**目录**

[**1 需求分析** 1](#_Toc136975884)

[1.1可行性分析 1](#_Toc136975885)

[1.2功能需求分析 2](#_Toc136975886)

[**2 数据源的获取、读取与数据预处理 3**](#_Toc136975887)

[2.1读取数据 3](#_Toc136975888)

[2.2 加载数据文件 3](#_Toc136975889)

[2.3 提取数据 4](#_Toc136975890)

[2.3.1根据业务需要提取数据,提取2020年数据 4](#_Toc136975891)

[2.3.2提取数据时,处理与业务流程不符合数据、支付时间间隔过长数据 4](#_Toc136975892)

[2.3.3提取数据时,处理与业务流程不符合数据,订单金额与支付金额为负 4](#_Toc136975893)

[**3 数据分析、处理 5**](#_Toc136975894)

[3.1 数据清洗 5](#_Toc136975895)

[3.1.1清洗orderID 5](#_Toc136975896)

[3.1.2 清洗goodsID 5](#_Toc136975897)

[3.1.3 清洗chanelID 5](#_Toc136975898)

[3.1.4 清洗platformtype 6](#_Toc136975899)

[3.1.5 清洗payment 6](#_Toc136975900)

[3.2 数据规整 6](#_Toc136975901)

[3.3数据处理完毕 查看数据 7](#_Toc136975902)

[**4 分析数据及可视化 8**](#_Toc136975903)

[4.1 分析数据,整体运营情况 8](#_Toc136975904)

[4.2销售情况,各月份GMV\成交额趋势 8](#_Toc136975905)

[4.3 流量渠道来源分析 10](#_Toc136975906)

[4.4 用户行为, 研究周一到周日哪天订单量最高 12](#_Toc136975907)

[4.5 用户行为, 哪个时间段下单量最高 13](#_Toc136975908)

[4.6用户行为,客户情况 15](#_Toc136975909)

[4.7用户行为,客户复购率 16](#_Toc136975910)

[**5 结论 20**](#_Toc136975911)

[**6 小结 21**](#_Toc136975912)

[**7 参考文献 22**](#_Toc136975913)

# 需求分析

## 1.1可行性分析

电商平台数据分析是电商企业运营中不可或缺的一部分，通过对交易数据、用户行为数据、行业数据等多方面的分析，可以帮助企业了解用户需求、优化产品和服务、提高销售额和用户忠诚度。 首先，电商平台数据分析可以帮助企业了解用户需求。通过对用户行为数据的分析，可以发现用户的购买偏好、购买习惯、消费水平等信息，从而帮助企业优化产品和服务，提供更加符合用户需求的商品和服务。例如，企业可以根据用户的购买偏好推出相应的促销活动，提高用户的购买转化率。 其次，电商平台数据分析可以帮助企业优化产品和服务。通过对交易数据的分析，可以发现用户的购买行为和购买习惯，从而优化产品和服务。例如，企业可以根据用户的购买偏好调整产品的价格、品质、包装等方面，提高用户的购买意愿和满意度。 最后，电商平台数据分析可以帮助企业提高销售额。通过对销售数据的分析，可以发现销售趋势、热销产品、销售瓶颈等信息，从而帮助企业调整销售策略，提高销售额。例如，企业可以根据销售数据调整产品的库存量和销售策略，提高销售效率。 总之，电商平台数据分析对于电商企业来说是非常重要的，通过对多方面的数据进行分析，可以帮助企业了解用户需求、优化产品和服务、提高销售额和用户忠诚度，从而取得竞争优势。

随着电子商务的快速发展，越来越多的企业开始将电商平台作为重要的销售渠道。然而，随着竞争的加剧和用户需求的不断变化，电商企业需要不断优化其电商平台，提高销售额和用户忠诚度。 电商平台数据分析是电商企业运营中不可或缺的一部分，通过对交易数据、用户行为数据、行业数据等多方面的分析，可以帮助企业了解用户需求、优化产品和服务、提高销售额和用户忠诚度。 因此，在电商平台的建设和运营过程中，企业需要收集和分析大量的数据，以便更好地了解用户需求和市场趋势，并及时调整策略。同时，随着用户需求的不断变化和竞争的加剧，电商企业需要不断优化其数据分析方法和技术，以便更好地了解用户需求，提高销售额和用户忠诚度。 因此，本文将探讨电商平台数据分析的重要性和必要性，以及如何通过数据分析来优化电商平台，提高销售额和用户忠诚度。

数据的质量和完整性是电商平台数据分析的关键。电商企业需要收集和整理大量的交易数据、用户行为数据、行业数据等多种数据类型，并对这些数据进行清洗、整理和筛选，以确保数据的准确性和完整性。如果数据质量和完整性不够高，就会影响数据分析的准确性和可靠性。

## 1.2功能需求分析

python中matplotlib利用函数bar 用于绘制柱状图，函数barh 用于绘制条形图，函数hist 用于绘制直方图，函数polor 用于绘制极线图，函数polor 用于绘制极线图。从而实现对数据进行具体分析和可视化处理。

1. **数据源的获取、读取与数据预处理**

## 2.1读取数据



**图1 读取数据**

## 2.2 加载数据文件

import pandas as pd //加载处理数据所需要的库

df = pd.read\_excel('order2020.xlsx',index\_col='id') //读取数据文件

#pd.set\_option('display.width',None) //设置数据展示宽度

#print(df.tail(10)) //展示导入数据的后10行，以便检查

pd.set\_option('display.width',None) //设置数据展示宽度，解决列展示出现省略号问题

print(df.head()) //默认读取前5行记录

print(df.tail(10)) //读取后10行的数据，无参默认为5行

## 2.3 提取数据

### 2.3.1根据业务需要提取数据,提取2020年数据

# 1.引入时间模块, 确定周期时间

import datetime

# 2.确定开始时间节点与结束时间节点

startTime = datetime.datetime(2020, 1, 1) #开始时间

endTime = datetime.datetime(2020, 12, 31, 23, 59, 59) #结束时间

# 3.将2020年1月1日前数据删除

df[df.orderTime < startTime]

#删除数据 drop(index="索引", inplace=True,False)

df.drop(index=df[df.orderTime < startTime].index, inplace=True)

# 4.将2020年12月31日后数据删除

df[df.orderTime > endTime]

df.drop(index=df[df.orderTime > endTime].index, inplace=True)

### 2.3.2提取数据时,处理与业务流程不符合数据、支付时间间隔过长数据

# 1.下单时间与支付时间间隔

df['payinterval'] = (df.payTime-df.orderTime).dt.total\_seconds()

# 2.支付时间间隔大于30分钟与支付时间早于下单时间

df[df.payinterval>1800]

df.drop(index=df[df.payinterval>1800].index, inplace=True)

df.drop(index=df[df.payinterval<0].index, inplace=True)

### 2.3.3提取数据时,处理与业务流程不符合数据,订单金额与支付金额为负

# 1.订单金额为负

df[df.orderAmount < 0]

# 2.付款金额为负

df[df.payment < 0]

# 3.删除相应订单

df.drop(index=df[df.orderAmount < 0].index, inplace=True) #删除负订单金额

df.drop(index=df[df.payment < 0].index, inplace=True) #删除负付款金额

# 3 数据分析、处理

## 3.1 数据清洗

脏数据类型

重复值

异常值

缺失值

脏数据处理方式

### 3.1.1清洗orderID

#1.订单orderID不重复的个数

df.orderID.unique().size

#2.删除重复数据

df.drop(index=df[df.orderID.duplicated()].index, inplace=True)

df.info()

### 3.1.2 清洗goodsID

#PR000000 商品下架

df.goodsID[df.goodsID == 'PR000000'].size

df.drop(index=df[df.goodsID == 'PR000000'].index, inplace=True)

df.info()

### 3.1.3 清洗chanelID

#1.查看chanelID空值

df[df.chanelID.isnull()]

#2.对空值进行修补

df['chanelID'].fillna(value=df.chanelID.mode()[0], inplace=True)

df.info()

### 3.1.4 清洗platformtype

print(df.platformType.unique()) #列出所有不重复平台

df['platformType']=df['platformType'].str.replace(" ","") #空字符串代替空格

print(df.platformType.unique()) #列出所有不重复平台，再次检查

### 3.1.5 清洗payment

# 1.创建折扣字段

df['discount'] = (df.payment/df.orderAmount)

df.describe()

# 2.平均折扣

meanDiscount = df[df['discount']<=1].discount.sum() / df[df['discount']<=1].discount.size

meanDiscount

# 3.找到折扣大于1的数据

df[df['discount']>1]

df['payment'] = df['payment'].mask(df['discount']>1,None)

# 4.对折扣大于1的数据进行填补

df['payment'].fillna(value=df.orderAmount\*meanDiscount , inplace=True)

df.info()

# 5.处理折扣

df['discount'] = round((df.payment/df.orderAmount),2)

pd.set\_option('display.width',None)

print(df)

## 3.2 数据规整

df['month'] = df['orderTime'].dt.month #提取月份

df['weekday'] = df['orderTime'].dt.dayofweek+1 #周几

df['orderDate'] = df['orderTime'].dt.date

#print(df)

## 3.3数据处理完毕 查看数据

print(df.describe())

输出结果：

表格

描述已自动生成

**图2 输出结果**

# 4 分析数据及可视化

## 4.1 分析数据,整体运营情况

#总体概览

# 1.销售GMV

df.orderAmount.sum()/10000 #10835

# 2.成交总和

df.payment.sum()/10000 #10246

# 3.实际成交额

df[df.chargeback=="否"].payment.sum()/10000 # 8879

# 4.订单数量

df.orderID.unique().size #103146

# 5.退货订单数

df[df.chargeback=="是"].orderID.size #13590

# 6.退货率

df[df.chargeback=="是"].orderID.size/df.orderID.unique().size

# 7.用户数

df.userID.unique().size #78525

## 4.2销售情况,各月份GMV\成交额趋势

#1.绘制图形

#1.1引入相应的库

from PythonDataAnalyse import df #调用PythonDataAnalyse.py的df函数

import matplotlib.pyplot as plt

from matplotlib.font\_manager import FontProperties

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['font.serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

#1.2设置字体

my\_font =FontProperties(fname='C:/Windows/Fonts/Deng.ttf',size=12,)

#1.3设置画布大小

plt.figure(figsize=(10,6))

#1.4设置网格线 辅助线

plt.grid(alpha=0.4)

#2.横纵坐标值

#2.1所有横坐标都一致

x = df.groupby('month')['orderAmount'].sum().index

#2.2GMV

y1 = df.groupby('month')['orderAmount'].sum().values/10000

#2.3销售实际付款

y2 = df.groupby('month')['payment'].sum().values/10000

#2.4不含退单销售额

y3 = df[df.chargeback=="否"].groupby('month')['payment'].sum().values/10000

#2.5 X横轴坐标文字

x\_ticks\_label = ["{}月份".format(i) for i in x]

#2.6 x轴刻度，标签文字

plt.xticks(x,x\_ticks\_label,rotation = 45,fontproperties = my\_font)

#3. 绘制三条折线走势

#plot 折线图

#color 单词，#0022FF rgb(0,255)

plt.plot(x,y1,label='GMV',color="red",marker='o')

plt.plot(x,y2,label='销售额',color="orange",marker='\*')

plt.plot(x,y3,label='不含退单',color="blue",marker = '.')

#4.标记横纵轴名字与标题

plt.xlabel('月份',fontproperties=my\_font)

plt.ylabel("销售额万元",fontproperties=my\_font)

plt.title('销售额走势',fontproperties=my\_font,color='red',size=15)

#5.添加折点坐标

for a,b in zip(x,y1): plt.annotate('(%.2f)'%(b),xy=(a,b),xytext=(-10,10),textcoords='offset points')

#6.设置图例

plt.legend(prop=my\_font,loc='upper left')

#7.显示图形

#plt.show()

plt.savefig('t1.png')

图表, 折线图

描述已自动生成\*\* plt.rcParams['font.sans-serif'] = ['SimHei']  
plt.rcParams['font.serif'] = ['SimHei']  
plt.rcParams['axes.unicode\_minus'] = False \*\* 此三行根治matplotlib中文乱码

**图3 销售额走势折线图**

## 4.3 流量渠道来源分析

from PythonDataAnalyse import df

import matplotlib.pyplot as plt

from matplotlib.font\_manager import FontProperties

my\_font =FontProperties(fname='C:/Windows/Fonts/Deng.ttf',size=12,)

#1.按照渠道分组聚合，统计用户数

custom = df.groupby('chanelID')['userID'].count()

#2.设置字体

plt.rcParams['font.sans-serif']=['SimHei']

#3.使用pandas中方法直接画图

custom.plot.pie(figsize=(12,8),labels=custom.index,autopct="%1.1f%%",rotatelabels=True)

#4.设置标题

plt.title('各渠道来源用户占比',loc='left',fontproperties=my\_font,color='red',size=15)

plt.savefig('t2.png')

图表, 饼图

描述已自动生成

**图4 各渠道来源用户占比饼图**

## 4.4 用户行为, 研究周一到周日哪天订单量最高

from PythonDataAnalyse import df

import matplotlib.pyplot as plt

from matplotlib.font\_manager import FontProperties

my\_font =FontProperties(fname='C:/Windows/Fonts/Deng.ttf',size=12,)

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['font.serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

#1.按周几做聚合

week = df.groupby('weekday')['orderID'].count()

week

#2.设置横纵坐标

weekX = ['周一','周二','周三','周四','周五','周六','周日']

weekY = week.values

#3.设置X轴

plt.xticks(range(len(weekX)),weekX,fontproperties=my\_font)

#4.设置条形图

rects = plt.bar(range(len(weekX)),weekY,width=0.3,color=['r','g','b'])

#5.设置每个数据条的位置

for rect in rects:

height = rect.get\_height()

plt.text(rect.get\_x() + rect.get\_width() / 2, height+0.5, str(height),ha="center")

#6.设置标题

plt.title('用户下单时间分析',fontproperties=my\_font,color='red',size=15)

#7.显示保存

图表, 条形图

描述已自动生成plt.savefig('t3.png')

**图5 用户下单时间分析**

## 4.5 用户行为, 哪个时间段下单量最高

from PythonDataAnalyse import pd

from PythonDataAnalyse import df

import matplotlib.pyplot as plt

from matplotlib.font\_manager import FontProperties

my\_font =FontProperties(fname='C:/Windows/Fonts/Deng.ttf',size=12,)

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['font.serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

#1.备份整个数据

df1 = df.copy()

#2.设置时间周期30min

s = df1['orderTime'].dt.floor('30T')

#3.将下单时间转换成时间段

df1['orderTime'] = s.dt.strftime('%H:%M') + '-' + (s + pd.Timedelta(30 \* 60, unit='s')).dt.strftime('%H:%M')

df1

#4.根据时间段分组聚合

tiemdf = df1.groupby('orderTime')['orderID'].count()

tiemdf

tiemdfX = tiemdf.index

tiemdfY = tiemdf.values

tiemdfY

#5.设置画布大小

plt.figure(figsize=(20,8),dpi=80)

#6.设置样式风格

plt.style.use('ggplot')

#7.X轴形式

plt.xticks(range(len(tiemdfX)),tiemdfX,rotation=90)

#8.绘制数据条形图

rect = plt.bar(tiemdfX,tiemdfY,width=0.3,color=['orange'])

plt.title('用户下单时间段分析',fontproperties=my\_font,color='red',size=20)

plt.savefig('t4.png')

图表, 直方图

描述已自动生成

**图6 用户下单时间段分析**

## 4.6用户行为,客户情况

#1.客单价

df.orderAmount.sum()/df.userID.unique().size

#1379.8330665393187

#1.检查数据字段

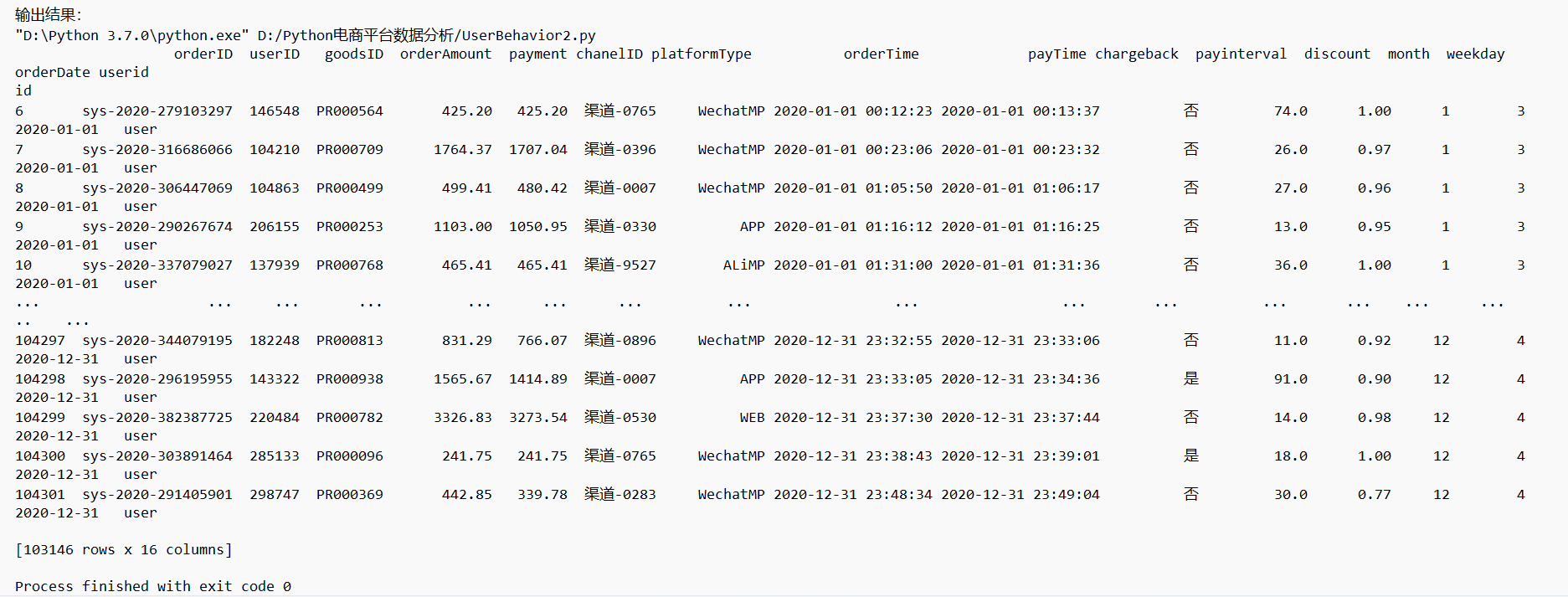
df['userid'] = df["userID"].str[0:4]

df['userid'].unique()

#2.userID只保留数字

df['userID'] = df["userID"].str[5:]

输出结果：



**图7 输出结果**

## 4.7用户行为,客户复购率

# 分析复购率

# 将用户消费数据进行数据透视。

# 统计每个用户在每月的订单量，所以userID是index，month是column。

pivoted\_counts=df.pivot\_table(index='userID',columns='month', values='orderTime',aggfunc='count').fillna(0)

pivoted\_counts.head()

print(pivoted\_counts)

日历

低可信度描述已自动生成输出结果：

**图8 输出结果**

###复购率分析图输出

from PythonDataAnalyse import df

from matplotlib.font\_manager import FontProperties

import matplotlib.pyplot as plt

my\_font =FontProperties(fname='C:/Windows/Fonts/Deng.ttf',size=12,)

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['font.serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

pivoted\_counts=df.pivot\_table(index='userID',columns='month',

values='orderTime',aggfunc='count').fillna(0)

pivoted\_counts.head()

#复购率

# 复购率的定义是在某时间窗口内消费两次及以上的用户在总消费用户中占比。这里的时间窗口是月。

#1.引入numpy

import numpy as np

#2. 将数据转换一下，消费两次及以上记为1，消费一次记为0，没有消费记为NaN。

pcRepeatBuy =pivoted\_counts.applymap(lambda x: 1 if x>1 else np.NaN if x==0 else 0)

pcRepeatBuy.head()

#3.绘图

#用sum和count相除即可计算出复购率。count是总的消费用户数，sum是两次以上的消费用户数。

(pcRepeatBuy.sum()/pcRepeatBuy.count()).plot(figsize=(15,6))

plt.title('复购率分析',fontproperties=my\_font,color='red',size=15)

图表, 折线图

描述已自动生成plt.savefig('t5.png')

**图9 复购率分析**

##各类型客户图

import matplotlib.pyplot as plt

from matplotlib.font\_manager import FontProperties

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['font.serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

my\_font =FontProperties(fname='C:/Windows/Fonts/Deng.ttf',size=12,)#1.绘制图形

rfmdf.label.value\_counts().plot.bar(figsize=(20,9))

#2.设置X轴

plt.xticks(rotation=0,fontproperties=my\_font)

图表, 条形图

描述已自动生成plt.savefig('t6.png')

**图10 各类型客户图**

# 5 结论

在Python数据分析课程中，学生应该能够：

使用不同的Python数据分析库（如pandas、numpy、matplotlib等）来处理、清理和可视化数据。

理解数据分析的基本概念，如数据类型、变量、样本大小、置信区间等。

应用数据分析方法（如描述性统计、相关分析、回归分析、异常检测等）来解释和理解数据。

使用Python和其他语言（如R、SQL等）进行数据交流和共享。

在实际项目中应用数据分析技能，如数据挖掘、预测分析等。 通过学习Python数据分析课程，学生应该能够将数据转化为有意义的信息，并且能够利用这些信息做出更好的决策。同时，他们应该能够理解和应用各种数据分析方法和技术，以解决各种领域的问题。

# 6 小结

编程是一件高精度、模范化的事情稍有疏乎都会影响全局，也可能因为某一处的小的错误而导致整个程序的无法运行。所以认真仔细就是非常重要的了。本人实现了文件的读取、利用数据进行数图结合的分析，修改部分以及部分论文的撰写。通过这次课程设计，使我对python数据分析基础教程有了更进一步的认识和了解，要想学好它要重在实践，要通过不断的上机操作才能更好地学习它，我们也发现我的许多不足之处，有对python的一些标准库函数不太了解，还有对函数调用的正确使用不够熟悉，还有对数据结构中经常出现的错误也不了解，通过实践的学习，认识到学好数据结构需要重视实践操作，在今后的专业学习中，我会更加上心，争取学好这门专业课。

# 7 参考文献

[1] 郑单青编著.python数据分析基础教程 --北京：人民邮电出版社，2020.3

[2] 张驭宇,译Ivan Idris.Python数据分析基础教程:Numpy学习指南[M].2版.北京:人民邮电出版社，2020

[3] 杜晓春,译.Fabio Nelli. Python数据分析实战[M].北京:人民邮电出版社，2021