ANALYSIS OF U.S. STOCK MARKET FLUCTUATION IN 2020

Financial and Accounting Analysis of the Stock Market Plunge and Recovery in 2020 During Covid-19 Pandemic

- A financial/accounting analysis of a large set of U.S. companies affected by the stock market shocks in 2020.
- A predictive analytics exercise to explain what types of companies did the best/worst during the **initial COVID shock (January-March 2020)**.
- A predictive analytics exercise to explain what types of companies did the best/worst during the **market recovery (April-December 2020)**.

Environment Setup

In [ ]:

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Mounted at /content/gdrive/
```

In [ ]:

```
import os
root_dir = "/content/gdrive/MyDrive/Colab Notebooks/"
project_folder = "BA870 Finance/Project"
# change the OS to use your project folder as the working directory
os.chdir(root_dir + project_folder)
# print current working directory
os.getcwd()
```

Out[ ]:

```
'/content/gdrive/MyDrive/Colab Notebooks/BA870 Finance/Project'
```

In [ ]:

```
!pip install yfinance
```

In [ ]:

```
!pip install transformers
```

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import yfinance as yf
import requests
from bs4 import BeautifulSoup
import torch
import transformers as ppb
import statsmodels.api as sm
from scipy.stats.mstats import winsorize
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.dummy import DummyClassifier
from sklearn.metrics import confusion_matrix
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Preprocessing Stock Data

Monthly Returns of Stocks and S&P 500 in 2020

`monthlystock.csv` contains the stock data for all the U.S. companies in WRDS CRSP database.

- `PERMCO` is the unique identifier of a company in CRSP.
- `date` is the date of the last trading day of each month in 2020.
- `ticker` is the ticker for each stock.
- `price` is the closing price on the last trading day in each month in 2020.
- `ret` is the holding period (monthly) return for each stock.

In [ ]:

```
stock = pd.read_csv('monthlystock.csv')
```

In [ ]:

```
# clean up the columns
stock.rename(columns={'TICKER': 'ticker', 'PRC': 'price', 'RET': 'ret'}, inplace=True)
```

```
stock.date = pd.to_datetime(stock.date, format="%Y%m%d")
stock['month'] = pd.DatetimeIndex(stock.date).month
```

Use the data from Yahoo Finance to calculate the monthly return on S&P 500 in 2020.

In [ ]:

```
# extract S&P 500 prices from Yahoo Finance
df_yahoo = yf.download('^GSPC', start="2019-12-31", end="2021-01-01", group_by='ticker')
dates = ['2019-12-31']
dates.extend(list(stock.date.astype(str).unique()))
prices = round(df_yahoo.loc[dates, 'Close'], 2).to_list()
```

```
[*********************100%***********************]  1 of 1 completed
```

In [ ]:

```
# calculate monthly returns in 2020
returns = []

for i, v in enumerate(prices[1:]):
    this = v
    last = prices[i]
    ret = round((this-last)/last, 6)
    returns.append(ret)

sp = pd.DataFrame({'month': [i for i in range(1, 13)], 'SPret': returns})
sp.head(1)
```

Out[ ]:

	month	SPret
0	1	-0.001628

Store the monthly returns on S&P 500 under `SPret` in the dataframe.

In [ ]:

```
stock = stock.merge(sp, on='month', how='left')
```

In [ ]:

```
# drop observations with missing values in ticker or monthly return
stock.dropna(subset=['ticker', 'ret'], inplace=True)
stock.reset_index(drop=True, inplace=True)
```

There are 1457 companies that do not have valid returns for all 12 months.

In [ ]:
```
tmp = stock.ticker.value_counts()
tmp.lt(12).sum()
```

Out[ ]:
```
1457
```

In [ ]:
```
# remove 1457 observations
print("Number of unique tickers:", stock.ticker.nunique())
stock = stock[stock.ticker.isin(tmp.index[tmp.lt(12)])==False]
print("Number of unique tickers:", stock.ticker.nunique())
```

```
Number of unique tickers: 8399
Number of unique tickers: 6942
```

There are 19 companies that have two sets (24) of monthly returns.

In [ ]:
```
tmp = stock.ticker.value_counts()
tmp.value_counts()
```

Out[ ]:
```
12    6923
24      19
Name: ticker, dtype: int64
```

Cross examine two sets of stock prices from CRSP with those listed on Yahoo Finance and only keep the ones that match.

In [ ]:
```
tics = " ".join(tmp.index[tmp.gt(12)].to_list())
df_yahoo = yf.download(tics, start="2019-12-31", end="2021-01-01", group_by='ticker')
dates = ['2020-01-31', '2020-02-28', '2020-03-31', '2020-04-30', '2020-05-29', '2020-06-30',
         '2020-07-31', '2020-08-31', '2020-09-30', '2020-10-30', '2020-11-30', '2020-12-31']

for i in tmp.index[tmp.gt(12)]:
    if df_yahoo[i].dropna().empty:
        continue
    else:
        try:
            prices = round(df_yahoo[i].loc[dates, 'Close'], 2).to_list()
            stock.loc[stock.ticker==i, 'price'] = stock[stock.ticker==i]['price'].apply(lambda x: x if round(x, 2) in prices else
            stock.dropna(subset=['price'], inplace=True)
        except:
            pass
```

```
[*********************100%***********************]  19 of 19 completed
```

```
7 Failed downloads:
- BF: No data found for this date range, symbol may be delisted
- CRD: No data found for this date range, symbol may be delisted
- LGF: No data found for this date range, symbol may be delisted
- JW: No data found for this date range, symbol may be delisted
- RDS: No data found for this date range, symbol may be delisted
- BRK: No data found for this date range, symbol may be delisted
- AKO: No data found for this date range, symbol may be delisted
```

There are still 7 companies with two sets of monthly data because their stock data cannot be found on Yahoo Finance. These companies will be dropped later when we merge the stock data with the accounting data.

In [ ]:

```
tmp = stock.ticker.value_counts()
tmp.value_counts()
```

Out[ ]:

```
12    6935
24       7
Name: ticker, dtype: int64
```

Market Betas in 2019

`betas19.csv` contains the market beta for all the U.S. companies in WRDS CRSP database.

- `PERMCO` is the unique identifier of a company in CRSP.
- `beta19` is the market beta of a stock in 2019.

In [ ]:

```
betas = pd.read_csv('betas19.csv')
```

In [ ]:

```
# clean up the dataframe
betas.rename(columns={'betav': 'beta19'}, inplace=True)
betas.head(1)
```

Out[ ]:

	PERMNO	beta19
0	10028	0.24393

In [ ]:

```
# drop duplicated observations
betas.drop_duplicates(inplace=True)
```

```
betas.reset_index(drop=True, inplace=True)
```

There are 25 observations with two betas.

In [ ]:
```
tmp = betas.PERMNO.value_counts()
tmp.value_counts()
```

Out[ ]:
```
1    6097
2      25
Name: PERMNO, dtype: int64
```

After examining the betas of these 25 observations, we found that all the duplicated betas are equal to 0. Drop these 25 zero betas and keep the 25 non-zeros betas.

In [ ]:
```
betas[(betas.PERMNO.isin(tmp.index[tmp.gt(1)])) & (betas.beta19==0)].shape
```

Out[ ]:
```
(25, 2)
```

In [ ]:
```
# drop 25 zero betas
tmp = betas[(betas.PERMNO.isin(tmp.index[tmp.gt(1)])) & (betas.beta19==0)].index
betas = betas[~betas.index.isin(tmp)]
```

In [ ]:
```
# merge the dataframes
stock = stock.merge(betas, on='PERMNO', how='left')
```

In [ ]:
```
stock = stock[['month', 'ticker', 'ret', 'SPret', 'beta19']].reset_index(drop=True)
print("Number of unique tickers:", stock.ticker.nunique())
stock.head(2)
```

```
Number of unique tickers: 6942
```

Out[ ]:

	month	ticker	ret	SPret	beta19
0	1	JJSF	-0.100016	-0.001628	0.01282
1	2	JJSF	-0.030270	-0.084110	0.01282

Preprocessing Accounting Data

Import data about all the U.S. companies in WRDS Compustat database.

- `des` contains the descriptions of 2852 companies in the Russell 3000 Index scraped from Yahoo Finance.
- `wrds` contains the company financial data downloaded from Compustat (Fiscal Year 2018 and 2019).

In [ ]:

```
des = pd.read_csv('2-3 stock_des.csv')
wrds = pd.read_csv('compustat1819.csv')
```

In [ ]:

```
# keep 4379 companies that have stock data from CRSP
print('Number of unique tickers:', wrds.tic.nunique())
wrds = wrds[wrds.tic.isin(stock.ticker.unique())]
print('Number of unique tickers:', wrds.tic.nunique())
```

```
Number of unique tickers: 8209
Number of unique tickers: 4379
```

In [ ]:

```
# keep 4348 companies that have accounting data from 2018 and 2019
wrds = wrds.groupby('tic').filter(lambda x: x['fyear'].count() == 2)
print('Number of unique tickers:', wrds.tic.nunique())
```

```
Number of unique tickers: 4348
```

In [ ]:

```
# clean up the columns
wrds.rename(columns={'tic': 'ticker'}, inplace=True)
wrds.fyear = wrds.fyear.astype(str)
wrds[['gsector', 'ggroup', 'gind', 'gsubind']] = wrds[['gsector', 'ggroup', 'gind', 'gsubind']].astype(int)
```

In [ ]:

```
# merge the dataframes
df = wrds.merge(des, on='ticker', how='left')
```

In [ ]:

```
# organize the dataframe
df = df.set_index(['ticker', 'gsector', 'ggroup', 'gind', 'gsubind', 'naics', 'sic', 'spcsrc', 'description', 'fyear']).unstack()
# label the year in column names
df.columns = [col[0]+col[1][2:] for col in df.columns]
```

```
df.reset_index(inplace=True)
df.head(1)
```

Out[ ]:

	ticker	gsector	ggroup	gind	gsubind	naics	sic	spcsrc	description	at18	at19	act18	act19	invt18	invt19	lt18	lt
0	A	35	3520	352030	35203010	334516.0	3826.0	B	Agilent Technologies, Inc. provides applicatio...	8541.0	9452.0	3848.0	3189.0	638.0	679.0	3970.0	470

Organizing GICS Classification

The GICS structure consists of 11 sectors, 24 industry groups, 69 industries.

In [ ]:

```
# scrape GICS classification from Wikipedia
wikiurl="https://en.wikipedia.org/wiki/Global_Industry_Classification_Standard#Classification[1]"
response=requests.get(wikiurl)
print(response.status_code)
```

```
200
```

In [ ]:

```
soup = BeautifulSoup(response.text, 'html.parser')
indiatable = soup.find('table', {'class':"wikitable"})
wiki = pd.read_html(str(indiatable))
wiki = pd.DataFrame(wiki[0])
wiki.head(1)
```

Out[ ]:

	Sector	Sector.1	Industry Group	Industry Group.1	Industry	Industry.1	Sub-Industry	Sub-Industry.1
0	10	Energy	1010	Energy	101010	Energy Equipment & Services	10101010	Oil & Gas Drilling

Create a GICS sector dictionary `gsectors` where `{"sector ID": "sector name"}` .

In [ ]:

```
sectorID = wiki['Sector'].unique().tolist()
sector = wiki['Sector.1'].unique().tolist()
gsectors = {sectorID[i]: sector[i] for i in range(len(sectorID))}
```

Create a GICS group dictionary `ggroups` where `{"group ID": "group name"}` .

In [ ]:

```
groupID = wiki['Industry Group'].unique().tolist()
group = wiki['Industry Group.1'].unique().tolist()
ggroups = {groupID[i]: group[i] for i in range(len(groupID))}
```

Create a GICS industry dictionary `ginds` where `{"industry ID": "industry name"}` .

In [ ]:

```
indID = wiki['Industry'].unique().tolist()
industry = wiki['Industry.1'].unique().tolist()
ginds = {indID[i]: industry[i] for i in range(len(indID))}
```

Replace `gsector` , `ggroup` and `gind` IDs with sector names, group names and industry names.

In [ ]:

```
df.gsector = df.gsector.apply(lambda x: gsectors[x])
df.ggroup = df.ggroup.apply(lambda x: ggroups[x])
df.gind = df.gind.apply(lambda x: ginds[x])
```

Organizing NAICS Classification Code

The NAICS codes can be grouped into 20 sectors using the first two digits.

In [ ]:

```
# scrape SIC classification from Wikipedia
import requests
from bs4 import BeautifulSoup

wikiurl="https://en.wikipedia.org/wiki/North_American_Industry_Classification_System#Codes"
response=requests.get(wikiurl)
print(response.status_code)
```

```
200
```

In [ ]:

```
soup = BeautifulSoup(response.text, 'html.parser')
indiatable = soup.find_all('table', {'class':"wikitable"})
wiki = pd.read_html(str(indiatable))
wiki = pd.DataFrame(wiki[2])
```

```
print(wiki.shape)
wiki.head(1)
```

(20, 3)

Out[ ]:

	Sector #	Description	Note
0	11	Agriculture, Forestry, Fishing and Hunting	NaN

Create a NAICS sector dictionary `nsectors` where `{"nsector ID": "nsector name"}` .

In [ ]:
```
wiki['Sector #'] = wiki['Sector #'].apply(lambda x: x[:2])
wiki['Sector #'] = wiki['Sector #'].astype(int)
```

In [ ]:
```
nsectorID = wiki['Sector #'].unique().tolist()
nsector = wiki['Description'].unique().tolist()
nsectors = {nsectorID[i]: nsector[i] for i in range(len(nsectorID))}
```

Create a new column `nsector` which stores the name of the NAICS sector that each company belongs to.

In [ ]:
```
df.insert(6, 'nsector', df.naics.astype(str))
df.nsector = df.nsector.apply(lambda x: x[:2])
df.nsector = np.where(df.nsector.astype(float)<21, '11', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==21, '21', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==22, '22', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==23, '23', df.nsector)
df.nsector = np.where((df.nsector.astype(float)>=31)&(df.nsector.astype(float)<41), '31', df.nsector)
df.nsector = np.where((df.nsector.astype(float)>=41)&(df.nsector.astype(float)<44), '41', df.nsector)
df.nsector = np.where((df.nsector.astype(float)>=44)&(df.nsector.astype(float)<48), '44', df.nsector)
df.nsector = np.where((df.nsector.astype(float)>=48)&(df.nsector.astype(float)<51), '48', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==51, '51', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==52, '52', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==53, '53', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==54, '54', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==55, '55', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==56, '56', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==61, '61', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==62, '62', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==71, '71', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==72, '72', df.nsector)
df.nsector = np.where(df.nsector.astype(float)==81, '81', df.nsector)
```

```
df.nsector = np.where(df.nsector.astype(float)>=91, '91', df.nsector)
df.nsector = df.nsector.apply(lambda x: nsectors[int(x)])
```

Organizing SIC Classification Code

The SIC codes can be grouped into 12 industry groups.

In [ ]:

```
# scrape SIC classification from Wikipedia
import requests
from bs4 import BeautifulSoup

wikiurl="https://en.wikipedia.org/wiki/Standard_Industrial_Classification#Range"
response=requests.get(wikiurl)
print(response.status_code)
```

```
200
```

In [ ]:

```
soup = BeautifulSoup(response.text, 'html.parser')
indiatable = soup.find('table', {'class':"wikitable"})
wiki = pd.read_html(str(indiatable))
wiki = pd.DataFrame(wiki[0])
print(wiki.shape)
wiki.head(1)
```

