Assignment 1 - Group 6

Xinglong Jia, YinYin Ye, Himanshu Gupta

12/10/2018

1 Code Link

1.1 DockerHub:

Problem 1 — `https://hub.docker.com/r/yinyinye/parse_file_docker.py/`
Problem 2 — `https://hub.docker.com/r/yinyinye/missing_data_docker.py/`

1.2 Github:

`https://github.com/xinglongjia/info7390_group6`

2 Problem 1: Data wrangling Edgar data from text files

2.1 Part 1: Parse files

Before we start the assignment, we watched a few videos which about how to web scrapper data to make sure we know how to precede through the assignment. The video link is provided in the Reference section of this report.

After we felt comfortable about the whole web scrapper process, we used [google chrome inspect tool] to take a look of the HTML formal of the webpage. Based on given CIK and Access number in config file, generated the index website url. For parsing html, we used BeautifulSoup4 library and lxml parser to parse the html derived from the url. We found all tag 'tr' and checked the column of states type of 10-Q to extract 10-Q url. Then we used panda read_html() method to extract all tables.

Finally after we cleaned up the data: replace useless symbol, shifted rows to left, dropped empty rows and columns and replaced nan to blank. After using make_archive() method in shutil library to compress result files, we used boto3 to connect with AWS and upload the zip file to AWS account

2.2 Part 2: Dockerize this pipeline

When we dockerize the pipeline, we need to create a Dockerfile first, then we build the image locally. After we have the image, we tag the image ID and states the version of the image. Finally we push the image to the docker hub, so the image can be accessed publicly through the docker hub by using the link that we provided on the top or using terminal command line : docker pull yinyinye/parsefile_docker.py.

There are quite few things need to be taken carefully when start part 2 of problem 1. We need to make sure that Amazon access key and the secert key are input parameter instead of hard coding, so that other people cannot randomly uses our account. Therefore, for our implementation, the python code will ask for the Amazon parameters in order to precede the process. Apart from the parameters issue, the Dockerfile also needs to be handle carefully since we import so many different package. We need to put RUN pip install package for every package that we are using in the code.

2.3 Handle exceptions

AWS S3 bucket path:
`https://s3.amazonaws.com/info7390-group6-parse-file/result.zip`

There are two files in result.zip:
cleaned_tables — directory that stores all cleaned tables, from table4 to table201. Every table is cleaned.
parse_file_log.log — store all log information with time stamps, config level is 'info', only info and warning messages in the program are written.

Handle exception for web scrapper is very necessary since there are always some missing data or null object. We use try-exception function in python to process these exceptions. Thus when there is no 10Q file, the logging is going to show 'wrong 10Q url, fail to get 10Q file' and exit the process. Also, we use try-exception when we ask for the AWS access key and secret key. If the input paramters are bad which means the aws cannot do the connection, the logging is going to show 'Error, please check aws configuration' and exit the process. By doing this way, our python code is working fine even when there are exceptions.

3 Problem 2: Missing Data Analysis

3.1 Process the file

handle missing data
Use dataframe.fillna() method, for 'browser' column, fill in 'unknown', for 'size' column, fill in the mean of non-null values.

Summary metrics
Use dataframe.describe() method, get the count, mean, std, min, 25%, 50%, 75% and max.

Check anomalies
Define a function, use box plot concept lower inner fence:

$$Q1 - 1.5 * IQ$$

and upper inner fence:

$$Q3 + 1.5 * IQ$$

as fences to check anomalies.

3.2 Docker Part

The Dockerize the pipline is similar to problem 1. We have to parameterize the AWS accessing key and secret key for safety prepose. We build the image locally and then push it to the dockerhub. The Dockerfile is also required and we need to RUN pip install all the package that we used in the python code for the docker to run the container. When we docker run the file, in the terminal, it will ask to input AWS access key and secret key for next step to upload the zip file to S3.

3.3 Handle Exceptions

AWS S3 bucket path:
`https://s3.amazonaws.com/info7390-group6-missing-data/problem2_result.zip`

There are two files in problem2_result.zip:
compiled_result.csv — dataset that store all data and summaries of 12 months. Missing data are replaced, anomalies are removed, summaries are added.
missing_data_log.log — store all log information with time stamps, config level is 'info', only info and warning messages in the program are written.

We use try exception to handle exception during the process. Similar to problem 1, we put try-exception in two place. One is when we try to find the file for a specific year and the other one is when we ask for aws credential inputs. Thus try-exception function return log information when something goes wrong and exit the process.

4 Reference

Web scrapping `https://www.youtube.com/watch?v=ng2o98k983k`
Docker tutorial `https://www.youtube.com/watch?v=YFl2mCHdv24`