نوید ابراهیمی ۹۹۵۲۱۰۰۱

Get Datas

```
[ ]  cryptos = ['BTC-USD', 'ETH-USD']

     start_date = datetime(2021,7,1, tzinfo=timezone.utc)
     end_date = datetime(2023,12,1, tzinfo=timezone.utc)

     train_start_date = datetime(2021,7,1, tzinfo=timezone.utc)
     train_end_date = datetime(2023,7,1, tzinfo=timezone.utc)

     test_start_date = datetime(2023,7,1, tzinfo=timezone.utc)
     test_end_date = datetime(2023,12,1, tzinfo=timezone.utc)
```

```
[ ]  prices = {}
     returns = {}

     for crypto in cryptos:
         ticker = yf.Ticker(crypto)
         prices_1d = ticker.history(start=start_date, end=end_date, interval='1d').Close
         returns_1d = prices_1d.pct_change().dropna()

         if crypto not in prices:
             prices[crypto] = []
             returns[crypto] = []

         prices[crypto].append(prices_1d)
         returns[crypto].append(returns_1d)
```

در این قسمت تاریخ‌هایی که قرار است موردارزیابی قرار دهیم را دریافت میکنیم. همانطور که در صورت سوال آمده بازه train از 1-7-2021 تا 1-7-2023 و بازه test از 1-7-2023 تا 1-12-2023 است.

```
train_data_price = []
test_data_price = []

train_data_return = []
test_data_return = []

data_price = []

for i, crypto in enumerate(cryptos):
  data_price.append(returns[crypto][0].loc[start_date:end_date])

  train_data_price.append(returns[crypto][0].loc[train_start_date:train_end_date]) # train_data[0] => BTC, train_data[1] => ETH
  test_data_price.append(returns[crypto][0].loc[test_start_date + timedelta(days=1):test_end_date]) # test_data[0] => BTC, test_data[1] => ETH

  train_data_return.append(returns[crypto][0].loc[train_start_date:train_end_date]) # train_data[0] => BTC, train_data[1] => ETH
  test_data_return.append(returns[crypto][0].loc[test_start_date + timedelta(days=1):test_end_date]) # test_data[0] => BTC, test_data[1] => ETH

print(train_data_price[0])
```

در این قسمت داده‌هایی که قرار است موردارزیابی قرار دهیم را دریافت میکنیم. Data_price تمامی داده‌های train و test است که دلیل گرفتن این دیتا برای قسمت پیش‌بینی و arch است.

Train Model with Brute Force

```
windows = {}
test_size = len(train_data_price[0])
residuals = []

def best_arima_model(i, j):
  train_data_R = data_price[i][j:j+test_size]
  windows[f'{cryptos[i]}, {j}'] = []
  best_p = 0
  best_q = 0
  best_aic = 99999999
  best_model = None
  for p in range(1,6):
    for q in range(1,6):
      try:
          model = ARIMA(train_data_R, order=(p,0,q))
          model_fit = model.fit()
          if best_aic > model_fit.aic:
            best_p = p
            best_q = q
            best_model = model_fit
            best_aic = model_fit.aic

      except:
          pass

  residual = best_model.resid
  residuals.append(residual)
  return best_p, best_q, residual
```

در اینجا arima را پیاده کردم. به این صورت که برای هر هر بازه rolling window تمامی حالات p , q های بین ۱ تا ۵ را امتحان کردم و بهترین مدل، p، q را ذخیره کردم. در انتها p,q,res را برگرداندم.

در سوال گفته شده انتخاب p q براساس aic باشد که دلیل وجود داخلی‌ترین if همین است.

