Ch2_(2)

March 27, 2024

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: from google.colab import drive
     drive.mount('/content/drive')
```

```
[3]: dataset = pd.read_csv('/content/drive/MyDrive/2024 DFMBA ㅂㅣㄱㄷㅔㅇㅣㅌㅓㅇㅘ␣
      ↪ㄱㅡㅁㅇㅠㅇㅈㅏㄹㅛㅂㅜㄴㅅㅓㄱ/data/BitstampData_sample.csv')
     # Data can be found at: https://www.kaggle.com/mczielinski/
      ↪bitcoin-historical-data
```

```
[4]: dataset.shape
```

```
[4]: (499999, 8)
```

```
[5]: dataset.tail(5)
```

```
[5]:          Timestamp  Open  High  Low  Close  Volume_(BTC)  Volume_(Currency)  \
     499994  1355317560   NaN   NaN  NaN    NaN           NaN                NaN
     499995  1355317620   NaN   NaN  NaN    NaN           NaN                NaN
     499996  1355317680   NaN   NaN  NaN    NaN           NaN                NaN
     499997  1355317740   NaN   NaN  NaN    NaN           NaN                NaN
     499998  1355317800   NaN   NaN  NaN    NaN           NaN                NaN

             Weighted_Price
     499994             NaN
     499995             NaN
     499996             NaN
     499997             NaN
     499998             NaN
```

```
[6]: dataset.describe()
```

```
[6]:          Timestamp          Open          High           Low         Close  \
     count  4.999990e+05  24521.000000  24521.000000  24521.000000  24521.000000
     mean   1.340318e+09      9.821951      9.824951      9.818321      9.821286
     std    8.660245e+06      2.666161      2.667011      2.664901      2.665678
```

```
min     1.325318e+09       3.800000       3.800000       3.800000       3.800000
25%     1.332818e+09       7.200000       7.200000       7.200000       7.200000
50%     1.340318e+09      10.800000      10.800000      10.790000      10.790000
75%     1.347818e+09      11.840000      11.840000      11.830000      11.840000
max     1.355318e+09      16.410000      16.410000      15.490000      16.000000

       Volume_(BTC)  Volume_(Currency)  Weighted_Price
count  24521.000000       24521.000000    24521.000000
mean      21.021827         206.749281        9.821525
std       55.478183         547.135377        2.665962
min        0.000000           0.000000        3.800000
25%        2.170000          21.019851        7.200000
50%        7.340000          67.920000       10.793242
75%       20.240016         199.280000       11.833367
max     2958.477574       31212.194780       16.386568
```

```
[7]: print('Null Values =',dataset.isnull().values.any())
```

```
Null Values = True
```

```
[8]: dataset[dataset.columns.values] = dataset[dataset.columns.values].ffill()
```

```
[9]: dataset=dataset.drop(columns=['Timestamp'])
```

```
[10]: dataset['short_mavg'] = dataset['Close'].rolling(window=10, min_periods=1,␣
       ↪center=False).mean()
      dataset['long_mavg'] = dataset['Close'].rolling(window=60, min_periods=1,␣
       ↪center=False).mean()
      dataset['signal'] = np.where(dataset['short_mavg'] > dataset['long_mavg'], 1.0,␣
       ↪0.0)
```

```
[11]: dataset.head()
```

```
[11]:    Open  High   Low  Close  Volume_(BTC)  Volume_(Currency)  Weighted_Price  \
      0  4.39  4.39  4.39   4.39      0.455581                2.0            4.39
      1  4.39  4.39  4.39   4.39      0.455581                2.0            4.39
      2  4.39  4.39  4.39   4.39      0.455581                2.0            4.39
      3  4.39  4.39  4.39   4.39      0.455581                2.0            4.39
      4  4.39  4.39  4.39   4.39      0.455581                2.0            4.39

         short_mavg  long_mavg  signal
      0        4.39       4.39     0.0
      1        4.39       4.39     0.0
      2        4.39       4.39     0.0
      3        4.39       4.39     0.0
      4        4.39       4.39     0.0
```

[12]:

```
def EMA(df, n):
    EMA = pd.Series(df['Close'].ewm(span=n, min_periods=n).mean(), name='EMA_' +␣
 ↪str(n))
    return EMA

dataset['EMA10'] = EMA(dataset, 10)
dataset['EMA30'] = EMA(dataset, 30)
dataset['EMA200'] = EMA(dataset, 200)
dataset.head()

def ROC(df, n):
    M = df.diff(n - 1)
    N = df.shift(n - 1)
    ROC = pd.Series(((M / N) * 100), name = 'ROC_' + str(n))
    return ROC

dataset['ROC10'] = ROC(dataset['Close'], 10)
dataset['ROC30'] = ROC(dataset['Close'], 30)

def MOM(df, n):
    MOM = pd.Series(df.diff(n), name='Momentum_' + str(n))
    return MOM

dataset['MOM10'] = MOM(dataset['Close'], 10)
dataset['MOM30'] = MOM(dataset['Close'], 30)

def RSI(series, period):
 delta = series.diff().dropna()
 u = delta * 0
 d = u.copy()
 u[delta > 0] = delta[delta > 0]
 d[delta < 0] = -delta[delta < 0]
 u[u.index[period-1]] = np.mean( u[:period] ) #first value is sum of avg gains
 u = u.drop(u.index[:(period-1)])
 d[d.index[period-1]] = np.mean( d[:period] ) #first value is sum of avg losses
 d = d.drop(d.index[:(period-1)])
 rs = u.ewm(com=period-1, adjust=False).mean() / \
 d.ewm(com=period-1, adjust=False).mean()
 return 100 - 100 / (1 + rs)

dataset['RSI10'] = RSI(dataset['Close'], 10)
dataset['RSI30'] = RSI(dataset['Close'], 30)
dataset['RSI200'] = RSI(dataset['Close'], 200)

def STOK(close, low, high, n):
 STOK = ((close - low.rolling(n).min()) / (high.rolling(n).max() - low.
 ↪rolling(n).min())) * 100
 return STOK

def STOD(close, low, high, n):
 STOK = ((close - low.rolling(n).min()) / (high.rolling(n).max() - low.
 ↪rolling(n).min())) * 100
 STOD = STOK.rolling(3).mean()
 return STOD

dataset['%K10'] = STOK(dataset['Close'], dataset['Low'], dataset['High'], 10)
dataset['%D10'] = STOD(dataset['Close'], dataset['Low'], dataset['High'], 10)
dataset['%K30'] = STOK(dataset['Close'], dataset['Low'], dataset['High'], 30)
dataset['%D30'] = STOD(dataset['Close'], dataset['Low'], dataset['High'], 30)
dataset['%K200'] = STOK(dataset['Close'], dataset['Low'], dataset['High'], 200)
dataset['%D200'] = STOD(dataset['Close'], dataset['Low'], dataset['High'], 200)

def MA(df, n):
    MA = pd.Series(df['Close'].rolling(n, min_periods=n).mean(), name='MA_' +␣
 ↪str(n))
    return MA

dataset['MA21'] = MA(dataset, 10)
dataset['MA63'] = MA(dataset, 30)
dataset['MA252'] = MA(dataset, 200)
```

[13]:
```
dataset.tail()
```

[13]:
```
         Open   High    Low  Close  Volume_(BTC)  Volume_(Currency)  \
499994  13.34  13.34  13.34  13.34           3.9             52.026
499995  13.34  13.34  13.34  13.34           3.9             52.026
499996  13.34  13.34  13.34  13.34           3.9             52.026
499997  13.34  13.34  13.34  13.34           3.9             52.026
499998  13.34  13.34  13.34  13.34           3.9             52.026

        Weighted_Price  short_mavg  long_mavg  signal  ...     RSI200  %K10  \
499994           13.34       13.34  13.343167     0.0  ...  44.066893   NaN
499995           13.34       13.34  13.342333     0.0  ...  44.066893   NaN
499996           13.34       13.34  13.341667     0.0  ...  44.066893   NaN
499997           13.34       13.34  13.341167     0.0  ...  44.066893   NaN
499998           13.34       13.34  13.341000     0.0  ...  44.066893   NaN

        %D10  %K30  %D30  %K200  %D200   MA21   MA63     MA252
499994   NaN   NaN   NaN   10.0   10.0  13.34  13.34  13.38030
499995   NaN   NaN   NaN   10.0   10.0  13.34  13.34  13.38005
499996   NaN   NaN   NaN   10.0   10.0  13.34  13.34  13.37980
499997   NaN   NaN   NaN   10.0   10.0  13.34  13.34  13.37955
499998   NaN   NaN   NaN   10.0   10.0  13.34  13.34  13.37930
```

```
[5 rows x 29 columns]
```

```
[14]: dataset=dataset.drop(['High','Low','Open',␣
       ↪'Volume_(Currency)','short_mavg','long_mavg'], axis=1)
```

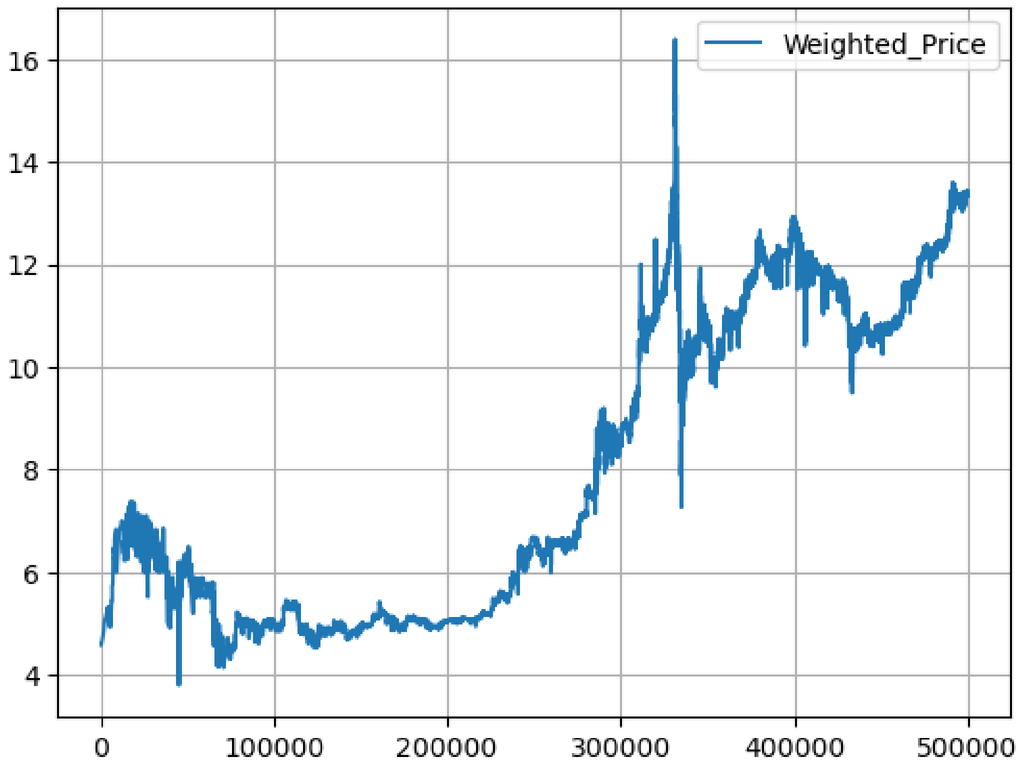
```
[15]: dataset = dataset.dropna( axis=0 )
```

```
[16]: dataset.round(2).head()
```

```
[16]:      Close  Volume_(BTC)  Weighted_Price  signal  EMA10  EMA30  EMA200  ROC10  \
      549   4.58           9.0            4.58     1.0   4.47   4.42     4.4   4.33
      550   4.58           9.0            4.58     1.0   4.49   4.43     4.4   4.33
      551   4.58           9.0            4.58     1.0   4.51   4.44     4.4   4.33
      552   4.58           9.0            4.58     1.0   4.52   4.45     4.4   4.33
      553   4.58           9.0            4.58     1.0   4.53   4.46     4.4   4.33

           ROC30  MOM10  ...  RSI200   %K10   %D10   %K30   %D30  %K200  %D200  \
      549   4.33   0.19  ...   100.0  100.0  100.0  100.0  100.0  100.0  100.0
      550   4.33   0.19  ...   100.0  100.0  100.0  100.0  100.0  100.0  100.0
      551   4.33   0.19  ...   100.0  100.0  100.0  100.0  100.0  100.0  100.0
      552   4.33   0.19  ...   100.0  100.0  100.0  100.0  100.0  100.0  100.0
      553   4.33   0.19  ...   100.0  100.0  100.0  100.0  100.0  100.0  100.0

           MA21  MA63  MA252
      549  4.45  4.41   4.39
      550  4.46  4.42   4.39
      551  4.48  4.42   4.39
      552  4.50  4.43   4.40
      553  4.52  4.43   4.40

      [5 rows x 23 columns]
```

```
[17]: dataset[['Weighted_Price']].plot( grid=True )
```

```
[17]: <AxesSubplot:>
```



[18]:

```
dataset.groupby(['signal']).size()
```

[18]:

```
signal
0.0    65722
1.0    54701
dtype: int64
```

[19]:

```
from sklearn.preprocessing import StandardScaler
Y = dataset["signal"]
X = dataset.loc[:, dataset.columns != 'signal']
```

[20]:

```
scaler = StandardScaler().fit(X)
rescaledDataset0 = pd.DataFrame( scaler.fit_transform( X ),
                              columns = X.columns, index = X.index )
rescaledDataset0.dropna( how='any', inplace=True )
rescaledDataset0.round(2).head(2)
```

[20]:

```
     Close  Volume_(BTC)  Weighted_Price  EMA10  EMA30  EMA200  ROC10  ROC30  \
549  -1.46         -0.25           -1.46  -1.50  -1.51   -1.52   3.24   2.78
550  -1.46         -0.25           -1.46  -1.49  -1.51   -1.52   3.24   2.78

     MOM10  MOM30  ...  RSI200  %K10  %D10  %K30  %D30  %K200  %D200  MA21  \
549   1.76   1.39  ...     2.8  1.02  1.04  1.04  1.06   1.14   1.15 -1.51
550   1.76   1.39  ...     2.8  1.02  1.04  1.04  1.06   1.14   1.15 -1.50

      MA63  MA252
549 -1.52  -1.53
550 -1.52  -1.52

[2 rows x 22 columns]
```

[21]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.95)
rescaledDataset = pca.fit_transform( rescaledDataset0 )
```

[22]:

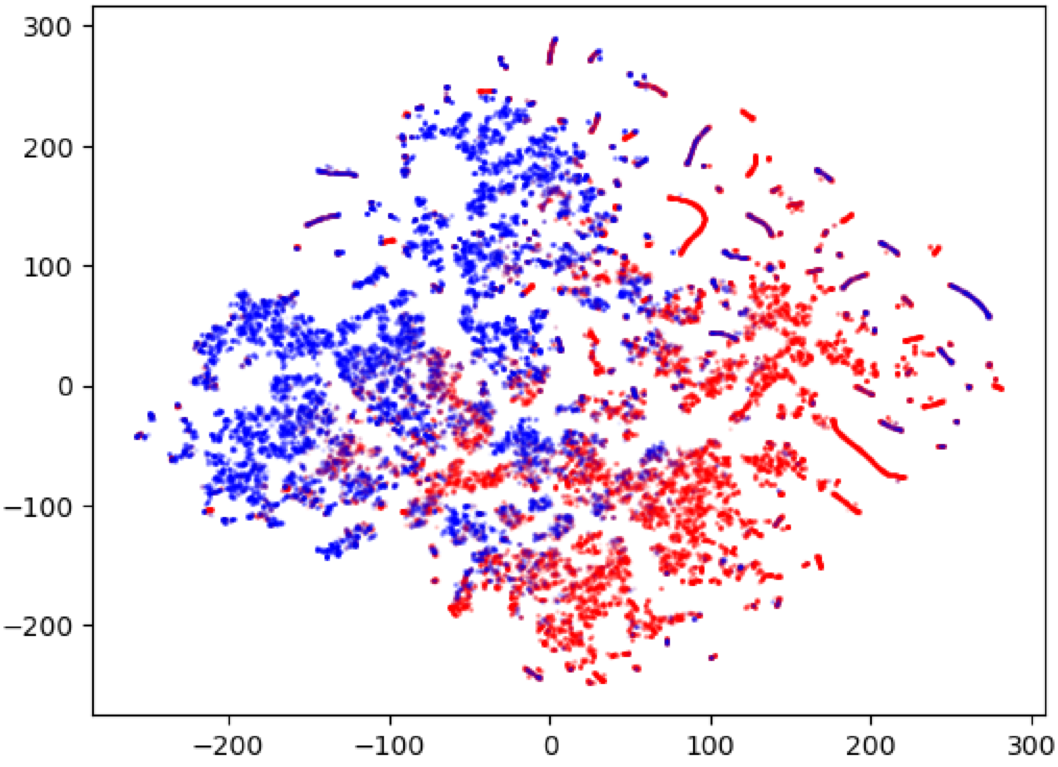
```
from sklearn.manifold import TSNE
tsne = TSNE( n_components=2, perplexity=100, n_iter = 10000, random_state=0,␣
 ↪verbose=1 )  ######### 더 큰값으로 다시 돌려보기
Z = tsne.fit_transform( rescaledDataset )
dftsne = pd.DataFrame( Z, columns=['x','y'] )
dftsne['signal'] = dataset['signal'].values
```

```
[t-SNE] Computing 301 nearest neighbors...
[t-SNE] Indexed 120423 samples in 0.077s...
[t-SNE] Computed neighbors for 120423 samples in 12.636s...
[t-SNE] Computed conditional probabilities for sample 1000 / 120423
[t-SNE] Computed conditional probabilities for sample 2000 / 120423
[t-SNE] Computed conditional probabilities for sample 3000 / 120423
[t-SNE] Computed conditional probabilities for sample 4000 / 120423
[t-SNE] Computed conditional probabilities for sample 5000 / 120423
...
[t-SNE] Computed conditional probabilities for sample 115000 / 120423
[t-SNE] Computed conditional probabilities for sample 116000 / 120423
[t-SNE] Computed conditional probabilities for sample 117000 / 120423
[t-SNE] Computed conditional probabilities for sample 118000 / 120423
[t-SNE] Computed conditional probabilities for sample 119000 / 120423
[t-SNE] Computed conditional probabilities for sample 120000 / 120423
[t-SNE] Computed conditional probabilities for sample 120423 / 120423
[t-SNE] Mean sigma: 0.000001
[t-SNE] KL divergence after 250 iterations with early exaggeration: 81.678078
[t-SNE] KL divergence after 10000 iterations: 0.951434
```

[23]:

```
plt.plot( dftsne.loc[ dftsne['signal']==0.0, 'x'], dftsne.
 ↪loc[dftsne['signal']==0.0, 'y'],
          'o', alpha=0.05, markersize=1, color='red')
plt.plot( dftsne.loc[ dftsne['signal']==1.0, 'x'], dftsne.
 ↪loc[dftsne['signal']==1.0, 'y'],
          'o', alpha=0.05, markersize=1, color='blue')
```

[23]: [<matplotlib.lines.Line2D at 0x15b482fed60>]



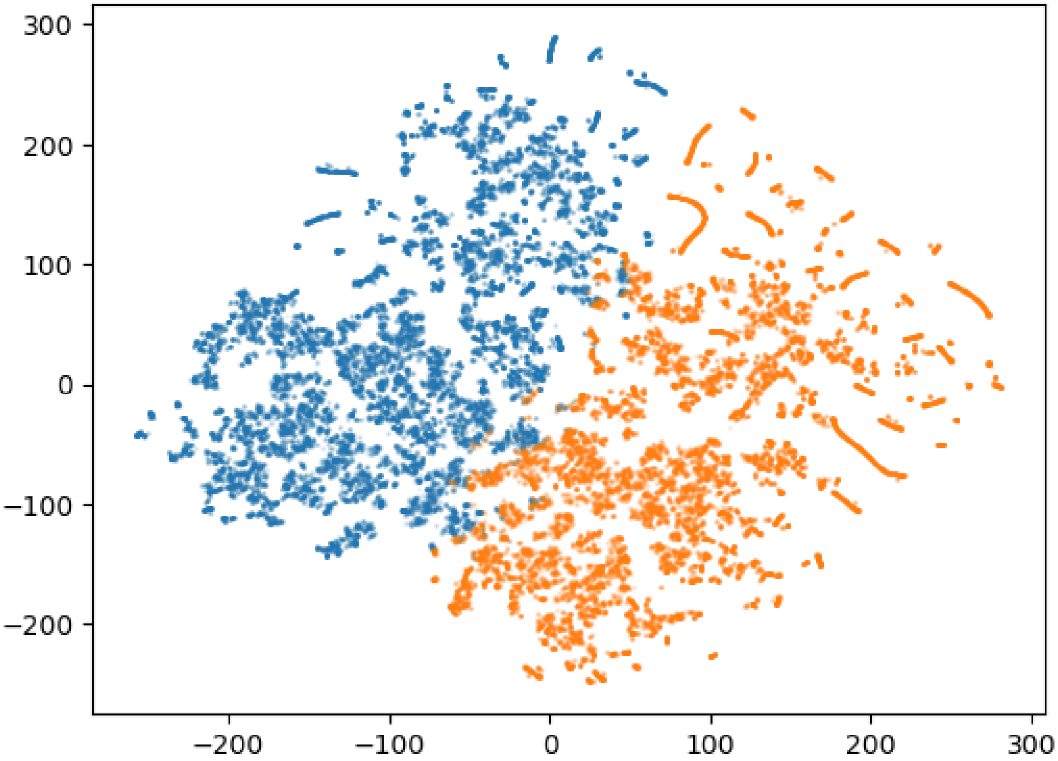
[24]:
```python
from sklearn.cluster import KMeans
km = KMeans( n_clusters= 2 )
km.fit_transform( rescaledDataset0 )
km.labels_
```

```
C:\Users\ahyun\AppData\Roaming\Python\Python39\site-
packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

[24]: array([0, 0, 0, ..., 1, 1, 1])

[25]:
```python
plt.plot( dftsne.loc[ km.labels_==0,'x'], dftsne.loc[km.labels_==0,'y'],
         'o', alpha=0.05, markersize=1)
plt.plot( dftsne.loc[ km.labels_==1,'x'], dftsne.loc[km.labels_==1,'y'],
         'o', alpha=0.05, markersize=1)
```

[25]: [<matplotlib.lines.Line2D at 0x15b48716100>]


300
200
100
0
−100
−200
−200
−100
0
100
200
300