关卡 4-1：Pyecharts 可视化大屏搭建

华为技术有限公司

Pyecharts 可视化大屏搭建

步骤 1　2020 年 3 月各省 GMV 数据玫瑰图绘制原始代码　（共 18 分）：

需要包括使用 import 导入第三方库，get_table 函数的编写，sql 语句，和绘制玫瑰图这一整套流程的所有代码。

```
# 请把相关代码贴在这里
import psycopg2
import pandas as pd
import pyecharts.options as opts
import numpy as np

def get_table(sql,ip,port,database,user, password):
    '''
    连接 DWS，操作相应的 sql 语句来进行查询
    '''
    connection = psycopg2.connect(host=ip, port=port, database=database, user=user, password=password) #连接到对应的数据库，具体请看数据库编程的内容
    connection.set_client_encoding('utf-8') # 把编码格式换成 utf8，以防止出现乱码
#以下部分内容需要学员自己完成：
#'''1.请完成以下命令，用于创建游标操作：'''
    cursor = connection.cursor()
    cursor.execute(sql) #执行 sql 命令
#'''2.请完成以下命令，用于获取所读取数据的元祖对象：'''
    rows = cursor.fetchall()
    cursor.close() #关闭游标对象
    connection.close() #关闭数据库连接
#'''3.请完成以下命令，将元组形式的数据转化为 DataFrame 的形式，方便数据分析，注意 index 需要从 1 开始编号'''
    df = pd.DataFrame(rows, index=range(1, len(rows)+1))
    return df #函数的返回值为我们得到的 DataFrame


#注意，在 Python 里，多行的字符串文本需要用连续三个单引号或双引号包裹起来：
```

```
sql1='''SELECT ad.province AS province, SUM(o.actual_price) AS GMV
FROM target.orders o, target.address_dimension ad, target.date_dimension dd
WHERE o.address_key = ad.address_key
    AND o.add_date = dd.date_key
    AND dd.year = 2020
    AND dd.month = 3
GROUP BY ad.province;
'''
#'''4.请完成以下命令，调用之前定义的 get_table 函数，执行 sql1，同时还需要传入对应的参数。'''
df1 = get_table(sql1, "114.116.200.18", "8000", 'zwq_demo', "dbadmin", "Dws@123456")
#'''5.请完成以下命令,用于指定表的列索引为 province 和 GMV '''
df1.columns = ['province', 'GMV']

df1.head(5)

from pyecharts.charts import Pie #调用 Pie 函数，注意 P 需要大写
#把 province 这一列的所有信息，也就是所有省份名称以列表的形式存入 x 中，df1['province'].values 表示取出 province 这一列的数据
x = list(df1['province'].values)

#'''6.下一步需要把 GMV 这一列的所有数据存入 y 中，请学员参考上一条命令，完成以下内容。'''
#由于数据太大，我们将其除以 1,000,000，转化为以（百万元）为单位的数据，并保留两位小数。推荐使用列表生成式：
y = [round(x/1000000, 2) for x in list(df1['GMV'].values)]

#开始画图
pie_2020_MAR = (
    Pie()
    .add(
        "",

        #'''7.请学员完成以下内容，向 data_pair 里传入数据：'''
        data_pair = [list(z) for z in zip(x, y)],
        #'''8.请学员指定内外直径：'''
        radius = ["50%", "70%"],
```

```
            center = ["40%", "50%"],#指定圆心所在的位置
            rosetype="area", #area 表示以环的大小来反映数据大小，也就是玫瑰图的特性
        )
        # 设定全局的配置项
        .set_global_opts(

            #""9.请学员调用 opts.TitleOpts 设定标题，标题为“2020 年 3 月各省 GMV 统计图”:
"'
            title_opts= opts.TitleOpts("2020 年 3 月各省 GMV 统计图", pos_left='center'),

            legend_opts=opts.LegendOpts(
                #设定图例，scroll 表示可滚动查看，pos_right 表示距离图片右端的距离，
orient="vertical"表示图例垂直排列
                type_="scroll", pos_right="2%", orient="vertical"
            )
        )
        #""10.请学员设定系列配置项，在括号内指定标签形式为 省份：数据+百万元 的形式"'
        .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}:{c}百万元"))
)
pie_2020_MAR.render_notebook() #在 Jupyter Notebook 里编译图像
```

步骤 2 分析各省用户数量，在中国地图上绘制热力图 （共 12 分）：

需要包括从 sql 语句开始直到 render 前的所有代码。

```
# 请把相关代码贴在这里
sql2="'SELECT ad.province, COUNT(DISTINCT o.user_key) AS totaluser
FROM target.orders o, target.address_dimension ad, target.date_dimension dd
WHERE o.add_date = dd.date_key
      AND o.address_key = ad.address_key
      AND dd.year = 2020
GROUP BY ad.province
ORDER BY COUNT(DISTINCT o.user_key) DESC;
"' #DESC 表示从大到小排列
```

```
#'''11.请完成以下命令，调用上面定义的 get_table 函数，执行 sql2，如有需要，注意把 ip 换成自己对应的 ip 地址: '''
df2 = get_table(sql2, "114.116.200.18", "8000", 'zwq_demo', "dbadmin", "Dws@123456")

#'''12.请完成以下命令,用于指定表的列索引为 province 和 totaluser '''
df2.columns = ['province', 'totaluser']

df2.head(5)

from pyecharts.charts import Map # 调用 Map 函数，注意 M 大写
# 把省份名称以列表方式写入 x 中。
# 由于 Pyecharts 默认的省份名称都是简写，因此我们需要把“省”、“市”、“xx 族自治区”等信息都去掉，只保留简写。
# 数据中未包含港澳台地区，除了黑龙江和内蒙古的简写是三个字，其余都是两个字，因此可直接按需截取：
x = [p[:2] if p not in ['黑龙江省','内蒙古自治区'] else p[:3] for p in df2['province'].values]
# 把用户数量这一列的所有信息存入 y 中。注意要强制转化为 int 类型，否则 numpy.int64 会无法正常显示。
y = [int(x) for x in df2['totaluser'].values]
map_user_num = (
    Map()
    .add(
        "用户数量", #指定系列名称

        #'''13.请学员完成以下内容，向 data_pair 里传入数据：'''
        data_pair = [list(z) for z in zip(x, y)],

        maptype='china', #指定地图类型是中国地图

        #'''14.请学员完成以下内容，设定显示标签：'''
        label_opts= opts.LabelOpts(is_show=True, position="top"),
    )
    .set_global_opts(
        #'''15.请学员调用 opts.TitleOpts 设定标题，标题为“各省用户量统计图”：'''
        title_opts= opts.TitleOpts("各省用户量统计图", pos_left='center'),
```

```
            #"'16.使用 opts.VisualMapOpts，指定 color bar 的显示范围为 90 到 1950：'"
            visualmap_opts=opts.VisualMapOpts(max_=1950, min_=90),

            #"'17.使用 opts.LegendOpts，指定图例的右端离画布的右端大约 20%的距离：'"
            legend_opts=opts.LegendOpts(pos_right="20%")
        )
)
map_user_num.render_notebook()
```

步骤 3 分析各商品销量及销售金额，同一副图里显示折线图与柱状图（共 18 分）：

需要包括从 sql 语句开始直到 render 前的所有代码。

```
# 请把相关代码贴在这里
sql3='''SELECT gd.goods_sn, gd.goods_name, SUM(og.number) AS totalnum,
SUM(og.price*og.number) AS totalsales
FROM target.order_goods og
      LEFT JOIN target.goods_dimension gd ON og.goods_key = gd.goods_key
GROUP BY gd.goods_sn, gd.goods_name, gd.category_name
ORDER BY SUM(og.number) DESC
'''

#"'18.请完成以下命令，调用上面定义的 get_table 函数，执行 sql3，如有需要，注意把 ip 换成自己对应的 ip 地址: '"
df3 = get_table(sql3, "114.116.200.18", "8000", 'zwq_demo', "dbadmin", "Dws@123456")

#"'19.请完成以下命令,用于指定表的列索引为 goods_sn,goods_name,totalnum,totalsales '"
df3.columns = ['goods_sn', 'goods_name', 'totalnum', 'totalsales']

df3.head(5)

from pyecharts.charts import Line,Bar
#"'20.请完成以下命令,将 df3 中的 goods_name 这一列以列表的形式赋值给 x：'"
x = list(df3['goods_name'].values)
```

```
#'''21.请参考之前的命令，将 df3 中的 totalnum 这一列以列表的形式赋值给 y1，并将每一个元素转化为 int 格式：'''
y1 = [int(x) for x in df3['totalnum'].values]

#'''22.请参考之前的命令，将 df3 中的 totalsales 这一列以列表的形式赋值给 y1，并将每一个元素除以 1,000,000，转化为百万元的格式，并保留两位小数：'''
y2 = [round(x/1000000, 2) for x in df3['totalsales'].values]
#先绘制折线图：
line1=(
    #设计风格
    #'''23.请调用 opts.InitOpts 方法，将画布的宽设为 1500px，高度设为 600px：'''
    Line(init_opts = opts.InitOpts(width='1500px',height='600px'))

    .add_xaxis(x) #添加 x 轴数据
    .add_yaxis('商品销量',y1) #将 y1 数据添加到 y 轴，注意'商品销量'指定了系列名称，如果不想要系列名称，需要用空字符串''来填充
    #下面的命令其实是添加了一个新的 y 轴，因为销售金额和销量的数量级相差太多，因此需要单独的纵轴。
    #注意，添加新的纵坐标，需要在第一次初始化图的时候就完成，而不是等到绘制新数据的时候再进行
    #比如在这里，虽然我们还没开始绘制销售金额，但对应的纵坐标需要先添加完成：
    .extend_axis(yaxis=opts.AxisOpts(axislabel_opts=opts.LabelOpts(formatter="{value} (百万元)")))
    .set_global_opts( #全局配置项
        #设置全局参数
        datazoom_opts=opts.DataZoomOpts(range_end=500,is_zoom_lock=False), #允许缩放
        title_opts=opts.TitleOpts(title="商品销量及销售金额统计图", pos_left="40%"),#设置 title

        #'''24.请调用 opts.LabelOpts 方法，将 x 轴上的标签顺时针旋转 23 度'''
        xaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate="-23")),
        legend_opts=opts.LegendOpts(pos_right="30%")
    )
)
bar2=(
```

```
#'''25.请完成柱状图的绘制，包括
#1）使用 Bar 方法创建柱状图对象，并调用 opts.InitOpts 方法，将画布的宽设为 1500px，高度设为 600px；
Bar( init_opts= opts.InitOpts(width='1500px',height='600px') )

#2）添加 x 轴的数据；
.add_xaxis(x)

#3）将 y2 数据添加到 y 轴，注意需要指定 yaxis_index=1：'''
.add_yaxis("销售金额", y2, yaxis_index=1)

)

line1.overlap(bar2) #调用 overlap 函数就可以完成同一张图里显示折线图和柱状图了
line1.render_notebook()
```

