**Assignment 1 Report**

**Problem 2 – Analysis of Edgar Log Files**

**Team 8**

**Missing Data Analysis and Visualization**

To analyze the EDGAR Log File Dataset from the url: [https://www.sec.gov/data/edgar-log-file- data-set.html](https://www.sec.gov/data/edgar-log-file-%20data-set.html) The EDGAR Log File Dataset contains information in CSV format extracted from Apache log files and store user access for the SEC.gov website. The page lists the datasets for every month and year ranging from 2003 to 2016.

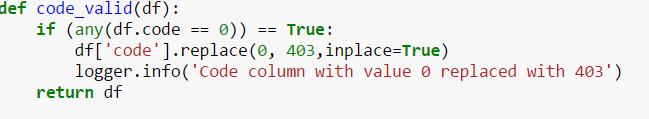
Every column’s description is defined in the url: <https://www.sec.gov/files/EDGAR_variables_FINAL.pdf>

We started with extracting the log files for first day of every month for all a particular year. We take user input for year. Once a valid year is provided its quarter is calculated using a function and based on the validation done, it downloads the zip files for first month’s file for that year.

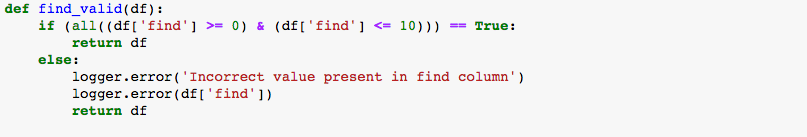
Once the download is complete, the csv files are extracted from them. We created a function that checks for empty values for each column in the log file. This function is run before the files are combined as one and sorted as one csv file. Once the combined file is generated we read the csv file into a data frame.

**Validations performed for the data**

Based on the column’s description we have performed validations on 8 columns separately.   
  
a) In column ‘code’, replaced code ‘0’ with code ‘403’ as code ‘0’ is not a valid HTTP status code using replace method.



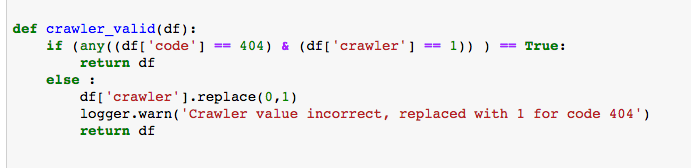
b) In column ‘find’, checked if the value of find is between 0 to 10 and if it’s is not then it is logged in the log file.



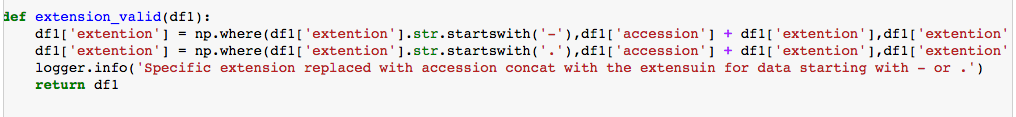
c) Date and time fields being continuous series, replaced them with previous date/time value using bfill method.

Screen%20Shot%202017-02-27%20at%206.49.19%20PM.png

d) In crawler, if the condition code is ‘404’ and crawler is ‘1’ evaluates to true then do nothing else replace crawler to ‘1’.



e) If Extension starts with ‘–‘ and ‘.’ then replace it with the accession number and append extension to it.



f) Date and Time columns is checked for correct format and if the format incorrect the error is logged in the log file.



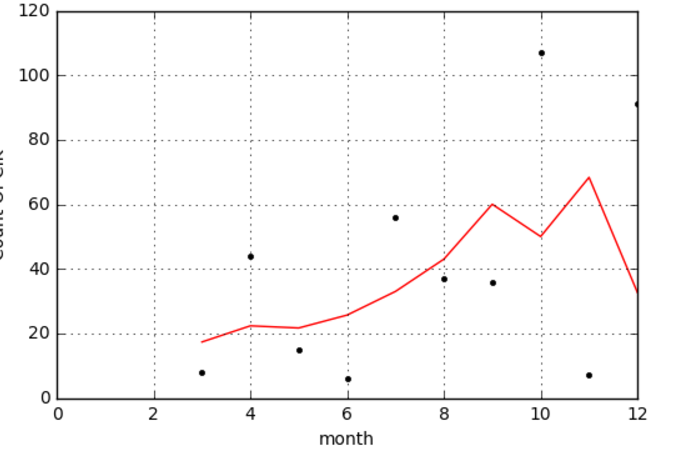


**Anomaly Detection**

Performed anomaly detection on the time series data using moving average to calculate any irregularities with the data.

