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：XXXXXX

学生姓名：陈浩如

联系电话：19823324153

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	
*****		*****	

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

(1)关键代码及叙述

1.利用 numpy 库的 random 包内的 rand 函数生成一个十行五列的数组，再由 pandas 库的 DataFrame 函数封装成 DataFrame 类型，且指定列名依次为“abcde”， 行名依次为 0-9，关键代码见图一。

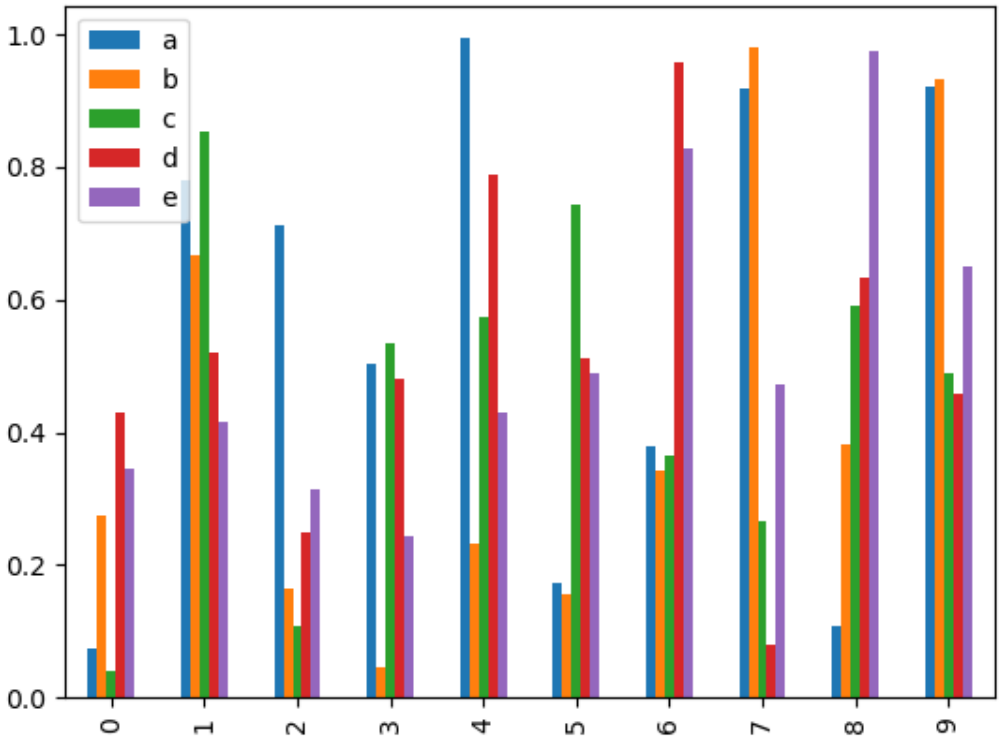
```
##用循环生成对应行列名
df = pd.DataFrame(np.random.rand(10, 5), columns=[i for i in "abcde"],
                  index=[chr(i + ord('0')) for i in range(10)]
                  )
```

图一、随机生成十行五列的 DataFrame

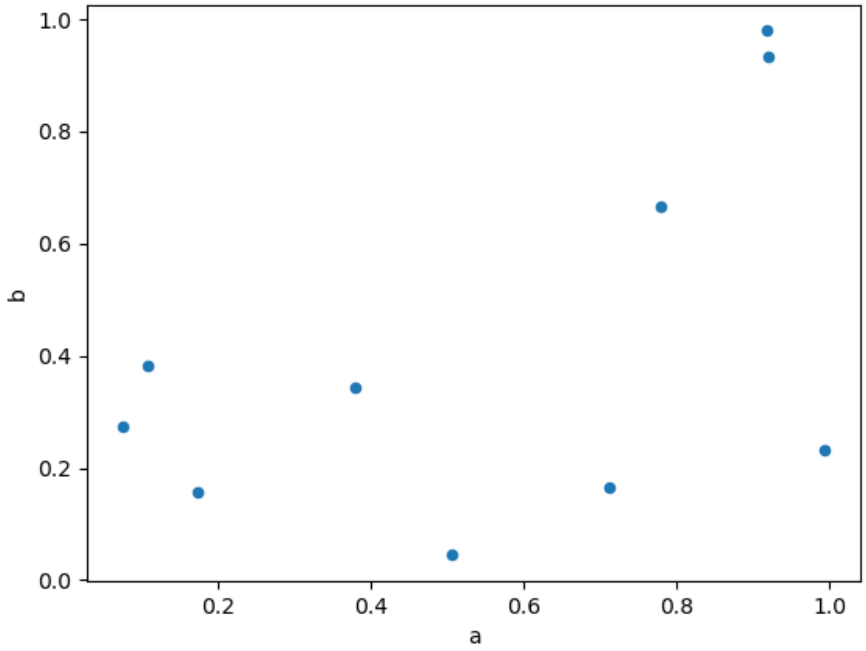
2.利用 pandas 库中 Plot 类进行简单地绘图，并使用 matplotlib 库里的 pyplot 类展示所绘画布和图表，关键代码见图二，生成的图像见图三至图七。

```
#柱状图
df.plot.bar()
#散点图
df.plot.scatter(x = 'a', y = 'b')
#直方图，开启堆叠
df.plot.hist(stacked = True, bins = 20, alpha = 0.7)
#面积图
df.plot.area()
#密度图
df.plot.kde()
plt.show()
```

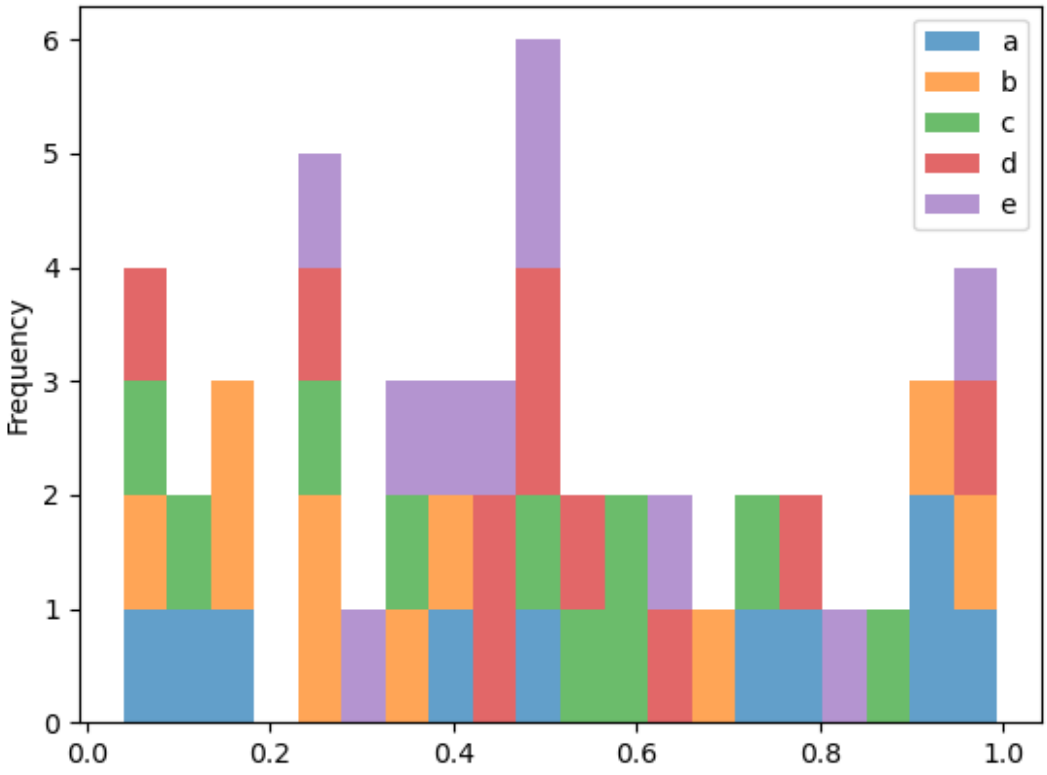
图二、利用 Plot 类生成相应的可视化图表



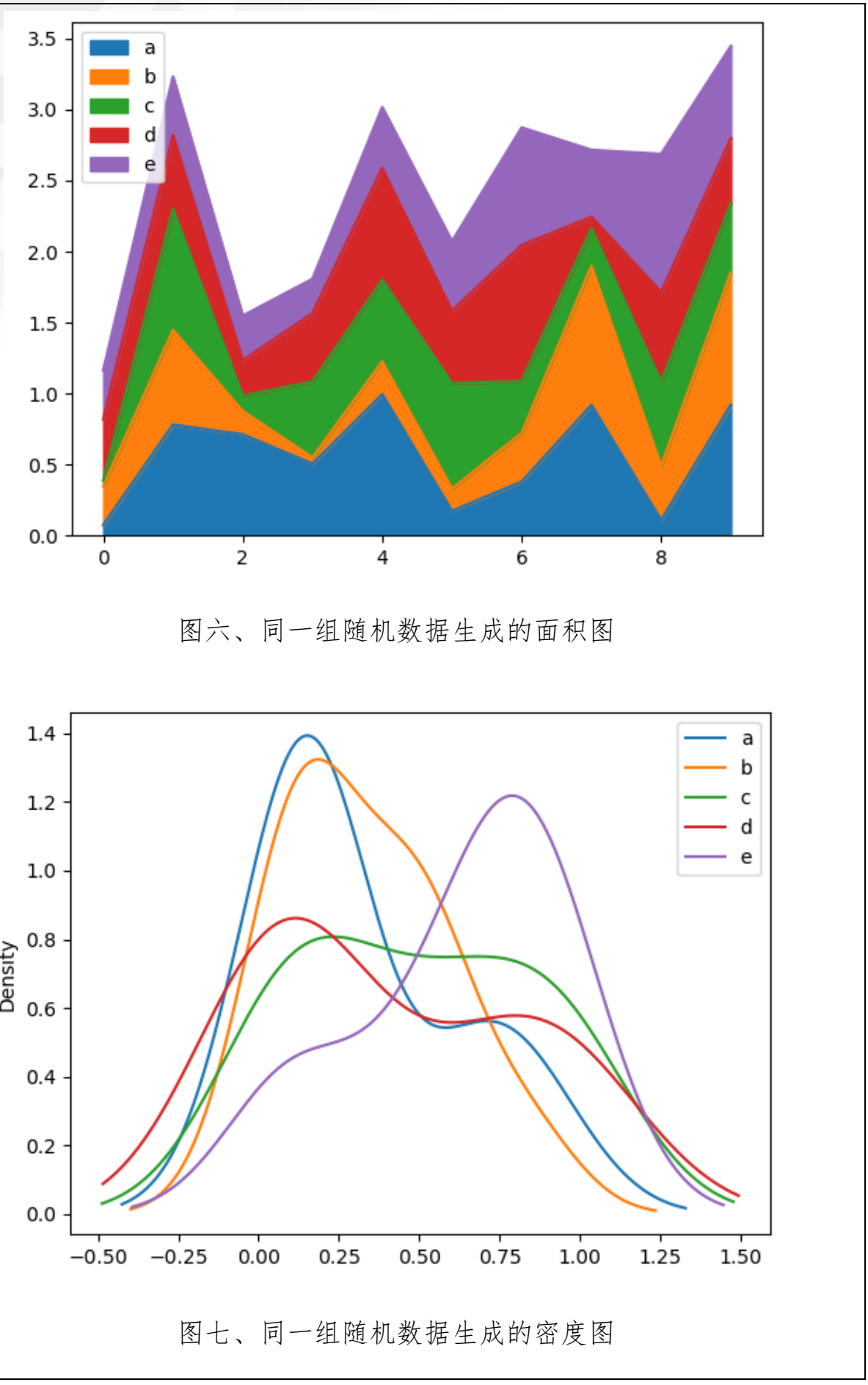
图三、同一组随机数据生成的柱状图



图四、同一组随机数据生成的散点图



图五、同一组随机数据生成的直方图，且开启堆叠



图六、同一组随机数据生成的面积图

图七、同一组随机数据生成的密度图

(2)实验结果与分析

当所绘制数据及所需求图像没有特别复杂时，可以避免使用 pyecharts 库和 matplotlib 库的复杂函数及冗长代码，采用 pandas 自带的 Plot 类直接对 DataFrame 型数据进行简单数据可视化图像绘制。

(3)完整实验代码

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
import xgboost as xgb
import lightgbm as lgb
from matplotlib import pyplot as plt

if __name__ == '__main__':
    ##用循环生成对应行列名
    df = pd.DataFrame(np.random.rand(10, 5), columns = [i for i in "abcde"],
                      index = [chr(i + ord('0')) for i in range(10)]
                     )
    #柱状图
    df.plot.bar()
    #散点图
    df.plot.scatter(x = 'a', y = 'b')
    #直方图，开启堆叠
    df.plot.hist(stacked = True, bins = 20, alpha = 0.7)
    #面积图
    df.plot.area()
    #密度图
    df.plot.kde()
    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

(1) 实验步骤

1. Kmeans 算法介绍及原理

Kmeans 是最经典也是最简单的基于划分的聚类方法，是十大经典数据挖掘算法之一。Kmeans 算法的算法思想：1. 首先随机选取多维空间中的 K 个初始簇中心，以该 K 个簇中心为中心计算各点到其的欧氏距离，将每个点归类到距离其最近的初始簇中心。2. 计算各簇内部各维度的平均值，以此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，将每个点归类到距离其最近的初始簇中心。4. 不断迭代重复步骤 2-3，直到各点归属簇不再变化，或者达到迭代上限。

