Apriori算法实现超市商品搭配出售方案

实验目标

通过本案例的学习和课后作业的练习：

1. 了解整个关联算法的流程；
2. 掌握数据的查看、预处理、结果展示等环节进行了展示。

你也可以将本案例相关的 ipynb 学习笔记分享到 AI Gallery Notebook (https://marketplace.huaweicloud.com/markets/aihub/notebook/list/) 版块获得成长值 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=9b8d7e7a-a150-449e-ac17-2dcf76d8b492)，分享方法请查看此文档 (https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=8afec58a-b797-4bf9-acca-76ed512a3acb)。

案例内容介绍

算法概述

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找出物品之间的关联规则。它的主要任务就是设法发现事物之间的内在联系。比如在常见的超市购物数据集，或者电商的网购数据集中，如果我们找到了频繁出现的数据集，那么对于超市，我们可以优化产品的位置摆放，对于电商，我们可以优化商品所在的仓库位置，达到节约成本，增加经济效益的目的。

关于这个算法有一个非常有名的故事："尿布和啤酒"。故事是这样的：美国的妇女们经常会嘱咐她们的丈夫下班后为孩子买尿布，而丈夫在买完尿布后又要顺手买回自己爱喝的啤酒，因此啤酒和尿布在一起被购买的机会很多。这个举措使尿布和啤酒的销量双双增加，并一直为众商家所津津乐道。

应用领域：

该算法已经被广泛的应用到商业、网络安全，移动通信等各个领域。

1. Apriori算法应用广泛，可用于消费市场价格分析，猜测顾客的消费习惯，比如较有名的"尿布和啤酒"的故事；

2.网络安全领域中的入侵检测技术；

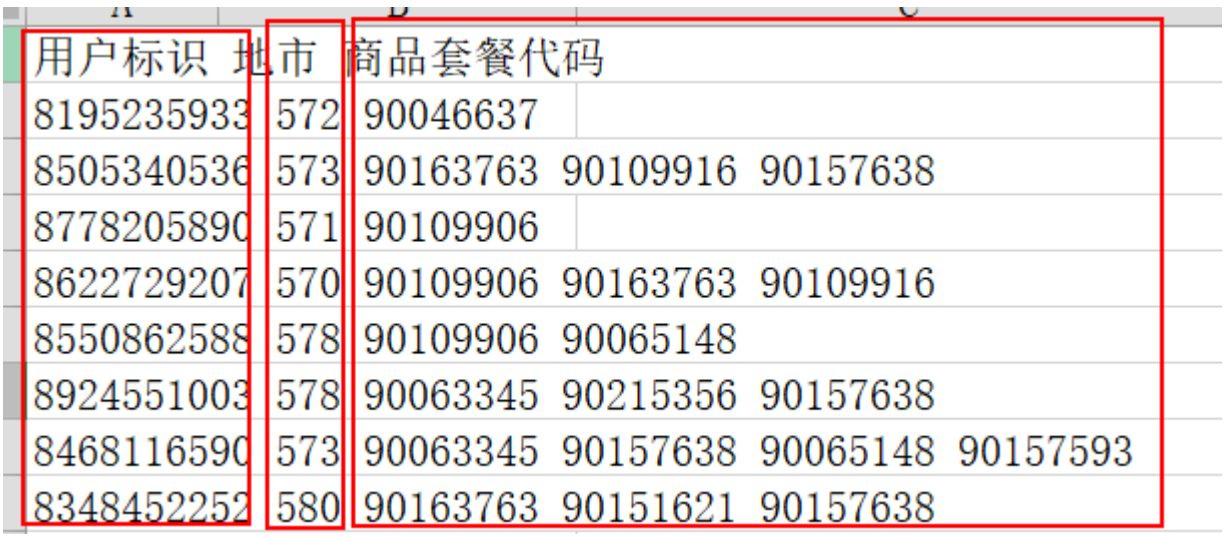
3.可用在用于高校管理中，根据挖掘规则可以有效地辅助学校管理部门有针对性的开展贫困助学工作；

4.也可用在移动通信领域中，指导运营商的业务运营和辅助业务提供商的决策制定。

关联规则算法的主要应用是购物篮分析，是为了从大量的订单中发现商品潜在的关联。其中常用的一个算法叫Apriori先验算法。

实验介绍

某大型连锁超市，在各个地区都有分店，在特定的时期需要做促销活动，由于每个地区消费者的行为有所差异，因此需要对每个地区的购物车数据单独分析，找出适合每个地区的商品促销套餐。购物车数据包含如下所示三个字段：用户标识，地区编号，购物车商品代码



用户标识	地市	商品套餐代码
8195235933	572	90046637
8505340536	573	90163763 90109916 90157638
8778205890	571	90109906
8622729207	570	90109906 90163763 90109916
8550862588	578	90109906 90065148
8924551003	578	90063345 90215356 90157638
8468116590	573	90063345 90157638 90065148 90157593
8348452252	580	90163763 90151621 90157638

本案例推荐的理论学习视频：

- 《AI技术领域课程--机器学习》 Apriori (https://education.huaweicloud.com/courses/course-v1:HuaweiX+CBUCNXE086+Self-paced/courseware/c2ea05f2357c443eacf554f37aa2e6a7/196a6b7ff71b43e6a67c1fe7a2dbe034/)

注意事项

1. 如果您是第一次使用 JupyterLab，请查看《ModelArts JupyterLab使用指导》(https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=03676d0a-0630-4a3f-b62c-07fba43d2857)了解使用方法；
2. 如果您在使用 JupyterLab 过程中碰到报错，请参考《ModelArts JupyterLab常见问题解决办法》(https://marketplace.huaweicloud.com/markets/aihub/article/detail/?content_id=9ad8ce7d-06f7-4394-80ef-4dbf6cfb4be1)尝试解决问题。

实验步骤

1、安装所需模块

In [1]:
```
!pip install pyfpgrowth
```

```
Requirement already satisfied: pyfpgrowth in /home/ma-user/anaconda3/envs/XGBoost-Sklearn/lib/python3.6/site-packages

You are using pip version 9.0.1, however version 21.2.4 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.
```

2、数据查看

In [2]:
```
#!-*- coding:utf-8 -*-

# 添加需要的模块
import csv
import itertools
import pyfpgrowth as fp
import matplotlib as plt
import moxing as mox
import os

if not os.path.exists('Correlation.CSV'):
    mox.file.copy('obs://modelarts-labs-bj4-v2/course/hwc_edu/machine_learning/datasets/apriori_data/Correlation.CSV',
                  'Correlation.CSV')

# 读取数据# 不能用pandas的原因是每一行的列数不相等
with open("Correlation.CSV", "r", encoding='gbk') as csv_file:
    csv_file1 = csv.reader(csv_file)
    for i, rows in enumerate(csv_file1):
        if 0 < i and i < 10:
            print(rows)
```

```
INFO:root:Using MoXing-v1.17.3-

INFO:root:Using OBS-Python-SDK-3.20.7

['8195235933 572 90046637']

['8505340536 573 90163763 90109916 90157638']

['8778205890 571 90109906']

['8622729207 570 90109906 90163763 90109916']

['8550862588 578 90109906 90065148']

['8924551003 578 90063345 90215356 90157638']

['8468116590 573 90063345 90157638 90065148 90157593']

['8348452252 580 90163763 90151621 90157638']

['8716034719 579 90215356']
```

