[ibovespa-stocks 브라질 주식 데이터 분석]

- [경제-주식]
- 브라질 증권 거래소의 주식 - Stocks from the Brazilian stock exchange
- 데이터 출처 : https://www.kaggle.com/datasets/felsal/ibovespa-stocks (https://www.kaggle.com/datasets/felsal/ibovespa-stocks)
- 데이터 분석 코드
  - github 코드 (https://github.com/LDJWJ/dataAnalysis/blob/main/01_08_ivospa_stock.ipynb)
  - HTML코드 (https://ldjwj.github.io/dataAnalysis/01_08_ivospa_stock.html)

대회 개요

- 1994년 2020년까지의 브라질 증권 B3증권 거래소에서 거래된 주식 정보
- 데이터 파일
  - b3_stocks_1994_2020.csv
  - selic.csv
  - usd2brl.csv

데이터 설명

- Input variables

```
datetime : 날짜, 시간
ticker   :
open     : 시작가
close    : 종가
high     : 최고가
low      : 최저가
volume   : 거래량
```

- Output variable

```
close_open : 종가 - 시작가
```

라이브러리 불러오기

In [1]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
```

데이터 불러오기

In [2]:

```
b3_stock = pd.read_csv("../dataset/ibovespa/b3_stocks_1994_2020.csv")
selic = pd.read_csv("../dataset/ibovespa/selic.csv")
usb2brl = pd.read_csv("../dataset/ibovespa/usd2brl.csv")

b3_stock.shape, selic.shape, usb2brl.shape
```

Out[2]:

```
((1883203, 7), (6651, 2), (6651, 2))
```

- 데이터 셋 : 180만개, 7열

In [3]:

```
print( b3_stock.columns )
print( selic.columns)
print( usb2brl.columns)
```

```
Index(['datetime', 'ticker', 'open', 'close', 'high', 'low', 'volume'], dtype='objec
t')
Index(['datetime', 'selic'], dtype='object')
Index(['datetime', 'usd_brl'], dtype='object')
```

In [4]:

```
b3_stock.head()
```

Out[4]:

	datetime	ticker	open	close	high	low	volume
0	1994-07-04	ACE 3	48.00	48.00	48.00	47.00	46550.0
1	1994-07-04	ALP 3	155.27	156.00	156.00	155.27	163405.8
2	1994-07-04	ALP 4	131.00	131.00	131.00	131.00	6550.0
3	1994-07-04	IBP 6	600.00	600.00	600.00	600.00	7800.0
4	1994-07-04	AQT 4	0.89	0.99	0.99	0.85	13137.0

In [5]:

```
selic.head()
```

Out[5]:

	datetime	selic
0	1994-07-04	0.003963
1	1994-07-05	0.003997
2	1994-07-06	0.003983
3	1994-07-07	0.003997
4	1994-07-08	0.003937

In [6]:

```
usb2brl.head()
```

Out[6]:

	datetime	usd_brl
0	1994-07-04	0.940
1	1994-07-05	0.932
2	1994-07-06	0.915
3	1994-07-07	0.910
4	1994-07-08	0.920

In [7]:

```
b3_stock.info()  # 180만개 데이터
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1883203 entries, 0 to 1883202
Data columns (total 7 columns):
 #   Column    Dtype  
---  ------    -----  
 0   datetime  object 
 1   ticker    object 
 2   open      float64
 3   close     float64
 4   high      float64
 5   low       float64
 6   volume    float64
dtypes: float64(5), object(2)
memory usage: 100.6+ MB
```

In [8]:

```
b3_stock.describe()
```

Out[8]:

	open	close	high	low	volume
count	1.883203e+06	1.883203e+06	1.883203e+06	1.883203e+06	1.883203e+06
mean	6.814208e+01	6.827674e+01	6.882414e+01	6.754027e+01	1.668286e+07
std	1.689781e+03	1.695070e+03	1.699966e+03	1.683604e+03	1.026344e+08
min	1.000000e-02	0.000000e+00	1.000000e-02	1.000000e-02	0.000000e+00
25%	3.960000e+00	3.960000e+00	4.000000e+00	3.890000e+00	1.800500e+04
50%	1.369000e+01	1.370000e+01	1.392000e+01	1.345000e+01	2.526560e+05
75%	3.670000e+01	3.673000e+01	3.714000e+01	3.608000e+01	4.794014e+06
max	1.297776e+06	1.297776e+06	1.297776e+06	1.297776e+06	4.298380e+10

In [9]:

```
selic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6651 entries, 0 to 6650
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   datetime  6651 non-null   object 
 1   selic     6651 non-null   float64
dtypes: float64(1), object(1)
memory usage: 104.0+ KB
```

In [10]:

```
usb2brl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6651 entries, 0 to 6650
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   datetime  6651 non-null   object 
 1   usd_brl   6651 non-null   float64
dtypes: float64(1), object(1)
memory usage: 104.0+ KB
```

과제 - 시작가에 비해 종가가 얼마가 달라지는가에 대한 예측 모델 만들기

In [11]:

```
b3_stock.head()
```

Out[11]:

	datetime	ticker	open	close	high	low	volume
0	1994-07-04	ACE 3	48.00	48.00	48.00	47.00	46550.0
1	1994-07-04	ALP 3	155.27	156.00	156.00	155.27	163405.8
2	1994-07-04	ALP 4	131.00	131.00	131.00	131.00	6550.0
3	1994-07-04	IBP 6	600.00	600.00	600.00	600.00	7800.0
4	1994-07-04	AQT 4	0.89	0.99	0.99	0.85	13137.0

In [12]:

```
# 새로운 컬럼 추가 : 종가 - 시작가
b3_stock['close_open'] = b3_stock['close'] - b3_stock['open']
b3_stock.head()
```

Out[12]:

	datetime	ticker	open	close	high	low	volume	close_open
0	1994-07-04	ACE 3	48.00	48.00	48.00	47.00	46550.0	0.00
1	1994-07-04	ALP 3	155.27	156.00	156.00	155.27	163405.8	0.73
2	1994-07-04	ALP 4	131.00	131.00	131.00	131.00	6550.0	0.00
3	1994-07-04	IBP 6	600.00	600.00	600.00	600.00	7800.0	0.00
4	1994-07-04	AQT 4	0.89	0.99	0.99	0.85	13137.0	0.10

In [13]:

```
### ticker은 몇개의 값이 존재할까?
len(b3_stock['ticker'].unique())
```

Out[13]:

```
3397
```

In [14]:

```
b3_stock.columns
```

Out[14]:

```
Index(['datetime', 'ticker', 'open', 'close', 'high', 'low', 'volume',
       'close_open'],
      dtype='object')
```

In [15]:

```
### target : close_open
### input : open, close, high, low, volume
sel = ['open', 'close', 'high', 'low', 'volume']

X_tr = b3_stock[sel]
y_tr = b3_stock['close_open']
```

In [16]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

X_train, X_test, y_train, y_test = train_test_split(X_tr, y_tr, test_size=0.3, random_state=0)

model1 = DecisionTreeRegressor()
model1.fit(X_train, y_train)

model2 = RandomForestRegressor(max_depth=3, n_estimators=5, random_state=0, n_jobs=-1)
model2.fit(X_train, y_train)
```

Out[16]:

```
RandomForestRegressor(max_depth=3, n_estimators=5, n_jobs=-1, random_state=0)
```

In [17]:

```
print("의사결정트리 score :", model1.score(X_train, y_train), model1.score(X_test, y_test) )
print("랜덤포레스트 score :", model2.score(X_train, y_train), model2.score(X_test, y_test) )
```

```
의사결정트리 score : 1.0 -0.2892431748648203
랜덤포레스트 score : 0.6806127222430476 -1.6176576427329867
```

최종 모델

In [18]:

```
model1 = RandomForestRegressor(max_depth=3, n_estimators=5, random_state=0, n_jobs=-1)
model1.fit(X_train, y_train)
pred = model1.predict(X_test)
```