Advance Data Science

Missing Data Analysis - Edgar Data

Assignment – 1 Part - 2

Report by:
Ann Sara Sajee
Gunjan Ratan Lalwani
Rishabh Jain

Goal:

The report summarizes the design and implementation of the analysis performed on the Edger log files. This report is divided into two sections:

1. Fetching and Analysis of Edgar logs file Data Set
2. Handling Missing Data
3. Compute summary metric for Edgar Log files
4. Log all the operations (with time stamps) into a log file
5. Dockerize the process using Docker file and Docker Hub
6. Create the Tableau representation of the analysis performed on the log

1. Fetching and analysis of EDGAR log file Data set:

The EDGAR Log File Data Set contains information in CSV format extracted from Apache log files that record and store user access statistics for the SEC.gov website. These log are captured on a daily basis and are stored in a zip format on Edgar website.
Edgar log files consist of the following columns:

Fetching Dataset:
The program takes year as the parameter from the user. It then outputs all the log files from the 1st day of every month. The years range from 2013 to 2017 and if user enters anything not valid, the program as for valid data.
On valid data, the program will fetch the url from Edgar dataset which consists of all the log files.

The log file consists of:

- CIK No: Every company has a CIK No for filing purpose
- Accession No: Every file has an accession no
- Extension: consist of the file that is requested by supplying CIK and accession number
- Code: implies the response code from the server
- Date: log file creation date

Analysis of Log file:

- Multiple missing values in the field of browser and size
- Many extensions incomplete
- CIK No length is max 10
- For Code 304, all files are of size 0

	ip	date	time	zone	cik	accession	extention	code	size	idx	norefer	noagent	find	crawler	browser
0	66.245.120.dfg	2003-11-01	00:01:42	500.0	774695.0	0001228027-03-000004	-index.htm	200.0	2555.0	1.0	0.0	0.0	1.0	0.0	win
1	66.245.120.dfg	2003-11-01	00:01:47	500.0	774695.0	0001228027-03-000004	xslF345X02/edgardoc.xml	200.0	15502.0	0.0	0.0	0.0	9.0	0.0	win
2	24.80.4.efj	2003-11-01	00:01:50	500.0	933136.0	0000891020-98-000348	.txt	200.0	1156287.0	0.0	0.0	0.0	1.0	0.0	win
3	164.164.89.djf	2003-11-01	00:01:55	500.0	75677.0	0001104659-03-024394	-index.htm	200.0	2504.0	1.0	0.0	0.0	1.0	0.0	win
4	164.164.89.djf	2003-11-01	00:02:00	500.0	75677.0	0001104659-03-024374	-index.htm	200.0	2504.0	1.0	0.0	0.0	1.0	0.0	win
5	164.164.89.djf	2003-11-01	00:02:06	500.0	75677.0	0001104659-03-024374	xslF345X02/a4.xml	200.0	36263.0	0.0	0.0	0.0	9.0	0.0	win

2. **<u>Handing missing values:</u>**
   1. **Download the extracted log files and merge it into one zip file:**

```
import csv
import os
import zipfile
import pandas as pd
import numpy as np
import glob
from bs4 import BeautifulSoup
from io import BytesIO
from urllib.request import urlopen
from zipfile import ZipFile

def assure_path_exists(path):
        if not os.path.exists(path):
                os.makedirs(path)

year="2003"
url='https://www.sec.gov/data/edgar-log-file-data-set.html'
html=urlopen(url)
soup = BeautifulSoup(html, "html.parser")

all_div=soup.findAll("div", attrs={'id':'asyncAccordion'})
for div in all_div:
    h2tag = div.findAll("a")
    for a in h2tag:
        if str(year) in a.get('href'):
            global ahref
            ahref=a.get('href')
linkurl='https://www.sec.gov'+ahref
linkhtml=urlopen(linkurl)
```

```
allzipfiles=BeautifulSoup(linkhtml, "html.parser")

ziplist=allzipfiles.find_all('li')

monthlistdata=[]
count=0
for li in ziplist:
    zipatags=li.findAll('a')
    for zipa in zipatags:
        if "01.zip" in zipa.text:
            monthlistdata.append(zipa.get('href'))
print(len(monthlistdata))

foldername=str(year)
path=str(os.getcwd())+"\\"+foldername

assure_path_exists(path)

for month in monthlistdata:
    with urlopen(month) as zipresp:
        with ZipFile(BytesIO(zipresp.read())) as zfile:
            zfile.extractall(path)
```

12

2. Load log files into pandas data frame:

```
In [5]: #Create a data frame for all 12 log files
        all_data = pd.DataFrame()
        for f in glob.glob(path +'//log20031101.csv'):
            df = pd.read_csv(f,parse_dates=[1])
            all_data = all_data.append(df,ignore_index=False)
        all_data
```

Out[5]:

	ip	date	time	zone	cik	accession	extention	code	size	idx	norefer	noagent	find	cr
0	66.245.120.dfg	2003-11-01	00:01:42	500.0	774695.0	0001228027-03-000004	-index.htm	200.0	2555.0	1.0	0.0	0.0	1.0	0.(
1	66.245.120.dfg	2003-11-01	00:01:47	500.0	774695.0	0001228027-03-000004	xslF345X02/edgardoc.xml	200.0	15502.0	0.0	0.0	0.0	9.0	0.(
2	24.80.4.efj	2003-11-01	00:01:50	500.0	933136.0	0000891020-98-000348	.txt	200.0	1156287.0	0.0	0.0	0.0	1.0	0.(
3	164.164.89.djf	2003-11-01	00:01:55	500.0	75677.0	0001104659-03-024394	-index.htm	200.0	2504.0	1.0	0.0	0.0	1.0	0.(
4	164.164.89.djf	2003-11-01	00:02:00	500.0	75677.0	0001104659-03-024374	-index.htm	200.0	2504.0	1.0	0.0	0.0	1.0	0.(
5	164.164.89.djf	2003-11-01	00:02:06	500.0	75677.0	0001104659-03-024374	xslF345X02/a4.xml	200.0	36263.0	0.0	0.0	0.0	9.0	0.(

3. Change data types of columns in the data frame:

```
all_data.dtypes
```

```
ip                  object
date        datetime64[ns]
time                object
zone               float64
cik                float64
accession           object
extention           object
code               float64
size               float64
idx                float64
norefer            float64
noagent            float64
find               float64
crawler            float64
browser             object
dtype: object
```

```
In [6]: #Change the data types for all column
        all_data['zone'] = all_data['zone'].astype('int64')
        all_data['cik'] = all_data['cik'].astype('int64')
        all_data['code'] = all_data['code'].astype('int64')
        all_data['idx'] = all_data['idx'].astype('int64')
        all_data['noagent'] = all_data['noagent'].astype('int64')
        all_data['norefer'] = all_data['norefer'].astype('int64')
        all_data['crawler'] = all_data['crawler'].astype('int64')
        all_data['find'] = all_data['find'].astype('int64')
```

```
all_data.dtypes
```

```
ip                   object
date        datetime64[ns]
time                 object
zone                  int64
cik                   int64
accession            object
extention            object
code                  int64
size                float64
idx                   int64
norefer               int64
noagent               int64
find                  int64
crawler               int64
browser              object
dtype: object
```

4. Change extension column to accession no:

```
In [7]: #changing the NaN value with the mean or most frequent occuring value
        all_data.loc[all_data['extention']=='.txt', 'extention'] = all_data["accession"].map(str) + all_data["extention"]
        all_data.loc[all_data['extention']=='.paper', 'extention'] = all_data["accession"].map(str) + all_data["extention"]
        all_data['browser']=all_data['browser'].fillna('missing')
        all_data
```

Out[7]:

	ip	date	time	zone	cik	accession	extention	code	size	idx	norefer	noagent	find	craw
0	66.245.120.dfg	2003-11-01	00:01:42	500	774695	0001228027-03-000004	-index.htm	200	2555.0	1	0	0	1	0
1	66.245.120.dfg	2003-11-01	00:01:47	500	774695	0001228027-03-000004	xslF345X02/edgardoc.xml	200	15502.0	0	0	0	9	0
2	24.80.4.ef)	2003-11-01	00:01:50	500	933136	0000891020-98-000348	0000891020-98-000348.txt	200	1156287.0	0	0	0	1	0
3	164.164.89.djf	2003-11-01	00:01:55	500	75677	0001104659-03-024394	-index.htm	200	2504.0	1	0	0	1	0
4	164.164.89.djf	2003-11-01	00:02:00	500	75677	0001104659-03-024374	-index.htm	200	2504.0	1	0	0	1	0
5	164.164.89.djf	2003-11-01	00:02:06	500	75677	0001104659-03-024374	xslF345X02/a4.xml	200	36263.0	0	0	0	9	0

5. Replace NaN values with appropriate value:

```
IPython console
Console 2/A
INFO - NaN values in browser replaced with maximum count browser.
START
INFO - Rows removed where ip, date, time, cik or accession were null.
INFO - NaN values in idx replaced with maximum idx.
INFO - NaN values in code replaced with maximum code.
INFO - NaN values in norefer replaced
INFO - NaN values in noagent replaced
INFO - NaN values in find replaced with minimum find.
INFO - NaN values in crawler replaced
INFO - NaN values in extension replaced with maximum extension.
INFO - NaN values in size replaced with mean value of size.
INFO - NaN values in zone replaced with maximum zone.
INFO -                ip        date      time   zone       cik  \
0  129.110.39.jca  2003-03-01  00:00:00  500.0   97349.0
1   61.115.76.jbf  2003-03-01  00:00:00  500.0  766351.0
2   61.115.76.jbf  2003-03-01  00:00:01  500.0  902584.0
3   61.115.76.jbf  2003-03-01  00:00:03  500.0  778207.0
4  129.110.39.jca  2003-03-01  00:00:07  500.0   97349.0

              accession   extention   code      size  idx  norefer  noagent  \
0  0000097349-01-000006   -0002.txt  200.0    3726.0  0.0      0.0      0.0
1  0000950134-03-003149        .txt  200.0  995957.0  0.0      1.0      0.0
2  0000902584-03-000044        .txt  200.0   15520.0  0.0      1.0      0.0
3  9999999997-03-006003        .txt  200.0    1670.0  0.0      1.0      0.0
4  0000097349-01-000006  -index.htm  200.0    4331.0  1.0      0.0      0.0

   find  crawler browser
0   9.0      0.0     win
IPython console   History log
Permissions: RW   End-of-lines: LF   Encoding: ASCII   Line: 15   Column: 1   Memory: 68 %
```

3. Compute the summary matric of the Edgar log file:

```
In [10]: # Create a summary that groups ip by date
         all_data['ip'].groupby(all_data['date']).describe()

Out[10]: date
         2003-11-01  count                10330
                     unique                1877
                     top        141.219.154.jie
                     freq                 1502
         Name: ip, dtype: object
```

```
In [11]: #get Top 10 count of all cik with their accession number
         all_data['COUNT'] =1 #initially, set that counter to 1.
         group_data = all_data.groupby(['date','cik','accession'])['COUNT'].count() #sum function
         group_data.rank().head(10)

Out[11]: date        cik   accession
         2003-11-01  20    0000893220-97-001715    2462.0
                     1750  0000001750-03-000067    6657.0
                     1800  0000912057-02-040993    5631.0
                           0001104659-03-022125    2462.0
                     1985  0000914039-01-500105    2462.0
                     2024  0000002024-00-000008    5631.0
                           0000002024-03-000002    6466.5
                           0000002024-99-000004    5631.0
                     2034  0000002034-97-000009    2462.0
                     2062  0001238603-03-000002    2462.0
         Name: COUNT, dtype: float64
```

```
In [19]: all_data['browser'].groupby(all_data['ip']).describe()

Out[19]: ip
         12.0.36.fgd      count       4
                          unique      1
                          top       win
                          freq        4
         12.102.126.cje   count       2
                          unique      1
                          top       win
                          freq        2
         12.105.174.jad   count       0
                          unique      0
         12.106.229.eja   count       2
                          unique      1
                          top       win
                          freq        2
         12.107.111.ghe   count       2
                          unique      1
                          top       win
                          freq        2
         12.108.119.ahb   count       1
                          unique      1
                          top       mie
                          freq        1
         12.108.233.igh   count       1
                          unique      1
                          top       mie
                          freq        1
         12.109.135.bji   count      13
                          unique      1
                          top       mie
                          freq       13
```

```
In [13]: all_data['COUNT'] =1
         status = all_data.groupby(['code']).count() #sum function
         status['COUNT']

Out[13]: code
         200    9136
         206      56
         302       6
         304    1105
         400       1
         404      26
         Name: COUNT, dtype: int64
```

```
In [14]: # Create a summary that groups ip by date
         all_data['extention'].groupby(all_data['cik']).describe()

Out[14]: cik
         1000184  count                          2
                  unique                         2
                  top                  -index.html
                  freq                           1
         1000228  count                          2
                  unique                         2
                  top                   -index.htm
                  freq                           1
         1000230  count                          1
                  unique                         1
                  top       0000916641-03-001229.txt
                  freq                           1
         1000297  count                          2
                  unique                         2
                  top       0001104659-03-024022.txt
                  freq                           1
         1000298  count                          1
                  unique                         1
                  top       0001000298-97-000008.txt
                  freq                           1
         100030   count                          1
                  unique                         1
                  top       0000950152-97-005620.txt
                  freq                           1
```

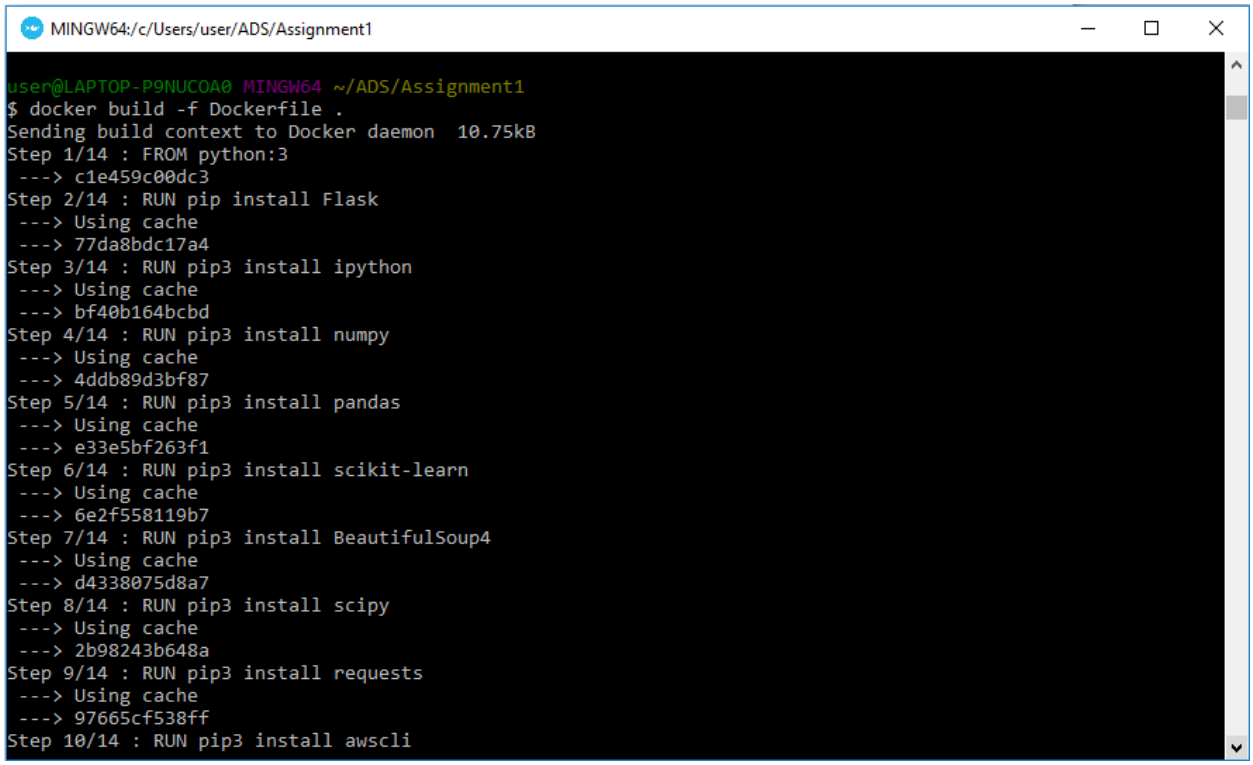
4. Log all the operations into a log file:

```
Console 2/A
Enter a year:2003
INFO - Directories cleanup completed
https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030101.zip
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030101.zip HTTP/1.1" 200 4515
INFO -  Downloaded Log file https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030101.zip for First date of month.
https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030201.zip
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030201.zip HTTP/1.1" 200 4515
INFO -  Downloaded Log file https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030201.zip for First date of month.
https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030301.zip
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030301.zip HTTP/1.1" 200 821853
INFO -  Downloaded Log file https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030301.zip for First date of month.
https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030401.zip
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030401.zip HTTP/1.1" 200 4906870
INFO -  Downloaded Log file https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030401.zip for First date of month.
https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030501.zip
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030501.zip HTTP/1.1" 200 5358624
INFO -  Downloaded Log file https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030501.zip for First date of month.
https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030601.zip
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030601.zip HTTP/1.1" 200 1899660
INFO -  Downloaded Log file https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030601.zip for First date of month.
https://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr3/log20030701.zip
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /dera/data/Public-EDGAR-log-file-data/2003/Qtr3/log20030701.zip HTTP/1.1" 200 5453646
IPython console   Variable explorer   File explorer   Help
```

```
INFO - NaN values in extension replaced with maximum extension.
INFO - NaN values in size replaced with mean value of size.
INFO - NaN values in zone replaced with maximum zone.
INFO -                ip        date      time   zone       cik  \
0  129.110.39.jca  2003-03-01  00:00:00  500.0   97349.0
1   61.115.76.jbf  2003-03-01  00:00:00  500.0  766351.0
2   61.115.76.jbf  2003-03-01  00:00:01  500.0  902584.0
3   61.115.76.jbf  2003-03-01  00:00:03  500.0  778207.0
4  129.110.39.jca  2003-03-01  00:00:07  500.0   97349.0

              accession   extention   code      size  idx  norefer  noagent  \
0  0000097349-01-000006   -0002.txt  200.0    3726.0  0.0      0.0      0.0
1  0000950134-03-003149        .txt  200.0  995957.0  0.0      1.0      0.0
2  0000902584-03-000044        .txt  200.0   15520.0  0.0      1.0      0.0
3  9999999997-03-006003        .txt  200.0    1670.0  0.0      1.0      0.0
4  0000097349-01-000006  -index.htm  200.0    4331.0  1.0      0.0      0.0

   find  crawler browser
0   9.0      0.0     win
1   0.0      0.0     NAV
2   0.0      0.0     NAV
3   0.0      0.0     NAV
4   1.0      0.0     win
INFO - Start of conversion of csv files
INFO - All dataframes of csvs are combined and exported as csv: master_csv.csv.
INFO - Compiled csv and log file zipped

In [3]:
IPython console   Variable explorer   File explorer   Help
```

5. **<u>Dockerize the process using Docker file and Docker Hub:</u>**
   1. **Build the docker file to create image:**

Building this dockerfile will create a DOCKER image with name annsara95/a1p1docker. Each step in this file will be executed in order to create an image. AWS CLI is also integrated inside the dockerfile



```
MINGW64:/c/Users/user/ADS/Assignment1

user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ docker build -f Dockerfile .
Sending build context to Docker daemon  10.75kB
Step 1/14 : FROM python:3
 ---> c1e459c00dc3
Step 2/14 : RUN pip install Flask
 ---> Using cache
 ---> 77da8bdc17a4
Step 3/14 : RUN pip3 install ipython
 ---> Using cache
 ---> bf40b164bcbd
Step 4/14 : RUN pip3 install numpy
 ---> Using cache
 ---> 4ddb89d3bf87
Step 5/14 : RUN pip3 install pandas
 ---> Using cache
 ---> e33e5bf263f1
Step 6/14 : RUN pip3 install scikit-learn
 ---> Using cache
 ---> 6e2f558119b7
Step 7/14 : RUN pip3 install BeautifulSoup4
 ---> Using cache
 ---> d4338075d8a7
Step 8/14 : RUN pip3 install scipy
 ---> Using cache
 ---> 2b98243b648a
Step 9/14 : RUN pip3 install requests
 ---> Using cache
 ---> 97665cf538ff
Step 10/14 : RUN pip3 install awscli
```

```
MINGW64:/c/Users/user/ADS/Assignment1
 ---> 6e2f558119b7
Step 7/14 : RUN pip3 install BeautifulSoup4
 ---> Using cache
 ---> d4338075d8a7
Step 8/14 : RUN pip3 install scipy
 ---> Using cache
 ---> 2b98243b648a
Step 9/14 : RUN pip3 install requests
 ---> Using cache
 ---> 97665cf538ff
Step 10/14 : RUN pip3 install awscli
 ---> Using cache
 ---> bb34e3aa3de0
Step 11/14 : RUN pip3 install luigi
 ---> Using cache
 ---> 532b01da1ce1
Step 12/14 : COPY edgarDataScrapping.py /src/edgarDataScrapping.py
 ---> 78abfc7b0fef
Step 13/14 : EXPOSE  8000
 ---> Running in d87752771306
Removing intermediate container d87752771306
 ---> 07a72bd8a9ad
Step 14/14 : CMD ["python", "/src/edgarDataScrapping.py", "-p 8000"]
 ---> Running in 576d15131fa1
Removing intermediate container 576d15131fa1
 ---> 6958311e00a8
Successfully built 6958311e00a8
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files
and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check
 and reset permissions for sensitive files and directories.
```

2. Create Docker image:

```
Select MINGW64:/c/Users/user/ADS/Assignment1
 ---> 07a72bd8a9ad
Step 14/14 : CMD ["python", "/src/edgarDataScrapping.py", "-p 8000"]
 ---> Running in 576d15131fa1
Removing intermediate container 576d15131fa1
 ---> 6958311e00a8
Successfully built 6958311e00a8
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and
 directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and r
eset permissions for sensitive files and directories.

user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ docker images
REPOSITORY                        TAG                 IMAGE ID            CREATED             SIZE
<none>                            <none>              6958311e00a8        51 seconds ago      1.32GB
lalwanigunjan/edgardatascrapping  latest              4dc37741e0c2        34 hours ago        1.31GB
annsara95/annsara95               latest              bfea8fe2f54d        4 days ago          53.1MB
alpine                            3.1                 2ba97bb89407        5 weeks ago         5.05MB
python                            3                   c1e459c00dc3        8 weeks ago         692MB
hello-world                       latest              f2a91732366c        2 months ago        1.85kB

user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to
https://hub.docker.com to create one.
Username (annsara95): annsara95
Password:
Login Succeeded
```

