Data_scraping_cleaning_checkpoint

November 15, 2019

Prepared by:
Joseph Leung
Gabriel Smith
Samuel Hsiung

1 Explanation of Data and Proposal

1.1 2. Potential problems with the data:

a. Our data is based on a fairly well-reputed sources, namely, the Fortune 1000 company ranking, the Wall Street Journal, Glassdoor, and the Harvard Business Review. As many other websites and companies base their evaluations on these sources, we feel that this justifies our decision to use information found in these websites as reliable data. Nevertheless, while these are fairly respected organizations whose rankings and information are known to be legitimate, it is fair to recognize that these sources are still likely biased by a number of factors, such as the different companies who may be donors to these sources.

However, we feel that due to using different resources, such as both the Glassdoor and HBR CEO rankings, this helps account for this bias somewhat. Another way to combat this bias is how we use the fact that existing on a ranking system can be an indicator in of itself. With company on 1000 companies, the fact that a CEO is ranked by a reputable source is a variable which can compare different companies and their results.

b. Our data is imperfect, and some missing values do exist due to the fact that we scraped from websites that intentionally make their websites difficult to scrape from. This can be resolves by rescraping the data at a slower pace, but as this occured with relatively few rows of data, it is also a possibility to simply remove those values. Otherwise, thanks to the reputability of our data, most of the general fixing we need to do is cleaning to allow for a clean merging of our data.

1.2 3. Suitability of Data

Our intial proposal was primarily focused on collecting profile data for individuals and how the overall network community and user profile info can help us predict decision biases.

However, the data required for our original proposal is expensive to collect in the timeframe given, thus we decided to reframe our project and focus instead on collecting profile data on high profile individuals namely C-suite executives and Board member profiles. This data consists of vital company statistics, CEO approval ratings/rankings, board of executives composition and diversity and other unqiue distinguishing features describing the type of community the company exists in.

Data on CEO rankings and approval ratings are scraped from both Glassdoor and the Harvard Business Review. We also scraped from Fortune Magazine website and Wall Street Journal data regarding assets, revenue, profit and its relative change from previous years. With this new approach data is much more accessible yet still can help us effectively answer a piece of our orginal proposal.

1.3 4. Proposal Revisions

Where as before we were eager to understand the impacts that a particular C-suite board's affiliations would have on the strategic decision making of a company and how that would bias their ultimate decision, with this data we have elected to focus more on the impact that a CEO and his or her board members would have on the general performance of a company, and to understand the different features that most affect that performance according to different kind metrics such as profitability or assets under possession.

Although much simpler than our original proposal question, solving this simplified profile c-suite/company profile analysis will allow us to create hypothesis and models that explain and predict bias in organizational deicision making. We plan on using insights found in this preliminary project to help us begin desigining weighting algorithims to predict said biases for the next semester.

Interesting resources we've focused on:

Info to scrape

https://hbr.org/2019/11/the-ceo-100-2019-edition

https://www.ceotodaymagazine.com/top-50-ceos/

https://fortune.com/fortune500/2019/search/

https://www.usnews.com/news/best-countries/articles/2018-01-23/people-around-the-world-approve-of-company-ceos-more-than-world-leaders

https://www.glassdoor.com/Award/Top-CEOs-2018-LST_KQ0,13.htm

References:

https://business.linkedin.com/talent-solutions/blog/trends-and-research/2018/what-12000-ceos-have-in-common

https://www.glassdoor.com/Award/Top-CEOs-LST_KQ0,8.htm

2 Scraping

2.1 Scraping the Fortune 1000 companies

[0]:

```
import re
import time
import requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import numpy as np
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
```

```python
from selenium.common.exceptions import NoSuchElementException
import pickle
from time import sleep
import pandas as pd

browser = webdriver.Chrome('chromedriver.exe')
base_url="https://fortune.com/fortune500/2019/search/"

page_data = []   #data found on each page
CEO_data = []    #data found on the page describing the CEO and further details
try:
    browser.get(base_url)
    next_page  = browser.find_elements_by_xpath('//button[@type="button"]')
    sleep(5)
    for p in range(10):
        print(p)
        browser.get(base_url)
        next_page  = browser.find_elements_by_xpath('//button[@type="button"]')
        sleep(5)
        for i in range(p):
            #find the next button, and move to the page we want to be on
            next_page  = browser.find_elements_by_xpath('//
 ↪button[@type="button"]')
            next_page[-3].click()
        #find the hyperlinks
        hl = browser.
 ↪find_elements_by_class_name('searchResults__cellWrapper--39MAj')
        hyperlinks1 = [w.get_attribute('href') for w in hl]
        hyperlinks2 = []
        for h in hyperlinks1:
            if h not in hyperlinks2:
                hyperlinks2.append(h)

        page_data.append([w.get_attribute('text') for w in hl])
        #getting the CEO data
        for h in hyperlinks2:
            browser.get(h)
            sleep(5)
            stuff = browser.find_elements_by_class_name('dataTable__row--34F3j')
            CEO_data.append([s.get_attribute('innerHTML') for s in stuff])

except NoSuchElementException:
    print("don't swear")
    raise
```

[0]:
```python
len(page_data) #check we got all the pages
```

[0]:
```
comp_data = [[a[i:i+11] for i in range(0,len(a),11)] for a in page_data]␣
 ↪#separate into the companies
#clean up all the variables
index = np.concatenate(np.array(comp_data)[:,:,0])
companies = np.concatenate(np.array(comp_data)[:,:,1])
revenues = np.concatenate(np.array(comp_data)[:,:,2])
rev_perc_change = np.concatenate(np.array(comp_data)[:,:,3])
profit = np.concatenate(np.array(comp_data)[:,:,4])
prof_perc_change = np.concatenate(np.array(comp_data)[:,:,5])
assets = np.concatenate(np.array(comp_data)[:,:,6])
market_value = np.concatenate(np.array(comp_data)[:,:,7])
rank_change1000 = np.concatenate(np.array(comp_data)[:,:,8])
employees = np.concatenate(np.array(comp_data)[:,:,9])
rank_change500 = np.concatenate(np.array(comp_data)[:,:,10])
#create a pandas dataframe
companies = pd.DataFrame({'index':index,'companies':companies,'revenues':
 ↪revenues,'rev_perc_change':rev_perc_change,'profit':profit,
                          'prof_perc_change':prof_perc_change,'assets':
 ↪assets,'market_value':market_value,'rank_change1000':rank_change1000,␣
 ↪'employees':employees,'rank_change500':rank_change500})
```

[0]:
```
CEO_data2 = [[re.findall(r">[\%\$\(\)\-\,\;\&\s\.a-zA-Z0-9]+<",t) for t in c]␣
 ↪for c in CEO_data] #clean up the CEO data received
#get the CEO names
ceo_names = []
for i in CEO_data2:
    if i != []:
        ceo_names.append(i[0][-1])
    else:
        ceo_names.append(np.nan)
```

[0]:
```
#add this information to the dataframe
companies['ceos_page_data'] = CEO_data2
companies['ceo_name'] = ceo_names
companies.to_csv('fortune.com_companies.csv')
```

2.2 Scraping the WSJ S&P 500 Board Member Data

[0]:
```
url = 'http://graphics.wsj.com/boards-of-directors-at-SP-500-companies/'

companies = browser.find_elements_by_xpath('//*[@companyName]')
info = []
for ii in companies:
    outerhtml = ii.get_attribute('outerHTML') #// to extract outerHTML
    tag_value=outerhtml.split("\" ") #// to extract board member info
    info.append(tag_value)
```

```python
#organizing the board-member data
companynames = []
marketcap = []
industry = []
tot_dir = []
med_age = []
medpay = []
perc_woman = []
board_ind = []
tenure = []
for i in info:
    element_added = [False for i in range(9)]
    for e in i:
        if 'companyName' in e:
            companynames.append(e[13:])
            element_added[0] = True
        elif 'mktcap' in e:
            marketcap.append(e[8:])
            element_added[1] = True
        elif 'industry' in e:
            industry.append(e[10:])
            element_added[2] = True
        elif 'directorstotal' in e:
            tot_dir.append(e[16:])
            element_added[3] = True
        elif 'age' in e:
            med_age.append(e[5:])
            element_added[4] = True
        elif 'unrelated' in e:
            board_ind.append(e[11:])
            element_added[5] = True
        elif 'tenure' in e:
            tenure.append(e[8:])
            element_added[6] = True
        elif 'medianpay' in e:
            medpay.append(e[11:])
            element_added[7] = True
        elif 'female' in e:
            perc_woman.append(e[8:])
            element_added[8] = True
    for n,e in enumerate(element_added):
        if e == False:
            if n == 0:
                companynames.append(np.nan)
            elif n == 1:
                marketcap.append(np.nan)
            elif n == 2:
```

```
                industry.append(np.nan)
            elif n == 3:
                tot_dir.append(np.nan)
            elif n == 4:
                med_age.append(np.nan)
            elif n == 5:
                board_ind.append(np.nan)
            elif n == 6:
                tenure.append(np.nan)
            elif n == 7:
                medpay.append(np.nan)
            elif n == 8:
                perc_woman.append(np.nan)

#turn into dataframe and save as csv for cleaning
wsj = pd.DataFrame({'Company':companynames,'Market Cap':marketcap,'Industry':
 ↪industry,'No_Directors':tot_dir,'Median_age':med_age,'Board_Independance':
 ↪board_ind,'Median_Tenure':tenure,'Median_pay':medpay,'women_on_board':
 ↪perc_woman})
wsj.to_csv('wsjS&P.csv')
```

2.3 Scraping the Glassdoor CEO rankings

[0]:
```
browser = webdriver.Chrome('chromedriver.exe')
url = 'https://www.glassdoor.com/Award/Top-CEOs-LST_KQ0,8.htm'
browser.get(url)
ids = browser.find_elements_by_xpath("//*[@class ='h2 m-0']")

#get the names from Glassdoor
glassdoor_name = []
for ii in ids:
    outerhtml = ii.get_attribute('outerHTML') #// to extract outerHTML
    glassdoor_name.append(outerhtml)

#get the corresponding companies
glassdoor_comp = []
companies = browser.find_elements_by_xpath("//*[@class ='mt-xsm mr-xl mb-0']")
for c in companies:
    glassdoor_comp.append(c.get_attribute('outerHTML'))

#clean out the CEO names and their approval rating
gd_names, gd_approve = [],[]
for j in range(len(glassdoor_name)):
    n,a = re.findall(r">[\%\$\(\)\-\,\;\&\s\.a-zA-Z0-9]+<",glassdoor_name[j])
    gd_names.append(n)
    gd_approve.append(a)
```

```
gd_comps = [re.findall(r">[\''\\\%\$\(\)\-\,\;\&\s\.
↪a-zA-Z0-9]+<",glassdoor_comp[j]) for j in range(len(glassdoor_comp))]
```

2.4 Scraping the Harvard Business Review CEO Rankings

[0]:
```
hbr = 'https://hbr.org/2019/11/the-ceo-100-2019-edition'
browser = webdriver.Chrome('chromedriver.exe')
browser.get(hbr)
hbr_name = browser.find_elements_by_xpath("//*[@class ='organisationname']")
hbr_info = browser.find_elements_by_xpath("//*[@class ='organisationinfo']")

#getting the hbr names and company
hbr_stuff = []
for h in hbr_info:
    outerhtml = h.get_attribute('outerHTML') #// to extract outerHTML
    hbr_stuff.append(outerhtml)

hbr_names, hbr_comp = [],[]
for j in range(len(hbr_ceo)):

    search = re.findall(r">[\''\\\%\$\(\)\-\,\;\&\s\.
↪\wéèüçîáa-zA-Z0-9]+<",hbr_ceo[j])
    if len(search) == 2:
        n,a = re.findall(r">[\''\\\%\$\(\)\-\,\;\&\s\.
↪\wéüèçîáa-zA-Z0-9]+<",hbr_ceo[j])
    else:
        print(j)
        continue
    hbr_names.append(n)
    hbr_comp.append(a)

# get the variables found associated with the ranking
hbr_industry, hbr_country,hbr_year_started,hbr_insider,hbr_MBA,hbr_finrank,␣
↪hbr_sustainalytics, hbr_csrhub = [],[],[],[],[],[],[],[]
for j in range(len(hbr_stuff)):
    print(j)
    search = re.findall(r">[\|\''\\\%\$\(\)\-\,\;\&\s\.
↪\wéèüçîáa-zA-Z0-9]+<",hbr_stuff[j])
    hbr_industry.append(search[0])
    hbr_country.append(search[1])
    hbr_year_started.append(search[2])
    hbr_insider.append(search[3])
    hbr_MBA.append(search[4])
    hbr_finrank.append(search[5])
    hbr_sustainalytics.append(search[6])
```

```
    hbr_csrhub.append(search[7])
```

[0]:
```
#put all CEO data together
ceo_rank = pd.DataFrame({'GD_CEO':gd_names, 'GD_Approval':
 ↪gd_approve,'GD_company':gd_comps})

ceo_rank['HBR_CEO'] = hbr_names
ceo_rank['HBR_Company'] = hbr_comp
ceo_rank['HBR_Industry'] = hbr_industry
ceo_rank['HBR_Country'] = hbr_country
ceo_rank['HBR_YearStarted'] = hbr_year_started
ceo_rank['HBR_Insider'] = hbr_insider
ceo_rank['HBR_MBA'] = hbr_MBA
ceo_rank['HBR_finrank'] = hbr_finrank
ceo_rank['HBR_sustainalytics'] = hbr_sustainalytics
ceo_rank['HBR_csrhub'] = hbr_csrhub

ceo_rank['Rank'] = ceo_rank.index + 1

#send to csv for further cleaning
glassdoor = [col for col in ceo_rank if col.startswith('GD')]
HBR = [col for col in ceo_rank if col.startswith('HBR')]
ceo_rank[['Rank'] + glassdoor ].to_csv('ceo_rank_glassdoor.csv')
ceo_rank[['Rank'] + HBR ].to_csv('ceo_rank_hbr.csv')
```

3 Cleaning The Data

Here are the following steps that we used to clean and merge the different data sources:

1. Clean the Fortune 1000 Data and extract the infor nested in the intial scrapped data
2. Clean the WSJ data and prepare the company names to match company name format in the fortune 1000 and merge data, via outer join
3. Clean the Glassdoor and Harvard Business Review and prepare company names for additional outer join into the main data set
4. Outer-join all four data sources together

Fortune 1000 Data Cleaning and extraction

3.1 1. Fortune 1000 Preparation and Extraction for merge

[0]:
```
import pandas as pd
import numpy as np
import scipy as sc

#Reading in the scraped data from Fortune 1000 list
df = pd.read_csv("fortune.com_companies.csv")
```

```
#Splitting the format seperated by "], ["
df["ceos_page_data"].values[0].split("], [")

#We will extract each element of the larger blob and make a dictionary pointin␣
↪each element to its corresponding value
#We will make the diamond from the junk and find stuff.
def make_dict(junk):
    diamond = dict()
    dumpster  = junk.split("], [")

    #Enumerating the index and values to iterate through and extract the string␣
↪wanted
    for i,trash in enumerate(dumpster):
        rubbish = trash.split("<', '>")

        #If the split string is only length one it means the element points to␣
↪no value
        if len(rubbish) == 1:
            if i == 0:
                key = rubbish[0][4:-2]
            elif i == len(dumpster) - 1:
                key = rubbish[0][2:-4]
            else:
                key = rubbish[0][2:-2]
            diamond[key] = None

        #If more than one element in each thing then we point to the value, and␣
↪the perc growth following the key
        else:
            #Different index positioning has different formatting just for term␣
↪and last term
            if i == 0:
                key = rubbish[0][4:]
                value = rubbish[-1][:-2]
                diamond[key] = value
            elif i == len(dumpster) - 1:
                key = rubbish[0][2:]
                value = rubbish[-1][:-4]
                diamond[key] = value
            else:
                if len(rubbish) == 3:
                    key = rubbish[0][2:]
                    value = rubbish[1]
                    diamond[key] = value
                    key2 = key + "Growth"
                    value2 = rubbish[-1][:-2]
```

```
                    diamond[key2] = value2
                else:
                    key = rubbish[0][2:]
                    value = rubbish[-1][:-2]
                    diamond[key] = value
        #returning the dictionary to be initialized as a new dataframe and then␣
 ↪merged with original data set
        return diamond

#returning the correctly formated dataframe
test_df = pd.DataFrame(df["ceos_page_data"].apply(make_dict).values.tolist())
cleaned_df = df.join(test_df).drop(columns =␣
 ↪["ceos_page_data","ceo_name","","Unnamed: 0","index"])
cleaned_df.to_csv("Cleaned_F500data.csv")
```

