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。

```
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)
    df.index=df.index+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','tsz_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) #查看前 5 条数据，重点关注每一列的行索引是否从 1 开始编号，以及列索引是否正确修改为 province 和 GMV。

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(i/1000000,2) for i in df1['GMV'].values]
y=[int(x/10000)/100 for x in 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 = [30, 75],
            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(title="2020 年 3 月各省 GMV 统计图"),
            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。

```
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','tsz_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) #查看前 5 条数据，重点关注每一列的行索引是否从 1 开始编号，以及列索引是否正确修改为 province 和 totaluser。

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(formatter="{b}"),
)
    .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(
            title="各省用户量统计图"
        ),

        #'''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。

```
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)
'''
#'''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','tsz_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']

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 = [int(x/10000)/100 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 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。**

```
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
from pyecharts.charts import Page
page = Page(layout=Page.DraggablePageLayout) # DraggablePageLayout 表示可拖拽布局
page.add(pie_2020_MAR,map_user_num,line1) # 注意，这里没有 drill_down 这张图
page.render("D:\\page1.html")

import re #用于正则表达式
with open("D:\\page1.html",'r',encoding="utf-8-sig") as html1:
      with open("D:\\drill_down.html",'r',encoding="utf-8-sig") as html2:
```

```
        content1 = html1.readlines() #读取 page1.html 的信息
        content2 = html2.readlines() #读取 drill_down.html 的信息
        #下面用了一个正则表达式，搜索 drill_down.html 里对应的画布 id 信息：
        drill_down_id = re.search('_[\w\d]* =',content2[11]).group().replace('_','').replace(' =','')
        #把画布 id 信息添加到 chart_id 的最后一个位置：
        content1[-27] = content1[-27][:-3]+",'"+drill_down_id+"'"+content1[-27][-3:]
        #下面用了正则表达式，把定义图表拖拽以及缩放代码中有关图表 id 的部分替换成了 drill_down.html 对应的画布 id：
        s1 = re.sub("#[\w\d]*","#"+drill_down_id,content1[-29],count=2)
        s2 = re.sub("#[\w\d]*","#"+drill_down_id,content1[-28])
        s2 = re.sub("_[\w\d]*","_"+drill_down_id,s2)
        #把新语句插入到前三条代码的后面：
        content1_new = content1[:-27]+[s1]+[s2]+content1[-27:]
        #以下是 drill_down.html 这张图表的声明过程，也就是我们需要插入作为第四张图表的声明信息
        add_info = content2[9:-2]
        #别忘了在声明信息的最前端添加<br/>：
        add_info[0] = '<br/>'+add_info[0]
        #注意，由于我们之前只放了三张图表，所以第四张图表被插在倒数第 37 行的位置。如果图表数量变多，这里的 37 是需要按实际情况调整的。具体可以打开 HTML 源文件，看一下新的图片应该放在当 前最后一个图表声明过程的后面，定义图表拖拽以及缩放的代码的前面。
        content1_new = content1_new[:-37]+add_info+content1_new[-37:]

with open("D:\\page_new.html",'w',encoding="utf-8-sig") as html_new:
    #写入文件即可
    html_new.writelines(content1_new)

Page.save_resize_html("D:\\page_new.html", cfg_file="D:\\chart_config.json",
dest="D:\\my_visualization.html")
```

my_visualization
_唐顺政.html

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）：

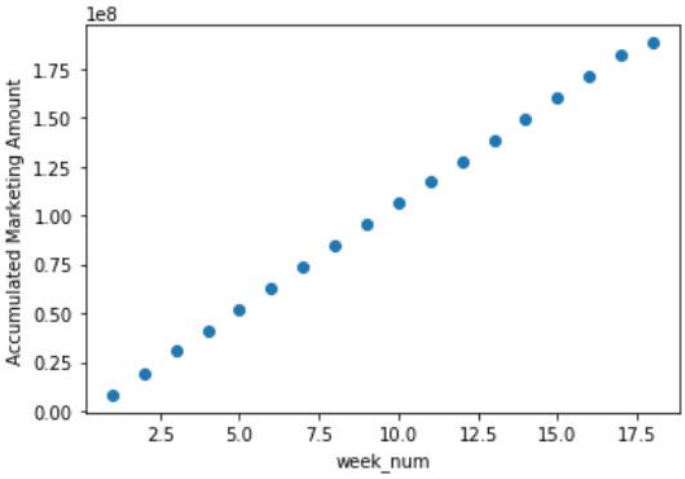
1. 请将 df_order_data.head(5)的结果截图并提交：

	order_id	order_status	user_key	order_price	add_date
1	3	401	4691	16999.00	20200425
2	11	401	2897	5899.00	20200125
3	13	401	424	2469.00	20200122
4	17	401	12927	4499.00	20200203
5	19	401	10701	5899.00	20200210

2. 请将 df.head(5)的结果截图并提交：

	order_price	add_date	week_num
1	16999.00	20200425	17
2	5899.00	20200125	4
3	2469.00	20200122	4
4	4499.00	20200203	6
5	5899.00	20200210	7

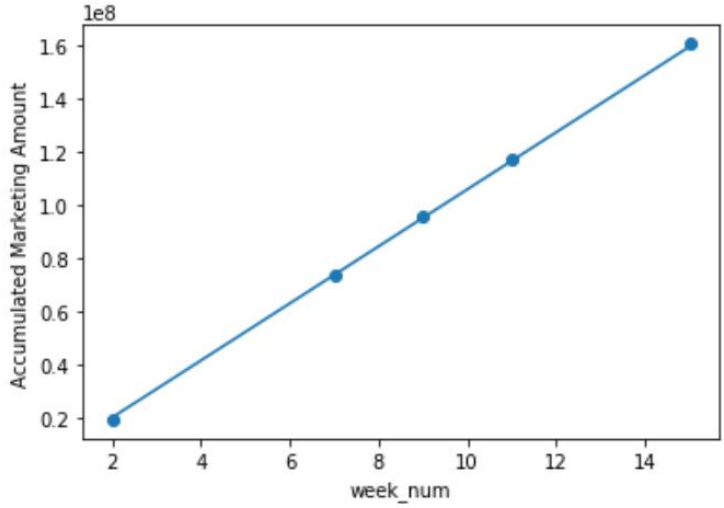
3. 请将散点图的结果截图并提交：



4. 请将线性回归的模型结果截图并提交：

```
The function should be y=10741465x+-1530868
```

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：



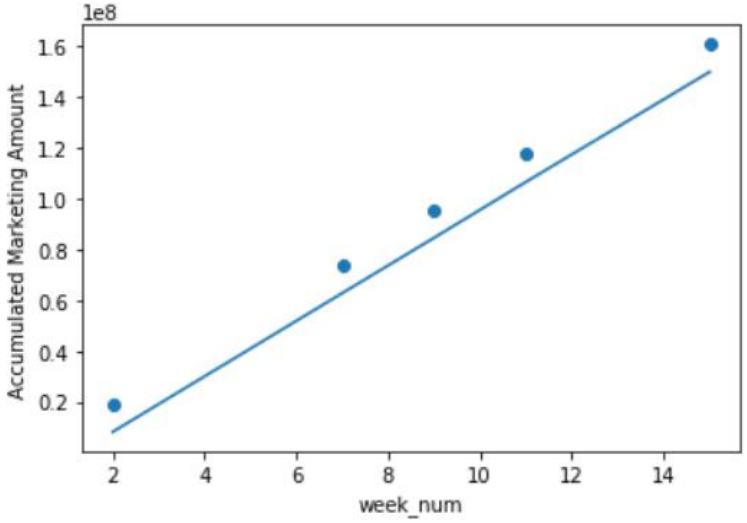
6. 请将线性回归模型的均方误差截图并提交：

```
The MSE of LR is 9.208710e-06
```

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：

```
array([2.77747226e+08])
```

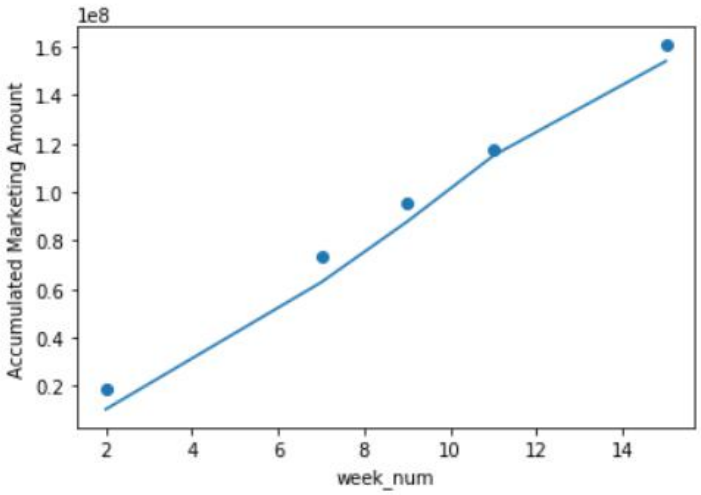
8. 请将决策树回归模型在测试集上拟合的曲线图截图并提交：



9. 请将决策树回归模型的均方误差截图并提交：

