Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Case Study - Clustering Stocks using k-Means

Name: Ooi Caaron

IC Number:990701-07-5837

Date :29/7/23

Introduction :

Conclusion :

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 [6]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import normalize
from sklearn.manifold import TSNE

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

Royal Dutch Shell	0.489997	-0.190003	-0.229999	0.070000	0.350002	-0.049999	0.169998	0.150002
SAP	-0.119999	-0.160000	0.579998	0.529999	0.530003	0.239997	-0.030003	-0.209999
Schlumberger	0.720002	0.310005	1.630005	1.020004	1.460000	-0.650001	0.389999	1.139999
Sony	0.500000	0.160000	-0.029999	0.059999	0.369999	-0.119999	0.100001	0.380001
Sanofi-Aventis	0.480000	-0.110001	0.160000	0.229999	0.219998	0.010003	0.250000	0.049999
Symantec	0.359999	-0.100001	0.140002	-0.149999	0.120001	-0.099998	0.139999	-0.049999
Toyota	0.330002	0.599999	0.489998	0.680000	1.260002	0.070000	0.070000	0.200004
Total	0.059997	0.010002	0.320000	0.169999	0.630005	-0.190002	0.269997	-0.160004
Taiwan Semiconductor Manufacturing	0.090000	-0.070000	-0.070000	-0.300000	-0.040000	-0.110000	-0.180000	0.270000
Texas instruments	-0.190001	-0.090000	-0.230000	0.150000	0.740000	-0.340000	-1.110000	0.010000

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

In [3]:

```
stocks_df.head()
```

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	0.069995
Amazon	-2.350006	1.260009	-2.350006	-2.009995	2.960006	-2.309997	-1.640007	1.209995
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	0.549995

5 rows × 963 columns

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

In [9]:

```
movements = stocks_df.values
companies = stocks_df.index
```

Step 4: Make the necessary imports:

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

In [8]:

```
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 [12]:

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

Out[12]:

```
Normalizer()
```

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

In [11]:

```
kmeans = KMeans(n_clusters=14, random_state=0)

kmeans
```

Out[11]:

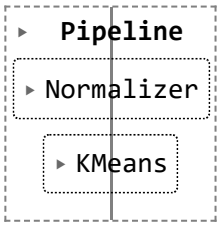
```
KMeans(n_clusters=14, random_state=0)
```

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

In [13]:

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

Out[13]:



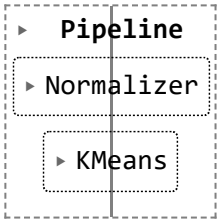
Step 6: Fit the pipeline to the `movements` array.

In [16]:

```
pipeline.fit(movements)
```

```
C:\Users\User\anaconda3\envs\python-dscourse\Lib\site-packages\sklearn\clu
ster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to sup
press the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\User\anaconda3\envs\python-dscourse\Lib\site-packages\sklearn\clu
ster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with 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[16]:



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.

Step 7: Predict the labels for `movements` using function provided by pipeline

In [17]:

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

Out[17]:

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

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 [24]:

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

df
```

Out[24]:

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

	labels	companies
37	1	Novartis
38	8	Pepsi
39	6	Pfizer
40	3	Procter Gamble
41	3	Philip Morris
42	1	Royal Dutch Shell
43	1	SAP
44	6	Schlumberger
45	5	Sony
46	1	Sanofi-Aventis
47	1	Symantec
48	5	Toyota
49	1	Total
50	0	Taiwan Semiconductor Manufacturing
51	0	Texas instruments
52	1	Unilever
53	6	Valero Energy
54	12	Walgreen
55	2	Wells Fargo
56	13	Wal-Mart
57	6	Exxon
58	2	Xerox
59	4	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 [25]:

```
df_sorted = df.sort_values(by='labels')

df_sorted
```

Out[25]:

	labels	companies
51	0	Texas instruments
24	0	Intel
50	0	Taiwan Semiconductor Manufacturing
35	1	Navistar
42	1	Royal Dutch Shell
6	1	British American Tobacco
49	1	Total
46	1	Sanofi-Aventis
31	1	McDonalds
37	1	Novartis
47	1	Symantec
20	1	Home Depot
43	1	SAP
19	1	GlaxoSmithKline
52	1	Unilever
26	2	JPMorgan Chase
1	2	AIG
58	2	Xerox
3	2	American express
16	2	General Electrics
55	2	Wells Fargo
5	2	Bank of America
18	2	Goldman Sachs
15	2	Ford
27	3	Kimberly-Clark
9	3	Colgate-Palmolive
40	3	Procter Gamble
41	3	Philip Morris
25	3	Johnson & Johnson
59	4	Yahoo
2	4	Amazon
21	5	Honda
48	5	Toyota
34	5	Mitsubishi
7	5	Canon
45	5	Sony
44	6	Schlumberger

	labels	companies
32	6	3M
53	6	Valero Energy
13	6	DuPont de Nemours
12	6	Chevron
10	6	ConocoPhillips
39	6	Pfizer
8	6	Caterpillar
57	6	Exxon
23	6	IBM
29	7	Lookheed Martin
36	7	Northrop Grumman
4	7	Boeing
28	8	Coca Cola
38	8	Pepsi
30	9	MasterCard
17	10	Google/Alphabet
0	10	Apple
33	11	Microsoft
22	11	HP
14	11	Dell
11	11	Cisco
54	12	Walgreen
56	13	Wal-Mart

In [ ]: