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

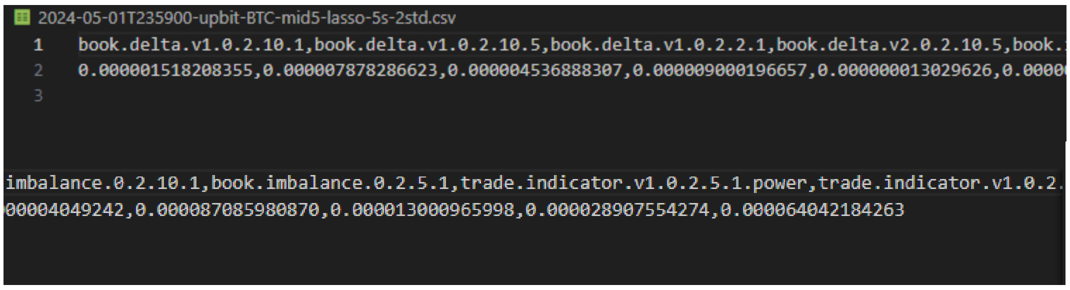
Assignment 3-1.

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="")
```

```r
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 :



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

```
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.
0.000001518208355,0.000007878286623,0.000004536888307,0.000009000196657,0.000000013029626,0.0000
```

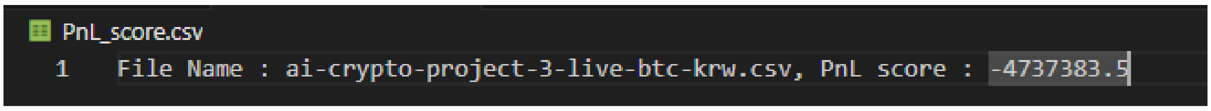
```
imbalance.0.2.10.1,book.imbalance.0.2.5.1,trade.indicator.v1.0.2.5.1.power,trade.indicator.v1.0.2.
00004049242,0.000087085980870,0.000013000965998,0.000028907554274,0.000064042184263
```

Assignment 3-2.

PnL scoring Code :

```python
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 :

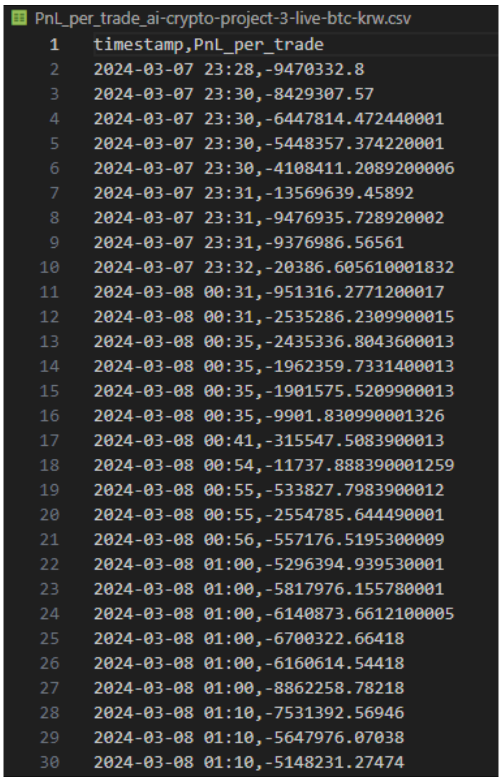


PnL_score.csv

```
File Name : ai-crypto-project-3-live-btc-krw.csv, PnL score : -4737383.5
```

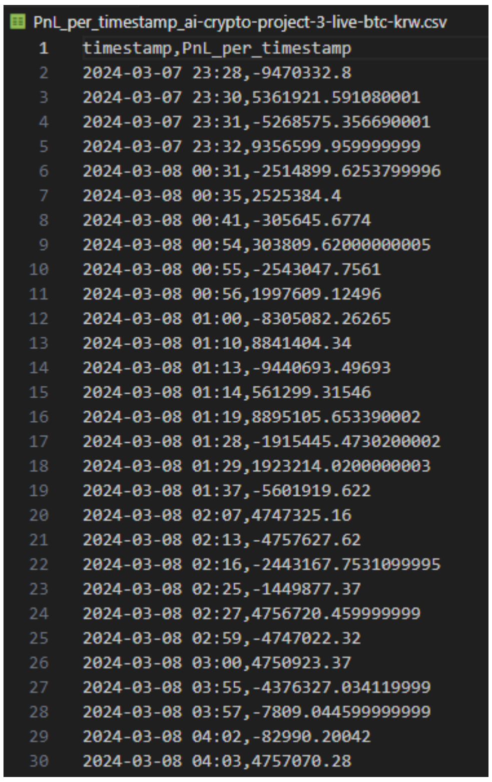
추가로 date, timestamp, trade별로 PnL score를 계산해서 파일로 저장하는 방식을 사용해서 특정 날짜 혹은 시간에 PnL score가 어떻게 되는지, 날짜에 따라 PnL score가 어떻게 변하는지에 대해서도 tracking 할 수 있도록 하였습니다.

Additional Results :



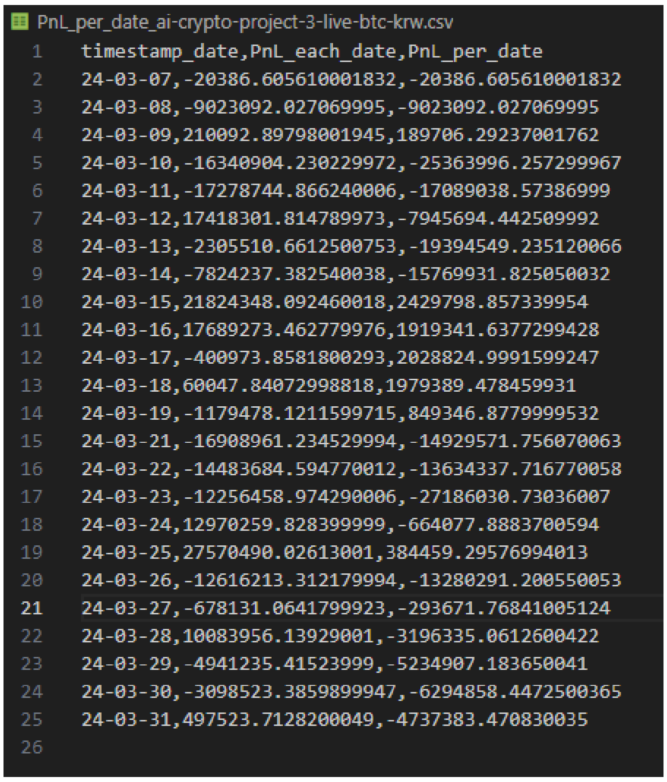
PnL_per_trade_ai-crypto-project-3-live-btc-krw.csv

```
timestamp,PnL_per_trade
2024-03-07 23:28,-9470332.8
2024-03-07 23:30,-8429307.57
2024-03-07 23:30,-6447814.472440001
2024-03-07 23:30,-5448357.374220001
2024-03-07 23:30,-4108411.2089200006
2024-03-07 23:31,-13569639.45892
2024-03-07 23:31,-9476935.728920002
2024-03-07 23:31,-9376986.56561
2024-03-07 23:32,-20386.605610001832
2024-03-08 00:31,-951316.2771200017
2024-03-08 00:31,-2535286.2309900015
2024-03-08 00:35,-2435336.8043600013
2024-03-08 00:35,-1962359.7331400013
2024-03-08 00:35,-1901575.5209900013
2024-03-08 00:35,-9901.830990001326
2024-03-08 00:41,-315547.5083900013
2024-03-08 00:54,-11737.888390001259
2024-03-08 00:55,-533827.7983900012
2024-03-08 00:55,-2554785.644490001
2024-03-08 00:56,-557176.5195300009
2024-03-08 01:00,-5296394.939530001
2024-03-08 01:00,-5817976.155780001
2024-03-08 01:00,-6140873.6612100005
2024-03-08 01:00,-6700322.66418
2024-03-08 01:00,-6160614.54418
2024-03-08 01:00,-8862258.78218
2024-03-08 01:10,-7531392.56946
2024-03-08 01:10,-5647976.07038
2024-03-08 01:10,-5148231.27474
```



PnL_per_timestamp_ai-crypto-project-3-live-btc-krw.csv

```
timestamp,PnL_per_timestamp
2024-03-07 23:28,-9470332.8
2024-03-07 23:30,5361921.591080001
2024-03-07 23:31,-5268575.356690001
2024-03-07 23:32,9356599.959999999
2024-03-08 00:31,-2514899.6253799996
2024-03-08 00:35,2525384.4
2024-03-08 00:41,-305645.6774
2024-03-08 00:54,303809.62000000005
2024-03-08 00:55,-2543047.7561
2024-03-08 00:56,1997609.12496
2024-03-08 01:00,-8305082.26265
2024-03-08 01:10,8841404.34
2024-03-08 01:13,-9440693.49693
2024-03-08 01:14,561299.31546
2024-03-08 01:19,8895105.653390002
2024-03-08 01:28,-1915445.4730200002
2024-03-08 01:29,1923214.0200000003
2024-03-08 01:37,-5601919.622
2024-03-08 02:07,4747325.16
2024-03-08 02:13,-4757627.62
2024-03-08 02:16,-2443167.7531099995
2024-03-08 02:25,-1449877.37
2024-03-08 02:27,4756720.459999999
2024-03-08 02:59,-4747022.32
2024-03-08 03:00,4750923.37
2024-03-08 03:55,-4376327.034119999
2024-03-08 03:57,-7809.044599999999
2024-03-08 04:02,-82990.20042
2024-03-08 04:03,4757070.28
```



PnL_per_date_ai-crypto-project-3-live-btc-krw.csv

```
timestamp_date,PnL_each_date,PnL_per_date
24-03-07,-20386.605610001832,-20386.605610001832
24-03-08,-9023092.027069995,-9023092.027069995
24-03-09,210092.89798001945,189706.29237001762
24-03-10,-16340904.230229972,-25363996.257299967
24-03-11,-17278744.866240006,-17089038.57386999
24-03-12,17418301.814789973,-7945694.442509992
24-03-13,-2305510.6612500753,-19394549.235120066
24-03-14,-7824237.382540038,-15769931.825050032
24-03-15,21824348.092460018,2429798.857339954
24-03-16,17689273.462779976,1919341.6377299428
24-03-17,-400973.8581800293,2028824.9991599247
24-03-18,60047.84072998818,1979389.478459931
24-03-19,-1179478.1211599715,849346.8779999532
24-03-21,-16908961.234529994,-14929571.756070063
24-03-22,-14483684.594770012,-13634337.716770058
24-03-23,-12256458.974290006,-27186030.73036007
24-03-24,12970259.828399999,-664077.8883700594
24-03-25,27570490.02613001,384459.29576994013
24-03-26,-12616213.312179994,-13280291.200550053
24-03-27,-678131.0641799923,-293671.76841005124
24-03-28,10083956.13929001,-3196335.0612600422
24-03-29,-4941235.41523999,-5234907.183650041
24-03-30,-3098523.3859899947,-6294858.4472500365
24-03-31,497523.7128200049,-4737383.470830035
```