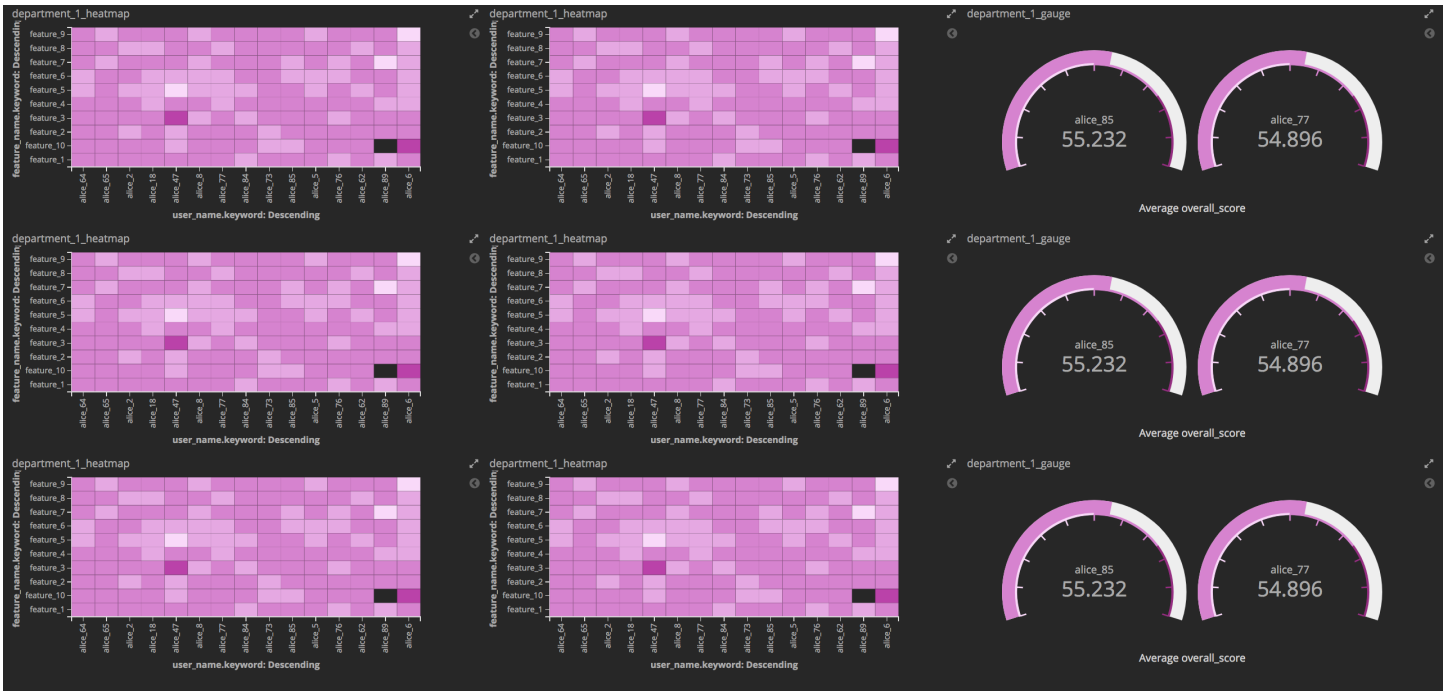
This is the code & real-time data generator folder for Graphen BOC ACMS project.

* Contact: kepu1994@outlook.com

==========

Example dashboard1:



Example dashboard2:

1. Prerequisites:
    elasticsearch 6.0.1
    kibana 6.0.1
    flask
    flask_cors

2. Architecture:
    realtime.py generates real-time data every one minute, and inserts new data in the index 'boc' in elasticsearch continuously. Each document inserted into 'boc' index is in the following format:

```
object = {
    "user_name": "alice1",
    "department_name": "sfdsf"
    "feature_name": "feature1",
    "total_events": 10,
    "upperbound_weekly_value": 10,
    "upperbound_daily_value": 10,
    "lowerbound_weekly_value": 10,
    "lowerbound_daily_value": 10,
    "feature_score": 98,
    "overall_score": 50,
    "timestamp": datetime.datetime.now()
}
```

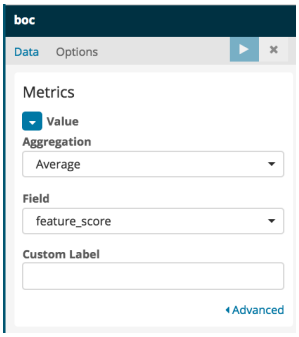
    /application folder is a flask folder, which serves the treemap&holtwinter page. server.py fetches realtime data from 'boc' index of elasticsearch and communicates with the front end js files(treemap.js and holtwinter.js)
    /application/static/js/treemap.js is the front-end controller communicating with the 'treemap' and 'click' endpoints in server.py. It renders the treemap and identifies which user is clicked, sending request to collect the info for the specific user and renders the user-specific holtwinters. It sends request recursively to the server.py every 10s to fetch the newest data.
    /application/static/js/holtwinter.js is the front-end controller communicating with the 'holtwinter' endpoint in server.py. It renders the top-dangerous user's holtwinters. It sends request recursively to the server.py every 10s to fetch the newest data.
    /application/static/js/echart.min.js is the echarts library.
    /application/static/templates/index.html is the html server.py, treemap.js, holtwinter.js serve.

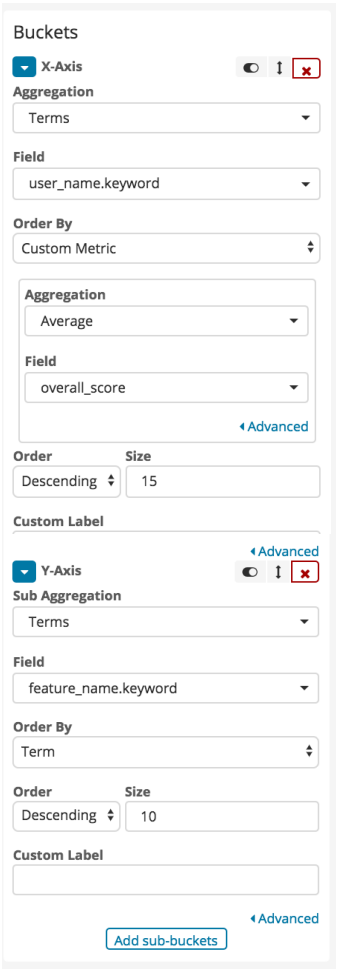
The project contains two dashboards, one using kibana, the other is served by the flask server. To run the project: run elasticsearch, run kibana, run realtime.py, run server,py. The kibana dashboard is on port 5601, the flask server is on port 9090.

3. Notice
    The server can be used with the actual data directly when the pipeline is established, if the new index 'boc' created in elasticsearch is in the exact format with the example showed in read color above. Also, several elasticsearch queries need to be added to make it work. The 'todo' query parts are commented with "TODO" in the server.py.
    The ports need to be changed in treemap.js, holtwinter.js, server.py when doing demos in BOC's screen. The addresses can be uncommented in those files.
    The other details about flask server and front end codes for dashboard2 can be found in the corresponding files' comments.

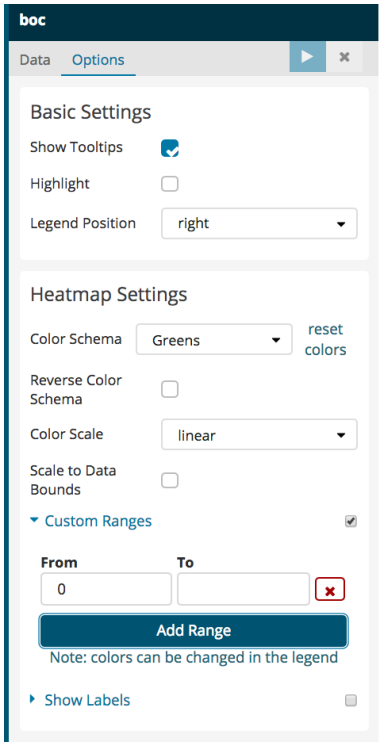
4. Dashboard1: Kibana configuration & tips

To config heatmap, save the visualization name as the department name if different department metric needs to be shown here. And also, for each separate department, we should have a index which has the same object with the overall index 'boc'. The exact configuration shows below:

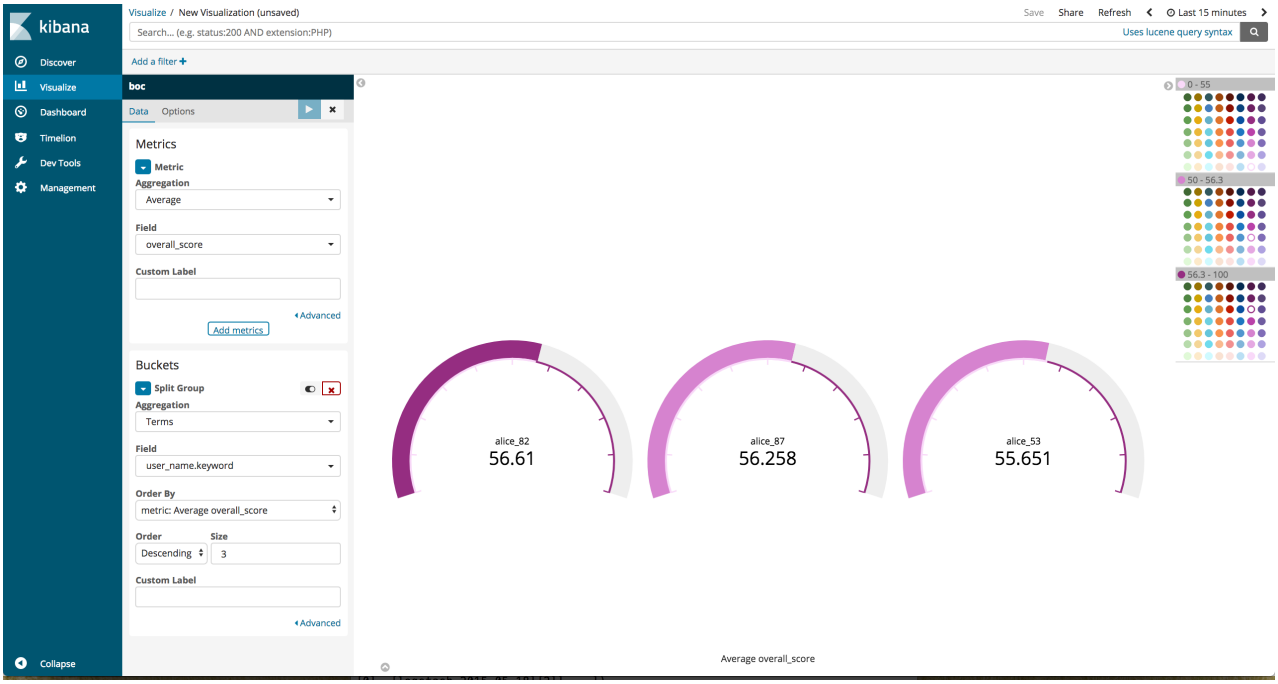




To customize the label ranges on the right:



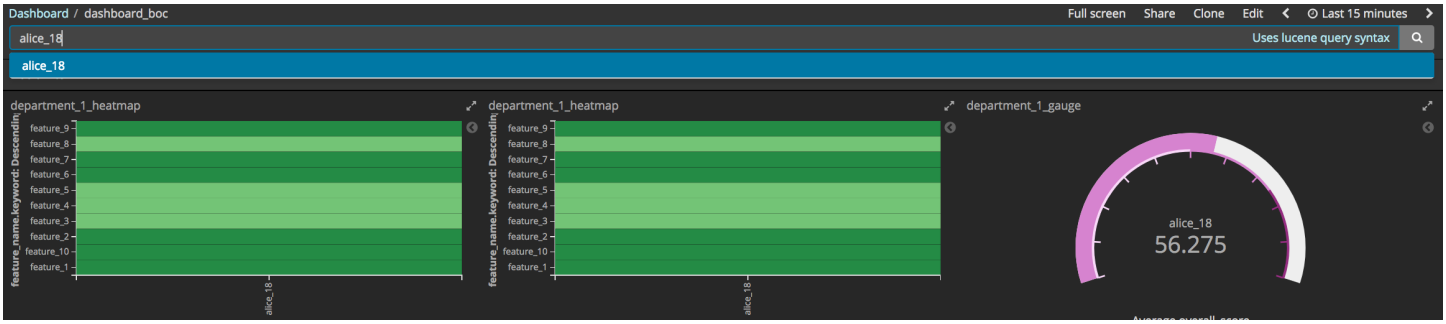
To config guage, for the top 3 users about overall_score:



Others:

Dark Theme:

Simple queries on kibana:



Timelion:

Timelion visualization type might be useful if we want to display changes along timeline: (choose visual type here). But if we intend to use that, I may suggest to create two dashboard, and put in the same html using <iframe>, because timelion uses accumulated data in different timestamps. And setting the ignore global filter to “Yes”.