```
(12, 2)
```

Out[ ]:

	Range of SIC Codes	Division
0	0100-0999	Agriculture, Forestry and Fishing

Create a SIC industry group dictionary `sgroups` where `{"sgroup ID": "sgroup name"}` .

In [ ]:

```
sgroupID = [1, 10, 15, 18, 20, 40, 50, 52, 60, 70, 91, 99]
sgroup = wiki['Division'].unique().tolist()
sgroups = {sgroupID[i]: sgroup[i] for i in range(len(sgroupID))}
```

Create a new column `sgroup` which stores the name of the SIC industry group that each company belongs to.

In [ ]:

```
df.insert(8, 'sgroup', df.sic.astype(str))
```

```
df.sgroup = np.where(df.sic<1000, '1', df.sgroup)
df.sgroup = np.where(df.sic.between(1000, 1499), '10', df.sgroup)
df.sgroup = np.where(df.sic.between(1500, 1799), '15', df.sgroup)
df.sgroup = np.where(df.sic.between(1800, 1999), '18', df.sgroup)
df.sgroup = np.where(df.sic.between(2000, 3999), '20', df.sgroup)
df.sgroup = np.where(df.sic.between(4000, 4999), '40', df.sgroup)
df.sgroup = np.where(df.sic.between(5000, 5199), '50', df.sgroup)
df.sgroup = np.where(df.sic.between(5200, 5999), '52', df.sgroup)
df.sgroup = np.where(df.sic.between(6000, 6799), '60', df.sgroup)
df.sgroup = np.where(df.sic.between(7000, 8999), '70', df.sgroup)
df.sgroup = np.where(df.sic.between(9100, 9729), '91', df.sgroup)
df.sgroup = np.where(df.sic.between(9900, 9999), '99', df.sgroup)
df.sgroup = df.sgroup.apply(lambda x: sgroups[int(x)])
```

In [ ]:

```
# take a look at the dataframe
print('Shape:', df.shape)
print('Number of unique tickers:', df.ticker.nunique())
df.head(1)
```

```
Shape: (4348, 41)
Number of unique tickers: 4348
```

Out[ ]:

	ticker	gsector	ggroup	gind	gsubind	naics	nsector	sic	sgroup	spcsrc	description	at18	at19	act
0	A	Health Care	Pharmaceuticals, Biotechnology & Life Sciences	Life Sciences Tools & Services	35203010	334516.0	Manufacturing	3826.0	Manufacturing	B	Agilent Technologies, Inc. provides applicatio...	8541.0	9452.0	384

Main Dataframe Setup

Calculate the 3-month and 9-month returns for the 4348 U.S. companies which we have accounting data for.

In [ ]:

```
# keep only the companies that we have accounting data for
stock = stock[stock.ticker.isin(df.ticker)]
print('Number of unique tickers:', stock.ticker.nunique())
stock.head(2)
```

```
Number of unique tickers: 4348
```

Out[ ]:

	month	ticker	ret	SPret	beta19
0	1	JJSF	-0.100016	-0.001628	0.01282
1	2	JJSF	-0.030270	-0.084110	0.01282

Split the `stock` dataframe into two dataframes to calculate:

- `RetEarly2020` the 3-month return for each stock during the initial COVID shock (from January to March 2020).
- `RetLate2020` the 9-month return for each stock during the market recovery (from April to December 2020).
- `SPEarly2020` the 3-month return for S&P 500 during the initial COVID shock (from January to March 2020).
- `SPLate2020` the 9-month return for S&P 500 during the market recovery (from April to December 2020).

In [ ]:

```
early = stock[stock.month.isin([1, 2, 3])].reset_index(drop=True)
late = stock[stock.month.isin([1, 2, 3])==False].reset_index(drop=True)
```

In [ ]:

```
# calculate 3-month and 9-month rolling returns for each stock
early['RetEarly2020'] = early['ret'].rolling(3).agg(lambda x: (x+1).prod()-1)
late['RetLate2020'] = late['ret'].rolling(9).agg(lambda x: (x+1).prod()-1)
```

In [ ]:

```
# calculate 3-month and 9-month returns for S&P 500
SPEarly2020 = (early.iloc[:3]['SPret']+1).prod()-1
SPLate2020 = (late.iloc[:9]['SPret']+1).prod()-1
```

In [ ]:

```
# keep only the 3-month returns calculated in March and the 9-month returns calculated in December
early = early[early.month==3].reset_index(drop=True)[['ticker', 'RetEarly2020']]
late = late[late.month==12].reset_index(drop=True)[['ticker', 'RetLate2020']]
```

In [ ]:

```
early.head(1)
```

Out[ ]:

	ticker	RetEarly2020
0	JJSF	-0.340234

In [ ]:
```
late.head(1)
```

Out[ ]:

	ticker	RetLate2020
0	JJSF	0.30034

Merge the tickers, 3-month returns, and 9-month returns into one dataframe `stock2`.

In [ ]:
```
stock2 = early.merge(late, on='ticker')
# insert the 3-month and 9-month returns for S&P 500
stock2['SPEarly2020'] = SPEarly2020
stock2['SPLate2020'] = SPLate2020
stock2['beta19'] = list(stock.beta19)[::12]
print(stock2.shape)
stock2.head()
```

```
(4348, 6)
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	SPEarly2020	SPLate2020	beta19
0	JJSF	-0.340234	0.300340	-0.20001	0.453255	0.01282
1	ELA	0.866665	1.063494	-0.20001	0.453255	0.24393
2	PLXS	-0.290876	0.433469	-0.20001	0.453255	1.27923
3	RMCF	-0.471810	-0.156250	-0.20001	0.453255	-0.01815
4	HNGR	-0.435712	0.411425	-0.20001	0.453255	0.76976

Merge the stock data and the accounting data.

In [ ]:
```
df = stock2.merge(df, on='ticker')
print('Shape:', df.shape)
print('Number of unique tickers:', df.ticker.nunique())
df.head(1)
```

```
Shape: (4348, 46)
Number of unique tickers: 4348
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	SPEarly2020	SPLate2020	beta19	gsector	ggroup	gind	gsubind	naics	nsector	si

	ticker	RetEarly2020	RetLate2020	SPEarly2020	SPLate2020	beta19	gsector	ggroup	gind	gsubind	naics	nsector	si
0	JJSF	-0.340234	0.30034	-0.20001	0.453255	0.01282	Consumer Staples	Food, Beverage & Tobacco	Food Products	30202030	311812.0	Manufacturing	2050.

Calculating Financial Ratios

We have the following data for 4348 U.S. companies.

- `at` : Total Assets
- `act` : Total Current Assets
- `invt` : Total Inventories
- `lt` : Total Liabilities
- `lct` : Total Current Liabilities
- `ap` : Accounts Payable
- `teq` : Total Stockholders' Equity
- `re` : Retained Earnings

- `sale` : Net Sales
- `cogs` : Cost of Goods Sold
- `xopr` : Total Operating Expenses
- `ni` : Net Income

- `oancf` : Net Operating Activities Cash Flow
- `ivncf` : Net Investing Activities Cash Flow
- `fincf` : Net Financing Activities Cash Flow

Keep only `at` and `teq` from 2018 to calculate average assets and equity.

In [ ]:

```
df = df.drop(['act18', 'invt18', 'lt18', 'lct18', 'ap18', 're18', \
```

```
            'sale18', 'cogs18', 'xopr18', 'ni18', 'oancf18', 'ivncf18', 'fincf18'], axis=1)
```

Drop 217 companies with 0 in `sale` in 2019.

In [ ]:

```
df = df[df.sale19!=0]
print('Number of unique tickers:', df.ticker.nunique())
```

```
Number of unique tickers: 4131
```

In [ ]:

```
# investigate missing values
tmp = pd.DataFrame({'Number of companies with NA': (df.isna().sum()).sort_values(ascending=False)})
tmp[tmp['Number of companies with NA']>0].T
```

Out[ ]:

	description	spcsrc	act19	lct19	re19	beta19	invt19	ap19	ivncf19	fincf19	oancf19	sale19	cogs19	xopr19	ni19
Number of companies with NA	1566	1553	926	923	107	92	54	26	12	12	12	1	1	1	1

Calculate accounting ratios for 2019.

In [ ]:

```
def calculate(year):
    previous = str(int(year)-1)
    df['avgat'+year] = (df['at'+previous] + df['at'+year])/2  # average assets
    df['avgteq'+year] = (df['teq'+previous] + df['teq'+year])/2  # average equity

    df['roa'+year] = df['ni'+year] / df['avgat'+year]  # return on assets
    df['atr'+year] = df['sale'+year] / df['avgat'+year]  # asset turnover ratio
    df['ros'+year] = df['ni'+year] / df['sale'+year]  # return on sales

    df['roe'+year] = df['ni'+year] / df['avgteq'+year]  # return on equity
    df['emulti'+year] = df['avgat'+year] / df['avgteq'+year]  # equity multiplier

    df['ai'+year] = df['at'+year] / df['sale'+year]  # asset intensity
    df['gmargin'+year] = df['sale'+year] - df['cogs'+year]  # gross margin

calculate('19')
```

In [ ]:

```
# investigate infinite values
tmp = pd.DataFrame({'INF #': df.isin([np.inf, -np.inf]).sum().sort_values(ascending=False)})
```

```
tmp[tmp['INF #']>0].T
```

Out[ ]:

INF #

In [ ]:

```
# find companies that have missing values for any of the 9 ratios we calculated
df[(df.avgat19.isna()) | (df.avgteq19.isna()) | (df.roa19.isna()) | (df.atr19.isna()) |
   (df.ros19.isna()) | (df.roe19.isna()) | (df.emulti19.isna()) | (df.ai19.isna()) | (df.gmargin19.isna())]
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	SPEarly2020	SPLate2020	beta19	gsector	ggroup	gind	gsubind	naics	nsector	sic
999	GYRO	-0.225464	0.138697	-0.20001	0.453255	0.69856	Real Estate	Real Estate	Real Estate Management & Development	60102020	531120.0	Real Estate and Rental and Leasing	6512.0

Drop this company from the dataframe because it has too many missing values.

In [ ]:

```
df.drop(999, inplace=True)
df.reset_index(drop=True, inplace=True)
```

In [ ]:

```
# take a look at the dataframe
print('Shape:', df.shape)
print('Number of unique tickers:', df.ticker.nunique())
df.head(1)
```

```
Shape: (4130, 42)
Number of unique tickers: 4130
```

Out[ ]:

ticker	RetEarly2020	RetLate2020	SPEarly2020	SPLate2020	beta19	gsector	ggroup	gind	gsubind	naics	nsector	si

	ticker	RetEarly2020	RetLate2020	SPEarly2020	SPLate2020	beta19	gsector	ggroup	gind	gsubind	naics	nsector	si
0	JJSF	-0.340234	0.30034	-0.20001	0.453255	0.01282	Consumer Staples	Food, Beverage & Tobacco	Food Products	30202030	311812.0	Manufacturing	2050.

In [ ]:

```
# # export into a csv file
# df.to_csv('maindf.csv', index=False)
```

1. Explaining Fluctuation Using Industry Indicators

Does industry or sector explain variation in stock returns for early and late 2020?

In [ ]:

```
df = pd.read_csv('maindf.csv')
```

`RetEarly2020` and `RetLate2020` are the variables whose variation is what we're trying to explain.

We have 8 types of industry classifications: `gsector` , `ggroup` , `gind` , `gsubind` , `naics` , `nsector` , `sic` , and `sgroup` .

In [ ]:

```
# make sure there's no missing values
pd.DataFrame({'Number of unique values': df[df.columns[6:14]].nunique(),
              'Number of companies with NA': df[df.columns[6:14]].isna().sum()}).T
```

Out[ ]:

	gsector	ggroup	gind	gsubind	naics	nsector	sic	sgroup
Number of unique values	11	24	69	157	612	19	378	10
Number of companies with NA	0	0	0	0	0	0	0	0

Using 11 GICS Sectors

Create binary indicators for the 11 GICS Sectors.

In [ ]:

```python
# extract relevant data from the main dataframe
df2 = df[['ticker', 'RetEarly2020', 'RetLate2020', 'gsector']].copy()
df2 = pd.get_dummies(df2, columns=['gsector'], prefix='', prefix_sep='')
df2.head(1)
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	Communication Services	Consumer Discretionary	Consumer Staples	Energy	Financials	Health Care	Industrials	Information Technology	Materials
0	JJSF	-0.340234	0.30034	0	0	1	0	0	0	0	0	0

In [ ]:

```python
# regress RetEarly2020 on 11 indicators
Y = df2['RetEarly2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg1 = sm.OLS(Y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           RetEarly2020   R-squared:                       0.077
Model:                            OLS   Adj. R-squared:                  0.075
Method:                 Least Squares   F-statistic:                     34.37
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           5.62e-65
Time:                        23:00:26   Log-Likelihood:                -1778.5
No. Observations:                4130   AIC:                             3579.
Df Residuals:                    4119   BIC:                             3649.
Df Model:                          10
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
Communication Services    -0.2678      0.027     -9.931      0.000      -0.321      -0.215
Consumer Discretionary    -0.3811      0.017    -22.050      0.000      -0.415      -0.347
Consumer Staples          -0.1920      0.030     -6.393      0.000      -0.251      -0.133
Energy                    -0.5521      0.022    -24.697      0.000      -0.596      -0.508
Financials                -0.3459      0.014    -25.381      0.000      -0.373      -0.319
Health Care               -0.1375      0.014     -9.556      0.000      -0.166      -0.109
Industrials               -0.3307      0.016    -20.368      0.000      -0.363      -0.299
Information Technology    -0.2290      0.016    -14.617      0.000      -0.260      -0.198
Materials                 -0.3581      0.025    -14.185      0.000      -0.408      -0.309
Real Estate               -0.3344      0.025    -13.216      0.000      -0.384      -0.285
Utilities                 -0.1787      0.038     -4.674      0.000      -0.254      -0.104
```

```
==============================================================================
Omnibus:                     8211.017   Durbin-Watson:                   1.950
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         39694157.648
Skew:                          15.596   Prob(JB):                         0.00
Kurtosis:                     482.265   Cond. No.                         2.81
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```python
# find the highest and lowest returns
print("Highest:", reg1.params.idxmax(), reg1.params.max())
print("Lowest:", reg1.params.idxmin(), reg1.params.min())
```

```
Highest: Health Care -0.13749028355529064
Lowest: Energy -0.5520537609263416
```

In [ ]:

```python
# regress RetLate2020 on 11 indicators
Y = df2['RetLate2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg2 = sm.OLS(Y, X).fit()
print(reg2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.058
Model:                            OLS   Adj. R-squared:                  0.055
Method:                 Least Squares   F-statistic:                     25.18
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           7.27e-47
Time:                        23:00:26   Log-Likelihood:                -7624.8
No. Observations:                4130   AIC:                         1.527e+04
Df Residuals:                    4119   BIC:                         1.534e+04
Df Model:                          10
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
Communication Services     0.7275      0.111      6.549      0.000       0.510       0.945
Consumer Discretionary     1.7427      0.071     24.480      0.000       1.603       1.882
Consumer Staples           0.5762      0.124      4.658      0.000       0.334       0.819
Energy                     0.9329      0.092     10.133      0.000       0.752       1.113
Financials                 0.5348      0.056      9.527      0.000       0.425       0.645
Health Care                0.7444      0.059     12.561      0.000       0.628       0.861
Industrials                1.0005      0.067     14.962      0.000       0.869       1.132
Information Technology     1.1785      0.065     18.265      0.000       1.052       1.305
```

