线上教育课程数据分析

目录

1 数据预处理

1.1 对照附录 1，理解各字段的含义，进行缺失值、重复值等方面的必要处理，并在报告中描述处理过程.

设置 Matplotlib 使用微软雅黑（Microsoft Yahei）字体来显示中文；禁用 Unicode 减号，以确保负号正常显示。考虑到数据的体量较大，因此使用 python 中的 pandas 对数据进行初步的描述性统计分析。使用 pandas 分别读取 CSV 文件，并指定编码格式为 GBK。使用 head()方法显示前五行数据，以便快速查看文件内容。

选择 users 数据框中的特定列(user_id, register_time, recently_logged, learn_time)，并创建一个新的数据框 users1。使用.copy()方法显式创建副本，以避免潜在的链式索引问题。

```
plt.rcParams['font.sans-serif'] = ['Microsoft Yahei']
plt.rcParams['axes.unicode_minus'] = False
Login = pd.read_csv(r'C:\Users\DELL\Desktop\数字图书馆关键技术期末作业-20250108\线上教育课程数据分析\Login.csv', encoding='gbk')
Login.head()
print(Login.head())
```

	user_id	login_time	login_place	
0	用户 3	2018-09-06	09:32:47	中国广东广州
1	用户 3	2018-09-07	09:28:28	中国广东广州
2	用户 3	2018-09-07	09:57:44	中国广东广州
3	用户 3	2018-09-07	10:55:07	中国广东广州
4	用户 3	2018-09-07	12:28:42	中国广东广州

```
study_information = pd.read_csv(r'C:\Users\DELL\Desktop\数字图书馆关键技术期末作业-20250108\线上教育课程数据分析\study_information.csv', encoding='gbk')
study_information.head()
print(study_information.head())
```

user_id	course_id	course_join_time	learn_process	price			
0	用户 3	课程 106	2020-04-21	10:11:50	width:	0%;	0.0
1	用户 3	课程 136	2020-03-05	11:44:36	width:	1%;	0.0
2	用户 3	课程 205	2018-09-10	18:17:01	width:	63%;	0.0
3	用户 4	课程 26	2020-03-31	10:52:51	width:	0%;	319.0
4	用户 4	课程 34	2020-03-31	10:52:49	width:	0%;	299.0

```
users = pd.read_csv(r'C:\Users\DELL\Desktop\数字图书馆关键技术期末作业-20250108\线上教育课程数据分析\users.csv', encoding='gbk')
```

```
users1 = users[['user_id', 'register_time', 'recently_logged', 'learn_time']].copy()  # 使用 .copy() 显式创建副本
users1.head()
print(users1.head())
```

	user_id	register_time	recently_logged	learn_time		
0	用户44251	2020/6/18	9:49	2020/6/18	9:49	41.25
1	用户44250	2020/6/18	9:47	2020/6/18	9:48	0
2	用户44249	2020/6/18	9:43	2020/6/18	9:43	16.22
3	用户44248	2020/6/18	9:09	2020/6/18	9:09	0
4	用户44247	2020/6/18	7:41	2020/6/18	8:15	1.8

1.2 对用户信息表中 recently_logged 字段的“--”值进行必要的处理，并在报告中描述处理过程。

缺失值处理:

为了解 Login 数据集中各列的缺失情况，进而为后续处理提供依据，使用 isnull 计算数据集中每一列的缺失值比例（即缺失值的数量除以总行数）。

```
Login.isnull().mean()
```

```
user_id          0.0
login_time       0.0
login_place      0.0
dtype: float64
```

```
study_information.isnull().mean()
```

```
user_id                0.000000
course_id              0.000000
course_join_time       0.000000
learn_process          0.000000
price                  0.021736
dtype: float64
```

因为价格信息对于分析课程收费情况至关重要，为了确保 price 列没有缺失值，将缺失值删除，将内容为 0 的记为免费课程。

```
# 删除 price 的缺失，认 0 为免费课程
study_information = study_information.dropna(subset=['price'])
study_information.head()
```

	user_id	course_id	course_join_time	learn_process	price		
0	用户 3	课程 106	2020-04-21	10:11:50	width:	0%;	0.0
1	用户 3	课程 136	2020-03-05	11:44:36	width:	1%;	0.0
2	用户 3	课程 205	2018-09-10	18:17:01	width:	63%;	0.0
3	用户 4	课程 26	2020-03-31	10:52:51	width:	0%;	319.0
4	用户 4	课程 34	2020-03-31	10:52:49	width:	0%;	299.0

```
users1.isnull().mean()
users1 = users1.dropna(subset=['user_id'])  # 修改为 users1 而不是 users
users1.head()
```

	user_id	register_time	recently_logged	learn_time		
0	用户 44251	2020/6/18	9:49	2020/6/18	9:49	41.25
1	用户 44250	2020/6/18	9:47	2020/6/18	9:48	0
2	用户 44249	2020/6/18	9:43	2020/6/18	9:43	16.22
3	用户 44248	2020/6/18	9:09	2020/6/18	9:09	0
4	用户 44247	2020/6/18	7:41	2020/6/18	8:15	1.8

异常值处理:

将 users1 数据框保存为名为 new_users1.csv 的 CSV 文件，不包含行索引（index=False），重新读取该文件，并从中提取特定列生成新的数据框 new_users，保证数据的完整性和可操作性。从 new_users1 中选择特定的四列（user_id, register_time, recently_logged, learn_time），创建新的数据框 new_users，只选择 new_users1 中存在的列。

new_users 数据框包含用户 ID、注册时间、最近登录时间和学习时间这四个关键字段，方便后续的数据分析和处理。

```
# excel 表格处理
users1.to_csv('new_users1.csv', index=False)
new_users1 = pd.read_csv('new_users1.csv')
new_users1.columns
new_users = new_users1[['user_id', 'register_time', 'recently_logged',
'learn_time']]  # 只选择存在的列
new_users.head()
```

	user_id	register_time	recently_logged	learn_time		
0	用户 44251	2020/6/18	9:49	2020/6/18	9:49	41.25

1	用户44250	2020/6/18	9:47	2020/6/18	9:48	0
2	用户44249	2020/6/18	9:43	2020/6/18	9:43	16.22
3	用户44248	2020/6/18	9:09	2020/6/18	9:09	0
4	用户44247	2020/6/18	7:41	2020/6/18	8:15	1.8

检查 Login、study_information 和 new_users 数据表中是否存在重复值，使用 duplicated()方法和 any()函数，并将结果打印出来。

```
print("Login 表存在重复值：", any(Login.duplicated()))
print("study_information 表存在重复值：
", any(study_information.duplicated()))
print("users 表存在重复值：", any(new_users.duplicated()))
```

```
Login 表存在重复值：  False
study_information 表存在重复值：  False
users 表存在重复值：  True
```

可以看出 users 表存在重复值，下面对其进行处理。

因为 new_users 是对 users 进行的复制，所以直接对 new_users 进行修改。使用 drop_duplicates()方法删除 new_users 表中的所有重复行，参数 inplace=True 表示直接在原数据框 new_users 上进行修改，而不是返回一个新的数据框。然后使用 head()方法显示 new_users 数据框处理后的前五行数据，以便快速验证重复值是否已被删除。

```
new_users.drop_duplicates(inplace=True)
new_users.head()
```

	user_id	register_time	recently_logged	learn_time		
0	用户44251	2020/6/18	9:49	2020/6/18	9:49	41.25
1	用户44250	2020/6/18	9:47	2020/6/18	9:48	0
2	用户44249	2020/6/18	9:43	2020/6/18	9:43	16.22
3	用户44248	2020/6/18	9:09	2020/6/18	9:09	0
4	用户44247	2020/6/18	7:41	2020/6/18	8:15	1.8

2. 平台用户活跃度分析

2.1 分别绘制各省份与各城市平台登录次数热力地图，并分析用户分布情况。

提取 login_place 列的前两个字符生成一个新的列 guobie，表示用户的国别信息。然后，显示处理后的前五行数据，以确保新列的正确性。

```
# 国内热力图
Login['guobie'] = Login['login_place'].apply(lambda x: x[0:2]).tolist()
Login.head(5)
```

	user_id	login_time	login_place	guobie
0	用户 3	2018-09-06	09:32:47	中国广东广州 中国
1	用户 3	2018-09-07	09:28:28	中国广东广州 中国
2	用户 3	2018-09-07	09:57:44	中国广东广州 中国
3	用户 3	2018-09-07	10:55:07	中国广东广州 中国
4	用户 3	2018-09-07	12:28:42	中国广东广州 中国

因为只有内蒙古省、黑龙江省的名字简写是三个字，其他省份、直辖市、自治区的名字简写都是两个字，所以首先对内蒙古与黑龙江进行处理。

从 Login_nei 数据框的 login_place 列中提取特定部分的内容，并将其分别存储到新的列 shengfen 和 chengshi 中。login_place 列中的值是类似于 "XX 省 XXX 市" 的格式，将 "省" 后面的三个字符分配给 shengfen 列，将 "市" 及其后面的部分分配给 chengshi 列。

```
#处理掉省份名为三个字的省，即内蒙古、黑龙江
# 使用 .copy() 显式创建副本，内蒙古
Login_nei = Login.loc[Login['login_place'].str.contains("内蒙古
")].copy()
Login_nei['shengfen'] = Login_nei['login_place'].apply(lambda x: x[2:5]
)
Login_nei['chengshi'] = Login_nei['login_place'].apply(lambda x: x[5:])
Login_nei.head()
```

	user_id	login_time	login_place	guobie	shengfen	chengshi	
15646	用户 1234	2020-02-03	21:38:07	中国内蒙古呼和浩特	中国	内蒙古	呼和浩特
17145	用户	2019-01-	14:19:26	中国内	中国	内蒙古	呼和浩

	1491	04		蒙古呼和浩特			特
17283	用户1544	2019-10-11	13:56:47	中国内蒙古呼和浩特	中国	内蒙古	呼和浩特
17698	用户1715	2019-01-10	09:37:46	中国内蒙古鄂尔多斯	中国	内蒙古	鄂尔多斯
17809	用户1765	2019-01-11	15:03:12	中国内蒙古兴安盟	中国	内蒙古	兴安盟

```
#处理黑龙江
Login_hei = Login.loc[Login['login_place'].str.contains("黑龙江
")].copy()
Login_hei['shengfen'] = Login_hei['login_place'].apply(lambda x: x[2:5]
)
Login_hei['chengshi'] = Login_hei['login_place'].apply(lambda x: x[5:])
Login_hei.head()
```

	user_id	login_time	login_place	guobie	shengfen	chengshi	
5682	用户186	2018-10-25	19:40:12	中国黑龙江	中国	黑龙江	
5683	用户186	2018-10-27	11:30:15	中国黑龙江	中国	黑龙江	
5684	用户186	2018-10-28	12:23:24	中国黑龙江	中国	黑龙江	
5821	用户196	2018-10-25	19:57:47	中国黑龙江	中国	黑龙江	
6196	用户229	2018-10-25	21:28:13	中国黑龙江哈尔滨	中国	黑龙江	哈尔滨

使用同样的方法，处理剩余省份，处理完成后查看效果。

```
#处理其他名字为两个字的省份
Login2 = Login.loc[~Login['login_place'].str.contains("内蒙古|黑龙江
")].copy()
Login2['shengfen'] = Login2['login_place'].apply(lambda x: x[2:4])
Login2['chengshi'] = Login2['login_place'].apply(lambda x: x[4:])
Login2.head(5)
```

	user_id	login_time	login_place	guobie	shengfen	chengshi	
0	用户3	2018-09-06	09:32:47	中国广东广州	中国	广东	广州
1	用户3	2018-09-	09:28:28	中国广	中国	广东	广州

		07		东广州			
2	用户 3	2018-09-07	09:57:44	中 国 广东广州	中国	广东	广州
3	用户 3	2018-09-07	10:55:07	中 国 广东广州	中国	广东	广州
4	用户 3	2018-09-07	12:28:42	中 国 广东广州	中国	广东	广州

将三组数据两两合并，得到所有处理完成的数据。

```
#将所有数据进行合并，得到完整数据
Login_he1 = pd.concat([Login_nei, Login2], ignore_index=True)
Login_he2 = pd.concat([Login_he1, Login_hei], ignore_index=True)
Login_he = Login_he2[['user_id', 'login_time', 'guobie', 'shengfen', 'c
hengshi']]
Login_he.head()
```

	user_id	login_time	guobie	shengfen	chengshi	
0	用户 1234	2020-02-03	21:38:07	中国	内蒙古	呼和浩特
1	用户 1491	2019-01-04	14:19:26	中国	内蒙古	呼和浩特
2	用户 1544	2019-10-11	13:56:47	中国	内蒙古	呼和浩特
3	用户 1715	2019-01-10	09:37:46	中国	内蒙古	鄂尔多斯
4	用户 1765	2019-01-11	15:03:12	中国	内蒙古	兴安盟

按照国别进行分组，划分出除中国之外各个国家的登录次数，放入Login_he3。

```
# 国外
# 按照国别进行分组，并统计每个国家的登录次数
Login_he2 = Login_he.groupby(by='guobie', as_index=False).count()
# 不包括中国的数据
Login_he3 = Login_he2[~Login_he2['guobie'].isin(["中国"])].copy()  # 使
用 .copy() 显式创建副本
Login_he3
```

	guobie	user_id	login_time	shengfen	chengshi
1	南非	3	3	3	3
2	希腊	1	1	1	1
3	德国	24	24	24	24
4	挪威	1	1	1	1
5	捷克	4	4	4	4
6	波兰	7	7	7	7

7	泰国	2	2	2	2
8	瑞典	1	1	1	1
9	瑞士	1	1	1	1
10	英国	151	151	151	151
11	荷兰	8	8	8	8
12	越南	27	27	27	27

从 Login_he3 数据框中提取 guobie 列和 user_id 列的数据，分别作为 x 轴和 y 轴的数据，绘制国外平台登录情况的折线图。

```
x_data = list(Login_he3['guobie'])
y_data = Login_he3['user_id']
```

从 Login_he 数据框中筛选出所有 guobie 列为 "中国" 的记录，然后按省份（shengfen）统计登录次数，并生成一个适合绘制热力图的数据框。

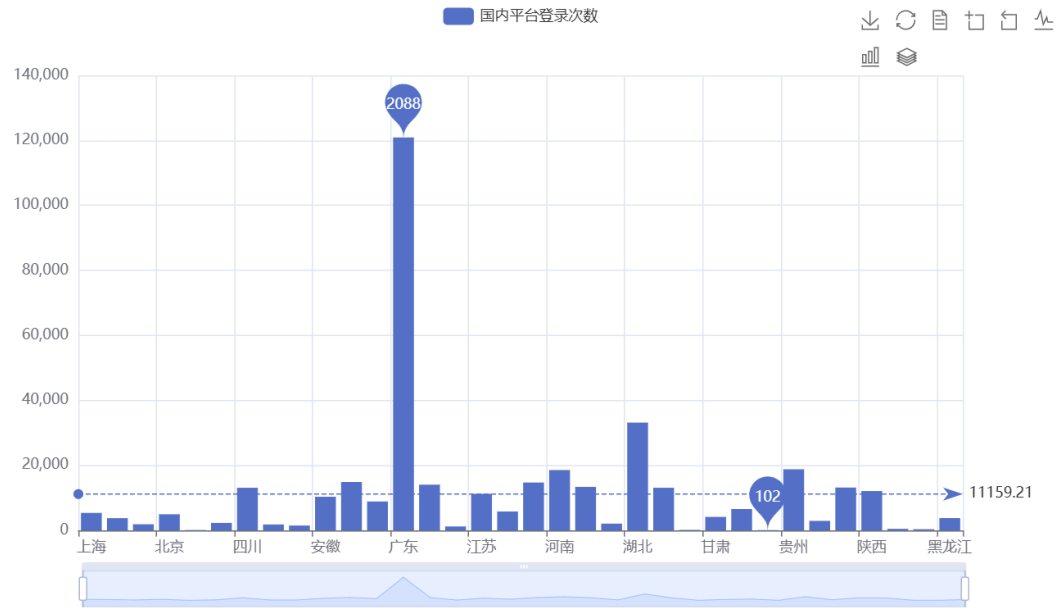
```
# 各省份登陆热力图
# 不包括国外
Login_china = Login_he[Login_he['guobie'].isin(["中国"])].copy()  # 使用 .copy() 显式创建副本
Login_china2 = Login_china.groupby(by='shengfen', as_index=False).count()
Login_china3 = Login_china2.iloc[1:,].copy()  # 使用 .copy() 显式创建副本
Login_china3['count'] = Login_china3['user_id']  # 使用 .loc 进行显式赋值
Login_china3.head()
print(Login_china3)
```

	shengfen	user_id	login_time	guobie	chengshi	count
1	上海	5365	5365	5365	5365	5365
2	云南	3750	3750	3750	3750	3750
3	内蒙古	1870	1870	1870	1870	1870
4	北京	4946	4946	4946	4946	4946
5	台湾	126	126	126	126	126
6	吉林	2285	2285	2285	2285	2285
7	四川	13099	13099	13099	13099	13099
8	天津	1805	1805	1805	1805	1805
9	宁夏	1481	1481	1481	1481	1481
10	安徽	10332	10332	10332	10332	10332
11	山东	14874	14874	14874	14874	14874
12	山西	8875	8875	8875	8875	8875
13	广东	120887	120887	120887	120887	120887
14	广西	14052	14052	14052	14052	14052
15	新疆	1201	1201	1201	1201	1201
16	江苏	11237	11237	11237	11237	11237
17	江西	5796	5796	5796	5796	5796
18	河北	14708	14708	14708	14708	14708
19	河南	18550	18550	18550	18550	18550

20	浙江	13366	13366	13366	13366	13366
21	海南	2062	2062	2062	2062	2062
22	湖北	33149	33149	33149	33149	33149
23	湖南	13103	13103	13103	13103	13103
24	澳门	171	171	171	171	171
25	甘肃	4138	4138	4138	4138	4138
26	福建	6558	6558	6558	6558	6558
27	西藏	102	102	102	102	102
28	贵州	18786	18786	18786	18786	18786
29	辽宁	2917	2917	2917	2917	2917
30	重庆	13163	13163	13163	13163	13163
31	陕西	12088	12088	12088	12088	12088
32	青海	455	455	455	455	455
33	香港	341	341	341	341	341
34	黑龙江	3775	3775	3775	3775	3775

从 Login_china3 数据框中提取 shengfen 列和 count 列的数据，分别作为 x 轴和 y 轴的数据，使用 pyecharts 库创建一个柱状图（Bar），展示全国各省份的登录用户人数分布，并在图表中标记最大值、最小值和平均值。在 Jupyter Notebook 中渲染图表，最终将图表保存为 HTML 文件，文件名为 map2.html。保存为 html 格式的图表还包含工具箱和横轴缩放功能，方便用户交互和查看数据。

map2.html




```
# 全国省份登陆用户人数分布
x_data = list(Login_china3['shengfen'])
y_data = list(Login_china3['count'])
```

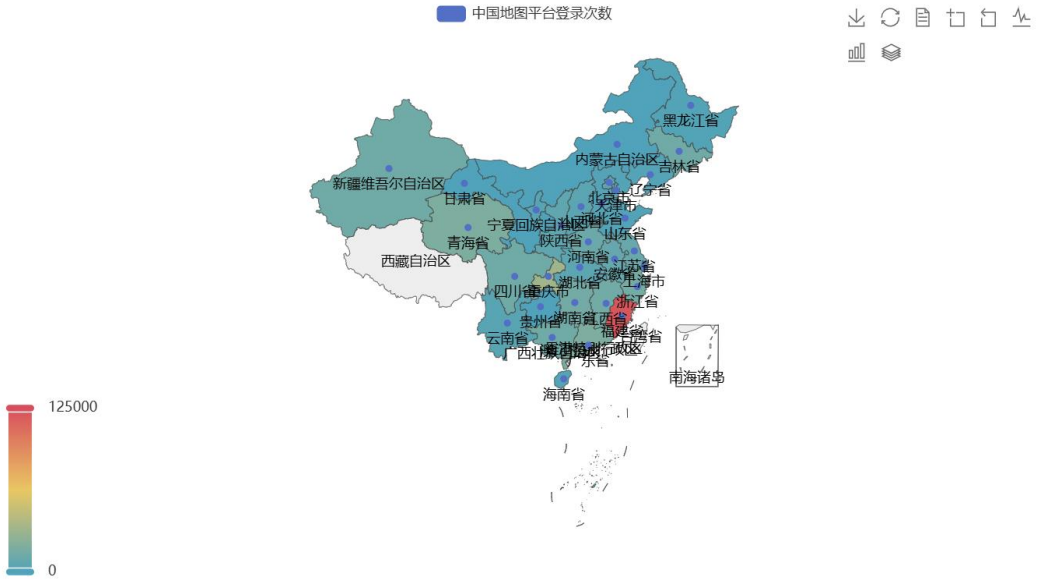
```
# 特殊值标记
bar = (Bar()
       .add_xaxis(x_data)
       .add_yaxis('国内平台登录次数', y_data)
       .set_series_opts(
        markpoint_opts=opts.MarkPointOpts(
            data=[
                opts.MarkPointItem(type_="max", name="最大值"),
                opts.MarkPointItem(type_="min", name="最小值"),
            ]))
       .set_global_opts(
            # title_opts=opts.TitleOpts(title="国内平台登录次数
", subtitle="副标题"),                          # 标题
            toolbox_opts=opts.ToolboxOpts(),                # 工具箱
            datazoom_opts=opts.DataZoomOpts(range_start=0, range_end=10
0)     # 横轴缩放
        )
       .set_series_opts(
        # 为了不影响标记点，这里把标签关掉
        label_opts=opts.LabelOpts(is_show=False),
        markline_opts=opts.MarkLineOpts(
            data=[
                opts.MarkLineItem(type_="average", name="平均值")
            ]))
      )
bar.render_notebook()
bar.render('map2.html')  # 保存到本地
```

为了创建全国各省份的热力图，首先定义一个包含所有省份名称的列表 province，从 Login_china3 数据框中提取 count 列的值，并转换为列表，表示每个省份的登录次数。使用 Map() 创建一个地图对象，并设置系列名称为 "中国地图平台登录次数"，指定地图类型为 'china'（中国地图）。热力图展示了全国各省份的登录用户人数分布，每个省份的颜色深浅表示该省份的登录次数多少，颜色越深表示登录次数越多。

```
province = ['北京市', '天津市', '河北省', '山西省', '内蒙古自治区', '辽宁省
', '吉林省', '黑龙江省', '上海市', '江苏省', '浙江省', '安徽省', '福建省
', '江西省',
            '山东省', '河南省', '湖北省', '湖南省', '广东省', '广西壮族自治
区', '海南省', '重庆市', '四川省', '贵州省', '云南省', '陕西省', '甘肃省
', '青海省', '宁夏回族自治区', '新疆维吾尔自治区']
values = list(Login_china3['count'])
data = [[province[i], values[i]] for i in range(len(province))]
map = (
    Map()
```

```
    .add("中国地图平台登录次数", data, 'china')
)
map.set_global_opts(
    visualmap_opts=opts.VisualMapOpts(max_=125000),  # 最大数据范围
    toolbox_opts=opts.ToolboxOpts()  # 工具箱
)
map.render_notebook()
map.render("cmap.html")  # 全国热力图
```

cmap.html



为了生成全国各省级行政区的热力图，定义函数 plot_city_login_map。首先，将 chengshi 列转换为字符串类型，并在每个城市名称后添加 "市" 字样，生成新的列 new_chengshi。其次，定义一个字典 city_name_mapping，用于修正某些城市名称的错误拼写，使用 replace 方法将 new_chengshi 列中的城市名称进行替换。再次，从 df 数据框中提取 new_chengshi 列的值并转换为列表用于表示城市，提取 user_id 列的值并转换为列表用于表示每个城市的登录次数。同时用 if 检查是否为空，如果将城市名称和对应的登录次数组合而成的二维列表 data 为空，则打印警告信息，跳过绘图并返回。最后，创建并配置地图对象，渲染图表并在 Jupyter Notebook 中显示，同时保存为 HTML 文件。

```
# 所有省份
def plot_city_login_map(df, province, max_value, filename):
    # 确保 chengshi 列为字符串类型
    df["new_chengshi"] = df['chengshi'].astype(str) + '市'
    city_name_mapping = {
        '恩施土家族苗族自治州市': '恩施土家族苗族自治州',
        '神农架林区市': '神农架林区',
        '黔东南苗族侗族自治州市': '黔东南苗族侗族自治州',
```

```
        '黔南布依族苗族自治州市': '黔南布依族苗族自治州',
        '黔西南布依族苗族自治州市': '黔西南布依族苗族自治州'
    }
    df['new_chengshi'] = df['new_chengshi'].replace(city_name_mapping)
    x_data = list(df['new_chengshi'])
    y_data = list(df['user_id'])
    data = [[x_data[i], y_data[i]] for i in range(len(y_data))]
    if not data:
        print(f"警告: {province} 的数据为空，跳过绘制热力图。")
        return
    map = (
        Map()
        .add(f"{province}各城市平台登录次数热力地图", data, province)
        .set_global_opts(
            visualmap_opts=opts.VisualMapOpts(
                max_=max_value,
                is_piecewise=True,  # 启用分段模式
                pieces=[
                    {"min": 100000, "color": "#c82423"} , # 蓝色
                    {"min": 50000, "max": 99999, "color": "#fa8878"},  # 青色
                    {"min": 10000, "max": 49999, "color":"#ffbe7a"},  # 绿色
                    {"min": 1000, "max": 9999, "color": "#3480b8"},  # 黄色
                    {"min": 100, "max": 999, "color":"#add3e2"},  # 橙色
                    {"min": 0, "max": 99, "color": "#8dcec88"},  # 红色
                ]
            ),
            toolbox_opts=opts.ToolboxOpts()
        )
    )
    map.render_notebook()
    map.render(f"{filename}.html")
    print(f"{filename}.html 文件已保存。")
```

```
# 定义省份列表
provinces = [
    '内蒙古','广东','北京','香港','广西','山东','湖南','湖北','浙江','福建','江苏','贵州','陕西','重庆',
    '四川','河北','甘肃','安徽','海南','河南','辽宁','江西','天津','新疆','上海','山西','云南','台湾',
    '吉林','宁夏','青海','澳门','西藏','黑龙江'
    # 可以继续添加其他省份
```

```
]
# 计算每个省份的最大登录次数
max_values = {}
for province in provinces:
    Login_prov1 = Login_he[Login_he['shengfen'].isin([province])].copy(
)
    Login_prov2 = Login_prov1.groupby(by='chengshi', as_index=False).co
unt()
    Login_prov = Login_prov2.iloc[1:,].copy()
    if not Login_prov.empty:
        max_value = Login_prov['user_id'].max()
        max_values[province] = max_value
    else:
        max_values[province] = 0  # 如果数据为空，设置为0或其他默认值
```

由于直辖市仅有 shengfen 数据，没有 chengshi 数据（如：中国北京→shengfen：北京，chengshi：；中国山东济南→shengfen：山东，chengshi：济南），需要分别处理直辖市与其他省份。

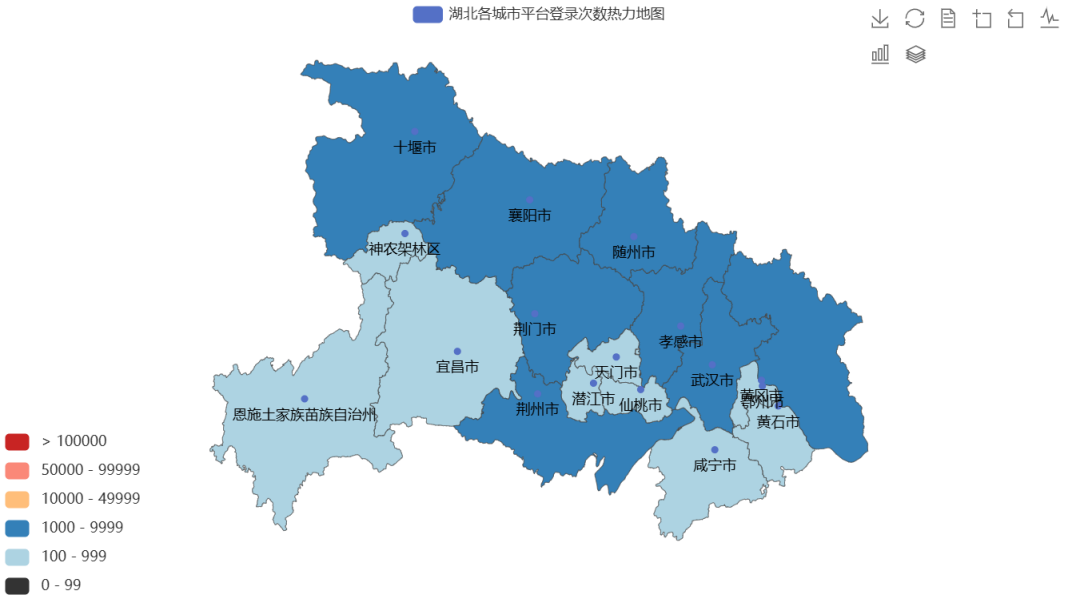
首先，填充 chengshi 列中为空的值为 "未知"。其次，遍历每个省份，筛选数据并按城市分组，计算每个城市的登录次数，找到每个省份的最大登录次数。通过定义一个包含多个省份名称的列表，使用 for 遍历每个省份，筛选出该省份的数据，然后按照城市分组并计算登录次数，进而找到最大登录次数。最后，调用先前定义的函数，根据计算的最大登录次数，为每个省份生成并保存热力图。

```
# 数据预处理：填充 chengshi 为空的值
Login_he['chengshi'] = Login_he['chengshi'].fillna('未知')
# 按省份分类处理
max_values = {}
provinces = Login_he['shengfen'].unique()
for province in provinces:
    # 筛选出当前省份的数据
    Login_prov1 = Login_he[Login_he['shengfen'] == province].copy()
    if Login_prov1.empty:
        print(f"警告：{province} 的数据为空，跳过绘制热力图。")
        max_values[province] = 0
        continue
    # 按城市分组并计算登录次数
    Login_prov2 = Login_prov1.groupby('chengshi', as_index=False).count
()
    if Login_prov2.empty:
        print(f"警告：{province} 的 groupby 数据为空，跳过绘制热力图。")
        max_values[province] = 0
        continue
    # 找到最大登录次数
    max_value = Login_prov2['user_id'].max()
```

```
    max_values[province] = max_value
# 循环遍历省份列表并绘制热力图
for province, max_value in max_values.items():
    Login_prov1 = Login_he[Login_he['shengfen'] == province].copy()
    if Login_prov1.empty:
        print(f"警告: {province} 的数据为空，跳过绘制热力图。")
        continue
    Login_prov2 = Login_prov1.groupby('chengshi', as_index=False).count
()
    if Login_prov2.empty:
        print(f"警告: {province} 的 groupby 数据为空，跳过绘制热力图。")
        continue
    # 调用 plot_city_login_map 函数
    plot_city_login_map(Login_prov2, province, max_value, f'{province}_
map')
```

此处以湖北为例，剩余图表附在文件里

湖北_map.html



使用数据透视表绘制不同国家用户的分布情况。

准备用于制作数据透视表的数据。首先对 users 数据框中的注册时间进行处理，使用 pd.to_datetime 将 users 数据框中的 register_time 列转换为日期时间格式。其次使用 apply 方法对 Login 数据框中的 login_place 列进行操作，提取前两个字符作为国家信息，并转换为列表。最后，注意到两个数据框都有 user_id 列，使用此列对 users 数据框和 Login 数据框进行合并。

```
# 统计不同国家用户的分布情况
users['register_time'] = pd.to_datetime(users['register_time'], errors=
'coerce')
```

```
users['year_month'] = users['register_time'].dt.to_period('M').astype(s
tr)
# 从 Login 数据框中提取国家信息
Login['guobie'] = Login['login_place'].apply(lambda x: x[0:2]).tolist()
# users 数据框中有一个 user_id 列，用于与 Login 数据框合并
# 合并 users 和 Login 数据框，根据 user_id 进行合并
users = pd.merge(users, Login[['user_id', 'guobie']].drop_duplicates(),
 on='user_id', how='left')
```

	user_id	register_time	recently_logged	...	school	year_month	guobie		
0	用户44251	2020-06-18	09:49:00	2020/6/18	9:49	...	NaN	2020-06	NaN
1	用户44250	2020-06-18	09:47:00	2020/6/18	9:48	...	NaN	2020-06	中国
2	用户44249	2020-06-18	09:43:00	2020/6/18	9:43	...	NaN	2020-06	中国
3	用户44248	2020-06-18	09:09:00	2020/6/18	9:09	...	NaN	2020-06	中国
4	用户44247	2020-06-18	07:41:00	2020/6/18	8:15	...	NaN	2020-06	中国

生成数据透视表，并使用 matplotlib 库绘制一个表格

```
# 使用数据透视表统计不同国家用户的分布情况
country_user_count = users.pivot_table(index='year_month', columns='guo
bie', values='user_id', aggfunc='count', fill_value=0)
# 设置表格样式
fig, ax = plt.subplots(figsize=(12, 8))
ax.axis('tight')
ax.axis('off')
# 绘制表格
table = ax.table(
    cellText=country_user_count.values,
    colLabels=country_user_count.columns,
    rowLabels=country_user_count.index,
    cellLoc='center',
    loc='center',
    bbox=[0, 0, 1, 1],  # 调整表格的边界框
    edges='open'  # 去掉表格边框
```

```
)
table.auto_set_font_size(False)
table.set_fontsize(12)
table.scale(1.2, 1.2)  # 调整表格大小
# 设置标题
plt.title("不同国家用户的分布情况", fontsize=16, fontweight='bold')
plt.show()
```

不同国家用户的分布情况

	中国	南非	希腊	德国	挪威	捷克	波兰	泰国	瑞典	瑞士	英国	荷兰	越南
2018-09	27	0	0	0	0	1	0	0	0	0	0	0	0
2018-10	589	0	0	0	0	0	0	0	0	0	0	0	0
2018-11	451	0	0	0	0	0	0	0	0	0	1	0	0
2018-12	263	0	0	0	0	0	0	0	0	0	1	0	0
2019-01	1063	0	0	0	1	0	0	0	0	0	0	0	0
2019-02	747	0	0	1	0	0	0	0	0	0	0	0	0
2019-03	2736	0	0	2	0	0	1	0	0	0	3	3	0
2019-04	2814	0	0	0	0	0	2	1	0	0	3	0	0
2019-05	1209	0	0	1	0	0	0	0	0	0	1	0	0
2019-06	1192	1	0	0	0	0	0	0	0	0	1	0	0
2019-07	1070	0	0	1	0	0	0	1	0	0	1	0	0
2019-08	1074	0	0	0	0	0	0	0	0	0	2	0	0
2019-09	1743	0	0	0	0	0	0	0	1	0	0	0	0
2019-10	2036	0	0	1	0	0	0	0	0	0	1	0	0
2019-11	1230	0	0	2	0	0	0	0	0	1	2	1	0
2019-12	1112	0	1	0	0	0	0	0	0	0	1	1	0
2020-01	638	1	0	0	0	0	0	0	0	0	0	0	0
2020-02	7446	1	0	1	0	0	0	0	0	0	3	0	0
2020-03	5286	0	0	0	0	1	1	0	0	0	1	1	1
2020-04	3137	0	0	0	0	0	0	0	0	0	1	1	0
2020-05	1818	0	0	1	0	0	0	0	0	0	1	0	0
2020-06	630	0	0	0	0	0	0	0	0	0	0	0	0

用户分布情况：通过对国内各个省市地区的分布情况，可以看出各个省市中都存在某几个重点地区，比如广州、武汉。周口等地。该地区的存在使得省份的占比数据大幅提升。

由数据透视图可以得出，该线上教育平台海外分布相对较少，大部分业务集中在中国地区。其中，在海外部分地区，欧洲占比较高，尤其是英国和德国。

2.2 对每月的新增用户数进行统计分析，用折线图的形式体现月环比增长趋势，并分析这种趋势的变化原因。

为了统计每个月新增用户的数量，使用 groupby('year_month') 按 year_month 列对 users 数据框进行分组。使用 size() 方法计算每个分组中的记录数，即每个月新增用户的数量，使用 reset_index(name='new_user_count') 将分组后的结果重置为数据框，并将计数结果命名为 new_user_count。

```
# 统计每月新增用户数
monthly_new_users = users.groupby('year_month').size().reset_index(name
='new_user_count')
```

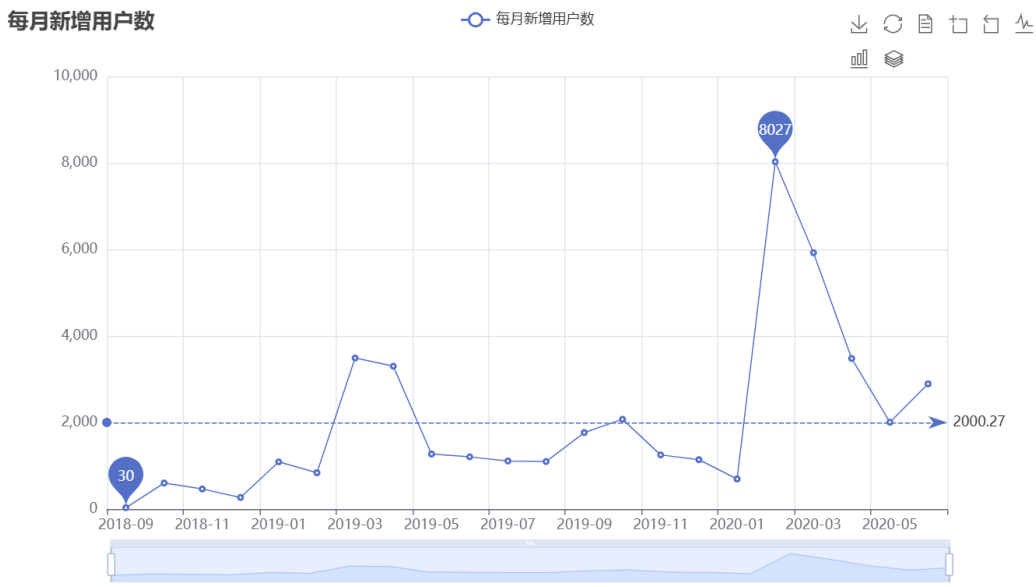
	year_month	new_user_count
0	2018-09	30
1	2018-10	599
2	2018-11	462
3	2018-12	266
4	2019-01	1089

绘制每月新增用户数折线图。

```
# 绘制每月新增用户数折线图
x_data = list(monthly_new_users['year_month'])
y_data = list(monthly_new_users['new_user_count'])
line = (
    Line()
    .add_xaxis(x_data)
    .add_yaxis('每月新增用户数', y_data)
    .set_series_opts(
        markpoint_opts=opts.MarkPointOpts(
            data=[
                opts.MarkPointItem(type_="max", name="最大值"),
                opts.MarkPointItem(type_="min", name="最小值"),
            ]))
    .set_global_opts(
        title_opts=opts.TitleOpts(title="每月新增用户数"),  # 标题
        toolbox_opts=opts.ToolboxOpts(),  # 工具箱
        datazoom_opts=opts.DataZoomOpts(range_start=0, range_end=100)  #
 横轴缩放
    )
    .set_series_opts(
        # 为了不影响标记点，这里把标签关掉
        label_opts=opts.LabelOpts(is_show=False),
        markline_opts=opts.MarkLineOpts(
            data=[
                opts.MarkLineItem(type_="average", name="平均值")
            ]))
)
line.render_notebook()
line.render('monthly_new_users.html')  # 每月新增用户数折线图
```

折线图

monthly_new_users.html



变化趋势：于 2019 年中和 2020 年初迎来了两拨高峰，推测 2019 年的高峰是软件起步后的扩张阶段，经过这波高峰后每月新用户稳定增长。2020 年的高峰是由于疫情发生，线上教育迎来一波集中的高潮，相应的后几个月新用户增长率虽有下降，但增幅保持稳定。

2.3 分别绘制工作日与非工作日各时段的用户登录次数柱状图，并分析用户活跃的主要时间段。

首先处理工作日数据。第一步，分割登录时间，将 login_time 列分割成日期和时间两部分，并提取 user_id 列。第二步，定义工作日时间范围，设置工作日的起始和结束时间。第三步，获取工作日日期，使用 chinese_calendar 库获取指定时间范围内的所有工作日日期。第四步，转换为日期列表，将工作日日期转换为字符串格式。第五步，筛选工作日数据，从分割后的数据中筛选出工作日的数据。第六步，将小时取整，提取并处理时间部分，只保留小时部分。

```
Login1 = Login["login_time"].str.split(" ", expand=True).fillna("")
Login1['login_data'] = Login1[0].copy()
Login1['login_hour'] = Login1[1].copy()
Login1['user_id'] = Login['user_id']
Login2 = Login1[['user_id', 'login_data', 'login_hour']].copy()  # 使用 .copy() 显式创建副本
Login2.head()
# 获取工作日时期
start_time = datetime.datetime(2018, 9, 6)
end_time = datetime.datetime(2020, 6, 18)
# 获取工作日日期
```

```
workdays = chinese_calendar.get_workdays(start_time, end_time)
# 转换为日期列表
date_string = [d.strftime('%Y-%m-%d') for d in workdays]
# 筛选工作日所含数据
Login3 = Login2[Login2['login_data'].isin(date_string)].copy()  # 使用 .copy() 显式创建副本
# 将小时取整
Login3['login_newhour'] = Login3['login_hour'].apply(lambda x: int(x[0:2]))
Login3.head()
```

	user_id	login_data	login_hour	login_newhour
0	用户 3	2018-09-06	09:32:47	9
1	用户 3	2018-09-07	09:28:28	9
2	用户 3	2018-09-07	09:57:44	9

按小时分组并计数，统计每个小时的用户登录次数，最终形成工作日登录次数。

```
gongzuori = Login3.groupby(by=Login3['login_newhour'],as_index=False)['user_id'].count()
gongzuori['gongzuori'] = gongzuori['user_id']
gongzuori = gongzuori[['login_newhour','gongzuori']]
gongzuori.head()
```

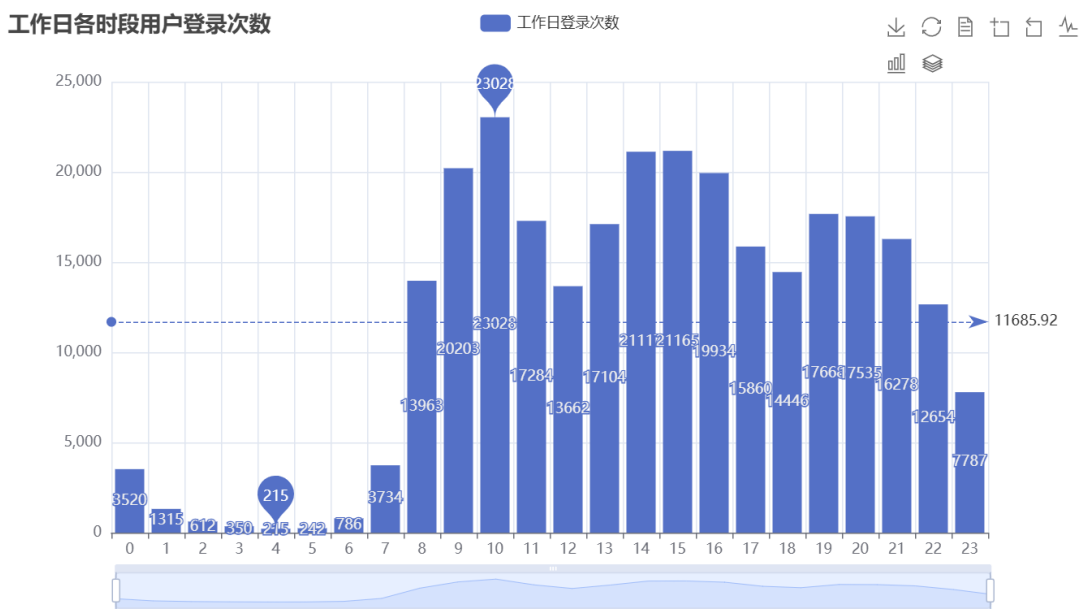
	login_newhour	gongzuori
0	0	3520
1	1	1315
2	2	612
3	3	350
4	4	215

制作图表

```
# 准备数据
attr = list(gongzuori['login_newhour'])
v1 = list(gongzuori['gongzuori'])
# 创建柱状图对象
bar = (
    Bar()
    .add_xaxis(attr)
    .add_yaxis('工作日登录次数', v1)
    .set_global_opts(
        title_opts=opts.TitleOpts(title="工作日各时段用户登录次数"),  # 标题
        toolbox_opts=opts.ToolboxOpts(),  # 工具箱
```

```
        datazoom_opts=opts.DataZoomOpts(range_start=0, range_end=100)  # 横轴缩放
    )
    .set_series_opts(
        label_opts=opts.LabelOpts(is_show=True),  # 显示数据标签
        markpoint_opts=opts.MarkPointOpts(
            data=[
                opts.MarkPointItem(type_="max", name="最大值"),
                opts.MarkPointItem(type_="min", name="最小值"),
            ]
        ),
        markline_opts=opts.MarkLineOpts(
            data=[
                opts.MarkLineItem(type_="average", name="平均值")
            ]
        )
    )
)
# 渲染图表
bar.render_notebook()
bar.render("workday_login_analysis.html")  # 保存到本地
```


workday_login_analysis.html



分析趋势：工作日活跃时间段集中在上午 9-10 点，下午 14-16 点，晚间 19;22 点。

同理，制作非工作日列表。峰值出现在上午 10 点。

```
# 获取非工作日日期
```

```
a = chinese_calendar.get_holidays(start_time, end_time)
# 转换为日期列表
date_string = [d.strftime('%Y-%m-%d') for d in a]
# 筛选非工作日所含数据
Login4 = Login2[Login2['login_data'].isin(date_string)].copy()  # 使用 .copy() 显式创建副本
# 将小时取整
Login4.loc[:, 'login_newhour'] = Login4['login_hour'].apply(lambda x: int(x[0:2]))  # 使用 .loc 进行显式赋值
Login4.head()
```

	user_id	login_data	login_hour	login_newhour
38	用户 3	2018-09-23	00:56:32	0
88	用户 3	2018-10-13	09:19:45	9
89	用户 3	2018-10-13	16:02:59	16
104	用户 3	2018-10-20	17:10:33	17
135	用户 3	2018-11-04	18:02:06	18

```
holidays = Login4.groupby(by=Login4['login_newhour'], as_index=False)['user_id'].count()
holidays['holidays'] = holidays['user_id']
holidays = holidays[['login_newhour', 'holidays']]
holidays.head()
```

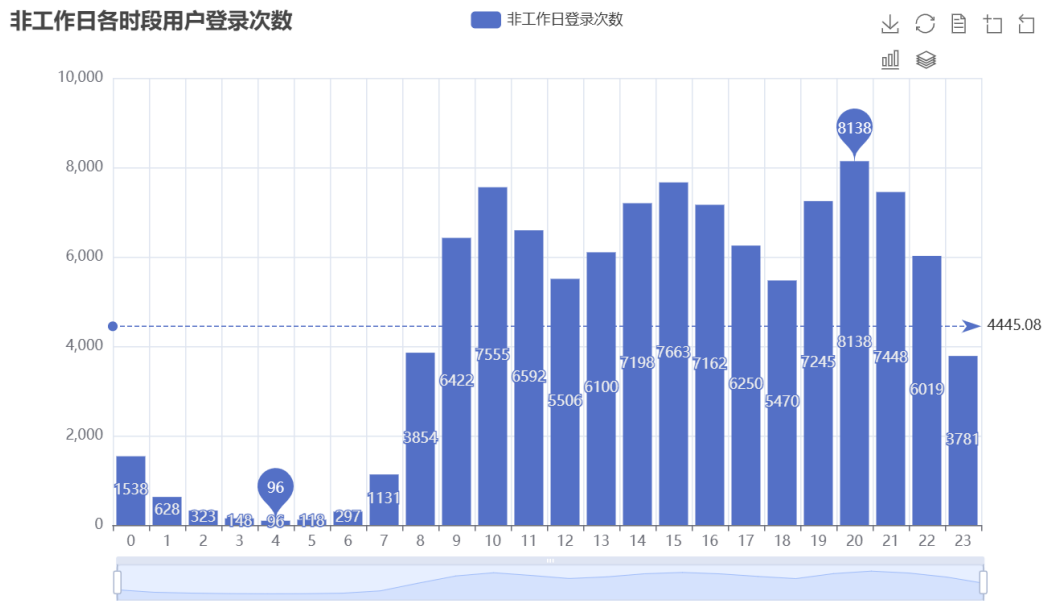
	login_newhour	holidays
0	0	1538
1	1	628
2	2	323
3	3	148
4	4	96

制作图表

```
# 准备数据
attr = list(holidays['login_newhour'])
v1 = list(holidays['holidays'])
# 创建柱状图对象
bar = (
    Bar()
    .add_xaxis(attr)
    .add_yaxis('非工作日登录次数', v1)
    .set_global_opts(
```

```
        title_opts=opts.TitleOpts(title="非工作日各时段用户登录次数
"),  # 标题
        toolbox_opts=opts.ToolboxOpts(),  # 工具箱
        datazoom_opts=opts.DataZoomOpts(range_start=0, range_end=100)  #
 横轴缩放
    )
    .set_series_opts(
        label_opts=opts.LabelOpts(is_show=True),  # 显示数据标签
        markpoint_opts=opts.MarkPointOpts(
            data=[
                opts.MarkPointItem(type_="max", name="最大值"),
                opts.MarkPointItem(type_="min", name="最小值"),
            ]
        ),
        markline_opts=opts.MarkLineOpts(
            data=[
                opts.MarkLineItem(type_="average", name="平均值")
            ]
        )
    )
)
# 渲染图表
bar.render_notebook()
bar.render("non_workday_login_analysis.html")  # 保存到本地
```

non_workday_login_analysis.html



非工作日活跃时间段为上午 9-11 点、下午 14-16 点、晚间 19-21 点。峰值

出现在晚间 20 点。

2.4 计算平台用户的流失率

设置时间范围，使用 pd.date_range 创建一个结束时间 Tend，设定为'2020-06-18 23:59:59'。对 new_users 数据进行去重处理，确保每个用户记录唯一。添加 Tend 列到 new_users 数据中，确保该列的长度与 new_users 一致。将 recently_logged 列中的无效数据转换为 NaT（Not a Time），以确保后续计算的准确性。计算每个用户的流失时长 longtime，即从最后一次登录时间 recently_logged 到结束时间 Tend 的天数差。删除包含 NaN 值的行，确保后续分析的数据完整性。最后查看 longtime 列的描述性统计信息，包括均值、标准差、最小值、最大值等，帮助理解用户流失时长的分布情况。按 longtime 分组并统计每组的数量，生成一个新的数据框 new_users4，用于进一步分析不同流失时长的用户数量。

```
Tend = pd.date_range('2020-06-18 23:59:59', periods=1)
new_users.drop_duplicates(inplace=True)
new_users['Tend'] = list(Tend) * len(new_users)  # 确保 Tend 列的长度
与 new_users 一致
# 处理 recently_logged 列中的无效数据
new_users['recently_logged'] = pd.to_datetime(new_users['recently_logge
d'], errors='coerce')
# 计算 longtime 列
new_users['longtime'] = (pd.to_datetime(new_users['Tend']) - pd.to_date
time(new_users['recently_logged'])).dt.days
# 删除包含 NaN 值的行
new_users3 = new_users.dropna(subset=['longtime'])
# 查看数据描述
new_users3['longtime'].describe()
# 按 longtime 分组并计数
new_users4 = new_users3.groupby(by='longtime', as_index=False).count()
```

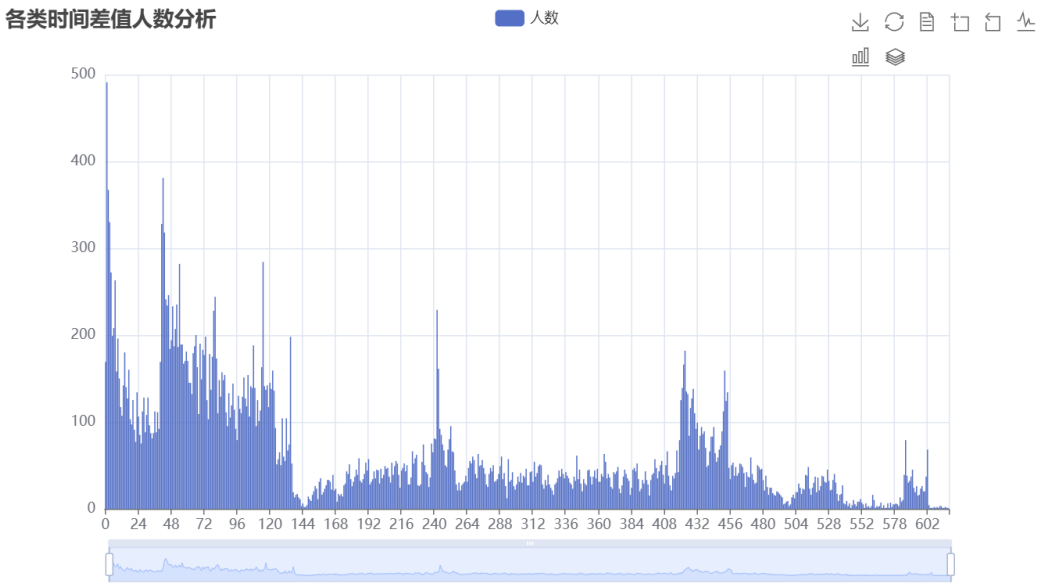
	longtime	user_id	register_time	recently_logged	learn_time	Tend
0	0.0	169	169	169	169	169
1	1.0	491	491	491	491	491
2	2.0	367	367	367	367	367
3	3.0	330	330	330	330	330
4	4.0	272	272	272	272	272

制作图表

```
x_data = list(new_users4['longtime'])
y_data = list(new_users4['user_id'])
bar = (Bar()
       .add_xaxis(x_data)
```

```
        .add_yaxis('人数', y_data)
        .set_series_opts(
         # 为了不影响标记点，这里把标签关掉
         label_opts=opts.LabelOpts(is_show=False))
        .set_global_opts(
            title_opts=opts.TitleOpts(title="各类时间差值人数分析
"),      # 标题
            toolbox_opts=opts.ToolboxOpts(),                          #
 工具箱
            datazoom_opts=opts.DataZoomOpts(range_start=0, range_end=10
0) # 横轴缩放
        )
      )
bar.render_notebook()
bar.render("lostgap.html")
```

lostgap.html




```
#流失天数及流失率
new_users5  = new_users3[new_users3['longtime'] >90]
new_users6 =  new_users5.groupby(by='longtime',as_index=False).count()
```

	longtime	user_id	register_time	recently_logged	learn_time	Tend
0	91.0	105	105	105	105	105
1	92.0	119	119	119	119	119
2	93.0	144	144	144	144	144
3	94.0	114	114	114	114	114
4	95.0	92	92	92	92	92

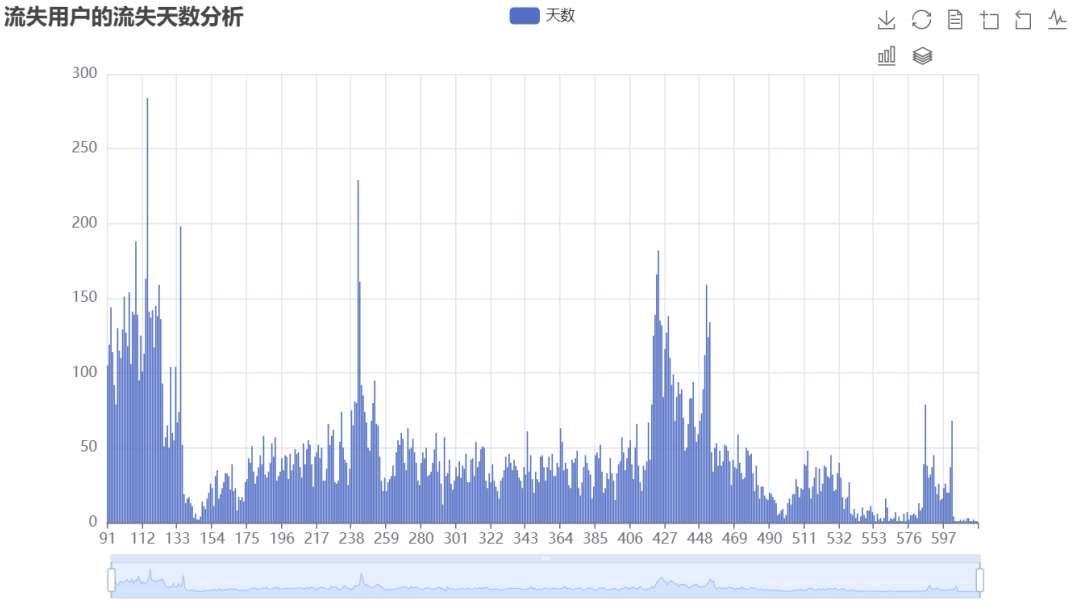
制作图表及输出内容

```
x_data = list(new_users6['longtime'])
y_data = list(new_users6['user_id'])
bar = (Bar()
       .add_xaxis(x_data)
       .add_yaxis('天数', y_data)
       .set_series_opts(
        # 为了不影响标记点，这里把标签关掉
        label_opts=opts.LabelOpts(is_show=False))
       .set_global_opts(
            title_opts=opts.TitleOpts(title="流失用户的流失天数分析
"),                          # 标题
            toolbox_opts=opts.ToolboxOpts(),
                    # 工具箱
            datazoom_opts=opts.DataZoomOpts(range_start=0,range_end=100
)                           # 横轴缩放
        )
      )
bar.render_notebook()
bar.render("lostdays.html")
print('流失率：',new_users5['longtime'].count()/43717)
print('流失人数：',new_users5['longtime'].count())
```

流失率： 0.5295651577189652

流失人数： 23151

lostdays.html



3. 线上课程推荐

3.1 根据用户参与学习的记录，统计每门课程的参与人数，计算每门课程的受欢迎程度，列出最受欢迎的前 10 门课程，并绘制相应的柱状图。

```
#3.线上课程推荐
study_information1= study_information.groupby(by='course_id',as_index=False)['user_id'].count()
study_information1['num'] = study_information1['user_id']
study_information2 = study_information1[['course_id','num']]
study_information2.head()
#选课人数分析
total_students = study_information2['num'].sum()
print(f'总选课人数：{total_students}')
```

总选课人数：190736

根据选课人数的不同区间，统计每个区间内的选课人数总和，并计算这些选课人数占总学生数的比例。使用一个列表 intervals 定义了多个选课人数区间，使用 for 循环遍历 intervals 列表中的每个区间。对于每个区间，从 study_information2 数据框中筛选出选课人数在该区间内的记录，并计算这些记录的选课人数总和 num。

```
# 定义选课人数区间
intervals = [
    (10000, float('inf'), '10000以上'),
    (5000, 10000, '5000-10000'),
    (1000, 5000, '1000-5000'),
    (500, 1000, '500-1000'),
    (100, 500, '100-500'),
    (0, 100, '100以下')
]
for lower, upper, label in intervals:
    num = study_information2[(study_information2['num'] > lower) & (study_information2['num'] <= upper)]['num'].sum()
    print(f'选课人数{label}：{num}')
    print(f'选课人数占比：{num / total_students:.4f}')
```

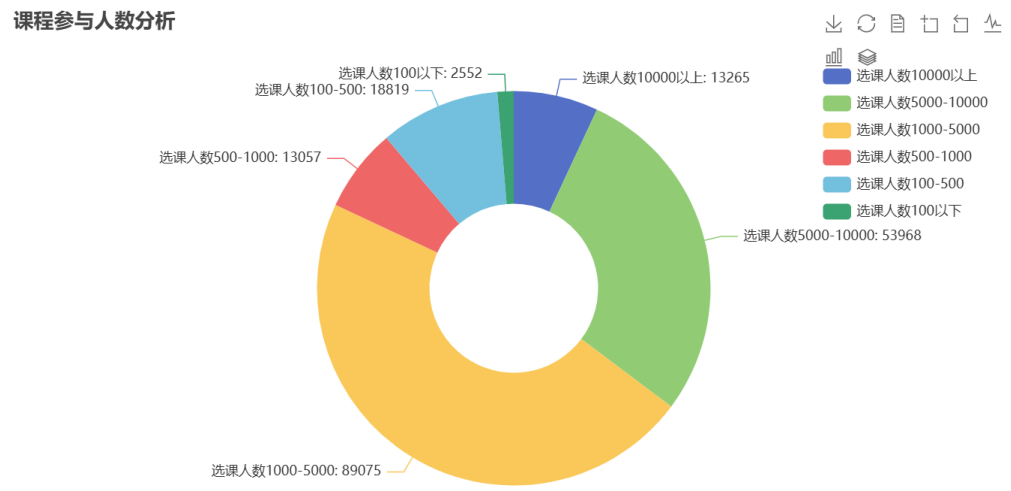
选课人数10000以上：13265
选课人数占比：0.0695
选课人数5000-10000：53968
选课人数占比：0.2829
选课人数1000-5000：89075

选课人数占比：0.4670
选课人数 500-1000：13057
选课人数占比：0.0685
选课人数 100-500：18819
选课人数占比：0.0987
选课人数 100 以下：2552
选课人数占比：0.0134

制作图表

```
from pyecharts.charts import Pie
x_data = ["选课人数 10000 以上", "选课人数 5000-10000", "选课人数 1000-5000", "选课人数 500-1000", "选课人数 100-500","选课人数 100 以下"]
y_data = [13265, 53968, 89075, 13057, 18819,2552]
c = (
    Pie()
    .add("", [list(z) for z in zip(x_data, y_data)],radius=["30%", "70%"])  # zip 函数两个部分组合在一起 list(zip(x,y))-----> [(x,y)]
    .set_global_opts(
        title_opts=opts.TitleOpts(title="课程参与人数分析"), # 标题
        toolbox_opts=opts.ToolboxOpts(),
                # 工具箱
        legend_opts=opts.LegendOpts(orient="vertical", pos_top="10%", pos_left="80%")) #图例设置
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}: {c}"))  # 数据标签设置
    )
c.render_notebook()
c.render("lesson.html")
```

lesson.html



对 study_information2 数据框进行描述性统计，并基于选课人数计算一个新的列 y，然后根据 y 列的值进行排序并提取前 10 条记录。

```
study_information2.describe()
study_information2['y'] = (study_information2['num']-1)/13624
study_information2 = study_information2.sort_values(by='y',ascending=False)
study_information3 = study_information2.iloc[0:10,:]
study_information3
```

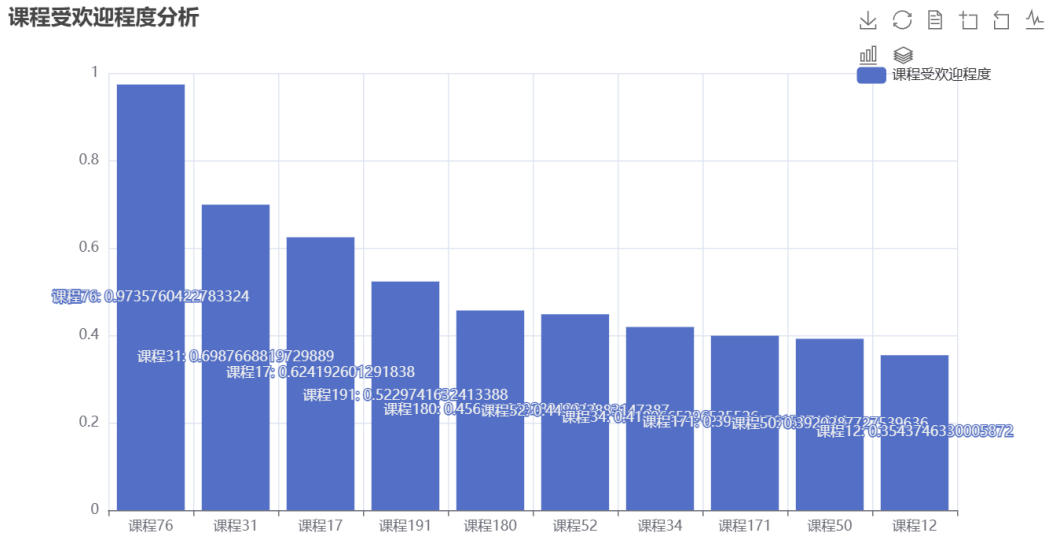
	course_id	num	y
214	课程 76	13265	0.973576
166	课程 31	9521	0.698767
79	课程 17	8505	0.624193
103	课程 191	7126	0.522974
91	课程 180	6223	0.456694

制作图表

```
x_data = list(study_information3['course_id'])
y_data_line = list(study_information3['y'])
from pyecharts.charts import Bar
from pyecharts import options as opts
b = (
    Bar()
    .add_xaxis(x_data)
    .add_yaxis("课程受欢迎程度", y_data_line)
    .set_global_opts(
        title_opts=opts.TitleOpts(title="课程受欢迎程度分析"),  # 标题
        toolbox_opts=opts.ToolboxOpts(),                        # 工具箱
        legend_opts=opts.LegendOpts(orient="vertical", pos_top="10%", pos_left="80%")  # 图例设置
    )
```

```
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}: {c}"))  # 数据标签设置
)
b.render_notebook()
b.render("lesson_popular_b.html")
```

lesson_popular_b.html



4. 自由分析

通过计算留存率，进行用户生命周期分析

新用户留存率：计算新用户在注册后的第一个月、三个月、六个月等时间段内的留存率，分析用户流失的原因。

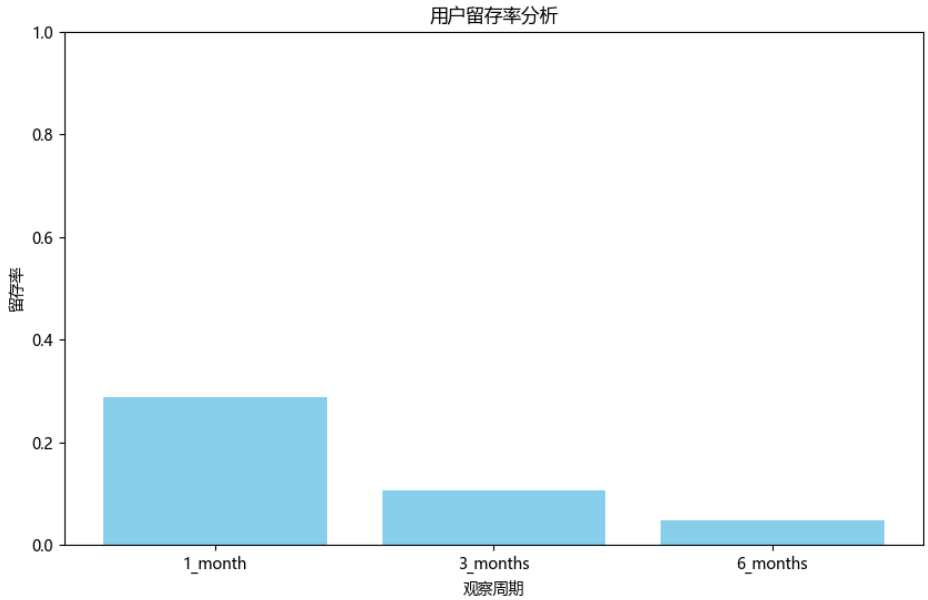
```
from datetime import timedelta
# 设置中文字体
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题
# 读取用户数据
users = pd.read_csv(r'C:\Users\DELL\Desktop\数字图书馆关键技术期末作业-20250108\线上教育课程数据分析\users.csv',
                    encoding='gbk')
# 将注册时间和最近登录时间转换为日期时间格式
users['register_time'] = pd.to_datetime(users['register_time'], errors='coerce')
users['recently_logged'] = pd.to_datetime(users['recently_logged'], errors='coerce')
```

```
# 定义观察日期
observation_dates = {
    '1_month': timedelta(days=30),
    '3_months': timedelta(days=90),
    '6_months': timedelta(days=180)
}
# 初始化留存率字典
retention_rates = {}
# 计算每个观察日期的留存率
for period, delta in observation_dates.items():
    # 计算观察日期
    observation_date = users['register_time'] + delta
    # 统计在观察日期仍然活跃的用户数
    active_users = users[users['recently_logged'] >= observation_date]
    retention_count = active_users.shape[0]
    # 计算留存率
    retention_rate = retention_count / users.shape[0]
    retention_rates[period] = retention_rate
# 打印留存率
for period, rate in retention_rates.items():
    print(f'{period} 留存率: {rate:.2%}')
```

1_month 留存率: 28.87%
3_months 留存率: 10.51%
6_months 留存率: 4.83%

绘制图表

```
# 绘制留存率图表
periods = list(retention_rates.keys())
rates = list(retention_rates.values())
plt.figure(figsize=(10, 6))
plt.bar(periods, rates, color='skyblue')
plt.title('用户留存率分析')
plt.xlabel('观察周期')
plt.ylabel('留存率')
plt.ylim(0, 1)
plt.show()
```



通过数据可以得出，经过一个月能留存的用户为 28.87%，三个月后下降至 10.51%，六个月后仅有 4.83%。作为线上教育平台，可以考虑是不是长期课程开设太少或使用上存在不便利，导致了用户流失，长期用户不足。