Microstructure 변수를 이용한 자동매매 알고리즘 개발

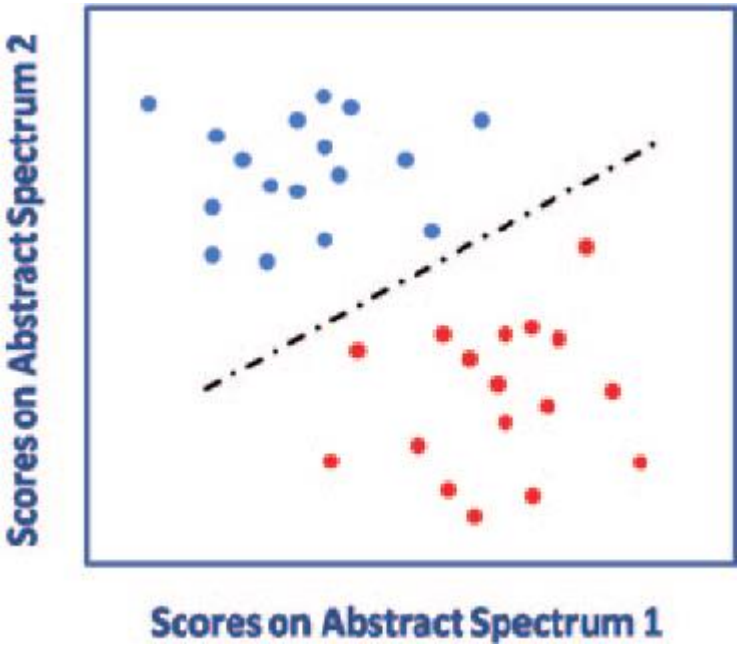
A Table of Contents.

Part 1

프로젝트 동기

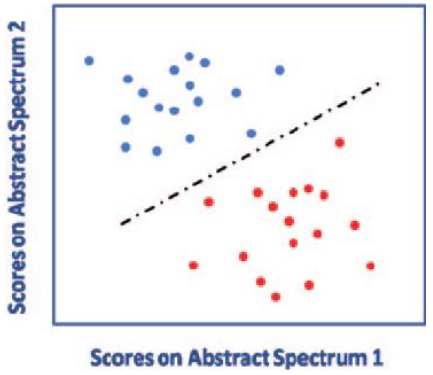
Part 1

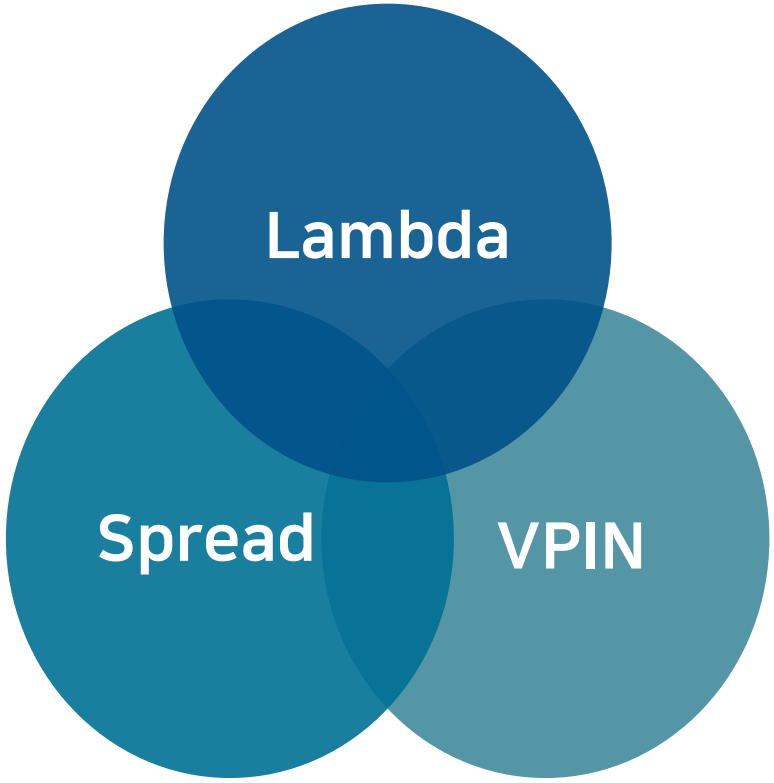
프로젝트 동기



Part 1

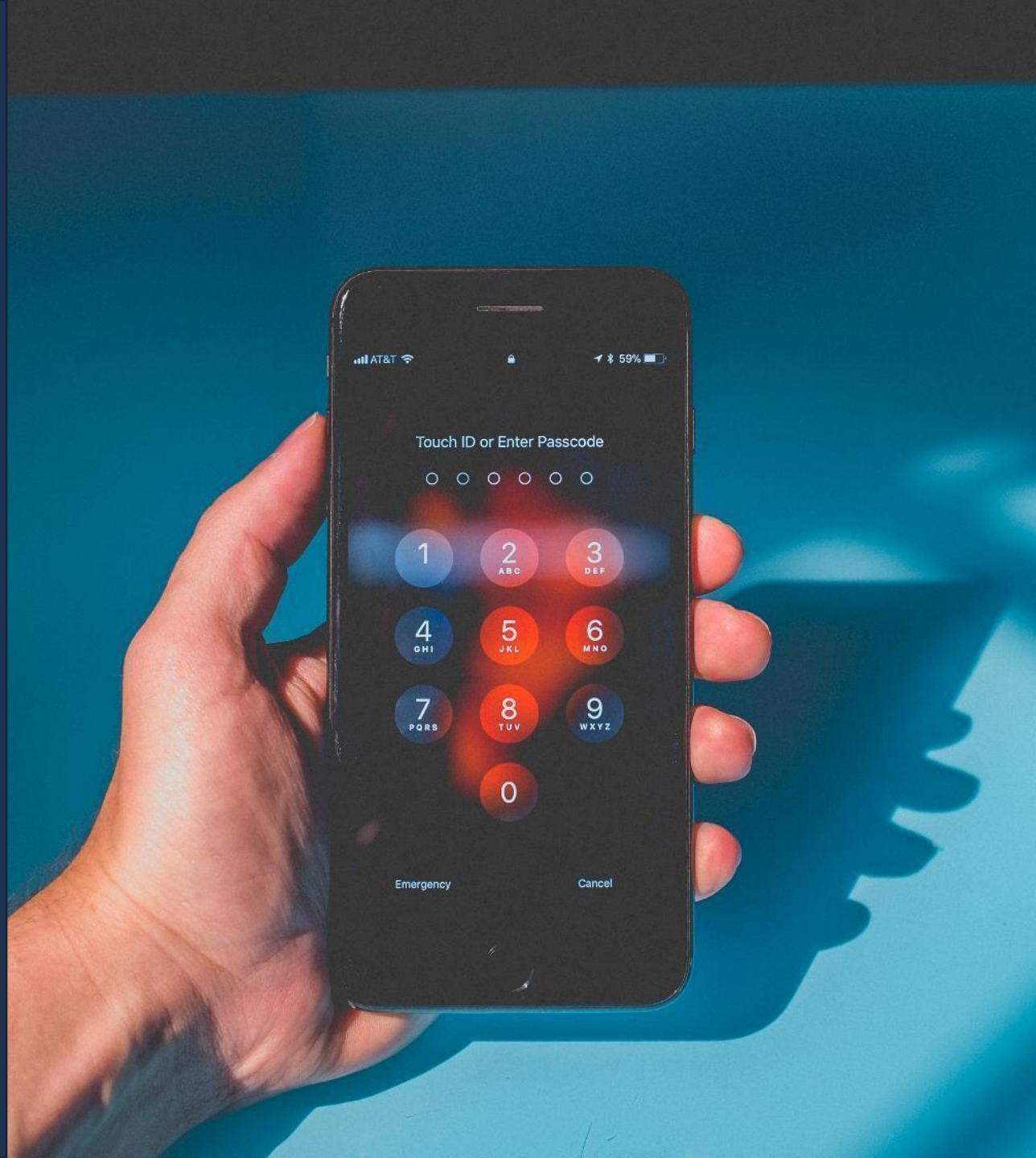
프로젝트 동기





Part 2

프로젝트 과정



Part 2

프로젝트 과정

STEP 1		STEP 2		STEP 3		STEP 4
데이터 준비 및 전처리	>>	볼린저 밴드를 이용한 거래 신호 생성 모델 구현 (primary model)	>>	거래 신호를 meta labeling 한 결과물을 분류하는 모델 구현 (secondary model)	>>	분류 모델을 이용한 Backtesting 수행

Part 2

프로젝트 과정

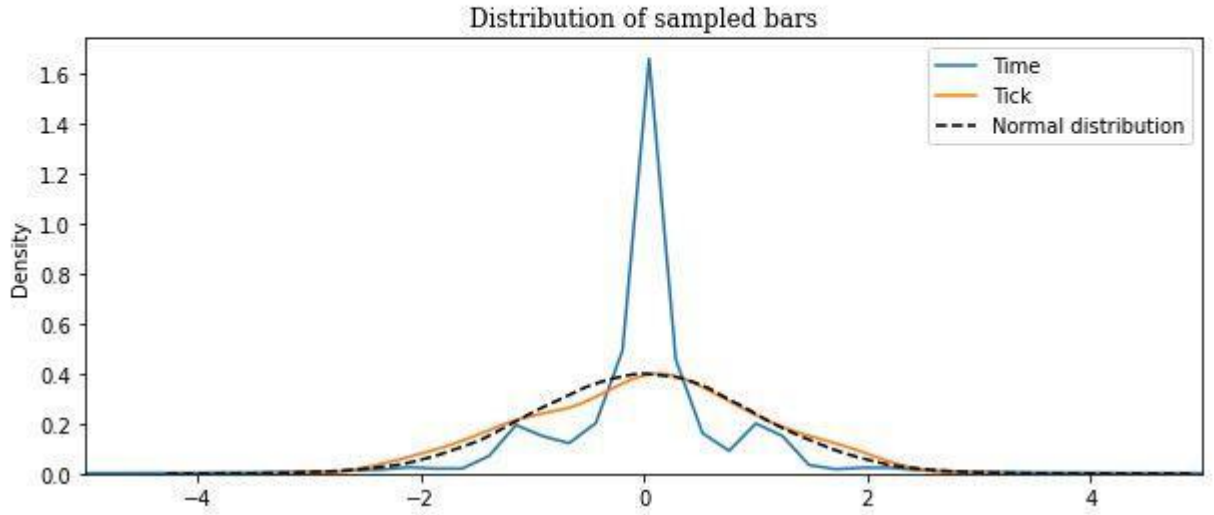
STEP 1

데이터 준비 및 전처리

프로젝트 과정

```
bar = StandardBarFeatures(file_path_or_df = df)
dollar = bar.dollar_bar(threshold =
int(np.mean(df.close*df.volume))*1000 )
```