步骤 4　下钻与上卷（共 12 分）：

需要包括 sql 语句，全国各省 GMV 统计图的绘制，用 for 循环插入每个省份所辖各区市 GMV 统计图的 JS 代码，以及最后写入到 drill_down.html 这几个部分的代码，**但不包括各 HTML 文件的 JS 代码本身，以及不包括仅为了方便理解的浙江省的例子。**

```
# 请把相关代码贴在这里
sql4 = '''SELECT ad.province AS province, ad.city AS city, SUM(o.actual_price) AS GMV
FROM target.orders o, target.address_dimension ad, target.date_dimension dd
WHERE o.address_key = ad.address_key
    AND o.add_date = dd.date_key
    AND to_date(dd.full_date) BETWEEN to_date('2020/1/1') AND current_date
    AND province not in ('北京市', '上海市', '天津市', '重庆市')
GROUP BY ad.province, ad.city
UNION ALL
SELECT ad.province AS province, ad.county AS city, SUM(o.actual_price) AS GMV
FROM target.orders o, target.address_dimension ad, target.date_dimension dd
WHERE o.address_key = ad.address_key
    AND o.add_date = dd.date_key
    AND to_date(dd.full_date) BETWEEN to_date('2020/1/1') AND current_date
```

```
    AND province in ('北京市', '上海市', '天津市', '重庆市')
GROUP BY ad.province, ad.county;
'''
#'''26.请完成以下命令，调用上面定义的 get_table 函数，执行 sql4，如有需要，注意把 ip 换成自己对应的 ip 地址: '''
df4 = get_table(sql4, "114.116.200.18", "8000", 'zwq_demo', "dbadmin", "Dws@123456")

#'''27.请完成以下命令,用于指定表的列索引为  province, city, GMV '''
df4.columns = ['province', 'city', 'GMV']

df4.head(5)

#从原始数据中获取各省对应的数据：
#groupby 方法表示把关键词相同的数据视为一个整体，比如这里我们把'province'相同的数据放在了一起，然后取出其中名为'GMV'的那一列
#sum 方法就是对所有元素进行求和，也就是对 GMV 数据进行求和。
df_GMV_province = df4.groupby(['province'])['GMV'].sum()
#'''28.请完成以下命令,将省份名称取出，作为列表传入 province_name 里'''
province_name = list(set(list(df4['province'].values)))
print(province_name) #输出看一下是否正确显示为列表。注意每个省份只出现一次。

#'''29.请完成以下命令,将 GMV 数据取出，转化为百万元，保留两位小数，作为列表传入 GMV_province 里'''
GMV_province = [round(x/1000000, 2) for x in df_GMV_province]
print(GMV_province) #输出看一下是否正确显示为列表，并且是否已经保留两位小数。

#开始画图
bar_province = (
    #'''30.请完成以下步骤，用于绘制各省 GMV 总量柱状图，包括：
    # 1) 初始化画布，宽 1500px，高 500px；'''
    Bar( init_opts= opts.InitOpts(width='1500px',height='600px') )

    # 2) 添加省份名称为 x 轴数据；
    .add_xaxis(province_name)
```

```
        # 3) 添加 GMV_province 数据到 y 轴，指定系列名称为：GMV(百万元)'"
        .add_yaxis("GMV(百万元)", GMV_province)

        .set_global_opts(
            title_opts=opts.TitleOpts(title="各省 GMV 总量统计图",pos_left='20%'), #设置标题
            datazoom_opts=opts.DataZoomOpts(range_end=500,is_zoom_lock=False), #允许缩放
            yaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(formatter = '{value} (百万元)')), #设置标签类型为： 数值（百万元）
            #'"4) 设定标签的配置项，将 x 轴上的标签顺时针旋转 30 度'"
            xaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate="-30")),
        )
)
bar_province.render('D:\\bar_province.html') #编译为 HTML 文件，注意此时必须完成编译

df_GMV_cities = [] #初始化一个空列表，用于存入每个省份下划区市对应的数据
for p in province_name: #遍历每一个省份
    df_GMV_cities.append(df4[df4['province']==p].groupby(['city'])['GMV'].sum())

    id_province = bar_province.chart_id #获取当前柱状图的 id，以便完成后续的上卷下钻操作
#编写触发条件：鼠标点击事件，由于鼠标点击事件仅有一个，所以放在循环外：
add_in = [' chart_%s.on("click", function (params) {\n'% id_province]

import numpy as np
for i,p in enumerate(province_name):
    #对每一个省份都写入下钻操作的相关代码。由于下钻操作的原理相同，因此可用循环来完成。
    city_name = list(df_GMV_cities[i].index) #地级市（市辖区）名称
    GMV_city = ['%.2f' % (x/1000000) for x in df_GMV_cities[i].values] #对应的 GMV 数据
    sorted_index = [i for i,v in sorted(enumerate(GMV_city), key=lambda x:x[1])] #对数据进行排序，保存排序后的索引
    city_name = list(np.array(city_name)[sorted_index]) #按照索引调整地级市（市辖区）名称
    GMV_city = list(np.array(GMV_city)[sorted_index]) #按照索引调整 GMV 数据
```

```
    #画图:
    bar_city = (
        Bar(init_opts = opts.InitOpts(width='1500px',height='500px'))
        .add_xaxis(city_name)
        .add_yaxis('GMV(百万元)',GMV_city)
        .set_global_opts(
            title_opts=opts.TitleOpts(title="%s 各市（区）GMV 总量统计图" %p,
pos_left='20%'),
            datazoom_opts=[opts.DataZoomOpts(range_end=500,is_zoom_lock=False)],
            xaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate=-30)),
            yaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(formatter = '{value} (百
万元)')),
        )
    )
    bar_city.render('D:\\bar_city.html') #编译

    f2 = open('D:\\bar_city.html','r') #打开 p 这个省下属各地级市（市辖区）对应柱状图的
HTML 源文件。
    content2 = f2.readlines()
    f2.close() #使用 open 函数后一定要记得 close。

    id_city = bar_city.chart_id #获取当前柱状图的 id，以便完成后续的上卷下钻操作
    if i == 0 : #对于第一次写入，我们要写入 if 条件；而对于剩下的，我们要写入 else if
条件。
        add_in = add_in+\
                [' if (params.name=="%s"){\n' % p]+\
                [' chart_%s.clear();\n'% id_province]+\
                content2[13:-4]+\
                [' chart_%s.setOption(option_%s);\n'% (id_province,id_city)]+\
                [' }\n']
    else:
        add_in = add_in+\
                [' else if (params.name=="%s"){\n' % p]+\
                [' chart_%s.clear();\n'% id_province]+\
                content2[13:-4]+\
                [' chart_%s.setOption(option_%s);\n'% (id_province,id_city)]+\
                [' }\n']
```

```
#以上是下钻的全部内容
#下面完成上卷。只要鼠标点击的不是省份名称，就返回到初始界面。
add_in = add_in+\
            [' else {\n']+\
            [' chart_%s.clear();\n'% id_province]+\
            [' chart_%s.setOption(option_%s);\n'% (id_province,id_province)]+\
            [' };\n']+\
            [' });\n']

f1 = open('D:\\bar_province.html','r') #打开省份柱状图的 HTML 源文件
content = f1.readlines()
f1.close()

new_content = [ele for ele in content[:-3]] +add_in+\
      [ele for ele in content[-3:]] #新 HTML 文件就是在 bar_province.html 文件的倒数第四行插入我们的鼠标点击事件以及所有的新图表的数据与配置项。

#创建一个新的 HTML 文件，将我们的内容写进该文件中
#注意，'w'表示只写，如果目录下存在同名文件是会被新文件覆盖的。
f3 = open('D:\\drill_down.html','w',encoding='utf-8-sig') #encoding='utf-8-sig'为了保证中文显示正确
f3.writelines(new_content) #使用 writelines 方法，把一整个列表一次性写入 drill_down.html 文件中
f3.close() #别忘了解除占用
```

