SHARE PRICE PREDICTION

WITH PRINCIPAL COMPONENT ANALYSIS

Avijit Biswas

Table of Contents

CONTENTS

INTRODUCTION

In the dynamic landscape of financial markets, understanding the intricate relationships between stock prices is crucial for investors and financial analysts. The interconnectedness of companies often extends beyond market trends, incorporating factors such as loans, partnerships, and economic conditions. In our project, we delve into the realm of share price prediction using Principal Component Analysis (PCA), focusing on five prominent entities: TATA Motors, Bharat Electronics Limited (BEL), NMDC, ONGC, and HDFC.

Our analysis centers on the hypothesis that the financial health and stock performance of TATA Motors, BEL, NMDC, and ONGC are intricately linked to their association with HDFC Bank through loans. Specifically, we explore how fluctuations in the share prices of HDFC, a significant financial institution, reverberate across the stock values of these four companies.

To conduct this analysis, we have gathered historical data encompassing open prices, close prices, and trading volumes of the aforementioned companies. By employing Principal Component Analysis, a powerful statistical method, we aim to unveil the underlying patterns and dependencies among these stocks. PCA allows us to extract essential features from the dataset, providing insights into the shared dynamics and latent factors influencing the stock prices of the selected companies.

This project is not only an exploration of the statistical intricacies of the stock market but also a demonstration of the ripple effect that financial decisions, such as taking loans from HDFC, can have on the performance of associated companies. As we unfold the results of our analysis, we anticipate uncovering valuable information that can guide investors, analysts, and decision-makers in navigating the complex landscape of the financial markets.

HOW TO DO PCA?

F	Ex 1	Ex 2	Ex 3	Ex 4
X_1	4	8	13	7
X_2	11	4	5	14

01 Calculate Mean

X_1 = (4+8+13+7) = 8 X_2 = (11+4+5+14) = 8.5

X_2 = (11 + 4 + 5 + 14) = 8.5

02 Calculation of Covariance Matrix

$$S = \begin{vmatrix} Cov(X_1,X_1) & Cov(X_1,X_2) \\ Cov(X_2,X_1) & Cov(X_2,X_2) \end{vmatrix}$$

$$Cov(X_1,X_2) = \frac{1}{N-1} \times \sum_{k=1}^{N} \left(X_{1k} - \bar{X}_1\right) \times \left(X_{2k} - \bar{X}_2\right)$$

$$= \tfrac{1}{3}((4-8)(11-8.5) + (8-8)(4-8.5) + (13-8)(5-8.5) + (7-8)(14-8.5)$$

$$= -11$$

03 Eigen Values of the covariance matrix

$$
\begin{aligned}
0 &= \det(S - \lambda I) \\
&= \begin{vmatrix} 14 - \lambda & -11 \\ -11 & 23 - \lambda \end{vmatrix} \\
&= (14 - \lambda)(23 - \lambda) - (-11) \times (-11) \\
&= \lambda^2 - 37\lambda + 201
\end{aligned}
$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

04 Calculation of Covariance Matrix

$$S = \begin{vmatrix} Cov(X_1, X_1) & Cov(X_1, X_2) \\ Cov(X_2, X_1) & Cov(X_2, X_2) \end{vmatrix}$$

$$Cov(X_1, X_2) = \frac{1}{N-1} \times \sum_{k=1}^{N} \left(X_{1k} - \bar{X}_1\right) \times \left(X_{2k} - \bar{X}_2\right)$$

$$
\begin{aligned}
&= \tfrac{1}{3}((4-8)(11-8.5) + (8-8)(4-8.5) \\
&\quad + (13-8)(5-8.5) + (7-8)(14-8.5) \\
&= -11
\end{aligned}
$$

05 Computation of the eigen vectors

$$U_1 = \begin{bmatrix} 11 \\ 14 - \lambda_1 \end{bmatrix}.$$

- To find a unit eigenvector, we compute the length of U_1 which is given by,

$$\|U_1\| = \sqrt{11^2 + (14 - \lambda_1)^2}$$
$$= \sqrt{11^2 + (14 - 30.3849)^2}$$
$$= 19.7348$$

$$e_1 = \begin{bmatrix} 11/\|U_1\| \\ (14 - \lambda_1)/\|U_1\| \end{bmatrix}$$
$$= \begin{bmatrix} 11/19.7348 \\ (14 - 30.3849)/19.7348 \end{bmatrix}$$
$$= \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

$$\overline{X_1} = 8$$

$$\overline{X_2} = 8.5$$

$$S = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

$$\lambda_1 = 30.3849$$

$$\lambda_2 = 6.615$$

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

06 Computation of first principal components

$$e_1^T \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix}$$

$$\bar{e}_1^T \begin{bmatrix} X_{1k} - \bar{X}_1 \\ X_{2k} - \bar{X}_2 \end{bmatrix} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} X_{11} - \bar{X}_1 \\ X_{21} - \bar{X}_2 \end{bmatrix}$$

$$= 0.5574(X_{11} - \bar{X}_1) - 0.8303(X_{21} - \bar{X}_2)$$

$$= 0.5574(4 - 8) - 0.8303(11 - 8,5)$$

$$= -4.30535$$

In [2]:
```
import pandas as pd
import seaborn as sns
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn.preprocessing import StandardScaler
```

In [3]:
```
df_tm_d=pd.read_csv('df_tm_d.csv')
df_bel_d=pd.read_csv('df_bel_d.csv')
df_nmdc_d=pd.read_csv('df_nmdc_d.csv')
df_hdfc_d=pd.read_csv('df_hdfc_d.csv')
df_ongc_d=pd.read_csv('df_ongc_d.csv')
```

In [4]:
```
df_tm_d
```

Out[4]:

	Open	High	Low	Close	Adj Close	Volume
0	344.801666	346.829895	332.632202	336.837097	334.226288	9215459
1	338.271698	338.370636	326.003296	329.911377	327.354248	6661518
2	328.476746	334.709900	326.596924	330.505005	327.943268	5971621
3	335.402466	337.183380	321.402649	327.833649	325.292664	5795207
4	325.508606	334.066803	324.766541	331.494385	328.924988	4927964
...	...	...	...	...	...	...
2504	645.000000	649.900024	643.299988	645.000000	645.000000	6588073
2505	648.000000	649.049988	641.900024	642.549988	642.549988	6551864
2506	644.000000	653.650024	642.700012	649.349976	649.349976	10155047
2507	649.349976	652.950012	644.000000	651.049988	651.049988	6425792
2508	653.000000	654.000000	649.299988	653.250000	653.250000	3279829

2509 rows × 6 columns

In [5]:
```
df_bel_d
```

Out[5]:

	Open	High	Low	Close	Adj Close	Volume
0	11.898989	12.212121	11.898989	12.101515	9.819762	2590929
1	11.969696	12.116161	11.888888	11.931818	9.682062	2565585
2	12.006060	12.070707	11.819191	11.915151	9.668537	6399855
3	11.919191	12.111111	11.677777	11.838888	9.606653	7690221
4	11.645959	11.822222	11.627272	11.650505	9.453794	154836
...	...	...	...	...	...	...
2504	137.550003	138.949997	136.699997	138.050003	138.050003	8587321
2505	137.750000	141.149994	137.699997	140.399994	140.399994	9755483
2506	139.949997	139.949997	137.600006	137.949997	137.949997	5481774
2507	137.500000	139.100006	137.050003	138.800003	138.800003	5781314
2508	139.149994	144.199997	139.050003	143.399994	143.399994	20245049

2509 rows × 6 columns

In [6]:
```
df_nmdc_d
```

Out[6]:

	Open	High	Low	Close	Adj Close	Volume
0	118.400002	126.050003	117.550003	124.949997	63.994610	4671600
1	126.650002	126.650002	123.150002	123.650002	63.328789	2950116
2	124.449997	125.900002	122.800003	124.750000	63.892162	2435500
3	126.500000	127.000000	123.000000	123.250000	63.123920	1654940
4	123.250000	125.250000	122.750000	123.900002	63.456825	2582314
...	...	...	...	...	...	...
2504	160.500000	164.550003	160.399994	163.050003	163.050003	16025618
2505	164.300003	165.000000	161.850006	162.399994	162.399994	6344587
2506	162.600006	164.899994	161.699997	163.050003	163.050003	6653234
2507	162.449997	169.600006	161.449997	168.699997	168.699997	24358320
2508	172.949997	177.800003	170.100006	175.250000	175.250000	26009145