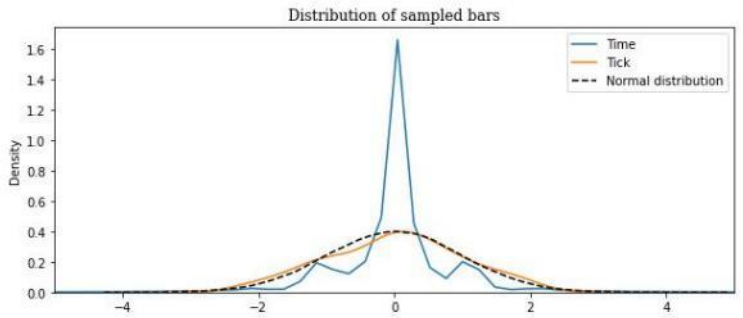
```
Reading data in batches:
Batch number: 0
Returning bars

0.035834167152643204
```



Part 2

프로젝트 과정

0.035834167152643204



	date	time	open	high	low	close	volume
0	20211229	1106	14800	14800	14800	14800	21
1	20211229	1403	14750	14750	14750	14750	3
2	20211229	1404	14700	14750	14700	14750	5
3	20211229	1405	14700	14750	14700	14700	1291
4	20211229	1406	14750	14750	14750	14750	4
...	...	...	...	...	...	...	...
136995	20231108	924	9810	9820	9810	9820	123
136996	20231108	923	9820	9820	9820	9820	8
136997	20231108	922	9830	9830	9830	9830	80
136998	20231108	941	9810	9810	9810	9810	241
136999	20231108	1029	9840	9840	9840	9840	10

137000 rows × 8 columns

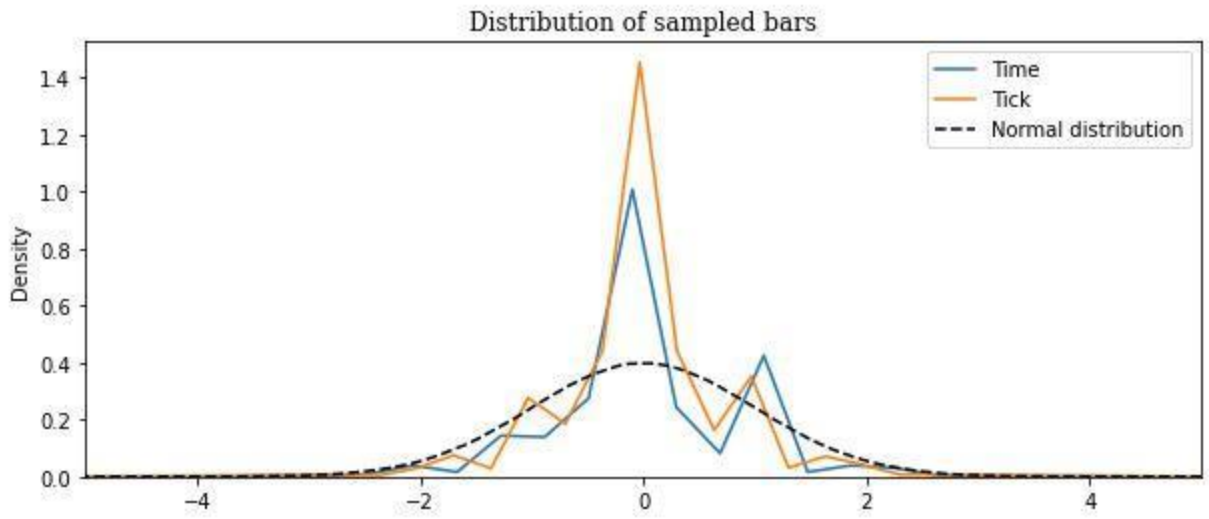
	date_time	tick_num	open	high	low	close	volume	cum_buy_volume	cum_ticks	cum_dollar_value
37	20220222	11937	1142.0	1530.0	901.0	1126.0	4172350	893400	327	4.954299e+09
38	20220223	12263	1124.0	1530.0	901.0	1143.0	4162650	956100	326	4.957347e+09
39	20220224	12600	1142.0	1530.0	901.0	1028.0	4196650	958450	337	4.953505e+09
40	20220228	12933	1047.0	1530.0	901.0	1325.0	4124650	1054600	333	4.950348e+09
41	20220228	13262	1347.0	1530.0	901.0	1107.0	4140100	831300	329	4.954124e+09
...	...	...	...	...	...	...	...	...	...	...
339	20231027	134918	1102.0	1530.0	901.0	1452.0	4071890	776190	430	4.953774e+09
340	20231030	135349	1451.0	1530.0	901.0	1149.0	4138830	1939670	431	4.951925e+09
341	20231101	135776	1151.0	1530.0	901.0	1330.0	4135500	900350	427	4.949941e+09
342	20231103	136217	1329.0	1530.0	901.0	1300.0	4235860	1894040	441	4.956160e+09
343	20231106	136654	1303.0	1530.0	901.0	1049.0	4243370	3574070	437	4.952413e+09

307 rows × 22 columns

Part 2

프로젝트 과정

```
tick=bar.volume_bar(threshold = 50000)
# np.mean(df.volume)=29087
```



프로젝트 과정

	date_time	tick_num	open	high	low	close	volume	cum_buy_volume	cum_ticks	cum_dollar_value
0	2017-04-24 09:03:00	2	41200.0	41200.0	41200.0	41200.0	82420	0	2	3.395704e+09
1	2017-04-24 09:05:00	4	41200.0	41200.0	41200.0	41200.0	82420	0	2	3.395704e+09
2	2017-04-24 09:07:00	6	41140.0	41140.0	41140.0	41140.0	82340	0	2	3.387468e+09
3	2017-04-24 09:09:00	8	41080.0	41080.0	41080.0	41080.0	82200	0	2	3.376776e+09
4	2017-04-24 09:11:00	10	41100.0	41120.0	41100.0	41120.0	82240	82240	2	3.380886e+09
...	...	...	...	...	...	...	...	...	...	...
118122	2019-05-15 15:13:00	188717	42700.0	42700.0	42650.0	42650.0	85450	0	2	3.646580e+09
118123	2019-05-15 15:15:00	188719	42650.0	42700.0	42650.0	42700.0	85400	42700	2	3.644445e+09
118124	2019-05-15 15:17:00	188721	42600.0	42700.0	42600.0	42700.0	85400	42700	2	3.642310e+09
118125	2019-05-15 15:19:00	188723	42650.0	42700.0	42650.0	42700.0	85400	42700	2	3.644445e+09
118126	2019-05-15 15:30:00	188725	42650.0	42650.0	42550.0	42550.0	85250	0	2	3.631658e+09

118127 rows × 10 columns

프로젝트 과정

```
tick['mfi']=money_flow_index(high = tick.high,
                             low = tick.low,
                             close = tick.close,
                             volume = tick.volume,
                             window = 20)

tick['obv']=on_balance_volume(close=tick.close,
                              volume=tick.volume,
                              )

tick['rsi']=rsi(close=tick.close, window=20)

tick['vwap']=volume_weighted_average_price(high=tick.high,
                                           low=tick.low,
                                           close=tick.close,
                                           volume=tick.volume,
                                           window=20)
tick['tsi']=tsi(close=tick.close,
                window_fast=13,
                window_slow=25)
```

프로젝트 과정

```
from FinancialMachineLearning.features.microstructure import *

corwin_schultz = CorwinSchultz(tick['high'], tick['low'])
spread = corwin_schultz.corwin_schultz_estimator(window = 20)
```

```
lambda_feature = BarbasedLambda(close = tick['close'],
                                volume = tick['volume'],
                                dollar_volume = tick.close * tick.volume)
kyle_lambda = lambda_feature.kyle()
```

프로젝트 과정

```
from FinancialMachineLearning.barsampling.bar_feature import BarFeature

def buy_volume(df):
    tick_signs = tick_rule(df['price'])
    return (df['volume'] * (tick_signs > 0)).sum()

def sell_volume(df):
    tick_signs = tick_rule(df['price'])
    return (df['volume'] * (tick_signs < 0)).sum()

buy_volume_feature = BarFeature(name='buy_volume', function=buy_volume)
sell_volume_feature = BarFeature(name='sell_volume', function=sell_volume)
bars = vpin_volume_bars('./tick_bar.csv', additional_features = [buy_volume_feature, sell_volume_feature])
vol_thres = 10000
vpin_series = vpin(bars['volume'], bars['buy_volume'], window = 5)
```

프로젝트 과정

STEP 2

볼린저 밴드를 이용한
거래 신호 생성 모델 구현
(primary model)

프로젝트 과정

```
bollinger_result=BollingerBands(close = tick["close"], window = 20, window_dev = 1.2)

tick['boll_hband']=np.array(bollinger_result.bollinger_hband_indicator()).astype(int)
tick['boll_lband']=np.array(bollinger_result.bollinger_lband_indicator())

label=[]

for indexs in tick.index:
    if tick.loc[indexs, 'boll_hband']==1:
        label.append(1)
    elif tick.loc[indexs, 'boll_lband']==1:
        label.append(-1)
    else:
        label.append(0)
```

프로젝트 과정

```
bollinger_result=BollingerBands(close = tick["close"], window = 20, window_dev = 2)
```

	precision	recall	f1-score	support
0	0.94	0.99	0.96	21066
1	0.82	0.51	0.63	2552
accuracy			0.93	23618
macro avg	0.88	0.75	0.80	23618
weighted avg	0.93	0.93	0.93	23618

프로젝트 과정

```
bollinger_result=BollingerBands(close = tick["close"], window = 20, window_dev = 0.5)
```

	precision	recall	f1-score	support
0	0.54	0.30	0.39	6038
1	0.79	0.91	0.85	17566
accuracy			0.76	23604
macro avg	0.67	0.61	0.62	23604
weighted avg	0.73	0.76	0.73	23604

프로젝트 과정

```
bollinger_result=BollingerBands(close = tick["close"], window = 20, window_dev = 1.2)
```

	precision	recall	f1-score	support
0	0.84	0.91	0.87	14739
1	0.82	0.71	0.76	8879
accuracy			0.83	23618
macro avg	0.83	0.81	0.82	23618
weighted avg	0.83	0.83	0.83	23618

Part 2

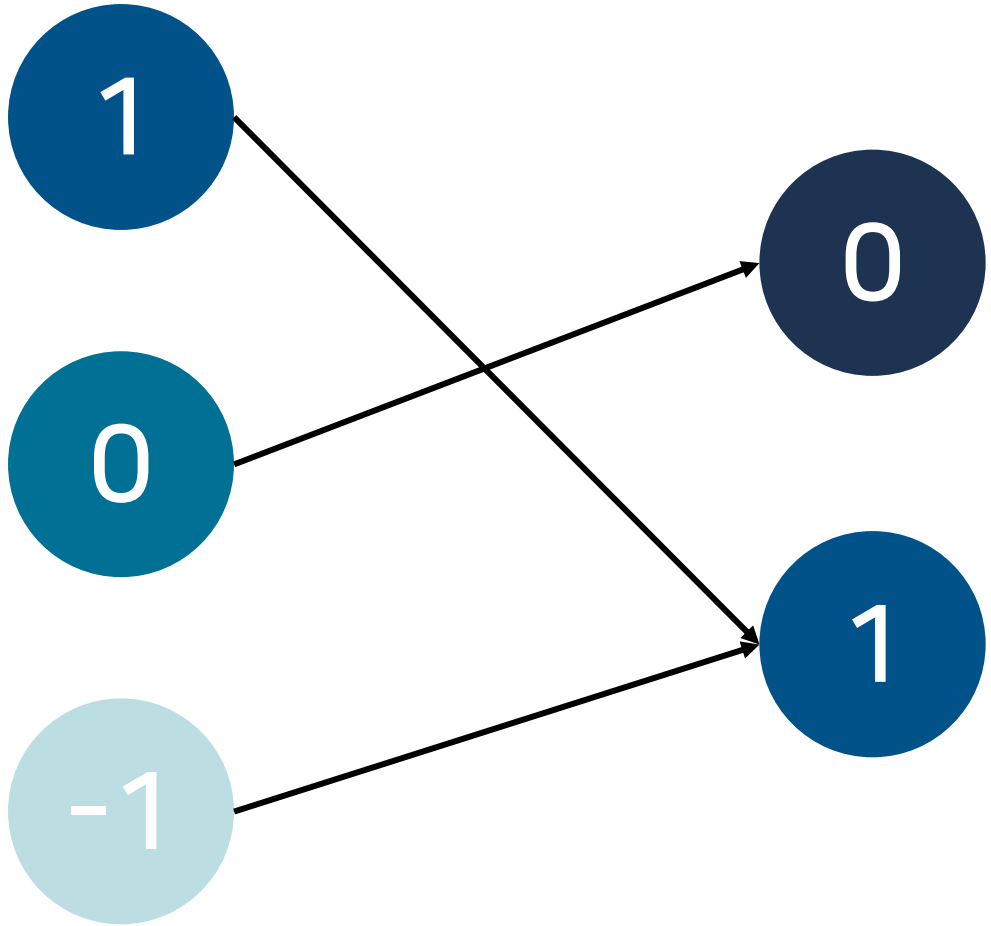
프로젝트 과정

STEP 3

거래 신호를
meta labeling 한 결과물을
분류하는 모델 구현
(secondary model)

프로젝트 과정

```
# meta labling
meta=[]
for data in tick.label:
    if data==1:
        meta.append(1)
    elif data==-1:
        meta.append(1)
    else:
        meta.append(0)
tick['meta']=meta
```



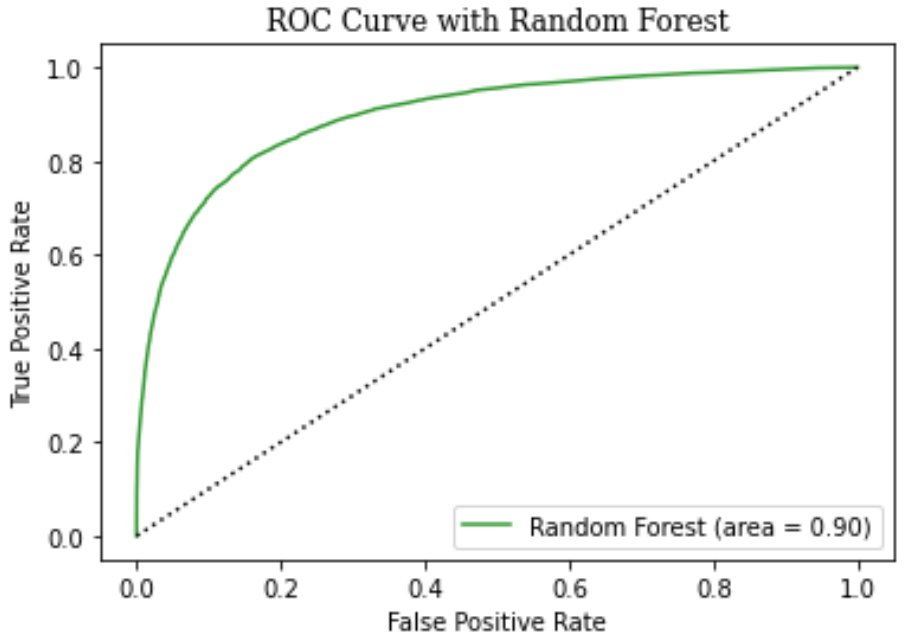
프로젝트 과정

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import roc_auc_score, classification_report, roc_curve
from sklearn.metrics import auc

tick=tick.dropna()

