INFO 7390- Advance Data Science

Working with Edgar datasets:

Wrangling, Pre-processing and exploratory data analysis

Course: INFO7390

Advance Data Science & Architecture

PROFESSOR:

Srikanth Krishnamurthy

Submitted By:

Team 8

Birwa Galia

Milony Mehta

Shantanu Deosthale

Table Of Contents:

Introduction

EDGAR, the Electronic Data Gathering, Analysis, and Retrieval system, performs automated collection, validation, indexing, acceptance, and forwarding of submissions by companies and others who are required by law to file forms with the U.S. Securities and Exchange Commission (the "SEC"). The database is freely available to the public via the Internet (Web or FTP). The goal of this assignment is to work with Edgar datasets

Programming Languages and Libraries :

To work with dataset we have used Python as a programing language and the following libraries of Python. We have used Python 3.x version.

- **Urllib:** urllib.request for opening and reading URLs
- **BeautifulSoup:** For web scrapping
- **Csv:** For Reading CSVs
- **OS:** For navigating, creating and deleting directories and files
- **Zipfile:** Handling zip files
- **Boto:** For handling AWS S3
- **Pandas:** For putting data into dataframes.
- **Logging:** For getting logging activities.

Logging:

Log entries are generated at each and every step of the program in problem 1 and 2. We have used the logging module in python which will log the operation with timestamp, level name and the message that we have customized.

We have mentioned two levels of in our logs:

- **Info**: These are the messages that gives information about each step as it gets executed in the program.
- **Warning:** This level is raised that needs attention but will not result in the program to fail.

Problem 1: Data wrangling Edgar data from text files

(Combined implementation for Part1 and Part2)

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

- Program takes 5 command line arguments for cik, accessionNumber, Amazon accessKey, Amazon secretAccessKey and location in the following format, but the order can be changed:

  For example: python Problem1.py

  "AWS_ACCESS_KEY_ID" =*YOUR_AMAZON_ACCESS_KEY*

  "CIK" =*CIK*

  "AWS_ACCESS_KEY_ID" =*YOUR_AMAZON_SECRET_ACCESS_KEY*

  "acc_no" =*ACCESSION_NUMBER*

  "Bucket location" = *YOUR_AMAZON_LOCATION*

- The program parses the command line arguments and puts them inside local variables.
- Exception Handling:
  - If amazon keys are not provided, then program exits.
  - If cik or accessionNumber is not provided, we have logger with invalid CIK or Acc_no
  - Establish connection to S3, if keys are invalid, the program exits after logging appropriate details.

With the help of python's urllib library, we are opening the requested URL. We are using BeautifulSoup library for handling html tags in python.

- Once the URL is open, we find all the <table> tags and their ‘td’ attributes and then look for ‘10q’ pattern and then we find all the <a> tags and the ‘href’ property and pick up the property and append it to the base URL. If nothing is found, we exit out of program, otherwise we open that URL.
- Fetching the tables doesn’t solve our problem, we need to find refined tables that contain statistical data. We found that the statistical data tables contain ‘background’ attribute in the table data. So we iterate through tables and look for pattern. If found, we break out of loop.
- In refined tables, we need to clean the table data which is inside <td> tags. We remove unwanted characters such as ‘\n’ or ‘\xa0’ characters.
- After the data is clean, we create a corresponding csv file for the table inside extracted_csv folder.
- The program zips the folder and put inside CIK.zip.
- Lastly, we create a bucket and upload the zip file. The bucket name is always unique as it’s the concatenation of CIK + Current_Timestamp. If the keys are invalid, the program will log an error and exit.

Running the code for problem 1 in docker

1.) Creating Docker File

FROM python

COPY edgar_scraping.py / edgar_scraping.py

RUN pip install pandas

pip install bs4

pip install boto

pip install requests

CMD [“python”, “./edgar_scraping.py”]

ENTRYPOINT [“python”, “./edgar_scraping.py”]

2.) Building a docker image:

Docker build -f DockerFile.

3.) Running the docker image:

Docker run milony/doc2 CIK here acc_no here *YOUR_AMAZON_ACCESS_KEY here* *YOUR_AMAZON_SECRET_ACCESS_KEY here*

4.) Docker image is available on docker hub. Use the following command

docker pull milony/prog1

- Docker image for problem 1 can be found at following link:

https://hub.docker.com/r/milony/prog1

Problem 2: Missing Data Analysis:

To tackle this problem 2 we have performed followings set in python environment.

- First, we have programmatically generated the URL while taking the Year as input, and if the year entered is not between 2003 and 2017, the program ends. Then, we download the log file for the first day of every month.
- Then we download the ZIP file and extract all the CSVs and store them in the current directory.
- Loaded each CSV into individual DataFrame
- Converted the data type of columns to int64 and date column into datetime.
- Performed missing data analysis:

Missing Data Analysis:

- If there are NaN values in IP, CIK and accession number we have dropped the entire the row as it should be the unique column.
- If there is a NaN value in date column, we have replaced it with the next observed value.
- If there is a NaN value in time and zone column, we have replaced it with the last observed value.
- This variable provides the filename of the file requested including the document extension. If the filename is missing and only the file extension is present, then the filename is the document accession number.
- If there is a NaN value in the code, we have replaced it with 0
- If there is a NaN value in the size, we have replaced it with 0
- If there is a NaN value in the idx, we have replaced it with 0

- If there is a NaN value in the norefer, we have replaced it with 0
- If there is a NaN value in the noagent, we have replaced it with 0
- If there is a NaN value in the find, we have replaced it with 0
- If there is a NaN value in browser, we have replaced it with default value.

Summary Metrices

- Getting total number of files as per time
- Getting number of files accessible as per status code
- Dividing size into quartiles and getting total number of files as per quartile range

- Combining All Data log (which are currently in one individual DataFrame) to a single DataFrame and then export it into CSV.

Lastly, create a bucket on S3 and upload the files:

- If the keys are not valid and not entered exit the program.
- If the location is not provided, default location is taken.

Running the code for problem 2 in docker

- Creating Docker File
  FROM python
  COPY edgar_scraping.py / edgar_scraping.py
  RUN pip install pandas
  pip install bs4
  pip install boto

  pip install requests
  CMD [“python”, “./edgar_scraping.py”]
  ENTRYPOINT [“python”, “./edgar_scraping.py”]
- Building a docker image:
  Docker build -f DockerFile.
- Running the docker image:
  Docker run milony/doc2
  CIK here
  acc_no here
  YOUR_AMAZON_ACCESS_KEY here
  YOUR_AMAZON_SECRET_ACCESS_KEY here
  YOUR_AMAZON_LOCATION here

- Docker image is available on docker hub. Use the following command

  docker pull milony/prog1

- Docker image for problem 1 can be found at following link:

  https://hub.docker.com/r/milony/prog1