问题1

In [1]:
```
import os
import pathlib

import plotly
import numpy as np
import pandas as pd

import plotly.express as px
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)

# import warnings
# warnings.filterwarnings("ignore")
```

In [2]:
```
ROOTDIR = pathlib.Path(os.path.abspath('.'))
IMG_HTML = ROOTDIR / 'img-html'
IMG_PNG = ROOTDIR / 'img-png'
IMG_SVG = ROOTDIR / 'img-svg'
DATA_RAW = ROOTDIR / 'data-raw'
DATA_COOKED = ROOTDIR / 'data-processed'
```

In [3]:
```
area_bound = 2500
dis_threshold = 10
macro_base_station = {"coverage": 30, "cost": 10}
micro_base_station = {"coverage": 10, "cost": 1}
```

读取并观察数据

In [4]:
```
weak_grid_data = pd.read_csv(DATA_RAW / '附件1 弱覆盖栅格数据(筛选).csv')
exist_site_data = pd.read_csv(DATA_RAW / '附件2 现网站址坐标(筛选).csv')
```

In [5]:
```
# TODO 查看附件1数据
print(weak_grid_data.info())
print(weak_grid_data.describe())
weak_grid_data  # 坐标x，坐标y，业务量
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 182807 entries, 0 to 182806
Data columns (total 3 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   x        182807 non-null  int64  
 1   y        182807 non-null  int64  
 2   traffic  182807 non-null  float64
dtypes: float64(1), int64(2)
memory usage: 4.2 MB
None
                   x              y        traffic
count  182807.000000  182807.000000  182807.000000
mean     1395.314528     995.012877      38.599343
std       783.230529     733.291862     336.383875
min         0.000000       0.000000       0.000192
25%       546.000000     348.000000       0.596106
50%      1678.000000     869.000000       3.604328
75%      2048.000000    1453.000000      17.901928
max      2499.000000    2499.000000   47795.011719
```

Out[5]:

	x	y	traffic
0	66	1486	140.581390
1	67	1486	140.518829
2	177	1486	48.919178
3	187	1486	4.322495
4	284	1486	71.528404
...	...	...	...
182802	2350	2123	0.178571
182803	2353	2123	5.159708
182804	2354	2123	5.134017
182805	2355	2123	2.599999
182806	2372	2123	57.814999

182807 rows × 3 columns

In [6]:

```
# TODO 查看附件2数据
print(exist_site_data.info())
```

```
print(exist_site_data.describe())
exist_site_data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1474 entries, 0 to 1473
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      1474 non-null   int64
 1   x       1474 non-null   int64
 2   y       1474 non-null   int64
dtypes: int64(3)
memory usage: 34.7 KB
None
                id            x            y
count  1474.000000  1474.000000  1474.000000
mean   4175.835821  1332.259837  1389.444369
std    2371.172197   704.638569   695.610340
min       1.000000     1.000000     1.000000
25%    2129.500000   763.250000   834.000000
50%    4205.000000  1303.500000  1498.500000
75%    6216.750000  1970.000000  1948.750000
max    8286.000000  2499.000000  2499.000000
```

Out[6]:

	id	x	y
0	1	818	2020
1	4	713	2013
2	33	2305	291
3	35	700	1953
4	36	949	2293
...	...	...	...
1469	8254	2324	1625
1470	8257	1135	852
1471	8278	2053	1818
1472	8282	1432	1797
1473	8286	1584	898

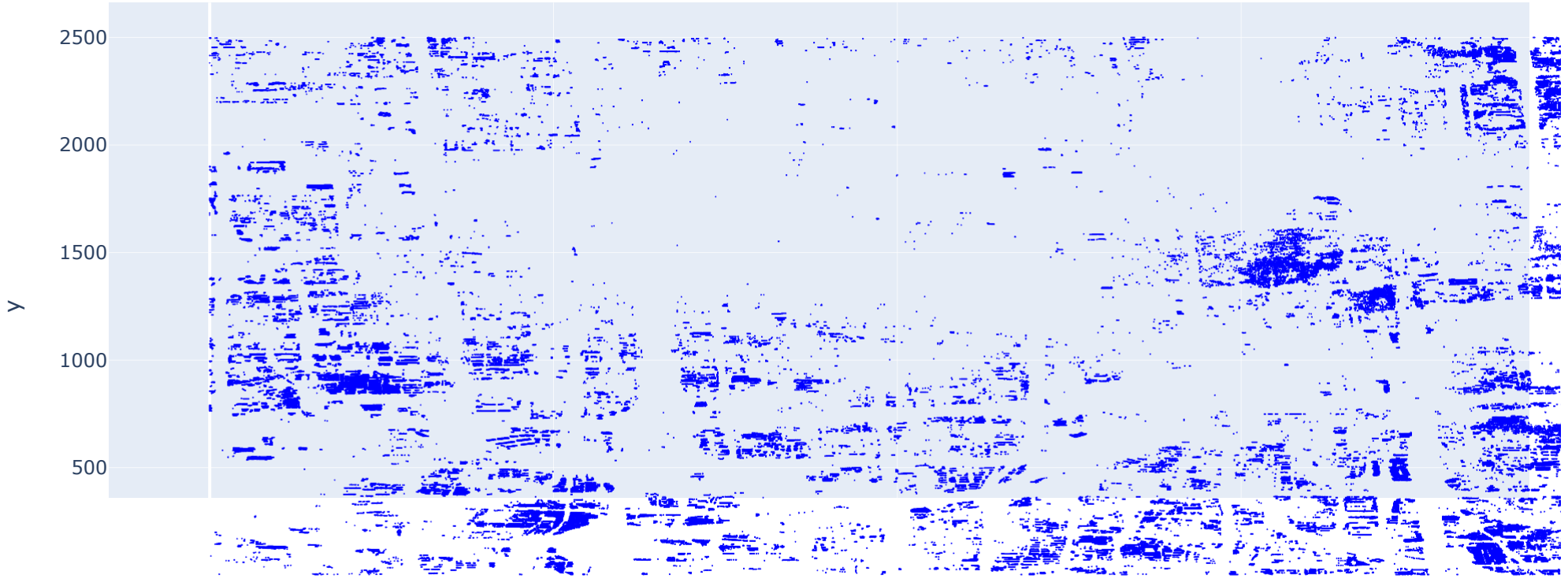
1474 rows × 3 columns

In [7]:
```python
def check_point(x, y, exist_site_data=exist_site_data):
    """检查某点是否已经存在基站"""
    return x in exist_site_data.loc[:, "x"].values and y in exist_site_data.loc[:, "y"].values
check_point(1585, 898), check_point(818, 2020)  # False, True
```

Out[7]:
```
(False, True)
```

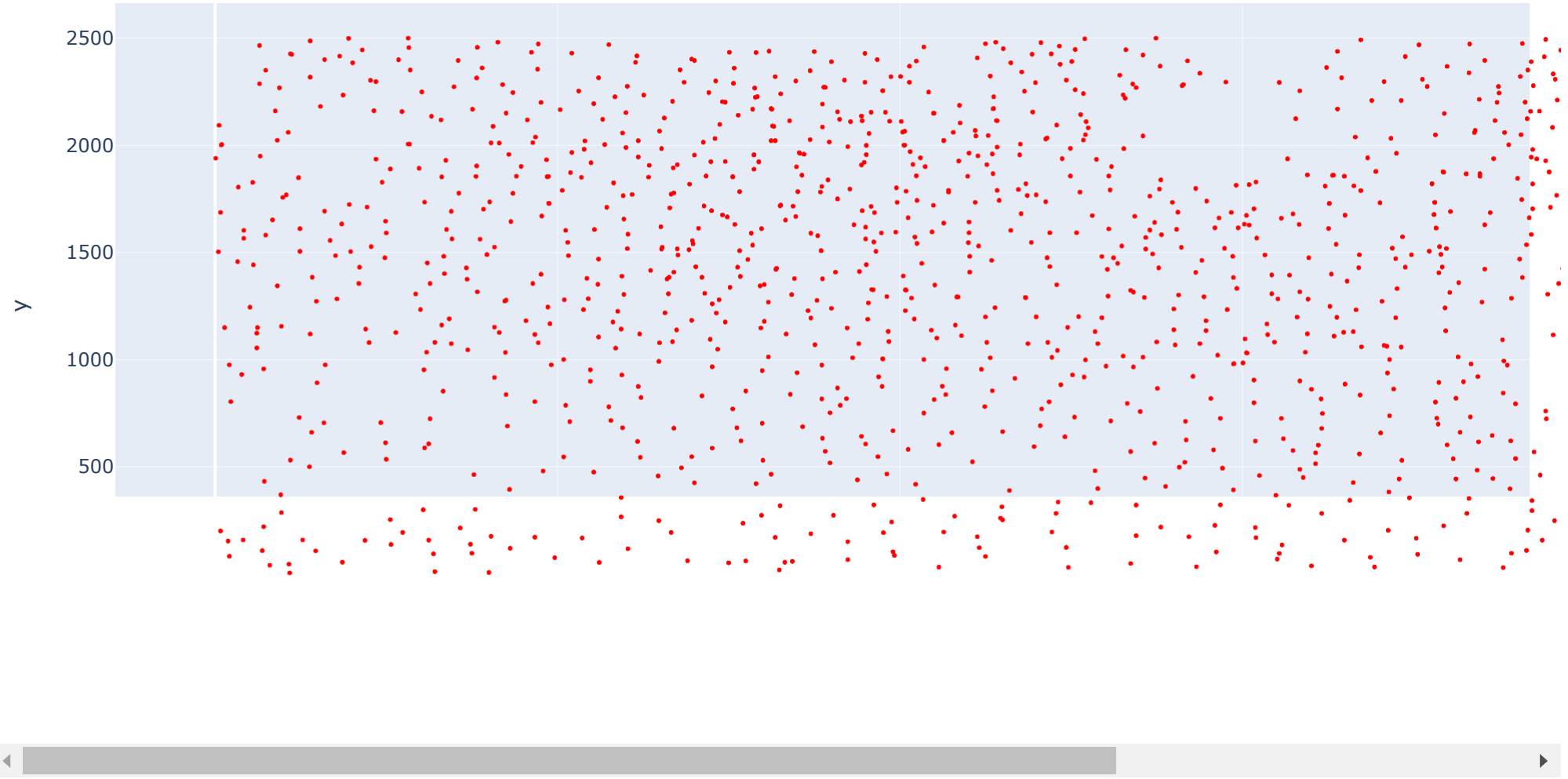
可视化数据

In [8]:
```python
def plot_weak_grid_data(data=weak_grid_data):
    """绘制 弱覆盖栅格 的散点图"""
#     import plotly.express as px
    fig = px.scatter(data_frame=data, x="x", y="y")
    fig.update_traces(marker=dict(color='blue', size=1))
    return fig

fig = plot_weak_grid_data()
fig.write_html(IMG_HTML / 'question1-weak_grid.html')
fig.write_image(IMG_PNG / 'question1-weak_grid.png')
fig.write_image(IMG_SVG / 'question1-weak_grid.svg')
fig.show()  # 图很大，不建议使用 notebook 查看，建议使用图片查看器/浏览器查看
del fig
```



In [9]:

```python
def plot_exist_site_data():
    """绘制 现网站址坐标 的散点图"""
#     import plotly.express as px
    fig = px.scatter(data_frame=exist_site_data, x="x", y="y")
    fig.update_traces(marker=dict(color='red', size=3))
    return fig

fig = plot_exist_site_data()
fig.write_html(IMG_HTML / 'question1-exist_site.html')
fig.write_image(IMG_PNG / 'question1-exist_site.png')
fig.write_image(IMG_SVG / 'question1-exist_site.svg')
fig.show()  # 图很大，不建议使用 notebook 查看，建议使用图片查看器/浏览器查看
del fig
```



In [10]:

```python
def plot_all_data():
    """绘制 所有数据（弱覆盖栅格、现网站址坐标） 的散点图"""
#     import plotly.express as px
#     import plotly.graph_objs as go
#     from plotly.offline import init_notebook_mode, iplot
#     init_notebook_mode(connected=True)
    trace1 = go.Scatter(
        mode='markers',
        x=weak_grid_data.loc[:, "x"], y=weak_grid_data.loc[:, "y"],
        marker={"color": "blue", "size": 1},
        name='弱覆盖点',
    )
```

```
    trace2 = go.Scatter(
        mode='markers',
        x=exist_site_data.loc[:, "x"], y=exist_site_data.loc[:, "y"],
        marker={"color": "red", "size": 3},
        name="现网基站",
    )
    data = [trace1, trace2]
    fig = go.Figure(data=data)
    return fig

fig = plot_all_data()
fig.write_html(IMG_HTML / 'question1-all.html')
fig.write_image(IMG_PNG / 'question1-all.png')
fig.write_image(IMG_SVG / 'question1-all.svg')
# fig.show()  # 图很大，不建议使用 notebook 查看，建议使用图片查看器/浏览器查看
del fig
```

这里文件太大，不方便展示，自行查看 `question1-all.html` 文件

求解模型

首先，根据单位成本业务量计算**每个基站的类型**

其次，**除去**已经被现网点(有 2 种基站类型)覆盖的**弱覆盖点**

然后，对剩下的还没被现网点(有 2 种基站类型)覆盖的弱覆盖点进行 **kmenas 聚类**，选取其聚类中心为**新建基站**

最后，对于不同的 k，**评价、规划**选出最佳新建基站方案（站址规划）（约束：使得弱覆盖点总业务量的 90% 被规划基站覆盖）

根据单位成本业务量计算每个基站的类型

为后面的步骤：除去已经被现网点覆盖的弱覆盖点，做准备

In [11]:
```
# TODO 根据单位成本业务量计算每个基站的类型（直接评价）（不使用 jit 即时编译加速计算）（已经弃用）
def out_of_use():
    """已弃用代码，最终使用的代码在后面"""
    import tqdm
    import numpy as np
    import pandas as pd

    exist_site = exist_site_data.copy()
    weak_grid = weak_grid_data.copy()
    coverage = [10, 30]
    cost = [1, 10]
```

```
    def is_far(i, j, cover):
        return abs(exist_site.iloc[i, 1] - weak_grid.iloc[j, 0]) > cover or \
                abs(exist_site.iloc[i, 2] - weak_grid.iloc[j, 1]) > cover

    def is_in(i, j, cover):
        return np.sqrt((exist_site.iloc[i, 1] - weak_grid.iloc[j, 0])**2 + \
                       (exist_site.iloc[i, 2] - weak_grid.iloc[j, 1])**2) < cover

    base_site_type = []
    for i in tqdm.trange(len(exist_site)):
        business_cost = []
        for idx, c in enumerate(coverage):
            business_cost_tmp = 0
            for j in range(len(weak_grid)):
                if not is_far(i, j, c) and is_in(i, j, c):
                    business_cost_tmp += weak_grid.iloc[j, 2]
            business_cost_tmp /= cost[idx]
            business_cost.append(business_cost_tmp)
        base_site_type.append(1 if business_cost[0] > business_cost[1] else 2)
    base_site_type
```

