Assignment 1 Report

Working with Edgar datasets: Wrangling, Pre-processing and exploratory data analysis

**INFO 7390**

**Advance Data Science & Architecture**

Professor: Srikanth Krishnamurthy

By: Team 5

Akash Jagtap

Jerin Rajan

Nitin Prince Reuben

Table of content

Programming Languages & Libraries……………….………....3

Problem 1: Data wrangling using Edgar data from text file..….3

Problem 2: Missing data analysis & visualization……………..6

**Programming Language and Libraries**

To solve this assignment, we have used Python as out programming language and the following libraries to tackle different problems.

• urllib – used for web scraping

• BeautifulSoup – used for web scraping

• csv – used for writing csv

• logging – used for maintaining log file

• os – used for navigating, creating and deleting directories and files

• zipfile – used to handle zip files

• shutil – used for removing files

• pandas – used for putting data into dataframes

• Glob – used for reading files

• boto – used for handling AWS S3

**Problem 1:Data Wrangling using Edgar data from text files**

The objective of this problem is to extract all statistical tables from 10Q filings using Python.

1. We have to enter 5 values of different variables in command line. These variables are:   
     
   accessKey=Amazon access key

cik=CIK   
location=S3 location  
secretKey=amazon secret access key

accessionNumber=accession number

1. The program parses the command line arguments and puts them inside local variables.
2. How exception handling is performed:   
     
   • If amazon keys are not provided, then the program will exit   
     
   • If cik or accessionNumber is not provided in the command line, we have taken a default cik and accessionNumber for Walmart   
     
   • If location is not provided, it creates the bucket in default location.
3. Establish a connection with S3, if keys are invalid, the program will exit and the details are mentioned in the log file.
4. With the help of python’s urllib library, we are opening the requested URL.
5. We are using BeautifulSoup library for handling html tags in python.
6. Once the url is open, we find all the tags and their href attributes and then look for ‘10q’ pattern. Once we find the url, we enter the link. If we couldn’t find a URL, it will exit the program.
7. Once the 10q is open, we fetch all the tables by looking for tables present inside tag.
8. When we parse the file, we can understand that there are many tables which are not necessary. We have to include only those files which have data related to finance for example data containing either ‘%’ or ‘$’. So, we have to refine the data according to that.
9. In refined tables, we need to clean the table data. We have to remove unwanted characters such as ‘\n’ or ‘\xa0’.
10. After the data is clean, we create a corresponding csv file for the table inside Extracted\_csv folder. This folder is automatically created if it doesn’t exist.
11. The program zips the folder and put inside Problem1.zip.
12. Lastly, we create a bucket and upload the zip file. The bucket name is always unique as it’s the concatenation of ACCESS\_KEY+ Current\_Timestamp. If the keys are invalid, the program will log an error and exit.

Running the code for Problem 1

1. Created a docker image

FROM python:latest

WORKDIR /usr/local/bin

RUN pip install boto

RUN pip install beautifulsoup4

ADD Problem1.py .

CMD ["python", "./Problem1.py"]

1. Building a docker image: docker build -t problem1-image . Assignment 1 Advance Data Science & Architecture
2. Running the docker image: docker run problem1-image python Problem1.py accessKey=Amazon access key

cik=CIK   
location=S3 location  
secretKey=amazon secret access key  
accessionNumber=accession number

**Problem 2: Missing data analysis & visualization**

To solve problem 2, we have performed following steps

1. Initializing the Log file.
2. Cleaning up the directories.

* Create the required directories if not present.

1. Download the log files for all the months of the given year.

* By default, the year is set as 2003.
* The given year should be between 2003 and 2017 or else exit the program by displaying error.
* If log file of any day of month is not available or empty then keep checking for next day’s log file for downloading and so on.

1. Extract the csv files by unzipping the downloaded log files.
2. Create separate *dataframes* for each csv file extracted.
3. Check anomalies for each log file.

* For each variable count the null values.
* For columns idx, noagent and norefer check for values other then 0 and 1.

1. Handle missing values of each log file.

* Skip the rows having null(NaN) values for columns cik, ip, accession, data or time.
* For size column replace NaN with mean value of it whereas for columns browser, idx, norefer, noagent, code, find, crawler, extension and zone with the most repeating value of that respective column.

1. Compute the summary metrics, for each log file.
   * Calculate the mean of the column size and append it to the dataframe as a new column sizeMean.
   * Find the most repeating browser and append it to the dataframe as a new column maxBrowser.
2. Remove the rows with anomalies.
3. Create a single dataframe by combining all individual dataframes of respective log files and export it to a csv file.
4. Create a bucket on S3 and upload both the files.

* Name the bucket by concatenation of AccessKey + Current\_Timestamp.
* The program will log an error and exit for providing invalid keys.
* The bucket will be created in default location if location is not provided.

### Running the code for Problem 2

1. Create a docker image file with following content.

FROM python:latest

WORKDIR /usr/local/bin

RUN pip install boto

RUN pip install pandas

RUN pip install datetime

ADD Problem2.py .

CMD ["python","./Problem2.py"]

1. Build a docker image:

docker build -t problem2-image *.*

1. Run the docker image:

The program takes four arguments. Year, Amazon accessKey , Amazon secretAccessKey and location in the following format, but the order can be changed:

docker run problem2-image python Problem2.py year= **Year** accessKey=**AwsAccessKey** secretKey=**AwsSecretAccessKey** location=**Location**

**Note: the one’s in bold are the values that user is supposed to enter.**