最后请把 my_visualization_张三.html 以文件的格式插入到这里（该文档最后附有如何插入文件到 word 文档的具体操作步骤）：

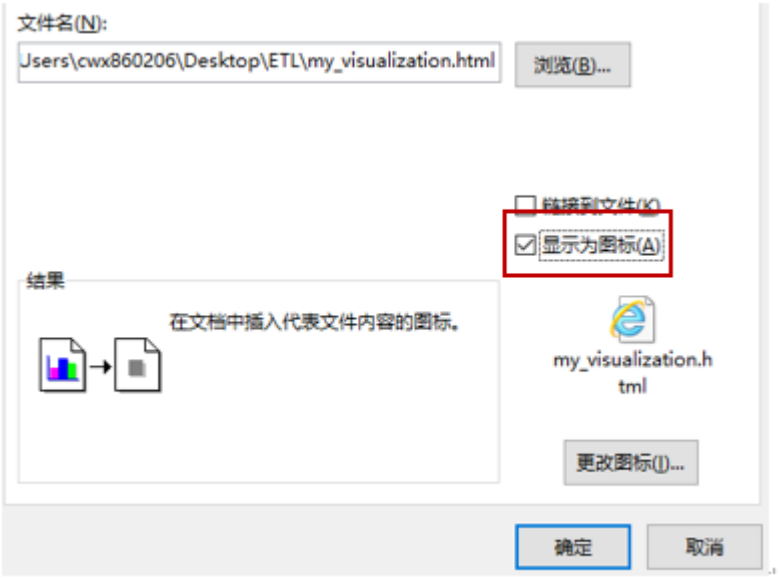
my_visualization_
张雯琪.html

点击 word 左上角的"插入"选项：

在右边看到“对象”选项，点击即可插入对象：

选择“由文件创建”－ “浏览” － 找到 D 盘下的 my_visualization_张三.HTML 文件：

选择“显示为图标”：



点击确定，即可添加文件到 word 中。

关卡 4-2：数据挖掘与分析

华为技术有限公司

基于 Litemall 的数据挖掘与分析

步骤 1 回归算法必做题（10 分）：

1. 请将 df_order_data.head(5)的结果截图并提交：

Out[7]:

	order_id	order_status	user_key	order_price	add_date
1	1	401	12251	3688.00	20200409
2	2	401	1516	2199.00	20200229
3	4	401	7635	2199.00	20200226
4	5	401	6428	2199.00	20200120
5	8	401	3088	16999.00	20200129