```
Materials                 1.0635      0.104     10.229      0.000       0.860       1.267
Real Estate               0.4483      0.104      4.302      0.000       0.244       0.653
Utilities                 0.3768      0.157      2.393      0.017       0.068       0.686
==============================================================================
Omnibus:                     5419.211   Durbin-Watson:                   1.952
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1329661.474
Skew:                           7.241   Prob(JB):                         0.00
Kurtosis:                      89.701   Cond. No.                         2.81
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# find the highest and lowest returns
print("Highest:", reg2.params.idxmax(), reg2.params.max())
print("Lowest:", reg2.params.idxmin(), reg2.params.min())
```

```
Highest: Consumer Discretionary 1.7427338131777348
Lowest: Utilities 0.37683366165013943
```

Using 24 GICS Groups

Create binary indicators for the 24 GICS Groups.

In [ ]:

```
df2 = df[['ticker', 'RetEarly2020', 'RetLate2020', 'ggroup']].copy()
df2 = pd.get_dummies(df2, columns=['ggroup'], prefix='', prefix_sep='')
df2.head(1)
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	Automobiles & Components	Banks	Capital Goods	Commercial & Professional Services	Communication Services	Consumer Durables & Apparel	Consumer Services	Diversified Financials	Energy	F S Re
0	JJSF	-0.340234	0.30034	0	0	0	0	0	0	0	0	0	

In [ ]:

```
# regress RetEarly2020 on 24 indicators
Y = df2['RetEarly2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
```

```
reg1 = sm.OLS(Y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          RetEarly2020   R-squared:                       0.083
Model:                           OLS   Adj. R-squared:                  0.078
Method:                Least Squares   F-statistic:                     16.23
Date:               Mon, 05 Jul 2021   Prob (F-statistic):           6.78e-62
Time:                       23:00:26   Log-Likelihood:                -1764.3
No. Observations:               4130   AIC:                             3577.
Df Residuals:                   4106   BIC:                             3728.
Df Model:                         23
Covariance Type:           nonrobust
============================================================================================================
                                               coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------------------
Automobiles & Components                    -0.4113      0.053     -7.739      0.000      -0.515      -0.307
Banks                                       -0.3593      0.018    -19.509      0.000      -0.395      -0.323
Capital Goods                               -0.3240      0.021    -15.386      0.000      -0.365      -0.283
Commercial & Professional Services          -0.3219      0.033     -9.714      0.000      -0.387      -0.257
Communication Services                      -0.1765      0.055     -3.218      0.001      -0.284      -0.069
Consumer Durables & Apparel                 -0.3872      0.034    -11.305      0.000      -0.454      -0.320
Consumer Services                           -0.3969      0.031    -12.893      0.000      -0.457      -0.337
Diversified Financials                      -0.3698      0.024    -15.240      0.000      -0.417      -0.322
Energy                                      -0.5521      0.022    -24.743      0.000      -0.596      -0.508
Food & Staples Retailing                    -0.0557      0.072     -0.778      0.436      -0.196       0.085
Food, Beverage & Tobacco                    -0.2354      0.039     -6.036      0.000      -0.312      -0.159
Health Care Equipment & Services            -0.1251      0.023     -5.401      0.000      -0.170      -0.080
Household & Personal Products               -0.1845      0.062     -2.976      0.003      -0.306      -0.063
Insurance                                   -0.2401      0.036     -6.613      0.000      -0.311      -0.169
Materials                                   -0.3581      0.025    -14.212      0.000      -0.407      -0.309
Media & Entertainment                       -0.2968      0.031     -9.606      0.000      -0.357      -0.236
Pharmaceuticals, Biotechnology & Life Sciences -0.1452   0.018     -7.934      0.000      -0.181      -0.109
Real Estate                                 -0.3344      0.025    -13.241      0.000      -0.384      -0.285
Retailing                                   -0.3515      0.030    -11.648      0.000      -0.411      -0.292
Semiconductors & Semiconductor Equipment    -0.2201      0.037     -5.917      0.000      -0.293      -0.147
Software & Services                         -0.1956      0.022     -8.892      0.000      -0.239      -0.152
Technology Hardware & Equipment             -0.2870      0.028    -10.349      0.000      -0.341      -0.233
Transportation                              -0.3664      0.039     -9.291      0.000      -0.444      -0.289
Utilities                                   -0.1787      0.038     -4.683      0.000      -0.254      -0.104
==============================================================================
Omnibus:                     8234.527   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         40472003.429
Skew:                          15.695   Prob(JB):                         0.00
Kurtosis:                     486.945   Cond. No.                         3.91
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```python
# find the highest and lowest returns
print("Highest:", reg1.params.idxmax(), reg1.params.max())
print("Lowest:", reg1.params.idxmin(), reg1.params.min())
```

```
Highest: Food & Staples Retailing -0.055730434846246284
Lowest: Energy -0.5520537609263416
```

In [ ]:

```python
# regress RetLate2020 on 24 indicators
Y = df2['RetLate2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg2 = sm.OLS(Y, X).fit()
print(reg2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.077
Model:                            OLS   Adj. R-squared:                  0.072
Method:                 Least Squares   F-statistic:                     14.87
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           5.11e-56
Time:                        23:00:27   Log-Likelihood:                -7582.1
No. Observations:                4130   AIC:                         1.521e+04
Df Residuals:                    4106   BIC:                         1.536e+04
Df Model:                          23
Covariance Type:            nonrobust
==================================================================================================
                                     coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
Automobiles & Components           2.2748      0.217     10.464      0.000       1.849       2.701
Banks                              0.4198      0.075      5.573      0.000       0.272       0.568
Capital Goods                      1.1853      0.086     13.758      0.000       1.016       1.354
Commercial & Professional Services 0.7080      0.136      5.223      0.000       0.442       0.974
Communication Services             0.3781      0.224      1.685      0.092      -0.062       0.818
Consumer Durables & Apparel        1.7160      0.140     12.249      0.000       1.441       1.991
Consumer Services                  1.0347      0.126      8.216      0.000       0.788       1.282
Diversified Financials             0.8089      0.099      8.149      0.000       0.614       1.004
Energy                             0.9329      0.091     10.222      0.000       0.754       1.112
Food & Staples Retailing           0.3189      0.293      1.089      0.276      -0.255       0.893
Food, Beverage & Tobacco           0.6017      0.160      3.772      0.000       0.289       0.914
Health Care Equipment & Services   0.6937      0.095      7.322      0.000       0.508       0.879
Household & Personal Products      0.7045      0.254      2.778      0.005       0.207       1.202
Insurance                          0.3677      0.149      2.476      0.013       0.077       0.659
```

```
Materials                                          1.0635      0.103     10.319      0.000       0.861       1.266
Media & Entertainment                              0.8383      0.126      6.634      0.000       0.591       1.086
Pharmaceuticals, Biotechnology & Life Sciences     0.7760      0.075     10.364      0.000       0.629       0.923
Real Estate                                        0.4483      0.103      4.340      0.000       0.246       0.651
Retailing                                          2.2721      0.123     18.408      0.000       2.030       2.514
Semiconductors & Semiconductor Equipment           1.3906      0.152      9.138      0.000       1.092       1.689
Software & Services                                1.1507      0.090     12.788      0.000       0.974       1.327
Technology Hardware & Equipment                    1.1050      0.113      9.742      0.000       0.883       1.327
Transportation                                     0.7669      0.161      4.754      0.000       0.451       1.083
Utilities                                          0.3768      0.156      2.414      0.016       0.071       0.683
==============================================================================
Omnibus:                     5425.911   Durbin-Watson:                   1.943
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1372805.310
Skew:                           7.244   Prob(JB):                         0.00
Kurtosis:                      91.134   Cond. No.                         3.91
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# find the highest and lowest returns
print("Highest:", reg2.params.idxmax(), reg2.params.max())
print("Lowest:", reg2.params.idxmin(), reg2.params.min())
```

```
Highest: Automobiles & Components 2.2747520438955586
Lowest: Food & Staples Retailing 0.31888394333019504
```

Using 69 GICS Industries

Create binary indicators for the 69 GICS Industries.

In [ ]:

```
df2 = df[['ticker', 'RetEarly2020', 'RetLate2020', 'gind']].copy()
df2 = pd.get_dummies(df2, columns=['gind'], prefix='', prefix_sep='')
df2.head(1)
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	Aerospace & Defense	Air Freight & Logistics	Airlines	Auto Components	Automobiles	Banks	Beverages	Biotechnology	Building Products	C Ma
0	JJSF	-0.340234	0.30034	0	0	0	0	0	0	0	0	0	

In [ ]:

```
# regress RetEarly2020 on 69 indicators
Y = df2['RetEarly2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg1 = sm.OLS(Y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           RetEarly2020   R-squared:                       0.110
Model:                            OLS   Adj. R-squared:                  0.095
Method:                 Least Squares   F-statistic:                     7.361
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           1.68e-62
Time:                        23:00:27   Log-Likelihood:                -1704.0
No. Observations:                4130   AIC:                             3546.
Df Residuals:                    4061   BIC:                             3982.
Df Model:                          68
Covariance Type:            nonrobust
==========================================================================================================================
                                                       coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------------------
Aerospace & Defense                                 -0.3166      0.056     -5.632      0.000      -0.427      -0.206
Air Freight & Logistics                             -0.1790      0.099     -1.817      0.069      -0.372       0.014
Airlines                                            -0.5370      0.092     -5.827      0.000      -0.718      -0.356
Auto Components                                     -0.4500      0.064     -7.012      0.000      -0.576      -0.324
Automobiles                                         -0.3314      0.092     -3.596      0.000      -0.512      -0.151
Banks                                               -0.3694      0.020    -18.259      0.000      -0.409      -0.330
Beverages                                           -0.2993      0.077     -3.893      0.000      -0.450      -0.149
Biotechnology                                       -0.1032      0.022     -4.610      0.000      -0.147      -0.059
Building Products                                   -0.2599      0.065     -3.988      0.000      -0.388      -0.132
Capital Markets                                     -0.3223      0.031    -10.528      0.000      -0.382      -0.262
Chemicals                                           -0.3652      0.041     -8.915      0.000      -0.445      -0.285
Commercial Services & Supplies                      -0.3526      0.045     -7.888      0.000      -0.440      -0.265
Communications Equipment                            -0.2576      0.048     -5.368      0.000      -0.352      -0.164
Construction & Engineering                          -0.2379      0.066     -3.593      0.000      -0.368      -0.108
Construction Materials                              -0.3741      0.102     -3.659      0.000      -0.575      -0.174
Consumer Finance                                    -0.3895      0.058     -6.765      0.000      -0.502      -0.277
Containers & Packaging                              -0.2514      0.085     -2.972      0.003      -0.417      -0.086
Distributors                                        -0.2919      0.123     -2.376      0.018      -0.533      -0.051
Diversified Consumer Services                       -0.1957      0.054     -3.640      0.000      -0.301      -0.090
Diversified Financial Services                      -0.2570      0.106     -2.415      0.016      -0.466      -0.048
Diversified Telecommunication Services              -0.1660      0.070     -2.383      0.017      -0.303      -0.029
Electric Utilities                                  -0.2192      0.061     -3.567      0.000      -0.340      -0.099
Electrical Equipment                                -0.3137      0.056     -5.645      0.000      -0.423      -0.205
Electronic Equipment, Instruments & Components      -0.2808      0.038     -7.345      0.000      -0.356      -0.206
```

Energy Equipment & Services	-0.6482	0.046	-13.957	0.000	-0.739	-0.557
Entertainment	-0.2123	0.058	-3.642	0.000	-0.327	-0.098
Equity Real Estate Investment Trusts (REITs)	-0.3344	0.028	-12.000	0.000	-0.389	-0.280
Food & Staples Retailing	-0.0557	0.071	-0.786	0.432	-0.195	0.083
Food Products	-0.2122	0.047	-4.497	0.000	-0.305	-0.120
Gas Utilities	-0.1682	0.102	-1.645	0.100	-0.369	0.032
Health Care Equipment & Supplies	-0.0874	0.030	-2.941	0.003	-0.146	-0.029
Health Care Providers & Services	-0.2086	0.041	-5.030	0.000	-0.290	-0.127
Health Care Technology	-0.0934	0.074	-1.267	0.205	-0.238	0.051
Hotels, Restaurants & Leisure	-0.4925	0.037	-13.292	0.000	-0.565	-0.420
Household Durables	-0.4089	0.049	-8.301	0.000	-0.506	-0.312
Household Products	-0.0613	0.111	-0.552	0.581	-0.279	0.157
IT Services	-0.2576	0.037	-6.918	0.000	-0.331	-0.185
Independent Power and Renewable Electricity Producers	-0.1834	0.092	-1.990	0.047	-0.364	-0.003
Industrial Conglomerates	-0.2752	0.130	-2.111	0.035	-0.531	-0.020
Insurance	-0.2401	0.036	-6.674	0.000	-0.311	-0.170
Interactive Media & Services	-0.2909	0.056	-5.175	0.000	-0.401	-0.181
Internet & Direct Marketing Retail	-0.1593	0.052	-3.056	0.002	-0.262	-0.057
Leisure Products	-0.2721	0.082	-3.301	0.001	-0.434	-0.110
Life Sciences Tools & Services	-0.1761	0.063	-2.785	0.005	-0.300	-0.052
Machinery	-0.3385	0.035	-9.630	0.000	-0.407	-0.270
Marine	-0.4855	0.075	-6.452	0.000	-0.633	-0.338
Media	-0.3554	0.047	-7.591	0.000	-0.447	-0.264
Metals & Mining	-0.3681	0.038	-9.682	0.000	-0.443	-0.294
Mortgage Real Estate Investment Trusts (REITs)	-0.5709	0.061	-9.420	0.000	-0.690	-0.452
Multi-Utilities	-0.1570	0.087	-1.807	0.071	-0.327	0.013
Multiline Retail	-0.4428	0.117	-3.798	0.000	-0.671	-0.214
Oil, Gas & Consumable Fuels	-0.5239	0.025	-20.837	0.000	-0.573	-0.475
Paper & Forest Products	-0.3852	0.111	-3.466	0.001	-0.603	-0.167
Personal Products	-0.2388	0.074	-3.238	0.001	-0.383	-0.094
Pharmaceuticals	-0.2409	0.035	-6.792	0.000	-0.310	-0.171
Professional Services	-0.2859	0.048	-5.907	0.000	-0.381	-0.191
Real Estate Management & Development	-0.3343	0.057	-5.877	0.000	-0.446	-0.223
Road & Rail	-0.2415	0.068	-3.527	0.000	-0.376	-0.107
Semiconductors & Semiconductor Equipment	-0.2201	0.037	-5.971	0.000	-0.292	-0.148
Software	-0.1633	0.027	-6.074	0.000	-0.216	-0.111
Specialty Retail	-0.4627	0.040	-11.434	0.000	-0.542	-0.383
Technology Hardware, Storage & Peripherals	-0.3693	0.070	-5.300	0.000	-0.506	-0.233
Textiles, Apparel & Luxury Goods	-0.4130	0.057	-7.260	0.000	-0.524	-0.301
Thrifts & Mortgage Finance	-0.3151	0.042	-7.453	0.000	-0.398	-0.232
Tobacco	-0.2271	0.139	-1.630	0.103	-0.500	0.046
Trading Companies & Distributors	-0.4217	0.056	-7.589	0.000	-0.531	-0.313
Transportation Infrastructure	-0.4759	0.150	-3.162	0.002	-0.771	-0.181
Water Utilities	-0.0953	0.106	-0.896	0.370	-0.304	0.113
Wireless Telecommunication Services	-0.1928	0.087	-2.219	0.027	-0.363	-0.022

==

Omnibus:	8337.693	Durbin-Watson:	1.950

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         44261418.235
Skew:                          16.133   Prob(JB):                         0.00
Kurtosis:                     509.131   Cond. No.                         7.44
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```python
# find the highest and lowest returns
print("Highest:", reg1.params.idxmax(), reg1.params.max())
print("Lowest:", reg1.params.idxmin(), reg1.params.min())
```

```
Highest: Food & Staples Retailing -0.055730434846246284
Lowest: Energy Equipment & Services -0.648216098869443
```

In [ ]:

```python
# regress RetLate2020 on 69 indicators
Y = df2['RetLate2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg2 = sm.OLS(Y, X).fit()
print(reg2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.108
Model:                            OLS   Adj. R-squared:                  0.094
Method:                 Least Squares   F-statistic:                     7.264
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           2.17e-61
Time:                        23:00:27   Log-Likelihood:                -7510.3
No. Observations:                4130   AIC:                         1.516e+04
Df Residuals:                    4061   BIC:                         1.560e+04
Df Model:                          68
Covariance Type:            nonrobust
===========================================================================================================
                                                  coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------------
Aerospace & Defense                             0.5823      0.229      2.539      0.011       0.133       1.032
Air Freight & Logistics                         0.6081      0.402      1.513      0.130      -0.180       1.396
Airlines                                        0.7710      0.376      2.051      0.040       0.034       1.508
Auto Components                                 1.5563      0.262      5.945      0.000       1.043       2.069
Automobiles                                     3.7566      0.376      9.993      0.000       3.020       4.494
Banks                                           0.4110      0.083      4.980      0.000       0.249       0.573
Beverages                                       0.9561      0.314      3.049      0.002       0.341       1.571
Biotechnology                                   0.8759      0.091      9.589      0.000       0.697       1.055
Building Products                               0.9424      0.266      3.545      0.000       0.421       1.464
Capital Markets                                 0.8089      0.125      6.477      0.000       0.564       1.054
```

Chemicals	1.0605	0.167	6.347	0.000	0.733	1.388
Commercial Services & Supplies	0.7888	0.182	4.325	0.000	0.431	1.146
Communications Equipment	0.7421	0.196	3.791	0.000	0.358	1.126
Construction & Engineering	1.3734	0.270	5.085	0.000	0.844	1.903
Construction Materials	0.7973	0.417	1.912	0.056	-0.020	1.615
Consumer Finance	0.7741	0.235	3.296	0.001	0.314	1.235
Containers & Packaging	0.5840	0.345	1.693	0.091	-0.092	1.260
Distributors	0.9169	0.501	1.829	0.067	-0.066	1.900
Diversified Consumer Services	0.5550	0.219	2.530	0.011	0.125	0.985
Diversified Financial Services	0.5066	0.434	1.167	0.243	-0.344	1.358
Diversified Telecommunication Services	0.3643	0.284	1.282	0.200	-0.193	0.921
Electric Utilities	0.2062	0.251	0.823	0.411	-0.285	0.698
Electrical Equipment	2.3106	0.227	10.192	0.000	1.866	2.755
Electronic Equipment, Instruments & Components	1.2866	0.156	8.251	0.000	0.981	1.592
Energy Equipment & Services	1.0587	0.189	5.588	0.000	0.687	1.430
Entertainment	0.7716	0.238	3.245	0.001	0.305	1.238
Equity Real Estate Investment Trusts (REITs)	0.3900	0.114	3.431	0.001	0.167	0.613
Food & Staples Retailing	0.3189	0.289	1.102	0.271	-0.248	0.886
Food Products	0.4715	0.193	2.449	0.014	0.094	0.849
Gas Utilities	0.0363	0.417	0.087	0.931	-0.781	0.854
Health Care Equipment & Supplies	0.6517	0.121	5.378	0.000	0.414	0.889
Health Care Providers & Services	0.7400	0.169	4.374	0.000	0.408	1.072
Health Care Technology	0.8060	0.301	2.680	0.007	0.216	1.396
Hotels, Restaurants & Leisure	1.2624	0.151	8.353	0.000	0.966	1.559
Household Durables	2.2537	0.201	11.215	0.000	1.860	2.648
Household Products	0.4950	0.453	1.092	0.275	-0.394	1.384
IT Services	1.1199	0.152	7.373	0.000	0.822	1.418
Independent Power and Renewable Electricity Producers	1.5159	0.376	4.032	0.000	0.779	2.253
Industrial Conglomerates	0.4603	0.532	0.866	0.387	-0.582	1.503
Insurance	0.3677	0.147	2.506	0.012	0.080	0.655
Interactive Media & Services	1.1230	0.229	4.897	0.000	0.673	1.573
Internet & Direct Marketing Retail	2.0563	0.213	9.669	0.000	1.639	2.473
Leisure Products	1.8269	0.336	5.433	0.000	1.168	2.486
Life Sciences Tools & Services	1.1981	0.258	4.646	0.000	0.693	1.704
Machinery	0.9775	0.143	6.817	0.000	0.696	1.259
Marine	0.7274	0.307	2.370	0.018	0.126	1.329
Media	0.6840	0.191	3.582	0.000	0.310	1.058
Metals & Mining	1.2073	0.155	7.784	0.000	0.903	1.511
Mortgage Real Estate Investment Trusts (REITs)	0.9456	0.247	3.825	0.000	0.461	1.430
Multi-Utilities	0.1353	0.354	0.382	0.703	-0.560	0.830
Multiline Retail	1.2158	0.476	2.557	0.011	0.284	2.148
Oil, Gas & Consumable Fuels	0.8961	0.103	8.737	0.000	0.695	1.097
Paper & Forest Products	0.9994	0.453	2.204	0.028	0.111	1.888
Personal Products	0.7967	0.301	2.649	0.008	0.207	1.386
Pharmaceuticals	0.3925	0.145	2.713	0.007	0.109	0.676
Professional Services	0.6134	0.197	3.107	0.002	0.226	1.001
Real Estate Management & Development	0.6911	0.232	2.979	0.003	0.236	1.146

```
Road & Rail                                    0.8776      0.279      3.143      0.002      0.330       1.425
Semiconductors & Semiconductor Equipment       1.3906      0.150      9.248      0.000      1.096       1.685
Software                                       1.1667      0.110     10.638      0.000      0.952       1.382
Specialty Retail                               2.6764      0.165     16.215      0.000      2.353       3.000
Technology Hardware, Storage & Peripherals     1.2662      0.284      4.455      0.000      0.709       1.823
Textiles, Apparel & Luxury Goods               0.9463      0.232      4.078      0.000      0.491       1.401
Thrifts & Mortgage Finance                     0.4584      0.172      2.657      0.008      0.120       0.797
Tobacco                                        0.5722      0.568      1.007      0.314     -0.542       1.687
Trading Companies & Distributors               1.3445      0.227      5.931      0.000      0.900       1.789
Transportation Infrastructure                  0.7497      0.614      1.221      0.222     -0.454       1.953
Water Utilities                                0.1012      0.434      0.233      0.816     -0.750       0.952
Wireless Telecommunication Services            0.3994      0.354      1.127      0.260     -0.295       1.094
==============================================================================
Omnibus:                     5392.900   Durbin-Watson:                   1.946
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1392755.810
Skew:                           7.147   Prob(JB):                         0.00
Kurtosis:                      91.821   Cond. No.                         7.44
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# find the highest and lowest returns
print("Highest:", reg2.params.idxmax(), reg2.params.max())
print("Lowest:", reg2.params.idxmin(), reg2.params.min())
```

```
Highest: Automobiles 3.756618195786346
Lowest: Gas Utilities 0.036267464535508181
```

Using 20 NAICS Sectors

Althought there are 20 NAICS sectors, only 19 are presented in our dataframe (no "Management of Companies and Enterprises").

Create binary indicators for the 19 NAICS Sectors.

In [ ]:

```
df2 = df[['ticker', 'RetEarly2020', 'RetLate2020', 'nsector']].copy()
df2 = pd.get_dummies(df2, columns=['nsector'], prefix='', prefix_sep='')
df2.head(1)
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	Accommodation and Food Services	Administrative and Support and Waste Management and Remediation Services	Agriculture, Forestry, Fishing and Hunting	Arts, Entertainment, and Recreation	Construction	Educational Services	Finance and Insurance	He Care S Assist
0	JJSF	-0.340234	0.30034	0	0	0	0	0	0	0	

In [ ]:

```
# regress RetEarly2020 on 19 indicators
Y = df2['RetEarly2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg1 = sm.OLS(Y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           RetEarly2020   R-squared:                       0.044
Model:                            OLS   Adj. R-squared:                  0.040
Method:                 Least Squares   F-statistic:                     10.58
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           4.97e-30
Time:                        23:00:27   Log-Likelihood:                -1850.5
No. Observations:                4130   AIC:                             3739.
Df Residuals:                    4111   BIC:                             3859.
Df Model:                          18
Covariance Type:            nonrobust
=====================================================================================================================================
==========
                                                                               coef    std err          t      P>|t|      [0.025
0.975]
-------------------------------------------------------------------------------------------------------------------------------------
----------
Accommodation and Food Services                                             -0.4525      0.046     -9.902      0.000      -0.542
-0.363
Administrative and Support and Waste Management and Remediation Services    -0.3502      0.044     -7.882      0.000      -0.437
-0.263
Agriculture, Forestry, Fishing and Hunting                                  -0.2129      0.120     -1.774      0.076      -0.448
0.022
Arts, Entertainment, and Recreation                                         -0.4372      0.081     -5.402      0.000      -0.596
-0.279
Construction                                                                -0.3893      0.053     -7.395      0.000      -0.493
```

```
-0.286
Educational Services                                        -0.1231      0.063     -1.945      0.052      -0.247
0.001
Finance and Insurance                                       -0.3444      0.014    -24.864      0.000      -0.372
-0.317
Health Care and Social Assistance                           -0.2270      0.052     -4.353      0.000      -0.329
-0.125
Information                                                 -0.2120      0.018    -12.006      0.000      -0.247
-0.177
Manufacturing                                               -0.2504      0.010    -25.786      0.000      -0.269
-0.231
Mining, Quarrying, and Oil and Gas Extraction               -0.5372      0.027    -19.609      0.000      -0.591
-0.484
Other Services                                              -0.4131      0.127     -3.265      0.001      -0.661
-0.165
Professional, Scientific, and Technical Services            -0.2669      0.035     -7.605      0.000      -0.336
-0.198
Public Administration                                       -0.3726      0.170     -2.195      0.028      -0.705
-0.040
Real Estate and Rental and Leasing                          -0.3408      0.023    -14.613      0.000      -0.387
-0.295
Retail Trade                                                -0.3354      0.032    -10.564      0.000      -0.398
-0.273
Transportation and Warehousing                              -0.4049      0.032    -12.666      0.000      -0.468
-0.342
Utilities                                                   -0.1909      0.039     -4.928      0.000      -0.267
-0.115
Wholesale Trade                                             -0.3366      0.037     -9.129      0.000      -0.409
-0.264
==============================================================================
Omnibus:                     8132.646   Durbin-Watson:                   1.956
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         37151715.592
Skew:                          15.270   Prob(JB):                         0.00
Kurtosis:                     466.639   Cond. No.                         17.5
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# find the highest and lowest returns
print("Highest:", reg1.params.idxmax(), reg1.params.max())
print("Lowest:", reg1.params.idxmin(), reg1.params.min())
```

```
Highest: Educational Services -0.12306387186084192
Lowest: Mining, Quarrying, and Oil and Gas Extraction -0.5372280583538552
```

In [ ]:

```
# regress RetLate2020 on 19 indicators
Y = df2['RetLate2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg2 = sm.OLS(Y, X).fit()
print(reg2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.044
Model:                            OLS   Adj. R-squared:                  0.040
Method:                 Least Squares   F-statistic:                     10.53
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           7.39e-30
Time:                        23:00:27   Log-Likelihood:                -7654.3
No. Observations:                4130   AIC:                         1.535e+04
Df Residuals:                    4111   BIC:                         1.547e+04
Df Model:                          18
Covariance Type:            nonrobust
=====================================================================================================
==========
                                                                           coef    std err          t      P>|t|      [0.025
0.975]
-----------------------------------------------------------------------------------------------------
----------
Accommodation and Food Services                                          1.1749      0.186      6.306      0.000       0.810
1.540
Administrative and Support and Waste Management and Remediation Services 0.8573      0.181      4.733      0.000       0.502
1.212
Agriculture, Forestry, Fishing and Hunting                               0.5553      0.489      1.135      0.257      -0.404
1.515
Arts, Entertainment, and Recreation                                      1.1178      0.330      3.388      0.001       0.471
1.765
Construction                                                             1.3564      0.215      6.320      0.000       0.936
1.777
Educational Services                                                     0.3130      0.258      1.214      0.225      -0.193
0.819
Finance and Insurance                                                    0.5362      0.056      9.496      0.000       0.426
0.647
Health Care and Social Assistance                                        0.8497      0.213      3.997      0.000       0.433
1.266
Information                                                              1.0529      0.072     14.624      0.000       0.912
1.194
Manufacturing                                                            1.0501      0.040     26.524      0.000       0.972
1.128
Mining, Quarrying, and Oil and Gas Extraction                            1.1648      0.112     10.429      0.000       0.946
1.384
Other Services                                                           3.3817      0.516      6.555      0.000       2.370
4.393
```

```
Professional, Scientific, and Technical Services                    0.7151      0.143      4.998      0.000      0.435
0.996
Public Administration                                               0.3963      0.692      0.573      0.567     -0.961
1.753
Real Estate and Rental and Leasing                                  0.5619      0.095      5.911      0.000      0.376
0.748
Retail Trade                                                        1.8607      0.129     14.377      0.000      1.607
2.114
Transportation and Warehousing                                      0.6201      0.130      4.758      0.000      0.365
0.876
Utilities                                                           0.3355      0.158      2.124      0.034      0.026
0.645
Wholesale Trade                                                     0.8243      0.150      5.483      0.000      0.530
1.119
==============================================================================
Omnibus:                     5361.376   Durbin-Watson:                   1.955
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1266193.827
Skew:                           7.107   Prob(JB):                         0.00
Kurtosis:                      87.593   Cond. No.                         17.5
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# find the highest and lowest returns
print("Highest:", reg2.params.idxmax(), reg2.params.max())
print("Lowest:", reg2.params.idxmin(), reg2.params.min())
```

```
Highest: Other Services 3.381705505613294
Lowest: Educational Services 0.31300909334690385
```

Using 10 SIC Industry Groups

Create binary indicators for the 10 SIC Industry Groups.

In [ ]:

```
df2 = df[['ticker', 'RetEarly2020', 'RetLate2020', 'sgroup']].copy()
df2 = pd.get_dummies(df2, columns=['sgroup'], prefix='', prefix_sep='')
df2.head(1)
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	Agriculture, Forestry and Fishing	Construction	Finance, Insurance and Real Estate	Manufacturing	Mining	Nonclassifiable	Retail Trade	Services	Transpor Communic Electric, G Sanitary s

	ticker	RetEarly2020	RetLate2020	Agriculture, Forestry and Fishing	Construction	Finance, Insurance and Real Estate	Manufacturing	Mining	Nonclassifiable	Retail Trade	Services	Transpor Communic Electric, G Sanitary :
0	JJSF	-0.340234	0.30034	0	0	0	1	0	0	0	0	

In [ ]:

```
# regress RetEarly2020 on 10 indicators
Y = df2['RetEarly2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg1 = sm.OLS(Y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           RetEarly2020   R-squared:                       0.032
Model:                            OLS   Adj. R-squared:                  0.030
Method:                 Least Squares   F-statistic:                     15.29
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           8.85e-25
Time:                        23:00:28   Log-Likelihood:                -1876.2
No. Observations:                4130   AIC:                             3772.
Df Residuals:                    4120   BIC:                             3836.
Df Model:                           9
Covariance Type:            nonrobust
==========================================================================================================
====
                                                         coef    std err          t      P>|t|      [0.025      0.
975]
----------------------------------------------------------------------------------------------------------
----
Agriculture, Forestry and Fishing                     -0.2249      0.115     -1.955      0.051      -0.450
0.001
Construction                                          -0.3731      0.058     -6.411      0.000      -0.487      -
0.259
Finance, Insurance and Real Estate                    -0.3410      0.012    -28.207      0.000      -0.365      -
0.317
Manufacturing                                         -0.2513      0.010    -25.831      0.000      -0.270      -
0.232
Mining                                                -0.5427      0.027    -19.758      0.000      -0.597      -
0.489
Nonclassifiable                                       -0.3505      0.156     -2.250      0.024      -0.656      -
0.045
```

```
Retail Trade                                                         -0.3622      0.028    -13.051      0.000      -0.417      -
0.308
Services                                                             -0.2445      0.014    -16.890      0.000      -0.273      -
0.216
Transportation, Communications, Electric, Gas and Sanitary service   -0.3071      0.020    -15.101      0.000      -0.347      -
0.267
Wholesale Trade                                                      -0.3366      0.037     -9.083      0.000      -0.409      -
0.264
==============================================================================
Omnibus:                     8070.000   Durbin-Watson:                   1.952
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         35389119.992
Skew:                          15.012   Prob(JB):                         0.00
Kurtosis:                     455.493   Cond. No.                         16.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# find the highest and lowest returns
print("Highest:", reg1.params.idxmax(), reg1.params.max())
print("Lowest:", reg1.params.idxmin(), reg1.params.min())
```

```
Highest: Agriculture, Forestry and Fishing -0.22489510749355432
Lowest: Mining -0.5426667053652766
```

In [ ]:

```
# regress RetLate2020 on 10 indicators
Y = df2['RetLate2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg2 = sm.OLS(Y, X).fit()
print(reg2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.038
Model:                            OLS   Adj. R-squared:                  0.036
Method:                 Least Squares   F-statistic:                     18.32
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           3.12e-30
Time:                        23:00:28   Log-Likelihood:                -7666.3
No. Observations:                4130   AIC:                         1.535e+04
Df Residuals:                    4120   BIC:                         1.542e+04
Df Model:                           9
Covariance Type:            nonrobust
====================================================================================================
====
                                                                        coef    std err          t      P>|t|      [0.025      0.
```

```
975]
----------------------------------------------------------------------------------------------------
----
Agriculture, Forestry and Fishing                                          0.5382      0.467      1.151      0.250      -0.378
1.455
Construction                                                               1.5181      0.236      6.421      0.000       1.055
1.982
Finance, Insurance and Real Estate                                         0.5191      0.049     10.567      0.000       0.423
0.615
Manufacturing                                                              1.0478      0.040     26.513      0.000       0.970
1.125
Mining                                                                     1.1643      0.112     10.433      0.000       0.945
1.383
Nonclassifiable                                                            0.4329      0.633      0.684      0.494      -0.808
1.674
Retail Trade                                                               1.7065      0.113     15.131      0.000       1.485
1.928
Services                                                                   1.0499      0.059     17.852      0.000       0.935
1.165
Transportation, Communications, Electric, Gas and Sanitary service         0.5352      0.083      6.477      0.000       0.373
0.697
Wholesale Trade                                                            0.8243      0.151      5.474      0.000       0.529
1.119
==============================================================================
Omnibus:                     5439.207   Durbin-Watson:                   1.960
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1361178.145
Skew:                           7.285   Prob(JB):                         0.00
Kurtosis:                      90.737   Cond. No.                         16.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# find the highest and lowest returns
print("Highest:", reg2.params.idxmax(), reg2.params.max())
print("Lowest:", reg2.params.idxmin(), reg2.params.min())
```

```
Highest: Retail Trade 1.7064742888387123
Lowest: Nonclassifiable 0.4328781953493601
```

Interpretation for `RetEarly2020`

- **What are the average returns for each industry?**
- **Do the highest and lowest return industries make economic sense?**

- **What is the explanatory power of these regressions using different types of industry variables?**

In the industry-fixed effect regression models above, the coefficients represent the average 3-month returns for each industry.

- The models built on GICS indicators have higher explanatory power - at least 8% of the variation in `RetEarly2020` can be explained by these models.
- The models built on NAICS and SIC indicators have lower explanatory power - they can only explain around 3% to 4% of the variation in `RetEarly2020` .

In all the models, all the industries suffered from negative returns in early 2020. However, the Service industry (especially Health Care Services) was the least impacted by the initial COVID shock, as indicated by the models. It makes economic sense because health care services were essential and growing, particularly in midst of a pandemic. On the other hand, the Energy industry took the strongest hit when the stay-at-home orders went into effect. This also makes economic sense. Since fewer people were commuting to work or traveling, the demand for transportation and energy declined significantly.

Interpretation for `RetLate2020`

In the industry-fixed effect regression models above, the coefficients represent the average 9-month returns for each industry.

- The models built on GICS indicators again have higher explanatory power - around 6% to 9% of the variation in `RetLate2020` can be explained by these models.
- The models built on NAICS and SIC indicators have similarly low explanatory power as before - they can only explain around 4% of the variation in `RetLate2020` .

In all the models, all the industries have positive 9-month returns. Among them, consumer-facing industries (such as Automobiles and Retail) saw the largest rebound from April to December 2020. This makes economic sense. Some reasons behind the surge include: 1) while staying at home, consumers shopping online had increased significantly, 2) people turned to purchasing vehicles to avoid taking public transportation, and 3) the dramatic rise of Tesla (TSLA). On the other hand, the Utilities and Insurance industries had a slower recovery. This also makes economic sense. One reason could be that comparing to stocks in other industries, these stocks are usually less volatile. Additionally, these industries did not see a drastic fall in early 2020, so a major rebound was not anticipated either.

2. Explaining Fluctuation Using Financial Ratios

In [ ]:

```
df = pd.read_csv('maindf.csv')
```

`wrds_additional.csv` contains additional 2019 accounting data downloaded from WRDS Compustat.

- `ch` : Cash
- `tie` : Total Interest Expense
- `dltt` : Total Long-Term Debt
- `ebit` : Earnings Before Interest and Taxes
- `fatd` : Fixed Assets
- `emp` : Number of Employees

In [ ]:
```
wrds = pd.read_csv('wrds_additional.csv')
wrds.rename(columns={'tic':'ticker', 'ch':'ch19', 'dltt':'dltt19', 'ebit':'ebit19',
                     'emp':'emp19', 'fatd':'fatd19', 'tie':'tie19'}, inplace=True)
```

In [ ]:
```
# only keep companies that are already in the main dataframe
wrds = wrds[wrds.ticker.isin(df.ticker)]
```

Drop `fatd` because there's no data in it. Drop `tie` because there's data for only 390 companies.

In [ ]:
```
wrds = wrds.drop(columns=['fatd19', 'tie19'])
```

In [ ]:
```
# merge the dataframes
df = df.merge(wrds, on='ticker', how='left')
print(df.shape)
```

```
(4130, 46)
```

Fill in missing values with data from Yahoo Finance.

In [ ]:
```
# # WARNING!! - Don't run these loops. The results are already saved in 'df_ratios.csv'
# # fill in missing ch19
# tickers = df[df['ch19'].isnull()].ticker.unique()

# for i in tickers:
#     tic = yf.Ticker(i)
#     try:
#         if df.loc[df.ticker==i, 'ch19'].isna().any():
#             df.loc[df.ticker==i, 'ch19'] = tic.info['totalCash']/1000000
```

```
#     except:
#         continue
```

In [ ]:

```
# # fill in missing lct19
# tickers = df[df['lct19'].isnull()].ticker.unique()

# def findinx(columns):
#     for i, v in enumerate(columns):
#         if v[:4]=='2019':
#             return i
#         else:
#             continue

# for i in tickers:
#     tic = yf.Ticker(i)
#     try:
#         if df.loc[df.ticker==i, 'lct19'].isna().any():
#             columns = tic.balance_sheet.columns.astype(str)
#             col = findinx(columns)
#             df.loc[df.ticker==i, 'lct19'] = tic.balance_sheet.loc['Total Current Liabilities'][col]/1000000
#     except:
#         continue
```

In [ ]:

```
# # fill in missing act19
# tickers = df[df['act19'].isnull()].ticker.unique()

# for i in tickers:
#     tic = yf.Ticker(i)
#     try:
#         if df.loc[df.ticker==i, 'lct19'].isna().any():
#             columns = tic.balance_sheet.columns.astype(str)
#             col = findinx(columns)
#             df.loc[df.ticker==i, 'lct19'] = tic.balance_sheet.loc['Total Current Assets'][col]/1000000
#     except:
#         continue
```

In [ ]:

```
# # fill in missing dltt19
# tickers = df[df['dltt19'].isnull()].ticker.unique()

# for i in tickers:
```

```
#     tic = yf.Ticker(i)
#     try:
#         if df.loc[df.ticker==i, 'dltt19'].isna().any():
#             columns = tic.balance_sheet.columns.astype(str)
#             col = findinx(columns)
#             df.loc[df.ticker==i, 'dltt19'] = tic.balance_sheet.loc['Long Term Debt'][col]/1000000
#     except:
#         continue
```

In [ ]:

```
# # fill in missing emp19
# tickers = df[df['emp19'].isnull()].ticker.unique()

# for i in tickers:
#     tic = yf.Ticker(i)
#     try:
#         if df.loc[df.ticker==i, 'emp19'].isna().any():
#             df.loc[df.ticker==i, 'emp19'] = tic.info['fullTimeEmployees']/1000
#     except:
#         continue
```

In [ ]:

```
# # fill in missing re19
# tickers = df[df['re19'].isnull()].ticker.unique()

# for i in tickers:
#     tic = yf.Ticker(i)
#     try:
#         if df.loc[df.ticker==i, 're19'].isna().any():
#             columns = tic.balance_sheet.columns.astype(str)
#             col = findinx(columns)
#             df.loc[df.ticker==i, 're19'] = tic.balance_sheet.loc['Retained Earnings'][col]/1000000
#     except:
#         continue
```

In [ ]:

```
# # fill in missing ebit19
# tickers = df[df['ebit19'].isnull()].ticker.unique()

# for i in tickers:
#     tic = yf.Ticker(i)
#     try:
#         if df.loc[df.ticker==i, 'ebit19'].isna().any():
```

```
#             columns = tic.balance_sheet.columns.astype(str)
#             col = findinx(columns)
#             df.loc[df.ticker==i, 'ebit19'] = tic.financials.loc['Ebit'][col]/1000000
#     except:
#         continue
```

Calculate additional financial ratios.

In [ ]:

```
df['cta19'] = df['ch19'] / df['at19']
df['cash19'] = df['ch19'] / df['lct19']
df['quick19'] = (df['act19']-df['invt19']) / df['lct19']
df['lda19'] = df['dltt19'] / df['at19']
df['se19'] = df['sale19'] / df['emp19']

df['T1'] = (df['act19']-df['lct19']) / df['at19']
df['T2'] = df['re19'] / df['at19']
df['T3'] = df['ebit19'] / df['at19']
```

In [ ]:

```
# # fill in missing quick19 with data from yahoo finance
# for i in df[df['quick19'].isnull()].ticker.unique():
#     try:
#         tic = yf.Ticker(i)
#         if df.loc[df.ticker==i, 'quick19'].isna().any():
#             df.loc[df.ticker==i, 'quick19'] = tic.info['quickRatio']
#     except:
#         continue
```

In [ ]:

```
# # export into a csv file
# df.to_csv('df_ratios.csv', index=False)
```

We now have the following financial ratios.

- `roa` : Return on Assets
- `atr` : Asset Turnover Ratio
- `ros` : Return on Sales
- `roe` : Return on Equity
- `emulti` : Equity Multiplier
- `ai` : Asset Intensity

- `gmargin` : Gross Margin
- `cta` : Cash to Total Assets
- `cash` : Cash Ratio
- `quick` : Quick Ratio
- `lda` : Long-Term Debt to Total Assets
- `se` : Sales per Employee
- `T1` : Working Capital to Assets (used in Altman's Z-score)
- `T2` : Retained Earnings to Assets (used in Altman's Z-score)
- `T3` : EBIT Return on Assets (used in Altman's Z-score)

In [ ]:

```
df = pd.read_csv('df_ratios.csv')
```

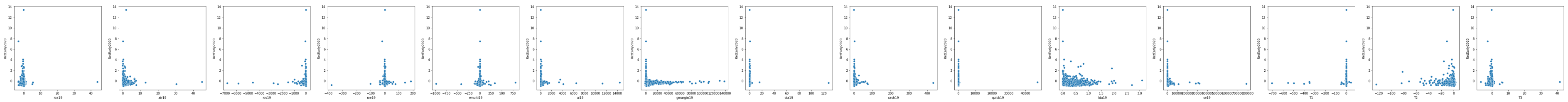
In [ ]:

```
# a list of ratios we might use
ratios = ['roa19', 'atr19', 'ros19', 'roe19', 'emulti19', 'ai19', 'gmargin19',
          'cta19', 'cash19', 'quick19', 'lda19', 'se19', 'T1', 'T2', 'T3']
```

Visualize the distribution of each ratio to look for outliers.

In [ ]:

```
fig, axs = plt.subplots(1, 15, figsize=(90, 5))
for i, v in enumerate(ratios):
    sns.scatterplot(data=df, x=v, y='RetEarly2020', ax=axs[i])
```



In [ ]:

```
# remove outliers for each ratio
print(df.shape)
df = df[~(df.roa19>10)]
df = df[~(df.atr19>10)]
df = df[~(df.ros19<-2000)]
df = df[~(abs(df.roe19)>100)]
df = df[~(abs(df.emulti19)>250)]
df = df[~(df.ai19>2000)]
df = df[~(df.gmargin19>80000)]
df = df[~(df.cta19>10)]
```

```
df = df[~(df.cash19>100)]
df = df[~(df.quick19>10000)]
df = df[~(df.lda19>2.5)]
df = df[~(df.se19>100000)]
df = df[~(df.T1<-100)]
df = df[~(df.T2<-60)]
df = df[~(df.T3>10)]
df.reset_index(drop=True, inplace=True)
print(df.shape)
```

```
(4130, 54)
(4085, 54)
```

Winsorize each ratio at 3% and 97% values.

In [ ]:
```
for i in ratios:
    df[i] = pd.Series(winsorize(df[i], limits=[0.03, 0.03]))
```

Replace missing ratios with the average of the GICS industry group a stock belongs to.

