SI 206 Final Project Report

Analyzing the years for each sector being in S&P 500 ranking

By Kedi Jiang, Annabel Zhuang

Goals：

The goal of this project is to figure out whether each sector in S&P 500 has been in the ranking for the same number of years, and what sector is relatively new in the ranking.

Problems that I faced:

I struggled to only extract the data I wanted and grab only 25 rows of data at a time.

Calculation File:

<https://github.com/annazjf/SI-206-Final-Project-JK.git>

Output:

{

    "Communication Services": {

        "newest\_added\_date": "2022-04-11",

        "oldest\_added\_date": "1976-06-30",

        "difference\_years": 45.78

    },

    "Consumer Discretionary": {

        "newest\_added\_date": "2021-03-22",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 64.05

    },

    "Consumer Staples": {

        "newest\_added\_date": "2023-03-15",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 66.03

    },

    "Energy": {

        "newest\_added\_date": "2022-10-12",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 65.61

    },

    "Financials": {

        "newest\_added\_date": "2022-11-01",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 65.66

    },

    "Health Care": {

        "newest\_added\_date": "2023-03-15",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 66.03

    },

    "Industrials": {

        "newest\_added\_date": "2022-09-19",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 65.54

    },

    "Information Technology": {

        "newest\_added\_date": "2023-03-20",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 66.04

    },

    "Materials": {

        "newest\_added\_date": "2019-06-07",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 62.26

    },

    "Real Estate": {

        "newest\_added\_date": "2022-09-19",

        "oldest\_added\_date": "2001-12-03",

        "difference\_years": 20.79

    },

    "Utilities": {

        "newest\_added\_date": "2022-10-03",

        "oldest\_added\_date": "1957-03-04",

        "difference\_years": 65.58

    }

}

from bs4 import BeautifulSoup

import requests

import os

import csv

import sqlite3

import json

from datetime import datetime

import matplotlib.pyplot as plt

#Web scraping form Wikipedia website to get the tabel of S&P 500 companies information

def get\_SP\_companies\_table():

  url = "https://en.wikipedia.org/wiki/List\_of\_S%26P\_500\_companies"

  resp = requests.get(url)

  soup =  BeautifulSoup(resp.content, "html.parser")

  table = soup.find('table', id = "constituents")

  rows = []

  tbody = table.select('tbody')

  trs  = tbody[0].find\_all('tr')

  #get the data in the following rows

  for tr in trs: # for every table row

        rows.append([td.get\_text(strip=True) for td in tr.find\_all('td')]) # data row

  return rows

#Write the rows from scraping to a csv file

def write\_row(rows, filename):

    with open(filename, "w", encoding="utf-8-sig", newline="") as f:

        writer = csv.writer(f)

        for i, row in enumerate(rows):

            row\_with\_number = [i] + row # Add the number to the beginning of the row

            writer.writerow(row\_with\_number)

# set up database

def setUpDatabase (db\_name):

   path = os.path.dirname(os.path.abspath('\_\_file\_\_'))

   conn = sqlite3.connect(path + "/" + db\_name)

   cur = conn.cursor()

   return cur, conn

#create the sector dictionary and table where the id is the Sector\_id in the companies table to make sure there is no duplicate string data

def creat\_dict(cur2, conn2):

  dic = {}

  dic["Communication Services"] = 1

  dic["Consumer Discretionary"] = 2

  dic["Consumer Staples"] = 3

  dic["Energy"] = 4

  dic["Financials"] = 5

  dic["Health Care"] = 6

  dic["Industrials"] = 7

  dic["Information Technology"] = 8

  dic["Materials"] = 9

  dic["Real Estate"] = 10

  dic["Utilities"] = 11

  # cur2.execute("CREATE TABLE IF NOT EXISTS Sector (id INTEGER PRIMARY KEY, sector\_name STRING UNIQUE)")

  # for i in dic:

  #   cur2.execute("INSERT INTO Sector (id, sector\_name) VALUES(?,?)", (dic[i], i))

  # conn2.commit

  return dic

#Create the companies table and read the data from csv, insert it to the companies table

def createdb (cur2, conn2, dicti ):

  # Check if the table exists

  cur2.execute("SELECT name FROM sqlite\_master WHERE type='table' AND name='companies'")

  table\_exists = cur2.fetchone() is not None

  idx = 0

  if table\_exists:

      # If the table exists, find the row number of the most recent id

      cur2.execute("SELECT MAX(id) FROM companies")

      idx = cur2.fetchone()[0]

  cur2.execute("CREATE TABLE IF NOT EXISTS companies (id PRIMARY KEY, Sector\_id INTEGER, Date\_added DATE)")

  # Read the CSV file and insert the remaining rows into the database

  with open('companies\_data.csv', 'r',encoding='utf-8') as f:

      reader = csv.reader(f)

      next(reader)  # Skip the header row

      i = 0

      for row in reader:

          if not row[6]:

             continue

          if i >= idx:

              cur2.execute("INSERT OR IGNORE INTO companies (id, Sector\_id, Date\_added) VALUES (?,?,?)",(int(row[0]), dicti[row[3]],row[6]))

          if i == idx + 24:

              break

          i += 1

  # Commit changes and close the database connection

  conn2.commit()

#create a jason file with the dictionary where the keys are each of the sector name and the value is another dictionary stating the newest\_added\_date, oldest\_added\_date, andyears in S&P 500

def get\_sector\_dictionary(cur, conn):

  cur.execute("SELECT Sector\_id, MAX(Date\_added), MIN(Date\_added), sector\_name FROM companies JOIN Sector ON companies.Sector\_id = Sector.id GROUP BY Sector\_id")

  rows = cur.fetchall()

  # create a dictionary with GICS Sector as the key and the newest and oldest date added

  result\_dict = {}

  for row in rows:

    dic = {}

    sectorname = row[3]

    newest\_added\_date = row[1]

    oldest\_added\_date = row[2]

    difference\_years = round((datetime.strptime(newest\_added\_date, '%Y-%m-%d') - datetime.strptime(oldest\_added\_date, '%Y-%m-%d')).days / 365.25, 2)

    dic[sectorname] = {

        'newest\_added\_date': newest\_added\_date,

        'oldest\_added\_date': oldest\_added\_date,

        'difference\_years': difference\_years

    }

    result\_dict.update(dic)

# write the result to a JSON file

  with open('result.json', 'w') as f:

    json.dump(result\_dict, f, indent=4)

  conn.commit()

  return result\_dict

#Plot the result to a bar graph stating the number of Years for Each Sector in S&P 500(sector name, years)

def plot\_the\_result (dic):

    Sector\_name = dic.keys()

    years = []

    for i in dic:

      year = dic[i]["difference\_years"]

      years.append(year)

    fig, ax = plt.subplots(figsize=(8, 7))

    plt.subplots\_adjust(bottom=0.4)

    plt.bar(Sector\_name, years)

    plt.xlabel('Sector Name')

    plt.ylabel('Years in S&P 500')

    plt.xticks(rotation=90)

    plt.title('Number of Years for Each Sector in S&P 500')

    plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

   cur2, conn2 = setUpDatabase ("founding\_data.db")

   dicti = creat\_dict(cur2, conn2)

   createdb (cur2, conn2, dicti)

   rows = get\_SP\_companies\_table()

   write\_row (rows, "companies\_data.csv")

  #  get\_db\_connection()

   result = get\_sector\_dictionary(cur2, conn2)

   plot\_the\_result (result)

Visualizations:

Chart, bar chart

Description automatically generated

Instructions for running the code:

First run the Python file, you will get a csv file and the two tables within the database called (founding data). Those two tables are companies and Sector.

Then you can see a bar chart visualization demonstrating the number of years for each sector being in S&P 500 ranking.

Code documentation:

1. get\_SP\_companies\_table() function performs web scraping on the Wikipedia page for the S&P 500 companies table. It uses the BeautifulSoup library to parse the HTML content of the page from a specific URL, extract the data from rows, and return a list of lists containing the rows of the table.

2. write\_row(rows, filename): This function takes the rows from the scraping and writes them to a csv file with the filename indicated.

3. setUpDatabase(db\_name): This function takes in a database name, creates a connection to a SQLite database with that name, sets up a cursor object, and returns the cursor and the connection.

4. creat\_dict(cur2, conn2): This function creates the table and the sector dictionary with the 11 GICS (Global Industry Classification Standard) sector names as keys and their corresponding sector id as values. The id is the Sector\_id in the companies table to make sure there is no duplicate string data. It returns the dictionary

5. createdb(cur2, conn2, dicti): This function takes in the cursor, connection, and the sector dictionary that was created previously. It creates a table named “companies” in the SQLite database, reads the data from the csv file created in write\_row() function, and inserts it into the table. It also takes the id in the dicti that passed in as the sector\_id in the table. It returns None.

6. get\_sector\_dictionary(cur, conn): This function takes in the connection and cursor. It queries the database to get the newest and oldest dates added for each sector, calculates the difference in years between them, and creates a dictionary with each sector name as a key and another dictionary as the value. The inner dictionary has the key and value of newest\_added\_date, oldest\_added\_date, and difference\_year. The output is the dictionary.

7. plot\_the\_result(dic): This function takes the dictionary returned from get\_sector\_dictionary() and creates a bar graph using the Matplotlib library with the sector names on the x-axis and the difference\_years on the y-axis, indicating the number of years for each sector in S&P 500. It returns None.

Documentation of resources used:

**Date**: 4/22/2023

**Issue Description**: Hope to autoincrement the id.

**Location of Resource**: https://stackoverflow.com/questions/2935658/beautifulsoup-get-the-contents-of-a-specific-table

**Result**: I referenced the practice on this website, and it solved the problem.