As the data is time-series, we used moving average instead of the regular functions to detect if the data we have is deviating from the moving average for that particular time.  
We analyzed the number of hits made in total over time to see if the traffic exceeded more than the average traffic for every month.

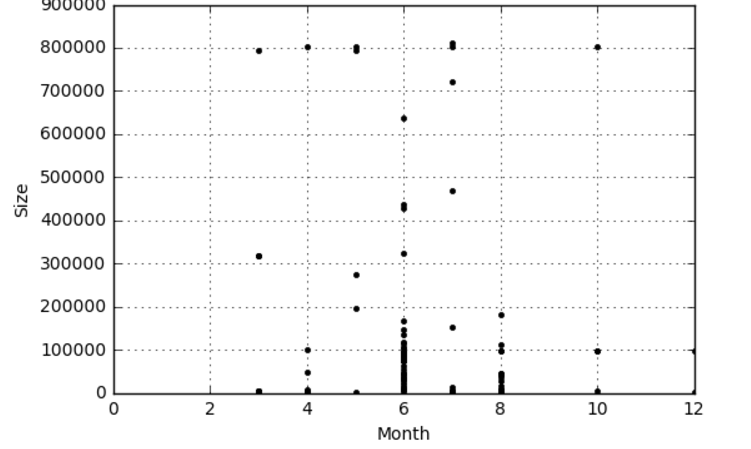
**Plot function for anomaly detection:**



The traffic or the counts are checked every month for a particular company to see if there is a deviation from the regular behavior.

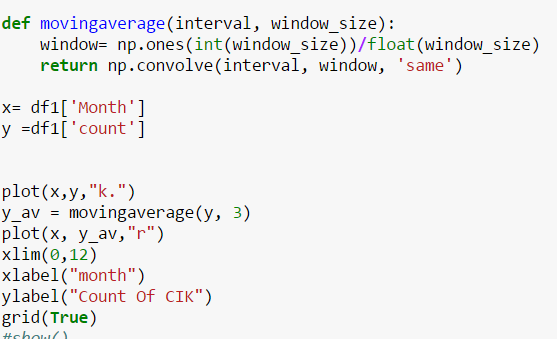
In this case, we can see some peaks in the 10th month and 11th month of the year which suggest some irregular behavior from the usual behavior. The red line in the graph shows the moving average for the traffic over 12 months.

We also performed similar analysis based on the size of the files downloaded of a particular company to see if there is any irregular data:



**The peaks generated for the size of the downloaded month suggest anomalies in the data.**

**Code snippet used:**

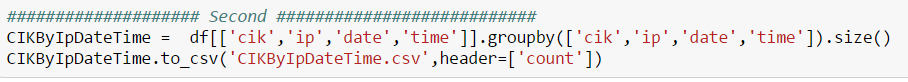


**Summarization functions in python**

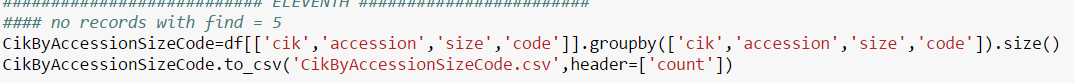
Summarization was done for different conditions and the generated summarized files were transferred to tableau for analysis.

We performed summarization of data grouping the data based on different columns. Some of the summarization are provided below

a) Summarized data for every company for different IP addresses over date and time to understand the traffic flow to each company.



b) Summarized data for every company, dropped down to individual document to individual size of every document and html code for every document to understand the file access pattern for each company along with status code.



c) Summarized data for every IP for the HTML code. For every individual IP and the sum total of requests made via that IP.



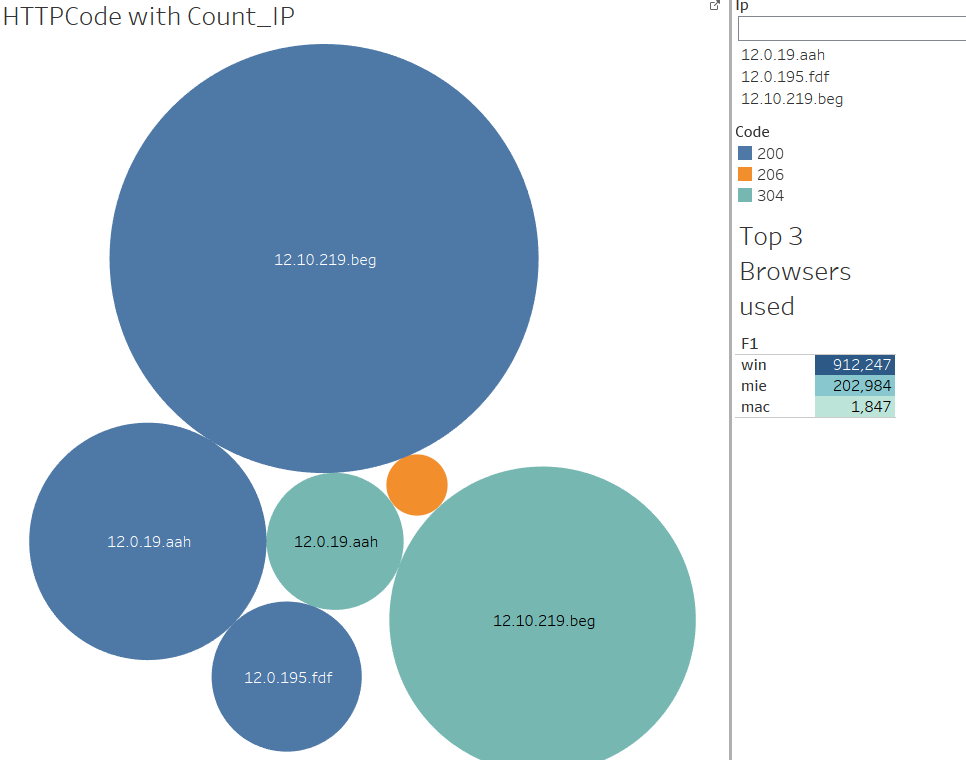
**Log files:**

Logger mechanism was implemented to log errors into a text file for error reporting detailing the timestamp of the error and the reason for the same.  
Logging was implemented on both the console and via text file using the ***logging*** library



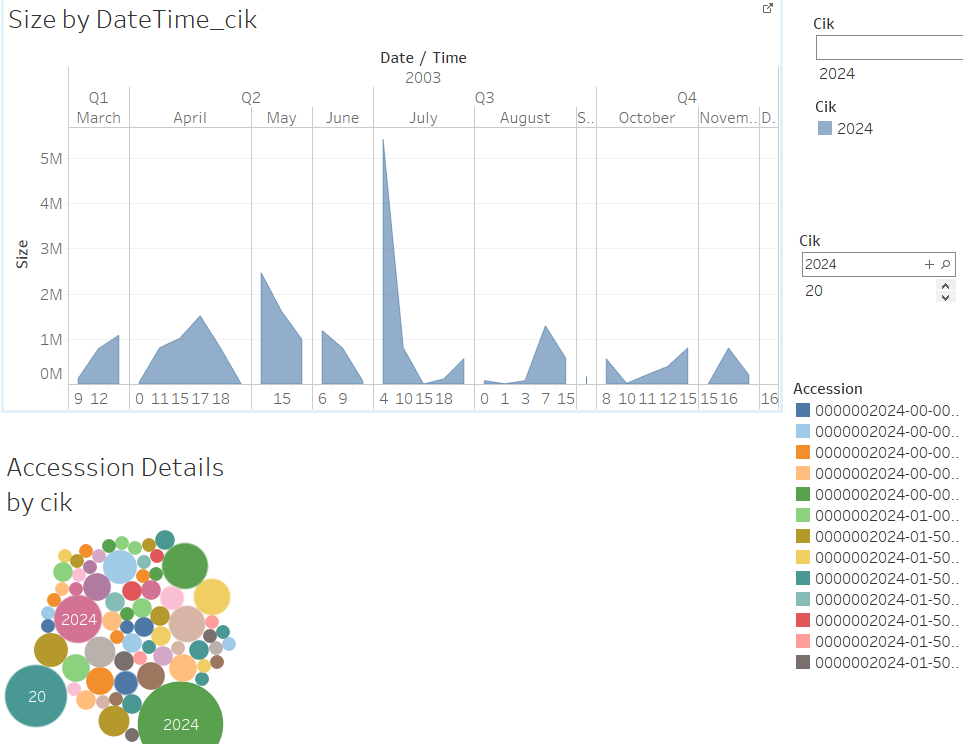
**Analysis Performed: (Tableau):**

1. Analyzing traffic to each IP addresses based on HTTP code and browser to understand how many hits were status code 200, 403, 404 etc and through which browser the traffic was more.



The individual circles represent different IP addresses, count of the requests made through that IP based on the HTML status code, color coded for different status.

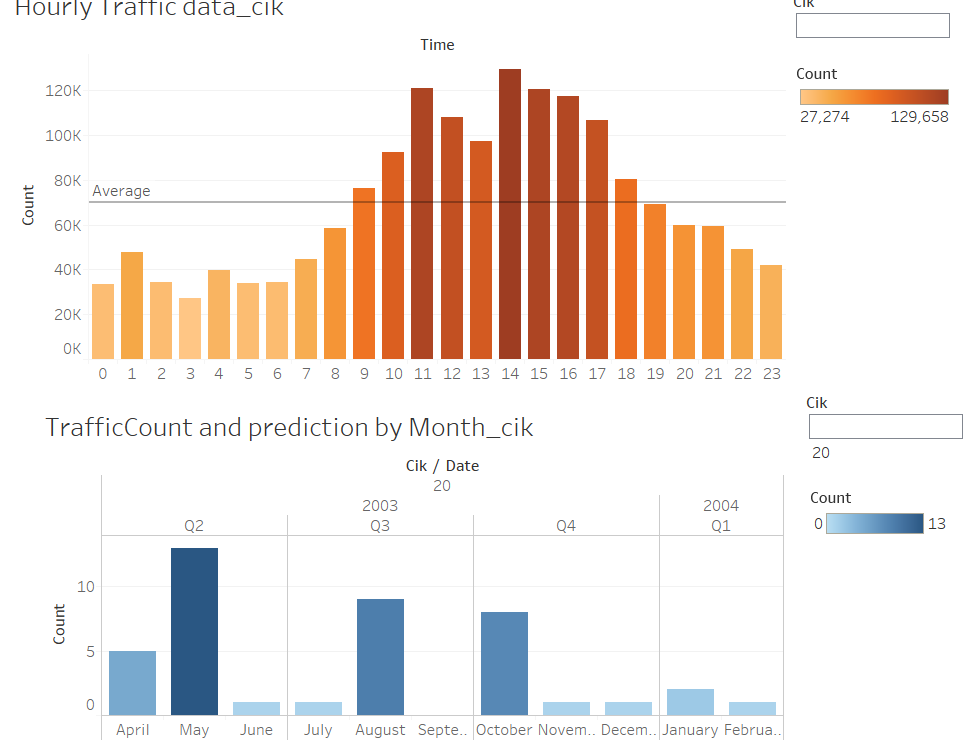
1. Analyzing size file downloaded based on each accession for every cik over time. Further drill down explains how many were successfully downloaded and how were not based on the status code.



The size of the files accessed for a particular company are shown over time, with data shown for each quarter, each month and for every hour of that month. Visualization shows pattern on when the data is accessed more or less based on time.

Analysis for individual document is also provided for every company and the HTTP request for every document accessed.

1. Analyzing the traffic of the requests for individual companies and collective data over a day and collective data over months.



The data is analyzed with reference to an average and checked for instances where the traffic is more or less than the average traffic over a day (24-hour span). The analysis can also be performed for an individual company.

The data is also analyzed over a year and forecasting is done for two months in next year (2004) based on the data analyzed for 2003.

**Design and implementation for both the Docker**

We created a DockerFile in a working directory with base image. Installed python and all the required packages. Some were picked up directly by docker. We set the working directory for the container in the DockerFile. This Working directory is used to store files in the container.

We then add this python script to the DockerFile and also add the python script in the Docker working directory. Once the DockerFile is updated and the python script is placed in the working directory, we then build the image on docker. A build image can be viewed by docker ps –a. We then run the image and start the container

After the container runs, we then set the parameters in the container and run the python script and check the zipped files in the S3 bucket

**Amazon S3**

Created a function to connect to amazon S3. We have parametrized the access and secret keys. The keys are asked by the user on the console. Once the valid keys are accessed the calls the S3 function.

Once connection is established with the help of the keys, it checks if a bucket with a given name already exists and if does not then it creates one and establish connection to it using tinsys3.

Once connection is established, get the csv files from the directory and upload it on AmazonS3.

**Validation**

The input keys are parameterized. We have used input method to get user input for access and secret keys and split them on space. If a space is not provided, then the keys are not accepted. We have used try and except clause to catch the exception.