In [ ]:
```
# examine missing values
pd.DataFrame({'Number of companies with NA':(df[ratios].isna().sum()).sort_values(ascending=False)}).T
```

Out[ ]:

	quick19	se19	T3	T2	T1	lda19	cash19	cta19	gmargin19	ai19	emulti19	roe19	ros19	atr19	roa19
Number of companies with NA	419	153	0	0	0	0	0	0	0	0	0	0	0	0	0

In [ ]:
```
# check if there's any industry group without any quick19 or se19
print(df.groupby('ggroup').quick19.count().sort_values()[:2])
print(df.groupby('ggroup').se19.count().sort_values()[:2])
```

```
ggroup
Banks                     14
Food & Staples Retailing  26
Name: quick19, dtype: int64
ggroup
Food & Staples Retailing          26
Household & Personal Products     36
Name: se19, dtype: int64
```

In [ ]:

```
# replace missing ratios with GICS industry group average
qmeans = df.groupby('ggroup').quick19.mean()
smeans = df.groupby('ggroup').se19.mean()
df['quick19'] = np.where(df['quick19'].isna(), qmeans[df['ggroup']], df['quick19'])
df['se19'] = np.where(df['se19'].isna(), smeans[df['ggroup']], df['se19'])
```

In [ ]:

```
# double check that there's no missing value anymore
pd.DataFrame({'Number of companies with NA':(df[ratios].isna().sum())}).T
```

Out[ ]:

	roa19	atr19	ros19	roe19	emulti19	ai19	gmargin19	cta19	cash19	quick19	lda19	se19	T1	T2	T3
Number of companies with NA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In [ ]:

```
# # export into a csv file
# df.to_csv('df_ratios_cleaned.csv', index=False)
```

Single-Variable Regressions

We will first examine the explanatory power of the single regression models built on different financial ratios.

In [ ]:

```
df = pd.read_csv('df_ratios_cleaned.csv')
```

In [ ]:

```
cols = ['ticker', 'RetEarly2020', 'RetLate2020']
cols.extend(ratios)
df2 = df.dropna(subset=ratios)[cols].copy()
df2['intercept'] = 1
```

In [ ]:

```
# regress RetEarly2020 on every ratio
Y = df2['RetEarly2020']
for i in ratios:
    X = df2[['intercept', i]]
    print(sm.OLS(Y, X).fit().summary())
```

In [ ]:

```
# regress RetEarly2020 on every ratio's reciprocal
```

```
Y = df2['RetEarly2020']
for i in ratios:
    if (i=='lda19') or (i=='T1') or (i=='T2'):
        tmp = df2[df2[i]!=0]
        Y2 = tmp['RetEarly2020']
        X = 1/tmp[['intercept', i]]
        print(sm.OLS(Y2, X).fit().summary())
    else:
        X = 1/df2[['intercept', i]]
        print(sm.OLS(Y, X).fit().summary())
```

In [ ]:

```
# regress RetLate2020 on every ratio
Y = df2['RetLate2020']
for i in ratios:
    X = df2[['intercept', i]]
    print(sm.OLS(Y, X).fit().summary())
```

In [ ]:

```
# regress RetLate2020 on every ratio's reciprocal
Y = df2['RetLate2020']
for i in ratios:
    if (i=='lda19') or (i=='T1') or (i=='T2'):
        tmp = df2[df2[i]!=0]
        Y2 = tmp['RetEarly2020']
        X = 1/tmp[['intercept', i]]
        print(sm.OLS(Y2, X).fit().summary())
    else:
        X = 1/df2[['intercept', i]]
        print(sm.OLS(Y, X).fit().summary())
```

Multi-Variable Regressions

Then, we try out different combinations of financial ratios to explain the variation in 2020 stock returns using multiple regression.

In [ ]:

```
# regress RetEarly2020 on all ratios
Y = df2['RetEarly2020']
X = pd.concat([df2['intercept'], df2[ratios]], axis=1)
print(sm.OLS(Y, X).fit().summary())
```

In [ ]:

```
# regress RetEarly2020 on significant ratios with higher R2 (>0.01)
Y = df2['RetEarly2020']
X = pd.concat([df2[['intercept', 'roa19', 'ros19', 'cta19', 'cash19', 'lda19', 'se19', 'T1', 'T2', 'T3']],
               1/df2[['se19']]], axis=1)
print(sm.OLS(Y, X).fit().summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           RetEarly2020   R-squared:                       0.039
Model:                            OLS   Adj. R-squared:                  0.037
Method:                 Least Squares   F-statistic:                     16.74
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           4.46e-30
Time:                        23:02:15   Log-Likelihood:                -1841.9
No. Observations:                4085   AIC:                             3706.
Df Residuals:                    4074   BIC:                             3775.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -0.2864      0.013    -22.550      0.000      -0.311      -0.262
roa19         -0.0085      0.085     -0.100      0.920      -0.175       0.158
ros19         -0.0079      0.004     -1.809      0.070      -0.016       0.001
cta19          0.0679      0.060      1.126      0.260      -0.050       0.186
cash19         0.0027      0.007      0.410      0.682      -0.010       0.015
lda19         -0.1745      0.030     -5.788      0.000      -0.234      -0.115
se19       -4.429e-06   1.87e-06     -2.370      0.018   -8.09e-06   -7.65e-07
T1             0.0647      0.019      3.321      0.001       0.027       0.103
T2            -0.0086      0.005     -1.573      0.116      -0.019       0.002
T3            -0.0138      0.093     -0.150      0.881      -0.195       0.168
se19           1.2753      1.258      1.014      0.311      -1.191       3.742
==============================================================================
Omnibus:                     7963.062   Durbin-Watson:                   1.939
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         35427994.037
Skew:                          14.915   Prob(JB):                         0.00
Kurtosis:                     458.253   Cond. No.                     7.27e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.27e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:

```
# regress RetEarly2020 on suggested ratios
Y = df2['RetEarly2020']
X = pd.concat([df2[['intercept', 'cta19', 'cash19', 'lda19']],
```

```
                 1/df2[['quick19', 'se19']]], axis=1)
print(sm.OLS(Y, X).fit().summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          RetEarly2020   R-squared:                       0.032
Model:                           OLS   Adj. R-squared:                  0.031
Method:                Least Squares   F-statistic:                     26.74
Date:               Mon, 05 Jul 2021   Prob (F-statistic):           1.09e-26
Time:                       23:02:15   Log-Likelihood:                -1858.3
No. Observations:               4085   AIC:                             3729.
Df Residuals:                   4079   BIC:                             3766.
Df Model:                          5
Covariance Type:           nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -0.3170      0.014    -23.211      0.000      -0.344      -0.290
cta19          0.2102      0.054      3.902      0.000       0.105       0.316
cash19         0.0066      0.006      1.056      0.291      -0.006       0.019
lda19         -0.1589      0.029     -5.446      0.000      -0.216      -0.102
quick19        0.0032      0.008      0.426      0.670      -0.012       0.018
se19           4.2215      0.976      4.323      0.000       2.307       6.136
==============================================================================
Omnibus:                     7980.394   Durbin-Watson:                   1.936
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         35581791.714
Skew:                          14.991   Prob(JB):                         0.00
Kurtosis:                     459.234   Cond. No.                         300.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# regress RetLate2020 on all ratios
Y = df2['RetLate2020']
X = pd.concat([df2['intercept'], df2[ratios]], axis=1)
print(sm.OLS(Y, X).fit().summary())
```

In [ ]:

```
# regress RetLate2020 on significant ratios with higher R2 (>0.01)
Y = df2['RetLate2020']
X = pd.concat([df2[['intercept', 'roa19', 'atr19', 'roe19', 'T2', 'T3']],
               1/df2['ai19']], axis=1)
print(sm.OLS(Y, X).fit().summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.050
Model:                            OLS   Adj. R-squared:                  0.048
Method:                 Least Squares   F-statistic:                     35.63
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           3.03e-42
Time:                        23:02:15   Log-Likelihood:                -7386.5
No. Observations:                4085   AIC:                         1.479e+04
Df Residuals:                    4078   BIC:                         1.483e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.5857      0.037     15.826      0.000       0.513       0.658
roa19         -0.4941      0.352     -1.405      0.160      -1.184       0.195
atr19          0.6778      0.271      2.505      0.012       0.147       1.208
roe19          0.0779      0.068      1.146      0.252      -0.055       0.211
T2            -0.0207      0.021     -0.984      0.325      -0.062       0.021
T3            -1.0307      0.351     -2.939      0.003      -1.718      -0.343
ai19          -0.2613      0.285     -0.917      0.359      -0.820       0.297
==============================================================================
Omnibus:                     4894.204   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           782425.371
Skew:                           6.251   Prob(JB):                         0.00
Kurtosis:                      69.638   Cond. No.                         37.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# regress RetLate2020 on suggested ratios
Y = df2['RetLate2020']
X = pd.concat([df2[['intercept', 'cta19', 'lda19']],
               1/df2[['cash19', 'quick19', 'se19']]], axis=1)
print(sm.OLS(Y, X).fit().summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.016
Model:                            OLS   Adj. R-squared:                  0.014
Method:                 Least Squares   F-statistic:                     12.96
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           1.55e-12
Time:                        23:02:15   Log-Likelihood:                -7458.7
No. Observations:                4085   AIC:                         1.493e+04
Df Residuals:                    4079   BIC:                         1.497e+04
```

```
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.5903      0.058     10.164      0.000       0.476       0.704
cta19          0.5094      0.174      2.922      0.004       0.168       0.851
lda19          0.2496      0.118      2.119      0.034       0.019       0.480
cash19        -0.0014      0.001     -1.709      0.088      -0.003       0.000
quick19        0.1703      0.029      5.849      0.000       0.113       0.227
se19          11.6182      3.844      3.022      0.003       4.082      19.154
==============================================================================
Omnibus:                     4949.839   Durbin-Watson:                   1.959
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           785916.908
Skew:                           6.391   Prob(JB):                         0.00
Kurtosis:                      69.738   Cond. No.                     5.89e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.89e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Interpretation

We will particularly focus on the `Cash/Total Assets`, `Cash/Current Liabilities`, `Quick`, `Long-Term Debt/Total Assets`, and `Net Sales/Number of Employees` ratios as suggested.

Looking at each ratio's single regression models, we can conclude that:

- `Cash/Total Assets` ratio can explain around 2% of the variation in `RetEarly2020` and 0.1% of the variation in `RetLate2020`. Firms with more cash at the end of 2019 have higher 3-month and 9-month returns in 2020.
- `Cash/Current Liabilities` ratio can explain 1% of the variation in `RetEarly2020` and the inverse of `Cash/Current Liabilities` can explain 0.1% of the variation in `RetLate2020`. Firms with more cash to cover current liabilities at the end of 2019 have higher 3-month and 9-month returns in 2020.
- The inverse of `Quick` ratio can explain 0.2% of the variation in `RetEarly2020` and 0.6% of the variation in `RetLate2020`. Firms with more liquid assets to cover current liabilities at the end of 2019 have higher 3-month return in early 2020 and lower 9-month return later.
- `Long-Term Debt/Total Assets` ratio can explain 1.3% of the variation in `RetEarly2020` and 0.2% of the variation in `RetLate2020`. Firms with lower long-term debt at the end of 2019 have higher 3-month return in early 2020 and lower 9-month return later.

- The inverse of `Sales per Employees` ratio can explain 1.2% of the variation in `RetEarly2020` and 0.3% of the variation in `RetLate2020` . Firms that have higher reliance on labor perform better in 2020.

When we use all of the suggested ratios to build multi-variable regression models, our models can explain 3.1% of the variation in `RetEarly2020` and 1.4% of the variation in `RetLate2020` . Whether this explanatory power is high or low depends on what we compare these regression models with. Intuitively, models that can only explains 3.1% or 1.4% of the variation seems to have a very low explanatory power. However, if we compare these amounts with the single regression models, it seems that we now have a little higher explanatory power. Additionally, if we add 10 more financial ratios, our models would be able to explain up to around 4% and 6.2% of the variation in `RetEarly2020` and `RetLate2020` . According to these multi-variable regressions, we see that in 2020, firms with more cash (to cover current liabilities), less liquid assets, higher long-term debt, and higher reliance on labor on average have higher 3-month and 9-month returns. This is consistent with our earlier industry-fixed effect regression results.

3. Explaining Fluctuation Using Market Betas

Do pre-COVID risk measures (i.e., in 2019) explain variation in stock returns for early and late 2020?

In [ ]:
```
df = pd.read_csv('maindf.csv')
```

`RetEarly2020` and `RetLate2020` are the variables whose variation is what we're trying to explain.

The pre-COVID risk measure we chose is the market beta of each stock in 2019 (downloaded from WRDS CRSP).

In [ ]:
```
# examine missing values
df['beta19'].isna().sum()
```

Out[ ]:
```
92
```

There are 92 stocks with missing market beta. We will replace a stock's missing beta with the average beta of the GICS industry it belongs to.

In [ ]:
```
# check if there's any industry without any market beta
df.groupby('gind').beta19.count().sort_values()
```

Out[ ]:
```
gind
Transportation Infrastructure                      6
Tobacco                                            7
Industrial Conglomerates                           8
```

```
Distributors                                  9
Multiline Retail                             10
                                            ...
Equity Real Estate Investment Trusts (REITs)    173
Software                                    179
Oil, Gas & Consumable Fuels                 212
Biotechnology                               261
Banks                                       325
Name: beta19, Length: 69, dtype: int64
```

In [ ]:

```
# replace missing betas with industry average
means = df.groupby('sic').beta19.mean()
df['beta19'] = np.where(df['beta19'].isna(), means[df['sic']], df['beta19'])
```

In [ ]:

```
# double check that there's no missing value anymore
df['beta19'].isna().sum()
```

Out[ ]: 0

In [ ]:

```
# # export into a csv file
# df.to_csv('df_beta.csv', index=False)
```

Regress `RetEarly2020` and `RetLate2020` on `beta19` respectively.

In [ ]:

```
# regress RetEarly2020 on beta19
df2 = df[['ticker', 'RetEarly2020', 'RetLate2020', 'beta19']].copy()
df2['intercept'] = 1
Y = df2['RetEarly2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg1 = sm.OLS(Y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           RetEarly2020   R-squared:                       0.005
Model:                            OLS   Adj. R-squared:                  0.005
Method:                 Least Squares   F-statistic:                     22.74
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           1.91e-06
Time:                        23:02:47   Log-Likelihood:                -1932.7
No. Observations:                4130   AIC:                             3869.
Df Residuals:                    4128   BIC:                             3882.
```

```
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
beta19        -0.0398      0.008     -4.769      0.000      -0.056      -0.023
intercept     -0.2587      0.010    -25.036      0.000      -0.279      -0.238
==============================================================================
Omnibus:                     7974.878   Durbin-Watson:                   1.948
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         32690420.597
Skew:                          14.628   Prob(JB):                         0.00
Kurtosis:                     437.871   Cond. No.                         3.20
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# regress RetLate2020 on beta19
Y = df2['RetLate2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg2 = sm.OLS(Y, X).fit()
print(reg2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.000
Method:                 Least Squares   F-statistic:                    0.8684
Date:                Mon, 05 Jul 2021   Prob (F-statistic):              0.351
Time:                        23:02:47   Log-Likelihood:                -7746.9
No. Observations:                4130   AIC:                         1.550e+04
Df Residuals:                    4128   BIC:                         1.551e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
beta19         0.0318      0.034      0.932      0.351      -0.035       0.099
intercept      0.8775      0.042     20.779      0.000       0.795       0.960
==============================================================================
Omnibus:                     5405.634   Durbin-Watson:                   1.962
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1276127.512
Skew:                           7.224   Prob(JB):                         0.00
Kurtosis:                      87.894   Cond. No.                         3.20
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Interpretation

Market beta represents the sensitivity of the stock to the movement of the market. When we regress `RetEarly2020` on the 2019 market beta, the intercept suggests that for stocks completely free of systematic risk, the average 3-month return is -0.26. The coefficient of the market beta is -0.04. It means that for stocks that are as volatile as the market (β=1), the average 3-month return is -0.3. The more sensitive a stock is to the market's swing (larger β), the lower its 3-month return in early 2020. It's worth noting that the R^2 and adjusted R^2 of this regression model is 0.005, meaning it has little explanatory power. It can only explain 0.5% of the variation in `RetEarly2020` .

When we regress `RetLate2020` on the 2019 market beta, the intercept suggests that for stocks completely free of systematic risk, the average 9-month return is 0.88. The coefficient of the market beta is 0.03. However, its *p*-value is greater than 0.05, meaning this coefficient is not significantly different from 0. Therefore, we would say that the 2019 market beta has no effect on stocks' 9-month return in 2020. Indeed, we also see that the R^2 and adjusted R^2 of this regression model is 0, meaning it has no explanatory power.

4. Explaining Fluctuation Using Historical Volatility

Does the volatility in 2019 stock returns explain variation in stock returns for early and late 2020?

In [ ]:
```
df = pd.read_csv('df_beta.csv')
```

Preprocessing Stock Data from 2019

`stock19.csv` contains the stock data for all the U.S. companies in WRDS CRSP database.

- `date` is the date of the last trading day of each month in 2019.
- `ticker` is the ticker for each stock.
- `price` is the closing price on the last trading day in each month in 2019.
- `ret` is the holding period (monthly) return for each stock.

In [ ]:
```
stock19 = pd.read_csv('stock19.csv')
```

In [ ]:
```
# clean up the columns
```

