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 [1]:
```
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 [2]:
```
stocks_df.head()
```

Out[2]:

	2010-01-04	2010-01-05	2010-01-06	2010-01-07	2010-01-08	2010-01-11	2010-01-12	2010-0 1
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 [3]:
```
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 [4]:

```
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 [5]:

```
normalizer = Normalizer()
```

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

In [6]:

```
kmeans = KMeans(n_clusters = 14)
```

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

In [7]:

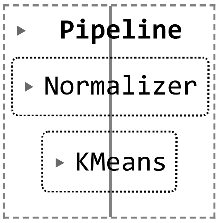
```
pipeline = make_pipeline(normalizer, kmeans)
```

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

In [8]:

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

Out[8]:



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 [9]:

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

Out[9]:

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

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 [10]:

```
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 [11]:

```
companies_df.sort_values('labels')
```

Out[11]:

	labels	companies
41	0	Philip Morris
6	0	British American Tobacco
18	1	Goldman Sachs
1	1	AIG
5	1	Bank of America
55	1	Wells Fargo
26	1	JPMorgan Chase
16	1	General Electrics
32	2	3M
44	2	Schlumberger
8	2	Caterpillar
50	2	Taiwan Semiconductor Manufacturing
35	2	Navistar
51	2	Texas instruments
13	2	DuPont de Nemours
53	2	Valero Energy
29	3	Lookheed Martin
36	3	Northrop Grumman
54	3	Walgreen
4	3	Boeing
40	4	Procter Gamble
56	4	Wal-Mart
27	4	Kimberly-Clark
25	4	Johnson & Johnson
9	4	Colgate-Palmolive
23	5	IBM
52	5	Unilever
47	5	Symantec
49	5	Total
43	5	SAP
42	5	Royal Dutch Shell
10	5	ConocoPhillips
57	5	Exxon
19	5	GlaxoSmithKline
37	5	Novartis

	labels	companies
46	5	Sanofi-Aventis
12	5	Chevron
59	6	Yahoo
2	6	Amazon
33	7	Microsoft
24	7	Intel
22	7	HP
11	7	Cisco
14	7	Dell
48	8	Toyota
58	8	Xerox
7	8	Canon
45	8	Sony
34	8	Mitsubishi
21	8	Honda
15	8	Ford
3	9	American express
30	9	MasterCard
20	9	Home Depot
38	10	Pepsi
28	10	Coca Cola
31	11	McDonalds
39	12	Pfizer
17	13	Google/Alphabet
0	13	Apple