امیرعلی فرازمند

۹۹۵۲۲۳۲۹

تکلیف چهارم

برای **بیتکوین**:

چند تابع ساده ای که در ابتدای کار تعریف کرده ایم و بعد ها به کار می آیند:

```python
# Function to collect historical data for a cryptocurrency using yfinance
def fetch_crypto_data(symbol, start_date, end_date, priod):
    data = yf.Ticker(symbol)

    tickerDf = data.history(period=priod, start=start_date, end=end_date)
    tickerDf = tickerDf[['Close']]
    # print(tickerDf)

    return tickerDf

def get_aic(train_data, p, q):
    model = ARIMA(train_data['Close'], order=(p, 0, q), enforce_stationarity=False, enforce_invertibility=False)
    model_fit = model.fit()
    aic = model_fit.aic
    return aic

def find_best_arima(train_data, max_loop=6):
    best_aic = float('inf')
    best_combination = {'p': 111, 'q': 111}

    for p in range(1, max_loop):
        for q in range(1, max_loop):
            temp_aic = get_aic(train_data, p, q)
            if temp_aic < best_aic:
                best_aic = temp_aic
                best_combination['p'] = p
                best_combination['q'] = q

    return (best_combination['p'], best_combination['q'])

```

0.0s Python

گرفتن دیتا:

```python

# Cofigurations
start_date = datetime(2021, 7, 1, tzinfo=timezone('Asia/Tehran'))
end_date = datetime(2023, 12, 1, tzinfo=timezone('Asia/Tehran'))


train_end = datetime (2023,6,30, tzinfo=timezone('Asia/Tehran'))
test_start =  datetime(2023,6,30, tzinfo=timezone('Asia/Tehran'))
test_end = datetime(2023,11,29, tzinfo=timezone('Asia/Tehran') )

```

0.0s Python

```python
# del data
data = fetch_crypto_data('ETH-USD', start_date, end_date, '1d')

# Initialize the 'prediction' column
data['prediction'] = np.nan
data['residual'] = np.nan

```

در لوپ روی بازه ی تست residual هر روز بر طبق مدل آریمایی که p,q بهینه اش پیدامیشود پیشبینی میشود. در هر روز ۳۶۵ روز قبلی را به مدل داده ایم. در جای دیگری با ۷۰۰ روز کار شده و نتایج آن در arima_700.csv قابل مشاهده است.

```python
# Convert the test_start and test_end to pandas Timestamps
test_start = pd.Timestamp(test_start)
test_end = pd.Timestamp(test_end)
# Create a date range for the test period
test_dates = pd.date_range(test_start, test_end)

# Loop over the dates in the test period
for date in test_dates:
    print('--------',date,'--------')
    # Select the data up to the current date
    train_data = data[data.index <= date][-365:]

    # Find the best ARIMA model for this data
    p, q = find_best_arima(train_data)

    # Fit the ARIMA model
    model = ARIMA(train_data['Close'], order=(p, 0, q), enforce_stationarity=False, enforce_invertibility=False)
    model_fit = model.fit()

    # Make a one-step prediction for the next day
    prediction = model_fit.forecast(steps=1).iloc[0]
    # Store the prediction in the 'prediction' column
    next_date_str = str((date + pd.Timedelta(days=1)).normalize())
    next_date = pd.Timestamp(next_date_str[:11]+ " 00:00:00+00:00")
    data.loc[next_date, 'prediction'] = prediction
    data.loc[next_date, 'residual'] = data.loc[next_date, 'Close'] - prediction

✓ 15m 53.8s                                                    Python
```

در لوپ دیگری روی بازه ی تست و با p,q بهینه برای آریمای بازه ی train مدل Garch را روی ستونresidual فیت میکنیم.همانطور که گفته شده، اگر میانگینی که میدهد مثبت بود long و در غیر این صورت short میکنیم که در چیزی که من زدم در هر بار ۲درصد از مقدار یا رمزارز را معامله میکند.

پ.ن: میشد دو لوپ بالا را باهم ادغام کرد ولی من بخاطر ران تایم بالای لوپ بالایی ترجیح دادم که جدا کنم و دیتافریم تا آن مرحله را در یک فایل csv. ذخیره داشته باشم.

```python
# Convert the test_start and test_end to pandas Timestamps
test_start = pd.Timestamp(test_start)
test_end = pd.Timestamp(test_end)
test_dates = pd.date_range(test_start, test_end)
my_deposit = 100
my_wallet = 0
data['equity'] = np.nan
for date in test_dates:
    print('--------',date,'--------')
    train_data = data['residual'][:date].dropna()

    # Check if there are enough data points
    if len(train_data) >= 1:
        # Fit the GARCH model on the residuals of the ARIMA model
        garch_model = arch_model(train_data, vol='Garch', p=best_p, q=best_q)
        garch_fit = garch_model.fit(disp='off')
        garch_forecast = garch_fit.forecast(horizon=1)
        # garch_mean = garch_forecast.mean
        garch_mean = garch_forecast.mean['h.1'][0]
        # print(garch_mean["h.1"][0])
        date_str = str((date + pd.Timedelta(days=1)).normalize())
        # date_str = str(date.normalize())
        new_date = pd.Timestamp(date_str[:11]+ " 00:00:00+00:00")
        if garch_mean > 0 :
            money_amount = 0.02 * my_deposit
            my_deposit -= money_amount
            my_wallet += money_amount / data.loc[new_date, 'Close']
            print("#######",my_deposit, my_wallet,money_amount,"#######")
        else :
            btc_amount = 0.02 * my_wallet
            my_wallet -= btc_amount
            my_deposit += data.loc[new_date, 'Close'] * btc_amount  # Also corrected the assignment
            print("*******",my_deposit, my_wallet,btc_amount,"*******")
        data.loc[new_date, 'equity'] = my_deposit + my_wallet * data.loc[new_date, 'Close']
    else:
        print(f"Not enough data points ({len(train_data)}) to fit GARCH model")
```

نتیجه ی back test ما این میشود که ۱۸.۷ درصد به پولمان اضافه میشود

```
print(my_deposit)
print(data.loc['2023-11-30 00:00:00+00:00', 'Close'] * my_wallet)
```

0.0s Python

```
11.18453364129516
118.76394344536605
```

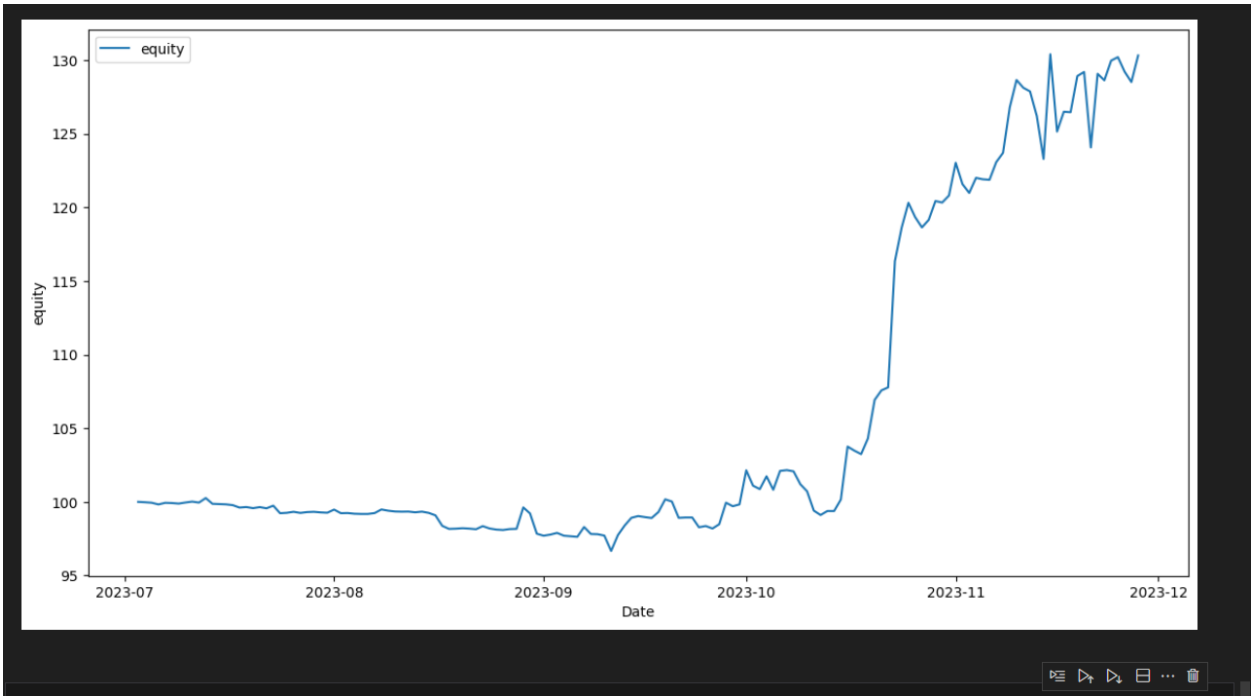
Sharpe ratio برابر ۰.۵۳ میباشد

```
# Calculate daily returns
data['returns'] = (data['Close'] - data['Close'].shift(1)) / data['Close'].shift(1)

# Calculate excess returns
risk_free_rate = 0.0  # Assuming a risk-free rate of 0%
data['excess_returns'] = data['returns'] - risk_free_rate

# Calculate the Sharpe Ratio
sharpe_ratio = np.sqrt(len(data)) * data['excess_returns'].mean() / data['excess_returns'].std()

print(f'Sharpe Ratio: {sharpe_ratio}')
```

0.0s Python

```
Sharpe Ratio: 0.529246847099481
```

نمودار equityما در بازه ی تست



در آخر دیتافریممان در فایل final_365.csv دخیره کردیم.

برای اتریوم:

کد عینا مثل بخش قبل است و من صرفا نتایج متناظرش را گذاشتم:

٪۱۶ ضرر دادیم:

```
print(my_deposit)
print(data.loc['2023-11-30 00:00:00+00:00', 'Close'] * my_wallet)
```

Python

```
21.987605044817606
84.06781143684837
```

Sharpe ratio: 0.47

```
# Calculate daily returns
data['returns'] = (data['Close'] - data['Close'].shift(1)) / data['Close'].shift(1)

# Calculate excess returns
risk_free_rate = 0.0  # Assuming a risk-free rate of 0%
data['excess_returns'] = data['returns'] - risk_free_rate

# Calculate the Sharpe Ratio
sharpe_ratio = np.sqrt(len(data)) * data['excess_returns'].mean() / data['excess_returns'].std()

print(f'Sharpe Ratio: {sharpe_ratio}')
```

Python

```
Sharpe Ratio: 0.4747816095610233
```

پ.ن: برای اتریوم من یادم رفت که در آخر دیتافریم را ذخیره کنم.

منابعی که در حل این تمرین کمک کردند:

کد تمرین قبلی

کدهای سر کلاس

Chat-GPT

نمونه کد

یک نمونه کد دیگر