Clustering - KMeans

Using K-Means algorithm to cluster 60 of cryptocurrencies

YU-HUNG, CHIANG 2021/10

Outline

- Get origin dataset
- Feature engineering
- Apply algorithm
- Decide numbers of cluster
- Show result

Get origin dataset

- Spot price from Binance exchange
- 2 years (2019/9/1 ~ 2021/9/30)
- Hourly data
- 60 of cryptocurrencies
- Only using Close price
- Split to two dataset, each for one year

Get origin dataset

Datetime	BTCUSDT	ETHUSDT	BNBUSDT	NEOUSDT	LTCUSDT	ADAUSDT	XRPUSDT	EOSUSDT	TUSDUSDT	IOTAUSDT	...	MTLUSDT
2019-09-01 00:00:00	9617.06	172.45	21.3579	8.828	64.70	0.04495	0.25837	3.3235	0.9995	0.2477	...	0.4065
2019-09-01 01:00:00	9614.99	172.71	21.4553	8.830	64.66	0.04504	0.25754	3.3259	0.9988	0.2478	...	0.4050
2019-09-01 02:00:00	9605.78	172.43	21.3114	8.820	64.60	0.04487	0.25697	3.3130	0.9991	0.2468	...	0.4047
2019-09-01 03:00:00	9623.02	172.57	21.4323	8.849	64.81	0.04503	0.25772	3.3176	0.9993	0.2478	...	0.4045
2019-09-01 04:00:00	9614.46	172.46	21.5311	8.833	64.82	0.04515	0.25713	3.3248	0.9991	0.2474	...	0.4040
...	...	...	...	...	...	...	...	...	...	...	...	...
2021-09-30 06:00:00	43371.92	3021.40	376.8000	38.460	151.50	2.09300	0.94410	3.8880	1.0000	1.0892	...	2.8330
2021-09-30 07:00:00	43286.88	3001.59	375.1000	38.110	150.70	2.08500	0.94160	3.8750	1.0000	1.0858	...	2.8120
2021-09-30 08:00:00	43062.03	2976.56	373.5000	37.880	150.00	2.07200	0.93390	3.8560	1.0000	1.0743	...	2.8080
2021-09-30 09:00:00	43147.37	2980.65	371.8000	38.070	150.30	2.07500	0.93320	3.8590	1.0000	1.0864	...	2.8220
2021-09-30 10:00:00	42826.78	2951.58	371.7000	37.820	149.90	2.07400	0.93190	3.8280	1.0000	1.0783	...	2.8050

18251 rows × 60 columns

```
[4]: # 是否有缺漏值
     df.isnull().any().all()

[4]: False

[5]: df_2019 = df.loc['2019-09-01':'2020-08-31']
     df_2020 = df.loc['2020-09-01':]
```

Feature engineering

- Calculate feature of crypto’s price
  - Mean of hourly return (in one year)
  - Standard deviation of hourly return
  - Skewness of hourly return
  - Kurtosis of hourly return
- Standardization
- Remove outliers

```python
def get_features(df, outlier_std):

    ## log return
    log_ret = np.log(df/df.shift(1))[1:]

    ## feature engirneering
    mean = log_ret.mean()
    std = log_ret.std()
    skew = log_ret.skew()
    kurt = log_ret.kurt()

    print('before row:', len(mean))

    ## standardized
    standard_mean = (mean - mean.mean()) / mean.std()
    standard_std = (std - std.mean()) / std.std()
    standard_skew = (skew - skew.mean()) / skew.std()
    standard_kurt = (kurt - kurt.mean()) / kurt.std()

    ## remove outliers
    features = pd.DataFrame({'mean':standard_mean, 'std':standard_std, 'skew':standard_skew, 'kurt':standard_kurt})
    features = features[(abs(features) > outlier_std) == False].dropna()

    print('after row:', len(features))
    return features
```

Feature engineering

```
features_2019 = get_features(df = df_2019, outlier_std = 2)
```

```
before row: 60
after row: 52
```

```
features_2019.head(2)
```

	mean	std	skew	kurt
BTCUSDT	-0.389254	-1.435759	-1.007337	0.738998
ETHUSDT	0.704156	-1.070515	-0.955507	0.186898

```
features_2020 = get_features(df = df_2020, outlier_std = 2)
```

```
before row: 60
after row: 56
```

```
features_2020.head(2)
```

	mean	std	skew	kurt
BTCUSDT	-0.156593	-1.169578	-0.168095	-0.155980
ETHUSDT	0.351400	-0.859241	-0.201200	-0.159164

