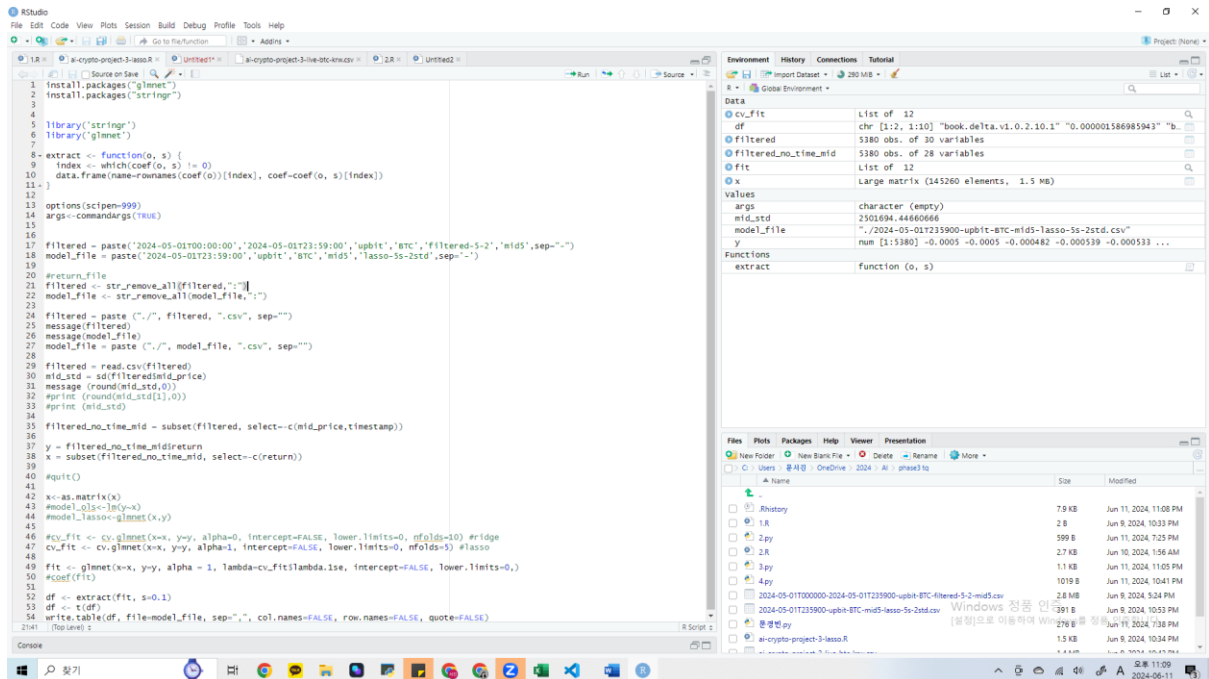


문경빈, 문서진 <https://github.com/Seojin980520/MOON>

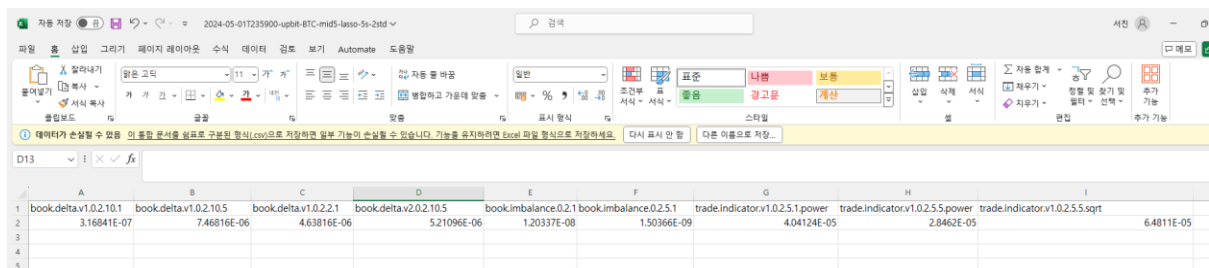
1-1 R 스크립트



```
1 install.packages("glmnet")
2 install.packages("stringr")
3
4
5 library("stringr")
6 library("glmnet")
7
8 extract <- function(o, s) {
9   index <- which(coef(o, s) != 0)
10  data.frame(name=row.names(coef(o)), coef=coef(o, s)[index])
11 }
12
13 options(scipen=999)
14 args <- commandArgs(TRUE)
15
16 filtered <- paste("2024-05-01T00:00:00", "2024-05-01T23:59:00", "upbit", "etc", "filtered-5-2", "mid5", sep="-")
17 model_file <- paste("2024-05-01T23:59:00", "upbit", "etc", "mid5", "lasso-5s-2std", sep="-")
18
19 #return_file
20 filtered <- str_remove_all(filtered, ".")
21 model_file <- str_remove_all(model_file, ".")
22
23 filtered <- paste("...", filtered, ".csv", sep="")
24 message(filtered)
25 message(model_file)
26 model_file <- paste("...", model_file, ".csv", sep="")
27
28 filtered <- read.csv(filtered)
29 mid_std <- sd(filtered$mid_price)
30 message(round(mid_std, 0))
31 #print(round(mid_std[1], 0))
32 #print(mid_std)
33
34 filtered_no_time_mid <- subset(filtered, select=-c(mid_price, timestamp))
35
36 y <- filtered_no_time_mid$return
37 x <- subset(filtered_no_time_mid, select=-c(return))
38 #quit()
39
40 x <- as.matrix(x)
41 #model_q1 <- lm(y~x)
42 #model_lasso <- glmnet(x, y)
43
44 cv_fit <- cv.glmnet(x=x, y=y, alpha=0, intercept=FALSE, lower.limits=0, rfolds=10) #ridge
45 cv_fit <- cv.glmnet(x=x, y=y, alpha=1, intercept=FALSE, lower.limits=0, rfolds=5) #lasso
46 fit <- glmnet(x=x, y=y, alpha = 1, lambda=cv_fit$lambda.1se, intercept=FALSE, lower.limits=0)
47 #coef(fit)
48
49 df <- extract(fit, s=0.1)
50 df <- t(df)
51
52 write.table(df, file=model_file, sep=",", col.names=FALSE, row.names=FALSE, quote=FALSE)
53
54 #stop script 1
```

