Github : <https://github.com/JihongJeong/AI_Crypto> Team : 정지홍, 윤준호

Assignment 3-1.

Model Code :

####### How to install R using Homebrew:

# $brew install r or $brew cask install rstudio

# $sudo r

####### How to install R packages:

# $sudo r

# $install.packages("glmnet")

####### How to run R script:

# $Rscript your-file.R

####### Usage: Finding a Model

# arguemnts: start-time end-time exchange coin-symbol mid5

# $Rscript ./ai-crypto-project-3-lasso.R '2024-05-01T00:00:00' '2024-05-01T23:59:00' upbit BTC mid5

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="")

message(filtered)

message(model\_file)

model\_file = paste ("./", model\_file, ".csv", sep="")

filtered = read.csv(filtered)

mid\_std = sd(filtered$mid\_price)

message (round(mid\_std,0))

#print (round(mid\_std[1],0))

#print (mid\_std)

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))

#quit()

x<-as.matrix(x)

#model\_ols<-lm(y~x)

#model\_lasso<-glmnet(x,y)

#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,)

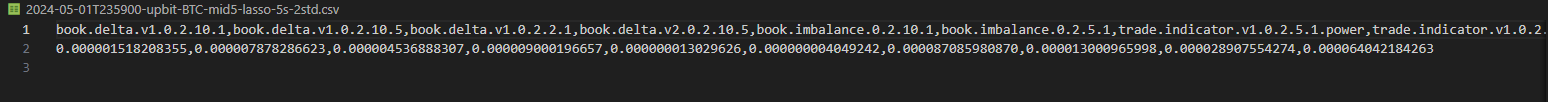
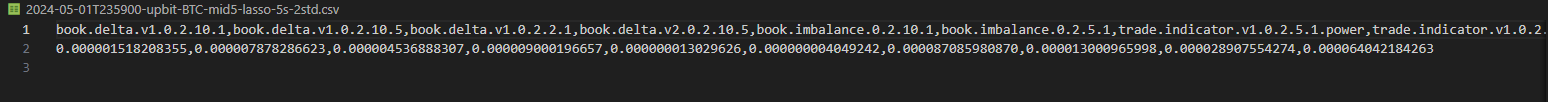
#coef(fit)

df <- extract(fit, s=0.1)

df <- t(df)

write.table(df, file=model\_file, sep=",", col.names=FALSE, row.names=FALSE, quote=FALSE)

Model Result :



Assignment 3-2.

PnL scoring Code :

import polars as pl

import numpy as np

from datetime import datetime

import sys

def main():

    pl.Config(set\_fmt\_float = "full")

    file\_name = sys.argv[1]

    data = pl.read\_csv(file\_name)

    data = data.with\_columns(data.select((pl.col('quantity') \* pl.col('price')).alias('px')))

    data = data.with\_columns(pl.when(pl.col('side') == 0).then(-1).otherwise(1).alias('type'))

    data = data.with\_columns(data.select((pl.col('px') \* pl.col('type') - pl.col('fee')).alias('tot\_px')))

    data = data.with\_columns(pl.cum\_sum('tot\_px').alias('PnL\_per\_trade'))

    trade\_amount = data.select(pl.col('timestamp', 'PnL\_per\_trade'))

    timestamp\_amount = data.select(pl.col('timestamp', 'tot\_px'))

    timestamp\_amount = timestamp\_amount.group\_by('timestamp').agg(pl.col('tot\_px').sum()).sort('timestamp')

    timestamp\_amount = timestamp\_amount.rename({'tot\_px' : 'PnL\_per\_timestamp'})

    timestamp\_day = data.select(pl.col('timestamp')).to\_numpy()

    pt = []

    for time in timestamp\_day:

        tmp = datetime.strptime(time[0], "%Y-%m-%d %H:%M").date()

        pt.append(datetime.strftime(tmp, "%y-%m-%d"))

    data = data.with\_columns(pl.Series(pt).alias('timestamp\_date'))

    date\_amount = data.group\_by('timestamp\_date').agg(pl.col('tot\_px').sum()).sort('timestamp\_date')

    prev\_date\_amount = date\_amount.select(pl.col('tot\_px')).to\_numpy().T

    prev\_date\_amount = np.pad(prev\_date\_amount.ravel(), (1,0))[:-1]

    date\_amount = date\_amount.with\_columns(prev\_tot\_px = prev\_date\_amount)

    date\_amount = date\_amount.with\_columns((pl.col('tot\_px') + pl.col('prev\_tot\_px')).alias('PnL\_each\_date'))

    date\_amount = date\_amount.with\_columns((pl.cum\_sum('tot\_px')).alias('PnL\_per\_date'))

    date\_amount = date\_amount.select(pl.col('timestamp\_date', 'PnL\_each\_date', 'PnL\_per\_date'))

    trade\_amount.write\_csv("PnL\_per\_trade\_" + file\_name)

    timestamp\_amount.write\_csv("PnL\_per\_timestamp\_" + file\_name)

    date\_amount.write\_csv("PnL\_per\_date\_" + file\_name)

    f = open("./PnL\_score.csv", 'w')

    f.write(f"File Name : {file\_name}, PnL score : {date\_amount['PnL\_per\_date'][-1]:.1f}")

    f.close()

if \_\_name\_\_ == '\_\_main\_\_':

    main()

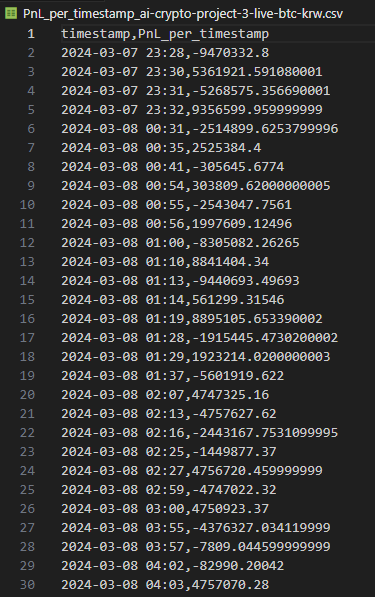
PnL Result :



추가로 date, timestamp, trade별로 PnL score를 계산해서 파일로 저장하는 방식을 사용해서 특정 날짜 혹은 시간에 PnL score가 어떻게 되는지, 날짜에 따라 PnL score가 어떻게 변하는지에 대해서도 tracking 할 수 있도록 하였습니다.

Additional Results :

텍스트, 스크린샷, 메뉴이(가) 표시된 사진

자동 생성된 설명 

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명