X=tick[['spread', 'lambda', 'mfi', 'obv', 'rsi', 'vwap', 'tsi', 'volume', 'vpin']]
y=tick['meta']
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2, shuffle=False)

tree = RandomForestClassifier(
        criterion='entropy', class_weight='balanced_subsample', min_weight_fraction_leaf=0.0, n_estimators=500,
        max_features=1.0, oob_score=True, n_jobs=-1)
tree.fit(X_train, y_train)

y_pred_tree = tree.predict(X_test)
```

Part 3

프로젝트 결과

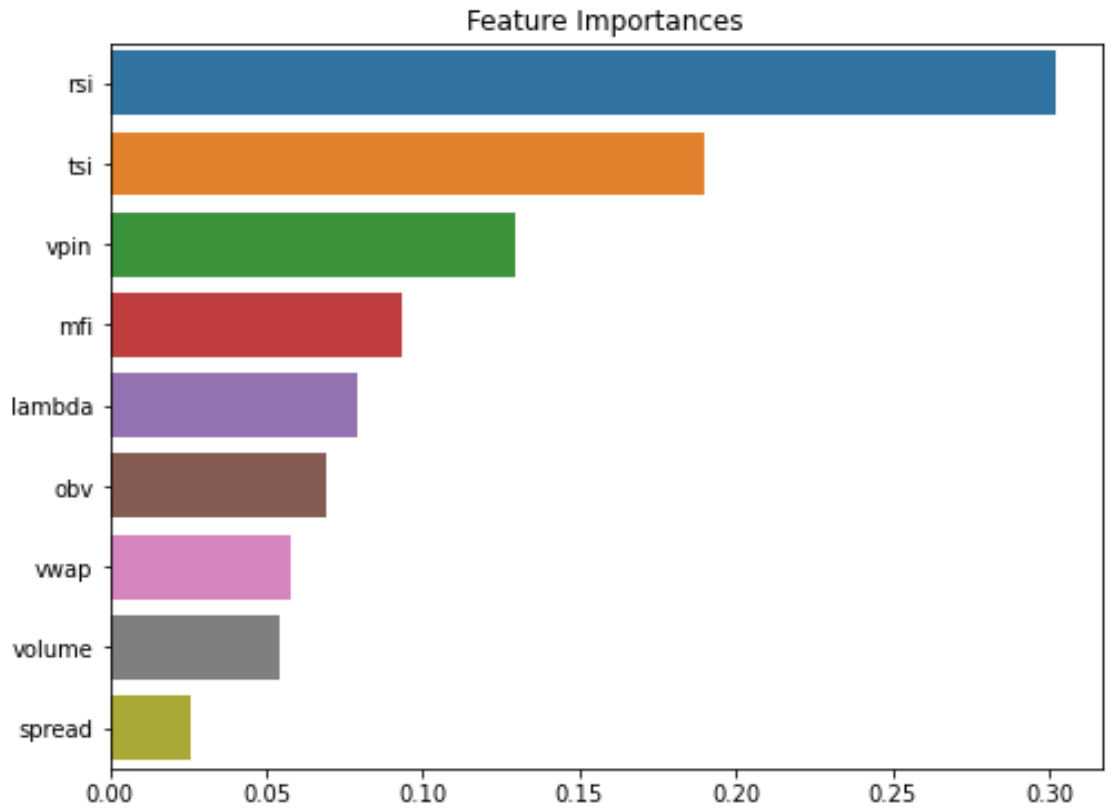
프로젝트 결과



	precision	recall	f1-score	support
0	0.84	0.91	0.87	14739
1	0.82	0.71	0.76	8879
accuracy			0.83	23618
macro avg	0.83	0.81	0.82	23618
weighted avg	0.83	0.83	0.83	23618

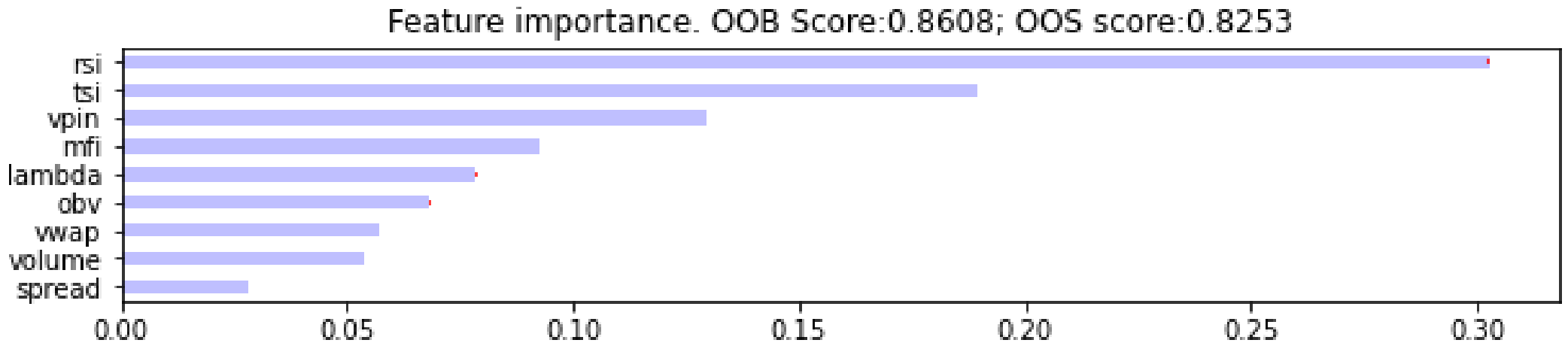
프로젝트 결과

```
ser = pd.Series(tree.feature_importances_, index=feature_name)
```



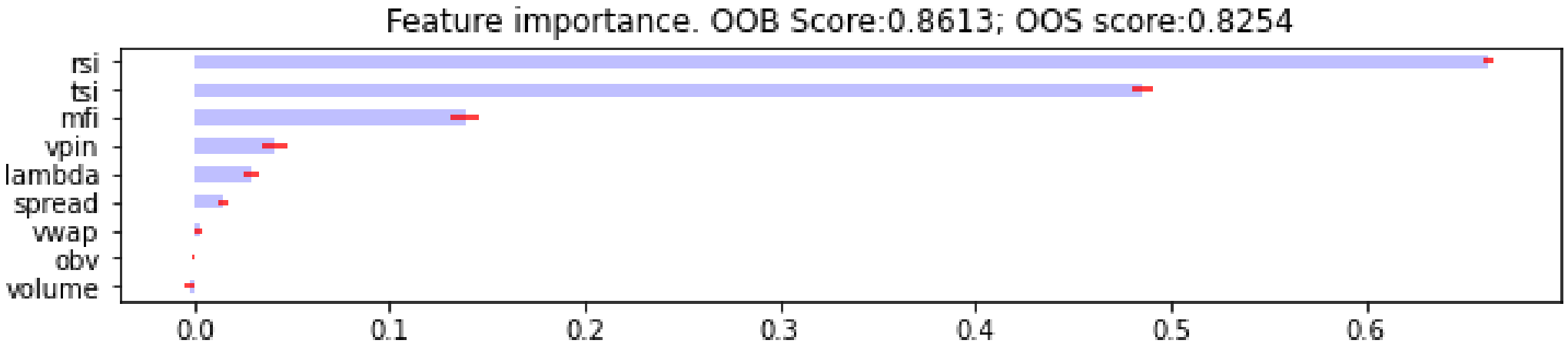
프로젝트 결과

```
test_data_func(X, tick, methods='MDI')
```



프로젝트 결과

```
test_data_func(X, tick, methods='MDA')
```



프로젝트 결과

```
start=datetime.datetime(2008,1,1)
end=datetime.datetime(2012,12,31)
data=pdr.DataReader("005930", "naver", start, end).astype("float")
```

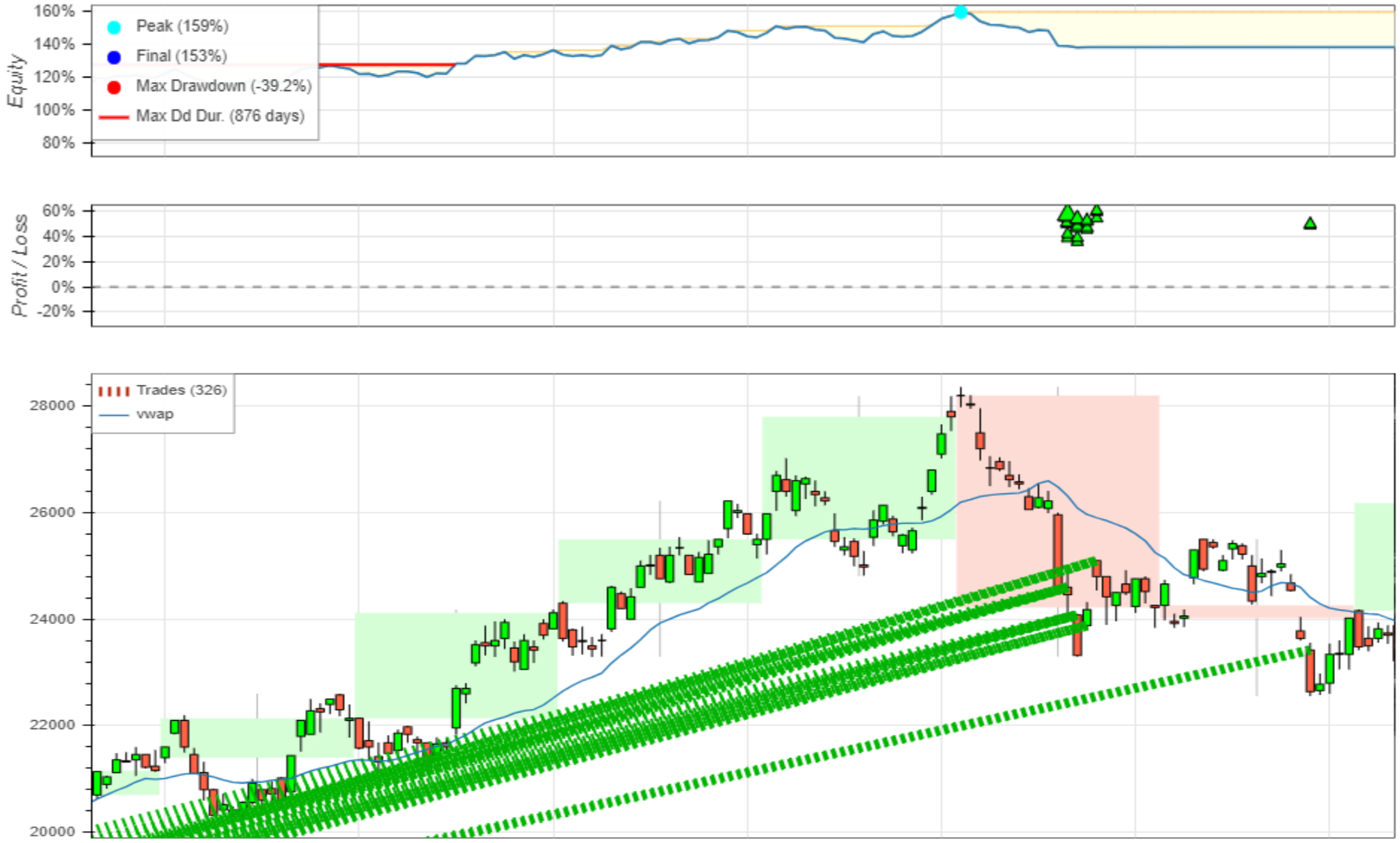
```
def bollinger_hband(df):
    bollinger_result=BollingerBands(close = pd.Series(df.Close), window = 20, window_dev = 1)
    boll_hband=np.array(bollinger_result.bollinger_hband_indicator()).astype(int)
    label=[]
    for data in boll_hband:
        if data==1:
            label.append(1)
        else:
            label.append(0)
    return label

def bollinger_lband(df):
    bollinger_result=BollingerBands(close = pd.Series(df.Close), window = 20, window_dev = 1)
    boll_lband=np.array(bollinger_result.bollinger_lband_indicator()).astype(int)
    label=[]
    for data in boll_lband:
        if data==1:
            label.append(-1)
        else:
            label.append(0)
    return label

def spread(high, low):
    corwin_schultz = CorwinSchultz(pd.Series(high), pd.Series(low))
    spread = corwin_schultz.corwin_schultz_estimator(window = 20)
    spread=spread.fillna(0)
    return spread

def lambda_data(close, volume):
    lambda_feature = BarbasedLambda(close = pd.Series(close),
                                    volume = pd.Series(volume),
                                    dollar_volume = tick.close * tick.volume)
    kyle_lambda = lambda_feature.kyle()
    kyle_lambda=kyle_lambda.fillna(0)
    return kyle_lambda
```

```
def model_predict(spread, kyle_lambda, mfi, obv, rsi_return, vwap, tsi_return, volume):
    data=pd.DataFrame({'spread': spread, 'kyle':kyle_lambda, 'mfi':mfi, 'obv' : obv, 'rsi':rsi_return, 'vwap':vwap, 'tsi':tsi_return, 'volum
    data=data.fillna(0)
    y_pred_tree = tree.predict(data)
    return y_pred_tree

def model_prob(spread, kyle_lambda, mfi, obv, rsi_return, vwap, tsi_return, volume):
    data=pd.DataFrame({'spread': spread, 'kyle':kyle_lambda, 'mfi':mfi, 'obv' : obv, 'rsi':rsi_return, 'vwap':vwap, 'tsi':tsi_return, 'volum
    data=data.fillna(0)
    y_pred_prob = tree.predict_proba(data)[:,1]
    return y_pred_prob

def mfi(data):
    return money_flow_index(high = pd.Series(data.High),
                            low = pd.Series(data.Low),
                            close = pd.Series(data.Close),
                            volume = pd.Series(data.Volume),
                            window = 20)

def obv(data):
    return on_balance_volume(close=pd.Series(data.Close),
                             volume=pd.Series(data.Volume))

def rsi_return(data):
    return rsi(close=pd.Series(data.Close), window=20)


def vwap(data):
    return volume_weighted_average_price(high=pd.Series(data.High),
                                         low=pd.Series(data.Low),
                                         close=pd.Series(data.Close),
                                         volume=pd.Series(data.Volume),
                                         window=20)

def tsi_return(data):
    return tsi(close=pd.Series(data.Close),
               window_fast=13,
               window_slow=25)
```

```
class my_strategy(Strategy):

    def init(self):
        self.bollinger_h = self.I(bollinger_hband, self.data)
        self.bollinger_l = self.I(bollinger_lband, self.data)
        self.bollinger=self.bollinger_h+self.bollinger_l
        self.spread=self.I(spread, self.data.High, self.data.Low)
        self.lambda_lst=self.I(lambda_data, self.data.Close, self.data.Volume)
        self.mfi=self.I(mfi, self.data)
        self.obv=self.I(obv, self.data)
        self.rsi=self.I(rsi_return, self.data)
        self.vwap=self.I(vwap, self.data)
        self.tsi=self.I(tsi_return, self.data)
        self.volume=self.data.Volume
        self.pred=self.I(model_predict, self.spread, self.lambda_lst, self.mfi, self.obv, self.rsi
        self.prob=self.I(model_prob, self.spread, self.lambda_lst, self.mfi, self.obv, self.rsi, s
        self.result=self.bollinger*self.pred


    def next(self):
        if self.result==1:
            self.buy( size=(self.prob[-1]) )

        elif self.result==-1:
            self.position.close( portion=(self.prob[-1]) )

bt = Backtest(data, my_strategy, cash=10000000, commission=.002)
stats = bt.run()
bt.plot()
```

Part 3

프로젝트 결과




Q&A