In [12]:

```
# TODO 根据单位成本业务量计算每个基站的类型（直接评价）（使用 jit 即时编译加速计算）（这块代码只需要运行一次）
# （因为改代码用于处理数据，故只需要运行一次即可）

import numba as nb
from numba import njit, prange, jit
from time import time

import warnings
warnings.filterwarnings("ignore")  # 忽略 jit 的警告

exist_site = np.array(exist_site_data).copy()  # 深拷贝，不改动原始数据
weak_grid = np.array(weak_grid_data).copy()  # 深拷贝，不改动原始数据
business_volume = 0
coverage = [10, 30]  # 微、宏基站的覆盖范围
cost = [1, 10]  # 微、宏基站的成本

def is_far(i, j, cover, exist_site=exist_site, weak_grid=weak_grid):
    """判断某个现网站址和某个弱覆盖栅格在 x 方向或者 y 方向的距离是否已经大于基站的覆盖范围
    :param i: 现网站址的索引
    :param j: 弱覆盖栅格的索引
    :param cover: 基站的覆盖范围
    :return: boolean
        使用 numpy
    """
```

```python
    distance_x = abs(exist_site[i, 1] - weak_grid[j, 0])
    distance_y = abs(exist_site[i, 2] - weak_grid[j, 1])
    return distance_x > cover or distance_y > cover

def is_in(i, j, cover, exist_site=exist_site, weak_grid=weak_grid):
    """判断某个现网站址和某个弱覆盖栅格的距离是否已经大于基站的覆盖范围
    :param i: 现网站址的索引
    :param j: 弱覆盖栅格的索引
    :param cover: 基站的覆盖范围
    :return: boolean
        使用 numpy
    """
    distance = np.sqrt((exist_site[i, 1] - weak_grid[j, 0])**2 + (exist_site[i, 2] - weak_grid[j, 1])**2)
#     distance = ((exist_site[i, 1] - weak_grid[j, 0])**2 + (exist_site[i, 2] - weak_grid[j, 1])**2)**0.5
    return distance < cover

# @njit(parallel=True)
# @jit(nopython=True)
# @njit
@jit  # 使用 jit 即时编译加速，用时大约 1h（不使用 jit 即使编译加速，用时大约 6h）
def cal_base_site_type():
    """根据 单位成本业务量(业务量/成本) 计算并返回 每个基站的类型
        如果微基站的单位成本业务量高，那么就选微基站，即类型为 1
        如果宏基站的单位成本业务量高，那么就选宏基站，即类型为 2
    :return: list 按照 id 顺序的每个基站的类型
    """
    t00 = time()
    l = len(exist_site)
    base_site_type = []
    for i in prange(l):  # numba: prange 并行计算，提高计算速度
        t0 = time()
        business_cost = []
        flag = 0
        for c in coverage:  # [10, 30]
            business_cost_tmp = 0
            for j in prange(len(weak_grid)):
                if not is_far(i, j, c) and is_in(i, j, c):
                    # trick: 先使用计算简单的，再使用计算复杂的，提高计算速度
                    business_cost_tmp += weak_grid[j, 2]
            business_cost_tmp /= cost[flag]
            flag += 1
            business_cost.append(business_cost_tmp)
        # business_cost[0] 微基站的单位成本业务量大，类型为1
        # business_cost[1] 宏基站的单位成本业务量大，类型为2
        base_site_type.append(1 if business_cost[0] > business_cost[1] else 2)
        if i % 100 == 0:  # jit 里面不太兼容 tqdm，所以手动打印查看进度
```

```
            t1 = time()
            _stars = '[' + '*' * (i // 100) + '.' * (l // 100 - i // 100) + ']'
            print("%4d / %4d" % (i, l), _stars, round(t1 - t0, 2), "s/iter")
    t11 = time()
    print('运行完毕，总用时：', t11 - t00)
    return base_site_type

# todo 获得基站类型的数据
base_site_list = cal_base_site_type()  # 跑一次这个大概要用 1h，已经提供运行完毕的文件，直接读取即可
print(base_site_list[:100])

def save_exist_site_type_data(
    exist_site=exist_site,
    base_site_list=base_site_list,
    columns=list(exist_site_data.columns)
):
    """保存并返回现网基站类型数据
    :param exist_site: np.array
    :param base_site_list: list
    :return: exist_site_type (pd.DataFrame)
    """
    exist_site_type = np.hstack((exist_site, np.array(base_site_list)[None, :].T))  # np.array
    exist_site_type_data = pd.DataFrame(exist_site_type, columns=columns + ['type'])  # pd.DataFrame

    exist_site_type_data.to_csv(DATA_COOKED / "question1-exist_site_type_data.csv")  # save

    return exist_site_type_data

# 保存读取
exist_site_type = save_exist_site_type_data()
exist_site_type
```

```
   0 / 1474 [..............] 3.7 s/iter
 100 / 1474 [*.............] 6.76 s/iter
 200 / 1474 [**............] 5.25 s/iter
 300 / 1474 [***...........] 2.76 s/iter
 400 / 1474 [****..........] 2.71 s/iter
 500 / 1474 [*****.........] 2.71 s/iter
 600 / 1474 [******........] 2.21 s/iter
 700 / 1474 [*******.......] 2.2 s/iter
 800 / 1474 [********......] 2.18 s/iter
 900 / 1474 [*********.....] 2.18 s/iter
1000 / 1474 [**********....] 2.2 s/iter
1100 / 1474 [***********...] 2.18 s/iter
1200 / 1474 [************..] 2.23 s/iter
1300 / 1474 [*************.] 2.17 s/iter
1400 / 1474 [**************] 2.04 s/iter
运行完毕，总用时： 4212.941307544708
[2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 1, 1, 2, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1,
2, 2, 2, 2, 2, 2, 2, 2]
```

Out[12]:

	id	x	y	type
0	1	818	2020	2
1	4	713	2013	2
2	33	2305	291	2
3	35	700	1953	2
4	36	949	2293	2
...	...	...	...	...
1469	8254	2324	1625	2
1470	8257	1135	852	2
1471	8278	2053	1818	2
1472	8282	1432	1797	2
1473	8286	1584	898	2

1474 rows × 4 columns

In [13]:

```
# # 保存读取
# exist_site_type = save_exist_site_type_data()
# exist_site_type
```

```
# 直接读取
exist_site_type = pd.read_csv(DATA_COOKED / "question1-exist_site_type_data.csv", index_col=0)
exist_site_type
```

Out[13]:

	id	x	y	type
0	1	818	2020	2
1	4	713	2013	2
2	33	2305	291	2
3	35	700	1953	2
4	36	949	2293	2
...	...	...	...	...
1469	8254	2324	1625	2
1470	8257	1135	852	2
1471	8278	2053	1818	2
1472	8282	1432	1797	2
1473	8286	1584	898	2

1474 rows × 4 columns

In [14]:

```
# 展示类型1数据
exist_site_type[exist_site_type.iloc[:, 3] == 1]
```

Out[14]:

	id	x	y	type
6	55	1787	1403	1
9	71	1359	1514	1
13	97	2491	758	1
48	358	2310	224	1
53	415	2104	1824	1
...	...	...	...	...
1448	8164	2143	1289	1
1449	8165	1828	279	1
1452	8188	1666	1230	1
1456	8200	2245	1436	1
1457	8206	1956	245	1

322 rows × 4 columns

In [15]:
```
# 展示类型2数据
exist_site_type[exist_site_type.iloc[:, 3] == 2]
```

Out[15]:

	id	x	y	type
0	1	818	2020	2
1	4	713	2013	2
2	33	2305	291	2
3	35	700	1953	2
4	36	949	2293	2
...	...	...	...	...
1469	8254	2324	1625	2
1470	8257	1135	852	2
1471	8278	2053	1818	2
1472	8282	1432	1797	2
1473	8286	1584	898	2

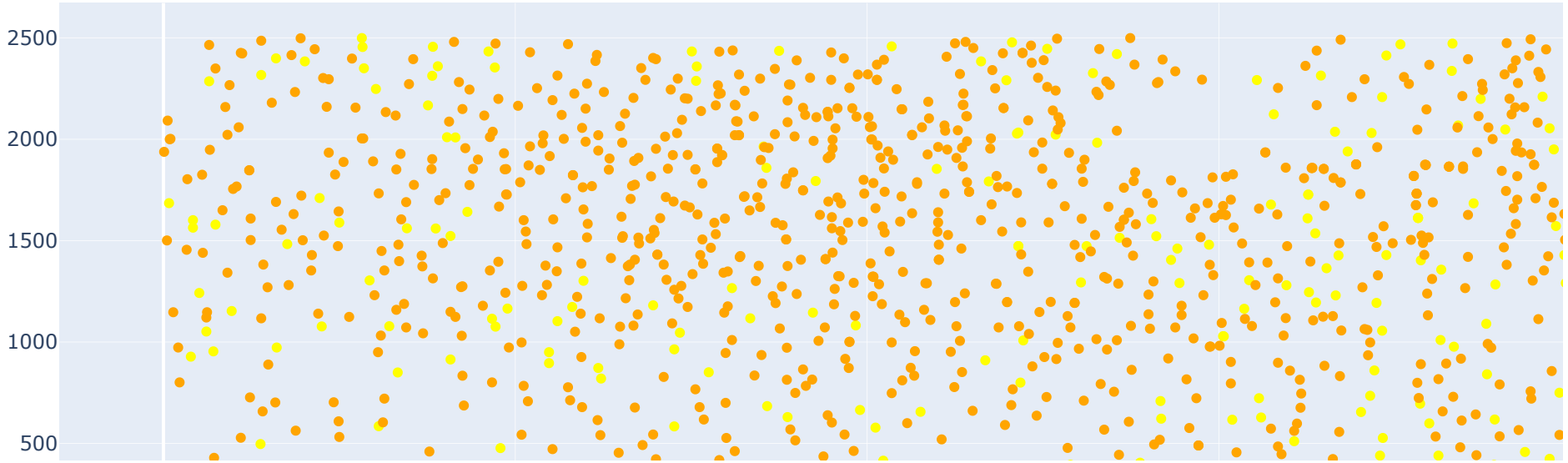
1152 rows × 4 columns

In [16]:
```python
def plot_diff_exist_site_type(exist_site_type=exist_site_type):
    """绘制、保存两种现网基站散点图"""
    trace_hong = go.Scatter(
        mode="markers",
        x=exist_site_type[exist_site_type.iloc[:, 3] == 2].iloc[:, 1],
        y=exist_site_type[exist_site_type.iloc[:, 3] == 2].iloc[:, 2],
        marker={"color": "orange"},
        name='宏基站',
    )
    trace_wei = go.Scatter(
        mode="markers",
        x=exist_site_type[exist_site_type.iloc[:, 3] == 1].iloc[:, 1],
        y=exist_site_type[exist_site_type.iloc[:, 3] == 1].iloc[:, 2],
        marker={"color": "yellow"},
        name='微基站',
    )
    data = [trace_wei, trace_hong]
    layout = go.Layout(title='现网基站的两种类型基站散点图')
    fig = go.Figure(data=data, layout=layout)
    fig.write_html(IMG_HTML / 'question1-exist_site_2types.html')
    fig.write_image(IMG_PNG / 'question1-exist_site_2types.png')
    fig.write_image(IMG_SVG / 'question1-exist_site_2types.svg')
```

```
    fig.show()
    return None
```

In [17]:
```
plot_diff_exist_site_type()
```

现网基站的两种类型基站散点图



除去已经被现网点覆盖的弱覆盖点

用于对其进行聚类，同时减少计算时间

In [18]:
```
# TODO 计算出要除去已经被现网点覆盖的弱覆盖点（使用 jit 即时编译加速计算，因为改代码用于处理数据，故只需要运行一次即可）
# 为什么重复代码？
# ①有些参数不一样，查看、修改参数方便
# ②由于 notebook 无法跳转代码，所以把所需的变量、函数写在一个 ceil 里面，方便
```

```python
import numba as nb
from numba import njit, prange, jit

import warnings
warnings.filterwarnings("ignore")  # 忽略 jit 的警告

exist_site = np.array(exist_site_type).copy()  # 跟上面不同，这里包含类型！！！
weak_grid = np.array(weak_grid_data).copy()
print(f"exist_site.shape: {exist_site.shape}\nweak_grid.shape: {weak_grid.shape}")

to_be_delete = [0 for _ in range(len(weak_grid))]  # 即将被“删除”的弱覆盖点的索引
coverage = [10, 30]
cost = [1, 10]

def is_far(i, j, cover):
    return abs(exist_site[i, 1] - weak_grid[j, 0]) > cover or \
           abs(exist_site[i, 2] - weak_grid[j, 1]) > cover
def is_in(i, j, cover):
    return np.sqrt((exist_site[i, 1] - weak_grid[j, 0])**2 + \
                   (exist_site[i, 2] - weak_grid[j, 1])**2) < cover

@jit  # 使用 jit 即时编译加速，用时大约 17min（不使用 jit 即使编译加速，用时大约 ?h，不知道没试过）
def del_weak_grid():
    """删除已经被现网点覆盖的弱覆盖点
    :return: None
        已经修改全局变量 to_be_delete
    """
    global to_be_delete

    t00 = time()
    l = len(exist_site)
    existing_business_volume = 0
    for i in prange(l):
        t0 = time()
        c = coverage[exist_site[i, 3] - 1]
        for j in prange(len(weak_grid)):
            if not is_far(i, j, c) and is_in(i, j, c):
                to_be_delete[j] = 1
                existing_business_volume += weak_grid[j, 2]
        if i % 100 == 0:  # jit 里面不太兼容 tqdm，所以手动打印查看进度
            t1 = time()
            _stars = '[' + '*' * (i // 100) + '.' * (l // 100 - i // 100) + ']'
            print("%4d / %4d" % (i, l), _stars, round(t1 - t0, 2), "s/iter")
    t11 = time()
    print('运行完毕，总用时：', t11 - t00)
```

```
    return None

del_weak_grid()  # 跑一次这个大概要用 1h，已经提供运行完毕的文件，直接读取即可

def save_remain_weak_grid_data(
    weak_grid_data=weak_grid_data,
    to_be_delete=to_be_delete,
):
    remain_weak_grid = pd.concat([weak_grid_data, pd.DataFrame(to_be_delete)], axis=1)
    remain_weak_grid.columns = list(weak_grid_data.columns) + ['remained']

    remain_weak_grid.to_csv(DATA_COOKED / "question1-remain_weak_grid_data.csv")  # save

    return remain_weak_grid

# 保存读取
remain_grid = save_remain_weak_grid_data()
remain_grid
```

```
exist_site.shape: (1474, 4)
weak_grid.shape: (182807, 3)
   0 / 1474 [..............] 0.61 s/iter
 100 / 1474 [*.............] 0.61 s/iter
 200 / 1474 [**............] 0.74 s/iter
 300 / 1474 [***...........] 0.59 s/iter
 400 / 1474 [****..........] 0.65 s/iter
 500 / 1474 [*****.........] 0.64 s/iter
 600 / 1474 [******........] 0.61 s/iter
 700 / 1474 [*******.......] 0.61 s/iter
 800 / 1474 [********......] 0.62 s/iter
 900 / 1474 [*********.....] 0.61 s/iter
1000 / 1474 [**********....] 0.61 s/iter
1100 / 1474 [***********...] 0.59 s/iter
1200 / 1474 [************..] 0.61 s/iter
1300 / 1474 [*************.] 0.61 s/iter
1400 / 1474 [**************] 0.61 s/iter
运行完毕，总用时： 920.439487695694
```

Out[18]:

	x	y	traffic	remained
0	66	1486	140.581390	0
1	67	1486	140.518829	0
2	177	1486	48.919178	1
3	187	1486	4.322495	1
4	284	1486	71.528404	0
...	...	...	...	...
182802	2350	2123	0.178571	0
182803	2353	2123	5.159708	0
182804	2354	2123	5.134017	0
182805	2355	2123	2.599999	0
182806	2372	2123	57.814999	0

182807 rows × 4 columns

In [19]:

```
# # 保存读取
# remain_grid = save_remain_weak_grid_data()
# remain_grid

# 文件读取
remain_grid = pd.read_csv(DATA_COOKED / "question1-remain_weak_grid_data.csv", index_col=0)
remain_grid  # delete or not
```

Out[19]:

	x	y	traffic	remained
0	66	1486	140.581390	0
1	67	1486	140.518829	0
2	177	1486	48.919178	1
3	187	1486	4.322495	1
4	284	1486	71.528404	0
...	...	...	...	...
182802	2350	2123	0.178571	0
182803	2353	2123	5.159708	0
182804	2354	2123	5.134017	0
182805	2355	2123	2.599999	0
182806	2372	2123	57.814999	0

182807 rows × 4 columns

In [20]:
```
# TODO 查看ba
cond = remain_grid.iloc[:, 3] == 1  # 删选出
print(f"已覆盖弱站点的业务量占比：{remain_grid[cond].sum()[2] / remain_grid.sum()[2] * 100}%")
total_bussiness = remain_grid.sum()[2]
print(f"总业务量：{total_bussiness}")
covered_business = remain_grid[cond].sum()[2]
print(f"已经覆盖的业务量：{covered_business}")
uncovered_bussiness = total_bussiness - covered_business
print(f"未覆盖的业务量：{uncovered_bussiness}")
need_to_cover_bussiness = total_bussiness * 0.9 - covered_business
print(f"需要覆盖的业务量：{need_to_cover_bussiness}")
```

```
已覆盖弱站点的业务量占比：46.345035166180985%
总业务量：7056230.114628
已经覆盖的业务量：3270212.3280309997
未覆盖的业务量：3786017.7865970004
需要覆盖的业务量：3080394.7751342
```

In [21]:
```
# todo plot 3 kinds point (suggest using pucharm to plot, opened with HTML)
def plot_diff_grid():
#     import plotly.graph_objs as go
#     from plotly.offline import init_notebook_mode, iplot
#     init_notebook_mode(connected=True)
```

```
    remain_cond = remain_grid.iloc[:, 3] == 0
    delete_cond = remain_grid.iloc[:, 3] == 1
    trace1 = go.Scatter(
        mode='markers',
        x=remain_grid[remain_cond].loc[:, 'x'], y=remain_grid[remain_cond].loc[:, 'y'],
        marker={"color": "blue", "size": 1},
        name='未覆盖的弱覆盖点',
    )
    trace2 = go.Scatter(
        mode='markers',
        x=remain_grid[delete_cond].loc[:, 'x'], y=remain_grid[delete_cond].loc[:, 'y'],
        marker={"color": "orange", "size": 1},
        name='已覆盖的弱覆盖点',
    )
    trace3 = go.Scatter(
        mode='markers',
        x=exist_site_data.loc[:, "x"], y=exist_site_data.loc[:, "y"],
        marker={"color": "red", "size": 2},
        name="现网基站",
    )
    data = [trace1, trace2, trace3]
    fig = go.Figure(data=data)
    return fig
fig = plot_diff_grid()
fig.write_html(IMG_HTML / 'question1-weak_grid(remained)&exist_site.html')
fig.write_image(IMG_PNG / 'question1-weak_grid(remained)&exist_site.png')
fig.write_image(IMG_SVG / 'question1-weak_grid(remained)&exist_site.svg')
# fig.show()  # 图很大，不建议使用 notebook 查看，建议使用图片查看器/浏览器查看
del fig
```

这里文件太大，不方便展示，自行查看 `question1-weak_grid(remained)&exist_site.html` 文件

聚类/直接选取站点

In [22]:
```
# TODO 重新读取数据
exist_site_type_data = pd.read_csv(DATA_COOKED / "question1-exist_site_type_data.csv", index_col=0)
remain_weak_grid_data = pd.read_csv(DATA_COOKED / "question1-remain_weak_grid_data.csv", index_col=0)
print(f"exist_site_type_data.shape: {exist_site_type_data.shape}")
print(f"remain_weak_grid_data.shape: {remain_weak_grid_data.shape}")
```

```
exist_site_type_data.shape: (1474, 4)
remain_weak_grid_data.shape: (182807, 4)
```

In [23]:
```
from sklearn.model_selection import train_test_split
```

```
from sklearn.cluster import KMeans
from sklearn import metrics
```

In [24]:
```
# TODO 查看数据
remain_cond = remain_weak_grid_data.iloc[:, 3] == 0
delete_cond = remain_weak_grid_data.iloc[:, 3] == 1
remain_weak_grid_data[remain_cond], remain_weak_grid_data[delete_cond]
```

Out[24]:
```
(           x     y     traffic  remained
 0         66  1486  140.581390         0
 1         67  1486  140.518829         0
 4        284  1486   71.528404         0
 6       1400  1486   17.549461         0
 7       1418  1486    5.520310         0
 ...      ...   ...         ...       ...
 182802  2350  2123    0.178571         0
 182803  2353  2123    5.159708         0
 182804  2354  2123    5.134017         0
 182805  2355  2123    2.599999         0
 182806  2372  2123   57.814999         0

 [136939 rows x 4 columns],
            x     y    traffic  remained
 2        177  1486  48.919178         1
 3        187  1486   4.322495         1
 5        309  1486   0.073663         1
 10      1483  1486   8.716330         1
 11      1554  1486   0.885215         1
 ...      ...   ...        ...       ...
 182786  2003  2123   1.830510         1
 182787  2004  2123   9.870605         1
 182788  2005  2123  12.248288         1
 182789  2202  2123   5.386420         1
 182799  2300  2123  15.452216         1

 [45868 rows x 4 columns])
```

In [25]:
```
import numpy as np
import pandas as pd
from time import time
from sklearn.cluster import KMeans
from numba import jit, njit, prange
import plotly.express as px
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
```

```python
class BestSites(object):
    def __init__(self, n_clusters=100):
        self.coverage = [10, 30]
        self.cost = [1, 10]
        self.bussiness = 0
        self.total_bussiness = 7056230.114628
        self.target_bussiness = 3080394.7751342
        self.mod = 200

        remain_weak_grid_data = pd.read_csv(DATA_COOKED / "question1-remain_weak_grid_data.csv", index_col=0)
        remain_cond = remain_weak_grid_data.iloc[:, 3] == 0
        self.remain_weak_grid_data = remain_weak_grid_data[remain_cond]

        self.cluster_centers_ = None
        self.labels_ = None
        self.remain_weak_grid_labels = None
#         self.n_clusters = n_clusters

        self.new_site_point_m2 = None

    # 自定义 n_clusters
    @property
    def n_clusters(self):
        return self._n_clusters
    @n_clusters.setter
    def n_clusters(self, n):
        self._n_clusters = n
        return None

    # todo 方法 2 从大到小直接选择最优站点（使用该方法）
    @jit
    def chose_best_point(self):
        remain_weak_grid = pd.DataFrame(self.remain_weak_grid_data.copy())
        remain_weak_grid.sort_values(by='traffic', ascending=False, inplace=True, ignore_index=True)
        print(remain_weak_grid)
        remain_weak_grid = np.array(remain_weak_grid)
        done_bussiness = 0
        new_site_point_m2 = []
        for i in prange(len(remain_weak_grid)):
            if done_bussiness >= self.target_bussiness:
                print("total_done_bussiness:", done_bussiness)
                self.new_site_point_m2 = new_site_point_m2
                return pd.DataFrame(new_site_point_m2, columns=["x", "y", 'type'])

            if remain_weak_grid[i, 3] == 0:
```

```
                remain_weak_grid[i, 3] = 1
                done_bussiness += remain_weak_grid[i, 2]
                x1, y1 = int(round(remain_weak_grid[i, 0])), int(round(remain_weak_grid[i, 1]))
#                 flag = np.random.randint(2)
                flag = 1 if i % 2 == 0 or i % 5 == 0 else 0
                new_site_point_m2.append([x1, y1, flag + 1])
                c = coverage[flag]

                for j in prange(len(remain_weak_grid)):
                    if remain_weak_grid[j, 3] == 0:
                        x2, y2 = remain_weak_grid[j, 0], remain_weak_grid[j, 1]
                        if not self.is_far(x1, y1, x2, y2, c) and self.is_in(x1, y1, x2, y2, c):
                            remain_weak_grid[j, 3] = 1
                            done_bussiness += remain_weak_grid[j, 2]

            if i % 300 == 0:
                print("i:", i, "business:", done_bussiness)
        self.new_site_point_m2 = new_site_point_m2.copy()
        return pd.DataFrame(new_site_point_m2, columns=["x", "y", 'type'])

    # todo 方法 1 使用kmeans聚类选择最优站点（未使用该方法，因为①耗时②选择站点效果不好）
#     @jit
    def clustering_kmeans(self):
        """计算 kmeans 聚类结果"""
        k_means = KMeans(n_clusters=self.n_clusters, random_state=10)
        print(k_means)
        time_begin = time()
        k_means.fit(self.remain_weak_grid_data.iloc[:, :2].values)

        self.cluster_centers_ = k_means.cluster_centers_.copy()
        self.labels_ =  k_means.labels_.copy()

        data_frame = self.remain_weak_grid_data.copy()
        data_frame.insert(3, "labels", self.labels_)
        data_frame.drop(columns="remained", axis=1, inplace=True)
#         print(data_frame)
        self.remain_weak_grid_labels = np.array(data_frame.copy())

        print(f"kmeans cost time: {time() - time_begin}")
        return k_means.cluster_centers_, k_means.labels_
    def plot_kmeans_res(self):
        """绘制 kmeans 聚类结果"""
#         print(self.remain_weak_grid_labels)
        remain_weak_grid_labels = self.remain_weak_grid_labels.copy()
        remain_weak_grid_labels = pd.DataFrame(
            remain_weak_grid_labels,
```

```
            columns=["x", "y", "traffic", "labels"],
        )
        fig = px.scatter(data_frame=remain_weak_grid_labels, x='x', y='y', color='labels')
        fig.update_traces(marker={"size": 1})
#         fig.show()
        return fig
#     @jit
    def is_far(self, x, y, i, j, c):
        return abs(x - i)> c or abs(y - j) > c
#     @jit
    def is_in(self, x, y, i, j, c):
        return (x - i)**2 + (y - j)**2 < c**2
    @jit
    def confirm_type(self):
        print("确定新建基站类型")
        new_base_site_type = []
        new_base_site = self.cluster_centers_
        remain_weak_grid = np.array(self.remain_weak_grid_labels)
#         print(new_base_site)
#         print(remain_weak_grid)
        for i in prange(len(new_base_site)):
            flag = 0
            bussiness_per_cost = []
            for coverage in self.coverage:
                business_per_cost_tmp = 0
                for j in prange(len(remain_weak_grid)):
                    x1, y1 = new_base_site[i, 0], new_base_site[i, 1]
                    x2, y2 = remain_weak_grid[j, 0], remain_weak_grid[j, 1]
                    c = coverage
                    if not self.is_far(x1, y1, x2, y2, c) and self.is_in(x1, y1, x2, y2, c):
                        business_per_cost_tmp += remain_weak_grid[j, 2]
                business_per_cost_tmp /= self.cost[flag]
                flag += 1
                bussiness_per_cost.append(business_per_cost_tmp)
            new_base_site_type.append(1 if bussiness_per_cost[0] > bussiness_per_cost[1] else 2)
#             if i % self.mod == 0:
#                 print("i:", i, "business:", done_bussiness)
        new_base_site_type = np.array(new_base_site_type)
#         print(new_base_site_type)
#         print(type(new_base_site_type))
#         print(self.cluster_centers_)
#         print(type(self.cluster_centers_))
        self.cluster_centers_ = np.concatenate((
            self.cluster_centers_.transpose(),
            new_base_site_type.reshape(1, -1)
        )).transpose()
```

```
#         print(self.cluster_centers_)
#         return new_base_site_type
#     @jit
    def cal_business(self):
        print("计算新建基站覆盖的业务量")
        self.bussiness = 0
        for i in prange(len(self.cluster_centers_)):
            coverage = self.coverage[int(self.cluster_centers_[i, 2]) - 1]
            for j in prange(len(self.remain_weak_grid_labels)):
                c = coverage
                x1, y1 = self.cluster_centers_[i, 0], self.cluster_centers_[i, 1]
                x2, y2 = self.remain_weak_grid_labels[j, 0], self.remain_weak_grid_labels[j, 1]
                if not self.is_far(x1, y1, x2, y2, c) and self.is_in(x1, y1, x2, y2, c):
                    self.bussiness += self.remain_weak_grid_labels[j, 2]
            if i % self.mod == 0:
                print(i)
        print(f"需要覆盖的业务量：{self.target_bussiness}")
        print(f"新建基站覆盖的业务量：{self.bussiness}")
        print(f"新建 / 需要：{self.bussiness / self.target_bussiness * 100}%")
        print(f"新建 / 总量：{self.bussiness / self.total_bussiness * 100}%")
        return self.bussiness >= self.target_bussiness
```