3.2 2. WSJ Preparation and merge with Fortune 1000 data

[0]:
```
import pandas as pd
import numpy as np
import scipy as sc
import re

#Reading in the data that will be prepped for merging
df2 = pd.read_csv("wsjS&P.csv")
df1 = pd.read_csv("Cleaned_F500data.csv")
df1 = df1.rename(columns = {"companies":"Company"})
test2 = df2.copy()
test1 = df1.copy()

#Cleaning and parsing operation to be performed on individual strings in each␣
 ↪entry
clean_amp = lambda text: re.sub(r"&amp;","&",text)
clean_space = lambda text: text[:-1] if text[-1] == " " else text
clean_period = lambda text: re.sub(r"\."," ",text)
clean_white = lambda text: re.sub(r" ","",text)
clean_company = lambda text: re.sub(r"company$","",text)
clean_companies = lambda text: re.sub(r"companies$","",text)
clean_group = lambda text: re.sub(r"group$","",text)
clean_corp = lambda text: re.sub(r"corporation","corp",text)
lowercase = lambda text: str.lower(text)
clean_adobe = lambda text: "adobe" if text == "adobesystems" else text

#Cleaning operation loaded into a list to be iterated and applied to the␣
 ↪DataFrame column
clean_funcs =␣
 ↪[clean_amp,clean_space,clean_period,clean_white,clean_company,clean_companies,clean_group,l
```

```
#Applying the functions to the data column
for func in clean_funcs:
    test2["Company"] = test2["Company"].apply(func)
    test1["Company"] = test1["Company"].apply(func)

#Merging on prepped data
wsj_F500_merged = test1.merge(test2,on = "Company",how = "outer")
wsj_F500_merged.to_csv("wsj_F500_merged.csv")
```

3.3 3. Glassdoor and HBR Cleaning code

[0]:
```
### Clean ceo_rank_glassdoor.csv

df = pd.read_csv('ceo_rank_glassdoor.csv')
df = df.iloc[:,2:]
df.head(5)

df.loc[0,'GD_CEO']

df['GD_CEO'] = df['GD_CEO'].str.slice(start=1,stop=-1)
df['GD_Approval'] = df['GD_Approval'].str.slice(2,-1)
df['GD_company'] = df['GD_company'].str.slice(3,-3)
df['CEO Rank'] = np.arange(len(df))+1
## PLEASE FIX: change GD_Approval to float object
df['GD_Approval'] = df['GD_Approval'].str.slice(stop=2).astype(float)/100
df.head(5)

df.to_csv("Cleaned_ceo_rank_glassdoor_data.csv")

### Clean ceo_rank_hbr.csv

df = pd.read_csv('ceo_rank_hbr.csv')
string = np.asarray(df.iloc[27,6:])
df.iloc[27,5:-1] = string
df.head(5)

for i in df1.columns:
    print('Column name:',i,', Example of Values:',str(df.loc[0,i]))

df['HBR_CEO'] = df['HBR_CEO'].str.slice(start=1,stop=-1)
df['HBR_Company'] = df['HBR_Company'].str.slice(start=3,stop=-1)
df['HBR_Industry'] = df['HBR_Industry'].str.slice(start=2,stop=-4)
df['HBR_Country'] = df['HBR_Country'].str.slice(start=2,stop=-4)
df['HBR_YearStarted'] = df['HBR_YearStarted'].str.slice(start=2,stop=-4).
 ↪astype(int)
```

```
df['HBR_Insider'] = df['HBR_Insider'].str.slice(start=2,stop=-4)
df['HBR_Insider'] = df['HBR_Insider'].map({'YES': 1, 'NO': 0})
df['HBR_MBA'] = df['HBR_MBA'].str.slice(start=2,stop=-4)
df['HBR_MBA'] = df['HBR_MBA'].map({'YES': 1, 'NO': 0})
df['HBR_finrank'] = df['HBR_finrank'].str.slice(start=2,stop=-4).astype(int)
df['HBR_sustainalytics'] = df['HBR_sustainalytics'].str.slice(start=2,stop=-4).
↪astype(int)
df.loc[27,'HBR_csrhub'] = '> 432<'
df['HBR_csrhub'] = df['HBR_csrhub'].str.slice(start=2,stop=-1).astype(int)
## Put mean of entire HBR_csrhub column as null China value
df.loc[27,'HBR_csrhub'] = df['HBR_csrhub'].mean()
df.head(5)

df = df.iloc[:,1:]
df.head(5)

df.to_csv("Cleaned_ceo_rank_hbr.csv")
```

3.4 4. Final outer join of all data sources

[0]:
```
import pandas as pd
import numpy as np
import scipy as sc
import re

#first data set is already merged
df1 = pd.read_csv("wsj_F500_merged.csv")

#Final 2 remaining data sets to be merged onto the first
df2 = pd.read_csv('Cleaned_ceo_rank_glassdoor_data.csv')
df3 = pd.read_csv('Cleaned_ceo_rank_hbr.csv')

#Renaming to match columns names for the joining company names
df2 = df2.rename(columns = {"GD_company": "Company","CEO Rank": "GD CEO Rank"})
df2 = df2.iloc[:,1:]
df3 = df3.rename(columns = {"HBR_Company": "Company","Rank": "HBR CEO Rank"})
df3 = df3.iloc[:,1:]
test3 = df3.copy()
test2 = df2.copy()
test1 = df1.copy()

#Cleaning Functionalities to merge on company
clean_amp = lambda text: re.sub(r"&amp;","&",text)
clean_space = lambda text: text[:-1] if text[-1] == " " else text
clean_period = lambda text: re.sub(r"\."," ",text)
clean_white = lambda text: re.sub(r" ","",text)
```

```python
clean_company = lambda text: re.sub(r"company$","",text)
clean_companies = lambda text: re.sub(r"companies$","",text)
clean_group = lambda text: re.sub(r"group$","",text)
clean_corp = lambda text: re.sub(r"corporation","corp",text)
lowercase = lambda text: str.lower(text)
clean_adobe = lambda text: "adobe" if text == "adobesystems" else text
clean_accented_e = lambda text: re.sub(r"é","e",text)
clean_accented_o = lambda text: re.sub(r"ó","o",text)

#Operations to be performed in iterable
clean_funcs = [clean_amp,
               clean_space,
               clean_period,
               clean_white,
               clean_company,
               clean_companies,
               clean_group,
               lowercase,
               clean_adobe,
               clean_accented_e,
               clean_accented_o]

#Applying each cleaning functional on to the data frames to be merged
for func in clean_funcs:
    test2["Company"] = test2["Company"].apply(func)
    test3["Company"] = test3["Company"].apply(func)

#Final merging of all data sets
final_merge = test1.merge(test2, on = "Company", how = "outer")
final_merge = final_merge.merge(test3, on = "Company", how = "outer")
final_merge.drop(columns = ["Unnamed: 0", "Unnamed: 0_x"], inplace = True)
final_merge.to_csv("Nov_15_CheckpointFinalMerge.csv")
```

[12]:
```python
df = pd.read_csv("FinalDraftCleanedMergedCheckPointData.csv")
df.drop(columns = "Unnamed: 0", inplace = True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1262 entries, 0 to 1261
Data columns (total 52 columns):
Company                                           1262 non-null object
Market Value (M)                                  946 non-null float64
rank_change1000                                   914 non-null float64
employees                                         1000 non-null float64
rank_change500                                    454 non-null float64
CEO                                               962 non-null object
CEO Title                                         990 non-null object
Sector                                            990 non-null object
```

```
Industry_x                                     772 non-null object
HQ Location                                    990 non-null object
Website                                        0 non-null float64
Years on Fortune 500 List                      498 non-null float64
Employees                                      990 non-null float64
Revenues ($M)                                  990 non-null float64
Revenues ($M)Growth                            987 non-null float64
Profits ($M)                                   989 non-null float64
Profits ($M)Growth                             863 non-null float64
Assets ($M)                                    990 non-null float64
Assets ($M)Growth                              0 non-null float64
Total Stockholder Equity ($M)                  990 non-null float64
Total Stockholder Equity ($M)Growth            0 non-null float64
Profit as % of Revenues                        989 non-null float64
Profits as % of Assets                         989 non-null float64
Profits as % of Stockholder Equity             930 non-null float64
Earnings Per Share ($)                         937 non-null float64
EPS % Change (from 2017)                       813 non-null float64
EPS % Change (5 year annual rate)              701 non-null float64
EPS % Change (10 year annual rate)             531 non-null float64
Total Return to Investors (2018)               934 non-null float64
Total Return to Investors (5 year, annualized)    849 non-null float64
Total Return to Investors (10 year, annualized)   747 non-null float64
Market Cap (M)                                 500 non-null float64
Industry_y                                     500 non-null object
No_Directors                                   500 non-null float64
Median_age                                     500 non-null float64
Board_Independance                             500 non-null float64
Median_Tenure                                  500 non-null float64
Median_pay                                     500 non-null float64
women_on_board                                 500 non-null float64
GD_CEO                                         100 non-null object
GD_Approval                                    100 non-null float64
GD CEO Rank                                    100 non-null float64
HBR CEO Rank                                   100 non-null float64
HBR_CEO                                        100 non-null object
HBR_Industry                                   100 non-null object
HBR_Country                                    100 non-null object
HBR_YearStarted                                100 non-null float64
HBR_Insider                                    100 non-null float64
HBR_MBA                                        100 non-null float64
HBR_finrank                                    100 non-null float64
HBR_sustainalytics                             100 non-null float64
HBR_csrhub                                     100 non-null float64
dtypes: float64(41), object(11)
memory usage: 512.8+ KB
```