Glmnet, stringr 등 필요한 패키지를 설치한 후 로드시킴. filtered 명령어를 사용해 데이터 파일의 이름 및 파일 경로를 생성함. mid_std를 이용해 중간값의 표준 편차를 구하고, 특히 glmnet 패키지를 이용해 교차검증에서 선택된 최적의 하이퍼파라미터로 라쏘 회귀모델을 학습함.

1-2 model file



	A	B	C	D	E	F	G	H	I
1	book.delta.v1.0.2.10.1	book.delta.v1.0.2.10.5	book.delta.v1.0.2.2.1	book.delta.v2.0.2.10.5	book.imbalance.0.2.1	book.imbalance.0.2.5.1	trade.indicator.v1.0.2.5.1.power	trade.indicator.v1.0.2.5.5.power	trade.indicator.v1.0.2.5.5.sqrt
2	3.16841E-07	7.46816E-06	4.63816E-06	5.21096E-06	1.20337E-08	1.50366E-09	4.04124E-05	2.8462E-05	6.4811E-05
3									
4									
5									

2-1 PnL 계산

```
import pandas as pd

# CSV 파일 읽기
file_path = 'ai-crypto-project-3-live-btc-krw.csv'
df = pd.read_csv(file_path)
```

```

# timestamp 를 datetime 으로 변환
df['timestamp'] = pd.to_datetime(df['timestamp'])

# side 값을 기준으로 거래 가치 부호 설정
df['signed_quantity'] = df.apply(lambda row: row['quantity'] if row['side'] == 1 else -
row['quantity'], axis=1)

# 거래 가치 계산 (signed_quantity 사용)
df['trade_value'] = df['signed_quantity'] * df['price']

# 수수료 반영
df['net_trade_value'] = df.apply(lambda row: row['trade_value'] + row['fee'] if
row['trade_value'] < 0 else row['trade_value'] - row['fee'], axis=1)

# 날짜별 순 거래 가치 합계 계산
daily_value = df.groupby(df['timestamp'].dt.date)['net_trade_value'].sum()

# 일일 PnL 계산
daily_pnl = daily_value.diff().fillna(daily_value)

# 누적 PnL 계산
cumulative_pnl = daily_pnl.cumsum()

# 일일 PnL 의 총합 계산 (누적 PnL 의 마지막 값)
total_PnL = cumulative_pnl.sum()

# 최종 PnL 출력
print(f"total PnL (profit minus fee): {total_PnL}")

```

이 코드는 주어진 CSV 파일에서 데이터를 불러와 각 거래의 가치를 계산하고, 일별 순 거래 가치와 일일 및 누적 이익 손실(PnL)을 계산함. 먼저 파일을 읽고, timestamp 를 datetime 형식으로 변환함. 그런 다음 거래 가치를 계산하고 수수료를 반영하여 순 거래 가치를 얻음. 이후에는 날짜별로 순 거래 가치를 합산하여 일일 PnL 을 계산하고, 누적 PnL 을 구함. 마지막으로, 일일 PnL 의 총합을 계산하여 최종 PnL 을 출력함.

2-2 PnL 계산 결과 값