Arch Model

```
from arch import arch_model

signals_all = []
means = []

for i in range(len(train_data_price)):
  means.append([])
  for j in range(len(test_data_price[0])):
    p_val, q_val, residual = best_arima_model(i, j)
    model = arch_model(residual, vol='Garch', p=p_val, q=q_val)
    arch_model_fit = model.fit()
    arch_model_forecast = arch_model_fit.forecast(horizon=1)
    arch_mean = arch_model_forecast.mean
    means[i].append(arch_mean.iloc[-1])
```

در این قسمت arch را پیاده کردم. Rolling window را به تعداد بازه test لغزاندم. For بیرونی برای محاسبه این مقدار برای دو crypto است. در ابتدا بهترین p q res را دریافت میکنیم و در ادامه مدل arch را با نوسان garch محاسبه کردیم. حال با استفاده از خط

```
arch_model_forecast = arch_model_fit.forecast(horizon=1)
```

قیمت یک روز آینده را همانطور که صورت سوال گفته است پیش‌بینی میکنیم. در ادامه mean را به دست میاوریم. دلیل این کار این است که در صورت سوال گفته براساس این میانگین سیگنال خرید یا فروش دریافت کنیم.

Get Signals from Means

```
signals = []
for i in range(len(means)):
  signals.append([])
  for j in range(len(means[i])):
    if means[i][j][0] > 0:
      signals[i].append(1)
    elif means[i][j][0] < 0:
      signals[i].append(-1)
    else:
      signals[i].append(0)
len(signals[0])
```

در اینجا همانطور که بالا گفتم اگر میانگین بزرگ‌تر از صفر بود سیگنال خرید دریافت میکنیم، اگر کوچک‌تر از صفر بود سیگنال فروش و اگر صفر بود هیچ کاری نمیکنیم.

Write Data in CSV file

```
[ ]  with open('output.csv', 'w', newline='') as file:
         writer = csv.writer(file)
         for item in signals:
             print(item)
             writer.writerow(item)
```

در اینجا این سیگنال‌ها را در فایل CSV ذخیره میکنیم.

Backtest

```
start_val = 100
history_BTC = {'Crypto' : 'BTC', 'Position' : [], 'Total' : [],'Balance' : []}
history_ETH = {'Crypto' : 'ETH', 'Position' : [], 'Total' : [],'Balance' : []}
total = 0
for i in range(len(train_data_price)):
  for j in range(len(means[i])):
      if signals[i][j] == 1:
        number_of_shares_consider_money = start_val / test_data_price[i][j]
        number_of_buy_share = number_of_shares_consider_money / 2
        total += number_of_buy_share
        start_val -= number_of_buy_share * test_data_price[i][j]
        if i == 0:
          history_BTC['Position'].append('BUY')
          history_BTC['Total'].append(total)
          history_BTC['Balance'].append(start_val)
        else:
          # print(start_val)
          history_ETH['Position'].append('BUY')
          history_ETH['Total'].append(total)
          history_ETH['Balance'].append(start_val)

      elif signals[i][j] == -1:
        start_val += total * test_data_price[i][j]
        total = 0
        if i == 0:
          # print(start_val)
          history_BTC['Position'].append('SELL')
          history_BTC['Total'].append(total)
          history_BTC['Balance'].append(start_val)
        else:
          history_ETH['Position'].append('SELL')
          history_ETH['Total'].append(total)
          history_ETH['Balance'].append(start_val)
      else:
          pass
  start_val = 100


# for key, value in history_BTC.items():
#   print(f"{key}: {value}")
print(pd.DataFrame(history_ETH))
print(pd.DataFrame(history_BTC))
```

در اینجا به ازای هر سیگنالی که دریافت کردیم معامله را انجام میدهیم. به این صورت که اگر سیگنال خرید (۱) داشتیم، در ابتدا تعداد سهامی که به پولمان میتوان بخریم محاسبه میکنیم. در ادامه بنا به استراتژی‌مان نصف این سهام را میخریم و این تعداد را به تعداد سهاممان اضافه و قیمت آن را از بودجه‌مان کم میکنیم. آن دو if هم برای تفکیک BTC و ETH است.

سیگنال فروش نیز ۱- است که کل سهام را میفروشیم و قیمت آن را به بودجه‌مان اضافه میکنیم. در ادامه هرکدام از داده‌ها را نیز ذخیره میکنیم.

Implement Sharpe Ratio

```
def sharpe_ratio_func(history):
  risk_free_rate = 0.02
  excess_returns = pd.DataFrame(history['Balance']).pct_change().dropna()
  annualized_std = excess_returns.std() * np.sqrt(365)
  sharpe_ratio = (excess_returns.mean() - risk_free_rate) / annualized_std
  return sharpe_ratio[0]

print("BTC Sharpe Ratio:", sharpe_ratio_func(history_BTC))
print("ETH Sharpe Ratio:", sharpe_ratio_func(history_ETH))
```

```
BTC Sharpe Ratio: -1061566775349615.4
ETH Sharpe Ratio: 0.004259563472486445
```

در اینجا تابع Sharpe Ratio را پیاده کردم. این پیاده‌سازی براساس فرمول Sharpe Ratio است.

Buy and Hold

```
balance = 100
balance_res = []

for i in range(len(test_data_price)):
  cumulative_returns = test_data_price[i].add(1).cumprod()
  balance *= cumulative_returns
  balance_res.append(balance)
  balance = 100

# print(balance_res)
backtest_BTC = balance_res[0].reset_index()
backtest_BTC.columns = ['Date', 'Balance']

backtest_ETH = balance_res[1].reset_index()
backtest_ETH.columns = ['Date', 'Balance']

print(backtest_BTC)
print(backtest_ETH)
```

```
                       Date     Balance
0   2023-07-02 00:00:00+00:00  100.100331
1   2023-07-03 00:00:00+00:00  101.851454
2   2023-07-04 00:00:00+00:00  100.612957
3   2023-07-05 00:00:00+00:00   99.751841
4   2023-07-06 00:00:00+00:00   97.774637
..                        ...         ...
147 2023-11-26 00:00:00+00:00  122.520514
148 2023-11-27 00:00:00+00:00  121.785135
149 2023-11-28 00:00:00+00:00  123.671099
150 2023-11-29 00:00:00+00:00  123.760691
151 2023-11-30 00:00:00+00:00  123.284242

[152 rows x 2 columns]
                       Date     Balance
0   2023-07-02 00:00:00+00:00  100.668849
1   2023-07-03 00:00:00+00:00  101.601569
2   2023-07-04 00:00:00+00:00  100.627031
3   2023-07-05 00:00:00+00:00   99.273711
4   2023-07-06 00:00:00+00:00   96.054724
..                        ...         ...
147 2023-11-26 00:00:00+00:00  107.207870
148 2023-11-27 00:00:00+00:00  105.344137
149 2023-11-28 00:00:00+00:00  106.483136
150 2023-11-29 00:00:00+00:00  105.474652
```

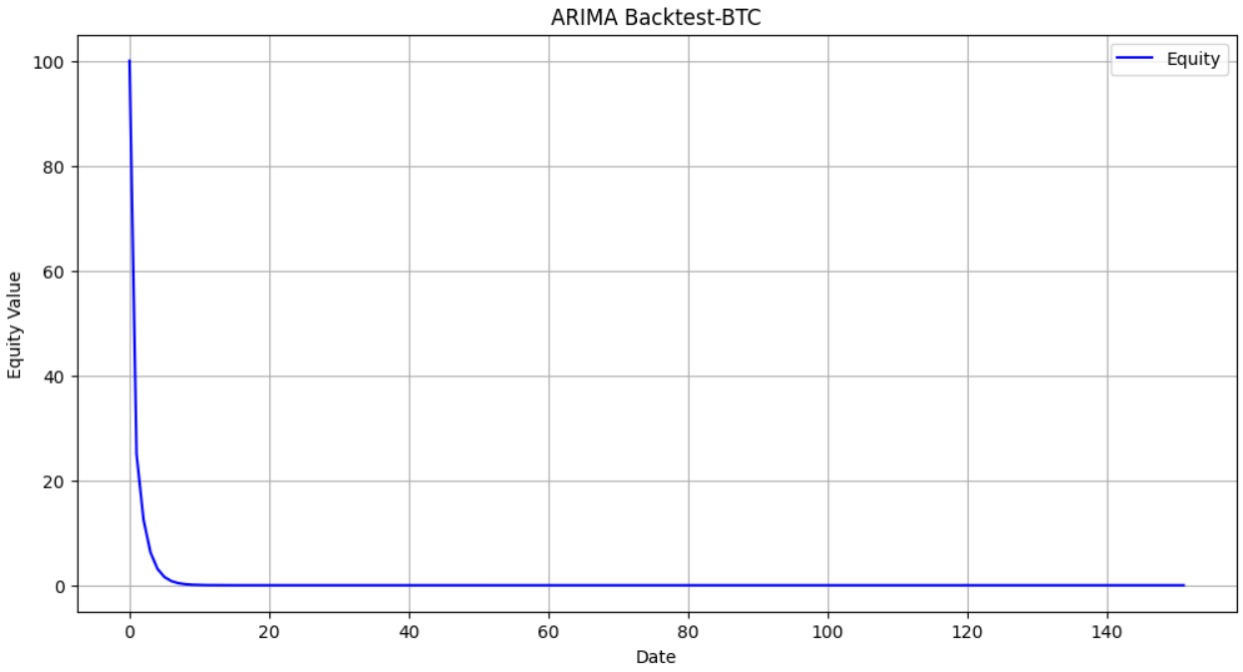
در اینجا استراتژی Buy and Hold را نوشتم. برای نوشتن این کد از

https://www.youtube.com/watch?v=00ejyn7eiw0

استفاده کردم.

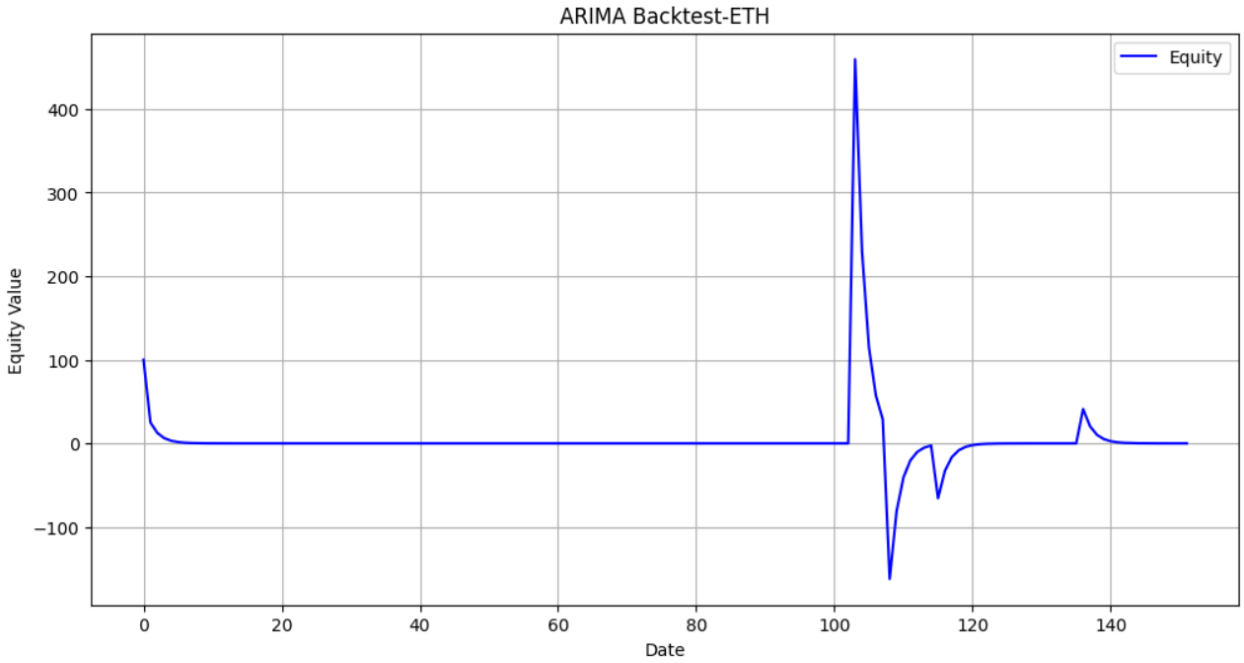
```python
def equity_computation(backtest, start_val):
    equity = [start_val]
    for i in range(1, len(backtest)):
        equity.append(backtest['Balance'].iloc[i])
    return equity
```

در اینجا نیز Equity Function را محاسبه کردم. مقدار balance در هر لحظه مقدار Equity است.



