MATH_628_FINAL_PROJECT

December 4, 2023

0.1 MATH 628 FINAL PROJECT

0.1.1 Chunlin Shi Noah Collins

[1]:
```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
data = pd.read_excel('data.xlsx')
data
```

[1]:
```
      PERMNO Names Date Ticker Symbol  \
0      11850 2021-12-31           XOM
1      11850 2022-01-03           XOM
2      11850 2022-01-04           XOM
3      11850 2022-01-05           XOM
4      11850 2022-01-06           XOM
…        …          …             …
4279   93002 2022-12-23          AVGO
4280   93002 2022-12-27          AVGO
4281   93002 2022-12-28          AVGO
4282   93002 2022-12-29          AVGO
4283   93002 2022-12-30          AVGO

      North American Industry Classification System  Price or Bid/Ask Average  \
0                                            324110                 61.189999
1                                            324110                 63.540001
2                                            324110                 65.930000
3                                            324110                 66.750000
4                                            324110                 68.320000
…                                              …                        …
4279                                         334413                552.429993
4280                                         334413                553.539978
4281                                         334413                544.890015
4282                                         334413                557.809998
4283                                         334413                559.130005
```

```
      Shares Outstanding  Returns without Dividends  \
0                4233567                   0.006580
1                4233567                   0.038405
2                4233567                   0.037614
3                4233567                   0.012437
4                4233567                   0.023521
…                    …                          …
4279              418000                  -0.001193
4280              418000                   0.002009
4281              418000                  -0.015627
4282              418000                   0.023711
4283              418000                   0.002366

      Value-Weighted Return-excl. dividends
0                                 -0.002426
1                                  0.006132
2                                 -0.002398
3                                 -0.021933
4                                  0.000103
…                                       …
4279                               0.005395
4280                              -0.003997
4281                              -0.012383
4282                               0.018135
4283                              -0.002500

[4284 rows x 8 columns]
```

[2]:
```
data_ret = data[['Ticker Symbol','Names Date','Returns without Dividends']]
data_ret
```

[2]:
```
     Ticker Symbol Names Date  Returns without Dividends
0              XOM 2021-12-31                   0.006580
1              XOM 2022-01-03                   0.038405
2              XOM 2022-01-04                   0.037614
3              XOM 2022-01-05                   0.012437
4              XOM 2022-01-06                   0.023521
…             …        …                          …
4279          AVGO 2022-12-23                  -0.001193
4280          AVGO 2022-12-27                   0.002009
4281          AVGO 2022-12-28                  -0.015627
4282          AVGO 2022-12-29                   0.023711
4283          AVGO 2022-12-30                   0.002366

[4284 rows x 3 columns]
```

[3]:
```
data_ret = data_ret.pivot_table(index='Names Date', columns='Ticker Symbol',␣
  ↪values='Returns without Dividends')
data_ret
```

[3]:
```
Ticker Symbol       AAL       AMD      AVGO       BAC       BRK       CVX  \
Names Date
2021-12-31    -0.006087 -0.008612  0.000496 -0.000898 -0.003884 -0.000681
2022-01-03     0.043987  0.044058 -0.003141  0.037986  0.007030  0.016276
2022-01-04     0.014400 -0.038738  0.011457  0.039194  0.025440  0.018196
2022-01-05    -0.017876 -0.057264 -0.041614 -0.016879  0.003916  0.006506
2022-01-06    -0.005889  0.000588 -0.009285  0.020136  0.011614  0.008509
…                   …         …         …         …         …         …
2022-12-23     0.011943  0.010335 -0.001193  0.002470  0.011400  0.030916
2022-12-27    -0.014162 -0.019374  0.002009  0.001848 -0.003093  0.012571
2022-12-28    -0.016760 -0.011064 -0.015627  0.007378 -0.005802 -0.014753
2022-12-29     0.030844  0.035960  0.023711  0.011291  0.018983  0.007572
2022-12-30     0.001575 -0.000771  0.002366 -0.000604 -0.000274  0.006561

Ticker Symbol       DAL       EOG      INTC       JPM       LUV      NVDA  \
Names Date
2021-12-31     0.001025 -0.003925 -0.004639 -0.000820  0.002809 -0.005915
2022-01-03     0.030962  0.026230  0.033204  0.021156  0.027077  0.024141
2022-01-04     0.007446  0.045963 -0.001316  0.037910  0.015000 -0.027589
2022-01-05    -0.007637 -0.018353  0.013737 -0.024132 -0.015002 -0.057562
2022-01-06    -0.004220  0.020513  0.002599  0.010624 -0.002273  0.020794
…                   …         …         …         …         …         …
2022-12-23     0.007290  0.034125  0.004621  0.004745  0.017767 -0.008671
2022-12-27    -0.007841  0.011255 -0.005749  0.003504 -0.059573 -0.071353
2022-12-28    -0.027660 -0.035433 -0.015420  0.005465 -0.051562 -0.006019
2022-12-29     0.023132  0.009655  0.026233  0.005738  0.036968  0.040396
2022-12-30     0.003972  0.006919  0.008394  0.006606  0.008688  0.000753

Ticker Symbol       SLB       UAL       WFC       XOM
Names Date
2021-12-31     0.004360 -0.007931 -0.002495  0.006580
2022-01-03     0.059098  0.039059  0.057316  0.038405
2022-01-04     0.048550  0.016707  0.039819  0.037614
2022-01-05     0.000000 -0.010162 -0.008720  0.012437
2022-01-06     0.023752 -0.000218  0.025626  0.023521
…                   …         …         …         …
2022-12-23     0.031135  0.002874  0.007375  0.026445
2022-12-27     0.009624 -0.004949  0.001464  0.013894
2022-12-28    -0.016822 -0.023822  0.001949 -0.016426
2022-12-29     0.005894  0.016895  0.005107  0.007566
2022-12-30     0.010395 -0.005802 -0.000968  0.010073

[252 rows x 16 columns]
```

[4]:
```
data_ret.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 252 entries, 2021-12-31 to 2022-12-30
Data columns (total 16 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   AAL     252 non-null    float64
 1   AMD     252 non-null    float64
 2   AVGO    252 non-null    float64
 3   BAC     252 non-null    float64
 4   BRK     252 non-null    float64
 5   CVX     252 non-null    float64
 6   DAL     252 non-null    float64
 7   EOG     252 non-null    float64
 8   INTC    252 non-null    float64
 9   JPM     252 non-null    float64
 10  LUV     252 non-null    float64
 11  NVDA    252 non-null    float64
 12  SLB     252 non-null    float64
 13  UAL     252 non-null    float64
 14  WFC     252 non-null    float64
 15  XOM     252 non-null    float64
dtypes: float64(16)
memory usage: 33.5 KB
```

0.2 From above, we can see that there is no null values for the return data

0.3 We then standardize the return data

[5]:
```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data_ret)
scaled_data = pd.DataFrame(scaled_data)
scaled_data.columns = data_ret.columns

scaled_data
```

[5]:
```
Ticker Symbol       AAL       AMD      AVGO       BAC       BRK       CVX  \
0              -0.150503 -0.160448  0.038864  0.003356 -0.293354 -0.124574
1               1.266021  1.213171 -0.116372  1.907877  0.485604  0.693871
2               0.429058 -0.946130  0.506723  1.967067  1.799625  0.786510
3              -0.483981 -1.429267 -1.758524 -0.779349  0.263372  0.222309
4              -0.144880  0.079468 -0.378600  1.033580  0.812832  0.319004
..                   …         …         …         …         …         …
247             0.359542  0.333677 -0.033247  0.168326  0.797544  1.400451
248            -0.378923 -0.441117  0.103450  0.137853 -0.236843  0.515021
249            -0.452407 -0.224393 -0.649296  0.408711 -0.430191 -0.803724
250             0.894238  1.001959  1.029743  0.600364  1.338716  0.273746
```

```
251            0.066252  0.044025  0.118693  0.017790 -0.035659  0.224983

Ticker Symbol       DAL       EOG      INTC       JPM       LUV      NVDA  \
0              0.044680 -0.205590 -0.094290 -0.017764  0.147358 -0.098446
1              1.082119  0.861953  1.481022  1.148928  1.176709  0.659338
2              0.267204  1.560554  0.044043  2.038400  0.664441 -0.644901
3             -0.255488 -0.716404  0.670665 -1.255357 -0.608103 -1.400616
4             -0.137078  0.659554  0.206991  0.589810 -0.068206  0.574961
..                  …         …         …         …         …         …
247            0.261814  1.141461  0.291160  0.277698  0.781784 -0.167926
248           -0.262534  0.331802 -0.140527  0.211809 -2.498588 -1.748327
249           -0.949329 -1.321064 -0.543107  0.315934 -2.158770 -0.101080
250            0.810786  0.275149  1.190850  0.330386  1.596218  1.069183
251            0.146815  0.178303  0.448225  0.376474  0.396707  0.069678

Ticker Symbol       SLB       UAL       WFC       XOM
0              0.049494 -0.231060 -0.098832  0.180330
1              1.798101  1.144038  2.691561  1.625934
2              1.461129  0.489938  1.875261  1.590008
3             -0.089769 -0.296348 -0.389281  0.446395
4              0.668986 -0.005357  1.213130  0.949832
..                  …         …         …         …
247            0.904807  0.085125  0.361615  1.082671
248            0.217678 -0.143796  0.085868  0.512559
249           -0.627154 -0.696081  0.108500 -0.864694
250            0.098498  0.495428  0.255825  0.225116
251            0.242294 -0.168742 -0.027595  0.339009

[252 rows x 16 columns]
```

[ ]:

0.4 Now we can do PCA analysis

[6]:
```
pca = PCA(n_components=3)
pca.fit(scaled_data)
```

[6]:
```
PCA(n_components=3)
```

0.5 We extract the second and the third eigenvector from PCA

[7]:
```
eigenvectors = pca.components_
second_eigenvector = eigenvectors[1]
third_eigenvector = eigenvectors[2]
```

[8]:
```
signs = pd.DataFrame({'Stock': scaled_data.columns,
                      'Second_Eigenvector': second_eigenvector,
                      'Third_Eigenvector': third_eigenvector})
```

[9]:
```
signs
```

[9]:

	Stock	Second_Eigenvector	Third_Eigenvector
0	AAL	0.218933	-0.210610
1	AMD	0.073633	0.412608
2	AVGO	0.069199	0.376026
3	BAC	0.000662	-0.165386
4	BRK	-0.064327	-0.001756
5	CVX	-0.451722	-0.026942
6	DAL	0.203492	-0.259594
7	EOG	-0.450143	-0.049374
8	INTC	0.034908	0.417784
9	JPM	0.025836	-0.158595
10	LUV	0.174883	-0.220278
11	NVDA	0.098801	0.420549
12	SLB	-0.423821	-0.103848
13	UAL	0.218353	-0.266802
14	WFC	0.032731	-0.194900
15	XOM	-0.468001	-0.032569

0.6 We extract industry code from the original table

[10]:
```
industry_data = data[['Ticker Symbol', 'North American Industry Classification␣
↪System']].drop_duplicates()
industry_data = industry_data.rename(columns={'Ticker Symbol': 'Stock'})
industry_data = industry_data.sort_values(by='North American Industry␣
↪Classification System')
industry_data = industry_data.reset_index(drop=True)
industry_data
```

[10]:

	Stock	North American Industry Classification System
0	EOG	211120
1	SLB	213112
2	XOM	324110
3	INTC	334413
4	AMD	334413
5	NVDA	334413
6	AVGO	334413
7	CVX	447190
8	AAL	481111
9	LUV	481111
10	UAL	481111

```
11   DAL                                            481111
12   WFC                                            522110
13   JPM                                            522110
14   BAC                                            522110
15   BRK                                            524126
```

[11]:
```python
signs_with_industry = signs.merge(industry_data, on='Stock')
signs_with_industry = signs_with_industry.sort_values('North American Industry Classification System')
signs_with_industry
```

[11]:
```
   Stock  Second_Eigenvector  Third_Eigenvector  \
7    EOG           -0.450143          -0.049374
12   SLB           -0.423821          -0.103848
15   XOM           -0.468001          -0.032569
1    AMD            0.073633           0.412608
2   AVGO            0.069199           0.376026
8   INTC            0.034908           0.417784
11  NVDA            0.098801           0.420549
5    CVX           -0.451722          -0.026942
0    AAL            0.218933          -0.210610
6    DAL            0.203492          -0.259594
10   LUV            0.174883          -0.220278
13   UAL            0.218353          -0.266802
3    BAC            0.000662          -0.165386
9    JPM            0.025836          -0.158595
14   WFC            0.032731          -0.194900
4    BRK           -0.064327          -0.001756

    North American Industry Classification System
7                                          211120
12                                         213112
15                                         324110
1                                          334413
2                                          334413
8                                          334413
11                                         334413
5                                          447190
0                                          481111
6                                          481111
10                                         481111
13                                         481111
3                                          522110
9                                          522110
14                                         522110
4                                          524126
```

From table above, we can see that within the same industry group, the signs of second eigenvector and the third eigenvector are the same

[12]:
```
correlation_matrix = scaled_data.corr()
correlation_matrix
```

[12]:

Ticker Symbol	AAL	AMD	AVGO	BAC	BRK	CVX
AAL	1.000000	0.606493	0.594144	0.550496	0.529382	0.157999
AMD	0.606493	1.000000	0.780958	0.559184	0.588605	0.303841
AVGO	0.594144	0.780958	1.000000	0.554185	0.613924	0.296872
BAC	0.550496	0.559184	0.554185	1.000000	0.707732	0.365815
BRK	0.529382	0.588605	0.613924	0.707732	1.000000	0.469095
CVX	0.157999	0.303841	0.296872	0.365815	0.469095	1.000000
DAL	0.924746	0.576782	0.593796	0.595608	0.568930	0.190125
EOG	0.132858	0.264088	0.273659	0.344822	0.412894	0.815066
INTC	0.514597	0.741271	0.750160	0.507697	0.596912	0.296874
JPM	0.548905	0.519383	0.564049	0.895891	0.709166	0.303311
LUV	0.842430	0.552735	0.573072	0.537038	0.519312	0.200572
NVDA	0.622287	0.887174	0.824741	0.547887	0.578304	0.273685
SLB	0.185075	0.248181	0.261785	0.338610	0.399552	0.774093
UAL	0.928827	0.554975	0.552298	0.548971	0.520238	0.139055
WFC	0.576018	0.527571	0.531020	0.865869	0.698938	0.307397
XOM	0.125324	0.259137	0.265218	0.315299	0.431941	0.873425

Ticker Symbol	DAL	EOG	INTC	JPM	LUV	NVDA
AAL	0.924746	0.132858	0.514597	0.548905	0.842430	0.622287
AMD	0.576782	0.264088	0.741271	0.519383	0.552735	0.887174
AVGO	0.593796	0.273659	0.750160	0.564049	0.573072	0.824741
BAC	0.595608	0.344822	0.507697	0.895891	0.537038	0.547887
BRK	0.568930	0.412894	0.596912	0.709166	0.519312	0.578304
CVX	0.190125	0.815066	0.296874	0.303311	0.200572	0.273685
DAL	1.000000	0.156082	0.508122	0.594361	0.865357	0.596280
EOG	0.156082	1.000000	0.261002	0.300045	0.180979	0.238775
INTC	0.508122	0.261002	1.000000	0.519479	0.500612	0.747413
JPM	0.594361	0.300045	0.519479	1.000000	0.529637	0.528567
LUV	0.865357	0.180979	0.500612	0.529637	1.000000	0.580431
NVDA	0.596280	0.238775	0.747413	0.528567	0.580431	1.000000
SLB	0.191830	0.748056	0.276862	0.304411	0.214634	0.216958
UAL	0.923427	0.132629	0.476877	0.548954	0.813319	0.569261
WFC	0.619314	0.290887	0.519873	0.801181	0.576508	0.520476
XOM	0.157380	0.837306	0.303884	0.277910	0.176487	0.227618

Ticker Symbol	SLB	UAL	WFC	XOM
AAL	0.185075	0.928827	0.576018	0.125324
AMD	0.248181	0.554975	0.527571	0.259137

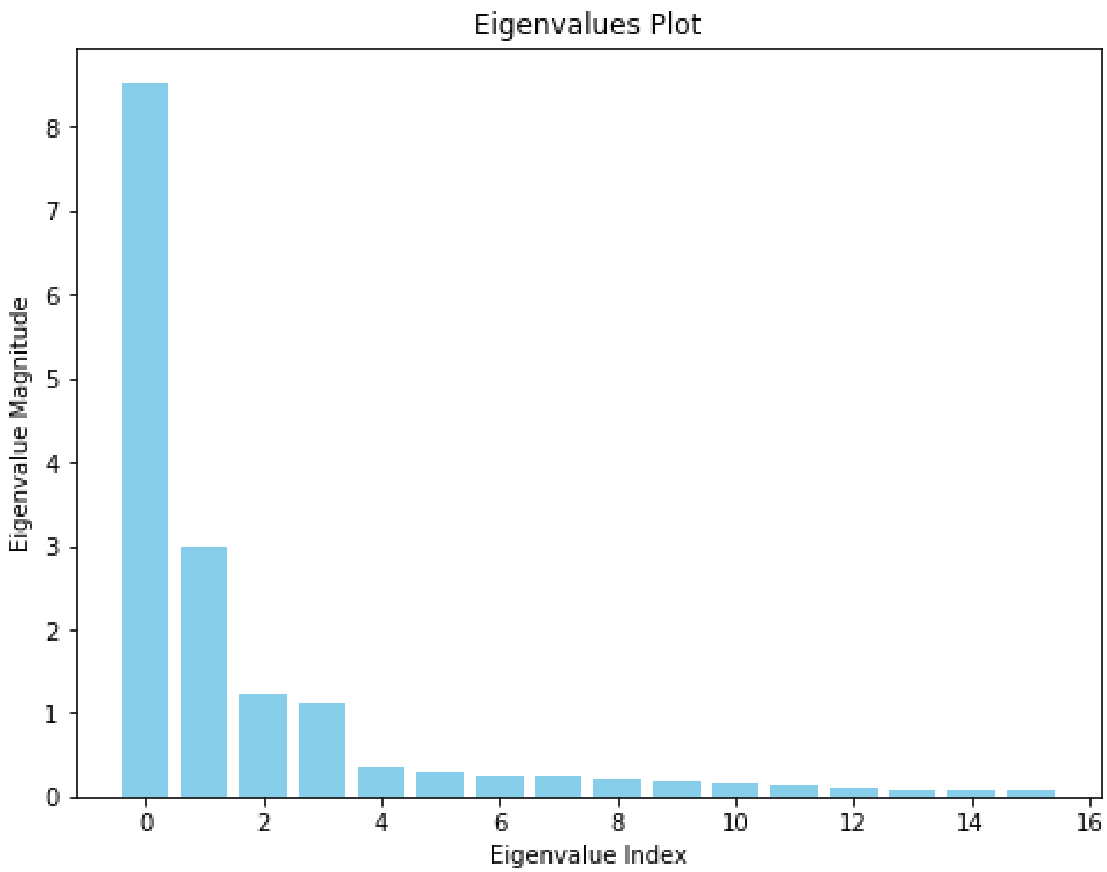
```
AVGO          0.261785  0.552298  0.531020  0.265218
BAC           0.338610  0.548971  0.865869  0.315299
BRK           0.399552  0.520238  0.698938  0.431941
CVX           0.774093  0.139055  0.307397  0.873425
DAL           0.191830  0.923427  0.619314  0.157380
EOG           0.748056  0.132629  0.290887  0.837306
INTC          0.276862  0.476877  0.519873  0.303884
JPM           0.304411  0.548954  0.801181  0.277910
LUV           0.214634  0.813319  0.576508  0.176487
NVDA          0.216958  0.569261  0.520476  0.227618
SLB           1.000000  0.168001  0.319815  0.788583
UAL           0.168001  1.000000  0.573084  0.124201
WFC           0.319815  0.573084  1.000000  0.276168
XOM           0.788583  0.124201  0.276168  1.000000
```

[13]:
```python
eigenvalues = np.linalg.eigvals(correlation_matrix)
eigenvalues
```

[13]:
```
array([8.52702392, 2.98514649, 1.23028626, 1.12078152, 0.34025521,
       0.30711004, 0.06357353, 0.06739485, 0.07830166, 0.10535933,
       0.11858138, 0.1726956 , 0.18402044, 0.24722549, 0.23754924,
       0.21469504])
```

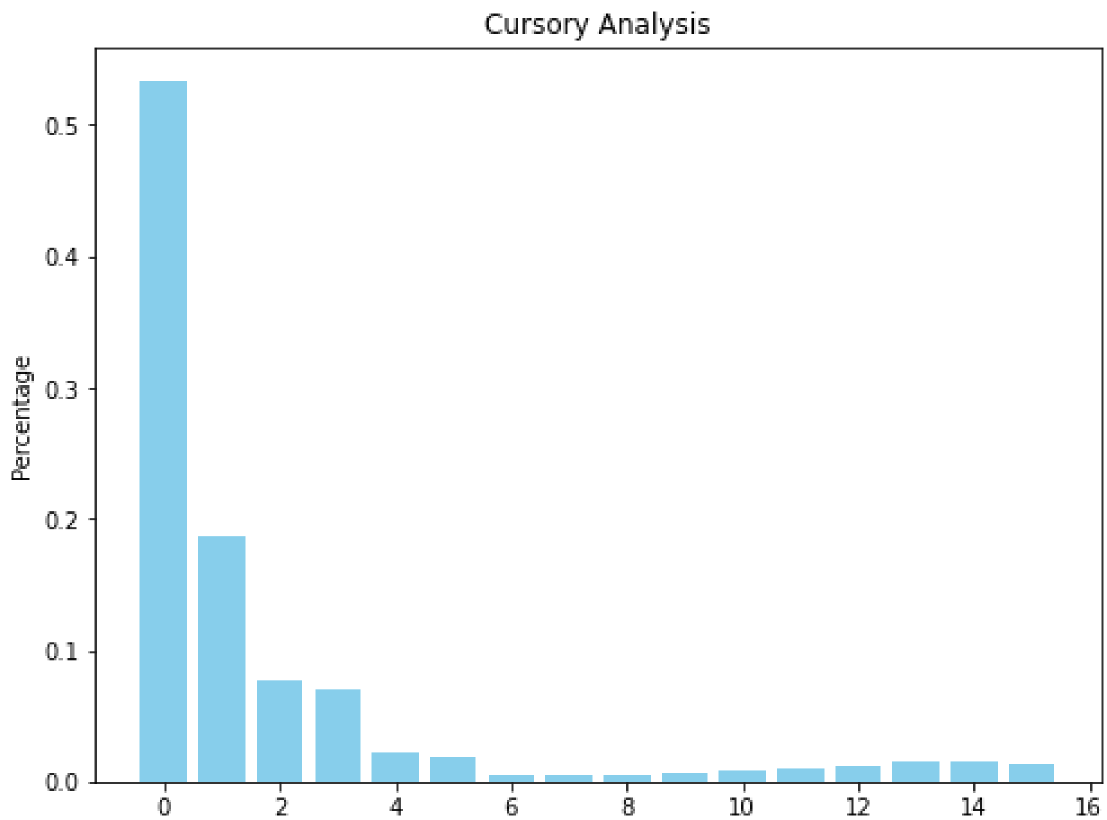
[14]:
```python
sorted_eigenvalues = np.sort(eigenvalues)[::-1]  # Reverse the order

# Plotting the eigenvalues
plt.figure(figsize=(8, 6))
plt.bar(range(len(sorted_eigenvalues)), sorted_eigenvalues, color='skyblue')
plt.xlabel('Eigenvalue Index')
plt.ylabel('Eigenvalue Magnitude')
plt.title('Eigenvalues Plot')
plt.show()
```




```
[15]: total_variance = np.sum(eigenvalues)
explained_variance_ratio = eigenvalues / total_variance

# Plotting the eigenvalues
plt.figure(figsize=(8, 6))
plt.bar(range(len(explained_variance_ratio)), explained_variance_ratio,␣
 ↪color='skyblue')
plt.ylabel('Percentage')
plt.title('Cursory Analysis')
plt.show()
```



0.7 We calculate the cumulative return of eigenportfolio

[16]:
```
cov_matrix = np.cov(scaled_data, rowvar=False)

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

# Select top eigenportfolio (e.g., first eigenvector)
top_eigenvector = eigenvectors[:, 0]  # Replace '0' with the index of the␣
 ↪desired eigenvector

# Construct eigenportfolio by normalizing weights
eigenportfolio = top_eigenvector / np.sum(top_eigenvector)
eigenportfolio_returns = np.dot(data_ret, eigenportfolio)

cumulative_return = np.cumprod(1 + eigenportfolio_returns) - 1

cumulative_return = pd.DataFrame(cumulative_return)
cumulative_return
```

[16]:
```
            0
0   -0.002287
1    0.028602
2    0.045073
3    0.027664
4    0.035953
..        …
247 -0.126009
248 -0.134971
249 -0.147446
250 -0.130102
251 -0.127446

[252 rows x 1 columns]
```

0.8 Now we calculate the cumulative return of market cap weighted portfolio

[17]:
```
mkt_cap = data[['Ticker Symbol','Names Date','Price or Bid/Ask Average','Shares␣
↪Outstanding','Returns without Dividends']]
mkt_cap
```

[17]:
```
     Ticker Symbol Names Date  Price or Bid/Ask Average  Shares Outstanding  \
0              XOM 2021-12-31                 61.189999             4233567
1              XOM 2022-01-03                 63.540001             4233567
2              XOM 2022-01-04                 65.930000             4233567
3              XOM 2022-01-05                 66.750000             4233567
4              XOM 2022-01-06                 68.320000             4233567
…              …          …                       …                  …
4279          AVGO 2022-12-23                552.429993              418000
4280          AVGO 2022-12-27                553.539978              418000
4281          AVGO 2022-12-28                544.890015              418000
4282          AVGO 2022-12-29                557.809998              418000
4283          AVGO 2022-12-30                559.130005              418000

      Returns without Dividends
0                      0.006580
1                      0.038405
2                      0.037614
3                      0.012437
4                      0.023521
…                          …
4279                  -0.001193
4280                   0.002009
4281                  -0.015627
4282                   0.023711
4283                   0.002366
```

```
[4284 rows x 5 columns]
```

[18]:
```
shr = pd.Series(mkt_cap.groupby('Ticker Symbol')['Shares Outstanding'].sum()/
 ↪252)
shr
```

[18]:
```
Ticker Symbol
AAL      6.494009e+05
AMD      1.567547e+06
AVGO     4.087312e+05
BAC      8.070363e+06
BRK      1.296293e+06
CVX      1.951552e+06
DAL      6.407667e+05
EOG      5.857664e+05
INTC     4.100060e+06
JPM      2.942416e+06
LUV      5.928831e+05
NVDA     2.491785e+06
SLB      1.412687e+06
UAL      3.261760e+05
WFC      3.826443e+06
XOM      4.201088e+06
Name: Shares Outstanding, dtype: float64
```

[19]:
```
mkt_ret = mkt_cap[['Ticker Symbol','Names Date','Price or Bid/Ask Average']]
mkt_ret = mkt_ret.pivot_table(index='Names Date', columns='Ticker Symbol',␣
 ↪values='Price or Bid/Ask Average')
mkt_ret
```

[19]:
```
Ticker Symbol        AAL         AMD        AVGO        BAC            BRK  \
Names Date
2021-12-31     17.959999  143.899994  665.409973  44.490002  225480.500000
2022-01-03     18.750000  150.240005  663.320007  46.180000  227300.395004
2022-01-04     19.020000  144.419998  670.919983  47.990002  233016.764999
2022-01-05     18.680000  136.149994  643.000000  47.180000  233792.085007
2022-01-06     18.570000  136.229996  637.030029  48.130001  236733.110001
…                    …           …           …          …              …
2022-12-23     12.710000   64.519997  552.429993  32.470001  231853.244995
2022-12-27     12.530000   63.270000  553.539978  32.529999  231130.274994
2022-12-28     12.320000   62.570000  544.890015  32.770000  230051.714996
2022-12-29     12.700000   64.820000  557.809998  33.139999  234517.029999
2022-12-30     12.720000   64.769997  559.130005  33.119999  234509.934372

Ticker Symbol         CVX        DAL         EOG       INTC         JPM  \
Names Date
2021-12-31     117.349998  39.080002   88.830002  51.500000  158.350006
```

```
2022-01-03     119.260002  40.290001   91.160004  53.209999  161.699997
2022-01-04     121.430000  40.590000   95.349998  53.139999  167.830002
2022-01-05     122.220001  40.279999   93.599998  53.869999  163.779999
2022-01-06     123.260002  40.110001   95.519997  54.009998  165.520004
…                     …          …           …          …           …
2022-12-23     177.399994  33.160000  130.610001  26.090000  131.279999
2022-12-27     179.630005  32.900002  132.080002  25.940001  131.740005
2022-12-28     176.979996  31.990000  127.400002  25.540001  132.460007
2022-12-29     178.320007  32.730000  128.630005  26.209999  133.220001
2022-12-30     179.490005  32.860001  129.520004  26.430000  134.100006

Ticker Symbol        LUV        NVDA        SLB        UAL        WFC  \
Names Date
2021-12-31     42.840000  294.109985  29.950001  43.779999  47.980000
2022-01-03     44.000000  301.209991  31.719999  45.490002  50.730000
2022-01-04     44.660000  292.899994  33.259998  46.250000  52.750000
2022-01-05     43.990002  276.040009  33.259998  45.779999  52.290001
2022-01-06     43.889999  281.779999  34.049999  45.770000  53.630001
…                    …           …          …          …          …
2022-12-23     36.090000  152.059998  52.990002  38.389999  40.980000
2022-12-27     33.939999  141.210007  53.500000  38.200001  41.040001
2022-12-28     32.189999  140.360001  52.599998  37.290001  41.119999
2022-12-29     33.380001  146.029999  52.910000  37.919998  41.330002
2022-12-30     33.669998  146.139999  53.459999  37.700001  41.290001

Ticker Symbol         XOM
Names Date
2021-12-31      61.189999
2022-01-03      63.540001
2022-01-04      65.930000
2022-01-05      66.750000
2022-01-06      68.320000
…                     …
2022-12-23     108.680000
2022-12-27     110.190002
2022-12-28     108.379997
2022-12-29     109.199997
2022-12-30     110.300003

[252 rows x 16 columns]
```

[20]:
```
market_caps = mkt_ret * shr
market_caps
```

[20]:
```
Ticker Symbol           AAL           AMD          AVGO           BAC  \
Names Date
2021-12-31     1.166324e+07  2.255700e+08  2.719738e+08  3.590505e+08
```

2022-01-03	1.217627e+07	2.355083e+08	2.711196e+08	3.726894e+08
2022-01-04	1.235161e+07	2.263852e+08	2.742259e+08	3.872967e+08
2022-01-05	1.213081e+07	2.134215e+08	2.628142e+08	3.807597e+08
2022-01-06	1.205937e+07	2.135469e+08	2.603740e+08	3.884266e+08
…	…	…	…	…
2022-12-23	8.253885e+06	1.011381e+08	2.257954e+08	2.620447e+08
2022-12-27	8.136993e+06	9.917871e+07	2.262491e+08	2.625289e+08
2022-12-28	8.000619e+06	9.808142e+07	2.227135e+08	2.644658e+08
2022-12-29	8.247391e+06	1.016084e+08	2.279943e+08	2.674518e+08
2022-12-30	8.260380e+06	1.015300e+08	2.285339e+08	2.672904e+08

Ticker Symbol	BRK	CVX	DAL	EOG \
Names Date				
2021-12-31	2.922889e+11	2.290147e+08	2.504116e+07	5.203363e+07
2022-01-03	2.946480e+11	2.327421e+08	2.581649e+07	5.339846e+07
2022-01-04	3.020581e+11	2.369770e+08	2.600872e+07	5.585282e+07
2022-01-05	3.030631e+11	2.385187e+08	2.581008e+07	5.482773e+07
2022-01-06	3.068756e+11	2.405483e+08	2.570115e+07	5.595240e+07
…	…	…	…	…
2022-12-23	3.005498e+11	3.462054e+08	2.124782e+07	7.650694e+07
2022-12-27	2.996126e+11	3.505573e+08	2.108122e+07	7.736802e+07
2022-12-28	2.982145e+11	3.453857e+08	2.049813e+07	7.462663e+07
2022-12-29	3.040029e+11	3.480008e+08	2.097229e+07	7.534713e+07
2022-12-30	3.039937e+11	3.502841e+08	2.105559e+07	7.586846e+07

Ticker Symbol	INTC	JPM	LUV	NVDA \
Names Date				
2021-12-31	2.111531e+08	4.659316e+08	2.539911e+07	7.328589e+08
2022-01-03	2.181642e+08	4.757887e+08	2.608686e+07	7.505506e+08
2022-01-04	2.178772e+08	4.938257e+08	2.647816e+07	7.298438e+08
2022-01-05	2.208702e+08	4.819089e+08	2.608093e+07	6.878324e+08
2022-01-06	2.214443e+08	4.870288e+08	2.602164e+07	7.021352e+08
…	…	…	…	…
2022-12-23	1.069706e+08	3.862804e+08	2.139715e+07	3.789008e+08
2022-12-27	1.063556e+08	3.876339e+08	2.012245e+07	3.518650e+08
2022-12-28	1.047155e+08	3.897525e+08	1.908491e+07	3.497470e+08
2022-12-29	1.074626e+08	3.919887e+08	1.979044e+07	3.638754e+08
2022-12-30	1.083646e+08	3.945780e+08	1.996237e+07	3.641495e+08

Ticker Symbol	SLB	UAL	WFC	XOM
Names Date				
2021-12-31	4.230997e+07	1.427998e+07	1.835927e+08	2.570646e+08
2022-01-03	4.481042e+07	1.483775e+07	1.941155e+08	2.669371e+08
2022-01-04	4.698596e+07	1.508564e+07	2.018449e+08	2.769777e+08
2022-01-05	4.698596e+07	1.493234e+07	2.000847e+08	2.804226e+08
2022-01-06	4.810198e+07	1.492907e+07	2.052122e+08	2.870183e+08
…	…	…	…	…

```
2022-12-23     7.485827e+07  1.252190e+07  1.568076e+08  4.565743e+08
2022-12-27     7.557874e+07  1.245992e+07  1.570372e+08  4.629179e+08
2022-12-28     7.430732e+07  1.216310e+07  1.573433e+08  4.553139e+08
2022-12-29     7.474525e+07  1.236859e+07  1.581469e+08  4.587588e+08
2022-12-30     7.552223e+07  1.229683e+07  1.579938e+08  4.633800e+08

[252 rows x 16 columns]
```

[21]:
```python
weights = market_caps.div(market_caps.sum(axis=1), axis=0)
weights
```

[21]:
```
Ticker Symbol       AAL       AMD      AVGO       BAC       BRK       CVX  \
Names Date
2021-12-31     0.000039  0.000764  0.000921  0.001215  0.989482  0.000775
2022-01-03     0.000041  0.000791  0.000910  0.001251  0.989274  0.000781
2022-01-04     0.000040  0.000742  0.000898  0.001269  0.989426  0.000776
2022-01-05     0.000040  0.000697  0.000858  0.001243  0.989721  0.000779
2022-01-06     0.000039  0.000689  0.000840  0.001253  0.989717  0.000776
…                   …         …         …         …         …         …
2022-12-23     0.000027  0.000334  0.000745  0.000864  0.991307  0.001142
2022-12-27     0.000027  0.000328  0.000749  0.000869  0.991334  0.001160
2022-12-28     0.000027  0.000326  0.000740  0.000879  0.991369  0.001148
2022-12-29     0.000027  0.000331  0.000744  0.000872  0.991401  0.001135
2022-12-30     0.000027  0.000331  0.000745  0.000872  0.991361  0.001142

Ticker Symbol       DAL       EOG      INTC       JPM       LUV      NVDA  \
Names Date
2021-12-31     0.000085  0.000176  0.000715  0.001577  0.000086  0.002481
2022-01-03     0.000087  0.000179  0.000732  0.001597  0.000088  0.002520
2022-01-04     0.000085  0.000183  0.000714  0.001618  0.000087  0.002391
2022-01-05     0.000084  0.000179  0.000721  0.001574  0.000085  0.002246
2022-01-06     0.000083  0.000180  0.000714  0.001571  0.000084  0.002264
…                   …         …         …         …         …         …
2022-12-23     0.000070  0.000252  0.000353  0.001274  0.000071  0.001250
2022-12-27     0.000070  0.000256  0.000352  0.001283  0.000067  0.001164
2022-12-28     0.000068  0.000248  0.000348  0.001296  0.000063  0.001163
2022-12-29     0.000068  0.000246  0.000350  0.001278  0.000065  0.001187
2022-12-30     0.000069  0.000247  0.000353  0.001287  0.000065  0.001188

Ticker Symbol       SLB       UAL       WFC       XOM
Names Date
2021-12-31     0.000143  0.000048  0.000622  0.000870
2022-01-03     0.000150  0.000050  0.000652  0.000896
2022-01-04     0.000154  0.000049  0.000661  0.000907
2022-01-05     0.000153  0.000049  0.000653  0.000916
2022-01-06     0.000155  0.000048  0.000662  0.000926
…                   …         …         …         …
```

```
2022-12-23     0.000247  0.000041  0.000517  0.001506
2022-12-27     0.000250  0.000041  0.000520  0.001532
2022-12-28     0.000247  0.000040  0.000523  0.001514
2022-12-29     0.000244  0.000040  0.000516  0.001496
2022-12-30     0.000246  0.000040  0.000515  0.001511

[252 rows x 16 columns]
```

