US Financial Performance during Pandemic Using Clustering

Haochen Li , Chung Shun Man

Group Project
Smart Data Analytics
University of St. Gallen

OUTLINE

1 - Introduction of Financial Performance during Pandemic

- Potential Impact of Covid-19
- General overview of S&P 500

2 - Data Collection

- Webscraping from Wiki for stocks
- Webscraping from Yahoo Finance for time-series data

3 - 4 Steps to Cluster Companies and Industries

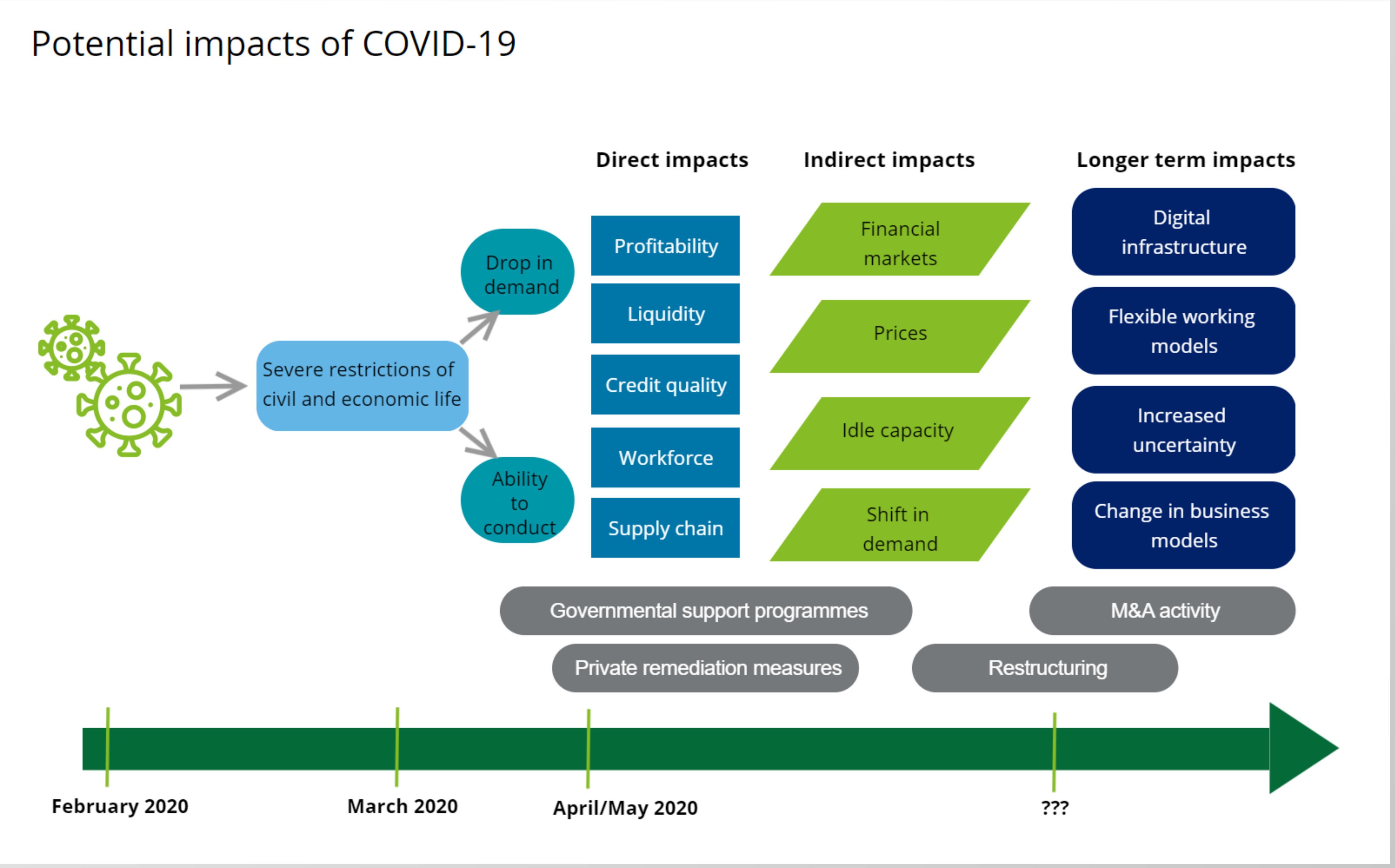
- Set-up
- Hierarchical Clustering based on industries
- Clustering based on all 500 observations
- Clustering based on all 500 observations (replaced by industries)

5 - Conclusion

- Analysis of results

Introduction

The COVID-19 pandemic is a major event with no precedence in recent history, which will have significant effects on many businesses all over the globe for an undetermined period of time.



Source:https://www2.deloitte.com/ch/en/pages/audit/articles/financial-reporting-survey-q1-2020.html

Introduction

Stock Prices, could be a good indicator for how pandemic affects global society. In our study, we choose the US market and S&P 500 stocks as the targets. The study period is from the first case in the US reported until recent date.

Source:https://edition.cnn.com/business/markets/coronavirus

Data Collecting

The S&P 500 includes 505 stocks from different industries. First, we need to collect the stock names and their industries from Wiki.

```
def save_sp500_tickers():
  resp = requests.get('http://en.wikipedia.org/wiki/List_of_S%26P_500_companies')
  soup = bs.BeautifulSoup(resp.text, 'lxml')
  table = soup.find('table', {'class': 'wikitable sortable'})
  tickers = []
  sectors = []
  for row in table.findAll('tr')[1:]:
     ticker = row.findAll('td')[0].text
     ticker = ticker[:-1]
     if "." in ticker:
         ticker = ticker.replace('.','-')
         print('ticker replaced to', ticker)
     tickers.append(ticker)
     sector = row.findAll('td')[3].text
     sectors.append(sector)
  with open("sp500tickers.pickle","wb") as f:
     pickle.dump(tickers,f)

  return tickers, sectors
```

Data Collecting

Second, use the company names as index to collect price data from Yahoo Finance.

```
def get_data_from_yahoo(reload_sp500=False):
    if reload_sp500:
        tickers = save_sp500_tickers()
    else:
        with open("sp500tickers.pickle", "rb") as f:
            tickers = pickle.load(f)
    if not os.path.exists('stock_dfs'):
        os.makedirs('stock_dfs')

    start = datetime(2020, 1, 20)
    end = dt.datetime.now()

    for ticker in tickers:
        # just in case your connection breaks, we'd like to save our progress!
        if not os.path.exists('stock_dfs/{}.csv'.format(ticker)):
            df = web.DataReader(ticker, 'yahoo', start, end)
            df.reset_index(inplace=True)
            df.set_index("Date", inplace=True)
            #df = df.drop("Symbol", axis=1)
            df.to_csv('stock_dfs/{}.csv'.format(ticker))
        else:
            print('Already have {}'.format(ticker))
```

Data Collecting

Second, use the company names as index to collect price data from Yahoo Finance.

```
def get_data_from_yahoo(reload_sp500=False):
    if reload_sp500:
        tickers = save_sp500_tickers()
    else:
        with open("sp500tickers.pickle", "rb") as f:
            tickers = pickle.load(f)
    if not os.path.exists('stock_dfs'):
        os.makedirs('stock_dfs')

    start = datetime(2020, 1, 20)
    end = dt.datetime.now()

    for ticker in tickers:
        # just in case your connection breaks, we'd like to save our progress!
        if not os.path.exists('stock_dfs/{}.csv'.format(ticker)):
            df = web.DataReader(ticker, 'yahoo', start, end)
            df.reset_index(inplace=True)
            df.set_index("Date", inplace=True)
            #df = df.drop("Symbol", axis=1)
            df.to_csv('stock_dfs/{}.csv'.format(ticker))
        else:
            print('Already have {}'.format(ticker))
```

4 steps to Cluster Companies and Industries

- Set-up
- Hierarchical Clustering based on industries
- Clustering based on all 500 observations
- Clustering based on all 500 observations (replaced by industries)

1. Set up

We have 500 observations, each of which belongs to a specific industry.

First, we use the group.by method to find the mean return and mean variance of all observations in an industry so that we can create hierarchical clustering.

Second, we consider all 500 observations and divide them into 4 clusters to see which companies have the closest performances

Python codes for Set-up

```
pdf=pd.read_csv('sp500_dataset.csv')
industry= pdf.groupby(['sector'])['annual_return_log', 'Std' ].mean()
industry = industry.reset_index()

featureset = industry[['annual_return_log']]
feature_mtx = featureset.values

feature500=pdf[['annual_return_log','Std']]
feature500_mtx=feature500.values
```

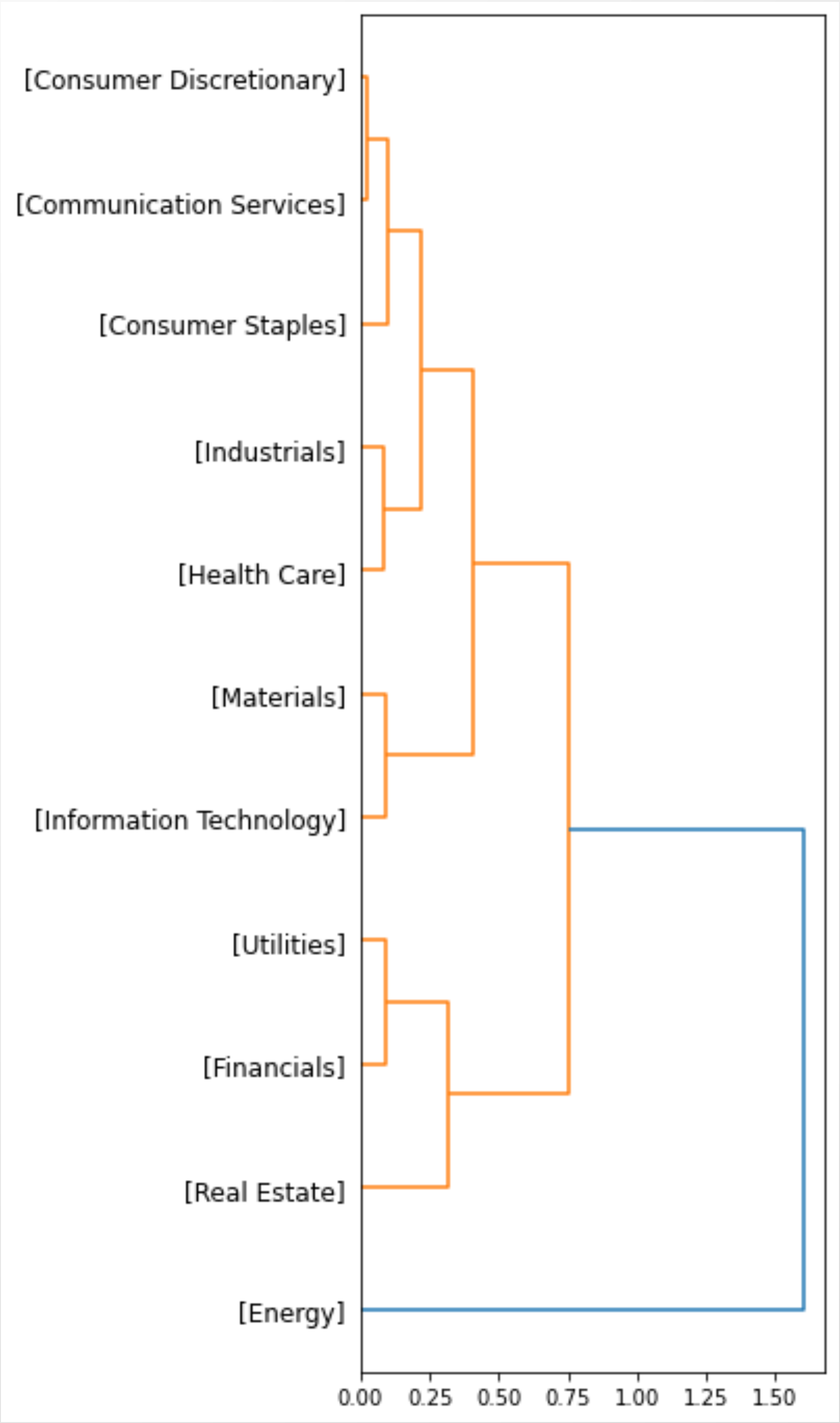
2. Hierarchical Clustering - Concept

What kind of hierarchical clustering is used?

We use agglomerative clustering, which is a down-to-top approach. For example,

- There are n observations. We find the closest pair and group them together as a single observation.
- Now, there are n-1 observations. We repeat the process as in the first step.
- We finish when all points are connected.

2. Hierarchical Clustering



- ‘Consumer Discretionary’ and ‘Communication’ are the closest and therefore are grouped together.
- Next, ‘Health care’ and the pair mentioned above are the closest and are therefore put together.
- We continue until all categories are connected.

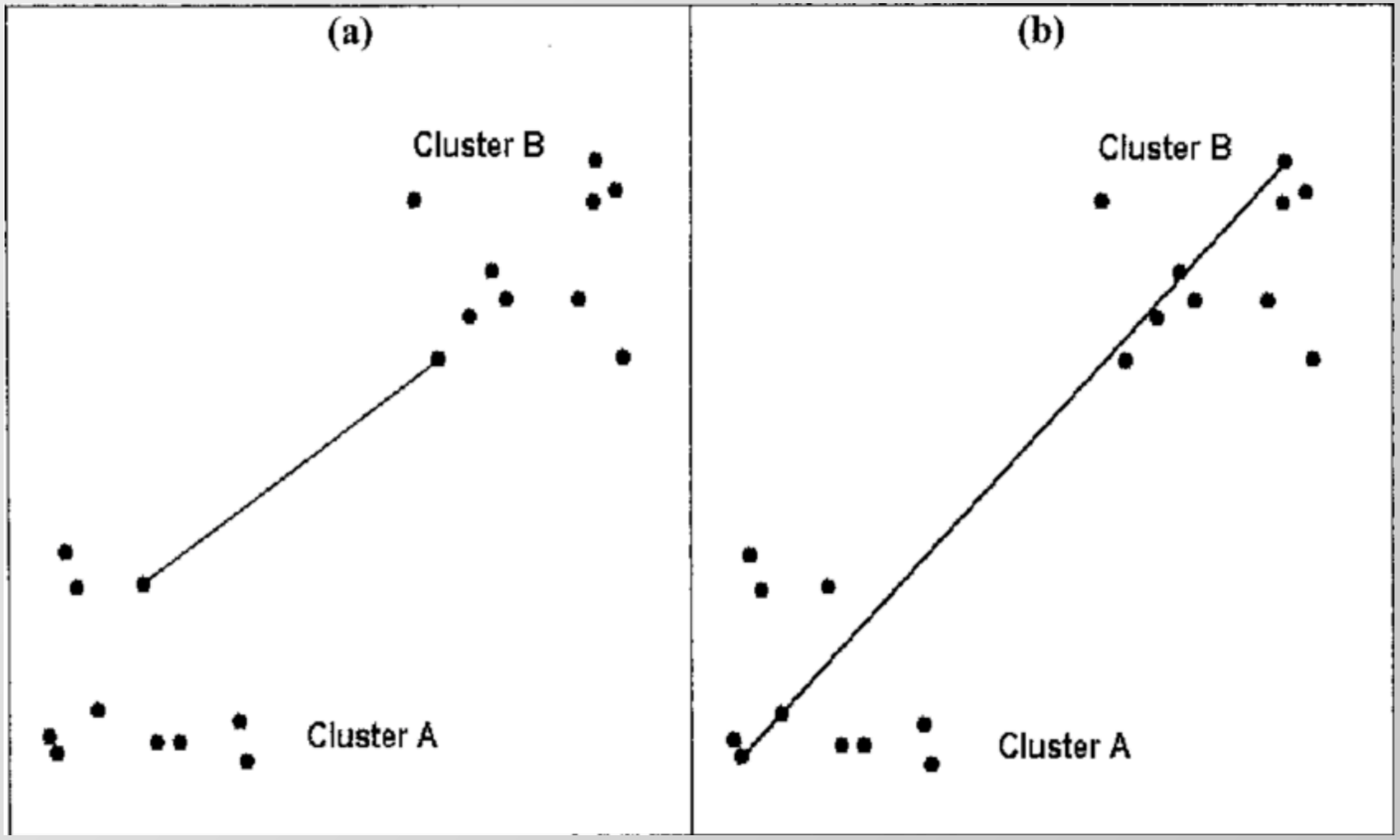
Python codes for clustering based on observations

```
import scipy
leng = feature_mtx.shape[0]

D = scipy.zeros([leng,leng])
for i in range(leng):
    for j in range(leng):
        D[i,j] = scipy.spatial.distance.euclidean(feature_mtx[i], feature_mtx[j])
import pylab
import scipy.cluster.hierarchy
Z = hierarchy.linkage(D, 'complete')

from scipy.cluster.hierarchy import fcluster
max_d = 3
clusters = fcluster(Z, max_d, criterion='distance')
clusters

#plot the denrogram
fig = pylab.figure(figsize=(4, 12))
def llf(id):
    return '[%s]' % ( (str(industry['sector'][id])) )
dendro = hierarchy.dendrogram(Z,  leaf_label_func=llf, leaf_rotation=0, leaf_font_size =12,
orientation = 'right')
```

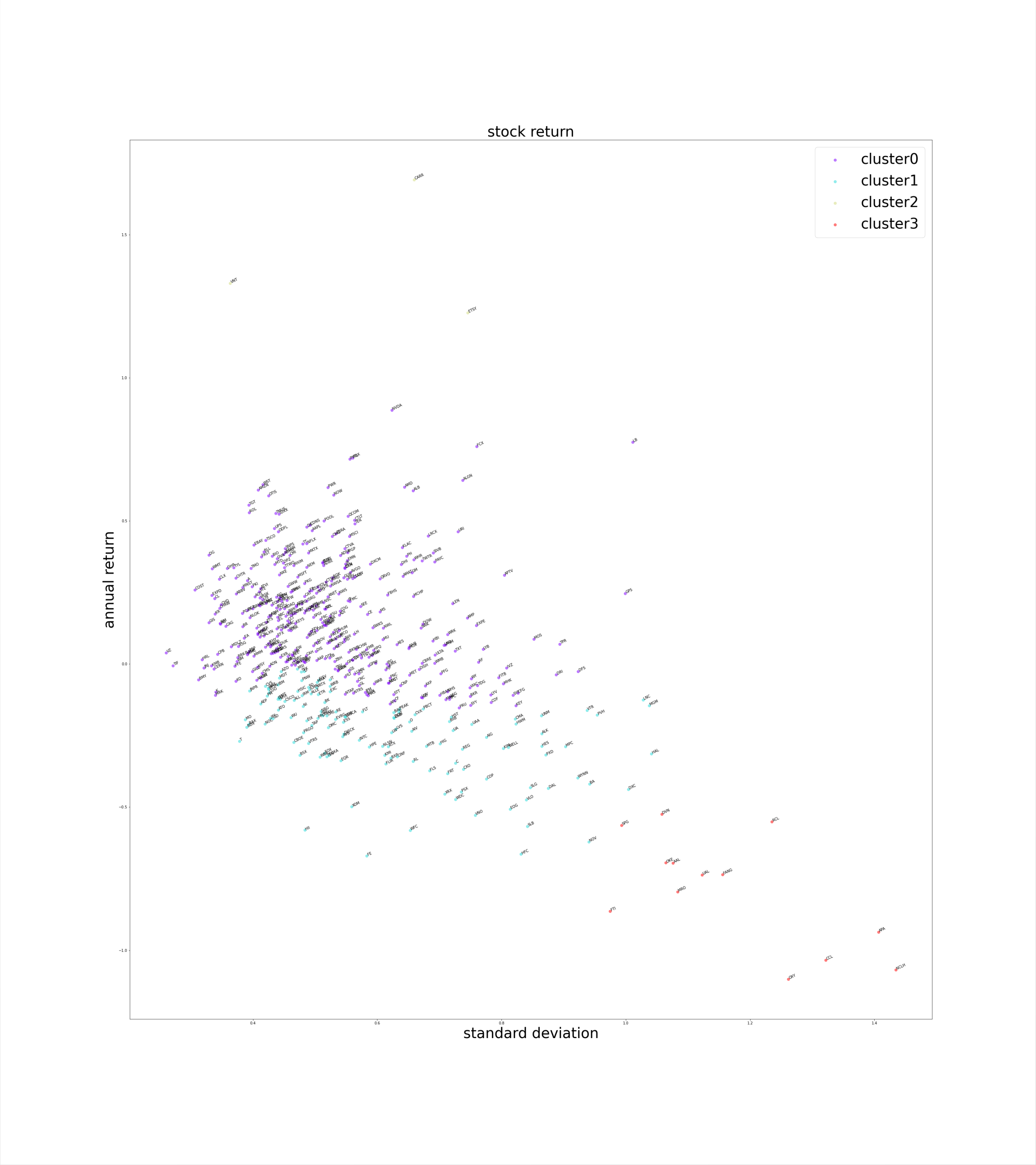
3. Clustering by complete distance - Concept

In our sample, we use the 'complete' distance method to group data. So the distance is based on a observation in a cluster which is the furtherest away from the one in another cluster.

The graph below compares two kinds of linkage: (a) Single linkage (b) Complete linkage.



3. Clustering by complete distance



Variables used to calculated the distance matrix:

- Stock return
- Stock variance

We set the number of clusters as 4.

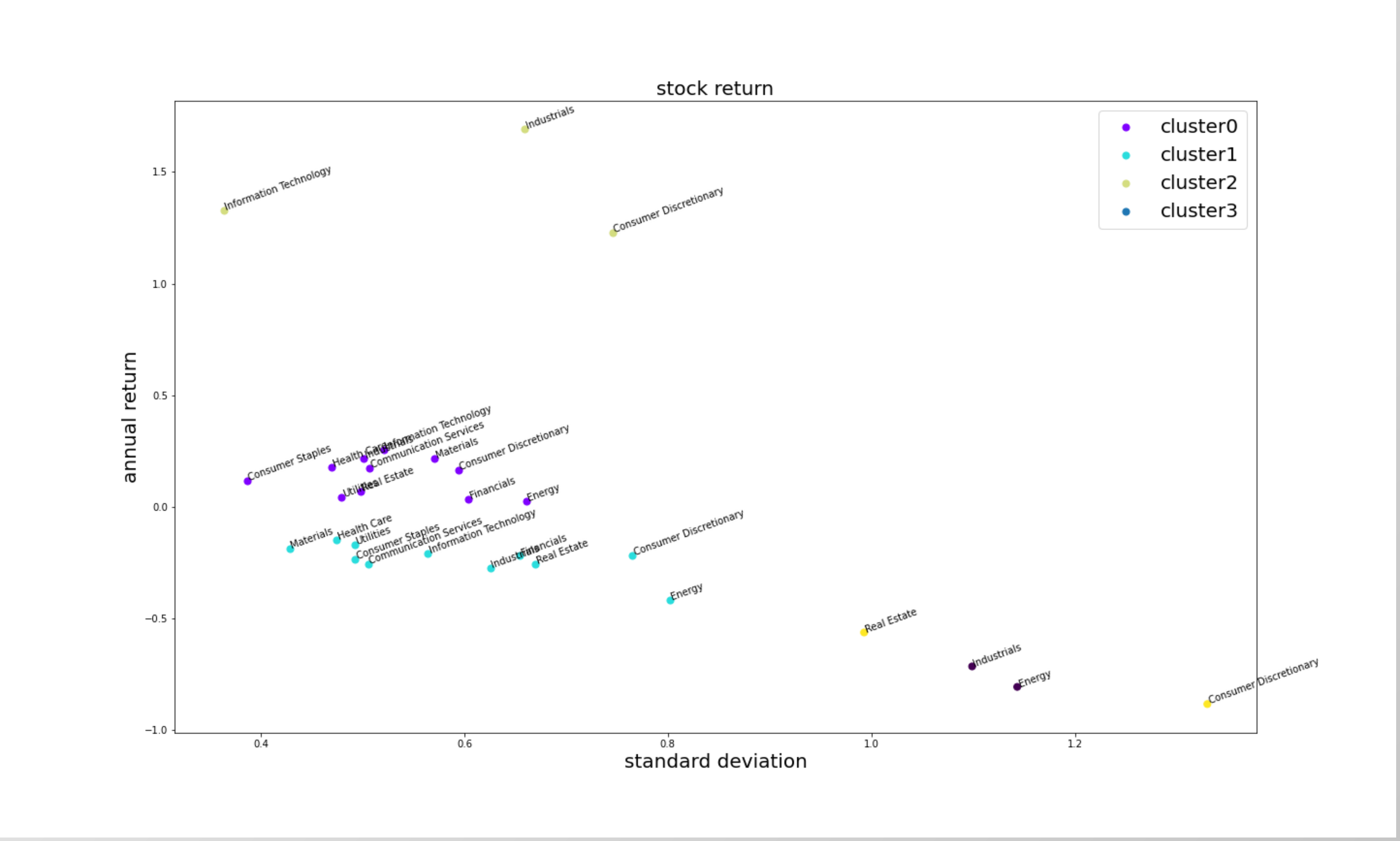
We can see that two clusters are in the centre (purple and light blue) and two are in SE and NW (orange and light green)

Python code for clustering based on observations

```
dist_matrix = distance_matrix(feature500_mtx,feature500_mtx)
agglom = AgglomerativeClustering(n_clusters = 4, linkage = 'complete')
agglom.fit(feature500_mtx)
agglom.labels_
pdf['cluster_'] = agglom.labels_
pdf.head()

import matplotlib.cm as cm
n_clusters = max(agglom.labels_)+1
colors = cm.rainbow(np.linspace(0, 1, n_clusters))
cluster_labels = list(range(0, n_clusters))
plt.figure(figsize=(40,45))

for color, label in zip(colors, cluster_labels):
    subset = pdf[pdf.cluster_ == label]
    for i in subset.index:
            plt.text(subset.Std[i], subset.annual_return_log[i], str(subset['Company'][i]), rotation=25)
    plt.scatter(subset.Std, subset.annual_return_log, s= 50, c=color,
label='cluster'+str(label),alpha=0.5)

plt.legend()
plt.title('stock return')
plt.xlabel('standard deviation')
plt.ylabel('annual return')
```

4. Clustering by complete distance (industry modified)

The result here is similar to the previous one.
We only identify the industries that each observation belongs to.



Python code for clustering based on observations (modified by industries)

```
df2= pdf.groupby(['cluster_', 'sector'])['annual_return_log', 'Std' ].mean()

plt.figure(figsize=(20,12))
for color, label in zip(colors, cluster_labels):
    subset = df2.loc[(label,),]
    for i in subset.index:
        plt.text(subset.loc[i][1], subset.loc[i][0], str(i), rotation=20)
    plt.scatter(subset.Std, subset.annual_return_log, s=20, c=color, label='cluster'+str(label))
plt.legend()
plt.title('stock return')
plt.xlabel('standard deviation')
plt.ylabel('annual return')
```

Conclusion

Based on stock returns and volatility during pandemic, stocks are distributed to 4 clusters, which have different performance.

There could be big variance among industries, though they have similar feature at most time.

Companies from IT, Industrial, Energy, Consumer Discretionary are more likely to have divergent performance.