- Split to 2019 and 2020 dataset

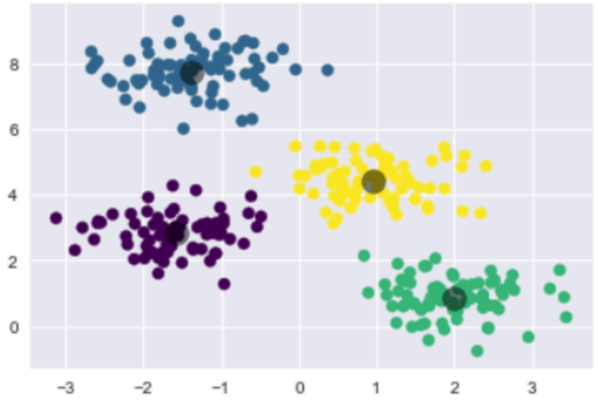
Apply algorithm

Resource:
https://ithelp.ithome.com.tw/articles/10207518

- KMeans - Base on distance
- Spectral Clustering - Base on density

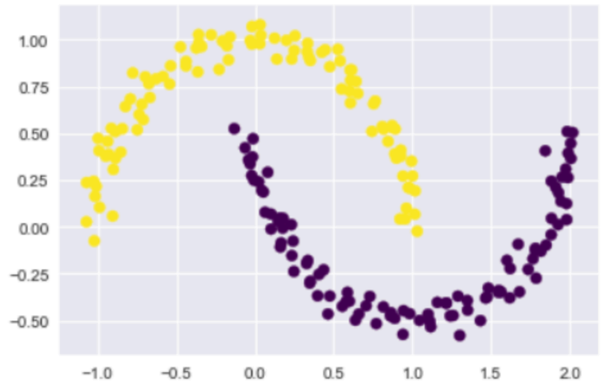
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```

由上圖，可以看到中心向量物件的位置(陰影部分)



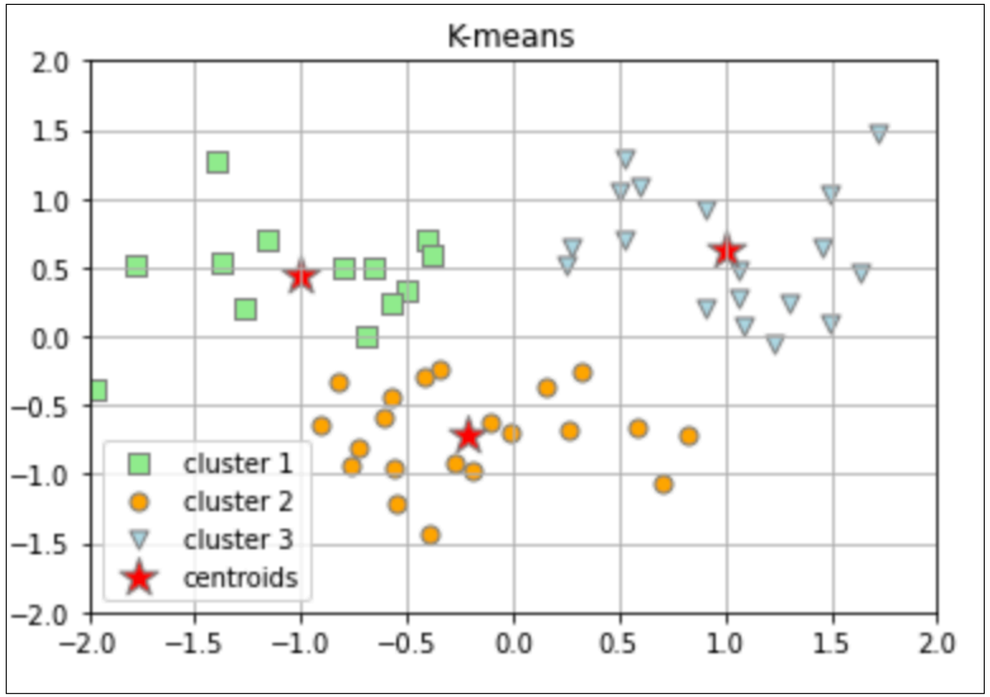
- 當然有時候資料數據離聚類的中心距離最近，並不一定為同一類別資料，反而會因為鄰居(neighbors)的關係，將資料分類為該類別，而我們匯入SKlearn中`SpectralClustering`模組來實作：

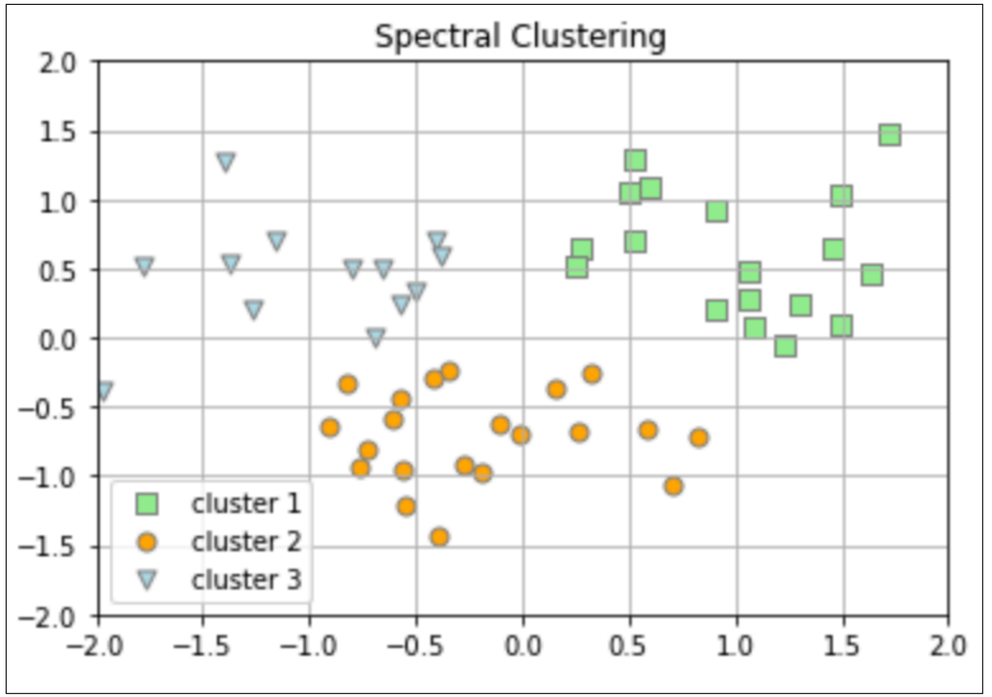
```
from sklearn.cluster import SpectralClustering
model = SpectralClustering(n_clusters=2, affinity='nearest_neighbors',
                           assign_labels='kmeans')
labels = model.fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels,
            s=50, cmap='viridis');
```



Apply algorithm

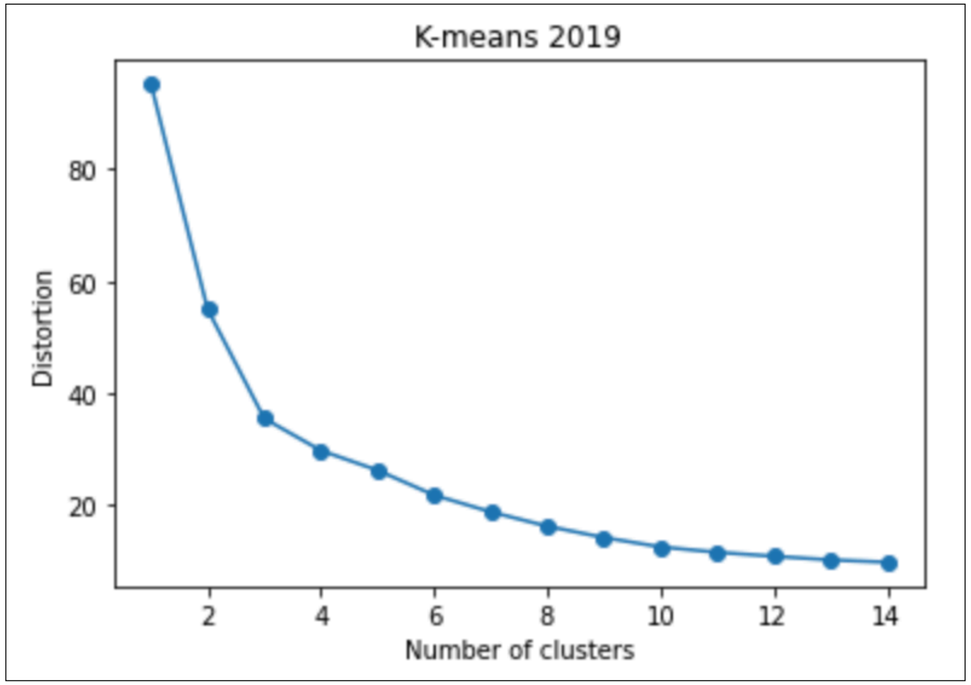
Try two algorithm with two features (Mean, Std) with cluster = 3

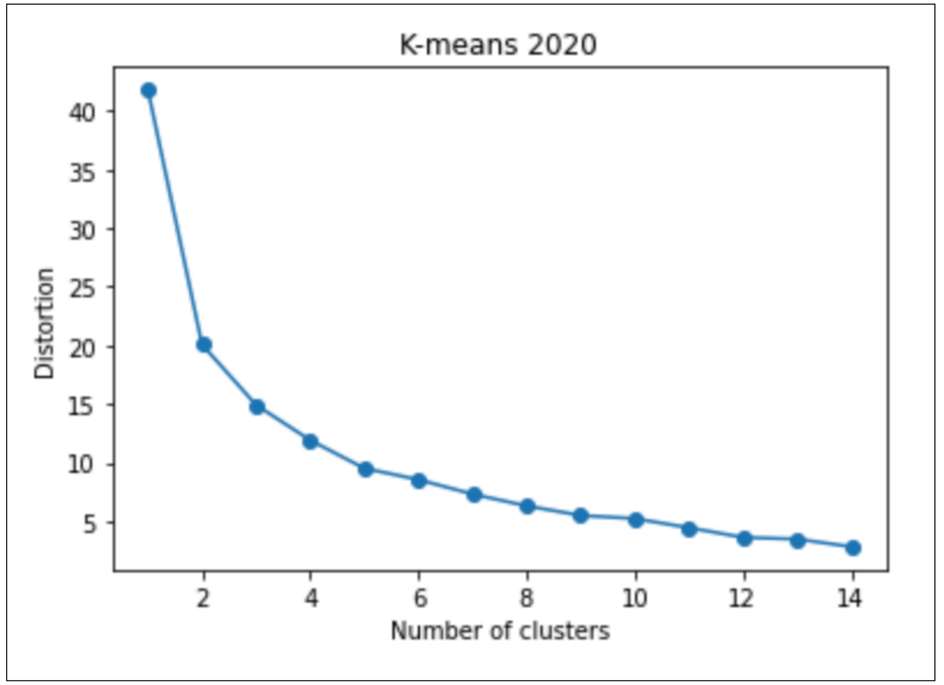




Because the output is similar, I just use K-Means for this work.

Decide numbers of cluster





Choose cluster = 3 for 2019 and 2020 features.

Show result

Example of data 2019

- Group 2 has the best return characteristic

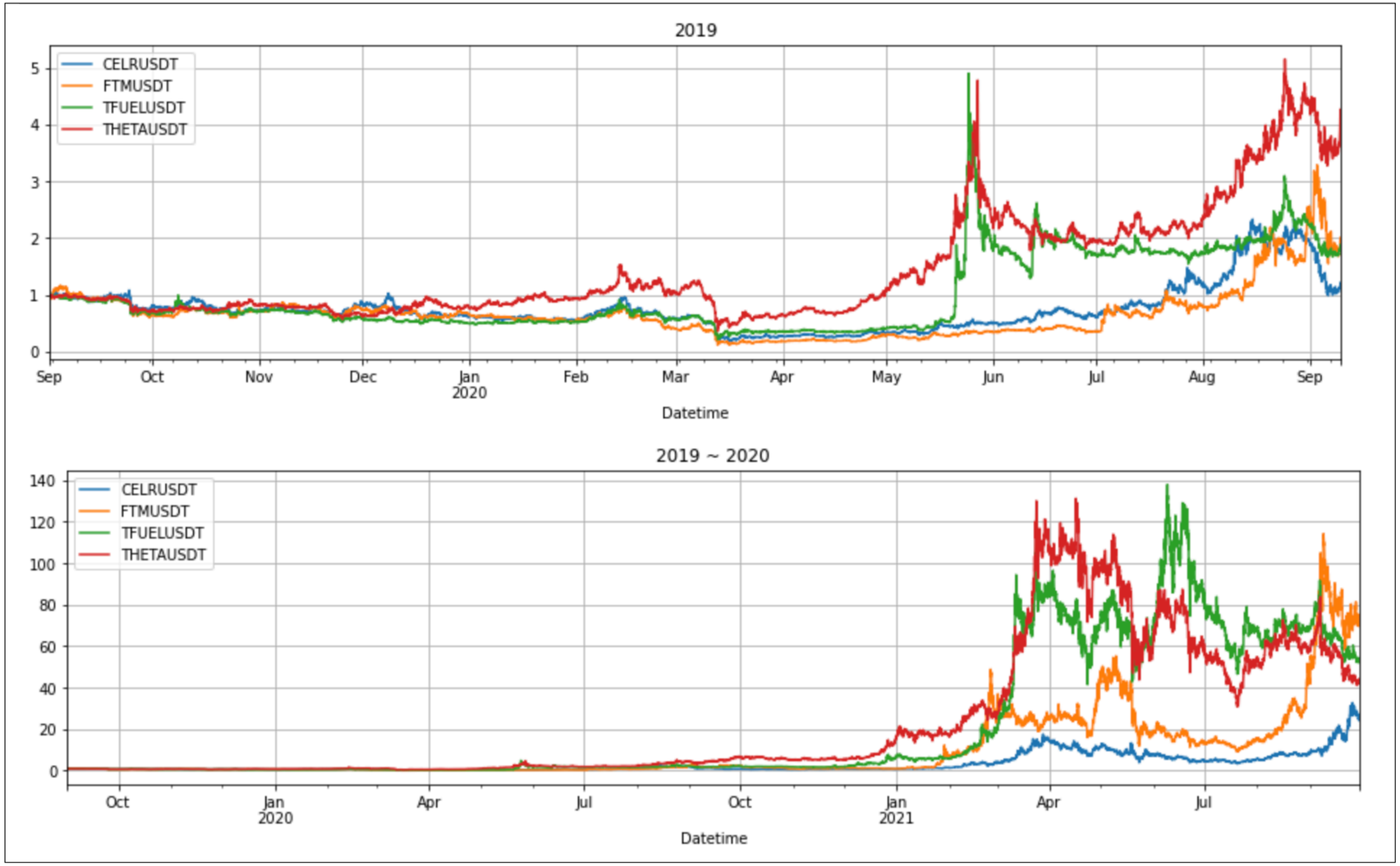
	mean	std	skew	kurt	group
THETAUSDT	1.463831	0.649692	-0.015694	-0.529716	2
ENJUSDT	1.068502	0.475396	0.330532	-0.489760	2
FETUSDT	0.523868	0.707620	-0.262033	-0.448594	2
ZRXUSDT	1.307401	0.239074	0.309018	-0.538364	2
ZILUSDT	1.069268	0.271600	0.062235	-0.463587	2
WAVESUSDT	1.090060	0.066265	0.154671	-0.465028	2
VETUSDT	1.496832	0.091206	0.125739	-0.437514	2
ICXUSDT	0.913932	0.199018	0.217892	-0.500657	2
OMGUSDT	1.642686	0.459330	0.978755	0.530453	2
CELRUSDT	0.278846	0.651387	-0.037395	-0.486122	2
DOCKUSDT	1.720727	1.481384	1.035385	0.085507	2
ATOMUSDT	1.231163	-0.050416	-0.297465	-0.323282	2
TFUELUSDT	0.525413	1.296534	1.321085	-0.228902	2
MFTUSDT	0.913153	0.920545	0.625501	-0.425906	2
FTMUSDT	0.597173	1.094163	0.038877	-0.447140	2
TOMOUSDT	0.500188	1.057177	-0.051976	-0.489822	2
ANKRUSDT	1.496616	1.037703	1.092235	-0.280630	2
ZECUSDT	0.151468	-0.363135	-0.276113	-0.064577	1
NANOUSDT	-0.419997	-0.297757	-0.282853	-0.454726	1
ETHUSDT	0.704156	-1.070515	-0.955507	0.186898	1
DOGEUSDT	-0.275251	-0.920693	0.475998	0.139939	1

BTCUSDT	-0.389254	-1.435759	-1.007337	0.738998	1
BATUSDT	0.321134	-0.263217	-0.226209	-0.416826	1
XMRUSDT	-0.195220	-0.972932	-0.330402	-0.222536	1
XLMUSDT	-0.012466	-0.694399	-0.270624	-0.383770	1
BNBUSDT	-0.555723	-0.962327	-1.296127	0.220044	1
NEOUSDT	0.580348	-0.653659	-0.066352	-0.306675	1
LTCUSDT	-0.766217	-0.928912	-0.587543	-0.157097	1
ADAUSDT	0.824389	-0.716679	-0.376276	-0.402639	1
XRPUSDT	-0.549705	-1.216030	-0.810277	-0.158030	1
EOSUSDT	-0.729018	-0.807164	-1.470697	0.159399	1
IOTAUSDT	-0.111288	-0.627585	-0.639734	-0.128434	1
ONTUSDT	-0.349522	-0.229834	-0.784548	-0.029300	1
TRXUSDT	0.262289	-0.685194	-0.747186	-0.083440	1
ETCUSDT	-0.605732	-0.587485	-0.664063	-0.253305	1
BTTUSDT	-0.911655	-0.641892	-0.454619	-0.096645	1
FUNUSDT	0.246988	0.510971	-0.692268	0.095216	1
WANUSDT	-0.507202	0.331077	0.101093	-0.532713	0
DENTUSDT	-1.269104	0.205808	0.313880	-0.394138	0
PERLUSDT	-1.398620	1.274020	-0.108308	-0.510920	0
COSUSDT	-1.783633	0.524429	0.244299	-0.475903	0
MTLUSDT	-0.804029	0.493515	0.462138	-0.335858	0
DASHUSDT	-0.573647	-0.442978	1.481255	0.489951	0
WINUSDT	-1.969917	-0.393753	0.011720	-0.445139	0
DUSKUSDT	-1.158947	0.706767	-0.116456	-0.425224	0

```
[19]: nice_2019

[19]: ['ANKRUSDT',
       'ATOMUSDT',
       'CELRUSDT',
       'DOCKUSDT',
       'ENJUSDT',
       'FETUSDT',
       'FTMUSDT',
       'ICXUSDT',
       'MFTUSDT',
       'OMGUSDT',
       'TFUELUSDT',
       'THETAUSDT',
       'TOMOUSDT',
       'VETUSDT',
       'WAVESUSDT',
       'ZILUSDT',
       'ZRXUSDT']

[20]: nice_2020

[20]: ['ADAUSDT',
       'BNBUSDT',
       'CELRUSDT',
       'CVCUSDT',
       'DENTUSDT',
       'FTMUSDT',
       'HOTUSDT',
       'MATICUSDT',
       'ONEUSDT',
       'TFUELUSDT',
       'THETAUSDT']
```

Show result

Observe equity of chosen crypto in both two year




```
[26]: # 1元投入，兩年後爆炸成長 24 ~ 72 倍（兩年報酬率 2311% ~ 7143%）
      cum_ret.iloc[-1]

[26]: CELRUSDT     24.110345
      FTMUSDT      72.434368
      TFUELUSDT    52.557173
      THETAUSDT    43.575566
      Name: 2021-09-30 10:00:00, dtype: float64

[25]: # 1元投入，兩年後爆炸成長 24 ~ 72 倍（兩年報酬率 2311% ~ 7143%）
      equity.iloc[-1]

[25]: CELRUSDT     24.110345
      FTMUSDT      72.434368
      TFUELUSDT    52.557173
      THETAUSDT    43.575566
      Name: 2021-09-30 10:00:00, dtype: float64
```

Next issue, how to find it ?