```python
stock19.rename(columns={'TICKER': 'ticker', 'PRC': 'price', 'RET': 'ret'}, inplace=True)
stock19.date = pd.to_datetime(stock19.date, format="%Y%m%d")
stock19['month'] = pd.DatetimeIndex(stock19.date).month
```

In [ ]:

```python
# drop stocks that are not in the main dataframe
stock19 = stock19[stock19.ticker.isin(df.ticker)]
stock19.reset_index(drop=True, inplace=True)
```

There are 13 companies that have two sets (24) of monthly returns.

In [ ]:

```python
tmp = stock19.ticker.value_counts()
tmp[tmp.index[tmp.gt(12)]]
```

Out[ ]:

```
TAP     24
GEF     24
HVT     24
LEN     24
WSO     24
MKC     24
BIO     24
BH      24
STZ     24
AGM     24
CWEN    24
HEI     24
GTN     23
Name: ticker, dtype: int64
```

Cross examine two sets of stock prices from CRSP with those listed on Yahoo Finance and only keep the ones that match.

In [ ]:

```python
tics = " ".join(tmp.index[tmp.gt(12)].to_list())
df_yahoo = yf.download(tics, start="2018-12-31", end="2020-01-01", group_by='ticker')
dates = ['2019-01-31', '2019-02-28', '2019-03-29', '2019-04-30', '2019-05-31', '2019-06-28',
         '2019-07-31', '2019-08-30', '2019-09-30', '2019-10-31', '2019-11-29', '2019-12-31']

for i in tmp.index[tmp.gt(12)]:
    if df_yahoo[i].dropna().empty:
        continue
    else:
        try:
            prices = round(df_yahoo[i].loc[dates, 'Close'], 2).to_list()
            stock19.loc[stock19.ticker==i, 'price'] = stock19[stock19.ticker==i]['price'].apply(lambda x: x if round(x, 2) in pric
```

```
            stock19.dropna(subset=['price'], inplace=True)
            stock19.reset_index(drop=True, inplace=True)
        except:
            pass
```

```
[*********************100%***********************]  13 of 13 completed
```

In [ ]:

```
# make sure there's no stock with more than 12 monthly returns
tmp = stock19.ticker.value_counts()
len(tmp.index[tmp.gt(12)])
```

Out[ ]:

```
0
```

Linear Regression Models

Now, we will determine if the standard deviation of a stock's monthly returns in 2019 can help explain the variation in its returns in 2020.

In [ ]:

```
# calculate standard deviation of each stock's 2019 monthly returns
std = pd.DataFrame(stock19.groupby('ticker').ret.std().reset_index())
std.rename(columns={'ret': 'std'}, inplace=True)
std.head(2)
```

Out[ ]:

	ticker	std
0	A	0.077494
1	AA	0.109564

In [ ]:

```
# drop missing standard deviation
std.dropna(subset=['std'], inplace=True)
```

In [ ]:

```
# # export into a csv file
# std.to_csv('std.csv', index=False)
```

In [ ]:

```
# only keep the stocks that have data from 2019
df2 = df[['ticker', 'RetEarly2020', 'RetLate2020']].copy()
df2 = df2[df2.ticker.isin(std.ticker)]
```

```
df2 = df2.merge(std, on='ticker')
df2.reset_index(drop=True, inplace=True)
print("Number of unique tickers:", df2.ticker.nunique())
```

```
Number of unique tickers: 4096
```

In [ ]:

```
# regress RetEarly2020 on standard deviation
df2['intercept'] = 1
Y = df2['RetEarly2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg1 = sm.OLS(Y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           RetEarly2020   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.000
Method:                 Least Squares   F-statistic:                    0.8766
Date:                Mon, 05 Jul 2021   Prob (F-statistic):              0.349
Time:                        23:03:16   Log-Likelihood:                -1928.0
No. Observations:                4096   AIC:                             3860.
Df Residuals:                    4094   BIC:                             3873.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
std            0.0431      0.046      0.936      0.349      -0.047       0.133
intercept     -0.3047      0.008    -35.983      0.000      -0.321      -0.288
==============================================================================
Omnibus:                     7902.667   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         32285272.126
Skew:                          14.598   Prob(JB):                         0.00
Kurtosis:                     436.958   Cond. No.                         7.72
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

```
# regress RetLate2020 on standard deviation
Y = df2['RetLate2020']
X = df2.drop(columns=['ticker', 'RetEarly2020', 'RetLate2020'])
reg2 = sm.OLS(Y, X).fit()
print(reg2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            RetLate2020   R-squared:                       0.026
Model:                            OLS   Adj. R-squared:                  0.026
Method:                 Least Squares   F-statistic:                     110.3
Date:                Mon, 05 Jul 2021   Prob (F-statistic):           1.73e-25
Time:                        23:03:16   Log-Likelihood:                -7612.6
No. Observations:                4096   AIC:                         1.523e+04
Df Residuals:                    4094   BIC:                         1.524e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
std            1.9352      0.184     10.505      0.000       1.574       2.296
intercept      0.6569      0.034     19.364      0.000       0.590       0.723
==============================================================================
Omnibus:                     5331.506   Durbin-Watson:                   1.981
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1343597.714
Skew:                           7.112   Prob(JB):                         0.00
Kurtosis:                      90.580   Cond. No.                         7.72
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Interpretation

The standard deviation of stock returns is another measure of risk. When we regress `RetEarly2020` on the standard deviation of 2019 monthly returns, the intercept suggests that for stocks with no variability in returns, the average 3-month return is -0.30. The coefficient of the standard deviation is 0.04. However, its *p*-value is greater than 0.05, meaning this coefficient is not significantly different from 0. Therefore, we would say that the volatility of 2019 returns has no effect on stocks' 3-month return in early 2020. Indeed, we also see that the R^2 and adjusted R^2 of this regression model is 0, meaning it has no explanatory power.

When we regress `RetLate2020` on the standard deviation of 2019 monthly returns, the intercept suggests that stocks with no variability in returns, the average 9-month return is 0.66. The coefficient of the market beta is 1.94. It means that the more volatile a stock in 2019 (large standard deviation), the higher its 9-month return in 2020. The R^2 and adjusted R^2 of this regression model is 0.03, meaning it does have a little explanatory power and that it can explain around 3% of the variation in `RetLate2020` .

5. Predicting Returns Using DistilBERT Model and Business

Descriptions

We are missing the business descriptions for 1565 companies.

In [ ]:

```
df = pd.read_csv('df_beta.csv')
```

In [ ]:

```
# a list of companies without business description
noDes = df.loc[df.description.isna(), 'ticker'].unique()
len(noDes)
```

Out[ ]: 1565

Scrape the business descriptions from Yahoo Finance for these companies.

In [ ]:

```
# # WARNING!! - Don't run this loop. The results are already saved in "missing_des.csv"

# DES = []
# tickers = noDes
# for i in tickers:
#   url ='https://finance.yahoo.com/quote/'+i+'/profile'
#   page = requests.get(url)
#   htmldata = BeautifulSoup(page.content, 'html.parser')
#   Business_Description = htmldata.find('p', {'class':'Mt(15px) Lh(1.6)'})
#   DES.append(Business_Description)
```

In [ ]:

```
# # create new dataframe that stores tickers and their corresponding descriptions
# company_des = pd.DataFrame({'ticker':tickers, 'description':DES})

# # drop the stocks that do not have Yahoo Finance company profiles
# company_des.dropna(inplace=True)
# company_des['description'] = company_des['description'].astype(str)
```

In [ ]:

```
# # remove regex text from description
# a = np.arange(1,300)
# a = a.astype(str)
# for i in a:
#   company_des['description']=company_des['description'].str.replace('<p class="Mt(15px) Lh(1.6)" data-reactid="'+i+'">','',regex
```

```
# company_des['description']=company_des['description'].str.replace('</p>','',regex=False)
```

In [ ]:

```
# # export company_des into a CSV file
# company_des.to_csv('missing_des.csv', index=False)
```

Insert the missing descriptions into the main dataframe.

In [ ]:

```
# Load the newly scraped business descriptions
company_des = pd.read_csv('missing_des.csv')
company_des.head(2)
```

Out[ ]:

	ticker	description
0	AACG	ATA Creativity Global, together with its subsi...
1	AAMC	Altisource Asset Management Corporation, an as...

In [ ]:

```
# insert the newly scraped business descriptions into the main dataframe
tmp = df[['ticker', 'description']]
tmp = tmp.merge(company_des, on='ticker', how='outer')
tmp.description_x = np.where(tmp['description_x'].isna(), tmp['description_y'], tmp['description_x'])
df['description'] = tmp['description_x']
```

Drop 86 companies that still do not have without business descriptions.

In [ ]:

```
# how many companies still don't have their business descriptions?
print(df.shape)
df = df[~df.description.isna()]
print(df.shape)
```

```
(4130, 42)
(4044, 42)
```

In [ ]:

```
# # export into a csv file
# df.to_csv('df_des.csv', index=False)
```

Load a pre-trained distilBERT model.

In [ ]:
```
model_class, tokenizer_class, pretrained_weights = (ppb.DistilBertModel, ppb.DistilBertTokenizer, 'distilbert-base-uncased')

# Load pretrained model/tokenizer
tokenizer = tokenizer_class.from_pretrained(pretrained_weights)
model = model_class.from_pretrained(pretrained_weights)
```

Due to Colab's RAM limitations, limit the description size to 350 characters.

In [ ]:
```
df['description'] = df['description'].str.slice(0, 350)
```

Tokenize the business descriptions for BERT and pad all lists of tokenized values to the same size.

In [ ]:
```
tokenized = df['description'].apply((lambda x: tokenizer.encode(x, add_special_tokens=True)))

max_len = max(map(len, tokenized.values))
padded = np.array([i + [0]*(max_len-len(i)) for i in tokenized.values])
padded.shape
```

Out[ ]:
```
(4044, 113)
```

Create attention mask variable for DistilBERT to ignore the padding when it's processing its input.

In [ ]:
```
attention_mask = np.where(padded != 0, 1, 0)
attention_mask.shape
```

Out[ ]:
```
(4044, 113)
```

Run the pretrained DistilBERT model on the prepared predictor, save the result in `last_hidden_states` , and keep the first layer of the hidden states in `features` .

In [ ]:
```
# # WARNING!! - Don't run the DistilBERT model. The results are already saved in "features.npy"
# input_ids = torch.tensor(padded)
# attention_mask = torch.tensor(attention_mask)

# with torch.no_grad():
#     last_hidden_states = model(input_ids, attention_mask=attention_mask)
```

```
# features = last_hidden_states[0][:,0,:].numpy()
```

In [ ]:

```
# # save features into a npy file
# np.save('features', features)
```

Predicting Stock Returns Using Business Descriptions

In [ ]:

```
features = np.load('features.npy')
```

Create binary labels for `RetEarly2020` and `RetLate2020` .

- `BetterEarly2020` is 1, if a stock's `RetEarly2020` is in the top 35% (i.e., higher than 65% of the companies); otherwise, 0.
- `BetterLate2020` is 1, if a stock's `RetLate2020` is in the top 35% (i.e., higher than 65% of the companies); otherwise, 0.

In [ ]:

```
df['BetterEarly2020'] = 0
df['BetterLate2020'] = 0
df['BetterEarly2020'] = np.where(df.RetEarly2020>=df.RetEarly2020.quantile(0.65), 1, 0)
df['BetterLate2020'] = np.where(df.RetLate2020>=df.RetLate2020.quantile(0.65), 1, 0)
```

1416 stocks are labeled as performing better than most in early 2020 and late 2020 respectively.

In [ ]:

```
print(df.BetterEarly2020.value_counts())
print(df.BetterLate2020.value_counts())
```

```
0    2628
1    1416
Name: BetterEarly2020, dtype: int64
0    2628
1    1416
Name: BetterLate2020, dtype: int64
```

Logistic Regression Model for `BetterEarly2020`

Split the data into training and test sets (`random_state=870`).

Train the logistic regression models on the training set (75%) and evaluate its accuracy on the test set (25%).

In [ ]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# logistic regression for BetterEarly2020
X_train, X_test, Y_train, Y_test = train_test_split(features, df['BetterEarly2020'], test_size=0.25, random_state=870)
log = LogisticRegression(max_iter=5000)
log.fit(X_train, Y_train)
print(log.score(X_test, Y_test))
```

```
0.712166172106825
```

In predicting `BetterEarly2020` for the test set, our model has an accuracy score of **0.71**.

Check if this approach works better than a random guess.

In [ ]:

```
from sklearn.dummy import DummyClassifier
from sklearn.model_selection import cross_val_score

# accuracy of a random guess
clf = DummyClassifier()
scores = cross_val_score(clf, X_train, Y_train)
print("Dummy classifier score: %0.3f (+/- %0.2f)" % (scores.mean(), scores.std()*2))
```

```
Dummy classifier score: 0.526 (+/- 0.03)
/usr/local/lib/python3.7/dist-packages/sklearn/dummy.py:132: FutureWarning: The default value of strategy will change from stratif
ied to prior in 0.24.
  "stratified to prior in 0.24.", FutureWarning)
```

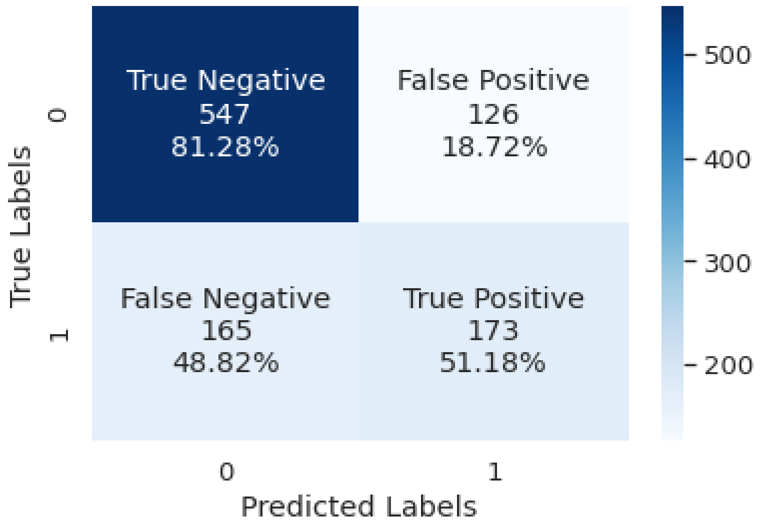
Create a confusion matrix.

In [ ]:

```
predictions = log.predict(X_test)
matrix = confusion_matrix(Y_test, predictions)

sns.set(font_scale=1.2)
group_names = ['True Negative', 'False Positive', 'False Negative', 'True Positive']
group_counts = ['{0:0.0f}'.format(value) for value in matrix.flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in np.array([row/np.sum(row) for row in matrix]).flatten()]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
```

```
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show();
```



Logistic Regression Model for `BetterLate2020`

In [ ]:

```
# logistic regression for BetterLate2020
X_train, X_test, Y_train, Y_test = train_test_split(features, df['BetterLate2020'], test_size=0.25, random_state=870)
log2 = LogisticRegression(max_iter=5000)
log2.fit(X_train, Y_train)
print(log2.score(X_test, Y_test))
```

```
0.6805143422354105
```

In predicting `BetterLate2020` for the test set, our model has an accuracy score of **0.68**.

In [ ]:

```
# accuracy of a random guess
clf = DummyClassifier()
scores = cross_val_score(clf, X_train, Y_train)
print("Dummy classifier score: %0.3f (+/- %0.2f)" % (scores.mean(), scores.std()*2))
```

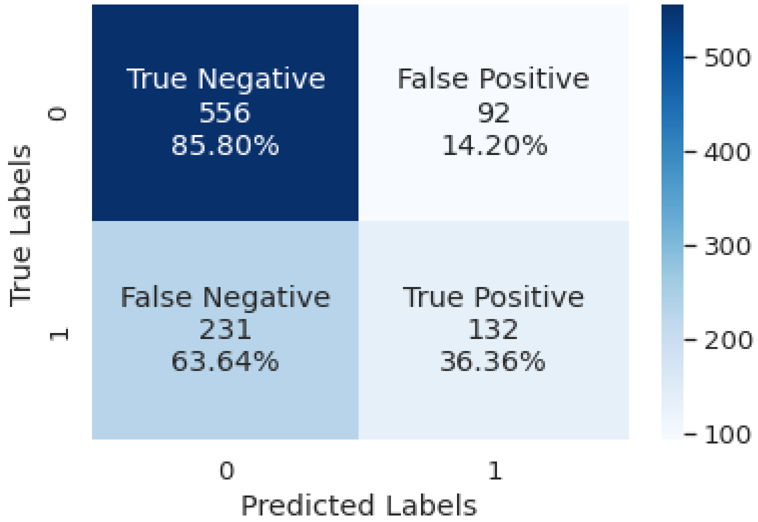
```
Dummy classifier score: 0.553 (+/- 0.04)
/usr/local/lib/python3.7/dist-packages/sklearn/dummy.py:132: FutureWarning: The default value of strategy will change from stratif
ied to prior in 0.24.
```

```
"stratified to prior in 0.24.", FutureWarning)
```

Create a confusion matrix.

```
In [ ]:
predictions = log2.predict(X_test)
matrix = confusion_matrix(Y_test, predictions)

sns.set(font_scale=1.2)
group_names = ['True Negative', 'False Positive', 'False Negative', 'True Positive']
group_counts = ['{0:0.0f}'.format(value) for value in matrix.flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in np.array([row/np.sum(row) for row in matrix]).flatten()]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show();
```



Interpretation

The logistic regressions built on the output of DistilBERT model have a decent amount of power (better than random guesses) in predicting whether a stock performed better than most in 2020. Given the business description of a company, our models are able to predict whether its 3-month and 9-month returns in 2020 are higher than 65% of the stocks. The models have an accuracy of 0.71 in predicting stock performance in early 2020 and an accuracy of 0.68 in predicting stock performance in late 2020. However, one limitation is that we could not easily tell which types of business description lead to better stock performance and which types do not.

6. Putting Everything Together

In [ ]:

```
df_cleaned = pd.read_csv('df_beta.csv')
df_ratios = pd.read_csv('df_ratios_cleaned.csv')
std = pd.read_csv('std.csv')
df_des = pd.read_csv('df_des.csv')
features = pd.DataFrame(np.load('features.npy'))
```

In [ ]:

```
# merge dataframes into one
print(df_cleaned.shape)
cols = ['ticker']
cols.extend(list(df_ratios.columns[42:]))
df = df_cleaned.merge(df_ratios[cols], on='ticker')
df = df.merge(std, on='ticker')
df_des = pd.concat([df_des['ticker'], features], axis=1)
df = df.merge(df_des, on='ticker')
print(df.shape)
```

```
(4130, 42)
(3967, 823)
```

In [ ]:

```
df.head(1)
```

Out[ ]:

	ticker	RetEarly2020	RetLate2020	SPEarly2020	SPLate2020	beta19	gsector	ggroup	gind	gsubind	naics	nsector	si
0	JJSF	-0.340234	0.30034	-0.20001	0.453255	0.01282	Consumer Staples	Food, Beverage & Tobacco	Food Products	30202030	311812.0	Manufacturing	2050.

1 rows × 823 columns

Linear Regression Models

`df` is our final dataframe that has 3967 stocks with all kinds of data.

In [ ]:
```
# create industry indicators
df2 = df['ggroup'].copy()
df2 = pd.get_dummies(df2, columns=['ggroup'], prefix='', prefix_sep='')
df2.head(1)
```

Out[ ]:

	Automobiles & Components	Banks	Capital Goods	Commercial & Professional Services	Communication Services	Consumer Durables & Apparel	Consumer Services	Diversified Financials	Energy	Food & Staples Retailing	Food, Beverage & Tobacco	Health Care Equipment & Services	Hou & Pe Pr
0	0	0	0	0	0	0	0	0	0	0	1	0	

In [ ]:
```
# a list of 15 financial ratios
print(ratios)
```

```
['roa19', 'atr19', 'ros19', 'roe19', 'emulti19', 'ai19', 'gmargin19', 'cta19', 'cash19', 'quick19', 'lda19', 'se19', 'T1', 'T2',
'T3']
```

In [ ]:
```
# add a constant
df['intercept'] = 1
```

Now that we have a well-set-up dataframe, we can put all the explanatory variables we've inspected together and see if we can better explain the variation in `RetEarly2020` and `RetLate2020` .

In [ ]:
```
# explain variation in RetEarly2020
Y = df['RetEarly2020']
X = pd.concat([
               df2, # industry indicators
        #    df['intercept'],
        #    df[ratios],
        #    df[['roa19', 'lda19', 'roe19']],
        #    df.se19,
        #    df['beta19'],
```

```
#    df['std'],
     df[df.columns[-769:-1]] # business descriptions
     ], axis=1)
reg = sm.OLS(Y, X).fit()
print('R-squared:', round(reg.rsquared, 3))
print('Adj. R-squared:', round(reg.rsquared_adj, 3))
```

```
R-squared: 0.552
Adj. R-squared: 0.44
```

In [ ]:

```
# explain variation in RetLate2020
Y = df['RetLate2020']
X = pd.concat([
               df2, # industry indicators
            #    df['intercept'],
            #    df[ratios],
            #    df[['roa19', 'lda19', 'roe19']],
            #    df.se19,
            #    df['beta19'],
            #    df['std'],
               df[df.columns[-769:-1]] # business descriptions
               ], axis=1)
reg = sm.OLS(Y, X).fit()
print('R-squared:', round(reg.rsquared, 3))
print('Adj. R-squared:', round(reg.rsquared_adj, 3))
# print(pd.DataFrame(reg.params[:-769])], columns=['coefficient']))
```

```
R-squared: 0.519
Adj. R-squared: 0.399
```

As it turns out, the models with the highest explanatory power (R^2 and adjusted R^2) are the ones built with GICS industry group indicators and business descriptions. 44% of the variation in `RetEarly2020` and about 40% of the variation in `RetLate2020` can be explained by these models. Including financial ratios and risk measures in the models does not significantly improve their explanatory power.

Logistic Regression Models

We can again try to predict whether a stock performed better than others in 2020, using all the explanatory variables at hand.

- `BetterEarly2020` is 1, if a stock's `RetEarly2020` is in the top 35% (i.e., higher than 65% of the companies); otherwise, 0.
- `BetterLate2020` is 1, if a stock's `RetLate2020` is in the top 35% (i.e., higher than 65% of the companies); otherwise, 0.

In [ ]:

```python
df['BetterEarly2020'] = 0
df['BetterLate2020'] = 0
df['BetterEarly2020'] = np.where(df.RetEarly2020>=df.RetEarly2020.quantile(0.65), 1, 0)
df['BetterLate2020'] = np.where(df.RetLate2020>=df.RetLate2020.quantile(0.65), 1, 0)
```

In [ ]:

```python
print(df.BetterEarly2020.value_counts())
print(df.BetterLate2020.value_counts())
```

```
0    2578
1    1389
Name: BetterEarly2020, dtype: int64
0    2578
1    1389
Name: BetterLate2020, dtype: int64
```

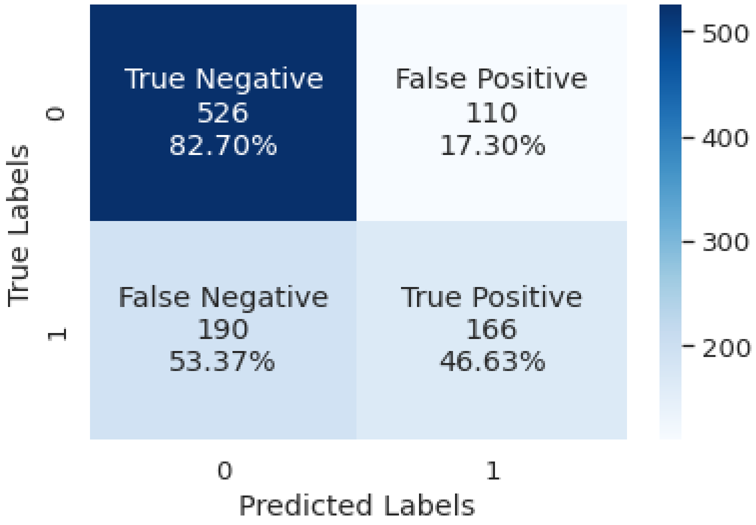
In [ ]:

```python
# predicting BetterEarly2020
Y = df['BetterEarly2020']
X = pd.concat([
               df2, # industry indicators
               df[ratios],
               df['beta19'],
               df['std'],
               df[df.columns[-771:-3]] # business descriptions
               ], axis=1)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=870)
log = LogisticRegression(max_iter=100000)
log.fit(X_train, Y_train)
print('Accuracy:', log.score(X_test, Y_test))

predictions = log.predict(X_test)
matrix = confusion_matrix(Y_test, predictions)
sns.set(font_scale=1.2)
group_names = ['True Negative', 'False Positive', 'False Negative', 'True Positive']
group_counts = ['{0:0.0f}'.format(value) for value in matrix.flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in np.array([row/np.sum(row) for row in matrix]).flatten()]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show();
```

```
Accuracy: 0.6975806451612904
```



In [ ]:

```
# accuracy of a random guess
clf = DummyClassifier()
scores = cross_val_score(clf, X_train, Y_train)
print("Dummy classifier score: %0.3f (+/- %0.2f)" % (scores.mean(), scores.std()*2))
```

```
Dummy classifier score: 0.566 (+/- 0.03)
/usr/local/lib/python3.7/dist-packages/sklearn/dummy.py:132: FutureWarning: The default value of strategy will change from stratif
ied to prior in 0.24.
  "stratified to prior in 0.24.", FutureWarning)
```

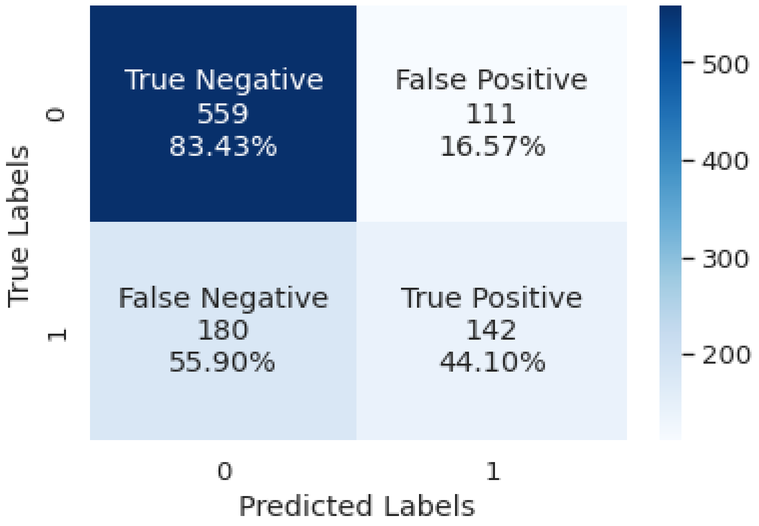
In [ ]:

```
# logistic regression for BetterLate2020
Y = df['BetterLate2020']
X = pd.concat([
               df2, # industry indicators
               df[ratios],
               df['beta19'],
               df['std'],
               df[df.columns[-771:-3]] # business descriptions
               ], axis=1)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=870)
log2 = LogisticRegression(max_iter=20000)
log2.fit(X_train, Y_train)
print('Accuracy:', log2.score(X_test, Y_test))
```

```
predictions = log2.predict(X_test)
matrix = confusion_matrix(Y_test, predictions)
sns.set(font_scale=1.2)
group_names = ['True Negative', 'False Positive', 'False Negative', 'True Positive']
group_counts = ['{0:0.0f}'.format(value) for value in matrix.flatten()]
group_percentages = ['{0:.2%}'.format(value) for value in np.array([row/np.sum(row) for row in matrix]).flatten()]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(group_names, group_counts, group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show();
```

```
Accuracy: 0.7066532258064516
```



In [ ]:

```
# accuracy of a random guess
clf = DummyClassifier()
scores = cross_val_score(clf, X_train, Y_train)
print("Dummy classifier score: %0.3f (+/- %0.2f)" % (scores.mean(), scores.std()*2))
```

```
Dummy classifier score: 0.552 (+/- 0.03)
/usr/local/lib/python3.7/dist-packages/sklearn/dummy.py:132: FutureWarning: The default value of strategy will change from stratif
ied to prior in 0.24.
  "stratified to prior in 0.24.", FutureWarning)
```

This time, we used all variables availabe in the logistic regression models - 24 GICS industry group indicators, 15 financial ratios, 2019 market betas, 2019 return standard deviations, and business descriptions. Similar as before, including more variables does not significantly improve our prediction accuracy. Our models do a good job predicting whether a stock' 3-month and 9month returns in 2020 are higher than 65% of the stocks. The accuracy for predicting performance in early 2020 is 0.704 and the accuracy for predicting performance in late 2020 is 0.708.

Conclusions

In industry-fixed regressions, we see that GICS codes have higher explanatory power than NAICS and SIC. To sum up, these are the industries that did the best and worst during the initial COVID shock and during market recovery.

- Least impacted by COVID shock: **Service** industry (especially **Health Care**)
- Most impacted by COVID shock: **Energy** industry (due to decline in transporation demand)
- Strongest rebound: **Automobiles & Retail** industries
- Slowest recovery: **Utilities** industry (historically relatively stable market)

Standing at the end of 2019, we observe that companies with the following characteristics tend to perform better in 2020.

- Have more cash (to cover current liabilities) at the end of 2019
- Have higher long-term debt to assets ratio at the end of 2019
- Have Higher reliance on labor at the end of 2019

In particular, stocks that have higher 3-month return are on average less sensitive to market swings. (They have lower 2019 market betas.) Stocks that have higher 9-month return are on average more volatile. (The standard deviations of their 2019 returns are higher.)

Last but not least, it turns out that business descriptions and industry indicators alone provide a decent amount of explanatory power in explaining the market fluctuation and are useful for predicting stock performance level.