```
The MSE of DT is 2.187856e-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：

```
The MSE of RF is 1.103771e-03
```

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：

	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：

	order_id	goods_key
1	21	9
2	37	4
3	41	5
4	46	5
5	65	7

3. 请将 df.head(5) 的结果截图并提交：

	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：

	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', 'Phone'], ['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：

华为商城的耳机，平板，手机等电子产品的关联性比较强，说明顾客在在购买手机时会同时购买耳机，购买手机时也有可能同时购买华为平板，litemall 可以在这些产品的销售页面推广相关的产品，并考虑捆绑销售这些产品并提供一定的折扣，有利于提高销售量。

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?

testsize 由 0.25 增大为 0.5 到 0.75 时，测试集的比例增大，训练集的比例减小。线性回归的模型的拟合效果没有太大影响。决策树回归模型的均方误差变小，随机森林回归模型的均方误差变大，而且这两种回归模型在数据集集中的时候比较接近准确，随着数据的增大拟合效果变差，表现都没有线性回归模型好。

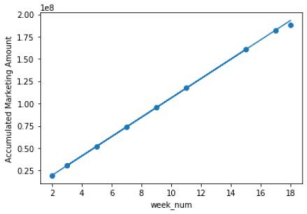
减小到 0.1 时，决策树回归模型和随机森林回归模型的均方误差都变大，拟合效果变差。

In [54]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5,random_state=0) #random_state 指定了随机的状态，使得每次随机的结果都是固定的
```

In [55]:
```
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression() #生成线性回归模型对象，所有参数使用默认参数即可，
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.intercept_)) #取出系数和截距，生成一元一次函数，该函数就是我们的一元线性回归
```

```
The function should be y=10858407x+-2322860
```

In [56]:
```
y_pred1 = model_lr.predict(x_test) #调用之前训练好的 model_lr 对象的 predict 方法，作用在测试集上
plt.figure(2) #展示在同一张图里
plt.scatter(x_test,y_test) #将测试集的实际值绘制成散点图
plt.plot(x_test,y_pred1) #将训练出来的预测值绘制成折线图
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



In [57]:
```
from sklearn.metrics import mean_squared_error
print('The MSE of LR is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test),y_pred1/np.linalg.norm(y_test))))
```

```
The MSE of LR is 2.151383e-05
```

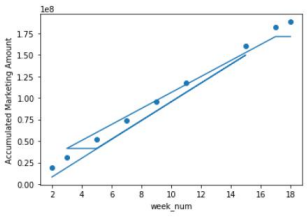
In [58]:
```
model_lr.predict([[26]])
```

Out[58]:
```
array([2.79995737e+08])
```

In [59]:
```
from sklearn.tree import DecisionTreeRegressor
model_DT = DecisionTreeRegressor() #生成决策树回归模型
model_DT.fit(x_train,y_train) #调用 fit 方法进行训练
```

Out[59]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [60]:
```
y_pred2 = model_DT.predict(x_test)
plt.figure(3)
plt.scatter(x_test,y_test) #绘制真实值的散点图
plt.plot(x_test,y_pred2) #绘制预测值的折线图
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



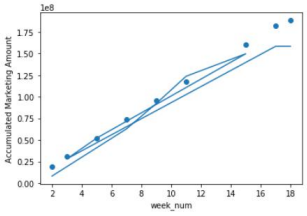
In [61]:
```
print('The MSE of DT is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test), y_pred2/np.linalg.norm(y_test))))
```

```
The MSE of DT is 1.080207e-03
```

In [62]:
```
from sklearn.ensemble import RandomForestRegressor
model_RF = RandomForestRegressor(n_estimators=10) #这里使用 10 个决策树，大家也可以试试不同数量的决策树对于结果的影响
model_RF.fit(x_train,y_train)
```

Out[62]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [63]:
```
y_pred4 = model_RF.predict(x_test)
plt.figure(5)
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pred4)
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



In [64]:
```
print('The MSE of RF is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test), y_pred4/np.linalg.norm(y_test))))
```

