

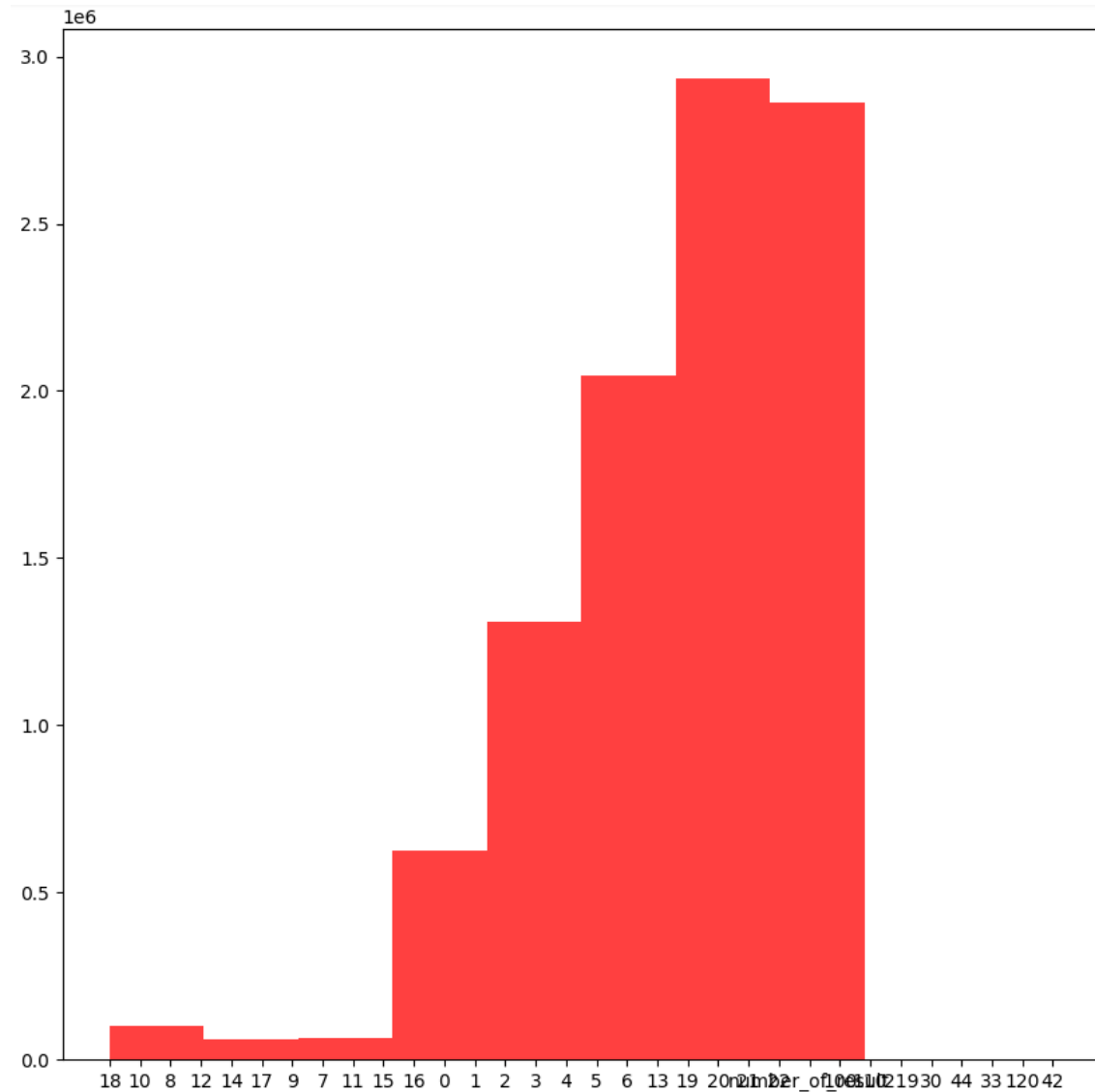
Class period 20

Histogram and render chart

ตัวอย่างข้อมูล wongnai.com

- ```
import matplotlib
matplotlib.rcParams['figure.figsize']=[15,15]
```
- ```
output =
plt.hist(list(data['number_of_result']),10,facecolor =
'red' ,alpha = 0.75)
```
- สร้างกราฟ Histogram โดยใช้ข้อมูลคอลัมน์ 'number_of_result' ในข้อมูล wongnai.com โดยกำหนดช่วงของข้อมูลหรือจำนวนแท่งเป็น 10 กำหนดสีเป็นสีแดง กำหนดค่าความโปร่งแสงเป็น 75%