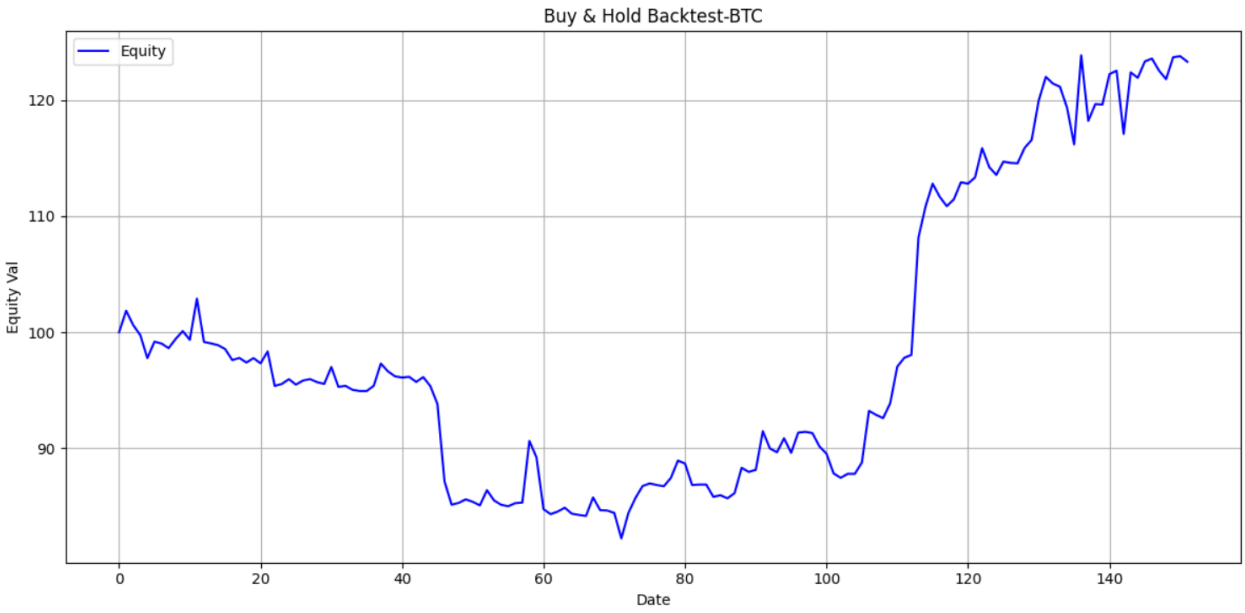
نمودار equity_values_arima_BTC

این نشان می‌دهد که پیش‌بینی‌های مدل ARIMA برای بیت‌کوین در این Backtest خوب عمل نکرده است، زیرا ارزش سهام به‌طور قابل‌توجهی کاهش یافته است. بک تست روشی برای ارزیابی عملکرد یک استراتژی یا مدل معاملاتی با استفاده از داده های تاریخی است و نتایج می تواند بر اساس دوره زمانی مورد استفاده برای آزمایش متفاوت باشد. در نظر گرفتن هزینه های تراکنش، لغزش و سایر عوامل هنگام ارزیابی نتایج بک تست مهم است.



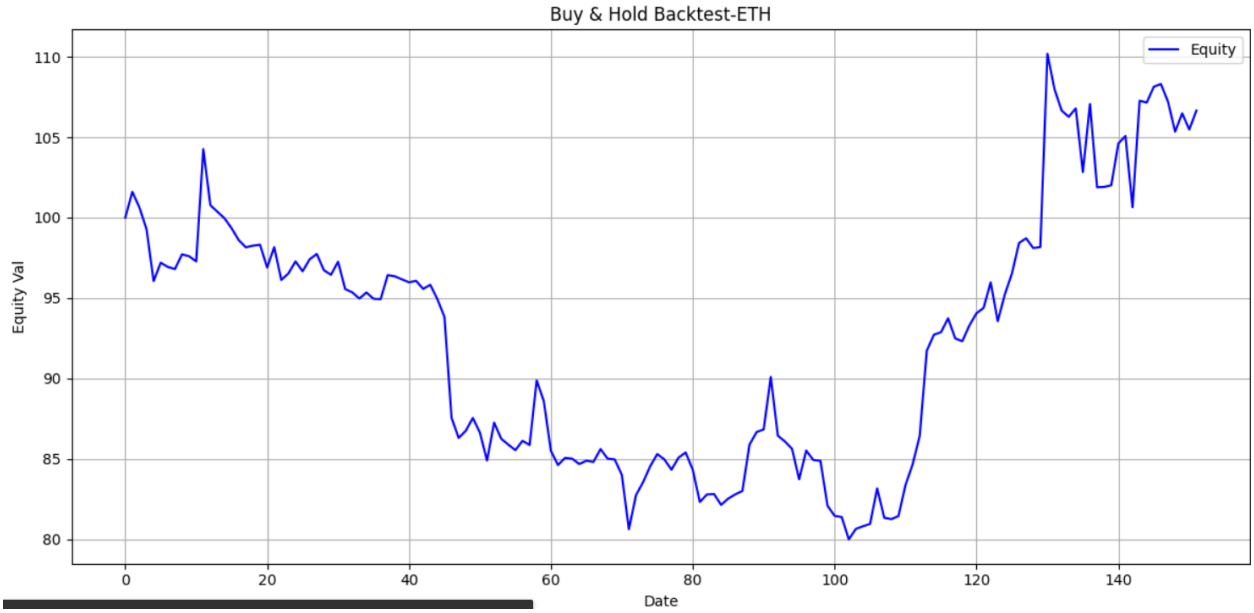
نمودار equity_values_arima_ETH

این نشان می‌دهد که پیش‌بینی‌های مدل ARIMA برای اتریوم در یک برهه زمانی خاص افزایش قابل‌توجهی در ارزش سهام داشته است، اما پس از آن دوباره کاهش یافت. این می تواند به دلیل عوامل مختلفی مانند نوسانات بازار یا حساسیت مدل به نقاط داده خاص باشد. مانند مدل بیت کوین، در نظر گرفتن هزینه های تراکنش، لغزش و سایر عوامل هنگام ارزیابی نتایج بک تست مهم است. همچنین بسیار مهم است که به یاد داشته باشید که عملکرد گذشته نشان دهنده نتایج آینده نیست.



نمودار buy_and_hold_equity_values_BTC

این دیاگرام نشان می دهد که استراتژی خرید و نگه داشتن بیت کوین در ابتدا با کاهش ارزش مواجه شد اما سپس افزایش قابل توجهی داشت. استراتژی خرید و نگهداری شامل خرید یک اوراق بهادار و نگهداری آن برای طولانی مدت بدون توجه به نوسانات در بازار است. سپس عملکرد این استراتژی با محاسبه تراز نهایی ارزیابی می شود. عملکرد گذشته نشان دهنده نتایج آینده نیست. همچنین در نظر گرفتن هزینه‌های تراکنش، لغزش و سایر عوامل هنگام ارزیابی نتایج Backtest بسیار مهم است.



نمودار buy_and_hold_equity_values_ETH

در ابتدا، کاهشی در ارزش سهام وجود دارد که به کمترین حد خود در حدود تاریخ ۶۰ رسیده است.

پس از تاریخ ۸۰، یک افزایش شدید در ارزش حقوق صاحبان سهام وجود دارد که به اوج یعنی خود کمی بالاتر از ۱۰۵ می رسد قبل از اینکه دوباره شروع به نوسان کند.