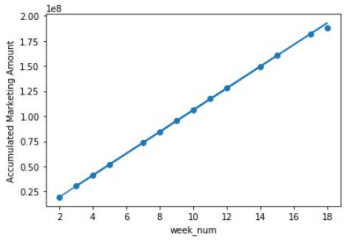
```
The MSE of RF is 1.640724e-03
```

In [82]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.75,random_state=0) #random_state 指定了随机的状态，使得每次随机的结果都是固定
```

In [83]:
```
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression() #生成线性回归模型对象，所有参数使用默认参数即可。
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.intercept_)) #取出系数和截距，生成一元一次函数，该函数就是我们的一元线性回归
```

```
The function should be y=10867558x+-2533782
```

In [84]:
```
y_pred1 = model_lr.predict(x_test) #调用之前训练好的 model_lr 对象的 predict 方法，作用在测试集上
plt.figure(2) #展示在同一张图里
plt.scatter(x_test,y_test) #将测试集的实际值绘制成散点图
plt.plot(x_test,y_pred1) #将训练出来的预测值绘制成折线图
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



In [85]:
```
from sklearn.metrics import mean_squared_error
print('The MSE of LR is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test),y_pred1/np.linalg.norm(y_test))))
```

```
The MSE of LR is 9.573259e-06
```

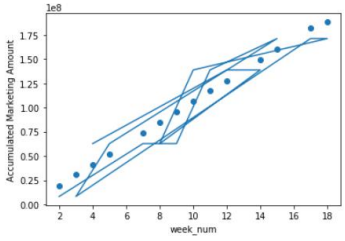
In [86]:
```
model_lr.predict([[26]])
```

Out[86]:
```
array([2.80022732e+08])
```

In [87]:
```
from sklearn.tree import DecisionTreeRegressor
model_DT = DecisionTreeRegressor() #生成决策树回归模型
model_DT.fit(x_train,y_train) #调用 fit 方法进行训练
```

Out[87]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [88]:
```
y_pred2 = model_DT.predict(x_test)
plt.figure(3)
plt.scatter(x_test,y_test) #绘制真实值的散点图
plt.plot(x_test,y_pred2) #绘制预测值的折线图
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



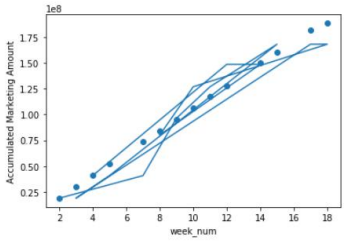
In [89]:
```
print('The MSE of DT is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test), y_pred2/np.linalg.norm(y_test))))
```

```
The MSE of DT is 1.969671e-03
```

In [90]:
```
from sklearn.ensemble import RandomForestRegressor
model_RF = RandomForestRegressor(n_estimators=10) #这里使用 10 个决策树，大家也可以试试不同数量的决策树对于结果的影响
model_RF.fit(x_train,y_train)
```

Out[90]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [91]:
```
y_pred4 = model_RF.predict(x_test)
plt.figure(5)
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pred4)
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



In [92]:
```
print('The MSE of RF is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test), y_pred4/np.linalg.norm(y_test))))
```

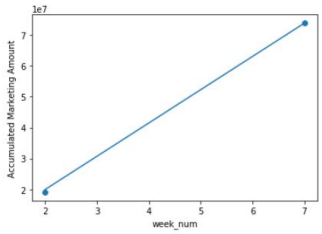
```
The MSE of RF is 1.133902e-03
```

In [93]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=0) #random_state 指定了随机的状态，使得每次随机的结果都是固定的
```

In [94]:
```
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression() #生成线性回归模型对象，所有参数使用默认参数即可。
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.intercept_)) #取出系数和截距，生成一元一次函数，该函数就是我们的一元线性回归
```

```
The function should be y=10753807x+-1516384
```

In [95]:
```
y_pred1 = model_lr.predict(x_test) #调用之前训练好的 model_lr 对象的 predict 方法，作用在测试集上
plt.figure(2) #展示在同一张图里
plt.scatter(x_test,y_test) #将测试集的实际值绘制成散点图
plt.plot(x_test,y_pred1) #将训练出来的预测值绘制成折线图
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



In [96]:
```
from sklearn.metrics import mean_squared_error
print('The MSE of LR is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test),y_pred1/np.linalg.norm(y_test))))
```

```
The MSE of LR is 6.240564e-05
```

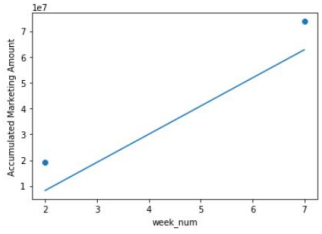
In [97]:
```
model_lr.predict([[26]])
```

Out[97]:
```
array([2.78082608e+08])
```

In [98]:
```
from sklearn.tree import DecisionTreeRegressor
model_DT = DecisionTreeRegressor() #生成决策树回归模型
model_DT.fit(x_train,y_train) #调用 fit 方法进行训练
```

Out[98]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [99]:
```
y_pred2 = model_DT.predict(x_test)
plt.figure(3)
plt.scatter(x_test,y_test) #绘制真实值的散点图
plt.plot(x_test,y_pred2) #绘制预测值的折线图
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



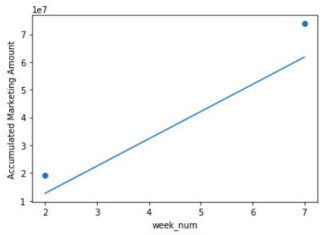
In [100]:
```
print('The MSE of DT is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test), y_pred2/np.linalg.norm(y_test))))
```

```
The MSE of DT is 2.061524e-02
```

In [101]:
```
from sklearn.ensemble import RandomForestRegressor
model_RF = RandomForestRegressor(n_estimators=10) #这里使用 10 个决策树，大家也可以试试不同数量的决策树对于结果的影响
model_RF.fit(x_train,y_train)
```

Out[101]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [102]:
```
y_pred4 = model_RF.predict(x_test)
plt.figure(5)
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pred4)
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```



In [103]:
```
print('The MSE of RF is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test), y_pred4/np.linalg.norm(y_test))))
```

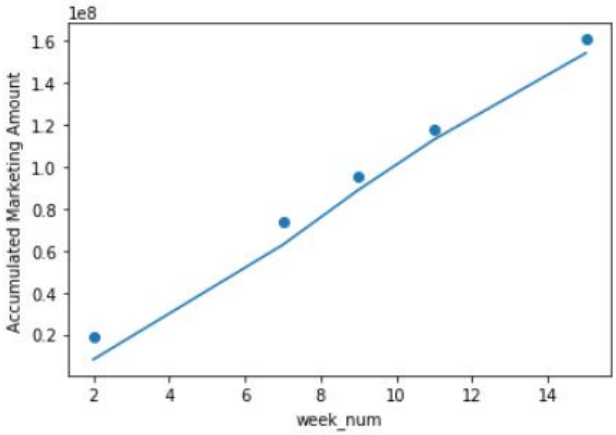
```
The MSE of RF is 1.604297e-02
```

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？

从 10 个决策树到 8 个决策树到 5 个决策树均方误差变大拟合效果变差。

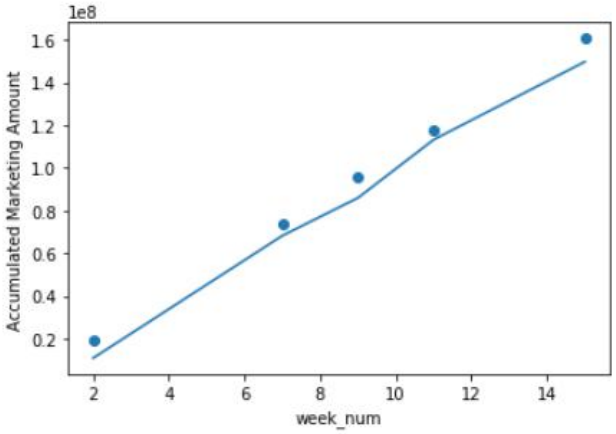
到 13 到 15 个决策树均方误差变小，拟合效果变好。

因此推测一定范围内，决策树数量越多，拟合效果相对越好。



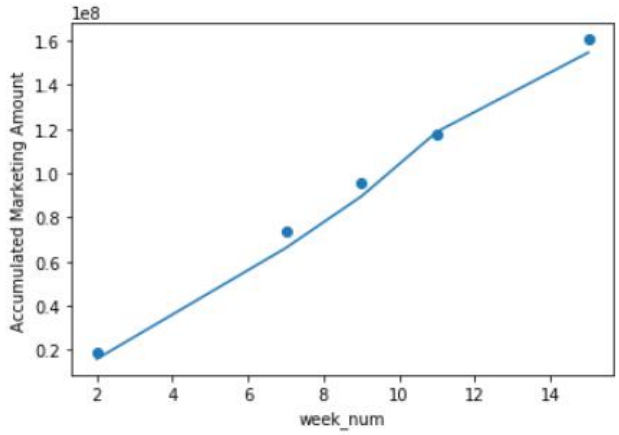

```
print('The MSE of RF is {:e}'.format(mean_squared_error(y_test/
```

```
The MSE of RF is 1.265476e-03
```



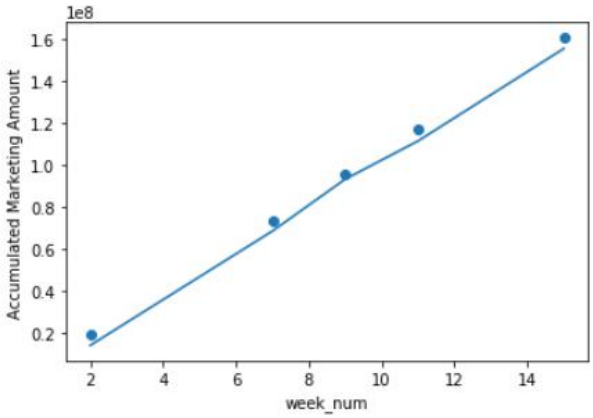

```
print('The MSE of RF is {:e}'.format(mean_squared_error(y_test/np.linalg.norm(y_test),
```

```
The MSE of RF is 1.185042e-03
```




```
print('The MSE of RF is {:e}'.format(mean_squared_error(y
```

```
The MSE of RF is 5.126925e-04
```



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：

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？

BaggingRegressor 回归算法：一个 Bagging 的回归器组合，用于集成多个回归器，用于回归预测的情况，集成解决过拟合的问题。

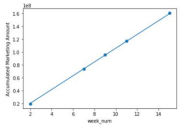
Python 使用 sklearn 实现的各种回归算法包括：

基本回归：线性、决策树、SVM、KNN

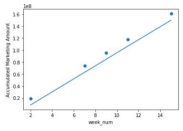
集成方法：随机森林、Adaboost、GradientBoosting、Bagging、ExtraTrees

回归算法比较

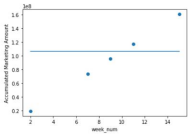
线性



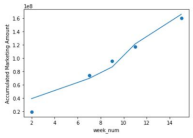
决策树



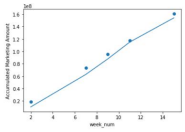
SVM



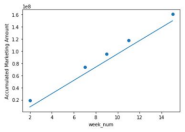
KNN



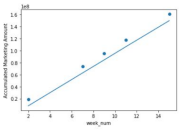
随机森林



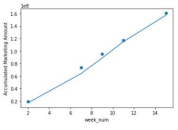
Adaboost



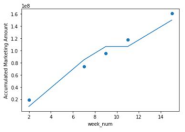
GradientBoosting



Bagging



ExtraTrees

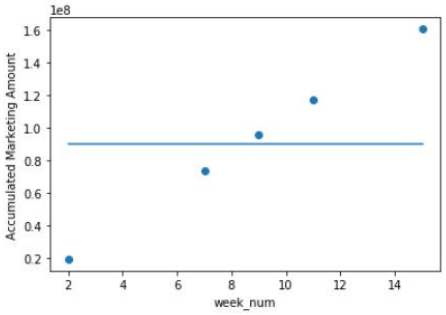


```
from sklearn.svm import SVR
from sklearn.ensemble import BaggingRegressor
from sklearn.datasets import make_regression

regr = BaggingRegressor(base_estimator=SVR(),
                        n_estimators=10, random_state=0).fit(x_train,y_train)
regr.predict([[0, 0, 0, 0]])
```

```
array([90029482.80284895])
```

```
y_pred3 = regr.predict(x_test)
plt.figure(5)
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pred3)
plt.xlabel('week_num')
plt.ylabel('Accumulated Marketing Amount')
plt.show()
```




```
format(mean_squared_error(y_test/np.linalg.norm(y_test), y_pred3/np.linalg.norm(y_test)))
```

```
The MSE of BR is 4.053565e-02
```

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。

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?

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

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：

可以考虑用 pytorch 替代 tensorflow 框架，并比较二者的优劣程度。