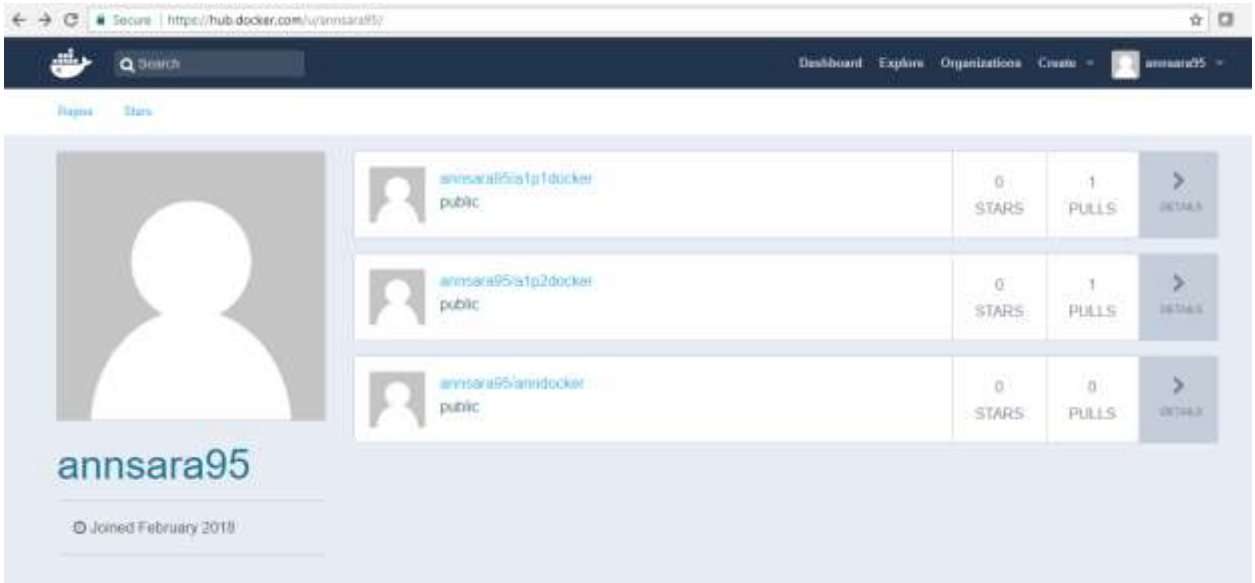
```
Select MINGW64:/c/Users/user/ADS/Assignment1
https://hub.docker.com to create one.
Username (annsara95): annsara95
Password:
Login Succeeded

user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ docker tag 6958311e00a8 annsara95/a1p1docker:latest
5a11f4418d8f: Pushed
40b5d14b099f: Pushed
40b5d14b099f: Pushing   5.53MB/45.13MB
96c2a0dc1139: Pushed
1fe855e92157: Preparing
76980e1c9e69: Pushed
c7012de491be: Pushed
d15eaaf669ad: Pushed
1a0128e38468: Pushed
262058ea6760: Pushed
6dce5c484bde: Mounted from library/python
057c34df1f1a: Mounted from library/python
3d358bf2f209: Mounted from library/python
0870b36b7599: Mounted from library/python
8fe6d5dcea45: Mounted from library/python
06b8d020c11b: Mounted from library/python
b9914afd042f: Mounted from library/python
4bcdffd70da2: Mounted from library/python
latest: digest: sha256:752087d46acd74e7fe88db228553dbfa460bf74a385ad06d2e9214b47d1a903a size: 4330

user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ _
```

3. **Push docker image to the hub:**

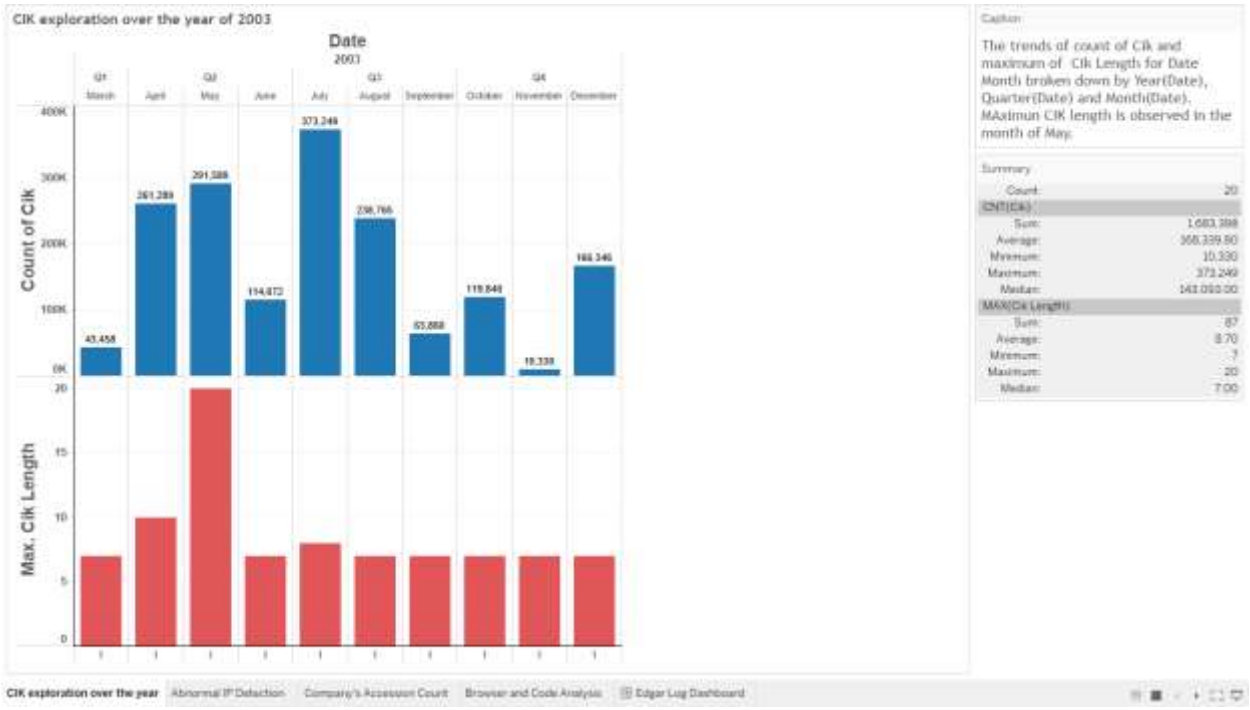
```
MINGW64:/c/Users/user/ADS/Assignment1
user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ docker tag ac1d00d2a444 annsara95/a1p2docker:latest

user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ docker push annsara95/a1p2docker
The push refers to repository [docker.io/annsara95/a1p2docker]
72693cbc5220: Pushed
5a11f4418d8f: Mounted from annsara95/a1p1docker
40b5d14b099f: Mounted from annsara95/a1p1docker
3197768ec91: Mounted from annsara95/a1p1docker
96c2a0dc1139: Mounted from annsara95/a1p1docker
218dc16cadf5: Mounted from annsara95/a1p1docker
76980e1c9e69: Mounted from annsara95/a1p1docker
c7012de491be: Mounted from annsara95/a1p1docker
d15eaaf669ad: Mounted from annsara95/a1p1docker
1a0128e38468: Mounted from annsara95/a1p1docker
262058ea6760: Mounted from annsara95/a1p1docker
6dce5c484bde: Mounted from annsara95/a1p1docker
057c34df1f1a: Mounted from annsara95/a1p1docker
3d358bf2f209: Mounted from annsara95/a1p1docker
0870b36b7599: Mounted from annsara95/a1p1docker
8fe6d5dcea45: Mounted from annsara95/a1p1docker
06b8d020c11b: Mounted from annsara95/a1p1docker
b9914afd042f: Mounted from annsara95/a1p1docker
4bcdffd70da2: Mounted from annsara95/a1p1docker
```

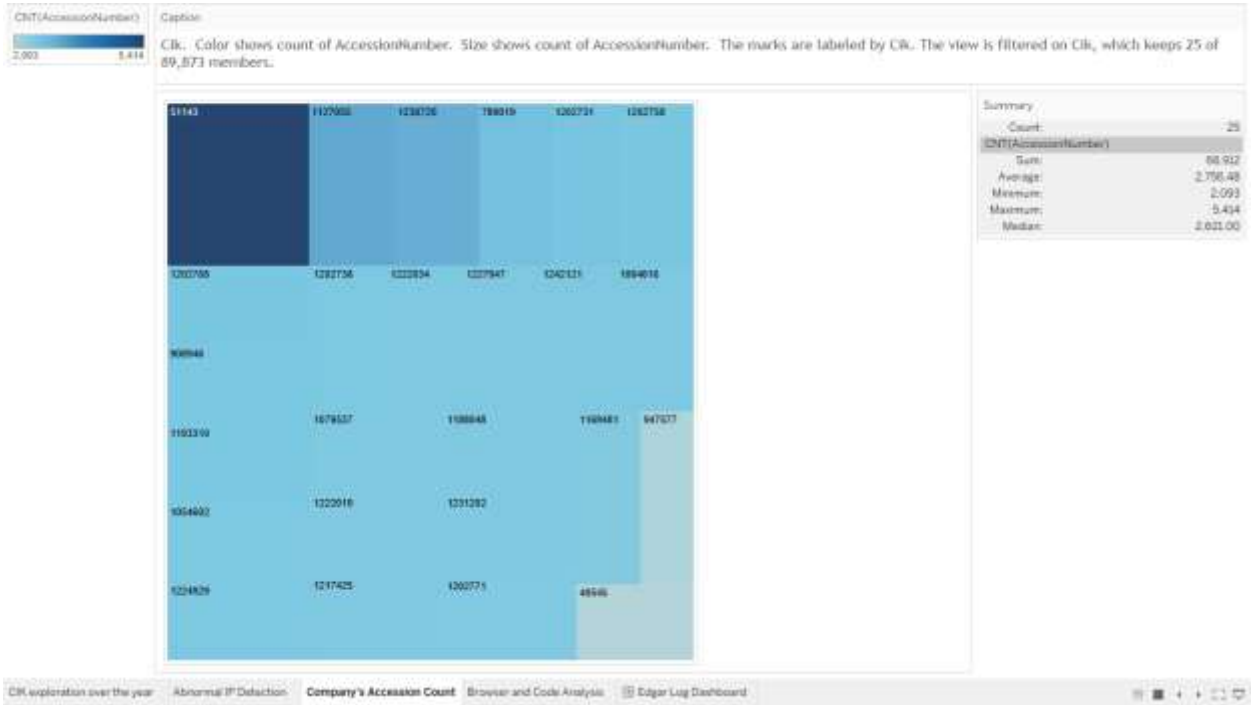
4. **Run docker image:**

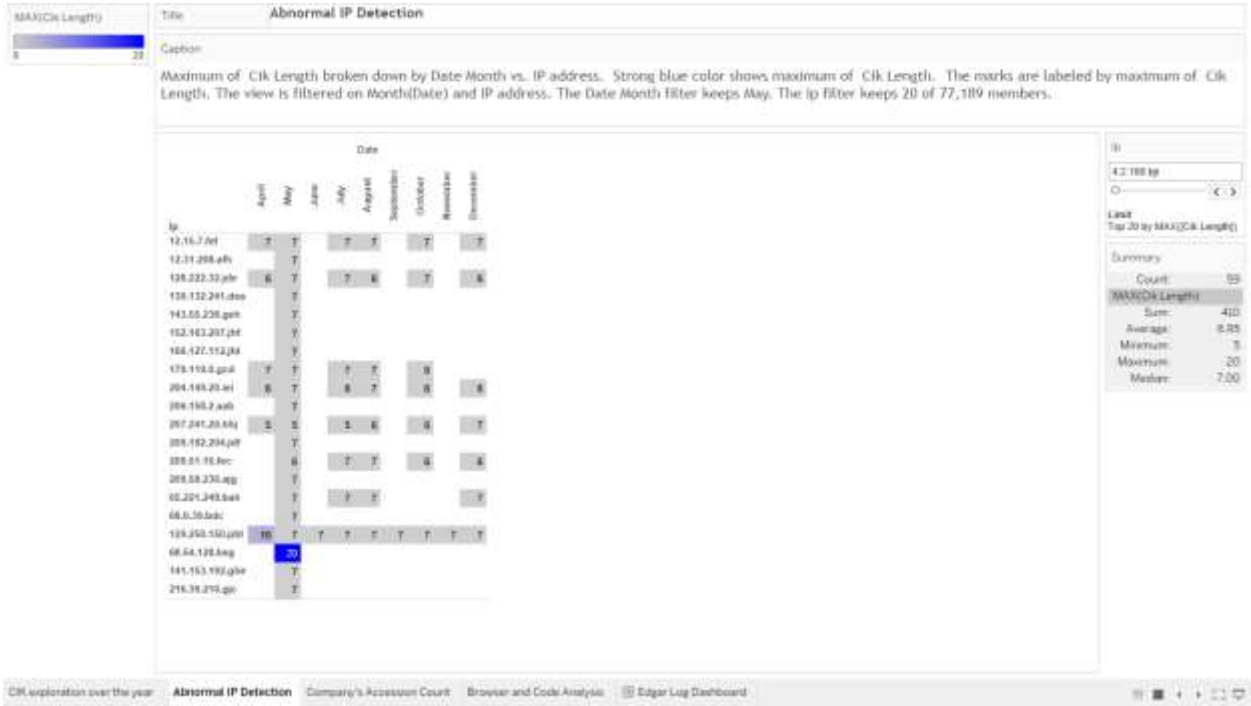
```
MINGW64:/c/Users/user/ADS/Assignment1
user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ docker run -d -p 8000:8000 annsara95/a1p1docker python /src/edgarDataScrapping.py -p 8000
3676d192679da47f79a1197b957ad5924eb40b9d5dda443e1f4bfc8ace6c643b

user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ docker ps -l
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS                     PORTS   NAMES
3676d192679d   annsara95/a1p1docker  "python /src/edgarDa"    22 seconds ago   Exited (0) 17 seconds ago          hardcore_shirley

user@LAPTOP-P9NUCOA0 MINGW64 ~/ADS/Assignment1
$ _
```



6. **<u>Create the Tableau representation of the analysis performed on the log:</u>**




CNT(AccessionNumber)
Caption
Cik. Color shows count of AccessionNumber. Size shows count of AccessionNumber. The marks are labeled by Cik. The view is filtered on Cik, which keeps 25 of 89,873 members.
Summary
Count: 25
CNT(AccessionNumber)
Sum: 68,912
Average: 2,756.48
Minimum: 2,093
Maximum: 5,414
Median: 2,631.00
CIK exploration over the year
Abnormal IP Detection
Company's Accession Count
Browser and Code Analysis
Edgar Log Dashboard


MAX(Cik Length)
Title
Abnormal IP Detection
Caption
Maximum of Cik Length broken down by Date Month vs. IP address. Strong blue color shows maximum of Cik Length. The marks are labeled by maximum of Cik Length. The view is filtered on Month(Date) and IP address. The Date Month filter keeps May. The ip filter keeps 20 of 77,189 members.
Date
April
May
June
July
August
September
October
November
December
ip
Summary
Count: 59
MAX(Cik Length)
Sum: 410
Average: 6.95
Minimum: 5
Maximum: 20
Median: 7.00
CIK exploration over the year
Abnormal IP Detection
Company's Accession Count
Browser and Code Analysis
Edgar Log Dashboard

Caption

Count of Size (color) broken down by Code vs. Browser. The data is filtered on Action (YEAR(Date),QUARTER(Date),MONTH(Date)), which keeps 10 members.

Browser and Code Analysis

Browser

Code	Windows*..	Missing	MS Internet Explorer	Mac	Linux	Opera	IEMobile|..	Safari
200	756,917	530,132	179,147	1,719	1,231	583	13	10
304	147,385	24,640	21,908	124	102	77	2	
206	6,780	894	1,336	2	2		2	
302	524	5,354	62					
416		2,617						
404	640	634	531	2	9			
400	1	14						
403		5						

Summary

Count:	33
CNT(Size)	
Sum:	1,683,399
Average:	51,012.09
Minimum:	1
Maximum:	756,917
Median:	531.00

Dashboard: