M5L1 Solutions

May 1, 2023

Exercise 7

Perform cluster analysis on DJIA Index components using K-Means and build an equal weight portfolio from the selected stocks. Retrieve the list of DJIA components; identify and clean any missing data points. Cluster stocks based on weekly ATR and compare it with the original dataset.

Solutions

K-Means Clustering

Clustering is a branch of unsupervised machine learning models that seeks to learn from the properties of the data by identifying groups or clusters in the dataset.

The k-means algorithm searches for a predetermined number of clusters within an unlabeled dataset and is based on the assumptions that the optimal cluster will have cluster center and each point is closer to its own cluster center than to other cluster centers.

[1]:
```
# Ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Import Libraries
import pandas as pd
import numpy as np
import pyfolio as pf

from kneed import KneeLocator
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
```

[2]:
```
# Load the pre-saved data dict
ohlc = np.load('data/dow_ohlc.npy', allow_pickle='TRUE').item()
ohlc['MMM'].head()
```

[2]:
```
                 Open       High        Low      Close   Volume
Date
2009-12-31  62.906018  63.123738  61.967570  62.065166  2049800
```

```
2010-01-04  62.380479  62.650753  62.065162  62.327927  3043700
2010-01-05  62.162779  62.485606  61.336941  61.937550  2847000
2010-01-06  62.973588  63.514135  62.695811  62.815929  5268500
2010-01-07  62.553137  62.883472  61.652230  62.860950  4470100
```

[3]:
```python
# List of DJIA stocks
dow_stocks = ['MMM', 'AXP', 'AMGN', 'AAPL', 'BA', 'CAT', 'CVX', 'CSCO', 'KO',␣
 ↪'DOW', 'GS', 'HD', 'HON', 'IBM', 'INTC', 'JNJ', 'JPM', 'MCD', 'MRK', 'MSFT',␣
 ↪'NKE', 'PG', 'CRM', 'TRV', 'UNH', 'VZ', 'V', 'WBA', 'WMT', 'DIS']
```

[4]:
```python
# Function to calculate average true range
def ATR(df,n):
    "function to calculate Average True Range"
    df = df.copy()

    df['H-L']  = abs(df['High']-df['Low'])
    df['H-PC'] = abs(df['High']-df['Close'].shift(1))
    df['L-PC'] = abs(df['Low']-df['Close'].shift(1))

    df['TR']   = df[['H-L','H-PC','L-PC']].max(axis=1,skipna=False)
    df['ATR']  = df['TR'].rolling(n).mean()

    df2 = df.drop(['H-L','H-PC','L-PC'],axis=1)
    return df2['ATR']
```

[5]:
```python
# Add ATR for each stocks
for symbol in dow_stocks:
    ohlc[symbol]['ATR'] = ATR(ohlc[symbol],21)
```

[6]:
```python
# Subsume into dataframe
df = pd.DataFrame({symbol: ohlc[symbol]['ATR'] for symbol in dow_stocks})

# Check for missing values
df.isnull().sum()
```

[6]:
```
MMM       22
AXP       22
AMGN      22
AAPL      22
BA        22
CAT       43
CVX       22
CSCO      22
KO        22
DOW     2340
GS        22
HD        22
```

```
HON       22
IBM       22
INTC      22
JNJ       22
JPM       22
MCD       22
MRK       22
MSFT      22
NKE       22
PG        22
CRM       22
TRV       22
UNH       22
VZ        22
V         22
WBA       22
WMT       22
DIS       22
dtype: int64
```

[7]:

```
# Fill forward the missing values and drop DOW company from the list
df.fillna(method='bfill', axis=0, inplace=True)
df.drop(['DOW'], axis=1, inplace=True)
```

[8]:

```
# Resample to a weekly timeframe for cluster analysis
px = df.resample('W-FRI').mean()
px = px.T

# Check output
px.head(2)
```

[8]:

```
Date  2010-01-01  2010-01-08  2010-01-15  2010-01-22  2010-01-29  2010-02-05  \
MMM     1.113625    1.113625    1.113625    1.113625    1.113625    1.153880
AXP     1.035185    1.035185    1.035185    1.035185    1.035185    1.046649

Date  2010-02-12  2010-02-19  2010-02-26  2010-03-05  …  2020-07-31  \
MMM     1.233820    1.308653    1.232967    1.071002  …    3.304308
AXP     1.079262    1.101860    0.908801    0.774188  …    2.843043

Date  2020-08-07  2020-08-14  2020-08-21  2020-08-28  2020-09-04  2020-09-11  \
MMM     3.337810    3.339979    3.176965    2.863492    2.869821    3.166674
AXP     2.639048    2.661429    2.715334    2.638858    2.682953    2.666072

Date  2020-09-18  2020-09-25  2020-10-02
MMM     3.413893    4.169714    4.322380
AXP     2.634953    3.009524    3.067619
```

```
[2 rows x 562 columns]
```

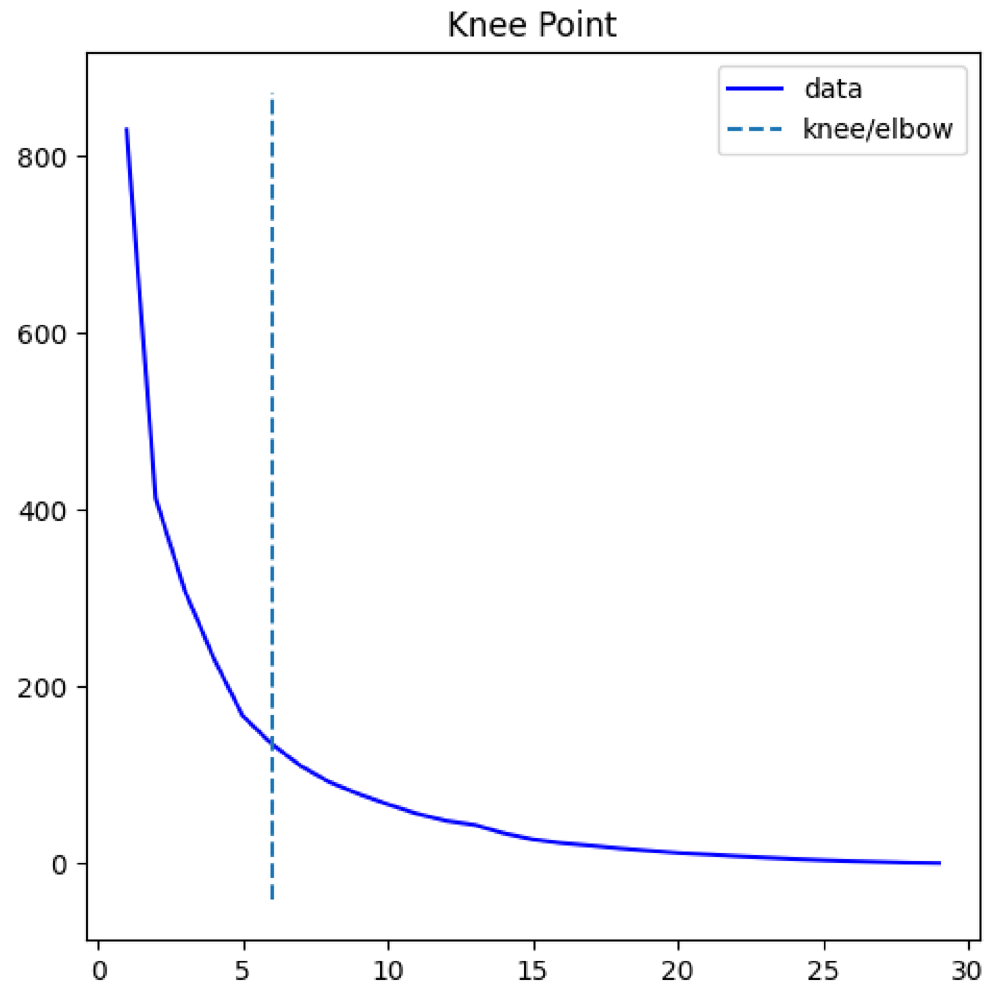
Elbow Plot The number of clusters is a hyperparameter to clustering models and choose the optimal number of clusters is critical for the model. We identify the elbow point programmatically for this exercise.

[9]:
```
scaler = MinMaxScaler()
scaled_px = scaler.fit_transform(px)
```

[10]:
```
# Get the inertia
sse = []
for k in range(1, 30):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(scaled_px)
    sse.append(kmeans.inertia_)

# Knee Locator
kl = KneeLocator(range(1, 30), sse, curve="convex", direction="decreasing")
kl.elbow
```

[10]:
```
6
```

[11]:
```
# Raw data and knee.
kl.plot_knee()
```



Build Clusters

We will now fit the cluster model.

[12]:
```
# Build clusters
model = KMeans(n_clusters=6)
model.fit(scaled_px)

labels = model.predict(scaled_px)
labels
```

[12]:
```
array([1, 0, 3, 4, 5, 1, 1, 4, 4, 2, 0, 0, 1, 4, 0, 0, 0, 4, 0, 4, 0, 1,
       0, 5, 4, 0, 0, 0, 0], dtype=int32)
```

[13]:
```
# Remove DOW from the list
companies = dow_stocks
```

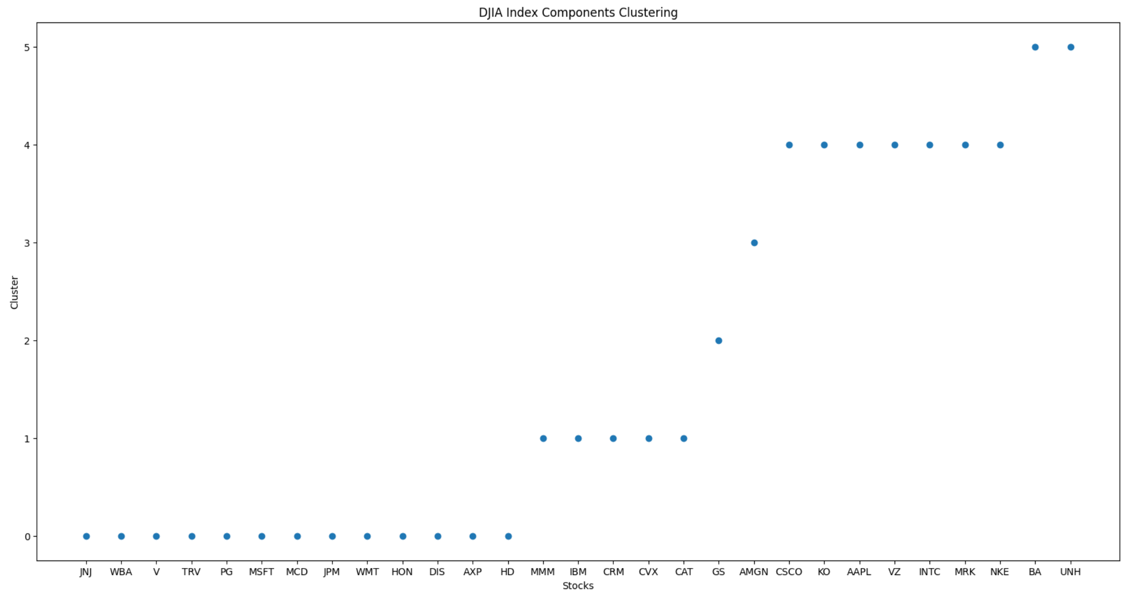
```
companies.remove('DOW')
```

[14]:
```
df1 = pd.DataFrame({'Cluster': labels,
                    'Companies': companies,
                    'ATR': px.mean(axis=1),
                  }).sort_values(by=['Cluster'], axis = 0)

df1 = df1.reset_index(drop=True)
df1
```

[14]:

	Cluster	Companies	ATR
0	0	JNJ	1.306309
1	0	WBA	1.119627
2	0	V	1.501398
3	0	TRV	1.423909
4	0	PG	1.030430
5	0	MSFT	1.290491
6	0	MCD	1.650644
7	0	JPM	1.310115
8	0	WMT	1.116245
9	0	HON	1.606682
10	0	DIS	1.486680
11	0	AXP	1.424071
12	0	HD	1.987495
13	1	MMM	2.101291
14	1	IBM	2.095919
15	1	CRM	2.199421
16	1	CVX	1.630457
17	1	CAT	2.079966
18	2	GS	3.640206
19	3	AMGN	2.664776
20	4	CSCO	0.538751
21	4	KO	0.493262
22	4	AAPL	0.680922
23	4	VZ	0.606684
24	4	INTC	0.695828
25	4	MRK	0.865679
26	4	NKE	0.969232
27	5	BA	3.791872
28	5	UNH	2.748506

[15]:
```
# Plot Clusters
plt.figure(figsize=(20,10))
plt.scatter(df1.Companies, df1.Cluster)
plt.xlabel('Stocks')
plt.ylabel('Cluster')
plt.title('DJIA Index Components Clustering');
```



Portfolio Construction Shortlisting the six stocks from the above clusters (one for each), we will now build a portfolio and compare the returns with all stock portfolio.

[16]:
```
# Cluster portfolio stocks
portfolio_stocks = ['GS', 'JNJ', 'UNH', 'CSCO', 'CRM', 'AMGN']
port = pd.DataFrame({symbol: ohlc[symbol]['Close'] for symbol in␣
 ↪portfolio_stocks})
port.dropna(inplace=True)

port
```

[16]:
```
                    GS         JNJ         UNH       CSCO         CRM  \
Date
2009-12-31  144.296997   46.714996   25.778193  18.048286   18.442499
2010-01-04  147.920776   46.910812   26.666224  18.613705   18.705000
2010-01-05  150.535919   46.366856   26.623932  18.530785   18.625000
2010-01-06  148.929138   46.743999   26.886124  18.410154   18.592501
2010-01-07  151.843475   46.410374   27.917923  18.493084   18.510000
…                  …           …           …          …           …
2020-09-23  186.119995  144.440002  292.140015  37.930145  235.990005
2020-09-24  195.110001  144.669998  292.660004  37.504074  237.550003
2020-09-25  194.949997  145.660004  302.500000  38.098591  242.740005
2020-09-28  199.070007  147.110001  303.230011  38.772377  246.669998
2020-09-29  196.789993  147.059998  304.149994  38.703018  247.449997

                  AMGN
Date
```

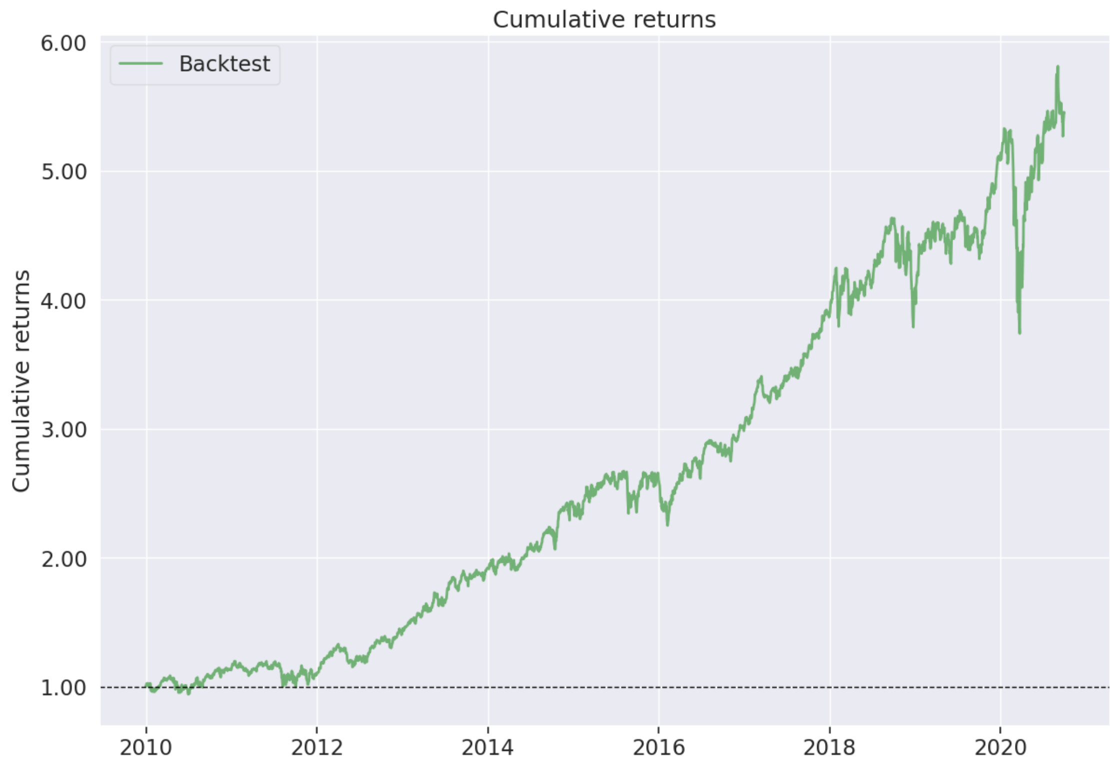
```
2009-12-31   45.404182
2010-01-04   46.327202
2010-01-05   45.925884
2010-01-06   45.580757
2010-01-07   45.163395
…                    …
2020-09-23  242.589996
2020-09-24  240.320007
2020-09-25  243.820007
2020-09-28  247.029999
2020-09-29  248.300003

[2705 rows x 6 columns]
```

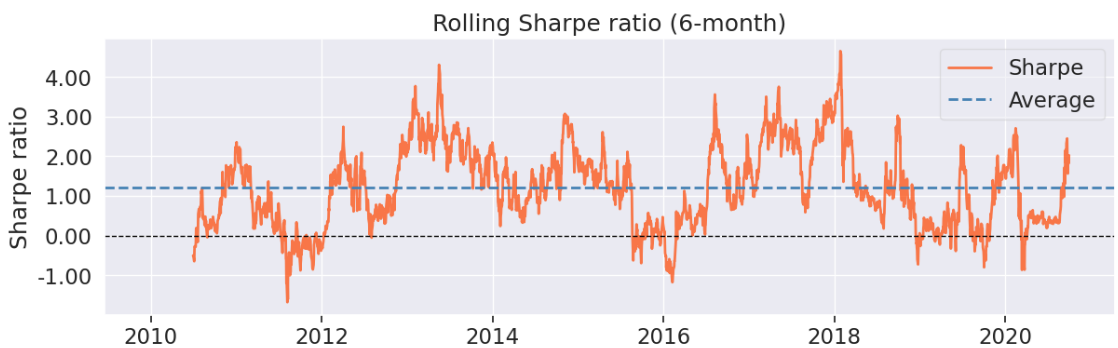
[17]:
```
# Calculate portfolio returns
portfolio_returns = port.pct_change().fillna(0)
port['Returns'] = portfolio_returns.mean(axis=1)
port.head(2)
```

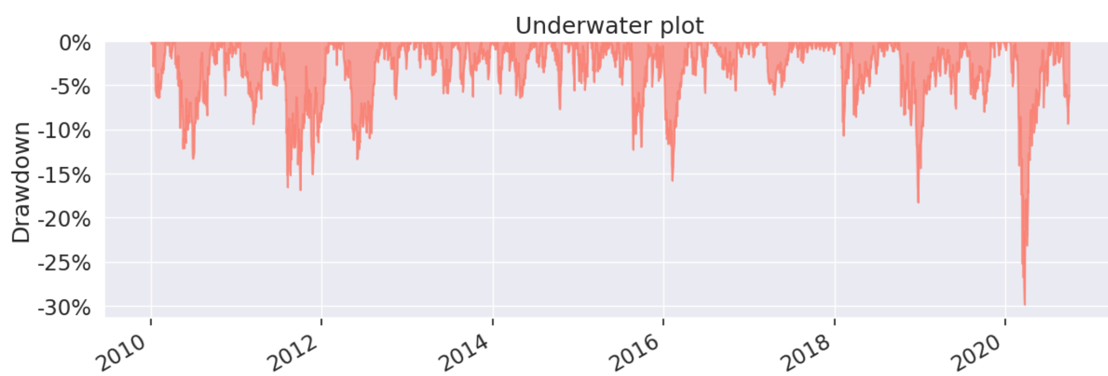
[17]:
```
                    GS        JNJ        UNH       CSCO        CRM       AMGN  \
Date
2009-12-31  144.296997  46.714996  25.778193  18.048286  18.442499  45.404182
2010-01-04  147.920776  46.910812  26.666224  18.613705  18.705000  46.327202

             Returns
Date
2009-12-31  0.000000
2010-01-04  0.021607
```

[18]:
```
# Create Tear sheet using pyfolio
pf.create_simple_tear_sheet(port['Returns'])
```

```
<IPython.core.display.HTML object>
```


Cumulative returns
Backtest
Cumulative returns
6.00
5.00
4.00
3.00
2.00
1.00
2010
2012
2014
2016
2018
2020


Rolling Sharpe ratio (6-month)
Sharpe
Average
Sharpe ratio
4.00
3.00
2.00
1.00
0.00
-1.00
2010
2012
2014
2016
2018
2020


Underwater plot
Drawdown
0%
-5%
-10%
-15%
-20%
-25%
-30%
2010
2012
2014
2016
2018
2020

[19]:
```
# All stocks portfolio
all_stocks = pd.DataFrame({symbol: ohlc[symbol]['Close'] for symbol in␣
 ↪companies})
all_stocks.dropna(inplace=True)
all_stocks.head(2)
```

[19]:
```
                  MMM        AXP       AMGN      AAPL         BA        CAT  \
Date
2009-12-31  62.065166  34.434761  45.404182  6.503574  41.856789  41.498463
2010-01-04  62.327927  34.774685  46.327202  6.604801  43.441975  42.634403

                  CVX       CSCO         KO          GS  …        NKE  \
Date                                                     …
2009-12-31  50.924435  18.048286  18.951757  144.296997  …  12.066024
2010-01-04  52.293617  18.613705  18.965061  147.920776  …  11.934528

                   PG        CRM        TRV        UNH         VZ          V  \
Date
2009-12-31  43.431492  18.442499  38.205708  25.778193  18.633041  17.935587
2010-01-04  43.782478  18.705000  38.167397  26.666224  18.717409  18.075037

                  WBA        WMT        DIS
Date
2009-12-31  28.350834  40.954620  28.090706
2010-01-04  28.798639  41.552284  27.933924

[2 rows x 29 columns]
```

[20]:
```
# Calculate all stocks portfolio returns
all_stocks_returns = all_stocks.pct_change().fillna(0)
all_stocks['Returns'] = all_stocks_returns.mean(axis=1)
all_stocks.head(2)
```

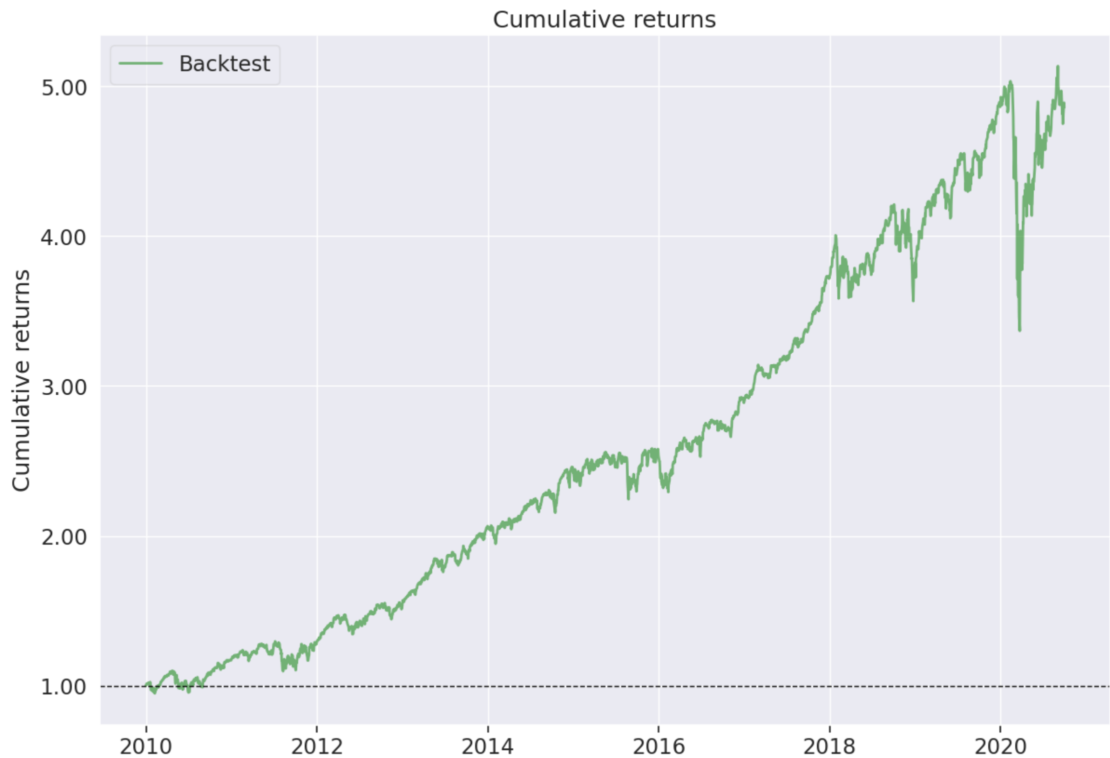
[20]:
```
                  MMM        AXP       AMGN      AAPL         BA        CAT  \
Date
2009-12-31  62.065166  34.434761  45.404182  6.503574  41.856789  41.498463
2010-01-04  62.327927  34.774685  46.327202  6.604801  43.441975  42.634403

                  CVX       CSCO         KO          GS  …         PG  \
Date                                                     …
2009-12-31  50.924435  18.048286  18.951757  144.296997  …  43.431492
2010-01-04  52.293617  18.613705  18.965061  147.920776  …  43.782478

                  CRM        TRV        UNH         VZ          V        WBA  \
Date
2009-12-31  18.442499  38.205708  25.778193  18.633041  17.935587  28.350834
2010-01-04  18.705000  38.167397  26.666224  18.717409  18.075037  28.798639
```