2509 rows × 6 columns

In [7]:
```
df_nmdc_d
```

Out[7]:

	Open	High	Low	Close	Adj Close	Volume
0	118.400002	126.050003	117.550003	124.949997	63.994610	4671600
1	126.650002	126.650002	123.150002	123.650002	63.328789	2950116
2	124.449997	125.900002	122.800003	124.750000	63.892162	2435500
3	126.500000	127.000000	123.000000	123.250000	63.123920	1654940
4	123.250000	125.250000	122.750000	123.900002	63.456825	2582314
...	...	...	...	...	...	...
2504	160.500000	164.550003	160.399994	163.050003	163.050003	16025618
2505	164.300003	165.000000	161.850006	162.399994	162.399994	6344587
2506	162.600006	164.899994	161.699997	163.050003	163.050003	6653234
2507	162.449997	169.600006	161.449997	168.699997	168.699997	24358320
2508	172.949997	177.800003	170.100006	175.250000	175.250000	26009145

2509 rows × 6 columns

In [8]:
```
df_hdfc_d
```

Out[8]:

	Open	High	Low	Close	Adj Close	Volume
0	316.500000	325.000000	312.774994	323.625000	300.909729	6345196
1	321.250000	323.000000	313.799988	316.975006	294.726501	8838268
2	316.225006	317.250000	310.500000	314.600006	292.518188	7751560
3	318.750000	324.450012	314.500000	321.399994	298.840851	6069608
4	319.000000	322.500000	316.799988	321.125000	298.585205	4216044
...	...	...	...	...	...	...
2504	1494.000000	1494.000000	1477.199951	1487.250000	1487.250000	20976424
2505	1483.949951	1493.449951	1481.000000	1491.550049	1491.550049	15699773
2506	1486.000000	1492.599976	1483.500000	1485.650024	1485.650024	11058370
2507	1480.099976	1496.000000	1480.050049	1491.500000	1491.500000	7007593
2508	1492.099976	1495.949951	1485.900024	1488.800049	1488.800049	10302957

2509 rows × 6 columns

In [9]:
```
df_ongc_d
```

Out[9]:

	Open	High	Low	Close	Adj Close	Volume
0	195.300003	195.466660	188.766663	192.866669	117.423248	5722174
1	195.199997	195.333328	185.033340	185.966660	113.222290	4587094
2	183.933334	189.733337	183.066666	188.466660	114.744339	4640854
3	192.666672	196.199997	186.666672	191.466660	116.570831	3781701
4	191.300003	192.333328	183.899994	186.466660	113.724670	5367991
...	...	...	...	...	...	...
2504	193.399994	198.000000	192.350006	193.199997	187.585144	13921916
2505	193.699997	196.550003	192.300003	195.350006	189.672668	10186886
2506	196.500000	196.699997	192.199997	192.699997	187.099670	5303833
2507	192.699997	196.500000	192.699997	195.899994	190.206665	6025889
2508	195.899994	197.300003	194.100006	195.800003	190.109589	11181086

2509 rows × 6 columns

In [10]:
```
df_tm_d=df_tm_d.rename(columns={'Close':'tm_Clo',
                                'Volume':'tm_Vol'})
```

In [11]:
```
df_bel_d=df_bel_d.rename(columns={'Close':'bel_Clo',
                                'Volume':'bel_Vol'})
df_nmdc_d=df_nmdc_d.rename(columns={'Close':'nmdc_Clo',
                                'Volume':'nmdc_Vol'})
df_hdfc_d=df_hdfc_d.rename(columns={'Close':'hdfc_Clo',
                                'Volume':'hdfc_Vol'})
df_ongc_d=df_ongc_d.rename(columns={'Close':'ongc_Clo',
                                'Volume':'ongc_Vol'})
```

In [12]:
```
result = pd.concat([df_tm_d["tm_Vol"],df_tm_d["tm_Clo"],df_bel_d["bel_Vol"],df_bel_
```

