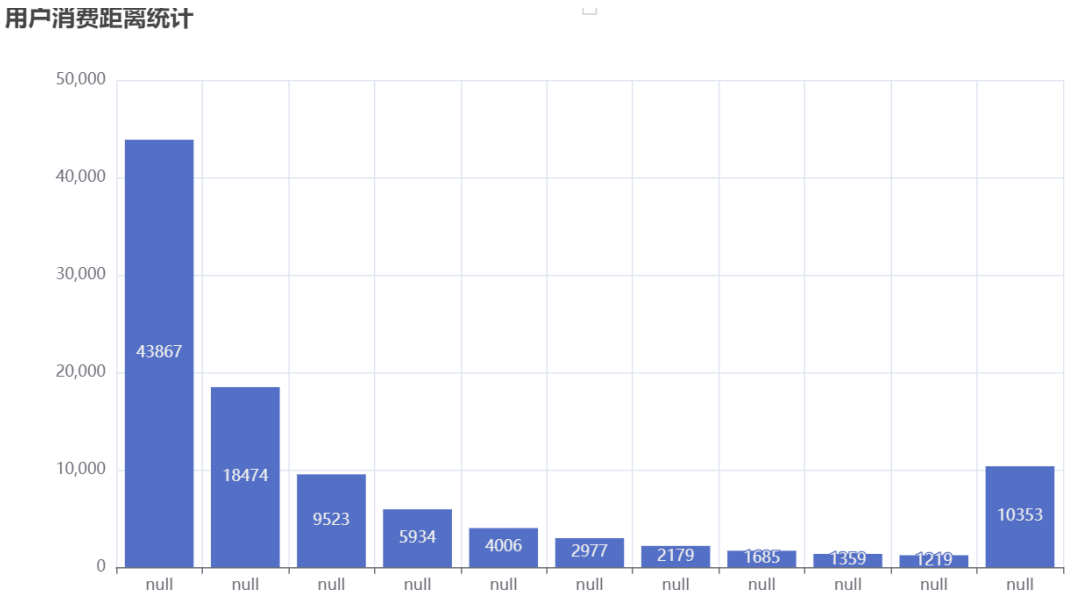
（1） 不同消费距离的优惠券数量

将数据按照消费距离分组并进行统计，且按照距离的远近进行排序，代码如下：

```
# 统计各类距离的消费次数
import collections
dis = offline[offline['Distance']!=-1]['Distance'].values
dis = dict(collections.Counter(dis))

x = list(dis.keys())
y = list(dis.values())
```

图四、不同消费优惠券数量统计

将所得的数据绘制成柱状图来进行分析：



图五、不同消费距离优惠券数量统计柱状图

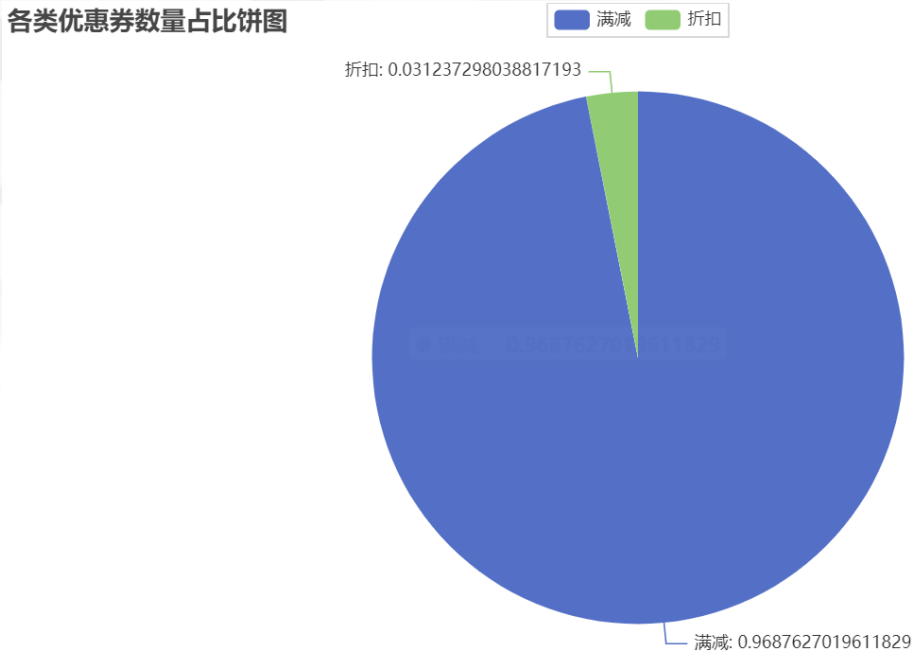
由上图可以看到，大多数人还是更关注离自己家比较近的商家，用户的领取商家优惠券的数量随着距离的递增二递减，但根据图表可知，在最后出现了反常情况，有少数的用户领取离自己家很远的商家的优惠券，可以看到商家的影响力和口碑。在后续的数据预处理中，可以将 7km 之后的用户进行打标，这些表示用户对商家有好的回头率，15 内使用优惠券的可能性比较大。

（2） 各类优惠券数量占比

将不同折扣类型的优惠券数据进行转换和提取后得到图形：

```
v1 = ['折扣','满减']
v2 = list(offline[offline['Date_received'].notna()]['is_manjian'].value_counts(True))
print(v2)
```

图六、相关代码

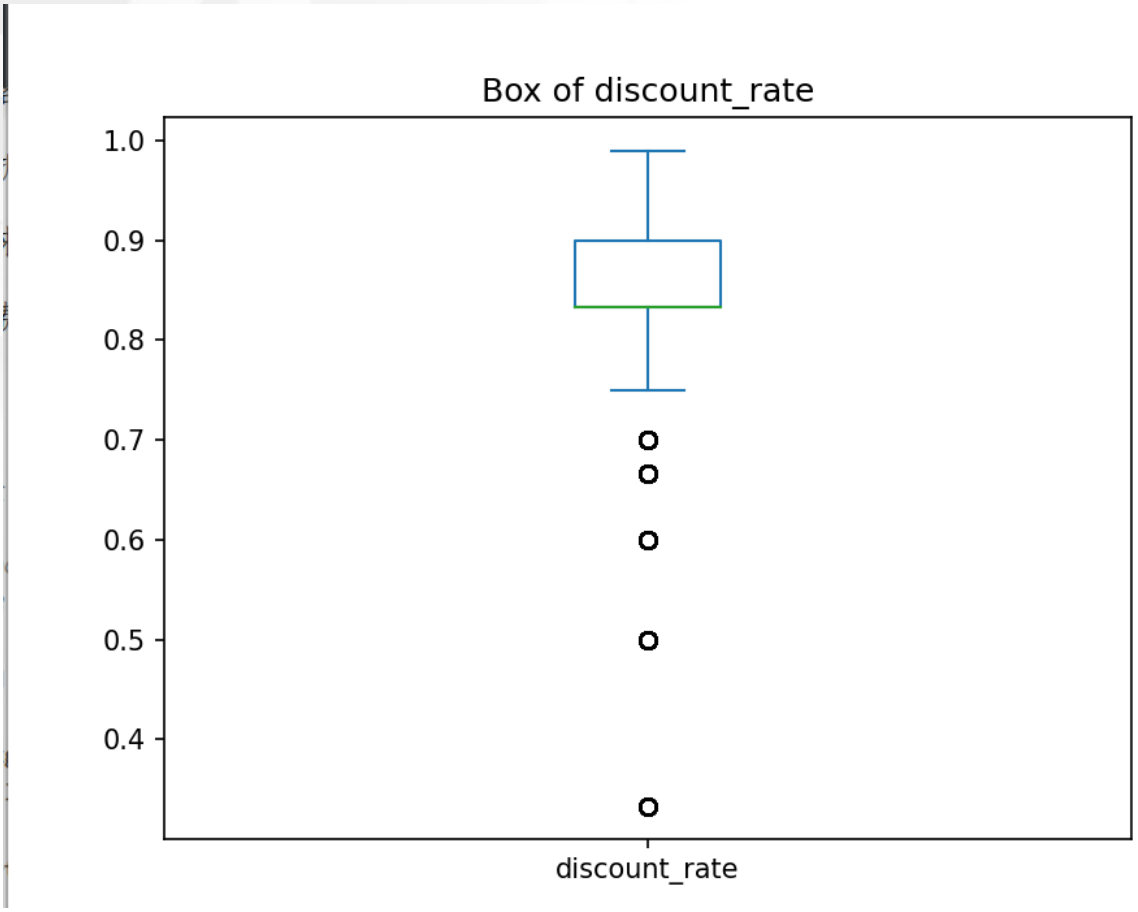


图十、各类优惠券数量占比饼图

由图十可以得知，满减类型的优惠券数量远多于直接折扣优惠券，但是直接折扣的优惠券，没有使用的指定额度，更受欢迎，所以在后续的处理中，可以加入使用券不同类型的权重，满减型消费券最低限制，也影响用户对优惠券的评价，如果太高则不符合市场和受众的实际需要。

（3） 折扣率整体的箱线图来检查异常值

提取相关特征绘制箱线图，以折扣率为特征判断数据集是否存在异常值，箱线图如下：



图十一、折扣率分布情况箱线图

由图可知，优惠券主要集中在七五折和九折之间，但小部分商家会尝试发放大额优惠券，因此会有群点，这是用户对优惠券的重要反响。

（三）总结

1.通过数据的观察，可以得知整日的数据比较合适，消费的距离存在缺失值，需要进行删除或拟合来处理缺失值。

2.通过数据的可视化分析，可以看出不同的特征存在重大差异，所以要对数据进行进一步处理：①将数据日期格式转化为 datatime 类，同时打标，标记出日期具体为那一天，周几等等 ②将满减和折扣类型的优惠券转换成折扣率，将满减的最低消费提取出来 ③标记数据是否为满减类型

（四）完整代码

```
import pandas as pd

import os

import pyecharts

from pyecharts.charts import Bar
```

```
from pyecharts import options as opts
from pyecharts.charts import Pie
from matplotlib import pyplot as plt
data = pd.read_csv("D:\桌面\o2o 实验练习\ccf_offline_stage1_test_revised.csv")
#共有多少条记录
record_count = data.shape[0]     #统计有多少记录
#共有多少条优惠券的领取记录
received_count = data['Date_received'].count()   #统计不含 nan 值的数量
#共有多少种不同的优惠券
coupon_count = len(data['Coupon_id'].value_counts())
#共有多少个用户
user_count = len(data['User_id'].value_counts())
#共有多少个商家
merchant_count = len(data['Merchant_id'].value_counts())
#最早领券时间
min_received = str(int(data['Date_received'].min()))
#最晚领券时间
max_received = str(int(data['Date_received'].max()))
print('record_count', record_count)
print('received_count', received_count)
print('coupon_count', coupon_count)
print('user_count', user_count)
print('merchant_count', merchant_count)
```

```
print('min_received', min_received)
print('max_received', max_received)


offline = data.copy()
offline['Distance'].fillna(-1,downcast= 'infer',inplace = True)#填充
#转换为时间类型
offline['date_received'] = pd.to_datetime(offline['Date_received'],format = '%Y%m%d')
#将折扣券转化为折扣率
offline['discount_rate'] = offline['Discount_rate'].map(lambda x: float(x) if ':' not in str(x) else
(float(str(x).split(':')[0]) - float(str(x).split(':')[1])) / float(str(x).split(':')[0]))
#为数据打标  因为没有消费时间所以打标不能进行
#显示在 15 天之内消费的人群，1 为消费了，0 为为消费
#offline['label'] = list(map(lambda x : 1 if x / (60 *60 *24) <= 15 else 0,offline['date_received']))
#添加优惠券是不是满减类型
offline['is_manjian'] = offline['Date_received'].map(lambda x :1 if ':' in str(x) else 0)
#领券时间为周几
offline['weekday_received'] = offline['date_received'].apply(lambda x: x.isoweekday())
#为 offline 数据添加领券时间
```

```
offline['received_month'] = offline['date_received'].apply(lambda x : x.month)
#减满最低消费
offline['min_cost_of_manjian'] = offline ['Discount_rate'].map(lambda x:1 if ':' not
in str(x) else int(str(x).split(':')[0]))


# path = "./tem/table1_output"
# if not os.path.exists(path) :
#         os.makedirs(path)
# offline.to_csv(path+'/table1_ouput.csv',index = False)
#
# #选取领券日期不为空的数据
# df_1 = offline[offline['Date_received'].notna()]
# # 以 Date_received 为分组目标并统计优惠券的数量
# tmp = df_1.groupby('Date_received', as_index=False)['Coupon_id'].count()
# # 建立柱状图
# bar_1 = (
#         Bar(
#                 init_opts = opts.InitOpts(width='1600px', height='600px')
#         )
#         .add_xaxis(list(tmp['Date_received']))
#         .add_yaxis('', list(tmp['Coupon_id']))
#         .set_global_opts(
#                 title_opts = opts.TitleOpts(title='每天被领券的数量'), # title
```

```
#             legend_opts = opts.LegendOpts(is_show=True), # 显示 ToolBox
#                                                          xaxis_opts =
opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate=60), interval=1), # 旋转 60
度
#         )
#         .set_series_opts(
#             opts.LabelOpts(is_show=False), # 显示值大小
#             markline_opts = opts.MarkLineOpts(
#                 data = [
#                     opts.MarkLineItem(type_='max', name='最大值')
#                 ]
#             )
#         )
# )
## 统计各类距离的消费次数
# import collections
# dis = offline[offline['Distance']!=-1]['Distance'].values
# dis.sort()
# dis = dict(collections.Counter(dis))
#
# x = list(dis.keys())
# y = list(dis.values())
## 建立柱状图
```

```
# bar_2 = (
#         Bar()
#         .add_xaxis(x)
#         .add_yaxis('', y)
#         .set_global_opts(
#                 title_opts=opts.TitleOpts(title='用户消费距离统计'), # title
#         )
# )
# path = './tem/bar_output'
# if not os.path.exists(path):
#         os.makedirs(path)
# bar_1.render(path+'/bar_1.html')
# bar_2.render(path+'/bar_2.html')
# #各种折扣率的优惠券数量
# received = offline[['discount_rate']]
# received['cnt']=1
# received = received.groupby('discount_rate').agg('sum').reset_index()
#各类优惠券数量占比
v1 = ['满减','折扣']
v2 = list(offline[offline['date_received'].notna()]['is_manjian'].value_counts(True))


pie_1 = (
      Pie()
```

```
        .add('',[list(v) for v in zip(v1,v2)])
        .set_global_opts(title_opts = {'text': '各类优惠券占比饼图'})
        .set_series_opts(label_opts = opts.LabelOpts(formatter='{b}:{c}'))
)
path = './tem/pie_output'
if not os.path.exists(path):
                os.makedirs(path)
pie_1.render(path+'Pie_1.html')
#观察近中远三个距离下多大的折扣力度才能吸引用户领券
#提取非空距离
df_1 = offline[offline['Distance'] !=-1]
#提取指定
df_1 = df_1.loc[:,['Distance','discount_rate']]
#分组统计各个距离下的各种优惠券领取情况，0，5，10
df_1 = df_1.groupby([df_1['Distance'],df_1['discount_rate']]).agg('size')
#提取对应距离对应折扣率的统计个数
cnt = [dict(df_1[i]) for i in range(11)]
idx = list(df_1[0].keys())


#将折扣率保存为三维小数
idx = [float('%.3f'%x) for x in idx]
dis = [0, 5, 10]
num = []
```

```
#将对应统计个数按相同长度存到一个 list 里
for i in dis:
    tmp = []
    for x in idx :
        if(x in cnt[i]):
            tmp.append(int(cnt[i][x]))
        else:
            tmp.append(int(0))
    num.append(tmp)
for x in range(3):
    pie = (
        Pie()
        .add('',[list(v) for v in zip(idx,num[x])])
        .set_global_opts(title_opts = opts.TitleOpts(title ='距离为%d 的优惠券折扣率占比饼图.html' %dis[x],pos_top='90%'))
    )
pie.render('距离为%d 的优惠券折扣率占比饼图.html' %dis[x])



offline['discount_rate'].plot.box(title = 'Box of discount_rate')
plt.show()
```

实验四

一、实验题目及内容

课后作业（五）3：对测试集 ccf_offline_stage1_test_revised 做数据预处理

二、实验过程步骤（注意是主要关键步骤，适当文字+截图说明）、实验结果及分析

（一） 实验步骤

1.将空缺的距离列进行填充（图一）

```
offline = data.copy()

offline['Distance'].fillna(-1,downcast='infer',inplace=True)#填充
```

图一、填充距离列里面的空缺值

2.数据日期格式转化进一步对日期抽取新的特征（图二）

```
#转换为时间类型
offline['date_received'] = pd.to_datetime(offline['Date_received'],format='%Y%m%d')
```

图二、日期类型转化

3.添加领券的周和月份（图三）

```
#领券时间为周几
offline['weekday_received'] = offline['date_received'].apply(lambda x: x.isoweekday())
#为offline数据添加领券时间
offline['received_month'] = offline['date_received'].apply(lambda x: x.month)
```

图三、领券的时间提取

4.折扣率转化（图四）

```
#将折扣券转化为折扣率
offline['discount_rate'] = offline['Discount_rate'].map(lambda x: float(x) if ':' not in str(x) else
(float(str(x).split(':')[0]) - float(str(x).split(':')[1])) / float(str(x).split(':')[0]))
```

图四、折扣率转化

5.满减类型判断和满减最低消费转化(图五)

```
#添加优惠券是不是满减类型
offline['is_manjian'] = offline['Date_received'].map(lambda x :1 if ':' in str(x)
else 0)
#减满最低消费
offline['min_cost_of_manjian'] = offline ['Discount_rate'].map(lambda x:1 if ':'
not in str(x) else int(str(x).split(':')[0]))
```

图五、满减以及最低消费转化

（二）实验结结果以及分析

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	User_id	Merchant_	Coupon_ic	Discount_r	Distance	Date_recei	date_recei	discount_r	is_manjian	weekday_r	received_r	min_cost_of_manjian	
2	4129537	450	9983	30:05:00	1	20160712	########	0.833333	0	2	7	30	
3	6949378	1300	3429	30:05:00	-1	20160706	2016/7/6	0.833333	0	3	7	30	
4	2166529	7113	6928	200:20:00	5	20160727	########	0.9	0	3	7	200	
5	2166529	7113	1808	100:10:00	5	20160727	########	0.9	0	3	7	100	
6	6172162	7605	6500	30:01:00	2	20160708	2016/7/8	0.966667	0	5	7	30	
7	4005121	450	9983	30:05:00	0	20160706	2016/7/6	0.833333	0	3	7	30	
8	4347394	450	9983	30:05:00	0	20160716	########	0.833333	0	6	7	30	
9	3094273	760	13602	30:05:00	1	20160727	########	0.833333	0	3	7	30	
10	5139970	450	9983	30:05:00	10	20160729	########	0.833333	0	5	7	30	
11	3237121	760	13602	30:05:00	1	20160703	2016/7/3	0.833333	0	7	7	30	
12	6224386	450	9983	30:05:00	3	20160716	########	0.833333	0	6	7	30	
13	6488578	760	13602	30:05:00	0	20160712	########	0.833333	0	2	7	30	
14	4164865	450	9983	30:05:00	2	20160703	2016/7/3	0.833333	0	7	7	30	
15	4164865	5138	8059	50:10:00	1	20160706	2016/7/6	0.8	0	3	7	50	
16	5468674	450	9983	30:05:00	-1	20160713	########	0.833333	0	3	7	30	
17	6258178	7024	9144	10:01	0	20160706	2016/7/6	0.9	0	3	7	10	
18	3659521	6000	7341	150:30:00	6	20160727	########	0.8	0	3	7	150	
19	3659521	2436	13181	20:01	3	20160717	########	0.95	0	7	7	20	
20	3659521	760	13602	30:05:00	0	20160718	########	0.833333	0	1	7	30	
21	7333378	760	13602	30:05:00	0	20160704	2016/7/4	0.833333	0	1	7	30	
22	7333378	6901	785	20:05	1	20160727	########	0.75	0	3	7	20	
23	4454914	3621	2978	20:05	0	20160711	########	0.75	0	1	7	20	
24	6817282	6362	8375	100:20:00	6	20160724	########	0.8	0	7	7	100	
25	3149569	5717	10418	30:05:00	0	20160721	########	0.833333	0	4	7	30	
26	6301186	8318	1715	30:01:00	1	20160718	########	0.966667	0	1	7	30	
27	6301186	2199	4203	300:50:00	1	20160708	2016/7/8	0.833333	0	5	7	300	
28	2891521	760	13602	30:05:00	0	20160724	########	0.833333	0	7	7	30	
29	3422977	760	13602	30:05:00	0	20160727	########	0.833333	0	3	7	30	
30	4771330	760	13602	30:05:00	7	20160726	########	0.833333	0	2	7	30	
31	4513282	1469	11799	100:10:00	7	20160712	########	0.9	0	2	7	100	
32	3439873	760	13602	30:05:00	0	20160713	########	0.833333	0	3	7	30	
33	6632962	760	13602	30:05:00	0	20160727	########	0.833333	0	3	7	30	

对数据进行预处理之后，将缺失值进行填充，考虑到后续建模的进行以及准确率，对数据进行更进一步的提取。将日期数据进行转化，提取特征周，将折扣率进行转换，并标记是否为满减类型以及满减最低消费，对后续建模起到铺垫作用。

（三）完整代码

```
import pandas as pd
import os
import pyecharts
from pyecharts.charts import Bar
from pyecharts import options as opts
from pyecharts.charts import Pie
from matplotlib import pyplot as plt
data = pd.read_csv("D:\桌面\o2o 实验练习\ccf_offline_stage1_test_revised.csv")
#共有多少条记录
record_count = data.shape[0]     #统计有多少记录
#共有多少条优惠券的领取记录
received_count = data['Date_received'].count()   #统计不含 nan 值的数量
#共有多少种不同的优惠券
coupon_count = len(data['Coupon_id'].value_counts())
#共有多少个用户
user_count = len(data['User_id'].value_counts())
#共有多少个商家
merchant_count = len(data['Merchant_id'].value_counts())
#最早领券时间
min_received = str(int(data['Date_received'].min()))
#最晚领券时间
max_received = str(int(data['Date_received'].max()))
print('record_count', record_count)
```

```
print('received_count', received_count)
print('coupon_count', coupon_count)
print('user_count', user_count)
print('merchant_count', merchant_count)
print('min_received', min_received)
print('max_received', max_received)


offline = data.copy()
offline['Distance'].fillna(-1,downcast= 'infer',inplace = True)#填充
#转换为时间类型
offline['date_received'] = pd.to_datetime(offline['Date_received'],format = '%Y%m%d')
#将折扣券转化为折扣率
offline['discount_rate'] = offline['Discount_rate'].map(lambda x: float(x) if ':' not in str(x) else
(float(str(x).split(':')[0]) - float(str(x).split(':')[1])) / float(str(x).split(':')[0]))
#为数据打标  因为没有消费时间所以打标不能进行
#显示在 15 天之内消费的人群，1 为消费了，0 为为消费
#offline['label'] = list(map(lambda x : 1 if x / (60 *60 *24) <= 15 else 0,offline['date_received']))
#添加优惠券是不是满减类型
offline['is_manjian'] = offline['Date_received'].map(lambda x :1 if ':' in str(x) else 0)
```

```
#减满最低消费
offline['min_cost_of_manjian'] = offline ['Discount_rate'].map(lambda x:1 if ':' not in str(x) else int(str(x).split(':')[0]))
#领券时间为周几
offline['weekday_received'] = offline['date_received'].apply(lambda x: x.isoweekday())
#为 offline 数据添加领券时间
offline['received_month'] = offline['date_received'].apply(lambda x : x.month)
# path = "./tem/table1_output"
# if not os.path.exists(path) :
#         os.makedirs(path)
# offline.to_csv(path+'/table1_ouput.csv',index = False)
```

实验五

一、实验题目及内容

课后作业（八）: 提取下列用户特征,完成模型训练，并在阿里云天池平台提交结果：1、领券数、2、领券并消费数、3、领券未消费数、4、 领券并消费数 / 领券数、5、领取并消费优惠券的平均折扣率、6、领取并消费优惠券的平均距离、7、在多少不同商家领取并消费优惠券、8、在多少不同商家领取优惠券、9、在多少不同商家领取并消费优惠券 / 在多少不同商家领取优惠券

二、实验过程步骤（注意是主要关键步骤，适当文字+截图说明）、实验结果及分析

（一）实验过程

1.逐一提取用户特征群特征：

先加入 cnt 列方便计数，并提取 label 数据集中主键 User_id，以此来把 label 数据集中出现过的用户的历史消费行为数据从 history_field 中提取出来，设置主表为u_feat,代码见图一。

```
data['cnt'] = 1

# 主键
keys = ['User_id']
# 特征名前缀,由history_field和主键组成
prefixs = 'history_field_' + '_'.join(keys) + '_'
# 返回的特征数据集
u_feat = label_field[keys].drop_duplicates(keep='first')
```

图一、主键提取及数据表准备

1.用户领券数

借助 pivot 数据透视表函数，以 User_id 为主键提取用户领卷数量，aggfunc 参数设置为 len，这样就能统计一个 User_id 有多少条不同的数据记录，并把 pivot 转换成 DataFrame 型且重设列名和索引，再用 merge 函数与主表 u_feat 合并。由于可能有用户是在 label_field 的时间窗口内第一次出现，所以会提取不到他的历史数据，故在最后将缺失值填充为 0。

```
# 用户领券数
# 以keys为键,'cnt'为值,使用len统计出现的次数
pivot = pd.pivot_table(data, index=keys, values='cnt', aggfunc=len)
# pivot_table后keys会成为index,统计出的特征列会以values即'cnt'命名,将其改名为特征名前缀+特征意义,并将index还原
pivot = pd.DataFrame(pivot).rename(columns={'cnt': prefixs + 'receive_cnt'}).reset_index()
# 将id列与特征列左连
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
# 缺失值填充为0,最好加上参数downcast='infer',不然可能会改变DataFrame某些列中元素的类型
u_feat.fillna(0, downcast='infer', inplace=True)
```

图二、用户领卷数

2.用户领卷并消费数

在提取 history_field 的数据时，对数据项添加限制，只提取有消费记录的数据项，即数据项的 Date 列不为空的数据项，再重复上述过程，即可统计用户领卷并消费数量。

```
# 用户领券并消费数
# 先筛选出Date非空的样本,以keys为键,'cnt'为值,使用len统计出现的次数
pivot = pd.pivot_table(data[data['Date'].map(lambda x: str(x) != 'nan')], index=keys, values='cnt',
                       aggfunc=len)  #筛选非空样本，并用len来统计长度
# pivot_table后keys会成为index,统计出的特征列会以values即'cnt'命名,将其改名为特征名前缀+特征意义,并将index还原
pivot = pd.DataFrame(pivot).rename(columns={'cnt': prefixs + 'receive_and_consume_cnt'}).reset_index() #没有index
# 将id列与特征列左连
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
# 缺失值填充为0,最好加上参数downcast='infer',不然可能会改变DataFrame某些列中元素的类型
u_feat.fillna(0, downcast='infer', inplace=True)
```

图三、用户领卷并消费数

3.用户领卷未消费数

与上述操作相同，将限制条件换为数据的 Date 列为空，即可统计用户领卷未消费数量。

```
# 用户领券未消费数
# 先筛选出Date为空即未核销的样本,以keys为键,'cnt'为值,使用len统计出现的次数
pivot = pd.pivot_table(data[data['Date'].map(lambda x: str(x) == 'nan')], index=keys, values='cnt',
                       aggfunc=len)    #筛选样本，并用len来统计所筛选样本的长度
# pivot_table后keys会成为index,统计出的特征列会以values即'cnt'命名,将其改名为特征名前缀+特征意义,并将index还原
pivot = pd.DataFrame(pivot).rename(
    columns={'cnt': prefixs + 'receive_not_consume_cnt'}).reset_index()  #列变量表示的数据
# 将id列与特征列左连
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
# 缺失值填充为0,最好加上参数downcast='infer',不然可能会改变DataFrame某些列中元素的类型
u_feat.fillna(0, downcast='infer', inplace=True)
```

图四、用户领卷未消费数

4.用户领卷并消费数/领卷数(用户核销率)

通过已经提取出的用户领卷并消费数及用户领卷数来计算用户核销率,设用户领卷并消费数为 x，用户领卷数为 y，则核销率 T = x / y，但由于label数据集中可能存在用户在label_field的时间窗口内是第一次出现，并没有历史数据可以采集，就要特殊讨论用户领卷数为 0 的情况，为防止浮点数溢出，在 y = 0 时将 T 设为

0 即可。

```
# 用户的消费率除以领券数的比率
#名称前缀加上特征意义
u_feat[prefixs + 'receive_and_consume_rate'] = list(
    map(lambda x, y: x / y if y != 0 else 0,
        u_feat[prefixs + 'receive_and_consume_cnt'], u_feat[prefixs + 'receive_cnt']))#接受并消费的数量和领券的数量
```

图五、用户核销率

5.领取并消费优惠卷的平均折扣率

添加限制条件为数据项的 Date 列非空，将统计量换为discount_rate列，aggfunc 参数设置为取平均值的 mean。即可提取用户领取并消费优惠卷的平均折扣率。

```
#领取并消费优惠卷的平均折扣率
pivot = pd.pivot_table(data[data['Date'].map(lambda x:str(x) != 'nan')],
                       index=keys, values='discount_rate',aggfunc='mean')
pivot = pd.DataFrame(pivot).rename(columns={'discount_rate':prefixs +
                                            'receive_and_consume_mean_discount_rate'}).reset_index()
u_feat = pd.merge(u_feat,pivot,on=keys,how='left')
u_feat.fillna(0, downcast='infer', inplace=True)
```

图六、领取并消费优惠卷的平均折扣率

6.领取并消费优惠卷的平均距离

首先添加限制条件为数据项的 Date 列非空，提取成临时数据表Data_tmp，再添加第二个限制条件，即数据项的Distance 列不为-1，在数据预处理过程中将消费距离为空的数据填充为了-1，故在此处计算平均距离的时候也要过滤掉-1 的数据。然后利用 pivot 函数提取平均距离即可。

```
#领取并消费优惠卷的平均距离
tmp = data[data['Date'].map(lambda x:str(x) !='nan')]
pivot = pd.pivot_table(tmp[tmp['Distance'].map(lambda x: int(x) != -1)],index=keys,
                       values='Distance',aggfunc="mean")
pivot = pd.DataFrame(pivot).rename(columns={'Distance': prefixs +
                                            'receive_and_consume_mean_distance'}).reset_index()
u_feat = pd.merge(u_feat,pivot,on=keys,how='left')
u_feat.fillna(0,downcast='infer',inplace=True)
```

图七、领取并消费优惠卷的平均距离

7.在多少不同商家领取并消费优惠卷

添加限制条件为数据项的Date列非空，同时统计量更改为Merchant_id，由于是要求不同商家的记录，所以aggfunc函数设置为len(set(x))，因为set作为stl容器可以将插入的数据项自动去重，将该用户所有领卷商家id插入set后，只要输出该set的长度就可以得到有多少不同的商家记录。

```
# 用户对领券商家的15天内的券并且消费的数量
# 先筛选出label为1即领券15天内核销的样本,以keys为键,'Merchant_id'为值,使用len统计去重后的商家出现的次数
pivot = pd.pivot_table(data[data['label'] == 1], index=keys, values='Merchant_id', aggfunc=lambda x: len(set(x)))
# pivot_table后keys会成为index,统计出的特征列会以values即'cnt'命名,将其改名为特征名前缀+特征意义,并将index还原
pivot = pd.DataFrame(pivot).rename(columns={'Merchant_id': prefixs +
                                                          'receive_and_consume_differ_Merchant_cnt_15'}).reset_index()
# 将id列与特征列左连
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
# 缺失值填充为0,最好加上参数downcast='infer',不然可能会改变DataFrame某些列中元素的类型
u_feat.fillna(0, downcast='infer', inplace=True)
return u_feat
```

图八、在多少不同商家领取并消费优惠卷

8.在多少不同商家领取优惠卷

不添加限制条件，直接重复上述操作，即可得到在多少不同商家领取优惠卷的数量。

```
# 用户领取了多少个不同商家的优惠券
# 以keys为键,'Merchant_id'为值,使用len统计去重后的商家出现的次数
pivot = pd.pivot_table(data, index=keys, values='Merchant_id', aggfunc=lambda x: len(set(x)))
# pivot_table后keys会成为index,统计出的特征列会以values即'Merchant_id'命名,将其改名为特征名前缀+特征意义,并将index还原
pivot = pd.DataFrame(pivot).rename(columns={'Merchant_id': prefixs + 'receive_differ_Merchant_cnt'}).reset_index()
# 将id列与特征列左连
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
# 缺失值填充为0,最好加上参数downcast='infer',不然可能会改变DataFrame某些列中元素的类型
u_feat.fillna(0, downcast='infer', inplace=True)
```

图九、在多少不同商家领取优惠卷

9.在多少不同商家领取并消费优惠卷 / 在多少不同商家领取优惠卷(不同商家核销率)

同样对第一次在label_field数据集中出现的用户做特殊处理，

利用〇7和〇8提取出来的数据直接计算。

```
# 商家的券消费的数量除以领取的数量的比率
m_feat[prefixs + 'received_and_consumed_rate'] = list(map(lambda x, y: x / y if y != 0 else 0,
                                m_feat[prefixs + 'received_and_consumed_cnt'], m_feat[prefixs + 'received_cnt']))
```

图十、在多少不同商家领取优惠卷核销率

2. 整合数据表并返回

将整理出来的主表与原先的 label_field 合并，因为整理出来的主表仅是各个用户的 9 项特征，并没有原先给出的一系列特征，故要将提取出的主表 u_feat 按照 User_id 为主键，填充到对应的每一条 User_id 数据项后面。最后整合为 history_feat 返回。

```
def get_history_field_feature(label_field, history_field):  #历史数据区间
    # 用户特征
    u_feat = get_history_field_user_feature(label_field, history_field)
    # 添加特征
    history_feat = label_field.copy()
    # 添加用户特征
    history_feat = pd.merge(history_feat, u_feat, on=['User_id'], how='left')
    # 返回
    return history_feat
```

图十一、整合数据表并返回

3.整理数据集

将得到的 history—feat 整理并重新格式化为 xgboost 所需的数据格式，修正数据类型为 int，float 等的数据类型

（1）构造数据集，删除共有特征列的基础特征。删除无用的属性，并将 label 置于最后一列。

针对不同的数据集，history_dataset 和 label_dataset 做分类讨论，当为训练集和验证集时，清除基础特征并将 label 置于最后一列。

```
history_feat = get_history_field_feature(label_field, history_field)
# 构造数据集
# 共有属性,包括id和一些基础特征,为每个特征块的交集
share_characters = list(set(label_field.columns.tolist()) & set(history_feat.columns.tolist()))
label_field.index = range(len(label_field))  #设置索引从1依次到长度
dataset = pd.concat([label_field, history_feat.drop(share_characters, axis=1)], axis=1)#去掉共同的部分之后，将打

# 删除无用属性并将label置于最后一列
if 'Date' in dataset.columns.tolist():  # 表示训练集和验证集    #差异
    # 删除无用属性
    dataset.drop(['Merchant_id', 'Discount_rate', 'Date', 'date_received', 'date'], axis=1, inplace=True)
    label = dataset['label'].tolist()
    dataset.drop(['label'], axis=1, inplace=True)#删除掉原始label，用现在的进行替换
    dataset['label'] = label
else:
    # 表示测试集
    dataset.drop(['Merchant_id', 'Discount_rate', 'date_received'], axis=1, inplace=True)
```

图十二、构造数据集

(2)数据修正

修改数据类型便于xgboost训练

```
# 修正数据类型
dataset['User_id'] = dataset['User_id'].map(int)
dataset['Coupon_id'] = dataset['Coupon_id'].map(int)
dataset['Date_received'] = dataset['Date_received'].map(int)
dataset['Distance'] = dataset['Distance'].map(int)
if 'label' in dataset.columns.tolist():
    dataset['label'] = dataset['label'].map(int)
```

图十三、修正数据类型

(3)去重并进行重置索引

```
# 去重
dataset.drop_duplicates(keep='first', inplace=True)
# 这里一定要重置index,若不重置index会导致pd.concat出现问题
dataset.index = range(len(dataset))
```

图十四、去重并重置索引

4. 进行 xgboost 训练

进行线上训练，即将训练集和验证集整合为大数据集，传入训练函数中进行训练。将训练次数改为250次。

将传入的训练集和测试集修正为 XGBoost 训练需要的 DMatrix

矩阵。训练集剥夺标签，并在建矩阵时传入标签列。

```
# 数据集
#设置XGBoost训练所需的DMatrix矩阵
dtrain = xgb.DMatrix(train.drop(['User_id', 'Coupon_id', 'Date_received', 'label'], axis=1), label=train['label'])
dtest = xgb.DMatrix(test.drop(['User_id', 'Coupon_id', 'Date_received'], axis=1))
```

图十七、构造训练矩阵

进行 XGBoost 训练，并输出预测结果。

```
# 训练
watchlist = [(dtrain, 'train')]
model = xgb.train(params, dtrain, num_boost_round=250, evals=watchlist)
# 预测
predict = model.predict(dtest,validate_features=False)
# 处理结果
predict = pd.DataFrame(predict, columns=['prob'])  # 增添列为prob
result = pd.concat([test[['User_id', 'Coupon_id', 'Date_received']], predict], axis=1)  #axis为1，表示列
result.to_csv("D:\桌面\o2o实验练习//tem//result.csv")
```

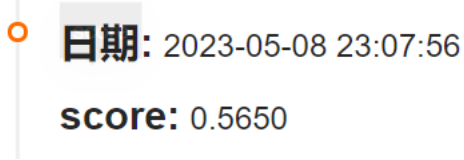
图十八、训练并预测

提取重要性特征矩阵并返回

```
# 特征重要性
feat_importance = pd.DataFrame(columns=['feature_name', 'importance'])
feat_importance['feature_name'] = model.get_score().keys()
feat_importance['importance'] = model.get_score().values()
feat_importance.sort_values(['importance'], ascending=False, inplace=True)
# 返回
return result, feat_importance
```

（二）实验结果分析

在提取了用户特征群 9 个参数之后，模型效果不太理想，可能是提取的特征数量过少，不具有概括性，不能有效表达用户使用优惠券的可能性，对整个实际的情况不具有整体泛化性。如下是提交到天池平台的结果：



显然分数是比较低的，因此在添加有关特征之后，同时也进行调整 xgboost 整体的有关参数。最后的到的效果有了提升

日期: 2023-05-09 14:27:22

score: 0.7376

因此，在后续的重构特征工程中要细致思考如何构建特征工程，提取不同的类别的特征，做出交叉特征群，注意在预测域的特征情况，同时在模型的考虑上，xgboost 模型本身也存在重大缺陷，可以考虑用其他模型，并对不同的模型进行整合。也可以对参数进行调整，达到更好的效果。

任务一总结（心得体会）

通过对 python 项目的实践，回顾了 python 的使用语法和相应的 numpy、pandas、matplotlib 等库的使用方式，熟悉了数据挖掘的相关流程，虽然在学习中遇到很多不同的问题，通过查阅相关博客和资料，解决了部分的问题，但是仍然有很多的问题仍然感到困惑，对相关模型的训练，也没有达到很好的效果。

在这段学习的经历中，对机器学习有了深入的了解，对于相关的模型，仅会调整参数，是远远不够的，要去了解机器学习的原理，理解其底层逻辑，才能够在实际的运用中，发挥更好的效果。不同的算法发挥不同的作用，通过对实际应用场景的分析，选择合适的机器学习的模型，达到良好的效果。

同时机器学习的各个算法模型是相辅相成的，算法和模型都是为了解决具体的实际的问题，做好一个项目，设计一套好的流程，分析相关的特征关联性，以及项目的目的，充分利用每一条的数据和特征，构建良好的框架。

我认为，最重要的是对数据的深层认识，搜索数据，挖掘数据，积累相关的经验，对数据有敏感度，查阅相关资料，合理分析。当然自己也有很多的不足，后续将会继续通过学习纠正。