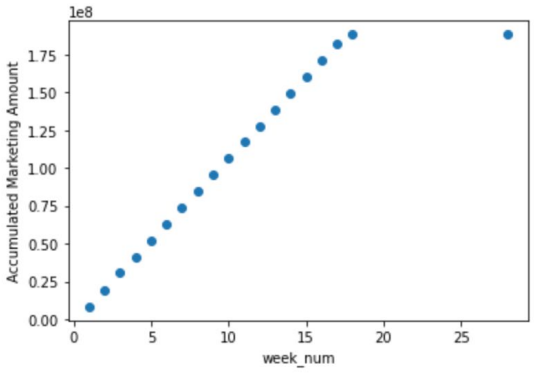
2. 请将 df.head(5)的结果截图并提交：

Out[8]:

	order_price	add_date	week_num
1	3688.00	20200409	15
2	2199.00	20200229	9
3	2199.00	20200226	9
4	2199.00	20200120	4
5	16999.00	20200129	5

3. 请将散点图的结果截图并提交：

Out[9]: Text(0, 0.5, 'Accumulated Marketing Amount')



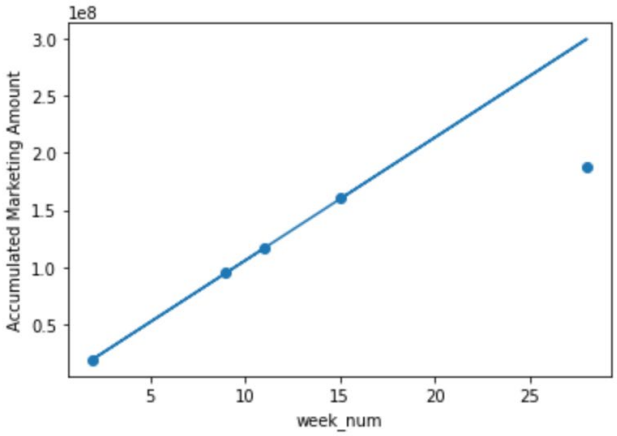
4. 请将线性回归的模型结果截图并提交：

In [12]:

```
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression() #生成线性回归模型对象。所有参数使用
model_lr.fit(x_train, y_train) #调用fit方法，完成模型的训练
print('The function should be y=%dx+%d'%(model_lr.coef_,model_lr.
```

```
The function should be y=10740699x+-1515131
```

5. 请将线性回归模型在测试集上的拟合曲线图截图并提交：



6. 请将线性回归模型的均方误差截图并提交：

```
In [14]:
from sklearn.metrics import mean_squared_error
print('The MSE of LR is {:e}'.format(mean_squared_error(y_test/np.
```

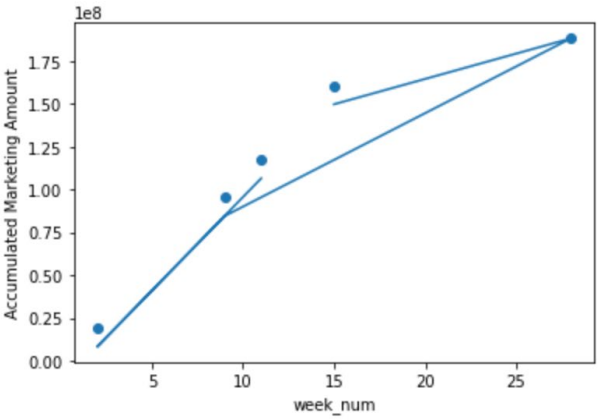
```
The MSE of LR is 2.916922e-02
```

7. 请将线性回归模型所预测得到的上半年的营销总额结果截图并提交：

```
In [15]:
model_lr.predict([[26]])
```

```
Out[15]: array([2.77743058e+08])
```

8. 请将决策树回归模型在测试集上拟合的曲线图截图并提交：

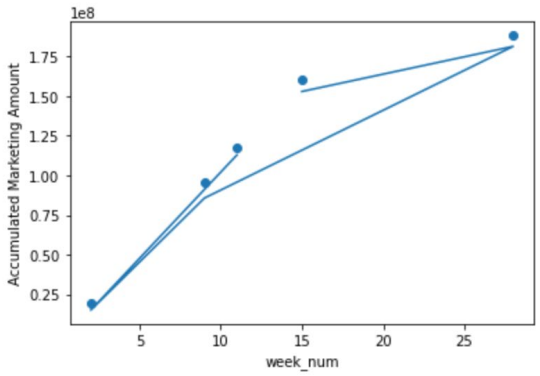


9. 请将决策树回归模型的均方误差截图并提交：

```
In [17]:
print('The MSE of DT is {:e}'.format(mean_squared_error(y_test/np.
```

```
The MSE of DT is 1.126153e-03
```

10. 请将随机森林回归模型在测试集上拟合的曲线图截图并提交：



11. 请将随机森林回归模型的均方误差截图并提交：

```
In [19]: print('The MSE of RF is {:e}'.format(mean_squared_error(y_test/np.
```

```
The MSE of RF is 5.661071e-04
```

步骤 2 关联算法必做题（10 分）：

1. 请将 df_order .head(5)的结果截图并提交：

Out[21]:

	order_id	order_status	user_key
1	1	401	12251
2	2	401	1516
3	4	401	7635
4	5	401	6428
5	8	401	3088

2. 请将 df_order_goods.head(5)的结果截图并提交：

Out[22]:

	order_id	goods_key
1	2	6
2	8	1
3	12	7
4	15	4
5	20	9

3. 请将 df.head(5) 的结果截图并提交：

Out[23]:

	order_id	order_status	user_key	goods_key
0	1	401	12251	4
1	2	401	1516	6
2	4	401	7635	6
3	5	401	6428	6
4	8	401	3088	1

4. 请将 df_new.head(5) 的结果截图并提交：

Out[25]:

	order_id	order_status	user_key	goods_key	goods_type
0	1	401	12251	4	Phone
1	2	401	1516	6	Phone
2	4	401	7635	6	Phone
3	5	401	6428	6	Phone
4	8	401	3088	1	Phone

5. 请将关联算法的输出结果（频繁项集）截图并提交：

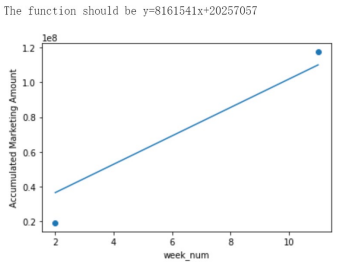
```
frequent itemset:
 [['Earphone'], ['Huawei-Pad'], ['Phone'], ['Earphone', 'Phon
e'], ['Huawei-Pad', 'Phone']]
```

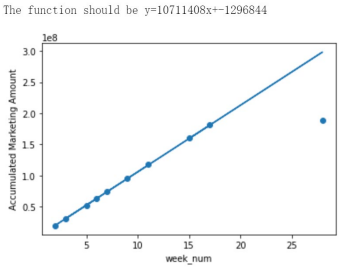
6. 请大家在此处提交自己的关于上述结果的分析，以及基于此结果为 litemall 提供的经营策略：

步骤 3 可供选择的创新实践结果（20 分）：

1. 在划分测试集时，选用不同的参数，比如使用不同的 test_size 会带来什么影响？

答：test_size 即测试集占总数据的大小，下图分别为 test_size=0.1 和 0.5 时的线性回归拟合

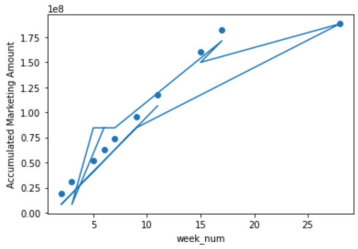




Test_size 过小则测试数据点较少，不利于建立预测模型，test_size 过大则容易过拟合

2. 在使用决策树回归模型时，选用不同的参数，会带来什么样的影响？为什么会有（或没有）这样的影响？

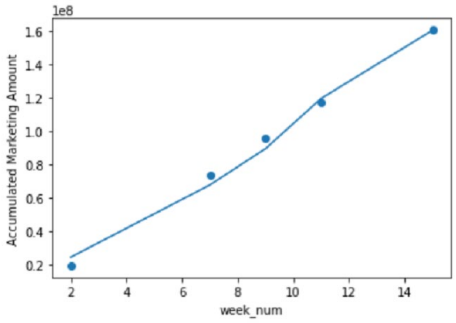
答：下图为 DecisionTreeRegressor 方法参数 splitter 选择随机策略，也就是在随机特征中产生分支



3. 使用网格搜索来找到随机森林最佳参数的实现代码与结果：

答

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
model_RF = RandomForestRegressor(n_estimators=10)
grid = GridSearchCV(estimator=model_RF, param_grid={'n_estimators':range(5,20)}, cv=6)
grid.fit(x_train, y_train)
y_pred5 = grid.predict(x_test)
plt.figure(5)
plt.scatter(x_test,y_test) #绘制真实值的散点图
plt.plot(x_test,y_pred5) #绘制预测值的折线图
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```

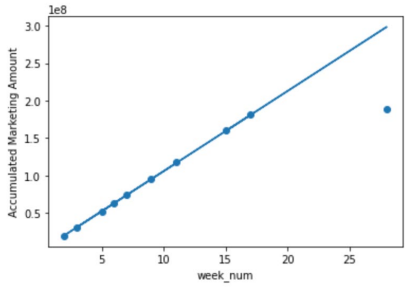


4. 其他回归算法实现的代码与结果截图。这些方法与实验中介绍的方法相比有哪些优缺点，或是有什么联系吗?

答：lasso 回归算法实现

```
from sklearn.datasets import make_regression
from sklearn.linear_model import Lasso
lasso = Lasso()
lasso.fit(x_train, y_train)
print("The function should be y = %dx + %d"%(lasso.coef_, lasso.intercept_))
y_pred6 =    lasso.predict(x_test)
plt.figure(6)
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pred6)
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```

```
The function should be y = 10711407x + -1296844
```



Lasso 回归在优化函数中增加一个偏置项，以减少共线性的影响，从而减少模型方差。

5. 在回归问题中，使用其他的模型评估标准，比如 MAE 或R^2误差等等的结果截图。有关这几种评估标准的优缺点的分析与思考。

答:

In [31]:

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error

from sklearn.metrics import r2_score

print('The MSE of DT is {:e}'.format(mean_squared_error(y_test/np.
print('The MAE of DT is {:e}'.format(mean_absolute_error(y_test, y
print('The RMSE of DT is {:e}'.format(np.sqrt(mean_squared_error(y
print('The R^2 of DT is {:e}'.format(r2_score(y_test, y_pred2)))
```

```
The MSE of DT is 1.069023e-03
The MAE of DT is 1.080912e+07
The RMSE of DT is 1.182019e+07
The R^2 of DT is 9.591370e-01
```

MSE 均方误差，由真实值-预测值，平方之后求和平均得到；RMSE 均方根误差由 MSE 开根号得到，便于数据的更好描述；MAE 平均绝对误差由差的绝对值取平均得到。这 3 种方法都适用于误差评估，但缺点在于其值不具有上下限。线性回归通常采用 R Squared 作为评估指标，其值往上越接近 1 说明模型越好，其值往下越接近 0 说明和基准模型越接近。

6. 在关联算法中，使用不同支持度带来的不同结果的截图。关于不同支持度可能带来的不同的结果，你有什么想法?

答：下图为最小支持度 minSup 取 0.5 和 0.9 的结果

```
frequent itemset:
 [['Earphone'], ['Huawei-Pad'], ['Laptop'], ['Phone'], ['Wearabl
e_Device'], ['Earphone', 'Huawei-Pad'], ['Earphone', 'Laptop'],
['Earphone', 'Phone'], ['Earphone', 'Wearable_Device'], ['Huawei
-Pad', 'Laptop'], ['Huawei-Pad', 'Phone'], ['Huawei-Pad', 'Weara
ble_Device'], ['Laptop', 'Phone'], ['Phone', 'Wearable_Device'],
['Earphone', 'Huawei-Pad', 'Laptop'], ['Earphone', 'Huawei-Pad',
'Phone'], ['Earphone', 'Huawei-Pad', 'Wearable_Device'], ['Earph
one', 'Huawei-Pad', 'Laptop', 'Phone'], ['Earphone', 'Huawei-Pa
d', 'Phone', 'Wearable_Device'], ['Earphone', 'Laptop', 'Phon
e'], ['Earphone', 'Huawei-Pad', 'Laptop', 'Wearable_Device'],
['Earphone', 'Phone', 'Wearable_Device'], ['Huawei-Pad', 'Lapto
p', 'Wearable_Device'], ['Huawei-Pad', 'Laptop', 'Phone', 'Weara
ble_Device'], ['Huawei-Pad', 'Phone', 'Wearable_Device'], ['Earp
hone', 'Huawei-Pad', 'Laptop', 'Phone'], ['Earphone', 'Huawei-Pa
d', 'Laptop', 'Phone', 'Wearable_Device'], ['Earphone', 'Lapto
p', 'Phone', 'Wearable_Device'], ['Huawei-Pad', 'Laptop', 'Phon
e', 'Wearable_Device']]
```

```
frequent itemset:
 [['Earphone'], ['Phone'], ['Earphone', 'Phone']]
```