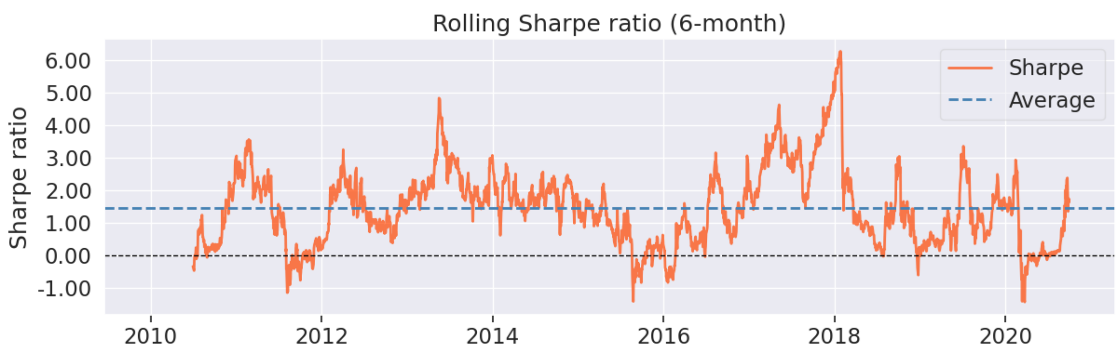
```
                  WMT        DIS   Returns
Date
2009-12-31  40.954620  28.090706  0.000000
2010-01-04  41.552284  27.933924  0.013946

[2 rows x 30 columns]
```

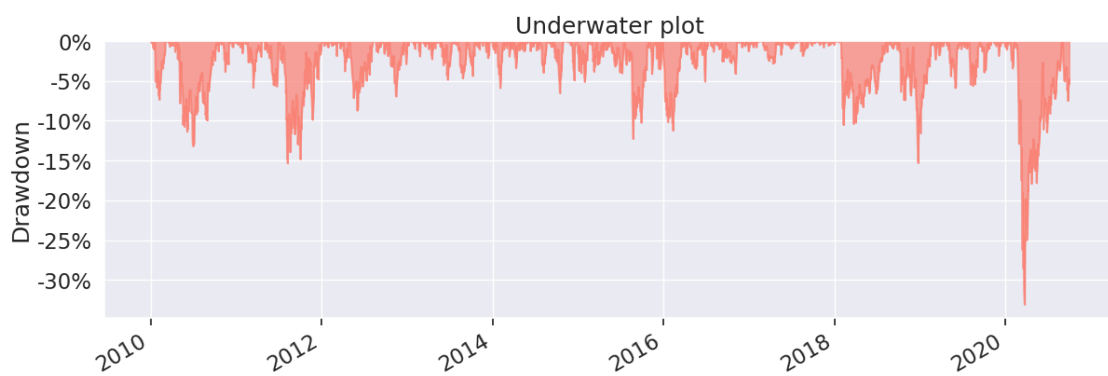
[21]:

```
# Create Tear sheet using pyfolio
pf.create_simple_tear_sheet(all_stocks['Returns'])
plt.show()
```

```
<IPython.core.display.HTML object>
```


Cumulative returns
Backtest
5.00
4.00
3.00
2.00
1.00
Cumulative returns
2010
2012
2014
2016
2018
2020


Rolling Sharpe ratio (6-month)
Sharpe
Average
6.00
5.00
4.00
3.00
2.00
1.00
0.00
-1.00
Sharpe ratio
2010
2012
2014
2016
2018
2020


Underwater plot
0%
-5%
-10%
-15%
-20%
-25%
-30%
Drawdown
2010
2012
2014
2016
2018
2020

Conclusion The cluster stocks generated a CAGR of 17% with a maximum drawdown of ~30% while the all stocks portfolio consisting of 29 DJIA index stocks generated an annualized return of ~16% with a maximum drawdown of 33%. This study highlight that with 20% of all stocks, we can construct a portfolio that can outperform the all stocks portfolio with an alpha of 59% and an improved sortino ratio.

Note:

1. The data is not treated for in/out sample as the objective here is to showcase the application of clustering methods. Accordingly, the actual results may vary.
2. Arbitrary selection of Cluster stocks can be avoided by adopting a minimum distance measure in stock selection.

References

- Scikit-learn K-Means Clustering
- Pyfolio-reloaded
- Python resources

* * *