Project 3 github : https://github.com/donghn5/ai_crypto 이건, 이동현

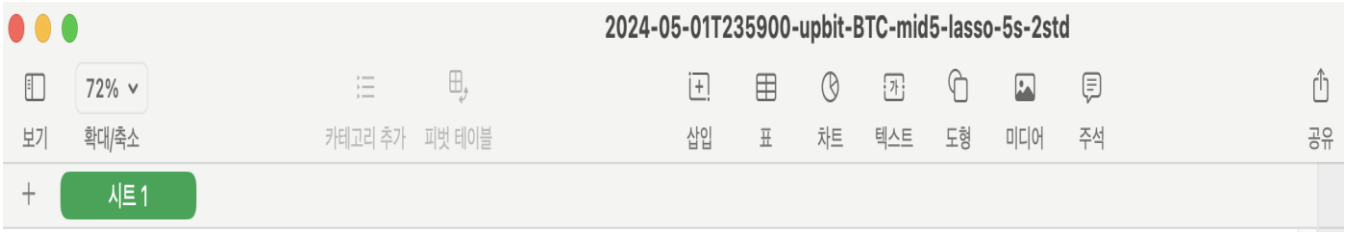
1.모델파일생성 및 결과파일

< 코드 >

```
library('stringr')
library('glmnet')

extract <- function(o, s) {
  index <- which(coef(o, s) != 0)
  data.frame(name=rownames(coef(o))[index], coef=coef(o, s)[index])
}

options(scipen=999)

args <- commandArgs(TRUE)

#args[1] s time
#args[2] e time
#args[3] exchange
#args[4]

filtered <- paste(args[1], args[2], args[3], args[4], 'filtered-5-2', args[5], sep="-")
model_file <- paste(args[2], args[3], args[4], args[5], 'lasso-5s-2std', sep='-')

#return_file
filtered <- str_remove_all(filtered, ":")
model_file <- str_remove_all(model_file, ":")
```

```
filtered <- paste("./", filtered, ".csv", sep="")
model_file <- paste("./", model_file, ".csv", sep="")

filtered <- read.csv(filtered)
mid_std <- sd(filtered$mid_price)
round_mid_std <- round(mid_std, 0)

filtered_no_time_mid <- subset(filtered, select=-c(mid_price, timestamp))

y <- filtered_no_time_mid$return
x <- subset(filtered_no_time_mid, select=-c(return))

x <- as.matrix(x)

#cv_fit <- cv.glmnet(x=x, y=y, alpha=0, intercept=FALSE, lower.limits=0, nfolds=10) #ridge
cv_fit <- cv.glmnet(x=x, y=y, alpha=1, intercept=FALSE, lower.limits=0, nfolds=5) #lasso

fit <- glmnet(x=x, y=y, alpha=1, lambda=cv_fit$lambda.1se, intercept=FALSE, lower.limits=0)

df <- extract(fit, s=0.1)
df <- t(df)
write.table(df, file=model_file, sep=",", col.names=FALSE, row.names=FALSE, quote=FALSE)
```

< 생성된 결과파일의 데이터 >



2024-05-01T235900-upbit-BTC-mid5-lasso-5s-2std

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.0.2.10.1	book.imbalance.0.2.5.1	trade.indicator.v1.0.2.5.1.power	trade.indicator.v1.0.2.5.1.sqrt	trade.indicator.v1.0.2.5.5.power	trade.indicator.v1.0.2.5.5.sqrt
0.000001992968987	0.000008076860897	0.000004044948273	0.000009734125795	0.000000013204137	0.000000004540436	0.000086476656982	0.000028840199011	0.000032791595342	0.000057938223752

< 설명 >

Lasso 회귀는 과적합을 방지하고, 변수 선택의 기능을 하는 회귀 분석 방법입니다. 기존의 선형 회귀 모델에 L1 규제를 추가하여 일부 계수를 정확히 0으로 만들어, 그에 해당하는 변수를 모델에서 제외합니다. 이 부분은 glmnet 함수와 cv.glmnet 함수에서 이루어집니다. Lasso 회귀의 특성상, α 값이 1로 설정되면 Lasso 규제가 적용되어 일부 계수가 0으로 설정됩니다. 코드를 보면 다음과 같은 부분이 있습니다.

“cv_fit <- cv.glmnet(x=x, y=y, alpha=1, intercept=FALSE, lower.limits=0, nfolds=5)

fit <- glmnet(x=x, y=y, alpha=1, lambda=cv_fit$lambda.1se, intercept=FALSE, lower.limits=0)”

여기서 중요한 부분은 alpha=1입니다. alpha 값이 1로 설정되면 Lasso 회귀가 수행됩니다. Lasso 회귀는 비용 함수에 $\lambda \sum_{i=1}^{p} |\beta_j|$항을 추가하여 일부 계수를 0으로 만듭니다. 이 두 줄에서 Lasso 회귀 모델이 학습되고, 교차 검증을 통해 최적의 λ 값(cv_fit$lambda.1se)이 선택됩니다. 이로 인해 일부 계수가 0이 되어, 그에 해당하는 변수가 모델에서 제외됩니다. 이런 방식으로, Lasso 회귀분석은 모델을 단순화하면서도 중요한 정보를 잃지 않도록 도와주며, 덕분에 우리가 결과를 이해하고 해석하기에도 훨씬 수월해집니다.

코드를 살펴보면 “extract <- function(o, s) {

index <- which(coef(o, s) != 0)

data.frame(name=rownames(coef(o))[index], coef=coef(o, s)[index])

}”에서모델객체 o와 람다값 s를 받아 회귀계수가 0이 아닌 변수들의 이름과 계수를 데이터프레임 형태로 반환하는 것을 볼 수 있습니다. 그리고 데이터프레임을 구성한 뒤에 “cv_fit <- cv.glmnet(x=x, y=y, alpha=1, intercept=FALSE, lower.limits=0, nfolds=5)”을 통해 Lasso 회귀 모델을 교차 검증을 통해 학습합니다. alpha=1은 위에서 언급한 것처럼Lasso를 의미하며, intercept=FALSE는 절편을 포함하지 않음을 의미합니다. lower.limits=0은 계수가 0보다 작을 수 없음을 의미하며, nfolds=5는 5겹 교차 검증을 사용함을 의미합니다.

“df <- extract(fit, s=0.1)”에서 extract 함수를 사용하여 학습된 모델 fit에서 계수가 0이 아닌 변수들의 이름과 계수를 추출합니다. 이후 이 데이터를 model_file 이름의 CSV파일로 저장하게 됩니다. 이렇게 생기는 파일이 저희가 구현하고자 한 최종 모델 파일입니다.

2. PnL 계산하는 코드

```
Import pandas as pd

#pands 라이브러리 호출 및 pd 로 모듈 간소화


#CSV file read

df = pd.read_csv('ai-crypto-project-3-live-btc-krw.csv')


#PnL(Profit and Loss) 계산

#PnL = 거래 수량 * 거래 가격 * (2 * side – 1)

#2*side-1 이란 sell일때quantity*price에 -1을  곱하고, buy일때 1을 곱하는 식

df['PnL'] = df['quantity'] * df['price'] * (df['side'] * 2 - 1)


#누적 PnL 계산

df['Cumulative PnL'] = df['PnL'].cumsum()

#Cumulative Quantity = (quantity * (2 * side - 1))

#quantity에 sell이면 -1, buy면 1을 곱한뒤Cumulativequantity에 누적하여 보유 수량 계산

df['Cumulativequantity'] = (df['quantity'] * (df['side'] * 2 - 1)).cumsum()


#결과 출력

#간략한 csv파일과보유 수량, 누적 PnL을 출력하고 누적 PnL로 손익판단 가능

print(df)

print(f"누적 보유 수량: {df['Cumulativequantity'].iloc[-1]}")

print(f"누적 PnL: {df['CumulativePnL'].iloc[-1]:.2f}")

#누적 PnL이 34405642.04 가 나왔으므로 2024.03.27 23:28 부터 2024.03.31 23:34 까지 거래한 암호화폐는 이익을 봤음.
```

< 결과화면 스크린샷 >

```
root@aicrypto:~/project3# python3 final_project3.py
             timestamp  quantity      price      fee  ...  side          PnL  Cumulative PnL  Cumulative quantity
0      2024-03-07 23:28  0.100000   94656000  4732.80  ...     0 -9.465600e+06   -9.465600e+06            -0.100000
1      2024-03-07 23:30  0.011000   94686000   520.77  ...     1  1.041546e+06   -8.424054e+06            -0.089000
2      2024-03-07 23:30  0.020937   94686000   991.24  ...     1  1.982484e+06   -6.441570e+06            -0.068063
3      2024-03-07 23:30  0.010561   94686000   499.97  ...     1  9.999571e+05   -5.441613e+06            -0.057502
4      2024-03-07 23:30  0.014159   94686000   670.30  ...     1  1.340616e+06   -4.100996e+06            -0.043343
...                 ...       ...        ...      ...  ...   ...           ...             ...                  ...
27096  2024-03-31 23:23  0.030000  100514000  1507.71  ...     1  3.015420e+06    3.713740e+07            -0.017251
27097  2024-03-31 23:24  0.017200  100519000   864.46  ...     1  1.728927e+06    3.886632e+07            -0.000051
27098  2024-03-31 23:34  0.030000  100462000  1506.93  ...     0 -3.013860e+06    3.585246e+07            -0.030051
27099  2024-03-31 23:34  0.011992  100461000   602.35  ...     0 -1.204704e+06    3.464776e+07            -0.042043
27100  2024-03-31 23:34  0.002410  100458000   121.05  ...     0 -2.421158e+05    3.440564e+07            -0.044453

[27101 rows x 9 columns]
누적 보유 수량 : -0.04445324000000103
누적 PnL: 34405642.04
root@aicrypto:~/project3#
```