Forward School

Program Code: J620-002-4:2020

Program Name: FRONT-END SOFTWARE DEVELOPMENT

Title : Case Study - Clustering Stocks using k-Means

Name: Chong Mun Chen

IC Number: 960327-07-5097

Date : 26/7/2023

Introduction : Practising on this case study using k-means clustering method.

Conclusion : Succeeded in chaining the Normalizer with k-means clustering method in the Pipeline, and achieving the desired clusters.

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.

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

In [13]:
```
import pandas as pd

import warnings

warnings.filterwarnings('ignore')

fn = '../data_samples2/company-stock-movements-2010-2015-incl.csv'
stocks_df = pd.read_csv(fn, index_col=0)
```

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

In [5]:
```
stocks_df.head()
```

Out[5]:

	2010-01-04	2010-01-05	2010-01-06	2010-01-07	2010-01-08	2010-01-11	2010-01-12	2010-01
Apple	0.580000	-0.220005	-3.409998	-1.170000	1.680011	-2.689994	-1.469994	2.77999
AIG	-0.640002	-0.650000	-0.210001	-0.420000	0.710001	-0.200001	-1.130001	0.06999
Amazon	-2.350006	1.260009	-2.350006	-2.009995	2.960006	-2.309997	-1.640007	1.20999
American express	0.109997	0.000000	0.260002	0.720002	0.190003	-0.270001	0.750000	0.30000
Boeing	0.459999	1.770000	1.549999	2.690003	0.059997	-1.080002	0.360000	0.54999

5 rows × 963 columns

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

In [36]:
```
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 [6]:
```
from sklearn.preprocessing import Normalizer
from sklearn.pipeline import make_pipeline
from sklearn.cluster import KMeans
```

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

In [7]:
```
normalizer = Normalizer()
```

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

In [8]:
```
kmeans = KMeans(n_clusters = 14)
```

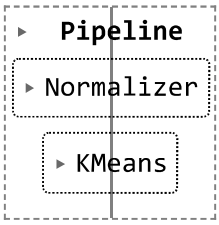
Step 5: Using `make_pipeline()`, create a pipeline called `pipeline` that chains `normalizer` and `kmeans`.

In [11]:
```
pipeline = make_pipeline(normalizer, kmeans)
```

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

In [14]:
```
pipeline.fit(movements)
```

Out[14]:



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 [25]:
```
labels = pipeline.predict(movements)
labels
```

Out[25]:
```
array([ 7,  2, 12,  2,  8,  2,  1,  4,  3,  6,  5,  7,  5,  3,  7,  2,
2,
       12,  2,  1,  4,  4,  7,  5, 10,  6,  2,  6,  0,  8,  4,  5,  3,
7,
        4,  3,  8,  1,  0,  1,  6,  6,  5,  1,  5,  4,  1, 10,  4,  1, 1
0,
       10,  1,  5,  9,  2, 13,  5,  3, 11])
```

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 [37]:
```
companies_df = pd.DataFrame({'labels':labels, 'companies':companies})
```

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 [38]:
```
companies_df.sort_values('labels')
```

Out[38]:

	labels	companies
38	0	Pepsi
28	0	Coca Cola
19	1	GlaxoSmithKline
37	1	Novartis
49	1	Total
46	1	Sanofi-Aventis
6	1	British American Tobacco
39	1	Pfizer
52	1	Unilever
43	1	SAP
16	2	General Electrics
15	2	Ford
26	2	JPMorgan Chase
5	2	Bank of America
55	2	Wells Fargo
3	2	American express
1	2	AIG
18	2	Goldman Sachs
58	3	Xerox
35	3	Navistar
32	3	3M
13	3	DuPont de Nemours
8	3	Caterpillar
21	4	Honda
30	4	MasterCard
34	4	Mitsubishi
7	4	Canon
45	4	Sony
20	4	Home Depot
48	4	Toyota
42	5	Royal Dutch Shell
44	5	Schlumberger
23	5	IBM
53	5	Valero Energy
12	5	Chevron

	labels	companies
10	5	ConocoPhillips
57	5	Exxon
31	5	McDonalds
27	6	Kimberly-Clark
25	6	Johnson & Johnson
40	6	Procter Gamble
41	6	Philip Morris
9	6	Colgate-Palmolive
33	7	Microsoft
22	7	HP
14	7	Dell
11	7	Cisco
0	7	Apple
29	8	Lookheed Martin
36	8	Northrop Grumman
4	8	Boeing
54	9	Walgreen
50	10	Taiwan Semiconductor Manufacturing
51	10	Texas instruments
24	10	Intel
47	10	Symantec
59	11	Yahoo
17	12	Google/Alphabet
2	12	Amazon
56	13	Wal-Mart