[22]:
```
daily_mkt_ret = data[['Ticker Symbol','Names Date','Returns without Dividends']]
daily_mkt_ret = daily_mkt_ret.pivot_table(index='Names Date', columns='Ticker␣
 ↪Symbol', values='Returns without Dividends')
daily_mkt_ret = (daily_mkt_ret * weights).sum(axis=1)
daily_mkt_ret
```

[22]:
```
Names Date
2021-12-31   -0.003867
2022-01-03    0.007260
2022-01-04    0.025291
2022-01-05    0.003626
2022-01-06    0.011630
                ...
2022-12-23    0.011400
2022-12-27   -0.003113
2022-12-28   -0.005826
2022-12-29    0.018955
2022-12-30   -0.000231
Length: 252, dtype: float64
```

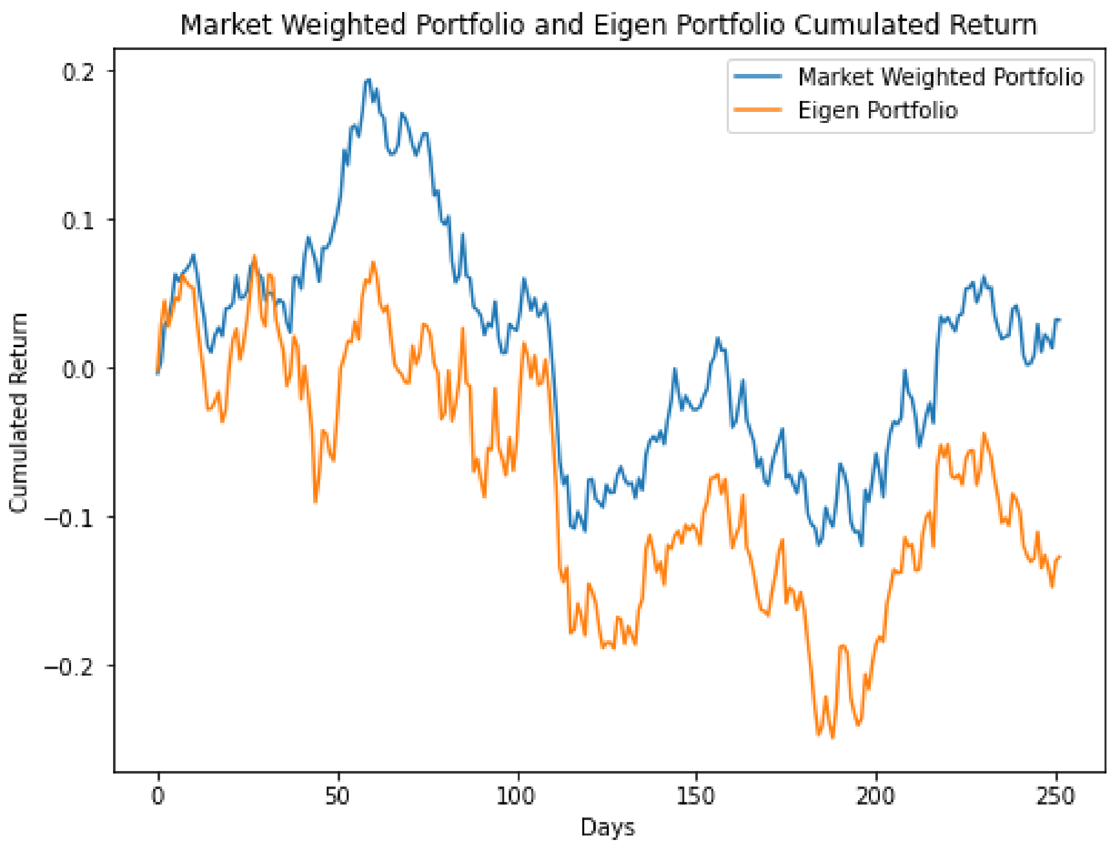
[23]:
```
daily_mkt_ret = daily_mkt_ret.reset_index(drop=True)
daily_mkt_ret = np.cumprod(1 + daily_mkt_ret) - 1
daily_mkt_ret = pd.DataFrame(daily_mkt_ret)
daily_mkt_ret
```

[23]:
```
            0
0   -0.003867
1    0.003365
2    0.028741
3    0.032471
4    0.044479
..        ...
247  0.022213
248  0.019031
249  0.013095
250  0.032298
251  0.032059

[252 rows x 1 columns]
```

0.9 We plot the cumulative return of eigenportfolio and market cap weighted portfolio in the same graph

[24]:
```
plt.figure(figsize=(8, 6))
plt.plot(daily_mkt_ret, label='Market Weighted Portfolio')
plt.plot(cumulative_return, label='Eigen Portfolio')
plt.xlabel('Days')
plt.ylabel('Cumulated Return')
plt.title('Market Weighted Portfolio and Eigen Portfolio Cumulated Return')
plt.legend()
plt.show()
```



[ ]:
```
```