2. Kmeans 算法分析：

Kmeans 的优点：实现简单，原理简单，聚类效果较好。Kmeans 的缺点：时间复杂度较大，K 值需要自行确定，且 K 个点一开始的随机分配情况也影响聚类效果。Kmeans 对孤立点和“噪声”非常敏感，不适用于发现非凸面形状的簇或者大小差别很大的簇。

3. KMeans 核心代码编写

(1) 传入参数设为 Kmeans(data, k, nums)，分别表示数据集(data)，K 值(k)，迭代上限次数(nums)，并初始化标识数据归属的列表(belong)和初

始簇中心(center)，初始簇中心从数据集中抽样，代码见图一。

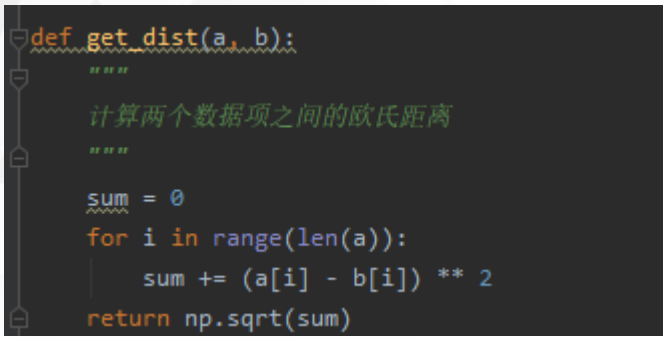
```
def Kmeans(data, k, nums):
    # 标注每个数据属于哪个集合
    belong = [i for i in range(len(data))]
    # 初始簇中心采用从数据中抽样
    center = data.sample(k, axis = 0)
```

图一、Kmeans 初始化

(2)设置 nums 次循环迭代，在每次迭代中提取 new_belong 表示该次迭代完每个数据归属于哪个簇，迭代终止条件(1.达到迭代上限 nums 次；2.归属簇不再变化，即 new_belong == belong)，计算簇离簇中心的欧式距离用 get_dist 函数(代码见图三)，整体 Kmeans 代码见图二。

```
# 迭代终止条件1，迭代到迭代上限次
for t in range(nums):
    new_belong = [i for i in range(len(data))]
    # 计算每项数据到任一簇中心的欧式距离，并更新为最近的簇中心
    for i in range(len(data)):
        # 哨兵确定最小值
        dist = get_dist(data.iloc[i], center.iloc[0])
        idx = 0
        # 枚举簇中心
        for j in range(1, k):
            # 计算数据距离簇中心的欧氏距离
            dist1 = get_dist(data.iloc[i], center.iloc[j])
            # 如果距离该簇更近，就更新为新的簇中心
            if dist1 < dist:
                dist = dist1
                idx = j
        new_belong[i] = idx
    # 更新新的簇中心各维数据
    for i in range(k):
        for j in range(len(center.iloc[0])):
            sum, cnt = 0, 0
            for s in range(len(data)):
                if new_belong[s] == i:
                    sum += data.iloc[s][j]
                    cnt += 1
            # 如果没有属于他的，就重新抽样
            if(cnt == 0):
                center.iloc[i] = data.sample(1, axis = 0).values
            center.iloc[i][j] = sum / cnt
    # 迭代终止条件2，聚类稳定
    if(new_belong == belong):
        break
    belong = new_belong
```

图二、Kmeans 迭代



```
def get_dist(a, b):
    """
    计算两个数据项之间的欧氏距离
    """
    sum = 0
    for i in range(len(a)):
        sum += (a[i] - b[i]) ** 2
    return np.sqrt(sum)
```

图三、计算两个数据项之间的欧氏距离

(3)返回聚类标签(belong)和结束迭代后的簇中心(center)

4.实验分析

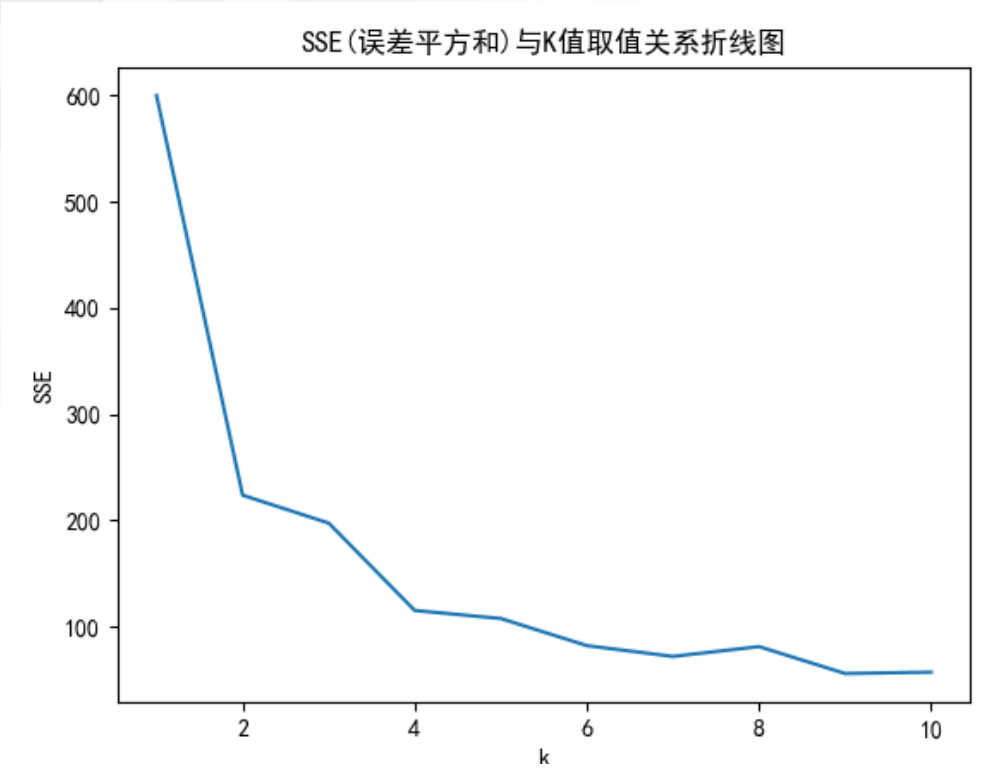
在 Kmeans 实现中，由于 K 值一开始的选取是非常重要的，不同的 K 值选取严重影响 Kmeans 的聚类效果，在此介绍下述三种确定 K 值的方法。

4.1 拍脑袋法

拍脑袋法是一个快速确定 K 值的方法，但准确性难以预计，在大多数情况下能取到较佳的效果，具体取值公式为$K \approx \sqrt{n/2}$,K 值=样本量除以 2 再开平方。

4.2 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。



图四、手肘法确定 K 值示例图

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。

4.3 轮廓系数

轮廓系数会衡量对象和所属簇之间的相似度，轮廓系数在越接近 1 聚类效果越好，越接近-1 聚类效果越差。

（1）计算样本到其他同类的其他样本的平均距离 a，平均距离 a 越小，

说明样本越应被聚类到该类，将这个平均距离 a 称为样本的簇内不相似度。

（2）计算样本到其他某类的所有样本的平均距离 b，平均距离为样本去其他类的簇间不相似度。

（3）根据样本的簇内不相似度 a 和簇内不相似度 b 定义样本的轮廓系数为 $s(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$，又可详细化为 $s(i)=\begin{cases}1-\frac{a(i)}{b(i)}, & a(i)<b(i)\\ 0, & a(i)=b(i)\\ \frac{b(i)}{a(i)}-1, & a(i)>b(i)\end{cases}$。

（4）当 s 近似于 1 时，说明该样本聚类合理；当 s 近似于-1 时，说明该样本更应该分到其他类，当 s 近似于 0 时，说明该样本在两个簇边界上。

5. 数据集测试：

①鸢尾花数据集(Iris)：

(1)读入数据并初步处理，因为 Kmeans 的聚类效果是由欧氏距离来评定的，所以要消除数据量纲上的区别，做数据标准化，代码见图五。

```
def Kmeans_Iris():
    data = pd.read_csv('iris.data', names=['sepal_len', 'sepal_wid', 'petal_len', 'petal_wid', 'class'])
    data.index = range(len(data))
    # 剥离所属类别
    label = data['class']
    data = data.drop(['class'], axis=1)
    # 数据标准化处理
    SS_x = StandardScaler().fit(data)
    data = pd.DataFrame(SS_x.transform(data))
```

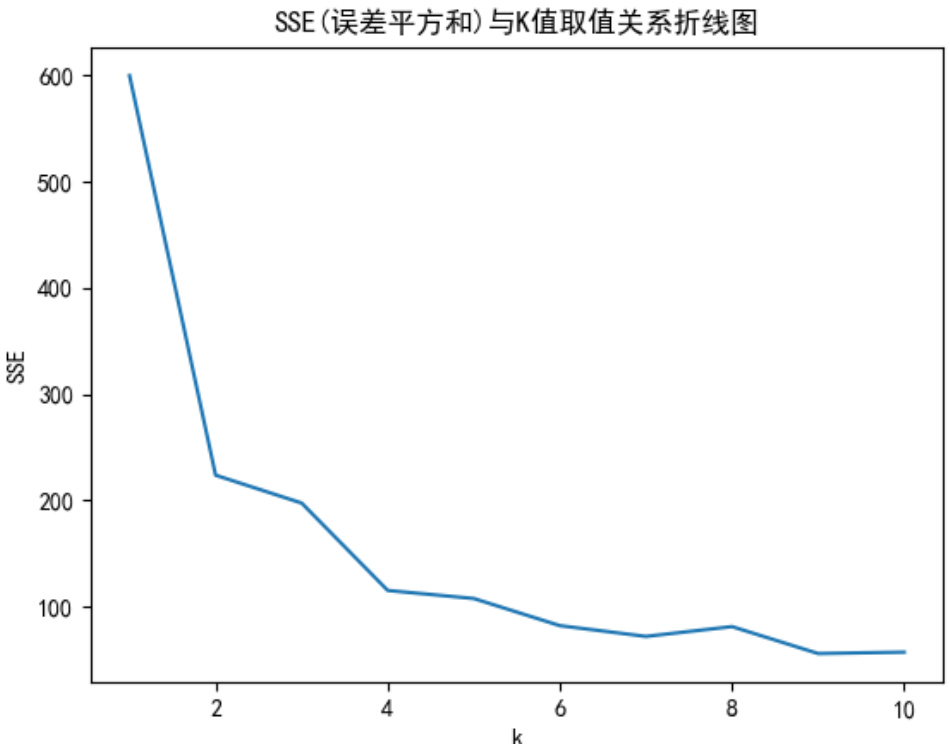
图五、鸢尾花数据集读入并预处理

(2)根据手肘法确定 Kmeans 的 K 值，设置 K 值上限为 n，并迭代处理查看聚类效果，得到不同 K 值情况下的 SSE(误差平方和)，并绘制折线图，代

码见图六。

```
# 手肘法做Kmeans
n = 10  # 手肘法K值上限
SSE = []# 误差平方和
for i in range(1, n + 1):      # 枚举k值为[1, 10]
    print('第%d次KMeans中'%i)
    belong, center = Kmeans(data, i, 100)
    sum = 0
    for j in range(i):
        for k in range(len(data)):
            if belong[k] == j:
                sum += get_SSE(data.iloc[k], center.iloc[j])
    SSE.append(sum)
x = range(1, n + 1)
# 绘制SSE与K值取值折线图
plt.xlabel('k')
plt.ylabel('SSE')
plt.title('SSE(误差平方和)与K值取值关系折线图')
plt.plot(x, SSE)
plt.show()
```

图六、手肘法确定 K 值并绘制 SSE 曲线图



图七、SSE(误差平方和)与 K 值取值关系折线图

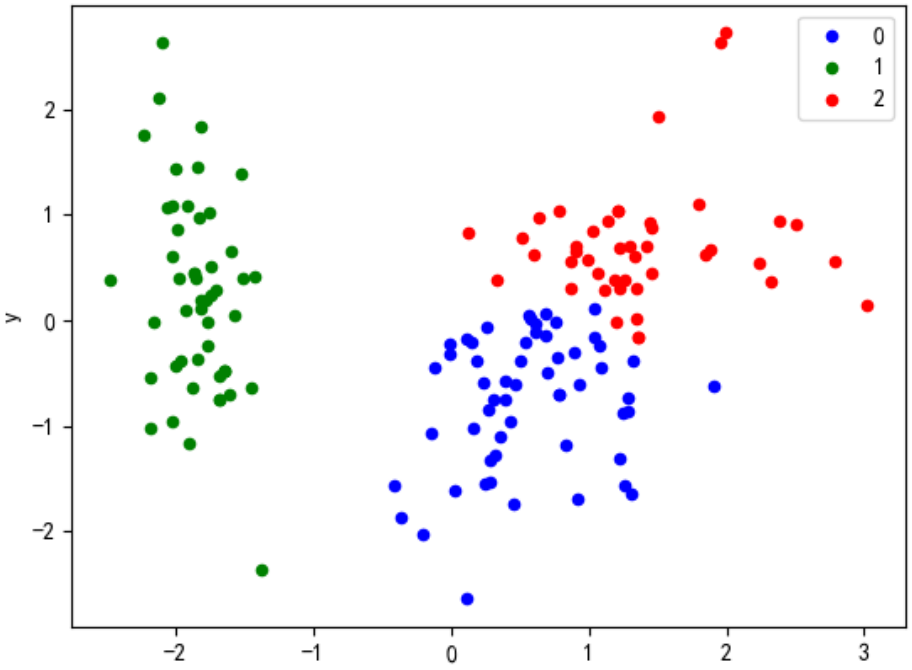
根据 SSE 与 K 值取值关系折线图，我们可以看到当 K 值为 3 时曲线

逐渐平滑，故取 K 值为 3。

(3)进行 Kmeans 聚类，且通过 PCA 降维将数据集投影到二维空间并绘制聚类情况散点图。

```
# 确定K值为3时SSE曲线接近平滑，故分为三类
belong, center = Kmeans(data, 3, 100)
# 主成分分析降至二维空间
pca = decomposition.PCA(n_components=2)
pca_data = pca.fit_transform(data.iloc[:,:-1].values)
new_data = pd.DataFrame()
new_data['x'] = pca_data[:, 0]
new_data['y'] = pca_data[:, 1]
new_data['label'] = belong
# 绘制散点图
ax = new_data[new_data['label'] == 0].plot(kind='scatter', x='x', y='y', color='blue', label='0')
new_data[new_data['label'] == 1].plot(kind='scatter', x='x', y='y', color='green', label='1', ax=ax)
new_data[new_data['label'] == 2].plot(kind='scatter', x='x', y='y', color='red', label='2', ax=ax)
plt.show()
```