直接选取（使用该方法）

In [26]:
```
# todo method 2: chose best point
bb = BestSites()
bb.chose_best_point()
```

```
           x     y       traffic  remained
0        932  2210  18680.960938         0
1        972  1238  15063.482422         0
2       1229  1894  12733.102539         0
3       1311  2005  12311.877930         0
4       1611  2233  11579.516602         0
...      ...   ...           ...       ...
136934  2178  2348      0.000192         0
136935  2217  2402      0.000192         0
136936  2239  2456      0.000192         0
136937  2175  2349      0.000192         0
136938  2170  2344      0.000192         0

[136939 rows x 4 columns]
i: 0 business: 19008.385769000004
i: 300 business: 1243831.1821389985
i: 600 business: 1861172.3909059966
i: 900 business: 2237787.6476250137
i: 1200 business: 2466160.5638240105
i: 1500 business: 2622924.669223021
i: 1800 business: 2761005.805976014
i: 2100 business: 2851370.820233041
i: 2400 business: 2953135.8486840385
i: 2700 business: 3024676.8779580435
total_done_bussiness: 3082428.6270970367
```

Out[26]:

	x	y	type
0	932	2210	2
1	972	1238	1
2	1229	1894	2
3	1311	2005	1
4	1611	2233	2
...	...	...	...
783	1171	812	1
784	250	1065	1
785	2353	24	1
786	2422	583	2
787	2249	1453	2

788 rows × 3 columns

In [27]:

```
new_site_point_m2 = pd.DataFrame(bb.new_site_point_m2, columns=['x', 'y', 'type'])
new_site_point_m2.to_csv(DATA_COOKED / "question1-new_build_sites.csv")
new_site_point_m2
```

Out[27]:

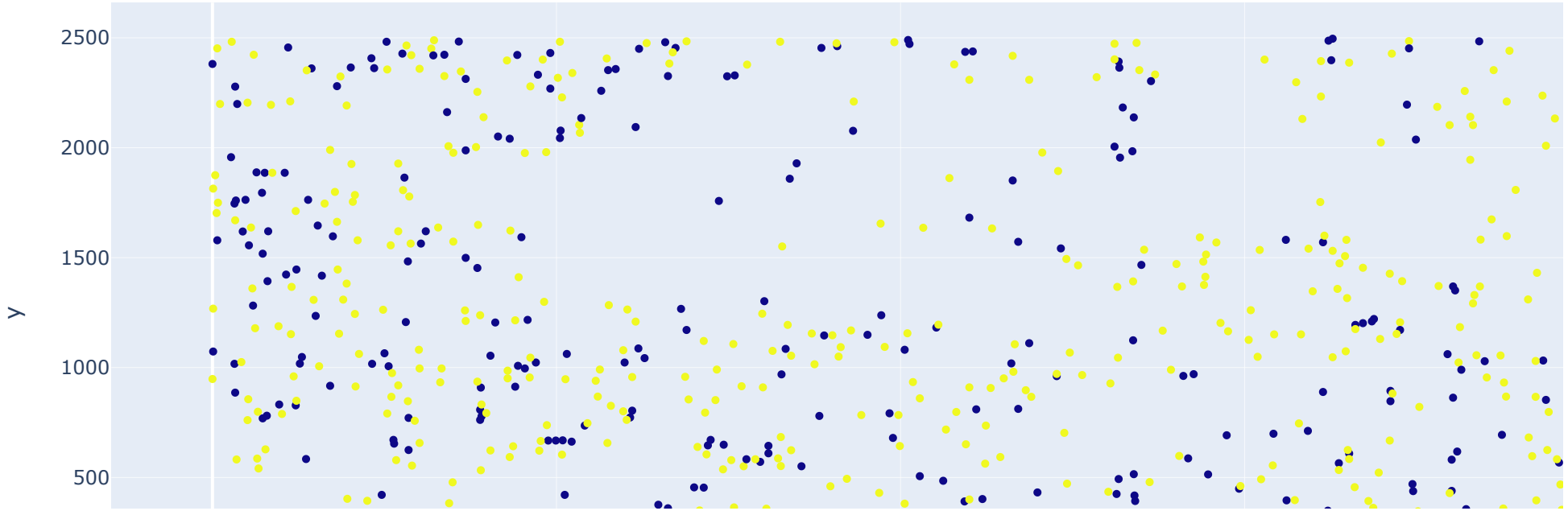
	x	y	type
0	932	2210	2
1	972	1238	1
2	1229	1894	2
3	1311	2005	1
4	1611	2233	2
...	...	...	...
783	1171	812	1
784	250	1065	1
785	2353	24	1
786	2422	583	2
787	2249	1453	2

788 rows × 3 columns

In [28]:

```
fig = px.scatter(data_frame=new_site_point_m2, x='x', y='y', color='type')
fig.update_layout(title='新建基站散点图')
fig.update_traces(marker={"size": 5})
fig.write_html(IMG_HTML / 'question1-new_build_sites.html')
fig.write_image(IMG_PNG / 'question1-new_build_sites.png')
fig.write_image(IMG_SVG / 'question1-new_build_sites.svg')
fig.show()
del fig
```

新建基站散点图



聚类选取（未使用该方法，可忽略）

kmeans 分几类，就以这几类的中心点为新基站坐标

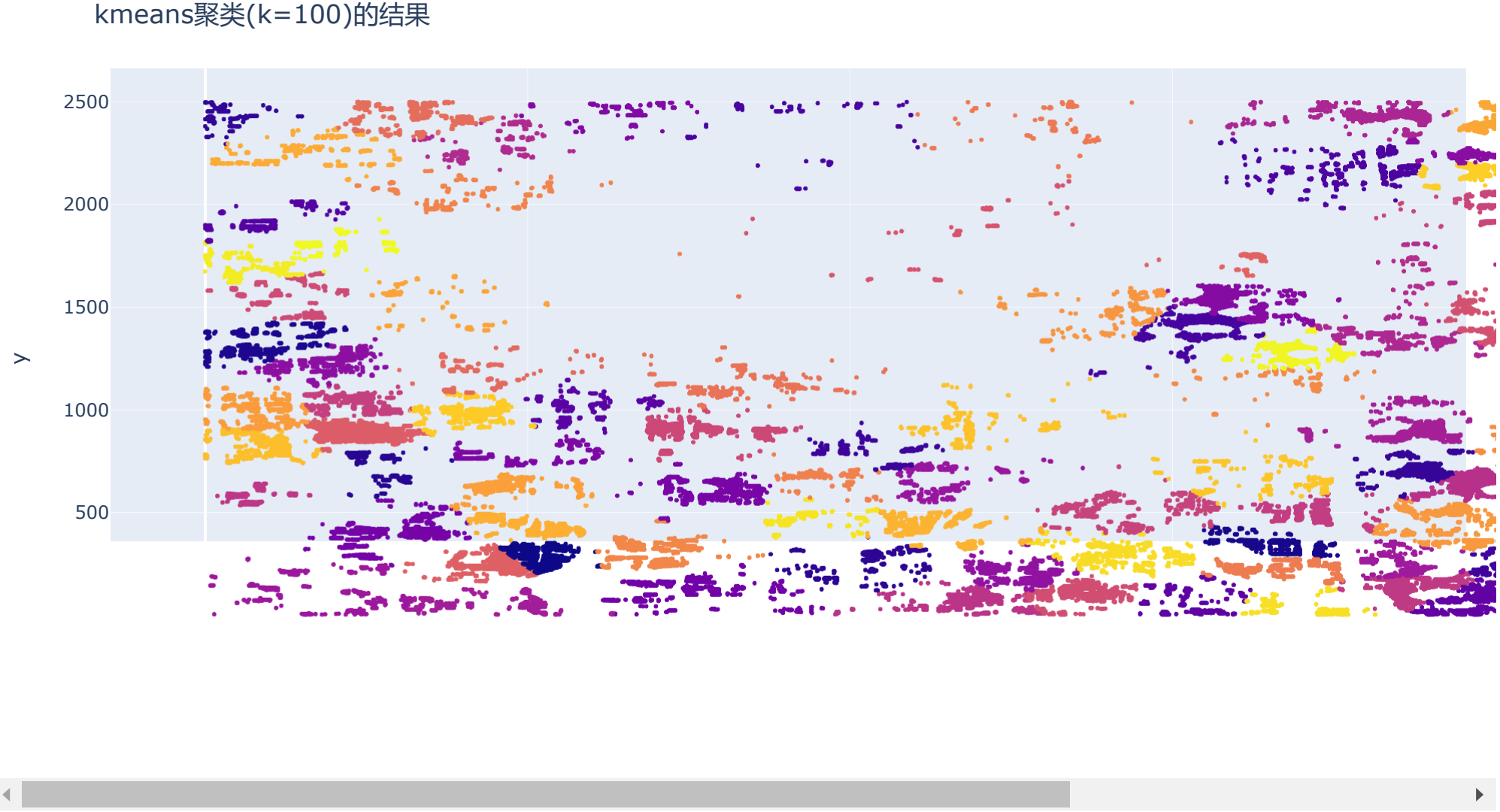
In [29]:
```
# todo method 1: kmeans
b = BestSites()
for n in range(100, 101):  # 这里以100为例(用时大约 1.5min)，实际上 1407 - 1419 才达到所要求的 90%，用时 4min
    b.n_clusters = n
    b.clustering_kmeans()    # kmeans
    time1 = time()
    b.confirm_type()         # confirm
    time2 = time()
    print(f"confirm cost time: {time2 - time1}s")
    flag = b.cal_business()  # business
```

```
    time3 = time()
    print(f"calculatet cost time: {time3 - time2}s")
    if flag:
        break
```

```
KMeans(n_clusters=100, random_state=10)
kmeans cost time: 21.72585892677307
确定新建基站类型
confirm cost time: 43.21766185760498s
计算新建基站覆盖的业务量
0
需要覆盖的业务量：3080394.7751342
新建基站覆盖的业务量：285833.464153
新建 / 需要：9.279117938399548%
新建 / 总量：4.050795672896914%
calculatet cost time: 17.47350525856018s
```

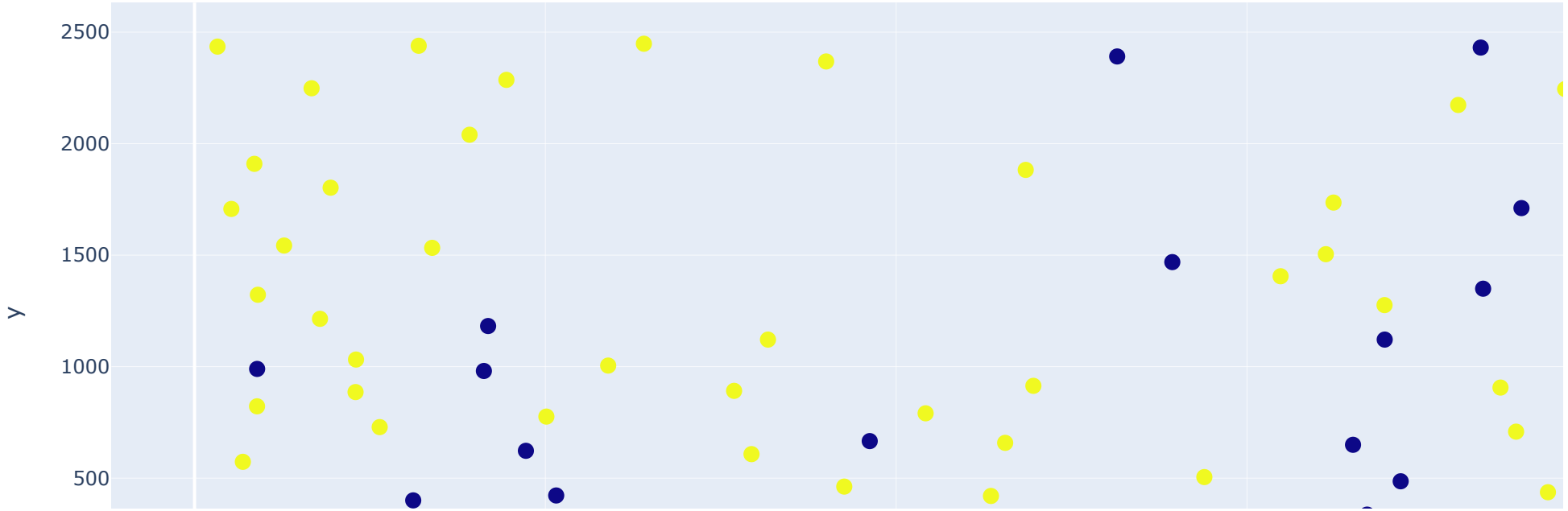
In [30]:

```
fig = b.plot_kmeans_res()
fig.update_layout(title='kmeans聚类(k=100)的结果')
fig.update_traces(marker={"size": 3})
fig.show()
```