3、工具函数构建

In [3]:
```
#  1、获取对应地区destID的数据集
def initData(destId):
    destDatalist = []
    with open("Correlation.CSV", "r", encoding='gbk') as csv_file:
        csv_file1 = csv.reader(csv_file)
        for i, rows in enumerate(csv_file1):
            if 0 < i:
                rowValue = rows[0].split(" ")
                if int(rowValue[1]) == destId:
                    if len(rowValue) > 2:
                        rowValue.remove(rowValue[0])
                        rowValue.remove(rowValue[0])
                        destDatalist.append(rowValue)
                else:
                    continue
    return destDatalist


#  2、获取类别数
def getlenRuleKinds(sortRuleslist):
    kindsdict = {}
    if isinstance(sortRuleslist, dict):
        # print sortRuleslist.items()
        for rule in sortRuleslist.items():
            rulekey = rule[0]
            rulekeyitem = len(rulekey) + 1
            if rulekeyitem in kindsdict:
                kindscount = kindsdict.get(rulekeyitem)
                kindsdict[rulekeyitem] = int(kindscount) + 1
            else:
                kindsdict[rulekeyitem] = 1

    elif isinstance(sortRuleslist, list):
        # print sortRuleslist
        for rule in sortRuleslist:
            # print rule
            rulekey = rule[0]
            rulekeyitem = len(rulekey) + 1
            if rulekeyitem in kindsdict:
                kindscount = kindsdict.get(rulekeyitem)
                kindsdict[rulekeyitem] = int(kindscount) + 1
            else:
                kindsdict[rulekeyitem] = 1

    kindlist = sorted(kindsdict.items(), key=lambda item: item[0], reve
rse=True)
    for item in kindlist:
        if item[0] > 2:
            kindsdict[item[0] - 1] = item[0] * kindsdict[item[0]] + kin
dsdict[item[0] - 1]

    return kindsdict


#  3、获取指定套餐个数的套餐类别
```

```
def getlenRules(sortRuleslist, rulelen):
    ruledict = {}
    # print sortRuleslist
    if isinstance(sortRuleslist, dict):
        # print sortRuleslist.items()
        for rule in sortRuleslist.items():
            rulekey = rule[0]
            rulevalue = rule[1]
            # print rulekey
            if (len(rulekey) + 1 >= rulelen):
                rulecell = rulekey + rulevalue[0]
                for antecedent in itertools.combinations(list(rulecel
l), rulelen):
                    if antecedent in ruledict:
                        ruleItem_conf_new = rulevalue[1]
                        ruleItem_conf_old = ruledict[antecedent]
                        if isinstance(ruleItem_conf_old, tuple):
                            if ruleItem_conf_old[0] == ruleItem_conf_ne
w[0]:
                                ruleItem_conf_new[1] = max(float(ruleIt
em_conf_new[1]), float(ruleItem_conf_old[1]))
                                ruledict[antecedent] = tuple(ruleItem_c
onf_new[0], ruleItem_conf_new[1])
                            else:
                                ruleItem_conf_value = float(ruleItem_co
nf_new[1]) + float(ruleItem_conf_old[1])
                                ruledict[antecedent] = ruleItem_conf_va
lue
                        else:
                            ruleItem_conf_value = max(float(ruleItem_co
nf_new), ruleItem_conf_old)
                            ruledict[antecedent] = ruleItem_conf_value
                    else:
                        ruleItem_conf_new = rulevalue[1]
                        ruledict[antecedent] = ruleItem_conf_new
            else:
                continue
            # if len(rule[0])==rulelen:
            #     rulelist.append(rule)
        if len(ruledict) == 0:
            print('不存在包含%d项集频繁项' % rulelen)
            return None
        else:
            return sorted(ruledict.items(), key=lambda x: x[1], reverse
=True)
    elif isinstance(sortRuleslist, list):
        # print sortRuleslist
        for rule in sortRuleslist:
            rulekey = rule[0]
            rulevalue = rule[1]
            # print rulekey
            if (len(rulekey) + 1 >= rulelen):
                rulecell = rulekey + rulevalue[0]
                for antecedent in itertools.combinations(list(rulecel
l), rulelen):
                    if antecedent in ruledict:
```