图八、聚类并通过 PCA 降维绘制散点图



图九、鸢尾花数据集 Kmeans 聚类效果图

(4)总结：Kmeans 聚类效果较为明显，但手写 Kmeans 时间复杂度也较高，借助 PCA 主成分分析降至低维空间绘制散点图，可以看到分布较为集中。

②葡萄酒数据集(Wine)

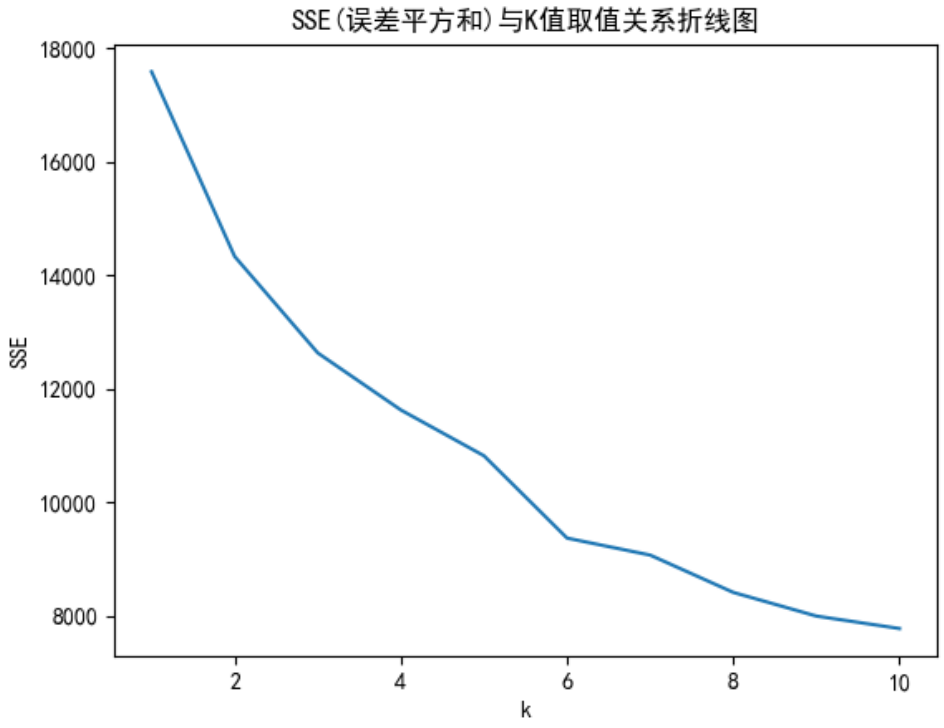
(1)数据读入并预处理。

```
def Kmeans_Wine():
    data = pd.read_csv('winequality-red.csv', sep=';')
    # 剥夺标签
    label = data['quality']
    data = data.drop(['quality'], axis=1)
    # 数据标准化处理
    SS_x = StandardScaler().fit(data)
    data = pd.DataFrame(SS_x.transform(data))
```

图十、葡萄酒数据集读入并标准化

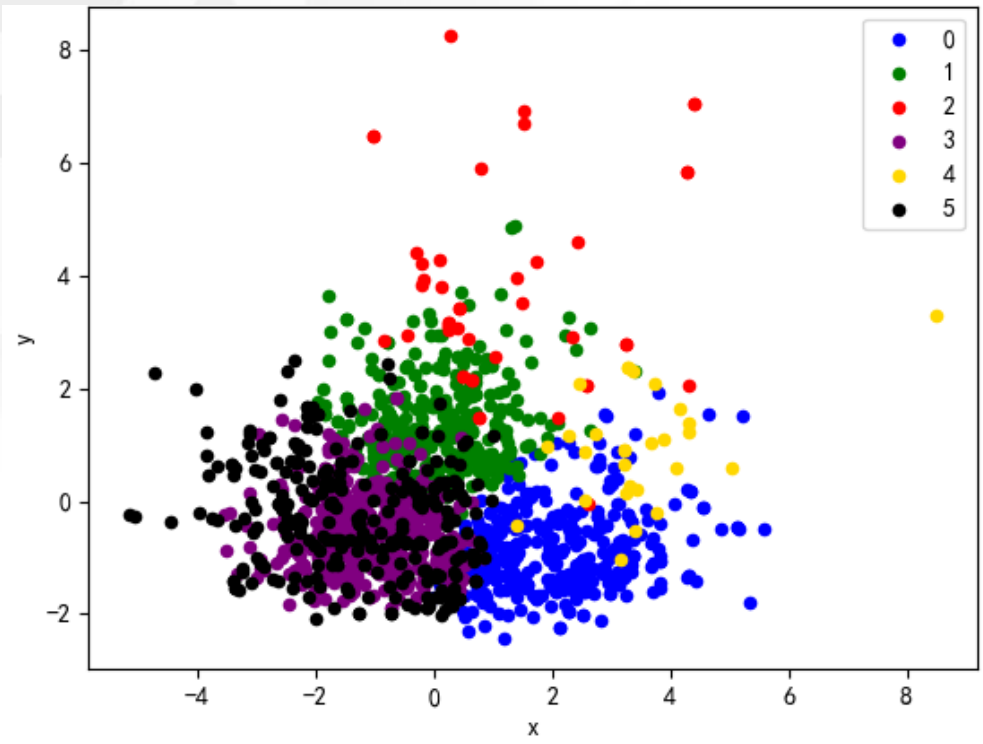
在输出了数据规模后，发现数据共有 1600 项，特征共有 12 个。

(2)手肘法确定 K 值，并绘制 SSE 与 K 值取值关系折线图



图十一、SSE 与 K 值取值关系折线图

由折线图可以看到，在 K 取 6 后 SSE 曲线逐渐平滑，故将 K 值取为 6，运行并输出聚类效果散点图。



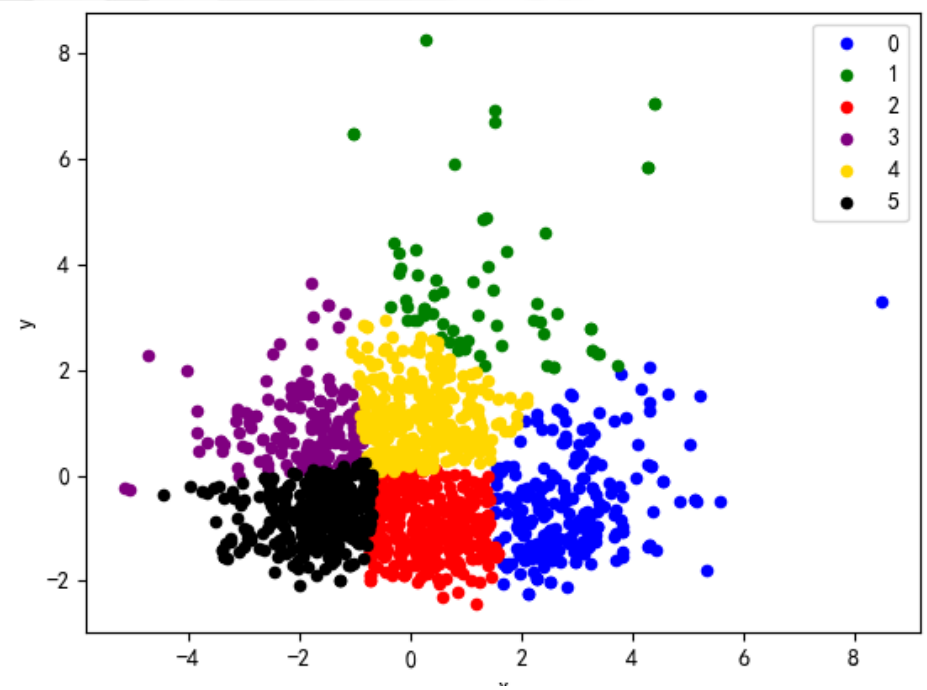
图十二、葡萄酒数据集聚类效果散点图

(3)分析：当涉及到大数据量时，Kmeans 效果较差，因数据分布比较集中，很难将数据分的清晰明了，且 Kmeans 时间复杂度较高，当前数据集为 1600 项数据和 12 个特征时，运行速度极慢。

(4)尝试 PCA 降维，将 12 维特征映射到二维空间，重新绘制了散点图，PCA 降维代码见图十三，聚类效果三点图见图十四。

```
# 主成分分析降至二维空间
pca = decomposition.PCA(n_components=2)
pca_data = pca.fit_transform(data.iloc[:, :-1].values)
print('Kmeans中，K值为6')
# 12维特征值空间降至二维
belong, center = Kmeans(pd.DataFrame(pca_data), 6, 10)
```

图十三、PCA 将 12 维特征空间降至 2 维空间

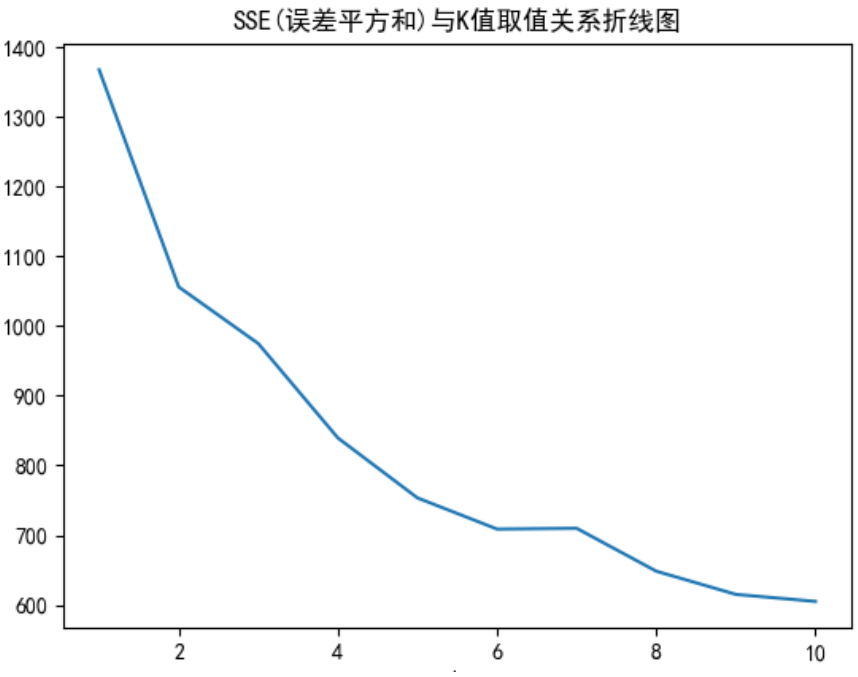


图十四、PCA 降维后的葡萄酒数据集聚类散点图

可见经过 PCA 降维将葡萄酒数据集特征空间降到二维后，Kmeans 聚类效果变佳，且收敛速度也变快了。

③宫颈癌行为风险数据集

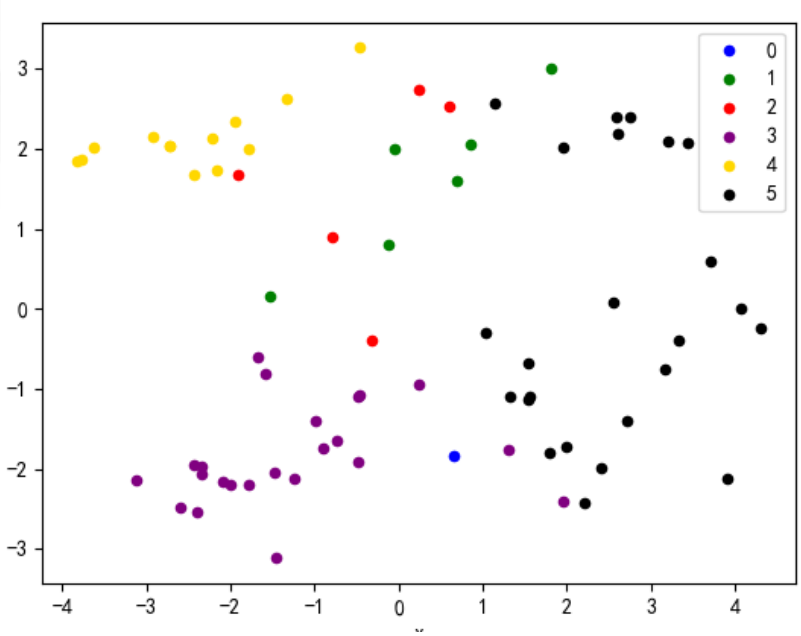
(1)数据读入并预处理，手肘法确定 K 值并输出 SSE 与所取 K 值关系折线图。



图十五、宫颈癌行为风险数据集 SSE 与 K 值曲线图

由图知 K 为 6 时曲线变得平缓，故取 K 值为 6。

(2)绘制聚类效果散点图。

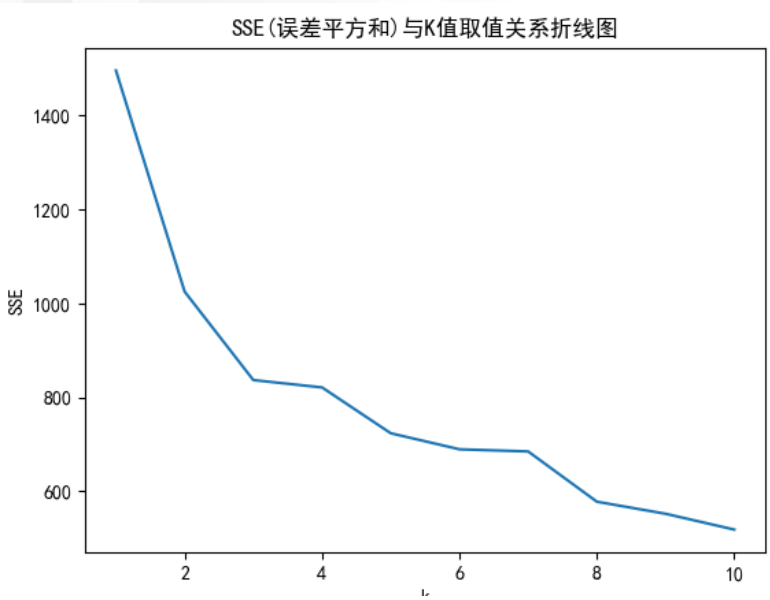


图十六、宫颈癌行为风险数据集聚类效果散点图

(3)总结：小数据量时运行时间较快，分类效果较为明显。

④陶瓷样品文化时期数据集

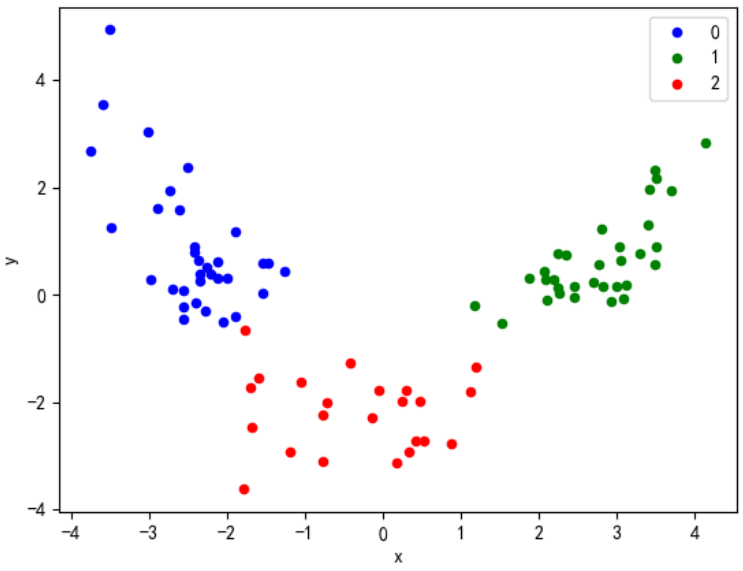
(1)读入数据并预处理，手肘法确定 K 值并输出 SSE 与 K 值取值关系折线图。



图十七、陶瓷数据集 SSE 与 K 值取值关系折线图

由上图可知，在 K 值取 3 后曲线逐渐变缓，故取 K 值为 3。

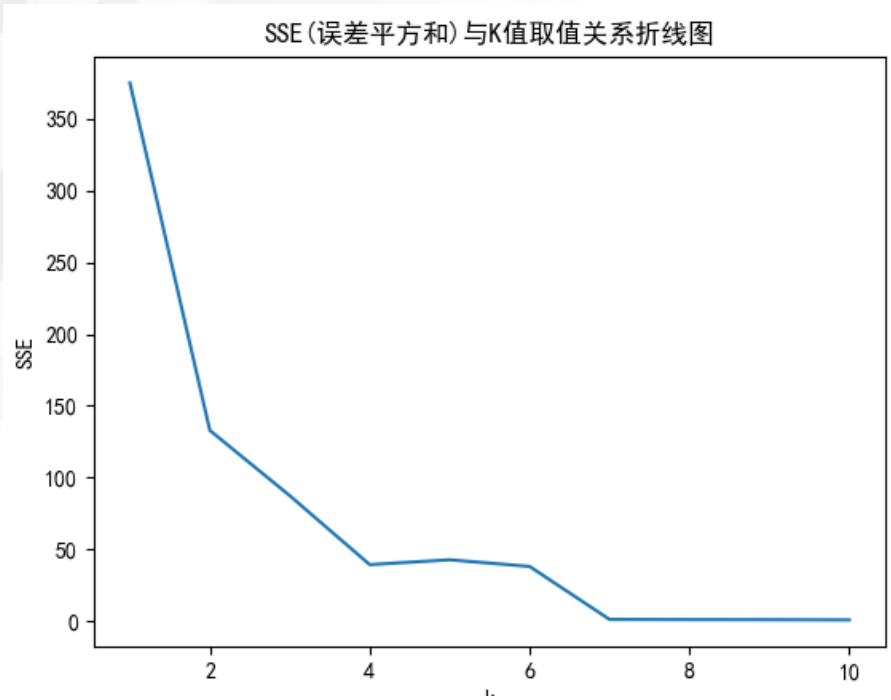
(2) 绘制聚类效果散点图。



图十八、陶瓷数据集聚类效果散点图

⑤酒精含量检测气体成分数据集

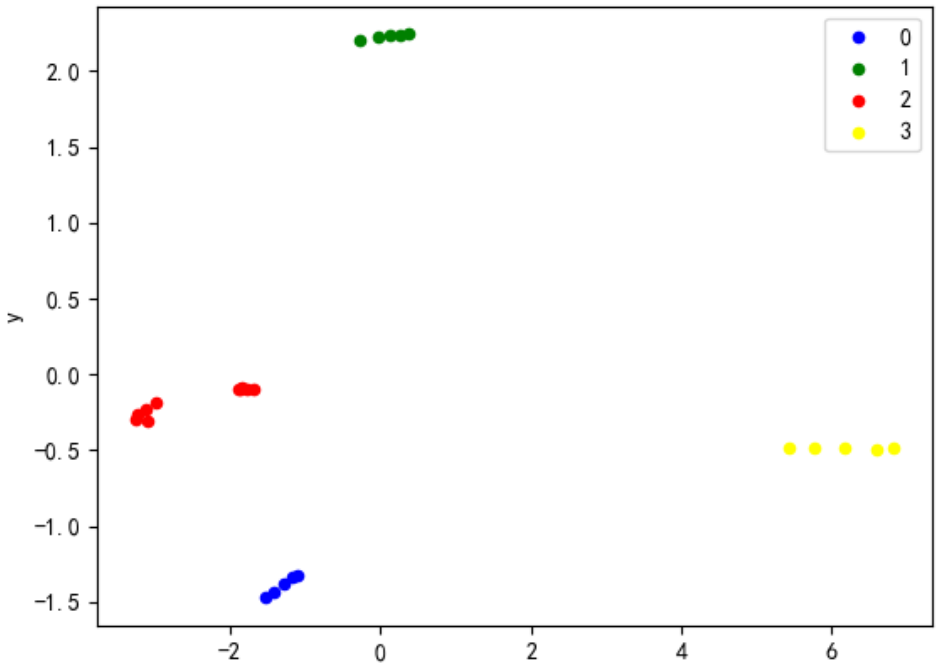
(1) 读入数据并预处理，手肘法确定 K 值并输出 SSE 与 K 值取值关系折线图。



图十九、酒精气体数据集 SSE 与 K 值取值关系折线图

由图可见 K 值取为 4 后，曲线变得平滑，故 K 值取为 4。

(2) 绘制聚类效果散点图。



图二十、酒精气体数据集聚类效果散点图

(2)实验总结与分析：

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 10 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，如⑤酒精气体数据集聚类效果散点图可见，聚类效果非常明显。

缺点：

1.Kmeans 的时间复杂度非常高，达到了 O(t*n*k*m)，t 为迭代次数，n 为数据项个数，m 为特征个数，k 为聚类中心个数。在②葡萄酒数据集中使用 Kmeans，由于数据项达到了 1600 个，且特征个数多达 12 个，Kmeans 进行一次迭代的速度极慢，手肘法测定 K 值运行时长达到了 10 分钟，这说明 Kmeans 只适用于小数据集，且聚类效果散点图显示，各簇数据分布较为集中，混杂在一起。当数据集特征过多时，应该采取 PCA 降维，就能得到较好的聚类效果。

2.当数据分离程度不够时，Kmeans 的聚类效果就显得比较一般。在③宫颈癌行为风险数据集中，数据分布的比较分散且无明显聚集，此时由聚类效果散点图可以看出 Kmeans 的聚类效果就非常一般。

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。

(3) 完整 Kmeans 代码(不含数据集测试代码)

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn import decomposition
from matplotlib import pyplot as plt

def get_dist(a, b):
    sum = 0
    for i in range(len(a)):
        sum += (a[i] - b[i]) ** 2
    return np.sqrt(sum)

def get_SSE(a, b):
    sum = 0
    for i in range(len(a)):
        sum += (a[i] - b[i]) ** 2
    return sum

def Kmeans(data, k, nums):
    # 标注每个数据属于哪个集合
    belong = [i for i in range(len(data))]
    # 初始簇中心采用从数据中抽样
    center = data.sample(k, axis = 0)
    # 迭代终止条件1，迭代到迭代上限次
    for t in range(nums):
        new_belong = [i for i in range(len(data))]
        # 计算每项数据到任一簇中心的欧式距离，并更新为最近的簇中心
        for i in range(len(data)):
            # 哨兵确定最小值
            dist = get_dist(data.iloc[i], center.iloc[0])
            idx = 0
            # 枚举簇中心
            for j in range(1, k):
                # 计算数据距离簇中心的欧氏距离
                dist1 = get_dist(data.iloc[i], center.iloc[j])
                # 如果距离该簇更近，就更新为新的簇中心
                if dist1 < dist:
                    dist = dist1
```

```
                    idx = j
            new_belong[i] = idx
        # 更新新的簇中心各维数据
        for i in range(k):
            for j in range(len(center.iloc[0])):
                sum, cnt = 0, 0
                for s in range(len(data)):
                    if new_belong[s] == i:
                        sum += data.iloc[s][j]
                        cnt += 1
                # 如果没有属于他的，就重新抽样
                if(cnt == 0):
                    center.iloc[i] = data.sample(1, axis = 0).values
                center.iloc[i][j] = sum / cnt
        # 迭代终止条件2，聚类稳定
        if(new_belong == belong):
            break
        belong = new_belong

    return belong, center

def confirm_K(n, data):  ##手肘法确定K值
    n = 10  # 手肘法K值上限
    SSE = []  # 误差平方和
    for i in range(1, n + 1):  # 枚举k值为[1, 10]

        print('第%d次KMeans中' % i)

        belong, center = Kmeans(data, i, 10)
        sum = 0
        for j in range(i):
            for k in range(len(data)):
                if belong[k] == j:
                    sum += get_SSE(data.iloc[k], center.iloc[j])
        SSE.append(sum)
    x = range(1, n + 1)
    # 绘制SSE与K值取值折线图
    plt.xlabel('k')
    plt.ylabel('SSE')

    plt.title('SSE(误差平方和)与K值取值关系折线图')

    plt.plot(x, SSE)
    plt.show()
```

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

(1)数据整体观察

1.先观察数据格式及所有特征和特征含义(见表 1)

User_id	Merchant_id	Coupon_id	Discount_rate	Distance	Date_received
4129537	450	9983	30:05:00	1	20160712
6949378	1300	3429	30:05:00	null	20160706

表一、特征和特征详情及数据格式

依次对应列名为用户名、商家 id、优惠卷 id、折扣率(分为满减和直接折扣)、用户常居地离店家距离、收到优惠卷的日期。折扣率分为两种情况，一种是满减类型，一种是折扣类型。

2.输出数据的详细情况和各列元素是否有缺失值需要处理。

```
共有 113640 条记录
共有 113640 条优惠卷的领取记录
共有 2050 种不同的优惠卷
共有 76309 个用户
共有 1559 个商家
优惠卷的领取时间在 20160701 到 20160731 之间
```

图一、数据表大体情况分析

由图一得知，记录比较完善，并没有缺失和不良数据，各项指标合理，且时间均处于 2016 年 7 月，可以对 2016 年 7 月的周末进行特殊时段处理。

4. 接下来查看数据是否有缺失值。

```
      User_id 列无缺失值
  Merchant_id 列无缺失值
    Coupon_id 列无缺失值
Discount_rate 列无缺失值
     Distance 列有缺失值
Date_received 列无缺失值
```

图二、数据缺失情况检索

由图二得知，“Distance”列存在缺失值情况，需要进一步处理。

(2)绘制各特征数据可视化图表，观察特征数据详细分布情况。

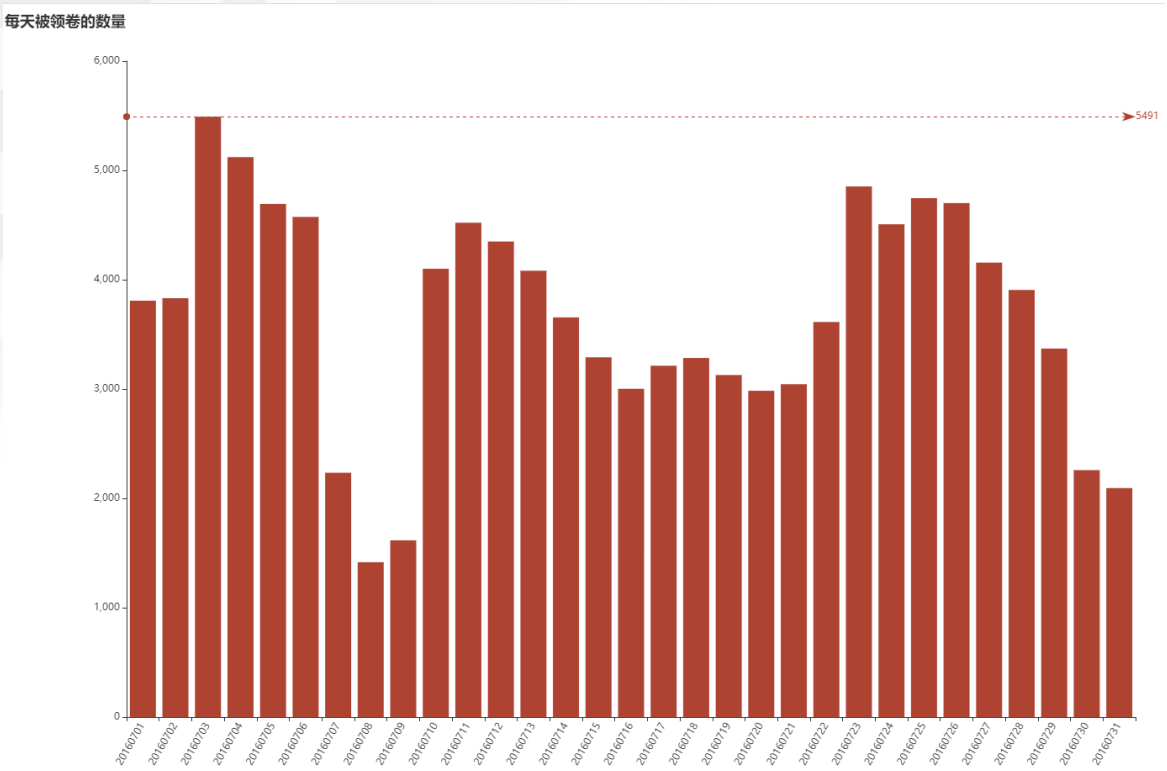
①每日领卷数量

先将每日领卷数量提取出来，借助 pandas 的 groupby 函数得到各日期领卷数量具体情况，关键代码见图三。

```
# 选取领卷日期不为空的数据
df_1 = offline[offline['Date_received'].notna()]   #并没有空数据
# 以领卷日期为分组依据并统计每日领卷数量，as_index参数表示是否以该关键字作为行索引
tmp = df_1.groupby('Date_received', as_index=False)['Coupon_id'].count()
```

图三、提取每日领卷数量代码

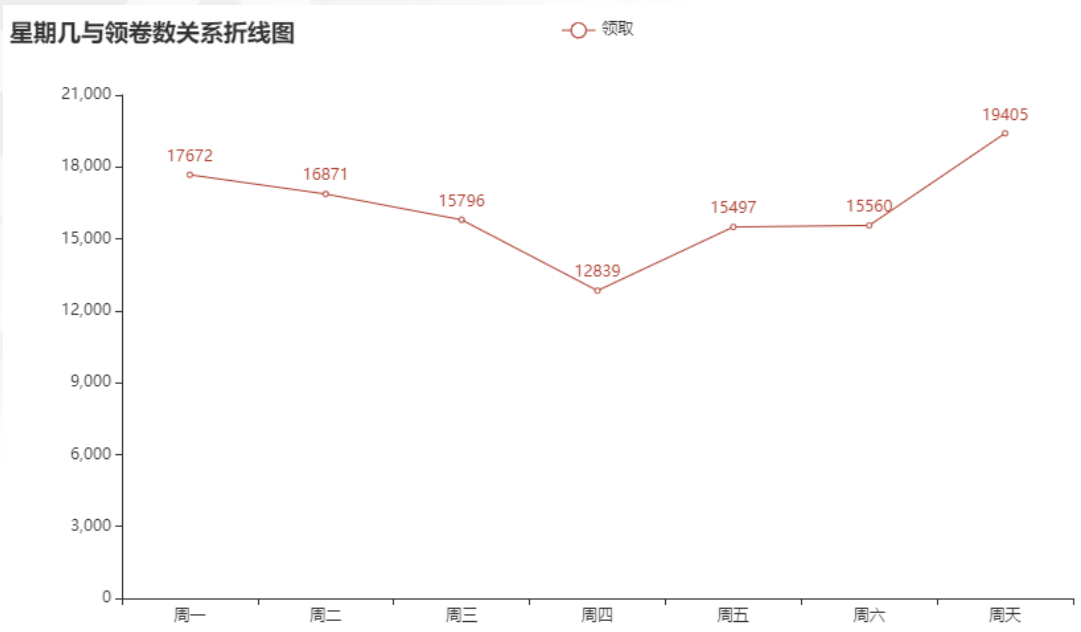
借助 pyecharts 库绘制对应柱状图如下(图四)。



图四、每日领卷数量柱状图

由图四可以很清晰地得到，整体领卷数量在月初、月中、月末有着较大的差距，月初领取数量明显少于月中和月末，考虑到大部分工资发放日期对消费周期的影响，应将数据添加一维考虑其是否为月初、月中、月末。每日领卷数量呈周期式分布且基本符合七天一循环，故针对每周内周几对于领卷数量具体影响进行深入调查。

为观察星期几与用户领取优惠卷数量的实际联系，提取指定数据并绘制成周一到周天各天领卷数量折线图(见图五)。



图五、星期几与领卷数关系折线图

由图五我们发现相对领卷数量较少的日期都是周三到周六，考虑到节假日及周末对优惠卷促销情况的影响，应在数据预处理轮加入对当前日期具体为周几的特征维度，对于周三到周六的情况应当适当降低影响权重。

②不同消费距离的优惠卷数量

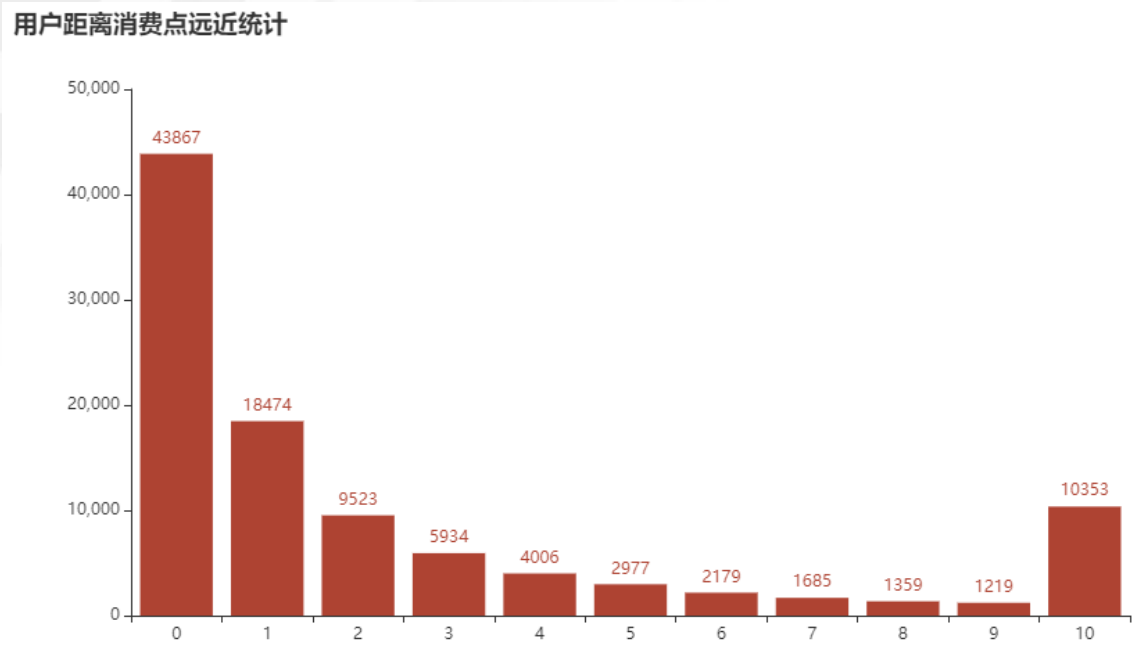
将数据按消费距离分组并统计，且按照距离远近进行排序，关键代码如下(见图六)。

```
# 统计各用户的消费距离
dis = offline[offline['Distance'] != -1]['Distance'].values
dis.sort()          ##按照距离排序，方便图表显示
dist = dict(collections.Counter(dis))       #将对应距离单独做成字典
x = list(dist.keys())          #消费距离
y = list(dist.values())        #对应距离的数量统计
```

图六、不同消费距离优惠卷数量统计

将所得数据绘图查看得不同消费距离优惠卷数量统计的柱状图（见图

七)。



图七、不同消费距离优惠卷数量统计柱状图

由图七我们可以看到，大多数人还是更关注距离自己更近商家，用户领取心仪商家优惠卷数量随着距离递增而递减，但在距离大于 5km 时，出现了极特殊的情况，有少部分用户领取了距离自己很远的商家的优惠卷，可见存在一部分用户消费是为了商家质量和商家口碑服务而去的。 在后续数据预处理过程中可以将距离商家距离超过 5km 且领取了优惠卷的用户打上标记，表示该用户对该商家拥有极高的回头率，很可能在 15 天内使用该优惠卷。

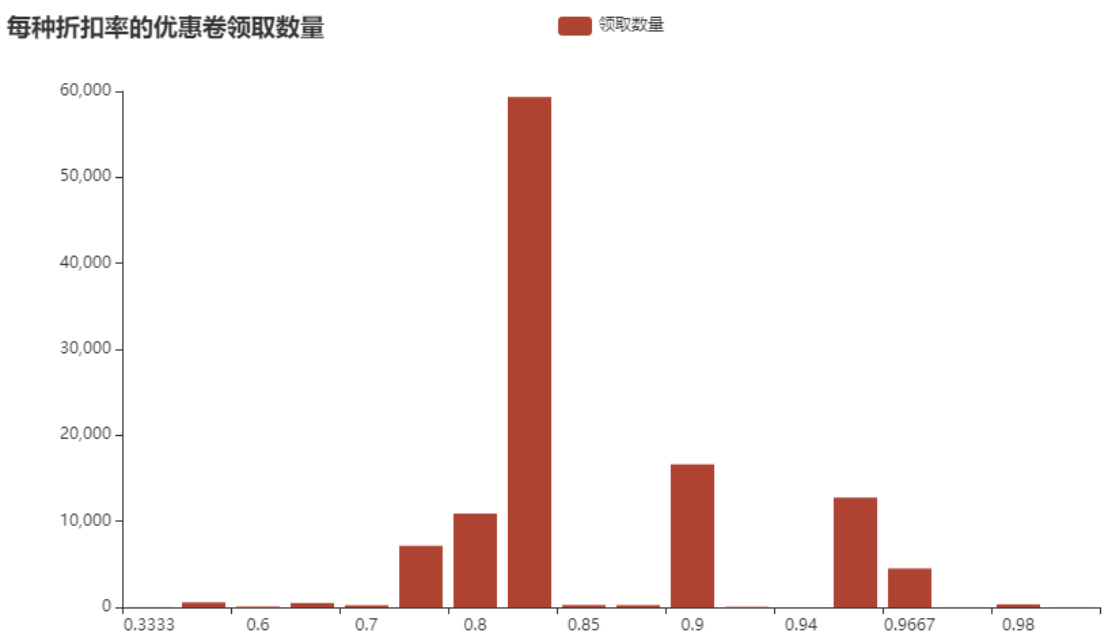
③各种折扣率的优惠卷数量

将折扣率转化后得到各种折扣率数量优惠卷的数据，关键代码如下(见图八)。

```
received = offline[['discount_rate']]
received['cnt'] = 1
received = received.groupby('discount_rate').agg('sum').reset_index()
```

图八、各种折扣率优惠卷数量数据统计

将所得数据绘图查看各种折扣率优惠卷数量的柱状图(见图九)。

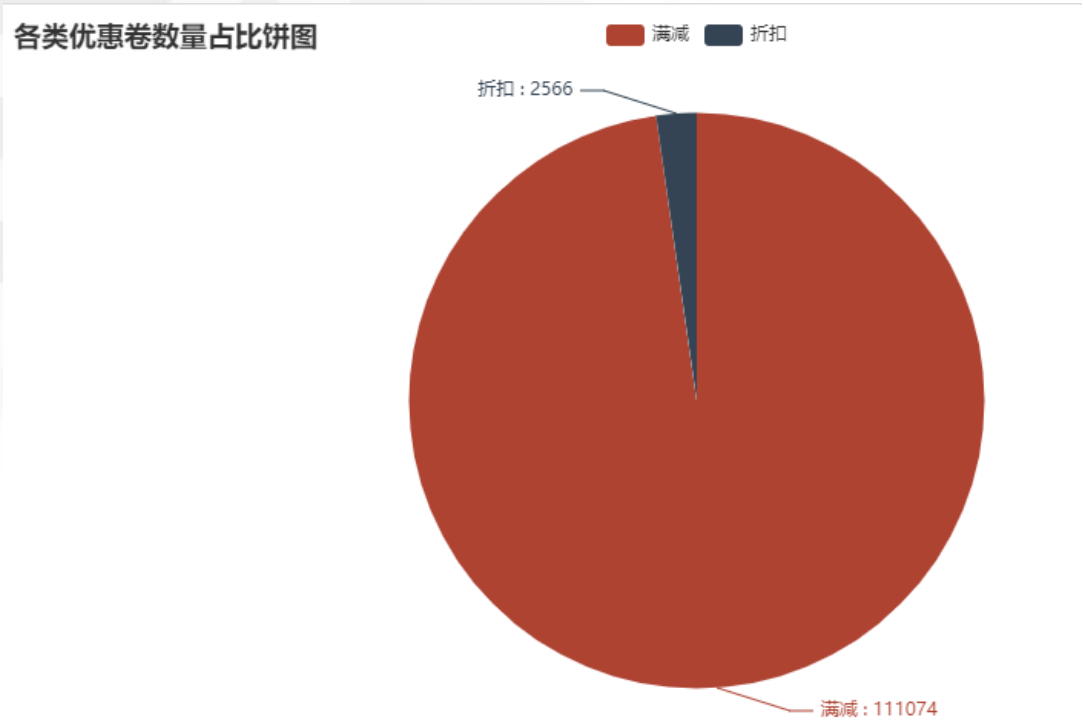


图九、各种折扣率优惠卷数量柱状图

由图九我们可以看到，优惠卷折扣率主要集中在八折及九五折左右，因为更高的折扣率可能会影响商家资金回收率，更低的折扣率又难以吸引客户。但显而易见的是，高折扣的优惠卷对用户的吸引程度是非常大的，适当给许久未消费的老客户发放大额优惠卷是非常有效的促销手段，可以在后续数据预处理及建模阶段加入折扣率对使用优惠卷概率的因素，并将五折以下优惠卷视为高折扣率优惠券，特殊处理和讨论。

④不同折扣类型优惠卷数量情况

将不同折扣类型的优惠卷数据进行转换和提取后，得到如下饼图(见图十)。



图十、各类优惠卷数量占比饼图

由图十我们可以得知，满减类型优惠卷数量远多于直接折扣类型优惠卷，由于直接折扣类型优惠卷没有使用下限和指定额度，更容易受到用户的欢迎，所以后续建模过程中应加入其是否为折扣类型对使用率的影响权重。而满减最低消费的阈值也影响着用户对该优惠卷的评价，太高的消费门槛会导致该优惠卷成为“废卷”，故应当加入满减最低消费的特征。

⑤不同消费距离下折扣率对用户的吸引程度

综合情况②和情况③，我们可以发现各种消费距离对用户的吸引力度是不同的，所以在此深入探究不同消费距离下多大的折扣力度才能吸引到用户领取消费卷。

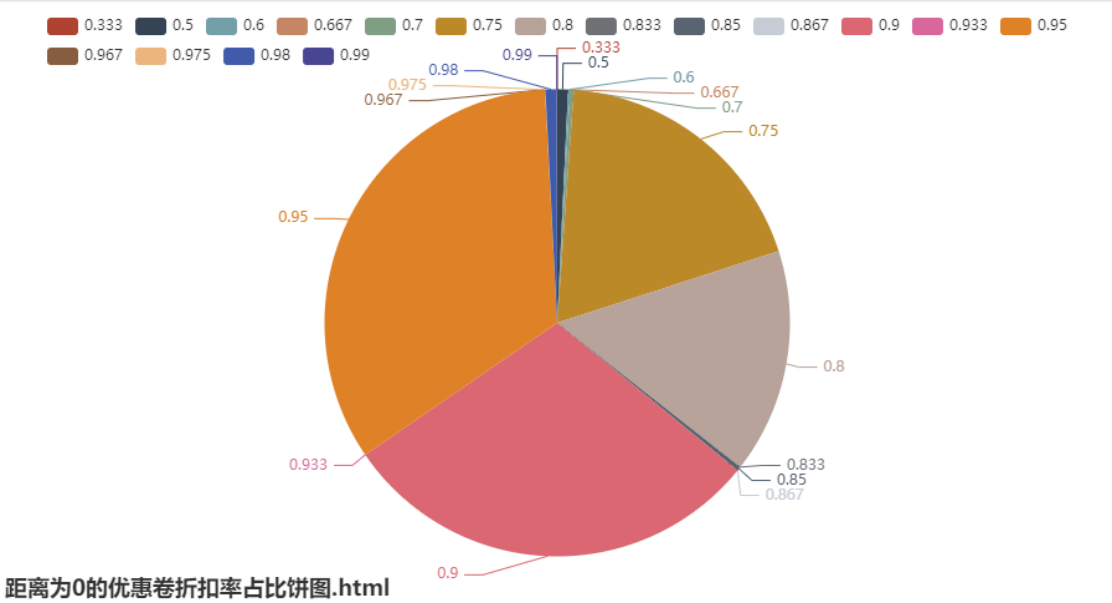
将数据按照不同消费距离分组，挑选近距离、中距离、远距离三种消费距离情况具有代表性的距离组别为代表，分部设为 0.5km 内为近距离，2.5km 左右为中距离，5km 以外为远距离。将三个距离的具体数据提取出来，做统计

处理，代码如下(见图十一)。

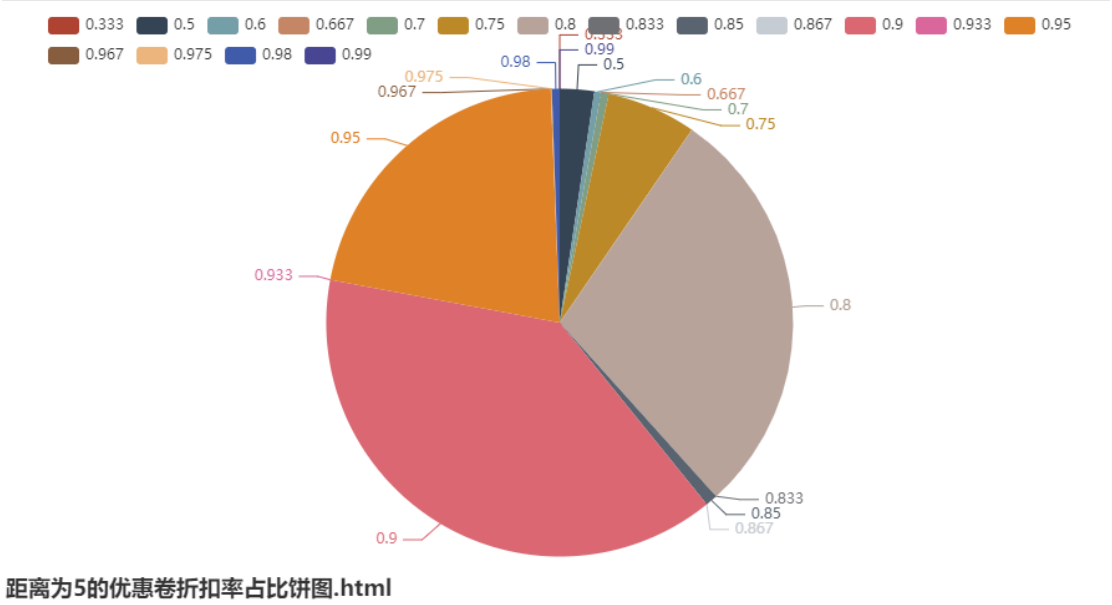
```
# 提取数据中距离非空记录
df_1 = offline[offline['Distance'] != -1]
# 提取指定特征
df_1 = df_1.loc[:, ['Distance', 'discount_rate']]
# 分组并统计各个距离下的各种优惠卷领取情况，选取距离为[0, 5, 10]
df_1 = df_1.groupby([df_1['Distance'], df_1['discount_rate']]).agg('size')
# 提取对应距离对应的折扣率的统计个数
cnt = [dict(df_1[i]) for i in range(11)]
idx = list(df_1[0].keys())
# 将折扣率保存为3位小数
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]))        #要转换成int，np.int会报错
        else:
            tmp.append(int(0))
    num.append(tmp)
```

图十一、提取不同消费距离不同折扣率领取情况数据

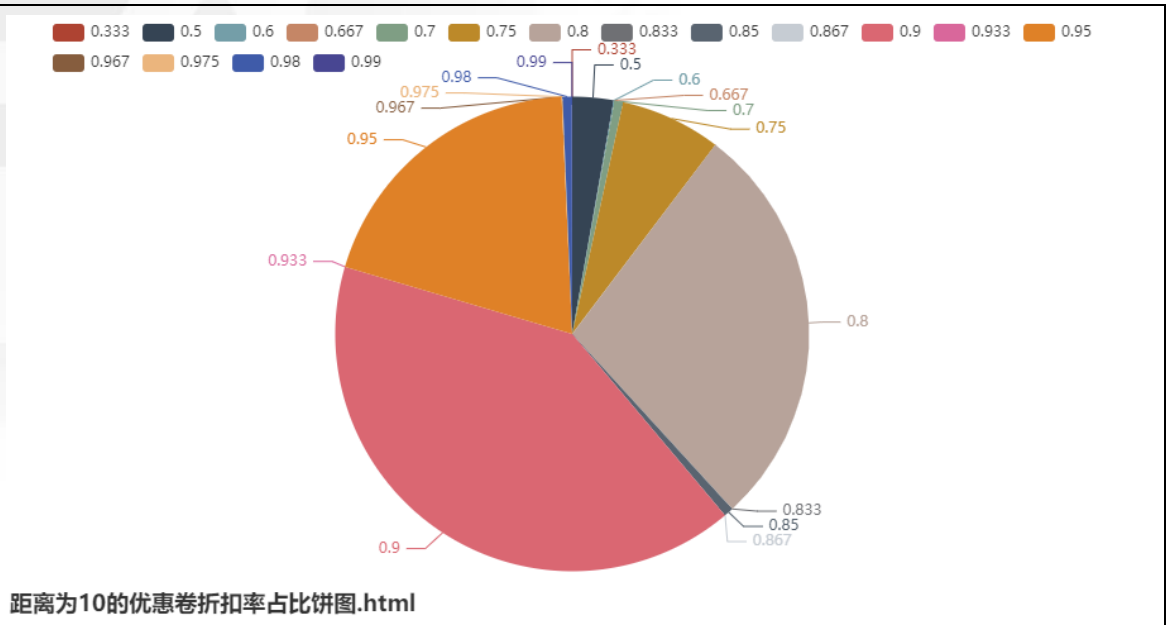
由该数据分别做出近距离、中距离、远距离情况下优惠卷领取折扣率情况占比饼图(见图十二到图十四)。



图十二、近距离情况下折扣率占比



图十三、中距离情况下折扣率占比

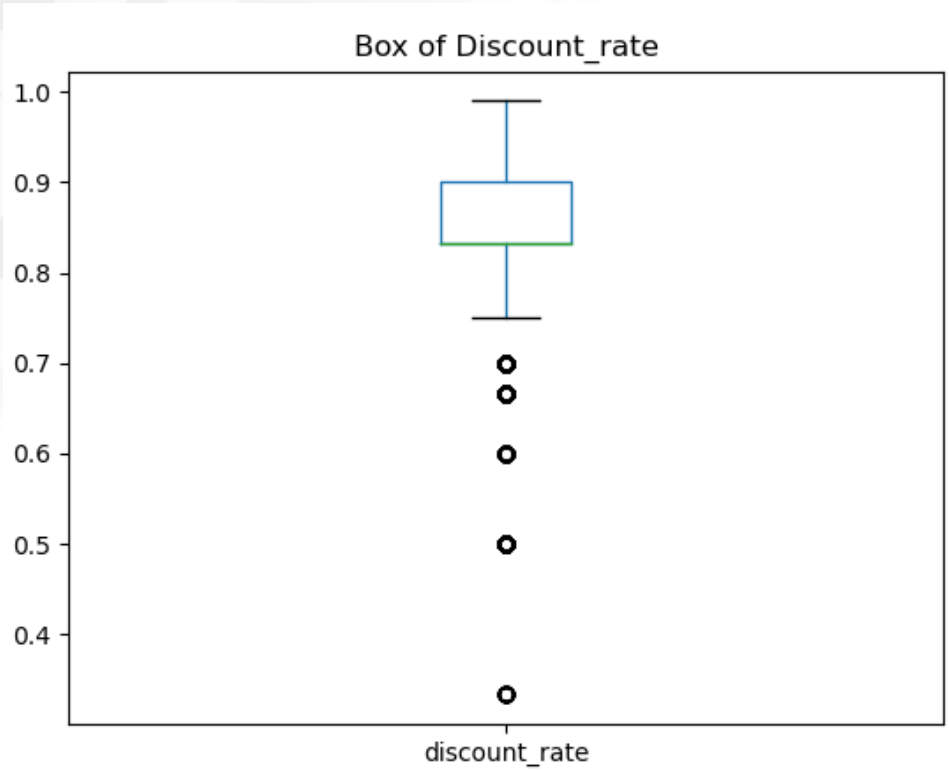


图十四、远距离情况下折扣率占比

由上述三张饼图我们可以看到，在近距离情况下还能大量吸引用户的九五折优惠卷在到了中距离和远距离情况下却并不好使了，结合用户在高折扣率下才会去远处消费的心理，应该合理设定阈值以探求用户在多大的折扣率下才会去远处消费。而远距离情况下的优惠卷分布与中距离情况的优惠卷分布相差无几，可见这部分顾客仍是被商家本身吸引，并不太在意优惠卷折扣率，后续建模过程中可以考虑添加阈值设定，以及对远距离用户特殊讨论，采用“回头客”机制。

⑥折扣率整体分布情况箱线图(异常值检测)

提取相关数据并绘制成箱线图，以折扣率为代表特征判断数据集是否存在异常值，绘制箱线图如下(见图十五)



图十五、折扣率分布情况箱线图

由图十五我们可以观测到，主要优惠卷都集中在七五折到九折之间，但小部分商家会尝试发出大额优惠卷，这部分数据是极其宝贵的用户对大额优惠卷反响情况的具体体现，所以不该删除此离群点。故无需进行异常值清洗。

(3)总结及分析

1.经过数据观察后，得知数据集整体情况分布较为规律，消费距离列存在缺失值，需要进行删除或拟合来处理缺失值。2.经过具体数据特征可视化分析后，发现数据并无异常值需要处理，但数据集的各类型特征比重存在较大差异，可以进行以下几点来对数据进一步处理：①将数据日期格式转化成 Datetime 类，并将数据添加一维标签，标记该日期具体为周几及月初、月中、月末，来探测当前日期为一星期中不同日子及工资发放日期对领卷情况

的影响。②将距离店家 5km 以外却仍领取了优惠卷的用户标记为“回头客”，以此增大“回头客”对优惠卷使用可能性的权重。③将满减和折扣类型的优惠卷一并转换成折扣率，并将七折以下的折扣卷标记为大额优惠卷，将满减最低消费提取出来。④标记数据是否为满减类型。3. 在建模过程中可以设定合适的在不同消费距离的情况下折扣率阈值来探求对用户核销优惠卷概率的影响。

(4) 完整实验代码

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
import datetime
import xgboost as xgb
import lightgbm as lgb
import pyecharts
from pyecharts.charts import Bar, Line, Pie
import collections
from pyecharts import options as opts
from matplotlib import pyplot as plt

def show_data_detail():
    # 展示共有多少条记录
    record_count = data.shape[0]

    print('共有 %d 条记录' % record_count)


    # 展示共有多少条优惠卷的领取记录
    received_count = data['Date_received'].count()

    print('共有 %d 条优惠卷的领取记录' % received_count)


    # 展示共有多少种不同的优惠卷
    coupon_count = len(data['Coupon_id'].value_counts())

    print('共有 %d 种不同的优惠卷' % coupon_count)


    # 展示共有多少个用户
    user_count = len(data['User_id'].value_counts())
```

```
    print('共有 %d 个用户' % user_count)

    # 展示共有多少个商家
    merchant_count = len(data['Merchant_id'].value_counts())
    print('共有 %d 个商家' % merchant_count)

    # 展示最早领卷时间
    min_received = str(int(data['Date_received'].min()))

    # 展示最晚领卷时间
    max_received = str(int(data['Date_received'].max()))
    print('优惠卷的领取时间在 %s 到 %s 之间' % (min_received, max_received))

    idx = list(data.columns.values)
    for i in idx:
        if(data[i].isnull().any()):
            print("%20s 列有缺失值" % i)
        else:
            print("%20s 列无缺失值" % i)


def data_pretreatment():
    # 将距离空值填充为-1
    offline['Distance'].fillna(-1, inplace = True)
    # 计算折扣率
    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])))
    # 大额优惠卷标记
    offline['high_discount_rate'] = offline['discount_rate'].map(lambda x : 1 if x <= 0.7
else 0)
    # 添加优惠卷是否为满减类型
    offline['isManjian'] = offline['Discount_rate'].map(lambda x : 1 if ':' in str(x) else
0)
    # 满减的最低消费
    offline['min_cost_of_manjian'] = offline['Siscount_rate'].map(lambda x : -1 if ':'
not in str(x) else int(str(x).split(':')[0]))
    # 将收到优惠卷日期转换类型
    offline['date_received'] = pd.to_datetime(offline['Date_received'],
format="%Y%m%d")
    # 添加领卷时间为周几
    offline['weekday_receive'] = offline['date_received'].apply(lambda x :
```

```
x.isoweekday())
    # 添加领卷时间为月初月中月末
    date1 = datetime.date(2016,7,10)   #月初和月中的分界线
    date2 = datetime.date(2016,7,20)   #月中和月末的分界线
    offline['month_pharse'] = offline['date_received'].apply(lambda x : 0 if x <= date1
else 1 if x <= date2 else 2)
    # 回头客机制
    offline['is_changke'] = offline['Distance'].apply(lambda x : 1 if int(x) == 10 else
0)
    # 将预处理完的数据保存，这样就不用重复预处理
    offline.to_csv('data.csv')

def show_Bar():
    ######################每日领卷数量的柱状图######################
    # 选取领卷日期不为空的数据
    df_1 = offline[offline['Date_received'].notna()]   #并没有空数据
    # 以领卷日期为分组依据并统计每日领卷数量，as_index 参数表示是否以该关键字作为行索引
    tmp = df_1.groupby('Date_received', as_index = False)['Coupon_id'].count()
    bar_1 = (
        Bar(init_opts=opts.InitOpts(width = "1500px", height = "900px"))
        .add_xaxis(list(tmp['Date_received']))
        .add_yaxis('', list(tmp['Coupon_id']))
        .set_global_opts(

            title_opts = opts.TitleOpts(title = "每天被领卷的数量"),   #标题

            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),
            markline_opts = opts.MarkLineOpts(

                data = [opts.MarkLineItem(type_ = "max", name = "最大值")]    #标注最大值

            )
        )
    )

    bar_1.render('bar_1.html')
    ######################消费距离柱状图######################
    # 统计各用户的消费距离
    dis = offline[offline['Distance'] != -1]['Distance'].values
    dis.sort()          ##按照距离排序，方便图表显示
    dist = dict(collections.Counter(dis))      #将对应距离单独做成字典
    x = list(dist.keys())         #消费距离
    y = list(dist.values())       #对应距离的数量统计
```

```
    bar_2 = (
        Bar()
        .add_xaxis(x)
        .add_yaxis('', y)
        .set_global_opts(
            title_opts = opts.TitleOpts(title = "用户距离消费点远近统计")
        )
    )
    bar_2.render('bar_2.html')

    ####################各种折扣率的优惠卷的数量##############
    received = offline[['discount_rate']]
    received['cnt'] = 1
    received = received.groupby('discount_rate').agg('sum').reset_index()
    bar_3 = (
        Bar()
        .add_xaxis([float("%.4f" % x) for x in list(received.discount_rate)])
        .add_yaxis("领取数量", list(received.cnt))
        .set_global_opts(title_opts={"text" : "每种折扣率的优惠卷领取数量"})
        .set_series_opts(opts.LabelOpts(is_show=False))
    )

    bar_3.render('bar_3.html')

def show_line():
    ####################展示周一到周天各天领卷数量折线图##################
    week_coupon = offline['weekday_receive'].value_counts()
    week_coupon.sort_index(inplace = True)

    line_1 = (
        Line()
        .add_xaxis([('周' + x) for x in '一二三四五六天'])
        .add_yaxis("领取", list(week_coupon))
        .set_global_opts(title_opts = {"text" : "星期几与领卷数关系折线图"})
    )
    line_1.render('line_1.html')

def show_Pie():
    ##################各类优惠卷数量占比饼图##################
```

```
    v1 = ['满减', '折扣']
    v2 = list(offline[offline['date_received'].notna()]['isManjian'].value_counts())
    pie_1 = (
        Pie()
        .add("", [list(v) for v in zip(v1, v2)])
        .set_global_opts(title_opts={'text' : "各类优惠卷数量占比饼图"})
        .set_series_opts(label_opts=opts.LabelOpts(formatter = "{b} : {c}"))
    )
    pie_1.render('pie_1.html')

    ######################观察近,中,远三个距离情况下多大的折扣力度才会吸引用户领卷
##################
    # 提取数据中距离非空记录
    df_1 = offline[offline['Distance'] != -1]
    # 提取指定特征
    df_1 = df_1.loc[:, ['Distance', 'discount_rate']]
    # 分组并统计各个距离下的各种优惠卷领取情况, 选取距离为[0, 5, 10]
    df_1 = df_1.groupby([df_1['Distance'], df_1['discount_rate']]).agg('size')
    # 提取对应距离对应的折扣率的统计个数
    cnt = [dict(df_1[i]) for i in range(11)]
    idx = list(df_1[0].keys())
    # 将折扣率保存为3 位小数
    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]))        #要转换成 int, np.int 会报错
            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 的优惠卷折扣率占比

饼图.html' % dis[x], pos_top = '90%'))
        )
```

```
        pie.render('距离为%d 的优惠卷折扣率占比饼图.html' % dis[x])


if __name__ == '__main__':
    # 读入数据
    data = pd.read_csv('ccf_offline_stage1_test_revised.csv')
    offline = data.copy()
    # 展示数据细节
    show_data_detail()
    # 数据预处理，并将处理完的数据保存为data.csv
    data_pretreatment()                    ########再做一个是否为节假日 周末的识别
    # 展示数据情况可视化柱状图
    show_Bar()
    # 展示数据情况可视化折线图
    show_line()
    # 展示数据情况可视化饼图
    offline['discount_rate'].plot.box(title = "Box of Discount_rate")
    plt.show()
    show_Pie()
```

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

(1)实验步骤

①)数据日期格式转化及进一步对日期特征抽取新的特征

将收到的优惠卷日期转换为 Datetime 类(见图一)。

```
# 将收到优惠卷日期转换类型
offline['date_received'] = pd.to_datetime(offline['Date_received'], format="%Y%m%d")
```

图一、日期类型转换

添加领卷日期为周几和月初月中月末(见图二)。

```
# 添加领卷时间为周几
offline['weekday_receive'] = offline['date_received'].apply(lambda x: x.isoweekday())
# 添加领卷时间为月初月中月末
offline['is_yuechu'] = offline['date_received'].map(lambda x: -1 if pd.isnull(x) else 1 if x.day <= 10 and x.day > 0 else 0)  # 判断月初
offline['is_yuezhong'] = offline['date_received'].map(lambda x: -1 if pd.isnull(x) else 1 if x.day <= 20 and x.day > 10 else 0)  # 判断月中
offline['is_yuemo'] = offline['date_received'].map(lambda x: -1 if pd.isnull(x) else 1 if x.day <= 30 and x.day > 20 else 0)  # 判断月末
```

图二、领卷日期数据进一步探索

②增加“回头客”机制

将距离该店家 5km 以上的用户打上标记，以此来记录有极大可能光顾商家的“回头客”。

```
# 回头客机制
offline['is_changke'] = offline['Distance'].apply(lambda x: 1 if int(x) == 10 else 0)
```

图三、每个商家回头客数据集的构建

③折扣率转换及大额优惠卷的标记、满减最低消费的转换

将满减类型转化成对应折扣率，并将七折以下的大额优惠卷标记以及是否为满减类型的标记和满减最低消费的提取(见图四)。

```
# 计算折扣率
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])))
# 大额优惠卷标记
offline['high_discount_rate'] = offline['discount_rate'].map(lambda x: 1 if x <= 0.7 else 0)
# 添加优惠卷是否为满减类型
offline['isManjian'] = offline['Discount_rate'].map(lambda x: 1 if ':' in str(x) else 0)
# 满减的最低消费
offline['min_cost_of_manjian'] = offline['Discount_rate'].map(lambda x: -1 if ':' not in str(x) else int(str(x).split(':')[0]))
```

图四、折扣率转换及大额优惠卷的标记

④缺失值处理

将消费距离为空的数据填充为-1。

```
# 将距离空值填充为-1
offline['Distance'].fillna(-1, inplace=True)
```

图五、缺失值处理

(2)实验结果总结及分析

数据预处理后的数据情况如下

	User_id	Merchant_id	...	month_pharse	is_changke
0	4129537	450	...	1	0
1	6949378	1300	...	0	0
2	2166529	7113	...	2	0
3	2166529	7113	...	2	0
4	6172162	7605	...	0	0
...	...	...	...	...	...
113635	5828093	5717	...	1	1
113636	6626813	1699	...	0	0
113637	6626813	7321	...	1	0
113638	4547069	760	...	1	0
113639	6675965	7487	...	2	0

[113640 rows x 14 columns]

图六、预处理完成后的数据情况

经过数据预处理后，将缺失值填充，考虑到后续可能会影响建模预测优惠卷核销情况的准确率，进行了数据特征的进一步高纬度提取。将日期数据转换，并添加了星期几及月初、月中、月末的工资发放情况对核销优惠卷的特征。添加“回头客”机制，将超过 5km 的用户标记为“回头客”。将折扣率转换，并标记了大额优惠卷和是否为满减类型以及满减最低消费。对后续建模起到了良好作用。

(3) 完整实验代码

```
def data_pretreatment():
    # 将距离空值填充为-1
    offline['Distance'].fillna(-1, inplace = True)
    # 计算折扣率
    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])))
    # 大额优惠卷标记
    offline['high_discount_rate'] = offline['discount_rate'].map(lambda x : 1 if x <= 0.7
else 0)
    # 添加优惠卷是否为满减类型
    offline['isManjian'] = offline['Discount_rate'].map(lambda x : 1 if ':' in str(x) else
0)
    # 满减的最低消费
    offline['min_cost_of_manjian'] = offline['Siscount_rate'].map(lambda x : -1 if ':'
not in str(x) else int(str(x).split(':')[0]))
    # 将收到优惠卷日期转换类型
    offline['date_received'] = pd.to_datetime(offline['Date_received'],
format="%Y%m%d")
    # 添加领卷时间为周几
    offline['weekday_receive'] = offline['date_received'].apply(lambda x :
x.isoweekday())
    # 添加领卷时间为月初月中月末
    offline['is_yuechu'] = offline['date_received'].map(lambda x: -1 if pd.isnull(x) else
1 if x.day <= 10 and x.day > 0 else 0)  # 判断月初
    offline['is_yuezhong'] = offline['date_received'].map(lambda x: -1 if pd.isnull(x)
else 1 if x.day <= 20 and x.day > 10 else 0)  # 判断月中
    offline['is_yuemo'] = offline['date_received'].map(lambda x: -1 if pd.isnull(x) else
1 if x.day <= 30 and x.day > 20 else 0)  # 判断月末
    # 回头客机制
```

```
    offline['is_changke'] = offline['Distance'].apply(lambda x : 1 if int(x) == 10 else
0)
    # 将预处理完的数据保存，这样就不用重复预处理
    offline.to_csv('data.csv')
```

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

(1)实验步骤

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 数据集中的主键 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。

```
# 方便特征提取
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，按label数据集中出现过的用户来提取history数据集中的用户消费数据
u_feat = label_field[keys].drop_duplicates(keep='first')
```

图一、主键提取及数据表准备

①用户领卷数

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。

```
# 1.用户领卷数
# 以keys为键，以'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列与特征左连接
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
# 缺失值填充为0，加上参数downcast = 'infer'
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。

```
# 2.用户领卷并消费数
#                       限制条件为数据的Date列非空
pivot = pd.pivot_table(data[data['Date'].map(lambda x: str(x) != 'nan')], index=keys, values='cnt', aggfunc=len)
pivot = pd.DataFrame(pivot).rename(columns={'cnt': prefixs + 'receive_and_consume_cnt'}).reset_index()
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
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。

```
# 3.用户领卷未消费数
pivot = pd.pivot_table(data[data['Date'].map(lambda x: str(x) == 'nan')], index=keys, values='cnt', aggfunc=len)
pivot = pd.DataFrame(pivot).rename(columns={'cnt': prefixs + 'receive_not_consume_cnt'}).reset_index()
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
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。

```
# 4.用户领卷核销率
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']
))
```

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。

```
# 5.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。

```python
# 6.领取并消费优惠卷平均距离
# 临时数据表，添加第一个限制条件
data_tmp = data[data['Date'].map(lambda x: str(x) != 'nan')]
# 在临时数据表的基础上添加第二个限制条件
pivot = pd.pivot_table(data_tmp[data_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。

```python
# 7.在多少不同商家领取并消费优惠卷
# 以keys为键，以'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['Date'].map(lambda x: str(x) != 'nan')], index=keys, values='Merchant_id', aggfunc=lambda x: len(set(x))
)
pivot = pd.DataFrame(pivot).rename(
    columns={'Merchant_id': prefixs + 'receive_and_consume_differ_Merchant_cnt'}).reset_index()
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
u_feat.fillna(0, downcast='infer', inplace=True)
```

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。

```python
# 8.在多少不同商家领取优惠卷
pivot = pd.pivot_table(data, index=keys, values='Merchant_id', aggfunc=lambda x: len(set(x)))
pivot = pd.DataFrame(pivot).rename(columns={'Merchant_id': prefixs + 'receive_differ_Merchant_cnt'}).reset_index()
u_feat = pd.merge(u_feat, pivot, on=keys, how='left')
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⑦和⑧提取出来的数据直接计算。

```
# 9.在多少不同商家领取优惠卷核销率
u_feat[prefixs + 'receive_differ_Merchant_consume_rate'] = list(map(
    lambda x, y: x / y if y != 0 else 0, u_feat[prefixs + 'receive_and_consume_differ_Merchant_cnt'],
    u_feat[prefixs + 'receive_differ_Merchant_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。

```
# 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。由于XGBoost训练时只能接受int，float等数据类型，所以要修正数据类型。

(1)构造数据集，删除共有特征列即基础特征。

```
# 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()) &
                        set(simple_feat.columns.tolist())
                        )
label_field.index = range(len(label_field))
dataset = pd.concat([label_field, history_feat.drop(share_characters, axis=1)], axis=1)
dataset = pd.concat([dataset, simple_feat.drop(share_characters, axis=1)], axis=1)
```

图十二、构造数据集

在这里我将 Baseline_o2o 中的 simple_feat 沿用，用来提高 XGBoost 的训练准确率。

(2)删除无用属性并将 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。

```
# 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
    # 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)
    dataset['label'] = label
```

图十三、训练集和验证集的特征清洗

测试集就仅需清除基础特征即可。

```
else:  # 表示测试集
    dataset.drop(['Merchant_id', 'Discount_rate','date_received'], axis=1, inplace=True)
```

图十四、测试集的特征清洗

(3)修正数据类型

由于 XGBoost 的训练只接受指定参数，故将数据全部转为整型。

```
# 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():  # 训练集和验证集
    dataset['label'] = dataset['label'].map(int)
```

图十五、修正数据类型

(4)去重并重置索引

```
# 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))
```

图十六、去重并重置索引。

4. 进行 XGBoost 模型训练

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。XGBoost 参数沿用 Baseline_o2o 中的参数设置，并将训练次数改为 5167 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
# 设置为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=5167, evals=watchlist)
# 预测
predict = model.predict(dtest, validate_features=False)
```

图十八、训练并预测

处理预测结果并整合出提交结果 csv 文件需要的形式。

```
# 处理结果
predict = pd.DataFrame(predict, columns=['prob'])
result = pd.concat([test[['User_id', 'Coupon_id', 'Date_received']], predict], axis=1)
```

图十九、处理预测结果并整合形式

提取特征重要性矩阵并返回。

```
# 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
```

图二十、提取特征重要性矩阵

5. 输出训练评分及特征重要性程度。

(1)特征工程仅采用用户特征群的 9 个特征的训练情况。

仅采用用户特征群训练出的 AUC 情况极低，最高只达到了 0.5882。

图二十一、仅采用用户特征群的模型训练 AUC 情况

再输出特征重要性矩阵

仅包含用户特征群九个特征的特征重要性矩阵

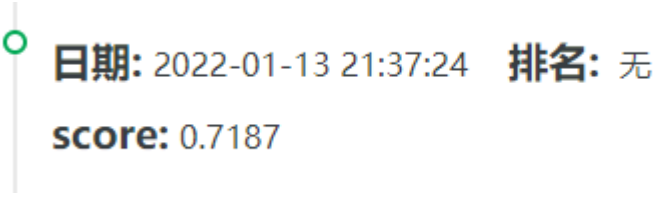
	feature_name	importance
9	discount_rate	16455
11	Distance	12954
1	history_field_User_id_receive_and_consume_mean...	12932
0	min_cost_of_manjian	12853
15	history_field_User_id_receive_cnt	10944
2	weekday_receive	8702
4	history_field_User_id_receive_not_consume_cnt	8580
5	history_field_User_id_receive_and_consume_rate	7584
6	history_field_User_id_receive_and_consume_mean...	7316
12	history_field_User_id_receive_differ_Merchant_cnt	7015
7	history_field_User_id_receive_and_consume_cnt	6122
8	history_field_User_id_receive_differ_Merchant_...	3669
14	week_3.0	2813
10	week_2.0	1885
21	week_1.0	1882
3	isManjian	1687
16	week_4.0	1636
22	is_null_distance	1592
17	week_5.0	1430
13	history_field_User_id_receive_and_consume_diff...	1240
20	is_weekend	1100
19	week_0.0	1077
18	week_6.0	808

图二十二、仅采用用户特征群的特征重要性

由该图可以看出， 提取的 9 个特征几乎不怎么占比重，只有用户平均消费距离和用户领卷数占到了一定的比重，由此可知，仅将该特征群做特征工程是远远不够的，模型训练效果会很差。

(2)沿用 Baseline_o2o 中的 simple_feature 和 week_feature 做训练

将 simple_feature 和 week_feature 也继承过来，并与用户特征群的 9 个特征合并做特征训练。得到的模型 AUC 评估明显好转，达到了 0.7187。



图二十三、融合特征后的模型评估分数

再输出此时的模型特征重要性矩阵。

包含simple_feat和week_feat和用户特征群9个特征的模型特征重要性矩阵

	feature_name	importance
10	discount_rate	15147
9	simple_User_id_receive_cnt	14903
2	simple_User_id_Coupon_id_receive_cnt	13016
0	min_cost_of_manjian	10739
3	Distance	9264
4	history_field_User_id_receive_cnt	7479
7	history_field_User_id_receive_and_consume_mean...	7106
8	weekday_receive	6889
6	history_field_User_id_receive_not_consume_cnt	6148
14	simple_User_id_Date_received_receive_cnt	6019
11	history_field_User_id_receive_and_consume_rate	4978
17	history_field_User_id_receive_differ_Merchant_cnt	4641
5	history_field_User_id_receive_and_consume_cnt	4177
19	history_field_User_id_receive_and_consume_mean...	3773
1	history_field_User_id_receive_differ_Merchant_...	2215
24	week_3.0	2204
12	week_4.0	1509
18	simple_User_id_Coupon_id_Date_received_receive...	1499
15	isManjian	1459
25	week_2.0	1341
20	week_1.0	1292
22	is_null_distance	1226
23	week_5.0	1181
27	is_weekend	1007
21	week_0.0	683
16	week_6.0	626
13	simple_User_id_Coupon_id_Date_received_repeat_...	573
26	history_field_User_id_receive_and_consume_diff...	570

图二十四、融合特征后的特征重要性矩阵

可以看到训练模型效果变好后，在模型训练过程中占比较大的都是simple_feature 里的特征， 用户特征群里的特征仅占了较少比重。故后续调整模型参数和重新构建特征工程时应适当抛弃和保留特征。

在查阅资料后得知，simple_feature 是提取了用户在预测域的一系列用户特征，如果是正常训练的情况下，不会如此有效提高优惠卷预测准确率。但在本次 020 优惠卷预测项目中，数据集提供了最终预测域的一系列情况，这

相当于对实际业务的一个真实情况泄露，在实际业务的时候是没有办法提取到这部分数据的。由于是在目标预测区间泄露了一系列用户的操作行为，故此可以有效将 AUC 评分提升 15%-20%。

6.总结

在提取了用户特征群的 9 个参数后，模型训练效果不太理想，可能是该特征并不能有效表达用户使用优惠卷的可能性，也可能是丢失了最能表现可能性的重要特征。在后续调整参数和重构特征工程时应当重新思考如何构建特征工程，再提取商家、优惠卷特征群的特征，做出交叉特征群，做好特征工程。同时也要注意好好利用预测域的一系列特征情况，能够有效提升 AUC 分数。

在模型训练方面，考虑到一个 XGBoost 模型可能不是能非常好的预测使用概率，应该多采用几个不同的训练模型，输出不同的模型得到的不同的使用概率。然后再用一个模型对这些整合出来的使用概率再做一次训练，起到模型融合的效果。同时也可以适当调节 XGBoost 参数，改变其学习率和收敛速度。

任务一总结（心得体会）

通过一系列对 python 的学习，回顾了 python 的使用语法和 numpy、pandas、matplotlib 等库的使用方式，熟悉了一个数据挖掘项目的整体流程，但在学习的过程中仍存在大量的疑问和问题未能处理，阅读了相关博客和资料，解决了部分问题，但仍未能将模型训练出很好的成果。

在学习的过程中，我认识到了光会调用库和调参数是不够的，要尽量去了解机器学习的原理和底层逻辑，才能更好地在训练模型过程中发挥一个优秀的机器学习算法的威力。各种的机器学习算法能发挥不同的作用，应当在当前最适用的情况下挑选最正确的算法和模型。比如 Kmeans 中大数据集就运行时长过长，且特征过多时必须采取 PCA 降维才可以达到良好的聚类成果。

认识到了机器学习的各个算法和模型不是独立的，他们是可以相互应用和结合的。每个算法和模型都是实现一定的功能，在使用其他模型和算法时可以沿用以达到自己的目的。算法和模型都是辅助条件，要做好一个数据挖掘的项目，最关键的应该是合理设计项目流程，探索数据情况，分析特征关联性和要做的项目目标，合理构建特征工程。只有这样才能充分利用数据中每一条特征和数据项的作用，达到良好的项目成果。

最重要的应该是认识数据， 探索数据， 挖掘数据。之后我应该积累处理数据的经验，查阅项目背景和相关资料，合理分析数据。

本科期间我一直在做的都是数据分析，此次感受到了数据挖掘的魅力，之后应该再多学习、多实验。