In [31]:
```
new_base_site_type = b.cluster_centers_  # 中心点
remain_weak_grid_labels = b.remain_weak_grid_labels  # 保留的弱覆盖的聚类标签数据

new_base_site_type = pd.DataFrame(new_base_site_type, columns=['x', 'y', 'type'])
fig = px.scatter(data_frame=new_base_site_type, x='x', y='y', color='type')
fig.update_layout(title='新建基站散点图')
fig.update_traces(marker={"size": 10})
fig.show()
```

新建基站散点图



绘制三种基站（现有、弱覆盖、新建）散点图

In [32]:
```
# todo plot（done）

trace1 = go.Scatter(mode='markers', x=new_site_point_m2['x'], y=new_site_point_m2['y'],
                    name='新建基站', marker={"color":"red", "size":5})
trace2 = go.Scatter(mode='markers', x=exist_site_data['x'], y=exist_site_data['y'],
                    name='现有基站',marker={"color":"pink", "size":5})
trace3 = go.Scatter(mode='markers', x=weak_grid_data['x'], y=weak_grid_data['y'],
                    name='弱覆盖点',marker={"color":"cyan", "size":1})
data = [trace3, trace2, trace1]
fig = go.Figure(data=data)
fig.write_html(IMG_HTML / "question1-new_exist_weak_3kinds_sites.html")
```

```
fig.write_image(IMG_PNG / 'question1-new_exist_weak_3kinds_sites.png')
fig.write_image(IMG_SVG / 'question1-new_exist_weak_3kinds_sites.svg')
# fig.show()  # 不建议在 notebook 查看该图
del fig
```

这里文件太大，不方便展示，自行查看 `question1-new_exist_weak_3kinds_sites.html` 文件

其他算法/模型（在分析数据的时候写的）

对弱覆盖栅格数据进行 kmeans 聚类(k=100)

In [33]:
```
kmeans100 = KMeans(n_clusters=100)
kmeans100.fit(weak_grid_data.iloc[:, :2])
pd.DataFrame(kmeans100.cluster_centers_)
```

Out[33]:

	0	1
0	87.753616	1315.726240
1	1883.493878	708.474212
2	1986.039390	2405.806600
3	1040.561162	262.899083
4	259.168357	2149.452333
...	...	...
95	622.049133	769.579961
96	1154.357724	1824.552846
97	428.804945	1170.425824
98	759.863463	125.238667
99	1555.964988	1408.413084

100 rows × 2 columns

绘制 kmean 聚类(k=100)图所有点的散点图

In [34]:
```
weak_grid_data_cp = pd.concat([weak_grid_data.copy(), pd.DataFrame(kmeans100.labels_)], axis=1)
weak_grid_data_cp.rename(inplace=True, columns={0: "labels"})
weak_grid_data_cp
```

Out[34]:

	x	y	traffic	labels
0	66	1486	140.581390	85
1	67	1486	140.518829	85
2	177	1486	48.919178	85
3	187	1486	4.322495	85
4	284	1486	71.528404	46
...	...	...	...	...
182802	2350	2123	0.178571	84
182803	2353	2123	5.159708	84
182804	2354	2123	5.134017	84
182805	2355	2123	2.599999	84
182806	2372	2123	57.814999	84

182807 rows × 4 columns

In [35]:
```
fig = px.scatter(data_frame=weak_grid_data_cp, x='x', y='y', color='labels')
fig.update_traces(marker={"size": 3})
fig.write_html(IMG_HTML / 'question1-kmeans100.html')
fig.write_image(IMG_PNG / 'question1-kmeans100.png')
fig.write_image(IMG_SVG / 'question1-kmeans100.svg')
# fig.show()  # 不建议在 notebook 上运行该行代码
del fig
```

这里文件太大，不方便展示，自行查看 `question1-kmeans100.html` 文件

其他

In [36]:
```
# todo 这个忘了干啥用的了，好像是查看聚类和新建的数据？
newnew_site_data = pd.DataFrame(kmeans100.cluster_centers_, columns=["x", "y"])
nest_site_data = pd.read_csv(DATA_COOKED / "question1-best_base_site_points.csv", index_col=0)
total_site = pd.concat([newnew_site_data, nest_site_data])
total_site
```

Out[36]:

	x	y	type
0	87.753616	1315.726240	NaN
1	1883.493878	708.474212	NaN
2	1986.039390	2405.806600	NaN
3	1040.561162	262.899083	NaN
4	259.168357	2149.452333	NaN
...	...	...	...
338	709.000000	455.000000	1.0
339	1627.000000	2404.000000	1.0
340	819.000000	564.000000	1.0
341	814.000000	610.000000	1.0
342	1.000000	1814.000000	1.0

443 rows × 3 columns

对弱覆盖栅格数据进行 kmeans 聚类(k=343)

In [37]:

```
# todo cal 343 points and sort（done）
from sklearn.cluster import KMeans

kmeans343 = KMeans(n_clusters=343)
kmeans343.fit(weak_grid_data.iloc[:, :2])

weak_grid_data_cp = pd.concat([weak_grid_data.copy(), pd.DataFrame(kmeans343.labels_)], axis=1)
weak_grid_data_cp.rename(inplace=True, columns={0: "labels"})
weak_grid_data_cp

weak_grid_data_cp_size = []  # 343 个数，分别为 343 个类包含的点的个数
for name, group in weak_grid_data_cp.groupby(by='labels'):
    l = len(group)
    weak_grid_data_cp_size.append(l)
# print(len(weak_grid_data_cp_size))

weak_grid_data_cp_size = pd.DataFrame(weak_grid_data_cp_size).sort_values(by=0, ascending=False)
# print(weak_grid_data_cp_size.describe())
weak_grid_data_cp_size.T
```

Out[37]:

	95	22	113	298	17	178	222	133	207	186	...	305	38	338	163	55	42	232	152	255	336
0	1996	1822	1697	1679	1534	1444	1426	1398	1374	1369	...	97	90	90	78	71	71	67	62	62	48

1 rows × 343 columns

绘制 kmeans 聚类(k=343)所有点的散点图

In [38]:

```
weak_grid_data_cp = pd.concat([weak_grid_data.copy(), pd.DataFrame(kmeans343.labels_)], axis=1)
weak_grid_data_cp.rename(inplace=True, columns={0: "labels"})
weak_grid_data_cp
```

Out[38]:

	x	y	traffic	labels
0	66	1486	140.581390	340
1	67	1486	140.518829	340
2	177	1486	48.919178	102
3	187	1486	4.322495	102
4	284	1486	71.528404	228
...	...	...	...	...
182802	2350	2123	0.178571	60
182803	2353	2123	5.159708	60
182804	2354	2123	5.134017	60
182805	2355	2123	2.599999	60
182806	2372	2123	57.814999	60

182807 rows × 4 columns

In [39]:

```
fig = px.scatter(data_frame=weak_grid_data_cp, x='x', y='y', color='labels')
fig.update_traces(marker={"size": 3})
fig.write_html(IMG_HTML / 'question1-kmeans343.html')
fig.write_image(IMG_PNG / 'question1-kmeans343.png')
fig.write_image(IMG_SVG / 'question1-kmeans343.svg')
# fig.show()  # 不建议在 notebook 上运行该行代码
del fig
```

这里文件太大，不方便展示，自行查看 `question1-kmeans343.html` 文件

其他

In [40]:
```
# todo 这个忘了干啥用的了，好像是查看聚类和新建的数据？
newnew_site_data = pd.DataFrame(kmeans343.cluster_centers_, columns=["x", "y"])
nest_site_data = pd.read_csv(DATA_COOKED / "question1-best_base_site_points.csv", index_col=0)
total_site = pd.concat([newnew_site_data, nest_site_data])
total_site
```

Out[40]:

	x	y	type
0	1966.755000	2162.870833	NaN
1	508.121160	51.238055	NaN
2	1886.529510	713.724023	NaN
3	248.806950	1057.656371	NaN
4	1412.564994	316.087516	NaN
...	...	...	...
338	709.000000	455.000000	1.0
339	1627.000000	2404.000000	1.0
340	819.000000	564.000000	1.0
341	814.000000	610.000000	1.0
342	1.000000	1814.000000	1.0

686 rows × 3 columns

In [ ]:
```

```