ผลลัพธ์จะได้กราฟ Histogram ที่แกน x ที่เรียงข้อมูลผิด



แก้ไข แกน X ที่เรียงข้อมูลผิด

- 1. ตรวจสอบ **data type** ของ ตัวแปร ด้วย
 - `data.dtypes`
- 2. เรียกดูและตรวจสอบ **data type** ของตัวแปรในคอลัมน์ทีละตัวด้วย
 - `type(data['number_of_result'][0])`
- `type(data['number_of_result'][0]) == int`
- 3. ลองบังคับเปลี่ยน **type** ข้อมูลเป็น **int**
- `new_type = data['number_of_result'].astype('int32')`

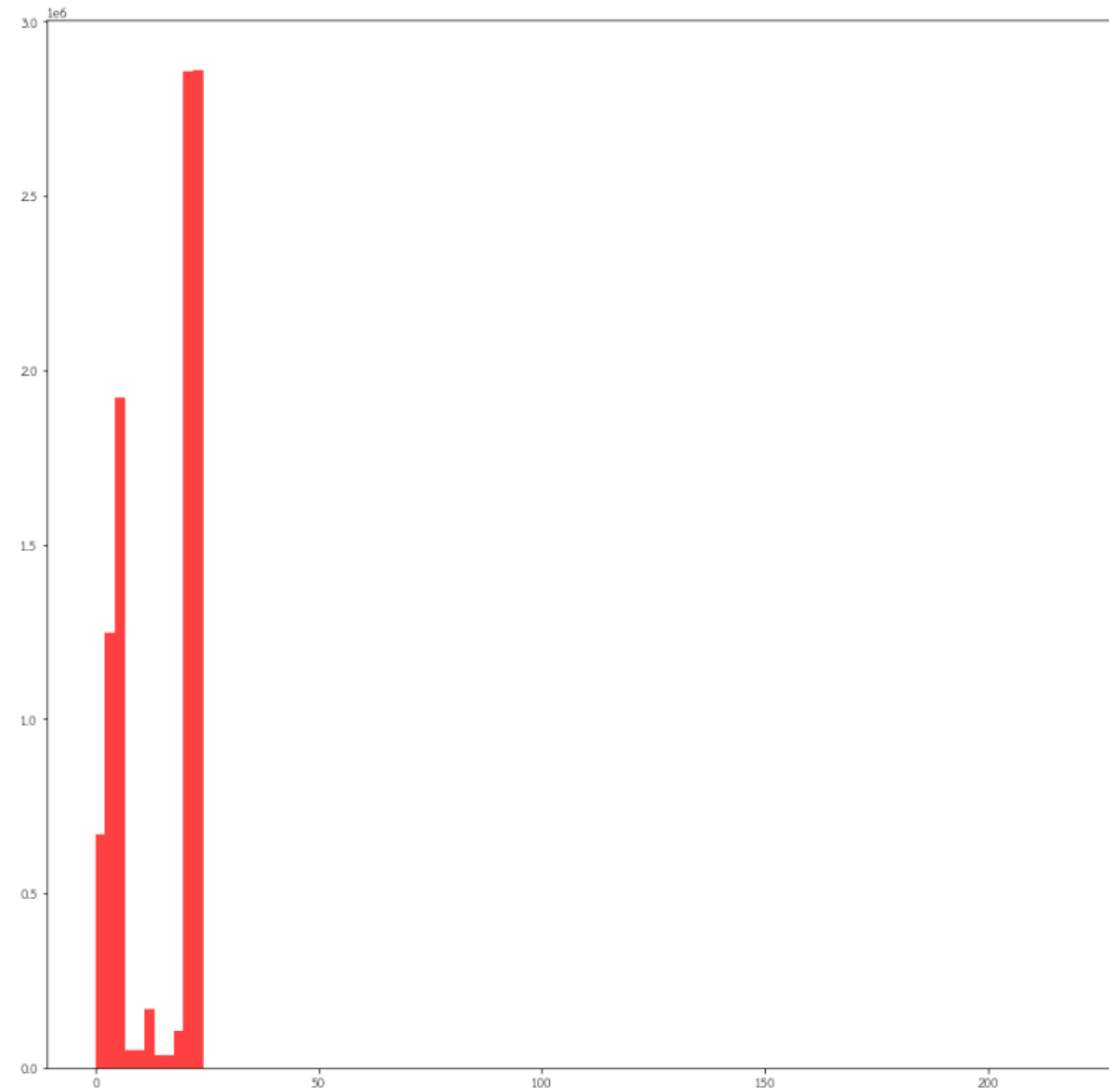
ลบ record

- จากการการบังคับเปลี่ยน **type** ข้อมูลเป็น **int** จะเจอ **Error** ว่ามีค่าในคอลัมน์ **'number_of_result'** ที่เป็น **number_of_result** ทำให้ไม่สามารถเปลี่ยน **type** ข้อมูลเป็น **int** ได้ ดังนั้นทำการใช้ **drop** ลบข้อมูลแถวที่มีค่าเป็น **number_of_result** ทิ้ง
- 1. ตรวจสอบว่า **record** ไ่บ้างที่มีค่าเป็น **number_of_result**
 - `data[data['number_of_result']=='number_of_result']`
 - ผลลัพธ์จะได้ **record** ที่ 1000016
- 2. ลบ **record** ที่มีค่าเป็น **number_of_result**
 - `data = data.drop(1000016)`

บังคับเปลี่ยน **type** ข้อมูลเป็น **int**

- บังคับเปลี่ยน **type** ข้อมูลเป็น **int** หลังจากลบข้อมูล **record** ที่ **1000016** แล้ว และเก็บข้อมูลที่แปลงแล้วไว้ในตัวแปร `new_type`
- `new_type = data['number_of_result'].astype('int32')`
- สร้างกราฟ **Histogram** ด้วยข้อมูลในตัวแปร `new_type`
- `output = plt.hist(new_type, 100, facecolor = 'red', alpha = 0.75)`

ผลลัพธ์จะได้กราฟ Histogram ที่มี outlier



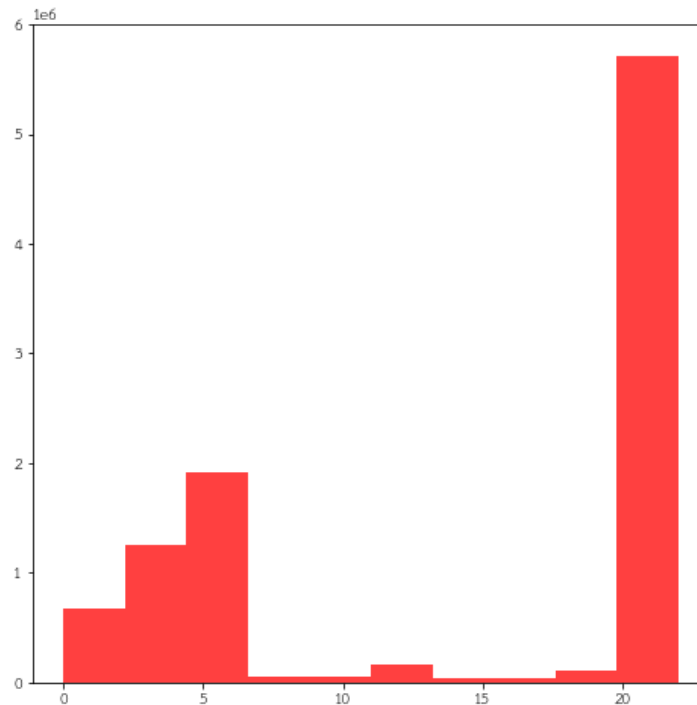
ลบ outlier

- ลบ **outlier** โดยการใส่เงื่อนไขให้กับเฉพาะข้อมูลที่มีค่าต่ำกว่า **25** ลงไป ไว้ในตัวแปรใหม่
- `new_type_nooutlier = new_type[new_type < 25]`
- ตรวจสอบว่าข้อมูลที่เป็น **outlier** มีกี่ตัว
- `new_type.shape[0] - new_type_nooutlier.shape[0]`
- ผลลัพธ์จะได้ว่ามี **outlier** ทั้งหมด **14** ตัว

```
new_type.shape[0] - new_type_nooutlier.shape[0]
```


สร้างกราฟ Histogram ด้วยข้อมูลในตัวแปร new_type_nooutlier

- `matplotlib.rcParams['figure.figsize']=[8,8]`
- `output = plt.hist(new_type_nooutlier,10,facecolor = 'red' ,alpha = 0.75)`



Quiz

- เปรียบเทียบความถี่ของแท่งที่มีค่ามากที่สุด กับ แท่งอื่นๆรวมกัน

เฉลย

- `output` ดูว่าค่าในแต่ละแท่งกราฟเป็นเท่าไร จากผลลัพธ์ที่ได้แท่งสุดท้ายคือ ค่าความถี่ที่มีค่ามากที่สุด 5717238

```
(array([ 670293., 1247269., 1921441.,   51703.,   50609., 167502.,
        36883.,   35914., 105490., 5717238.]),
 array([ 0. ,  2.2,  4.4,  6.6,  8.8, 11. , 13.2, 15.4, 17.6, 19.8, 22. ]),
 <a list of 10 Patch objects>)
```

- `output[0]` เลือกมาเฉพาะ **array** ตัวที่ **0** ของ `output` คือค่าความถี่ของแท่งกราฟ

```
array([ 670293., 1247269., 1921441.,   51703.,   50609., 167502.,
        36883.,   35914., 105490., 5717238.])
```

เฉลยต่อ

- `output[0][-1]` ค่าความถี่ของแท่งที่มีค่ามากที่สุด
- `5717238.0`

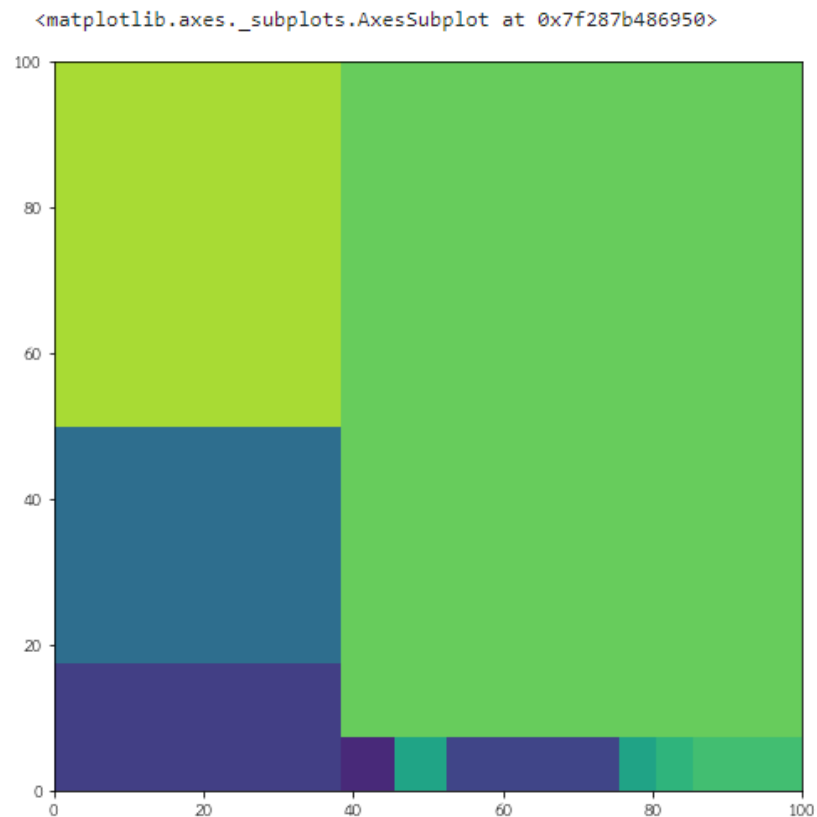
- `sum(output[0][: -1])` ค่าความถี่ของแท่งอื่นๆรวมกัน
- `4287104.0`

Tree map

- การนำปริมาณข้อมูลที่ต้องการมาเปรียบเทียบกันในรูปแบบพื้นที่ ต่างจากกราฟแท่งที่จะเปรียบเทียบความสูง
- สามารถใช้งานการสร้าง **Tree map** ด้วย **packet squarify** โดยจะต้อง **install packet** ก่อนใช้งานด้วยคำสั่ง
- `!pip install squarify`
- `import squarify`
- `import numpy as np`
- `import matplotlib.pyplot as plt`

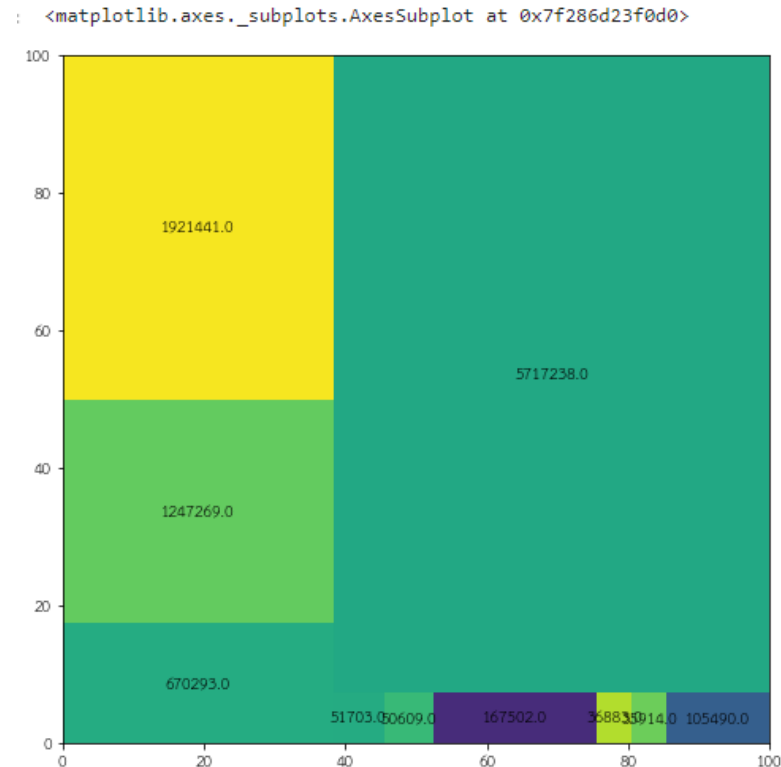
การใช้งาน squarify

- `squarify.plot('ตัวแปรข้อมูลที่ต้องการสร้าง tree map')` เช่น
- `squarify.plot(output[0])`



Parameter: value ของ squarify

- ใช้ Parameter: value เพื่อแสดงค่าในพื้นที่แต่ละพื้นที่ในกราฟ tree map เช่น
- `squarify.plot(output[0], value=output[0])`



Parameter: norm_y ของ squarify

- ใช้ Parameter: norm_y ในการเปลี่ยนรูปแบบของการจัดเรียงพื้นที่กราฟ tree map เพื่อให้ดูกราฟง่ายขึ้น เช่น
- `squarify.plot(output[0], value=output[0], norm_y=60)`

