Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Case Study - Clustering Stocks using k-Means

Name: Chuay Xiang Ze

IC Number: 021224070255

Date : 27/07/2023

Introduction : Learning about clustering with kMeans

Conclusion : Managed to complete tasks using real dataset.

Clustering stocks using KMeans

In this exercise, you'll cluster companies using their daily stock price movements (i.e. the dollar difference between the closing and opening prices for each trading day). You are given a NumPy array `movements` of daily price movements from 2010 to 2015, where each row corresponds to a company, and each column corresponds to a trading day.

Some stocks are more expensive than others. To account for this, include a `Normalizer` at the beginning of your pipeline. The Normalizer will separately transform each company's stock price to a relative scale before the clustering begins.

Normalizer vs StandardScaler

Note that `Normalizer()` is different to `StandardScaler()`, which you used in the previous exercise. While `StandardScaler()` standardizes **features** (such as the features of the fish data from the previous exercise) by removing the mean and scaling to unit variance, `Normalizer()` rescales **each sample** - here, each company's stock price - independently of the other.

This dataset was obtained from the Yahoo! Finance API.

Step 1: Load the data *(written for you)*

In [3]:

```
import pandas as pd

fn = './company-stock-movements-2010-2015-incl.csv'
stocks_df = pd.read_csv(fn, index_col=0)
```

Out[3]:

	2010-01-04	2010-01-05	2010-01-06	2010-01-07	2010-01-08	2010-01-11	2010-01-12	2010-01-13
Apple	0.580000	-0.220005	-3.409998	-1.170000	1.680011	-2.689994	-1.469994	2.779997
AIG	-0.640002	-0.650000	-0.210001	-0.420000	0.710001	-0.200001	-1.130001	0.069999
Amazon	-2.350006	1.260009	-2.350006	-2.009995	2.960006	-2.309997	-1.640007	1.209999
American express	0.109997	0.000000	0.260002	0.720002	0.190003	-0.270001	0.750000	0.300004
Boeing	0.459999	1.770000	1.549999	2.690003	0.059997	-1.080002	0.360000	0.549999
Bank of America	0.450000	0.460001	0.180000	0.250000	-0.199999	-0.060000	-0.359998	0.190001
British American Tobacco	0.180000	0.220001	0.040001	0.250000	-0.360001	-0.099999	0.570000	-0.139999
Canon	0.730000	0.369999	-0.099999	-0.169999	0.030003	0.110001	-0.079998	0.140000

Step 2: Inspect the first few rows of the DataFrame `stocks_df` by calling its `head()` function.

In [4]:

```
stocks_df.head()
```

Out[4]:

	2010-01-04	2010-01-05	2010-01-06	2010-01-07	2010-01-08	2010-01-11	2010-01-12	2010-01-13
Apple	0.580000	-0.220005	-3.409998	-1.170000	1.680011	-2.689994	-1.469994	2.779997
AIG	-0.640002	-0.650000	-0.210001	-0.420000	0.710001	-0.200001	-1.130001	0.069999
Amazon	-2.350006	1.260009	-2.350006	-2.009995	2.960006	-2.309997	-1.640007	1.209999
American express	0.109997	0.000000	0.260002	0.720002	0.190003	-0.270001	0.750000	0.300004
Boeing	0.459999	1.770000	1.549999	2.690003	0.059997	-1.080002	0.360000	0.549999

5 rows × 963 columns

Step 3: Extract the NumPy array `movements` from the DataFrame and the list of company names (*written for you*)

In [11]:

```
movements = stocks_df.values
companies = stocks_df.index
print(movements, companies)
```

```
[[ 5.8000000e-01 -2.2000500e-01 -3.4099980e+00 ... -5.3599620e+00
   8.4001900e-01 -1.9589981e+01]
 [-6.4000200e-01 -6.5000000e-01 -2.1000100e-01 ... -4.0001000e-02
  -4.0000200e-01  6.6000000e-01]
 [-2.3500060e+00  1.2600090e+00 -2.3500060e+00 ...  4.7900090e+00
  -1.7600090e+00  3.7400210e+00]
 ...
 [ 4.3000100e-01  2.2999600e-01  5.7000000e-01 ... -2.6000200e-01
   4.0000100e-01  4.8000300e-01]
 [ 9.0000000e-02  1.0000000e-02 -8.0000000e-02 ... -3.0000000e-02
   2.0000000e-02 -3.0000000e-02]
 [ 1.5999900e-01  1.0001000e-02  0.0000000e+00 ... -6.0001000e-02
   2.5999800e-01  9.9998000e-02]] Index(['Apple', 'AIG', 'Amazon', 'Americ
an express', 'Boeing',
       'Bank of America', 'British American Tobacco', 'Canon', 'Caterpilla
r',
       'Colgate-Palmolive', 'ConocoPhillips', 'Cisco', 'Chevron',
       'DuPont de Nemours', 'Dell', 'Ford', 'General Electrics',
       'Google/Alphabet', 'Goldman Sachs', 'GlaxoSmithKline', 'Home Depo
t',
       'Honda', 'HP', 'IBM', 'Intel', 'Johnson & Johnson', 'JPMorgan Chas
e',
       'Kimberly-Clark', 'Coca Cola', 'Lookheed Martin', 'MasterCard',
       'McDonalds', '3M', 'Microsoft', 'Mitsubishi', 'Navistar',
       'Northrop Grumman', 'Novartis', 'Pepsi', 'Pfizer', 'Procter Gambl
e',
       'Philip Morris', 'Royal Dutch Shell', 'SAP', 'Schlumberger', 'Son
y',
       'Sanofi-Aventis', 'Symantec', 'Toyota', 'Total',
       'Taiwan Semiconductor Manufacturing', 'Texas instruments', 'Unileve
r',
       'Valero Energy', 'Walgreen', 'Wells Fargo', 'Wal-Mart', 'Exxon',
       'Xerox', 'Yahoo'],
      dtype='object')
```

Step 4: Make the necessary imports:

- `Normalizer` from `sklearn.preprocessing`.
- `KMeans` from `sklearn.cluster`.
- `make_pipeline` from `sklearn.pipeline`.

In [6]:

```
from sklearn.preprocessing import Normalizer
from sklearn.cluster import KMeans
from sklearn.pipeline import make_pipeline
```

Step 3: Create an instance of `Normalizer` called `normalizer`.

In [7]:

```
normalizer = Normalizer()
normalizer
```

Out[7]:

```
Normalizer
Normalizer()
```

Step 4: Create an instance of `KMeans` called `kmeans` with `14` clusters.

In [8]:

```
kmeans = KMeans(n_clusters=14, random_state=0, n_init="auto")
kmeans
```

Out[8]:

```
KMeans
KMeans(n_clusters=14, n_init='auto', random_state=0)
```

Step 5: Using `make_pipeline()`, create a pipeline called `pipeline` that chains `normalizer` and `kmeans`.

In [12]:

```
model = make_pipeline(normalizer, kmeans)
```

Step 6: Fit the pipeline to the `movements` array.

In [13]:

```
model.fit(movements)
```

```
C:\Anaconda\envs\python-dscourse\lib\site-packages\sklearn\cluster\_kmean
s.py:1436: UserWarning: KMeans is known to have a memory leak on Windows w
ith MKL, when there are less chunks than available threads. You can avoid
it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[13]:

```
Pipeline
Normalizer
KMeans
```

So which company have stock prices that tend to change in the same way? Now inspect the cluster labels from your clustering to find out.

In [17]:

```
labels = model.predict(movements)
labels
```

Out[17]:

```
array([ 4,  9, 13,  0,  0, 12,  1,  6,  0,  7,  1,  0,  1,  0,  3,  6,  0,
        4, 12,  1,  0,  6,  3,  0,  5,  2, 12,  7, 10,  8,  0,  0,  0,  0,
        6,  1,  8,  1, 10,  2,  7, 11,  1,  1,  1,  6,  1,  0,  6,  1,  5,
        5,  1,  1,  2, 12,  2,  1,  0, 13])
```

Step 8: Align the cluster labels with the list of company names `companies` by creating a DataFrame `df` with `labels` and `companies` as columns.

In [18]:

```
df = pd.DataFrame({"labels": labels, "companies": companies})
df
```

Out[18]:

	labels	companies
0	4	Apple
1	9	AIG
2	13	Amazon
3	0	American express
4	0	Boeing
5	12	Bank of America
6	1	British American Tobacco
7	6	Canon
8	0	Caterpillar
9	7	Colgate-Palmolive
10	1	ConocoPhillips
11	0	Cisco
12	1	Chevron
13	0	DuPont de Nemours
14	3	Dell
15	6	Ford
16	0	General Electrics
17	4	Google/Alphabet
18	12	Goldman Sachs
19	1	GlaxoSmithKline
20	0	Home Depot
21	6	Honda
22	3	HP
23	0	IBM
24	5	Intel
25	2	Johnson & Johnson
26	12	JPMorgan Chase
27	7	Kimberly-Clark
28	10	Coca Cola
29	8	Lookheed Martin
30	0	MasterCard
31	0	McDonalds
32	0	3M
33	0	Microsoft
34	6	Mitsubishi
35	1	Navistar
36	8	Northrop Grumman

	labels	companies
37	1	Novartis
38	10	Pepsi
39	2	Pfizer
40	7	Procter Gamble
41	11	Philip Morris
42	1	Royal Dutch Shell
43	1	SAP
44	1	Schlumberger
45	6	Sony
46	1	Sanofi-Aventis
47	0	Symantec
48	6	Toyota
49	1	Total
50	5	Taiwan Semiconductor Manufacturing
51	5	Texas instruments
52	1	Unilever
53	1	Valero Energy
54	2	Walgreen
55	12	Wells Fargo
56	2	Wal-Mart
57	1	Exxon
58	0	Xerox
59	13	Yahoo

Step 9: Now display the DataFrame, sorted by cluster label. To do this, use the `.sort_values()` method of `df` to sort the DataFrame by the `'labels'` column.

In [21]:

```
df.sort_values('labels')
```

Out[21]:

	labels	companies
23	0	IBM
33	0	Microsoft
32	0	3M
31	0	McDonalds
30	0	MasterCard
58	0	Xerox
20	0	Home Depot
16	0	General Electrics
13	0	DuPont de Nemours
11	0	Cisco
47	0	Symantec
8	0	Caterpillar
3	0	American express
4	0	Boeing
57	1	Exxon
35	1	Navistar
49	1	Total
46	1	Sanofi-Aventis
10	1	ConocoPhillips
6	1	British American Tobacco
19	1	GlaxoSmithKline
53	1	Valero Energy
42	1	Royal Dutch Shell
43	1	SAP
44	1	Schlumberger
12	1	Chevron
52	1	Unilever
37	1	Novartis
54	2	Walgreen
56	2	Wal-Mart
39	2	Pfizer
25	2	Johnson & Johnson
14	3	Dell
22	3	HP
0	4	Apple
17	4	Google/Alphabet
51	5	Texas instruments

	labels	companies
50	5	Taiwan Semiconductor Manufacturing
24	5	Intel
21	6	Honda
48	6	Toyota
7	6	Canon
15	6	Ford
45	6	Sony
34	6	Mitsubishi
27	7	Kimberly-Clark
40	7	Procter Gamble
9	7	Colgate-Palmolive
29	8	Lookheed Martin
36	8	Northrop Grumman
1	9	AIG
38	10	Pepsi
28	10	Coca Cola
41	11	Philip Morris
26	12	JPMorgan Chase
18	12	Goldman Sachs
5	12	Bank of America
55	12	Wells Fargo
2	13	Amazon
59	13	Yahoo

In [ ]: