目录

一、 背景介绍

1、自动售货机简介

自动售货机是商业自动化的常用设备，它不受时间、地点的限制，能节省人力、方便交易。是一种全新的商业零售形式，又被称为 24 小时营业的微型超市。自动售货机以线上经营的理念，提供线下的便利服务，以小巧、自助的经营模式节省人工成本，让实惠、高品质的商品触手可及，成为当下零售经营的又一主流模式

2、自动售货机数据分析的必要性

自动售货机内商品的供给频率、种类选择、供给量、站点选择是自动售货机运营者需要重点关注的问题。因此，科学的商业数据分析能够帮助经营者了解用户需求，掌握商品需求量，为用户提供精准贴心的服务，是掌握经营方向的重要手段，对自动售货机这一营销模式的发展有着非常重要的意义

二、数据说明

1、数据来源

数据来源：泰迪云课堂

2、数据简介：

某商店在不同地点安放了 5 台自动售货机，编号分别为 A、B、C、D、E。附件 1 包含了从 2017 年 1 月 1 日至 2017 年 12 月 31 日每台自动售货机的商品销售数据，附件 2 提供难过了商品的分类

三、商品数据分析

1、任务一：数据预处理

1.1

将数据分门别类归为各个不同的地点的数据，并存放到对应文件夹。

先将附件一中的数据进行读取

all = pd.read_csv('D:\python 打开文件\项目数据\附件 1.csv',encoding ='gbk')

再使用 loc 对条件进行一个筛选即可提取不同地点的数据。

place_A=all.loc[all['地点']=='A',['订单号','设备 ID','应付金额','实际金额','商品','支付时间','地点','状态','提现']]

提取出来后使用 pd 中的 to_csv 进行保存

pd.DataFrame.to_csv(place_A,"task1-1A.csv",',',encoding ='gbk')

1.2

由于要求将 5 月份的数据进行一个汇报，所以需要先将 all[‘支付时间’]这一列的数据改写成为 datetimeIndex 类型始数据才可以对它进行一个时间上的判断

place_A['支付时间'] = pd.DatetimeIndex(place_A['支付时间'])

同样使用 loc 进行条件对 A 五月份的数据进行提取

place_A_5=place_A.loc[place_A['支付时间'].dt.month==5,['订单号','设备 ID','应付金额','实际金额','商品','支付时间','地点','状态','提现']]

当对 C 售货机部分进行同样的操作后发现会报错，这是由于 C 售货机中的数据有一个数据为 2017/2/29。由于 2017 年没有 29 日，所以这是个异常值，有两种方法对它进行操作，一是将着一个数据删除，而是对这个数据进行更改。我的理解是这个数据应该是 2 月份最后一天的数据所以将其改成了 2/28 接着进行操作

交易额仅需将对应五月的数据中'实际金额'那一列进行相加即可得到。

```
np.sum(place_A_5['实际金额'])
```

由于里面所有数据都是成功交易的，所以订单量则等于数据的行数。

```
place_A_5.iloc[:,0].size
```

1.3

每个月的交易额和订单量可以使用和上述方法相同的办法进行求解

为了减少代码的长度，可以构造一个求某月交易额和订单量量的函数

```
def whichmonth(i):
xxxxx
xxxxx
return
```

传入的数据 i 则是你想要求的月份

然后在构建一个 for 循环即可迅速代出 12 个月份对应的交易额和订单量

```
for I in range(1,13):
   whichmonth(1)
```

2、任务二：数据可视化

2.1 绘制 6 月商品销量前 5 的柱状图

这是对所有售货机进行操作，所以先将 all[‘支付时间’]这一列的数据转换为 datetimeIndex 类型。并将 5 月份所有数据进行提取，由于不需要很多列的数据，减轻电脑工作的负担，不需要提取这么多列。

```
all['支付时间'] = pd.DatetimeIndex(all['支付时间'])

number=all.loc[all['支付时间'].dt.month==5,['商品','提现']]
```

然后建造空列表存放'商品'列的数据

```
list=[]
```

通过 for 循环将商品一个一个加入到 list 中

```
for row in number.itertuples():
    list.append(getattr(row, '商品'))
```

并对其中出现最多次数的五个商品进行提取

```
Counter(list).most_common(5)
```

在构建包含 x 轴和 y 轴的数据列表，将所需要的数据放入其中

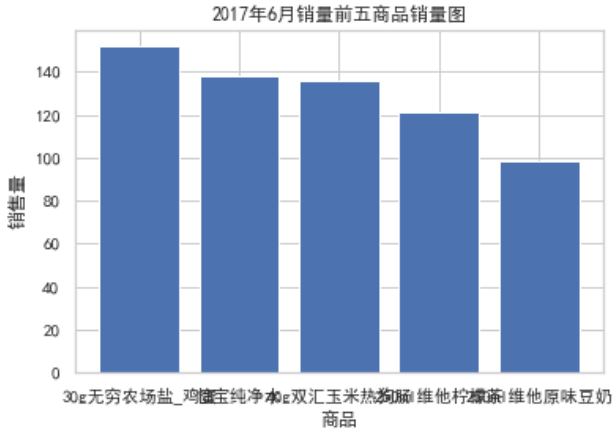
```
x=[]

y=[]

for i in range(5):
```

x.append(Counter(list).most_common(5)[i][0])

y.append(Counter(list).most_common(5)[i][1])

最后画出柱状图即可

plt.bar(x,y)



2.2 绘制每台售货机每月总交易额折线图及交易额月环比增长率的柱状图

先构造 month 列表存放所有月份作为 x 轴和一个列表放入每个月的数据

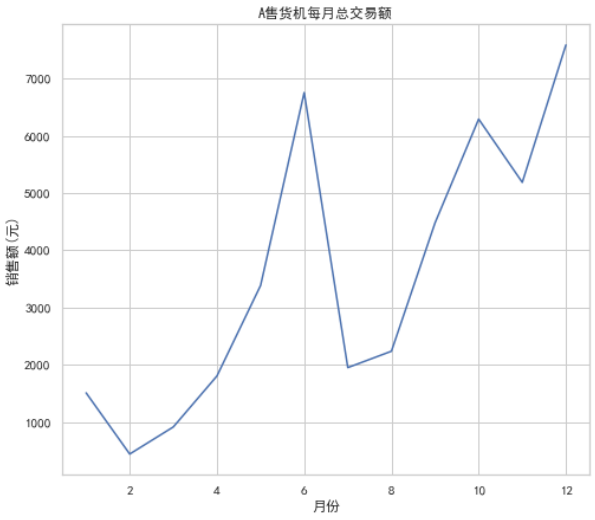
```
month=[1,2,3,4,5,6,7,8,9,10,11,12]

A=[]
```

通过 for 循环实现将每个月的数据放入 A 列表

```
for i in range(1,13):

    place_A_month=place_A.loc[place_A['支付时间'].dt.month==i,['实际金额','商品','支付时间','状态','提现']]

    A.append(np.sum(place_A_month['实际金额']))
```

最后进行画图即可

```
plt.plot(month[:],A[:])
```



其他售货机用类似的方法即可

环比增长率的计算

A[i+1]-A[i])/A[i]

环比增长率由于第一个月没有增长率，所以新建一个只有 11 个元素的 month1 和一个列表存放增长率

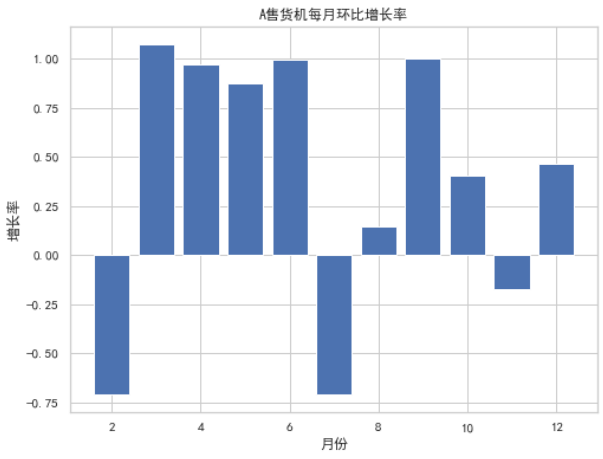
month1=[2,3,4,5,6,7,8,9,10,11,12]，a=[]

使用 for 循环将 a 列表填充完整

```
for i in range(0,11):
    a.append((A[i+1]-A[i])/A[i])
```

最后进行构图

plt.bar(month1,a)



其他售货机用类似的方法即可

2.3 绘制每台售货机饮料毛利润占总的多少

由于涉及到饮料这个类别，仅靠附件 1 中的数据是不够的，所以将附件二中的数据进行一个合并

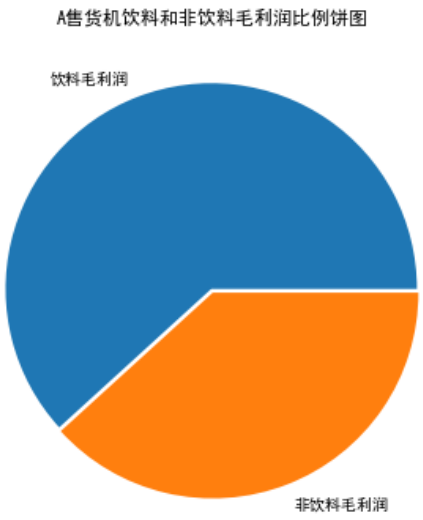
```
place_A_1=pd.merge(place_A,all2,on='商品')
```

再将饮料类和非饮料类的数据进行提取，并求出利润

```
drink=place_A_1.loc[place_A_1['大类']=='饮料',['实际金额','商品','状态','提现']]

not_drink=place_A_1.loc[place_A_1['大类']=='非饮料',['实际金额','商品','状态','提现']]

money=[np.sum(drink['实际金额'])*0.25,np.sum(not_drink['实际金额'])*0.2]
```

最后进行画图即可

```
plt.pie(money)
```



其他售货机用类似方法即可

2.4 绘制每月交易额均值的气泡图

先将 1 月份的数据提取出来

```
Jan=all1.loc[all1['支付时间'].dt.month==1,['实际金额','二级类','支付时间']]
```

求出其中金额的平均值，并将其字典化

```
Jan_sum=dict(Jan.groupby('二级类')['实际金额'].mean())
```

再把它变作 dataframe 格式并设置名称

```
Jan_sum=pd.DataFrame(pd.Series(Jan_sum),columns=['平均值'])
Jan_sum=Jan_sum.reset_index().rename(columns ={'二级类':'平均值'})
```

在新添一列作为月份的标识

```
Jan_sum['月份'] = 1
```

pao 作为最后画图的一个数据

```
pao=Jan_sum
```

后续的数据则使用 append 方法进行添加即可

```
pao=pao.append(Feb_sum)
```

最后 pao 列表即会包含十二个月份的数据，进行构图即可

```
sns.set(style = "whitegrid")
```

```
x = pao['月份']

y = pao['index']

z = pao['平均值']

cm = plt.cm.get_cmap('RdYlBu')

fig,ax = plt.subplots(figsize = (12,10))

bubble = ax.scatter(x, y , s = (z - np.min(z) + 0.1) * 1000, c = z, cmap = cm, linewidth = 0.5,
alpha = 0.5)

ax.grid()

fig.colorbar(bubble)

ax.set_xlabel('月份', fontsize = 15)

ax.set_ylabel('二级类', fontsize = 15)

plt.show()
```

2.5 绘制售货机 C 6，7，8 三个月的订单量的热力图

list_1 存放日期，list_2 存放销售量（默认值为 0），将 list_1 作为字典的 key 值，list_2 作为字典的 value 值

```
list_1=[]

list_2=[]

for i in range(1,32):

   list_1.append(i)

   list_2.append(0)

month_C_6 = dict(zip(list_1,list_2))
```

list_1 存放时间（单位小时间），list_2 存放每小时的销售数据（默认值为 0），将 list_1 作为字典的 key 值，list_2 作为字典的 value 值

```
list_1=[]

list_2=[]

for i in range(24):

      list_1.append(i)

      list_2.append(0)

date=dict(zip(list_1,list_2))
```

将 C 售货机中 6 月份的数据提取出来

```
place_C=all.loc[all['地点']=='C',['订单号','设备 ID','应付金额','实际金额','商品','支付时间','地点','状态','提现']]

place_C['支付时间'] = pd.DatetimeIndex(place_C['支付时间'])

place_C_6=place_C.loc[place_C['支付时间'].dt.month==6,['商品','支付时间',]]

place_C_6['支付时间'] = pd.DatetimeIndex(place_C_6['支付时间'])
```

先将 6 月份第一天的每小时数据输入到 daily_6 中

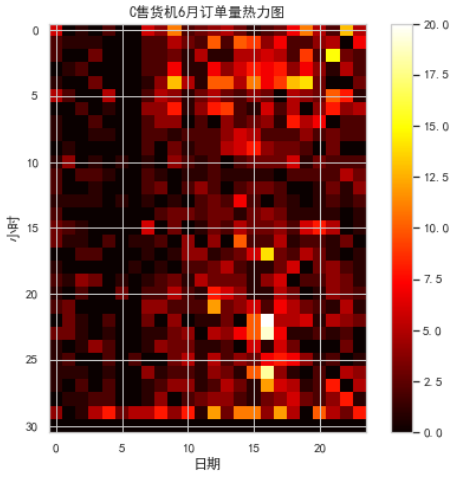
```
for i in range(24):
    a = place_C_6[(place_C_6['支付时间'].dt.hour==i) & (place_C_6['支付时间'].dt.day==1)].index.tolist()
    date[i]=len(a)
    daily_6=pd.DataFrame(date,index=[1])
```

用 for 循环将 6 月份每天每小时的数据输入到 daily_6 中

```
for x in range(2,32):
  for i in range(24):
      a = place_C_6[(place_C_6['支付时间'].dt.hour==i) & (place_C_6['支付时间'].dt.day==x)].index.tolist()
      date[i]=len(a)
  daily_6 = daily_6.append(pd.DataFrame(date,index=[x]))
```

最后画图即可

```
plt.imshow(daily_6)
```



其他两个月份用相似方法即可

可以看出月初购买商品的人比较少，月中和月末购买人数较多。

3、任务三：自动售货机画像

对每台售货机的每种商品在该售货机类销售情况，贴上标签

将饮品类进行分类提取，用计数函数算出每个的销售量

```
Pivot_A=pd.pivot_table(drink_A[['商品','提现']],index='商品',aggfunc='count')
```

以所有商品中排名前百分之 20（向下取整）的作为热销商品，排名后百分之 20（向下取整）的作为滞销商品，剩余的则是正常商品

```
top=math.floor(len(Pivot_A)*0.2)
```

对 Pivot_A 进行排序，从大到小

```
Pivot_A=Pivot_A.reindex(Pivot_A['提现'].abs().sort_values(ascending=False).index)
```

先将所有商品标记为正常，再前销售量在百分之 20 商品改标记改为热销，最后将销售量在后百分之 20 的商品改标记为滞销

```
Pivot_A['商标']='正常'
Pivot_A.head(top)['商标']='热销'
Pivot_A.tail(top)['商标']='滞销'
```

建立空列表分别存放'序号','商品名称'，'标签'

```
list_1=[]
list_2=[]
list_3=[]
```

通过 for 循环一一放入对应列表

```
for i in range(Pivot_A.iloc[:,0].size-1):

list_1.append(i+1)

list_2.append(Pivot_A.index[i+1])

list_3.append(Pivot_A['商标'][i])
```

将三个列表合为一个字典，并将字典转化为 dataframe 格式

```
dic1={'序号':list_1,'饮料类商品':list_2,'标签':list_3}

task_3_1A=pd.DataFrame(dic1)
```

保存该已含标签的 csv 文件

```
pd.DataFrame.to_csv(task_3_1A,"task3-1A.csv",',',encoding ='gbk')
```

4、任务四：业务预测

预测的原理：将已获得的数据，拆分成为训练集、测试集。通过训练集来对数据进行分析，并形成某些规则，合理的对未来的可能会获得的数据进行一个预测。

能否预测？

我认为不可以根据附件提供的数据对每台售货机的每个大类商品在 2018 年 1 月的交易额进行预测。

原因：附件中的数据的数量不够多，对于 1 月的数据来说，只有 2017 年 1 月的数据。而自动售货机的销售额虽然不是直接和月份有相关联，但是不同月份的假期、温度、天黑和天亮的时间都不完全相同，而这些因素又会对我们是否购买商品和购买什么商品产生比较大的影响。哪怕我们忽略不同月份的影响，也只有 12 个月份的数据，这样去推测误差也会十分大，不具有太大的参考价值。所以我认为是无法对 2018 年 1 月的交易额进行预测的

建议：经营者需要给我们更多的数据，而且没有必要每一次交易的详情都给到我们，只需要每个月份每种商品出售了多少份即可进行预测，数据最好至少有 5 年的数据，这样才能对预测的加过有不低的把