支持度表示同时包含 A 和 B 的事务占所有事务的比例，体现关联规则的强度

7. 在 Python 中实现其他的关联算法，比如 FP-growth 算法，FreeSpan 算法等等。

答： FP-growth 算法实现

```
class treeNode:
    def __init__(self, nameValue, numOccur, parentNode):
        self.name = nameValue    #节点元素名称，在构造时初始化为给定值
        self.count = numOccur     # 出现次数，在构造时初始化为给定值
        self.nodeLink = None      # 指向下一个相似节点的指针，默认为 None
```

```
        self.parent = parentNode     # 指向父节点的指针，在构造时初始化为给定值
        self.children = {}   # 指向子节点的字典，以子节点的元素名称为键，指向子节点的指针为值，初始化为空字典

    # 增加节点的出现次数值
    def inc(self, numOccur):
        self.count += numOccur

    # 输出节点和子节点的 FP 树结构
    def disp(self, ind=1):
        print('  ' * ind, self.name, ' ', self.count)
        for child in self.children.values():
            child.disp(ind + 1)

# 对不是第一个出现的节点，更新头指针块。就是添加到相似元素链表的尾部
def updateHeader(nodeToTest, targetNode):
    while (nodeToTest.nodeLink != None):
        nodeToTest = nodeToTest.nodeLink
    nodeToTest.nodeLink = targetNode

# 根据一个排序过滤后的频繁项更新 FP 树
def updateTree(items, inTree, headerTable, count):
    if items[0] in inTree.children:
        # 有该元素项时计数值+1
        inTree.children[items[0]].inc(count)
    else:
        # 没有这个元素项时创建一个新节点
        inTree.children[items[0]] = treeNode(items[0], count, inTree)
        # 更新头指针表或前一个相似元素项节点的指针指向新节点
        if headerTable[items[0]][1] == None:    # 如果是第一次出现，则在头指针表中增加对该节点的指向
            headerTable[items[0]][1] = inTree.children[items[0]]
        else:
            updateHeader(headerTable[items[0]][1], inTree.children[items[0]])

    if len(items) > 1:
        # 对剩下的元素项迭代调用 updateTree 函数
```

```
            updateTree(items[1::], inTree.children[items[0]], headerTable, count)

# 主程序。创建 FP 树。dataSet 为事务集，为一个字典，键为每个事物，值为该事物出现的次数。minSup 为最低支持度
def createTree(dataSet, minSup=1):
    # 第一次遍历数据集，创建头指针表
    headerTable = {}
    for trans in dataSet:
        for item in trans:
            headerTable[item] = headerTable.get(item, 0) + dataSet[trans]
    # 移除不满足最小支持度的元素项
    keys = list(headerTable.keys()) # 因为字典要求在迭代中不能修改，所以转化为列表
    for k in keys:
        if headerTable[k] < minSup:
            del(headerTable[k])
    # 空元素集，返回空
    freqItemSet = set(headerTable.keys())
    if len(freqItemSet) == 0:
        return None, None
    # 增加一个数据项，用于存放指向相似元素项指针
    for k in headerTable:
        headerTable[k] = [headerTable[k], None]  # 每个键的值，第一个为个数，第二个为下一个节点的位置
    retTree = treeNode('Null Set', 1, None) # 根节点
    # 第二次遍历数据集，创建 FP 树
    for tranSet, count in dataSet.items():
        localD = {} # 记录频繁 1 项集的全局频率，用于排序
        for item in tranSet:
            if item in freqItemSet:    # 只考虑频繁项
                localD[item] = headerTable[item][0] # 注意这个[0]，因为之前加过一个数据项
        if len(localD) > 0:
            orderedItems = [v[0] for v in sorted(localD.items(), key=lambda p: p[1], reverse=True)] # 排序
            updateTree(orderedItems, retTree, headerTable, count) # 更新 FP 树
    return retTree, headerTable
```

```
# 直接修改 prefixPath 的值，将当前节点 leafNode 添加到 prefixPath 的末尾，然后递归添加其父节点。
# prefixPath 就是一条从 treeNode（包括 treeNode）到根节点（不包括根节点）的路径
def ascendTree(leafNode, prefixPath):
    if leafNode.parent != None:
        prefixPath.append(leafNode.name)
        ascendTree(leafNode.parent, prefixPath)

# 为给定元素项生成一个条件模式基（前缀路径）。basePet 表示输入的频繁项，treeNode 为当前 FP 树中对应的第一个节点
# 函数返回值即为条件模式基 condPats，用一个字典表示，键为前缀路径，值为计数值。
def findPrefixPath(basePat, treeNode):
    condPats = {}  # 存储条件模式基
    while treeNode != None:
        prefixPath = []  # 用于存储前缀路径
        ascendTree(treeNode, prefixPath)  # 生成前缀路径
        if len(prefixPath) > 1:
            condPats[frozenset(prefixPath[1:])] = treeNode.count  # 出现的数量就是当前叶子节点的数量
        treeNode = treeNode.nodeLink  # 遍历下一个相同元素
    return condPats

# 根据事务集获取 FP 树和频繁项。
# 遍历频繁项，生成每个频繁项的条件 FP 树和条件 FP 树的频繁项
# 这样每个频繁项与他条件 FP 树的频繁项都构成了频繁项集

# inTree 和 headerTable 是由 createTree()函数生成的事务集的 FP 树。
# minSup 表示最小支持度。
# preFix 请传入一个空集合（set([])），将在函数中用于保存当前前缀。
# freqItemList 请传入一个空列表（[]），将用来储存生成的频繁项集。
def mineTree(inTree, headerTable, minSup, preFix, freqItemList):
    # 对频繁项按出现的数量进行排序进行排序
```

```
    sorted_headerTable = sorted(headerTable.items(), key=lambda p: p[1][0])  #返回重新排序的列表。每个元素是一个元组，[（key,[num,treeNode],()）
    bigL = [v[0] for v in sorted_headerTable]  # 获取频繁项
    for basePat in bigL:
        newFreqSet = preFix.copy()  # 新的频繁项集
        newFreqSet.add(basePat)  # 当前前缀添加一个新元素
        freqItemList.append(newFreqSet)  # 所有的频繁项集列表
        condPattBases = findPrefixPath(basePat, headerTable[basePat][1])  # 获取条件模式基。就是 basePat 元素的所有前缀路径。它像一个新的事务集
        myCondTree, myHead = createTree(condPattBases, minSup)  # 创建条件 FP 树

        if myHead != None:
            # 用于测试
            print('conditional tree for:', newFreqSet)
            myCondTree.disp()
            mineTree(myCondTree, myHead, minSup, newFreqSet, freqItemList)  # 递归直到不再有元素

# 将数据集转化为目标格式
def createInitSet(dataSet):
    retDict = {}
    for trans in dataSet:
        retDict[frozenset(trans)] = 1
    return retDict

D = [list(group[-1]) for group in df_new.groupby('user_key')['goods_type']]
minSup =5
initSet = createInitSet(D)  # 转化为符合格式的事务集
myFPtree, myHeaderTab = createTree(initSet, minSup)  # 形成 FP 树
# myFPtree.disp()  # 打印树
freqItems = []  # 用于存储频繁项集
mineTree(myFPtree, myHeaderTab, minSup, set([]), freqItems)  # 获取频繁项集
print('frequent itemset:\n', freqItems)
```

```
frequent itemset:
 [{'Earphone'}, {'Wearable_Device', 'Earphone'}, {'Earphone', 'L
aptop'}, {'Huawei-Pad', 'Earphone'}, {'Earphone', 'Phone'}, {'We
arable_Device'}, {'Wearable_Device', 'Huawei-Pad'}, {'Wearable_D
evice', 'Phone'}, {'Laptop'}, {'Wearable_Device', 'Laptop'}, {'H
uawei-Pad', 'Laptop'}, {'Phone', 'Laptop'}, {'Phone'}, {'Huawei-
Pad', 'Phone'}, {'Huawei-Pad'}]
```

8. 针对我们的数据，有哪些场景可以用上分类和聚类算法，而相关的算法以及参数又要如何选择呢?

步骤 4 其他我们没有想到的创新点（上不封顶！）：

可以在这里畅所欲言，把你们所能想到的创新点付诸实践，这才是数据挖掘的意义所在: