《机器学习》

-无监督学习

华为技术有限公司

目录

1 实验介绍

无监督学习：主要使用无标记类别信息的样本训练模型，从而实现样本群体类别划分、样本关联抽取和主成分分析，最终实现群体细分、关系计算和成份分析的目的。

本章实验难度分为中级和高级。

中级实验：消费者聚类、购物篮分析、电信用户分析

高级实验：运营商套餐推荐、红酒聚类实验

1.1 实验目的

本章实验的主要目的是掌握机器学习的无监督算法，包括聚类算法和关联规则算法相关原理及业务应用，了解无监督方向的聚类算法（KMeans）、主成份分析算法（PCA）和关联规则算法（FP-growth）等。熟悉使用 Jupyter notebook 学习框架，掌握数据挖掘的一般建模流程。

1.2 实验清单

实验	简述	难度	软件环境	开发环境
消费者聚类	基于消费者购物数据，使用KMeans算法进行聚类分析	中级	Python3.7、ModelArts	PC 64bit
电信用户分析	基于电信用户消费行为数据，使用KMeans聚类实现对不同人群的聚类分析	中级	Python3.7、ModelArts	PC 64bit
购物篮分析	基于购物篮数据，使用关联规则算法Apriori实现频繁模式挖掘，挖掘商品之间的内在关系	中级	Python3.7、ModelArts	PC 64bit
运营商套餐推荐	基于运营商套餐数据，使用关联规则算法实现频繁套餐组合挖掘	高级	Python3.7、ModelArts	PC 64bit

红酒聚类实验	基于wine数据集，使用KNN算法实现不同品种的聚类	高级	ModelArts、MindSpore	PC 64bit

2 电信用户分析

2.1 实验介绍

2.1.1 简介

电信客户信用度是指根据客户在网时间、缴费情况、客户积分等相关要素对集团客户或个人客户进行评分，用于衡量客户缴费行为的好坏。

本案例使用电信用户的通信行为数据集，目的是进行用户信用分析且最终对客户进行分群。由于是没有标注的训练样本，特征仅 7 个维度，在大数据挖掘实验中并非特征较多的情景。本实验重点在掌握降维算法的实践，以及对降至 3 维后的特征进行可视化呈现，最后利用聚类这一无监督学习算法将用户进行分群，针对不同客户群体再结合原始业务指标进行分析，确定每个聚类群体的信用行为特点。

数据集：(data\telecom.csv.csv)

2.1.2 实验目的

通过案例实践，掌握数据分析挖掘的基本流程，及通过聚类实现群体细分和通过 PCA 降维实现主成分分析的基本思想，并结合实践操作加深理论的理解和认识，为实际的业务应用奠定理论和技术基础。

2.2 实验环境要求

本实验在 python3.7 环境下完成，可下载 Anaconda，下载地址为：https://www.anaconda.com/distribution/。

另外需要安装 seaborn、imblearn，pandas，sklearn 使用 pip 安装命令如下：

```
pip install seaborn
pip install numpy
pip install sklearn
pip install pandas
```

其他安装包如上进行安装。

2.3 实验总体设计

本实验遵从数据挖掘的一般流程，首先对已经下载本地的数据进行读取，常规的探索后，进行数据降维操作，随后直接选择 sklearn 模块中的 KMeans 算法进行建模，通过轮廓系数对 KMeans 算法进行 K 值调优，选择最优的 K 值进行模型训练，最后通过统计分析对聚类结果进行画像分析，从而明确每一类人群的特征。

2.4 实验详细设计与实现

2.4.1 导入实验环境

步骤 1 导入相应的模块

本实验使用到的框架主要包括 numpy，pandas，scikit-learn，matplotlib，seaborn 库。scikit-learn 库是 Python 的机器学习库，提供一些常用的机器学习算法模型及模型参数优化功能；numpy ，pandas 库是 Python 中结构化数据处理的库，主要用于结构化数据的统计分析及操作；matplotlib，seaborn 主要用于数据分析过程的可视化展示。

```
#加载 Python 库
import numpy as np
#加载数据预处理模块
import pandas as pd
#加载绘图模块
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style(style="darkgrid")
```

2.4.2 数据准备

步骤 1 离线数据读取

这里读取的数据是与项目文件同级目录下，或同一个文件夹中。

```
#encoding=utf8
#  模块导入与数据读取
X = pd.read_csv(r"data\telecom.csv",encoding="utf8")
X.head(2)
```

输出如下结果：

	入网时间	套餐价格	每月流量	每月话费	每月通话时长	欠费金额	欠费月份数
0	27.0	389.0	140.198242	390.0	14.325	0.0	0.0
1	29.0	159.0	0.000000	5.0	0.000	5.0	1.0

图2-1 数据集前 2 行

步骤 2 数据标准化

通过如下代码，实现数据集的标准化处理。

```
# 数据标准化
from sklearn import preprocessing
X_scaled = preprocessing.scale(X)
X_scaled
```

输出如下结果：

```
array([[-0.32484896,  1.63037135, -0.14052452, ..., -0.5953273 ,
        -0.2199407 , -0.31623138],
       ...,
       [-0.12245085, -0.29646995, -0.16747396, ...,  0.20572063,
        -0.2199407 , -0.31623138]])
```

2.4.3 数据降维

步骤 1 使用 PCA 进行数据降维

通过降维实现主要特征提取，为后续的聚类提供数据准备。

```
# 使用 PCA 进行数据降维
from sklearn.decomposition import PCA
#此处的主成分维度我们人为设定为 3，对于属性较少的数据集，属于常规会选择的维度数，后面也会看到，这个也是出于可以可视化的需求
pca = PCA(n_components = 3)
#将设置了维数的模型作用到标准化后的数据集并输出查看
X_pca = pca.fit_transform(X_scaled)
X_pca_frame = pd.DataFrame(X_pca,columns=['pca_1','pca_2','pca_3'])
X_pca_frame.head(5)
```

将降维后的数据前 5 行显示出来如下：

	pca_1	pca_2	pca_3
0	1.549988	-0.211833	-1.038576
1	-1.034264	0.257341	-0.385973
2	-0.244564	-0.431907	-1.222455
3	-0.151285	-0.400462	-1.430211
4	-1.163608	-0.439145	0.526530

图2-2 降维后指标

步骤 2　降维指标与原指标关联

这三个指标与原始字段的系数可以通过 pca 的 components_属性获取。

```
#三个指标与原始字段的系数获取
pd.DataFrame(pca.components_,columns = X.columns, index=['pca_1','pca_2','pca_3']).T
```

降维后生成的 3 个特征'pca_1','pca_2','pca_3'与原始的特征之间的系数关系如下图表，这一步并不是实验数据挖掘的必要环节，但是对理解 PCA 的变换效果非常有必要。

输出如下结果：

	pca_1	pca_2	pca_3
入网时间	-0.106231	-0.151105	0.814896
套餐价格	0.416307	-0.093391	-0.375968
每月流量	0.541932	-0.057296	0.034052
每月话费	0.570941	0.084457	0.068408
每月通话时长	0.430969	-0.128013	0.410924
欠费金额	0.099574	0.677448	0.095135
欠费月份数	-0.009541	0.694771	0.104136

图2-3 指标与原始字段的系数

步骤 3　对不同的 K 值进行计算，筛选最优的 K 值

当 K 值设置不同时，聚类效果会存在较大的差别，因此在实际应用中需要依据聚类效果，对 K 值进行优化。执行如下代码，通过 Calinski-Harabasz 分数值对不同的 K 值进行评估。

```
# 对不同的 K 值进行计算，筛选最优的 K 值
from mpl_toolkits.mplot3d import Axes3D
from sklearn import metrics
from sklearn.cluster import KMeans
#KMeans 算法实例化，将其设置为 K=range(2,14)
```

```
d={}
fig_reduced_data=plt.figure(figsize=(12,12))
for k in range(2,14):
    est=KMeans(n_clusters=k,random_state=111)
    #作用到降维后的数据集上
    y_pred=est.fit_predict(X_pca)
    #评估不同 K 值聚类算法效果
    score=metrics.calinski_harabasz_score(X_pca_frame,y_pred)
    d.update({k: score})
    print('calinski_harabasz_score with k={0} is {1}'.format(k,score))
```

输出如下结果：

```
calinski_harabaz_score with k=2 is 3407.50124972
calinski_harabaz_score with k=3 is 2748.05780001
calinski_harabaz_score with k=4 is 5022.47175241
calinski_harabaz_score with k=5 is 4456.76540543
calinski_harabaz_score with k=6 is 2948.96211991
calinski_harabaz_score with k=7 is 4891.93078587
calinski_harabaz_score with k=8 is 4286.07061391
calinski_harabaz_score with k=9 is 43434.6693977
calinski_harabaz_score with k=10 is 54600.4910359
calinski_harabaz_score with k=11 is 15385.5000285
calinski_harabaz_score with k=12 is 6215.94606742
calinski_harabaz_score with k=13 is 5611.65516489
```

图2-4 不同 K 取值聚类效果评估

步骤 4　绘制不同 k 值对应的 score，找到最优的 K 值

```
x=[]
y=[]
for k,score in d.iteritems():
    x.append(k)
    y.append(score)

plt.plot(x,y)
plt.xlabel('k value')
plt.ylabel('calinski_harabasz_score')
```

输出如下结果：

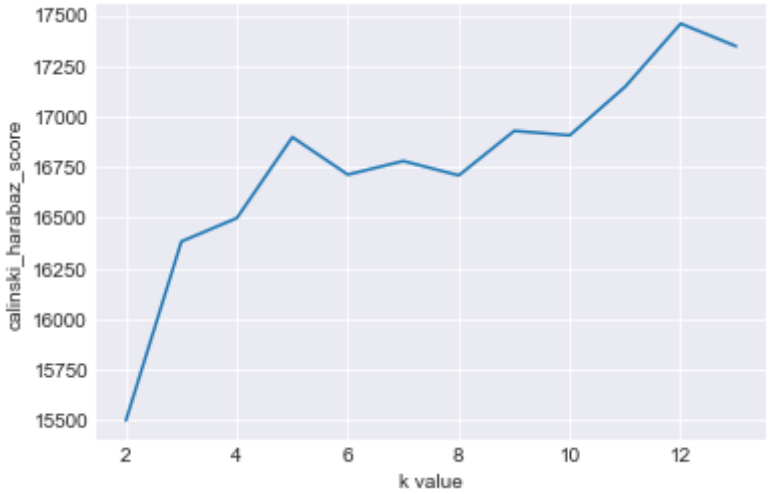


图2-5 聚类效果 Calinski-Harabasz 分数值

从聚类结果的图片可以看出，当 K 为 12 时，聚类效果较好。

2.4.4 K-Means 聚类模型训练

在降维的数据上使用 K-means 聚类算法将数据聚成 10 类，这里的 10 类是事先人工指定的划分群数，通常是结合后续的业务运营方案（如有 10 类运营方案支持划分 10 类），或是结合业务专家的建议凭经验指定，或者是指定不同的 K 值后比较得来。代码如下：

步骤 1 训练简单模型

```
#  K-means 聚类建模
#运行 KMeans 聚类算法
from sklearn.cluster import KMeans
#此处指定 K=12
est = KMeans(n_clusters=12)
est.fit(X_pca)
#获取数据标签值
kmeans_clustering_labels = pd.DataFrame(est.labels_,columns=['cluster'])
#将聚类结果与降维特征数据进行拼接
X_pca_frame = pd.concat([X_pca_frame,kmeans_clustering_labels], axis=1)
X_pca_frame.head(5)
```

输出如下结果：

	pca_1	pca_2	pca_3	cluster
0	1.549988	-0.211833	-1.038576	5
1	-1.034264	0.257341	-0.385973	0
2	-0.244564	-0.431907	-1.222455	6
3	-0.151285	-0.400462	-1.430211	6
4	-1.163608	-0.439145	0.526530	1

图2-6 聚类示例

步骤 2　样本筛选

通过对聚类数据进行筛选，删除每一个聚类中的噪音和异常点，提取每一个聚类中的典型用户。为了避免离群点的影响，按照每一类的分位数筛选离中心点较近的用户作为该类的代表用户。这里我们引入了一个经典的箱型图筛选技术。代码如下：

```
#样本筛选
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
X.index = X_pca_frame.index
X_full = pd.concat([X, X_pca_frame],axis=1)
grouped = X_full.groupby('cluster')
result_data = pd.DataFrame()
#此处利用箱型图进行异常值分析
for name,group in grouped:
    print (name,group['pca_1'].count())
    desp = group[['pca_1','pca_2','pca_3']].describe()
    for att in ['pca_1','pca_2','pca_3']:
        lower25 = desp.ix['25%',att]
        upper75 = desp.ix['75%',att]
        IQR = upper75 - lower25
        min_value = lower25 - 1.5*IQR
        max_value = upper75 + 1.5*IQR
        group = group[(group[att] > min_value) & (group[att] < max_value) ]
    result_data = pd.concat([result_data, group],axis=0)
    print(name,group['pca_1'].count())
#分别列出 筛选前后每个特征的数量
print('remanin sample : ',result_data['pca_1'].count())
```

筛选前后的用户样本按信用类别列举如下：

```
0 11129
0 9530
1 6613
1 5844
2 2021
```

```
2 1500
3 1947
3 1606
4 47
4 30
5 2891
5 2218
6 3602
6 2808
7 21
7 20
8 369
8 297
9 1360
9 1248
```

查看仍保留下的数据数量

```
remanin sample :  25101
```

2.4.5 聚类结果分析

步骤 1 查看降维数据

代码如下：

```
#1)主成分
components_frame = pd.DataFrame(pca.components_,index=[u'主成分 1',u'主成分 2',u'主成分 3'],columns=X.columns)
#2）降维数据和聚类初步结果
X_full.groupby('cluster').describe()
```

结果如下：

	pca_1								pca_2		...	每月话费		每月通话时长	
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean
cluster															
0	2628.0	0.576172	0.570799	-0.385017	0.136194	0.431924	0.908006	2.770590	2628.0	-0.351355	...	163.497083	687.673333	2628.0	790.814897
1	9179.0	-0.853999	0.305842	-1.507802	-1.072522	-0.958371	-0.712261	0.311594	9179.0	-0.129899	...	62.120000	295.000000	9179.0	67.702144
2	369.0	0.130412	1.415525	-1.196165	-0.655300	-0.528785	0.338870	5.864293	369.0	7.150453	...	189.000000	593.308333	369.0	49.615349
3	1594.0	4.390803	1.582823	2.615001	3.689127	4.105464	4.639750	25.062341	1594.0	-0.370417	...	418.030417	1795.206667	1594.0	1127.480556
4	1194.0	-0.710311	1.110725	-2.107839	-1.492433	-1.261561	-0.107465	3.758484	1194.0	-0.909106	...	108.212917	480.735000	1194.0	406.981789

图2-7 降维后数据示例

代码如下：

```
#筛选样本聚类统计
result_data.groupby('cluster').describe()
```

结果如下：

	pca_1								pca_2		...	每月话费		每月通话时长	
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean
luster															
0	4480.0	0.522515	0.512841	-0.385017	0.121566	0.389860	0.859951	2.058277	4480.0	-0.486538	...	155.022083	534.320000	4480.0	785.243316
1	15696.0	-0.903714	0.249449	-1.507802	-1.082547	-0.984155	-0.791014	-0.173087	15696.0	-0.214780	...	44.099167	176.666667	15696.0	56.826149
2	594.0	-0.505442	0.438998	-1.196165	-0.686981	-0.546216	-0.306443	1.277095	594.0	7.185686	...	129.000000	406.333333	594.0	2.906453
3	2258.0	4.114018	0.612364	2.615001	3.673931	4.024502	4.483814	6.004661	2258.0	-0.635338	...	411.896667	620.083333	2258.0	993.371086
4	2168.0	-0.867451	0.894138	-1.961428	-1.498749	-1.333605	-0.318820	1.941573	2168.0	-0.908209	...	93.994167	346.796667	2168.0	325.984089
5	3190.0	-0.879264	0.192266	-1.354796	-1.031019	-0.827030	-0.797812	-0.334813	3190.0	2.087987	...	90.000000	240.000000	3190.0	28.675883

步骤 2　聚类结果可视化

代码如下：

```
#绘图
cluster_2_color = {0:'red',1:'green',2:'blue',3:'yellow',4:'cyan',5:'black',6:'magenta', 7:'#fff0f5', 8:'#ffdab9',9:'#ffa500' }
#1. 原始数据降维后的可视化图
fig_reduced_data = plt.figure()
ax_reduced_data = fig_reduced_data.add_subplot(111, projection='3d')
ax_reduced_data.scatter(X_pca_frame['pca_1'].values,X_pca_frame['pca_2'].values,X_pca_frame['pca_3'].values)
ax_reduced_data.set_xlabel('Component 1')
ax_reduced_data.set_ylabel('Component 2')
ax_reduced_data.set_zlabel('Component 3')
```

结果如下：

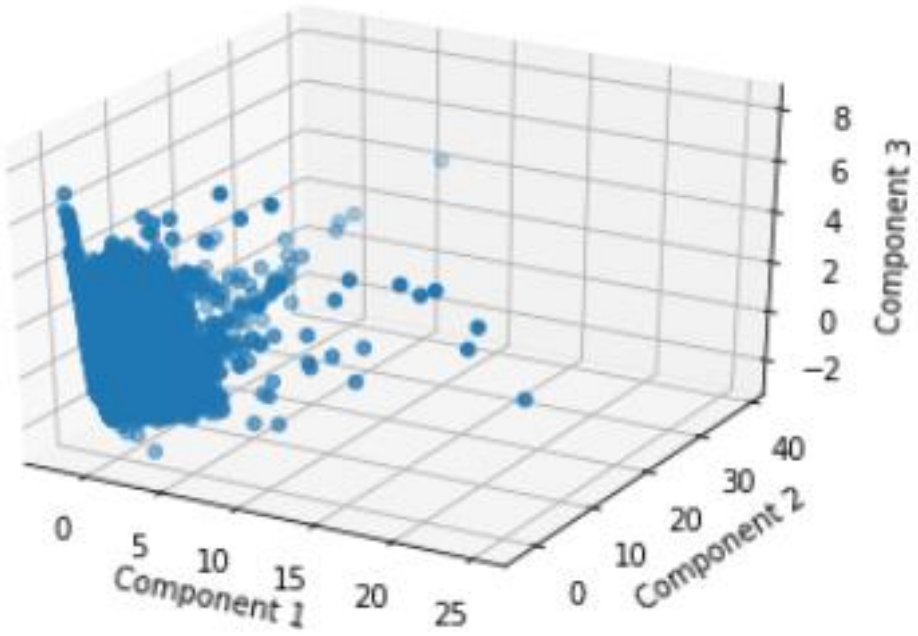


图2-8 原始数据降维后的可视化图

对不同群分别指定不同颜色表示，代码如下：

```
#2. 聚类之后的数据可视化图
colors_clustered_data = X_pca_frame['cluster'].map(cluster_2_color)
fig_clustered_data = plt.figure()
ax_clustered_data = fig_clustered_data.add_subplot(111, projection='3d')
ax_clustered_data.scatter(X_pca_frame['pca_1'].values,X_pca_frame['pca_2'].values,X_pca_frame['pca_3'].values,c=colors_clustered_data)
ax_clustered_data.set_xlabel('Component 1')
ax_clustered_data.set_ylabel('Component 2')
ax_clustered_data.set_zlabel('Component 3')
```

结果如下：

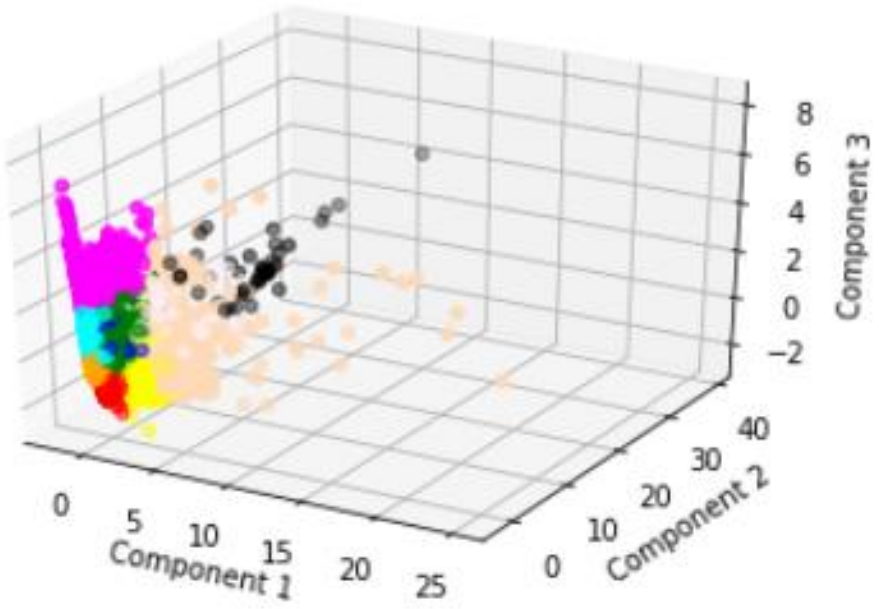


图2-9 聚类之后的数据可视化图

查看去除异常值后的不同群，代码如下：

```
#3. 筛选后的数据聚类可视化图
colors_filtered_data = result_data['cluster'].map(cluster_2_color)
fig_filtered_data = plt.figure()
ax_filtered_data = fig_filtered_data.add_subplot(111, projection='3d')
ax_filtered_data.scatter(result_data['pca_1'].values,result_data['pca_2'].values,result_data['pca_3'].values,c=colors_filtered_data)
ax_filtered_data.set_xlabel('Component 1')
ax_filtered_data.set_ylabel('Component 2')
ax_filtered_data.set_zlabel('Component 3')
```

结果如下：

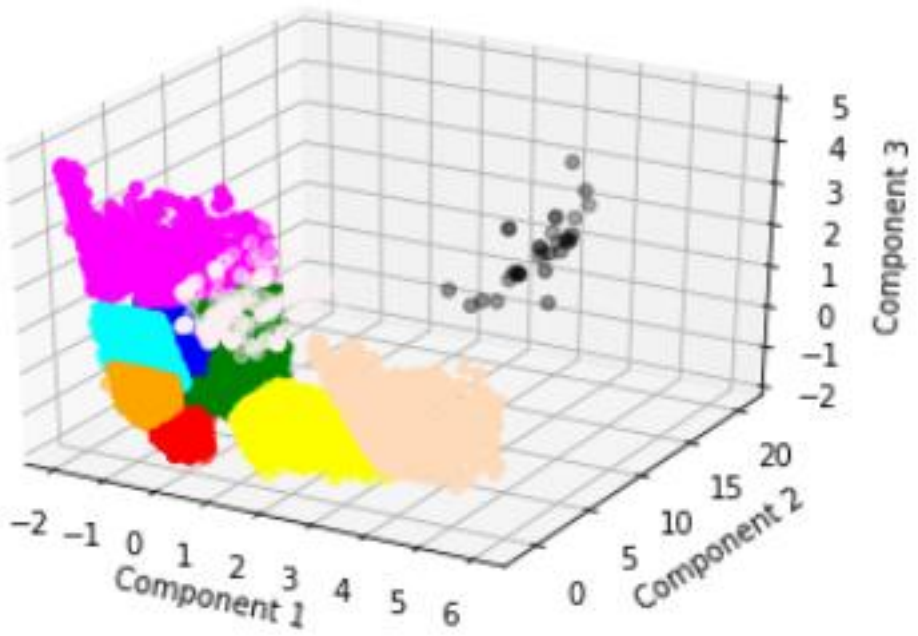


图2-10 筛选后的数据聚类可视化图

代码如下：

```
#4. 每个聚类的在 pca1,pca2,pca3 上的均值可视化图
grouped = result_data.groupby('cluster')
#每一个聚类的盒图
colors = result_data['cluster'].map(cluster_2_color)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(result_data['pca_1'].values,result_data['pca_2'].values,result_data['pca_3'].values,
c=colors)
ax.set_xlabel('1st component')
ax.set_ylabel('2nd component')
ax.set_zlabel('3rd component')
plt.show()
```

结果如下：

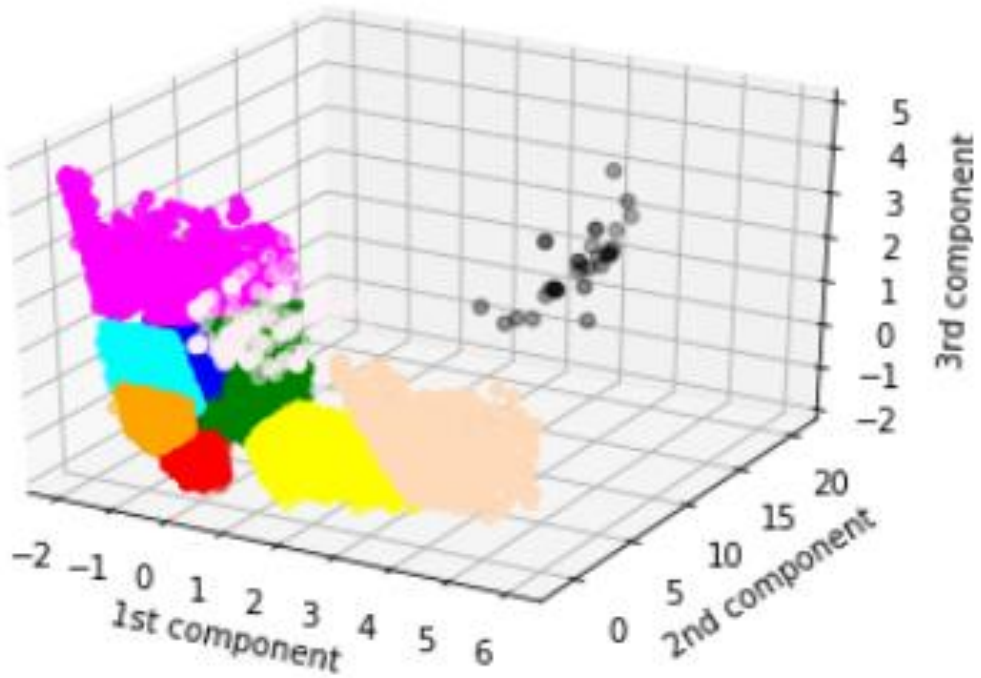


图2-11 每个聚类的在 pca1,pca2,pca3 上的均值可视化图

步骤 3　用户画像分析

对于 10 个聚类中的用户，通过"欠费金额","每月话费"和"入网时间"三个原始字段的均值来分析不同类用户的特点。为了显示正常，新建三个英文名命名的字段 arrearage(欠费金额)，telephone_fare(每月话费)和 month_of_access（入网时间）。代码如下：

```
X_full["arrearage"] = X_full[u"欠费金额"]
X_full["telephone_fare"] = X_full[u"每月话费"]
X_full["month_of_access"] = X_full[u"入网时间"]
%matplotlib inline
X_full.groupby("cluster")[["arrearage","telephone_fare","month_of_access"]].mea
n().plot(kind="bar")
```

结果如下：

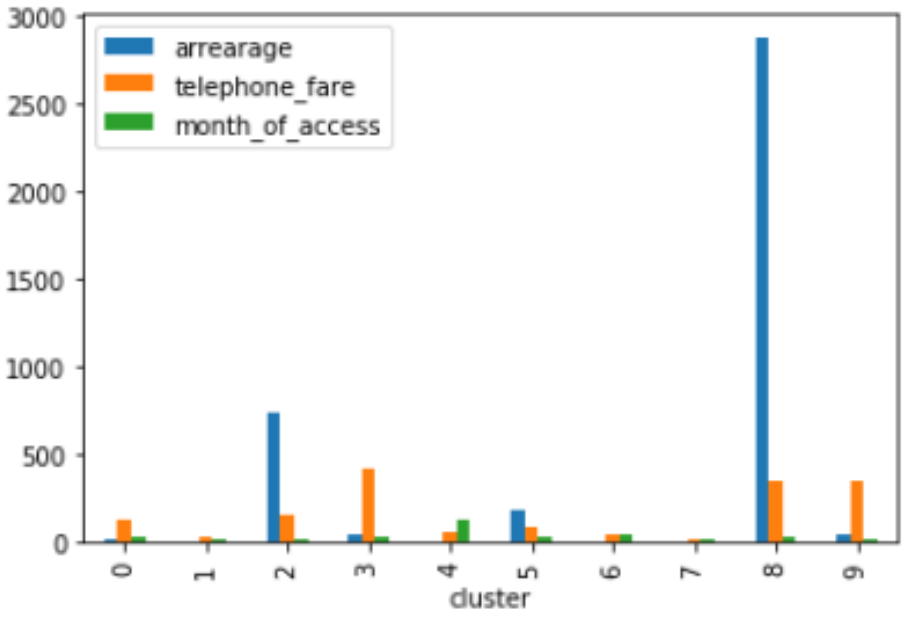


图2-12 用户分类结果

从上图可以分析出，第 4、10 类用户（下标为 3、9）入网时间较长，消费能力中等，但是基本无欠费行为，综合信用水平较好。第 3、9 类用户（下标为 2 与 8）欠费金额很高，消费能力也较强，且大部分入网时间较短，尤其是对于第 9 类用户属于需要进一步细分和评估，其中会存在一些劣质用户。第 1、5、6 类用户消费水平一般，但有较长的入网时间，可以尝试进一步营销活动或者防止流失策略。

注：由于 K-means 聚类具有随机性（每次初始质心的选择不同造成），不同的运行结果可能会与上图存在差异。

2.5 思考题

1、无监督学习适应场景？

2、 K-means 算法中 K 的确定还有哪些方法？

【答案】

2.6 实验小结

本章主要讲解了电信用户实际业务场景下的数据聚类实验的相关操作，包含数据的标准化，数据降维、建模，最终的用户分析等过程。

2.7 思考题-汇总

1、无监督学习适应场景?

2、 K-means 算法中 K 的确定还有哪些方法?

【参考答案】

1、无监督学习适用于数据集无标签的情况。无监督学习采用输入集，并尝试查找数据中的模式。比如，将其组织成群（聚类）或查找异常值（异常检测）。

2、层次聚类法、稳定性方法等。此处我们实验指定k=10是否可以调整，尝试看下指定不同k值，然后观察聚类后的效果如何，选择聚类效果好的k值。（答案不唯一）

3 购物篮分析

3.1 实验说明

3.1.1 简介

关联规则最初提出的动机是针对“购物篮分析”问题提出的。它的主要任务就是设法发现事物之间的内在联系。算法本质上是利用 Apriori 原理和其逆否命题，如果一个项集是非频繁的，那么它的所有超集也是非频繁的，频繁项集就是支持度大于最小支持度的集合。

该挖掘分析可以通过筛选出强相关性产品组合，用于相应的推荐场景等。本实验操作总体上，包括第 1 部分构建频繁项集和第 2 部分挖掘关联规则两个部分。利用 Apriori 算法（先验原理）对用户购物数据进行分析。

数据来源：Online Retail.xlsx

3.1.2 实验目的

通过本案例实践，掌握频繁模式挖掘的基本思路、应用和对业务数据的要求，同时结合实操加深对算法应用的理解，为后续的实际应用奠定理论和技术基础。

3.2 实验环境要求

本实验在 python3.7 环境下完成，可下载 Anaconda，下载地址为：https://www.anaconda.com/distribution/。

另外需要安装 seaborn、imblearn，pandas，mlxtend 使用 pip 安装命令如下：

```
pip install seaborn
pip install numpy
pip install mlxtend
pip install pandas
```

其他安装包如上进行安装。

3.3 实验总体设计

本实验遵从数据挖掘的一般流程，首先对已经下载本地的数据进行读取，常规的探索后，进行基本的数据预处理操作，随后直接选择 mlxtend 模块中的 Apriori 算法进行建模分析，最终通过调整支持度和置信度，获取不同的商品组合，为套餐设计提供数据准备。

3.4 预备知识

我们的目标是分析每种商品的关联关系。这里用 A，B，C，D，E 各表示一种商品，如下项集图），在实验操作中为了描述简化，商品都用数字表示，如列表 [[1,2],[1,3,0],[0,1],[0,2],[1,2,3,0]] 就是一组由 5 个购物单组成的数据对象（根据集合元素知识，可知有 31 种集合可能），其中每个子列表代替一个购物单（如[1,3,4] 表示一笔购物同时买了商品 1，3，4），自然的，当商品的种类越来越多，集合的组成数是指数级增长的，在计算处理中，工作量比较大，利用相关算法，如 Apriori，帮助我们在分析过程中减少计算复杂度。如下图，常用来表示所有可能的项集，Apriori 算法就是通过一种不用计算支持度，事先删除掉某些项集，减少候选项集从而达到降低计算复杂度的目的。

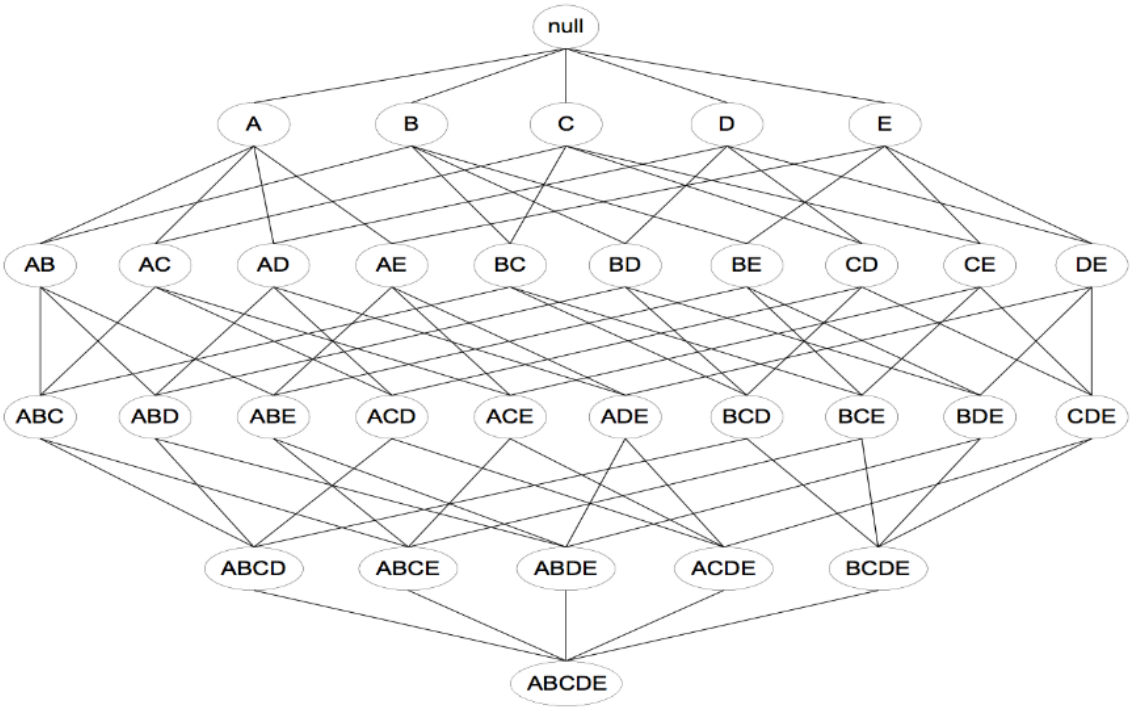


图3-1 Apriori 算法中集合关联关系

3.5 实验任务操作指导

3.5.1 数据准备

实验中所用的数据集来自 UCI Machine Learning repository，名称为“Online Retail”。它来自 01/12/2010 - 09/12/2011 英国一家网上零售商量的共 541909 条交易记录。

它包含的属性特征信息解释如下：

InvoiceNo：发票号，6 位连续整数，唯一对应每个发票，如果是 C 开头，表示该笔交易取消；

StockCode：产品代码，一个唯一对应每个商品的 5 位整数；

Description：商品描述，标称属性；

Quantity：每笔交易的每个产品（实验）的数量，是数值类型；

InvoiceDate：发票生成的日期，数值；

UnitPrice：英镑单位商品单价，数值；

CustomerID：用户 id,唯一对应每位顾客的 5 位数字，标称属性；

Country：国家名称，记录每位客户声明的归属国。

3.5.2 实验步骤

步骤 1　导入相应模块

注意，此处引入一个由 Sebastian Raschka 提供的具有 Apriori 算法的 MLxtend 库，以方便我们进一步分析数据，可以使用 pip 安装 MLxtend，只有安装了 MLxtend 下面的代码才能真正运行，代码如下：

```
#coding: utf-8
#导入相关的包，注意 mlxtend 来自第三方包
```

步骤 2　读取并显示数据集

代码如下：

```
#读取数据，并显示数据前几行进行观察,利用 pandas 包可以直接读取 excel 一维表
df =_____                    #加载数据集
df.head(10)
#数据集较大，处理会有延迟
```

结果如下：

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

表3-1 数据集前几行数据

步骤 3 数据探索

- 查看数据维度特征等，代码如下：

```
#查看数据的特征数，各特征数据类型等信息
print("The shape of the data:{}".format(df.shape))
df.info()
```

结果如下：

```
The shape of the data:(541909, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
InvoiceNo          541909 non-null object
StockCode          541909 non-null object
Description        540455 non-null object
Quantity           541909 non-null int64
InvoiceDate        541909 non-null datetime64[ns]
UnitPrice          541909 non-null float64
CustomerID         406829 non-null float64
Country            541909 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

可以看到该数据集合，有 54 万多条记录，即购买行为 8 个特征属性，且均没有空值。

- 处理异常数据，代码如下：

```
#通过数据解读，了解该交易记录数据不会存在合理负值，此处对负值进行处理：全部取绝对值处理
df.Quantity = _____          #对 Quantity 求绝对值
```

- 查看每张发票购买量的分布情况，代码如下：

```
#查看发票购买商品的数据分布：按照每笔发票的单号聚合数据
grouped_df =_____      #对'InvoiceNo'进行 groupby 操作，并对'Quantity'聚集求和
grouped_df.head()
grouped_df.plot.hist(bins=1000)
plt.xlim([-10,1000])
```

```
plt.xlabel('The quantity for each Invoice')
plt.legend(['Quantity of Invoice No'])
plt.title("The distribution of Quantity for each InvoiceNo")
```

可以看到，大量购物交易购买的商品总数主要集中在 100 件左右，随着购买数量的增加，交易数量迅速递减。

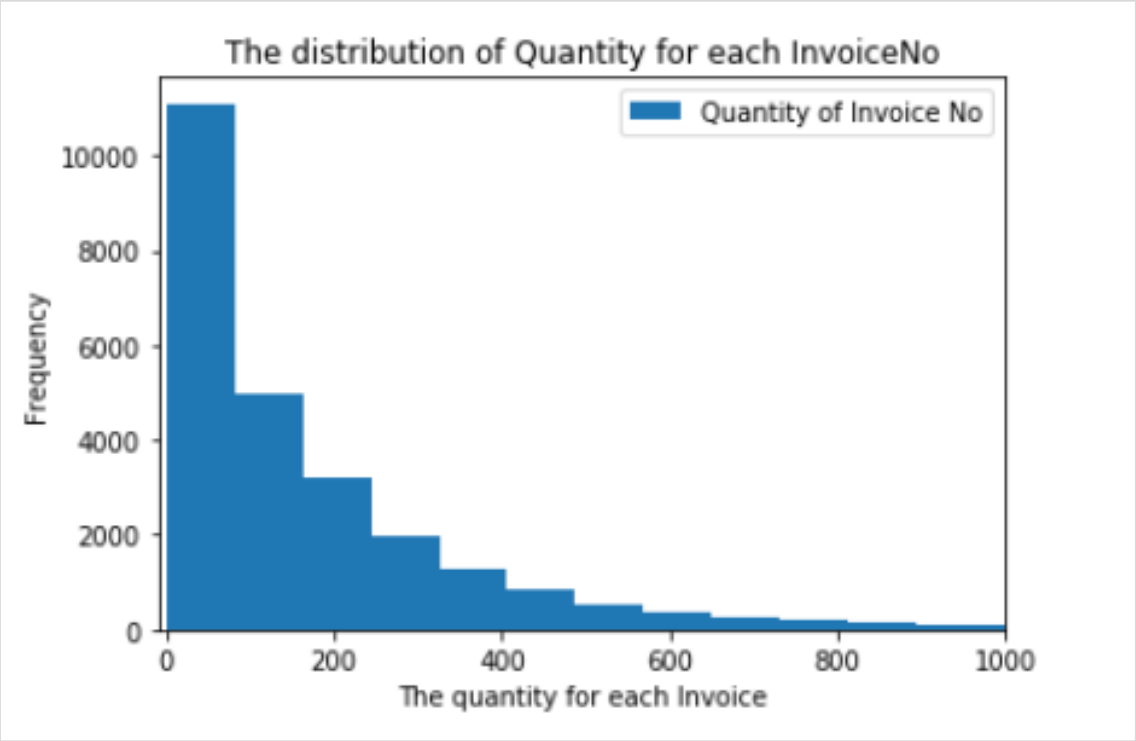


图3-2 每笔购买商品数量分布

- 查看全部商品中最畅销的 20 种物品

在正式使用算法建模之前，可以从多个角度尝试去了解商品的数据特性，代码如下：

```
#再查看交易清单里最畅销的前 20 款商品
item_df =_______  #对'StockCode','Description'进行聚合操作，并对'Quantity'进行求和聚合
item_df.set_index('Description').sort_values(by = 'Quantity',ascending = False)[0:20][::-1].plot.barh()
#item_df.Description.plot.barh()
```

结果如下：

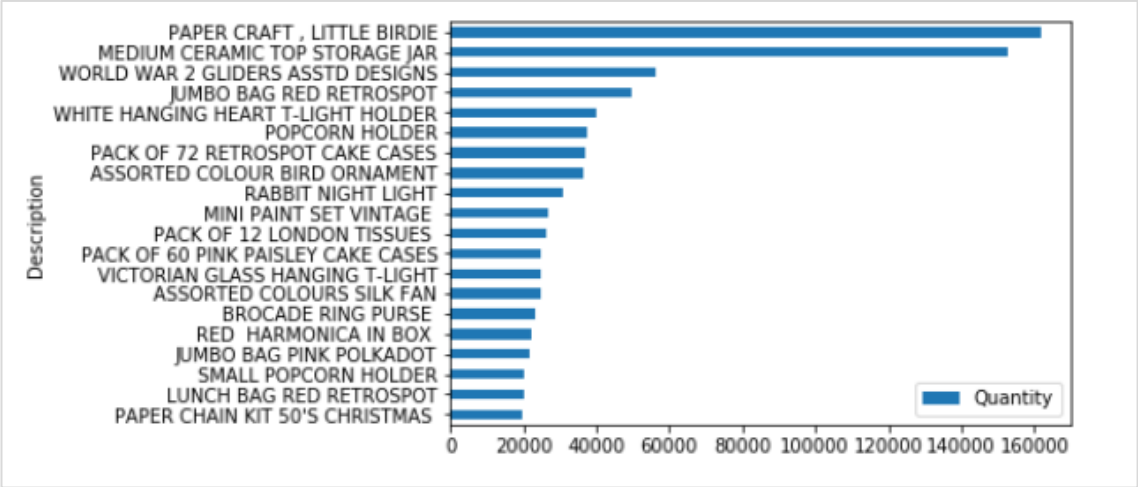


图3-3 最畅销的 20 种物品

步骤 4　数据预处理

- 去除空值，代码如下：

进入数据预处理阶段：清除商品描述中的空格和移除无效的发票号的记录

```
df.Description = df.Description.str.strip()
df._____　#将所有含有 nan 项的 row 删除，且原数组直接就被替换
#输出清除无效发票后剩余数据大小
print('The shape of dropna dataset:{}'.format(df.shape))
```

结果如下：

```
The shape of dropna dataset:(541909, 8)
```

- 去除发票信息中为“C”（支票支付）的数据，代码如下：

```
#移除数据中，发票号码开头为 C 的记录
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]
df.shape
```

结果如下：

```
(532621, 8)
```

- 读取法国部分的数据并按物品分组计算，代码如下：

鉴于 Python 的计算效率和 Aprior 算法的计算逻辑，此处为了将数据集的数据进一步缩小，只筛选出法国客户的消费记录进行下一步分析，即接下来我们的分析范围都是该网站此段时间法国消费者购买商品的相关关系，代码如下：

```
#查找国家标识=法国的记录，并且按照发票号和商品描述进行聚合
grouped_df =_____
#可以得到一个多索引的数据框，类似数据透视效果
```

```
grouped_df.head()
```

结果如下：

```
InvoiceNo   Description
536370       ALARM CLOCK BAKELIKE GREEN          12
             ALARM CLOCK BAKELIKE PINK           24
             ALARM CLOCK BAKELIKE RED            24
             CHARLOTTE BAG DOLLY GIRL DESIGN     20
             CIRCUS PARADE LUNCH BOX             24
Name: Quantity, dtype: int64
```

- 对分组数据的行旋转为列，代码如下：

```
#将上面多索引的数据框，通过转置拆分成单索引数据表形式
unstack_df =_____
unstack_df.head(10)
#将所有的空值填充为 0
fillna_df = unstack_df.reset_index().fillna(0)
#设置 InvoiceNO 列为索引
basket_df = fillna_df.set_index('InvoiceNo')
basket_df.head(5)
```

Description	50'S CHRISTMAS GIFT BAG LARGE	DOLLY GIRL BEAKER	I LOVE LONDON MINI BACKPACK	NINE DRAWER OFFICE TIDY	SET 2 TEA TOWELS I LOVE LONDON	SPACEBOY BABY GIFT SET	TRELLIS COAT RACK	1(COLOUF SPACEBOY PEN
InvoiceNo								
536370	0.0	0.0	0.0	0.0	24.0	0.0	0.0	0.(
536852	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
536974	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
537065	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
537463	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(

5 rows × 1565 columns

图3-4 转换后的单索引数据表

- 数据转换为 0-1 形式，代码如下：

```
#定量特征二值化:大于阈值 1 的设置为 1 ， 小于等于 0 的设置为 0
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1
basket_sets = basket_df.applymap(encode_units)
basket_sets.head(10)
```

结果如下：

Description InvoiceNo	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY
536370	0	0	0	0	0	0	0	0
536852	0	0	0	0	0	0	0	0
536974	0	0	0	0	0	0	0	0
537065	0	0	0	0	0	0	0	0
537463	0	0	0	0	0	0	0	0
537468	1	0	0	0	0	0	0	0
537693	0	0	0	0	0	0	0	0
537897	0	0	0	0	0	0	0	0

表3-2 0-1 转换后的数据集

随后，在二值化的基础上移除掉分析中不需要关注的列：邮费列，这一步非常重要，因为邮费作为商品列表一项非常频繁地出现，会干扰正常商品的频繁项组合结果。代码如下：

```
basket_sets.drop('POSTAGE', inplace=True, axis=1)#axis=0 表述列 ， axis=1 表述行
print('The shape of basket set:{}'.format(basket_sets.shape))
basket_sets.head()
```

结果如下：

```
The shape of basket set:(392, 1562)
```

以及

Description InvoiceNo	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	P
536370	0	0	0	0	0	0	0	
536852	0	0	0	0	0	0	0	
536974	0	0	0	0	0	0	0	
537065	0	0	0	0	0	0	0	
537463	0	0	0	0	0	0	0	

5 rows × 1562 columns

表3-3 移除邮费列后数据集

步骤 5　创建频繁集

现在有了合适的数据结构，我们可以在此基础上生成频繁的项目集，代码如下：

```
#此处先设置，最小的支持度为 0.08 为了先尽量保留多的商品项，最小支持度设置的较小
frequent_itemsets = _____
```

步骤 6　创建关联规则

最后一步，是生成根据相应的支持度、置信度和提升度生成的规则，代码如下:

```
rules = _____
rules.head(5)
```

结果如下：

	antecedants	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN SPACEBOY)	0.168367	0.137755	0.089286	0.530303	3.849607	0.066092	1.835747
1	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN CIRCUS PARADE)	0.137755	0.168367	0.089286	0.648148	3.849607	0.066092	2.363588
2	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN CIRCUS PARADE)	0.170918	0.168367	0.102041	0.597015	3.545907	0.073264	2.063681
3	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.168367	0.170918	0.102041	0.606061	3.545907	0.073264	2.104592
4	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN SPACEBOY)	0.170918	0.137755	0.104592	0.611940	4.442233	0.081047	2.221939

表3-4 数据集支持度、置信度和提升度

步骤 7　结果可视化

- 定义可视化函数

注意其中导入的 NetworkX 是一款 Python 的软件包，用于创造、操作复杂网络，以及学习复杂网络的结构等。代码如下：

```
import networkx as nx
import numpy as np
def draw_graph(rules, rules_to_show):
    G1 = nx.DiGraph()
    color_map=[]
    N = 50
    colors = np.random.rand(N)
    strs=['R0', 'R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R8', 'R9', 'R10', 'R11']
    for i in range (rules_to_show):
        G1.add_nodes_from(["R"+str(i)])
        for a in rules.iloc[i]['antecedants']:
            G1.add_nodes_from([a])
            G1.add_edge(a, "R"+str(i), color=colors[i] , weight = 2)
        for c in rules.iloc[i]['consequents']:
            G1.add_nodes_from([c])
            G1.add_edge("R"+str(i), c, color=colors[i],  weight=2)
    for node in G1:
        found_a_string = False
        for item in strs:
```

```
                if node==item:
                    found_a_string = True
            if found_a_string:
                color_map.append('yellow')
            else:
                color_map.append('green')
        edges = G1.edges()
        colors = [G1[u][v]['color'] for u,v in edges]
        weights = [G1[u][v]['weight'] for u,v in edges]
        pos = nx.spring_layout(G1, k=16, scale=1)
        nx.draw(G1, pos, edges=edges, node_color = color_map, edge_color=colors,
                width=weights, font_size=16, with_labels=False)
        for p in pos:   #raise text positions
            pos[p][1] += 0.07
        nx.draw_networkx_labels(G1, pos)
        plt.show()
```

- 展示前 6 个关联规则 R0，R1，…，R5，代码如下：

```
draw_graph(rules,6)
#plt.show()
```

结果如下：

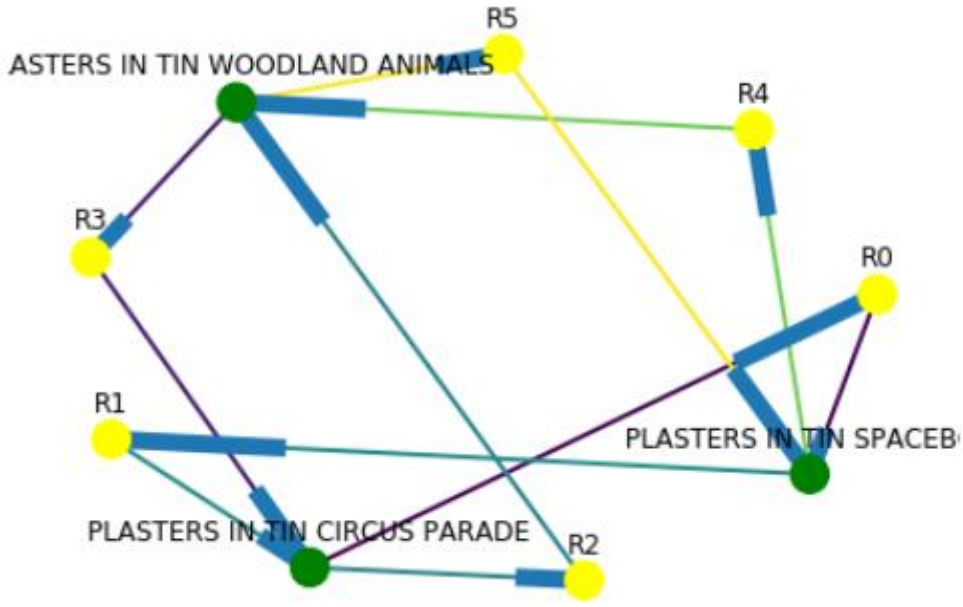


图3-5 前 6 项关联组合

可以看到，出现频次最多的商品组合是由 3 种商品：PLASTERS IN TIN SPACEBOY、PLASTERS IN TIN WOODLAND ANIMALS、PLASTERS IN TIN CIRCUS PARADE 的各种组合，其中，最明显的是，PLASTERS IN TIN CIRCUS PARADE 总会带来 PLASTERS IN TIN SPACEBOY 的销售。

- 统计上述 3 种商品销售次数，代码如下：

```
print(basket_sets['PLASTERS IN TIN CIRCUS PARADE'].sum())
print(basket_sets['PLASTERS IN TIN SPACEBOY'].sum())
print(basket_sets['PLASTERS IN TIN WOODLAND ANIMALS'].sum())
```

结果如下：

```
66
54
67
```

步骤 8　调整分析维度

- 限制置信度大于 80%，查看此时对应的关联规则，代码如下：

```
rules[ (rules['confidence'] >= 0.8) ]
```

结果如下：

	antecedants	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
8	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.132653	0.102041	0.800000	6.030769	0.085121	4.336735
10	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.127551	0.122449	0.888889	6.968889	0.104878	7.852041
11	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.127551	0.137755	0.122449	0.960000	6.968889	0.104878	21.556122
12	(SET/6 RED SPOTTY PAPER CUPS, SET/6 RED SPOTTY...	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.122449	0.132653	0.099490	0.812500	6.125000	0.083247	4.625850
13	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.127551	0.099490	0.975000	7.644000	0.086474	34.897959
14	(SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.137755	0.099490	0.975000	7.077778	0.085433	34.489796

表3-5 置信度限制后关联规则结果

- 制作可视化图

```
draw_graph(rules[ (rules['confidence'] >= 0.8) ],6)
```

结果如下：

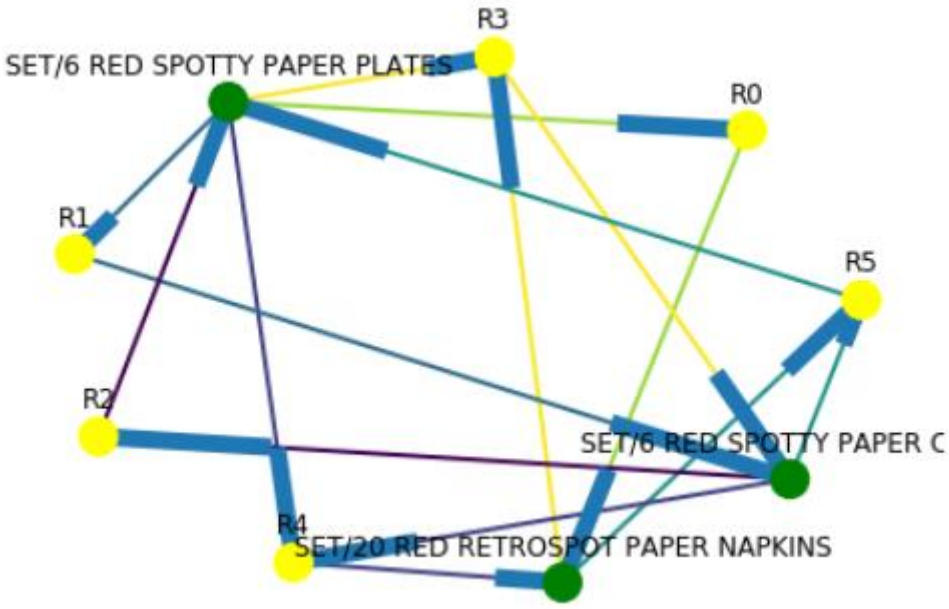


图3-6 调整置信度约束后的前 6 个商品组合关系

通过上图规则可以看出，在高于总体概率水平的约束下，最频繁出现的商品组合是：SET/6 RED SPOTTY PAPER PLATES、SET/6 RED SPOTTY PAPER CUPS、SET/20 RED RETROSPOT PAPER NAPKINS 的各种组合。

- 统计 3 个商品销售次数，代码如下：

```
basket_sets['SET/6 RED SPOTTY PAPER PLATES'].sum()
basket_sets['SET/6 RED SPOTTY PAPER CUPS'].sum()
basket_sets['SET/20 RED RETROSPOT PAPER NAPKINS'].sum()
```

结果如下：

```
50
54
52
```

- 类似地，可以再查看德国客户的消费商品的关联性

```
basket2 = (df[df['Country'] =="Germany"]
                .groupby(['InvoiceNo', 'Description'])['Quantity']
                .sum().unstack().reset_index().fillna(0)
                .set_index('InvoiceNo'))

basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)
rules2[ (rules2['lift'] >= 4) & (rules2['confidence'] >= 0.5)]
```

结果如下：

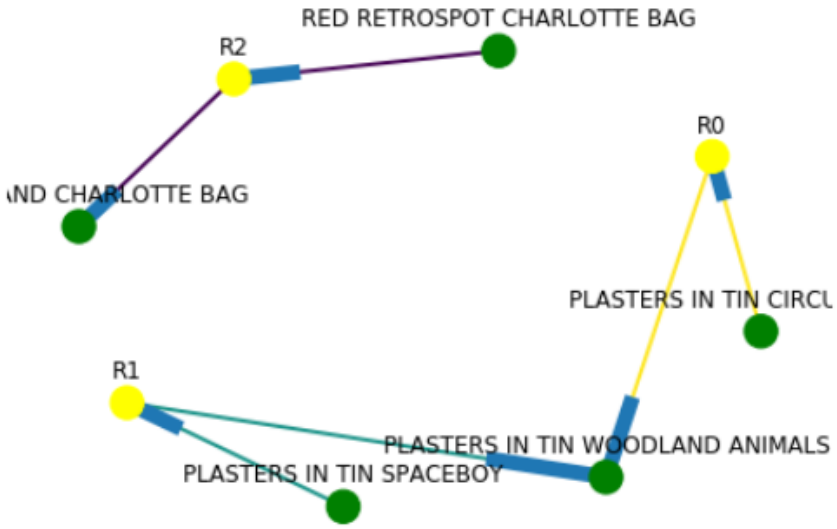
	antecedants	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
1	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.115974	0.137856	0.067834	0.584906	4.242887	0.051846	2.076984
7	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.107221	0.137856	0.061269	0.571429	4.145125	0.046488	2.011670

表3-6 按约束规则得前 3 个关联规则

- 展示前 3 个规则

```
draw_graph(rules2[ (rules2['lift'] >= 4) & (rules2['confidence'] >= 0.5)],3)
```

结果如下：



按约束规则得到的前 3 个关联规则

3.6 实验小结

该实验主要是利用 Apriori 算法对用户购物数据“Online Retail”进行分析，通过频繁项集构建和关联规则挖掘，筛选出强相关性产品组合，用于相应的推荐场景等。

4 运营商套餐推荐

4.1 实验说明

对不同用户（每各用户作为一个事务）使用过的电信产品清单进行分析，试图挖掘哪些电信产品存在相关性，或者哪些地区的用户更倾向于订购哪种套餐。

该实验主要是利用 Apriori 算法对电信用户使用过的电信产品清单数据进行分析。筛选出强相关性产品组合，可用于相应的推荐场景等。实验主要包括了频繁项集构建和关联规则挖掘两个部分。

Apriori 算法是一种挖掘关联规则的频繁项集算法，其核心思想是通过频繁项集生成和关联规则生成两个阶段来挖掘频繁项集。它的主要任务就是设法发现事物之间的内在联系。

本地离线数据地址和名称：data\Correlation.CSV

数据中的编码代表：

电信产品代码	中文名称	地市名称	地市编码
90109916	蚂蚁大宝卡	衢州	570
90063345	腾讯大王卡	杭州	571
90065148	滴滴小王卡	湖州	572
90065147	滴滴大王卡	嘉兴	573
90046637	腾讯视频小王卡	宁波	574
90151621	滴滴大橙卡	绍兴	575
90127327	百度大神卡	台州	576
90046638	腾讯呢音频小王卡	温州	577
90157593	百度女神卡	丽水	578
90151624	滴滴小橙卡	金华	579
90109906	蚂蚁小宝卡	舟山	580
90138402	招行大招卡	舟山群岛新区	580
90129503	京东小强卡		

90151622	微博大 V 卡		
90157638	哔哩哔哩 22 卡		
90163763	饿了么大饿卡		
90199605	懂我卡		
90215356	阿里 YunOS-9 元卡		

注：编码只是示例，不代表和商用真实编码相同，套餐属性也不代表和真实商用套餐相同。

4.2 实验环境要求

Python 3.7

4.3 实验建模流程要求

套餐推荐是一个关联规则问题，这里使用关联规则算法进行建模分析，挖掘频繁项。

4.3.1 实验实现步骤要求

步骤 1 安装 mlxtend 库

步骤 2 导入相应模块

步骤 3 读取并显示数据集

步骤 4 查看数据维度和类型

步骤 5 分组统计

步骤 6 可视化显示各地市的用户数量

步骤 7 生成标记每个用户的注册地市的初始字典

步骤 8 将注册地市字段转换为 0/1 编码的格式

步骤 9 将套餐代码转换为 0-1 形式

步骤 10 生成 0/1 编码格式的 df

步骤 11 创建频繁集

步骤 12 创建关联规则

步骤 13 定义可视化函数

步骤 14 展示前 6 个关联规则

4.4 参考答案

参考“运营商套餐推荐.ipynb”

5 消费者聚类

5.1 实验说明

在本案例中，使用人工智能技术的聚类算法对超市购物中心客户的一些基本数据进行建模分析，把客户分成不同的群体，供营销团队参考并相应地制定营销策略。

我们使用的数据集是超市用户会员卡的基本数据以及根据购物行为得出的消费指数，总共有 5 个字段，解释如下：

CustomerID：客户 ID

Gender：性别

Age：年龄

Annual Income (k$)：年收入

Spending Score (1-100)：消费指数

5.2 根据提供的数据集，设计一个完整的聚类分析实验

5.2.1 要求如下：

- 申请 ModelArts 的 notebook 实验环境申请，对接 OBS 数据源；
- 实现 OBS 数据加载与数据理解；
- 结合不同的绘图方式，如柱状图、散点图、盒图等分析不同属性字段的统计分布规律；
- 对不同聚类取值（k）效果进行评估，获取较好的聚类个数；
- 采用最佳聚类值进行聚类，同时结合散点分布图对聚类效果进行可视化展示。

5.3 参考答案

参考“消费者聚类.ipynb”

6 红酒聚类实验

6.1 实验介绍

6.1.1 简介

K 近邻算法（K-Nearest-Neighbor, KNN）是一种用于分类和回归的非参数统计方法，是机器学习最基础的算法之一。KNN 是无监督学习算法，无需训练，但是每次预测都需要遍历数据集，效率不高。KNN 的三个基本要素：

- K 值，一个样本的分类是由 K 个邻居的“多数表决”确定的。K 值越小，容易受噪声影响，反之，会使类别之间的界限变得模糊。
- 距离度量，反映了特征空间中两个样本间的相似度，距离越小，越相似。常用的有 Lp 距离（p=2 时，即为欧式距离）、曼哈顿距离、海明距离等。
- 分类决策规则，通常是多数表决，或者基于距离加权的多数表决（权值与距离成反比）。

本实验主要介绍使用 MindSpore 在部分 wine 数据集上进行 KNN 实验。

6.1.2 实验目的

- 了解 KNN 的基本概念；
- 了解如何使用 MindSpore 进行 KNN 实验。

6.2 实验环境要求

- MindSpore 0.5.0（MindSpore 版本会定期更新，本指导也会定期刷新，与版本配套）；
- 华为云 ModelArts（控制台左上角选择“华北-北京四”）：ModelArts 是华为云提供的面向开发者的一站式 AI 开发平台，集成了昇腾 AI 处理器资源池，用户可以在该平台下体验 MindSpore。

6.3 实验过程

6.3.1 数据准备

步骤 1　下载数据

Wine 数据集是模式识别最著名的数据集之一，Wine 数据集的官网：[Wine Data Set](http://archive.ics.uci.edu/ml/datasets/Wine)。这些数据是对来自意大利同一地区但来自三个不同品种的葡萄酒进行化学分析的结果。数据集分析了三种葡萄酒中每种所含 13 种成分的量。这些 13 种属性是

1. Alcohol，酒精
2. Malic acid，苹果酸
3. Ash，灰
4. Alcalinity of ash，灰的碱度
5. Magnesium，镁
6. Total phenols，总酚
7. Flavanoids，类黄酮
8. Nonflavanoid phenols，非黄酮酚
9. Proanthocyanins，原花青素
10. Color intensity，色彩强度
11. Hue，色调
12. OD280/OD315 of diluted wines，稀释酒的 OD280/OD315
13. Proline，脯氨酸

- 方式一：从 Wine 数据集官网下载[wine.data 文件](http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data)。
- 方式二：从华为云 OBS 中下载[wine.data 文件](https://share-course.obs.cn-north-4.myhuaweicloud.com/dataset/wine.data)。

Key	Value	Key	Value
Data Set Characteristics:	Multivariate	Number of Instances:	178
Attribute Characteristics:	Integer, Real	Number of Attributes:	13

| Associated Tasks: | Classification | Missing Values? | No |

步骤 2　上传数据到 OBS

点击新建的 OBS 桶名，通过“上传”、“新建文件夹”等功能，将数据集上传到 OBS 桶中。

图6-1 上传数据集到 OBS

6.3.2 数据读取与处理

步骤 1　导入 MindSpore 模块和辅助模块

```
import os
# os.environ['DEVICE_ID'] = '4'
import csv
import numpy as np

import mindspore as ms
from mindspore import context
from mindspore import nn
from mindspore.ops import operations as P
from mindspore.ops import functional as F

context.set_context(device_target="Ascend")
```

步骤 2　读取 Wine 数据集 wine.data，并查看部分数据。

输入：

```
with open('wine.data') as csv_file:
    data = list(csv.reader(csv_file, delimiter=','))
print(data[56:62]+data[130:133]) #  打印部分数据
```

输出：

```
[['1', '14.22', '1.7', '2.3', '16.3', '118', '3.2', '3', '.26', '2.03', '6.38', '.94', '3.31', '970'], ['1',
'13.29', '1.97', '2.68', '16.8', '102', '3', '3.23', '.31', '1.66', '6', '1.07', '2.84', '1270'], ['1', '13.72',
'1.43', '2.5', '16.7', '108', '3.4', '3.67', '.19', '2.04', '6.8', '.89', '2.87', '1285'], ['2', '12.37', '.94',
'1.36', '10.6', '88', '1.98', '.57', '.28', '.42', '1.95', '1.05', '1.82', '520'], ['2', '12.33', '1.1', '2.28',
'16', '101', '2.05', '1.09', '.63', '.41', '3.27', '1.25', '1.67', '680'], ['2', '12.64', '1.36', '2.02', '16.8',
'100', '2.02', '1.41', '.53', '.62', '5.75', '.98', '1.59', '450'], ['3', '12.86', '1.35', '2.32', '18', '122',
'1.51', '1.25', '.21', '.94', '4.1', '.76', '1.29', '630'], ['3', '12.88', '2.99', '2.4', '20', '104', '1.3', '1.22',
'.24', '.83', '5.4', '.74', '1.42', '530'], ['3', '12.81', '2.31', '2.4', '24', '98', '1.15', '1.09', '.27', '.83',
'5.7', '.66', '1.36', '560']]
```

步骤 3 取三类样本（共 178 条），将数据集的 13 个属性作为自变量 X。将数据集的 3 个类别作为因变量 Y。

```
X = np.array([[float(x) for x in s[1:]] for s in data[:178]], np.float32)
Y = np.array([s[0] for s in data[:178]], np.int32)
```

步骤 4 取样本的某两个属性进行 2 维可视化，可以看到在某两个属性上样本的分布情况以及可分性。

```
attrs = ['Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Total phenols',
         'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins', 'Color intensity', 'Hue',
         'OD280/OD315 of diluted wines', 'Proline']
plt.figure(figsize=(10, 8))
for i in range(0, 4):
    plt.subplot(2, 2, i+1)
    a1, a2 = 2 * i, 2 * i + 1
    plt.scatter(X[:59, a1], X[:59, a2], label='1')
    plt.scatter(X[59:130, a1], X[59:130, a2], label='2')
    plt.scatter(X[130:, a1], X[130:, a2], label='3')
    plt.xlabel(attrs[a1])
    plt.ylabel(attrs[a2])
    plt.legend()
plt.show()
```

步骤 5 将数据集按 128:50 划分为训练集（已知类别样本）和验证集（待验证样本）：

```
train_idx = np.random.choice(178, 128, replace=False)
test_idx = np.array(list(set(range(178)) - set(train_idx)))
X_train, Y_train = X[train_idx], Y[train_idx]
X_test, Y_test = X[test_idx], Y[test_idx]
```

6.3.3 计算距离

利用 MindSpore 提供的 tile, suqare, ReduceSum, sqrt, TopK 等算子，通过矩阵运算的方式同时计算输入样本 x 和已明确分类的其他样本 X_train 的距离，并计算出 top k 近邻。

```
class KnnNet(nn.Cell):
    def __init__(self, k):
        super(KnnNet, self).__init__()
        self.tile = P.Tile()
```

```
            self.sum = P.ReduceSum()
            self.topk = P.TopK()
            self.k = k

        def construct(self, x, X_train):
            # Tile input x to match the number of samples in X_train
            x_tile = self.tile(x, (128, 1))
            square_diff = F.square(x_tile - X_train)
            square_dist = self.sum(square_diff, 1)
            dist = F.sqrt(square_dist)
            # -dist mean the bigger the value is, the nearer the samples are
            values, indices = self.topk(-dist, self.k)
            return indices


def knn(knn_net, x, X_train, Y_train):
    x, X_train = ms.Tensor(x), ms.Tensor(X_train)
    indices = knn_net(x, X_train)
    topk_cls = [0]*len(indices.asnumpy())
    for idx in indices.asnumpy():
        topk_cls[Y_train[idx]] += 1
    cls = np.argmax(topk_cls)
    return cls
```

6.3.4 预测

在验证集上验证 KNN 算法的有效性，取 k = 5，验证精度接近 80%，说明 KNN 算法在该 3 分类任务上有效，能根据酒的 13 种属性判断出酒的品种。

```
acc = 0
knn_net = KnnNet(5)
for x, y in zip(X_test, Y_test):
    pred = knn(knn_net, x, X_train, Y_train)
    acc += (pred == y)
    print('label: %d, prediction: %s' % (y, pred))
print('Validation accuracy is %f' % (acc/len(Y_test)))
```

输出：

```
    label: 1, prediction: 1
    label: 3, prediction: 2
    label: 3, prediction: 3
    label: 1, prediction: 1
    label: 1, prediction: 1
    label: 1, prediction: 1
    label: 3, prediction: 3
    label: 1, prediction: 1
    ……
```

```
Validation accuracy is 0.800000
```

6.4 实验小结

本实验使用 MindSpore 实现了 KNN 算法，用来解决 3 分类问题。取 wine 数据集上的 3 类样本，分为已知类别样本和待验证样本，从验证结果可以看出 KNN 算法在该任务上有效，能根据酒的 13 种属性判断出酒的品种。