In [13]:
```
result
```

Out[13]:

	tm_Vol	tm_Clo	bel_Vol	bel_Clo	nmdc_Vol	nmdc_Clo	ongc_Vol	ongc_Clo
0	9215459	336.837097	2590929	12.101515	4671600	124.949997	5722174	192.866669
1	6661518	329.911377	2565585	11.931818	2950116	123.650002	4587094	185.966660
2	5971621	330.505005	6399855	11.915151	2435500	124.750000	4640854	188.466660
3	5795207	327.833649	7690221	11.838888	1654940	123.250000	3781701	191.466660
4	4927964	331.494385	154836	11.650505	2582314	123.900002	5367991	186.466660
...	...	...	...	...	...	...	...	...
2504	6588073	645.000000	8587321	138.050003	16025618	163.050003	13921916	193.199997
2505	6551864	642.549988	9755483	140.399994	6344587	162.399994	10186886	195.350006
2506	10155047	649.349976	5481774	137.949997	6653234	163.050003	5303833	192.699997
2507	6425792	651.049988	5781314	138.800003	24358320	168.699997	6025889	195.899994
2508	3279829	653.250000	20245049	143.399994	26009145	175.250000	11181086	195.800003

2509 rows × 10 columns

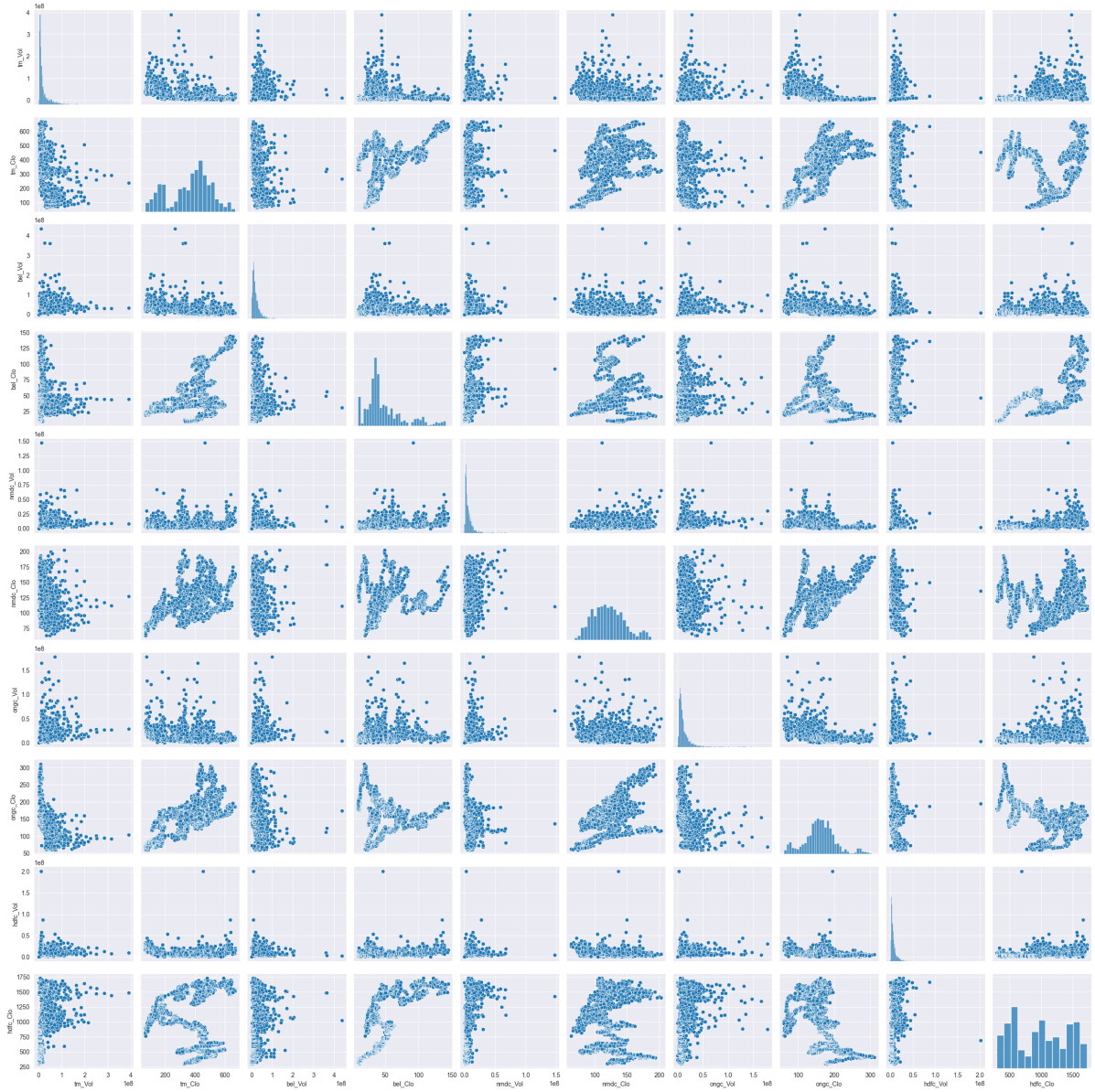
In [15]:
```
result.shape
```

Out[15]:
```
(2509, 10)
```

In [27]:
```
sns.pairplot(data = result)
```

```
C:\ProgramData\anaconda3\lib\site-packages\seaborn\axisgrid.py:118: UserWarning: T
he figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[27]:
```
<seaborn.axisgrid.PairGrid at 0x233917e1690>
```



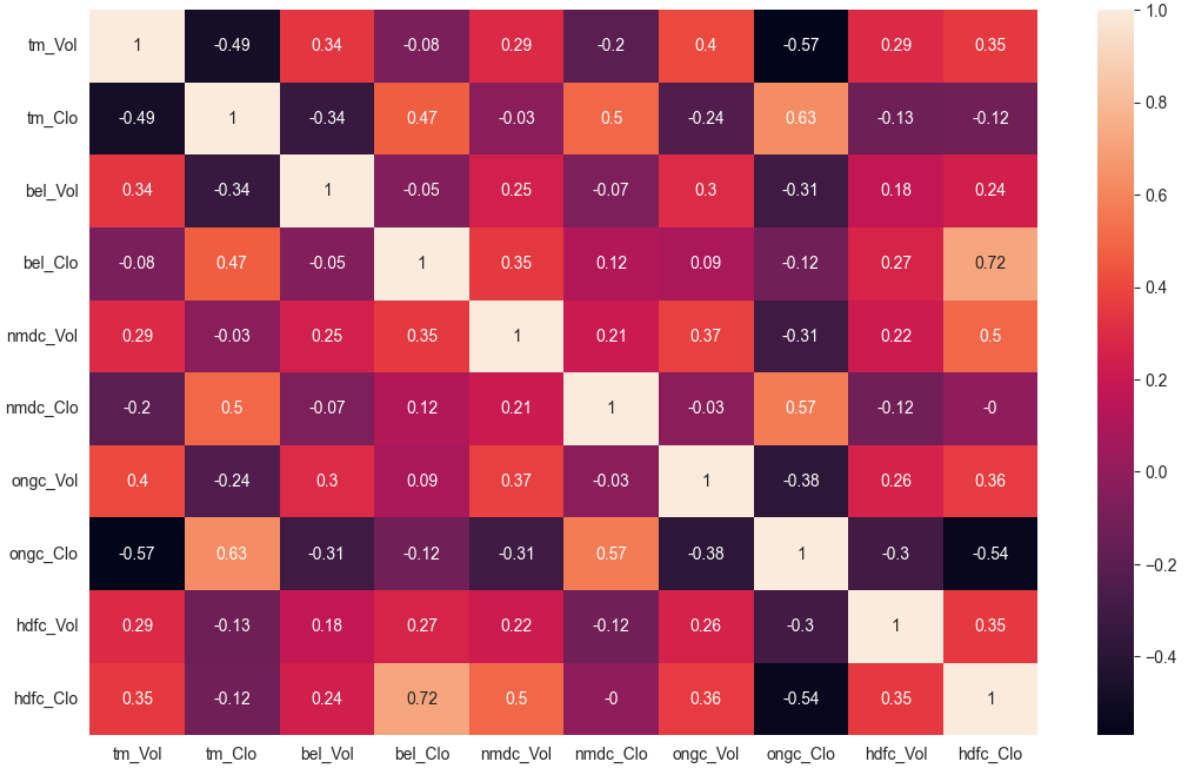
In [28]:
```
result_corr_matrix = result.corr().round(2) #[-1 to +1]
```

In [29]:
```
result_corr_matrix
```

Out[29]:

	tm_Vol	tm_Clo	bel_Vol	bel_Clo	nmdc_Vol	nmdc_Clo	ongc_Vol	ongc_Clo	hdfc_Vol
tm_Vol	1.00	-0.49	0.34	-0.08	0.29	-0.20	0.40	-0.57	0.29
tm_Clo	-0.49	1.00	-0.34	0.47	-0.03	0.50	-0.24	0.63	-0.13
bel_Vol	0.34	-0.34	1.00	-0.05	0.25	-0.07	0.30	-0.31	0.18
bel_Clo	-0.08	0.47	-0.05	1.00	0.35	0.12	0.09	-0.12	0.27
nmdc_Vol	0.29	-0.03	0.25	0.35	1.00	0.21	0.37	-0.31	0.22
nmdc_Clo	-0.20	0.50	-0.07	0.12	0.21	1.00	-0.03	0.57	-0.12
ongc_Vol	0.40	-0.24	0.30	0.09	0.37	-0.03	1.00	-0.38	0.26
ongc_Clo	-0.57	0.63	-0.31	-0.12	-0.31	0.57	-0.38	1.00	-0.30
hdfc_Vol	0.29	-0.13	0.18	0.27	0.22	-0.12	0.26	-0.30	1.00
hdfc_Clo	0.35	-0.12	0.24	0.72	0.50	-0.00	0.36	-0.54	0.35

In [30]:
```
plt.figure(figsize=(13,8))
sns.heatmap(data = result_corr_matrix,annot=True)
plt.show()
```



In [16]:
```
scaler=StandardScaler()
scaler.fit(result)
```

Out[16]:
```
▾ StandardScaler

StandardScaler()
```

In [17]:
```
scaled_data=scaler.transform(result)
```

In [18]:
```
scaled_data
```

Out[18]:
```
array([[-0.43608261, -0.20681054, -0.79138337, ...,  0.69758548,
        -0.08063298, -1.57242739],
       [-0.52109147, -0.25619882, -0.79241756, ...,  0.54463587,
         0.25532705, -1.58806476],
       [-0.54405494, -0.25196558, -0.63595598, ...,  0.60005232,
         0.10888505, -1.59364954],
       ...,
       [-0.40480809,  2.02176272, -0.67341928, ...,  0.69389094,
         0.55450234,  1.1600587 ],
       [-0.5289377 ,  2.03388575, -0.66119622, ...,  0.76482393,
         0.00862996,  1.17381484],
       [-0.63365222,  2.04957434, -0.07098769, ...,  0.76260747,
         0.45270481,  1.16746594]])
```

In [19]:
```
pca=PCA(n_components=2)
```

In [20]:
```
pca.fit(scaled_data)
```

Out[20]:
```
PCA
PCA(n_components=2)
```

In [21]:
```
x_pca=pca.transform(scaled_data)
```

In [22]:
```
scaled_data.shape
```

Out[22]:
```
(2509, 10)
```

In [23]:
```
x_pca.shape
```

Out[23]:
```
(2509, 2)
```

In [24]:
```
scaled_data
```

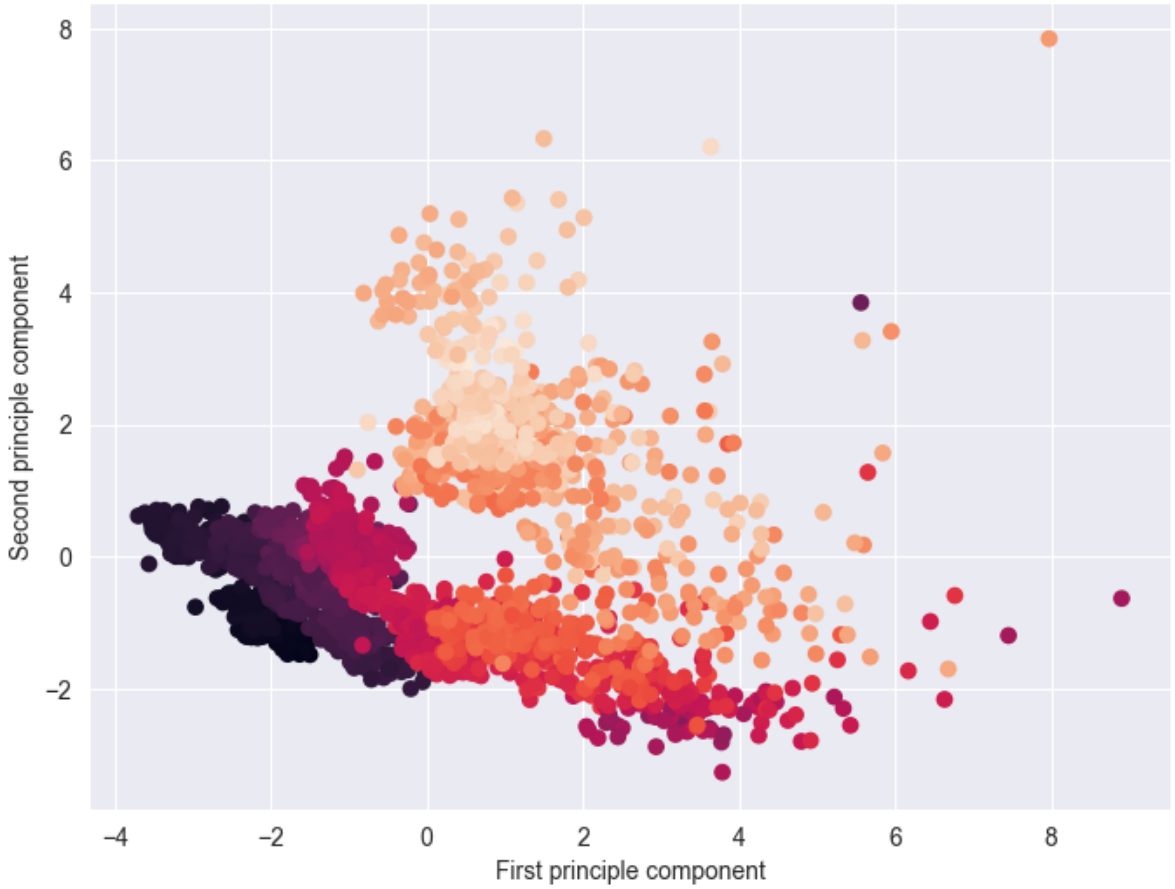
Out[24]:
```
array([[-0.43608261, -0.20681054, -0.79138337, ...,  0.69758548,
        -0.08063298, -1.57242739],
       [-0.52109147, -0.25619882, -0.79241756, ...,  0.54463587,
         0.25532705, -1.58806476],
       [-0.54405494, -0.25196558, -0.63595598, ...,  0.60005232,
         0.10888505, -1.59364954],
       ...,
       [-0.40480809,  2.02176272, -0.67341928, ...,  0.69389094,
         0.55450234,  1.1600587 ],
       [-0.5289377 ,  2.03388575, -0.66119622, ...,  0.76482393,
         0.00862996,  1.17381484],
       [-0.63365222,  2.04957434, -0.07098769, ...,  0.76260747,
         0.45270481,  1.16746594]])
```

In [25]:
```
x_pca
```

Out[25]:
```
array([[-1.67936898, -1.22545305],
       [-1.62618384, -1.32529124],
       [-1.68751344, -1.35348895],
       ...,
       [-0.80982778,  3.98984521],
       [-0.356676  ,  4.86814275],
       [ 0.03935053,  5.19287329]])
```

In [26]:
```
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=result['hdfc_Clo'])
plt.xlabel('First principle component')
plt.ylabel('Second principle component')
```

Out[26]:
```
Text(0, 0.5, 'Second principle component')
```


Second principle component
First principle component
8
6
4
2
0
−2
−4
−2
0
2
4
6
8

In [ ]: