

feature extraction maysam

maysam rashidi

2024-06-01

```
library(data.table)
library(keras)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##   between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##   filter, lag
```

```
## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union
```

```
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:data.table':  
##  
##   yearmon, yearqtr
```

```
## The following objects are masked from 'package:base':  
##  
##   as.Date, as.Date.numeric
```

```
##  
## ##### Warning from 'xts' package #####  
## #  
## # The dplyr lag() function breaks how base R's lag() function is supposed to #  
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or #  
## # source() into this session won't work correctly. #  
## #  
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #  
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop #  
## # dplyr from breaking base R's lag() function. #  
## #  
## # Code in packages is not affected. It's protected by R's namespace mechanism #  
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning. #  
## #  
## #####
```

```
##  
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':  
##  
##   first, last
```

```
## The following objects are masked from 'package:data.table':  
##  
##   first, last
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method                from
##   as.zoo.data.frame zoo
```

```
library(zoo)
library(TTR)
# Attempt to load the CSV file with a suspected delimiter
data <- read.csv("titlonmoyin.csv", header = TRUE, sep = "c") # If "c" is suspected as part of the delimiter

# try to see if there's a complex delimiter like "c\" or similar
#data <- read.csv("titlonmoyin.csv", header = TRUE, sep = 'c') # If 'c"' is the delimiter

# Check the structure of the data to see if columns are properly separated
str(data)
```

```
## 'data.frame':    1872367 obs. of  48 variables:
## $ Date           : chr  "2019-07-15" "2019-07-16" "2019-07-17" "2019-07-18" ...
## $ Internal.code   : int   2014128 2014128 2014128 2014128 2014128 2014128 2014128 2014128 2014128 20
14128 ...
## $ SecurityId      : int   1304857 1304857 1304857 1304857 1304857 1304857 1304857 1304857 1304857 13
04857 ...
## $ CompanyId       : int   12720 12720 12720 12720 12720 12720 12720 12720 12720 12720 ...
## $ Symbol          : chr    "2020" "2020" "2020" "2020" ...
## $ ISIN            : chr    "BMG9156K1018" "BMG9156K1018" "BMG9156K1018" "BMG9156K1018" ...
## $ Name            : chr    "2020 Bulklers" "2020 Bulklers" "2020 Bulklers" "2020 Bulklers" ...
## $ BestBidPrice     : num    80 80.5 80.4 77.4 78 ...
## $ BestAskPrice     : num    81 81 81 80 79 78.5 79 78 77.4 77 ...
## $ Open            : num    83 81 81 80 78 ...
## $ High            : num    84 81 81 80 79 ...
## $ Low             : num    81 80 80.5 77 78 ...
## $ Close           : num    81 81 81 80 79 ...
## $ NumberOfTrades   : int    NA NA NA NA NA NA NA NA NA NA ...
## $ Volume          : int   8810 21400 12966 4789 5030 8989 500 2151 12126 2195 ...
## $ Turnover         : num   720755 1733377 1049996 382497 397340 ...
## $ VolumeWeightedAveragePrice: num   81.8 81 81 79.9 79 ...
## $ Price           : num    81 81 81 80 79 ...
## $ AdjustedPrice    : num   49.9 49.9 49.9 49.3 48.7 ...
## $ Dividends        : num    0 0 0 0 0 0 0 0 0 0 ...
## $ LDividends       : num    0 0 0 0 0 0 0 0 0 0 ...
## $ CorpAdj          : num    1 1 1 1 1 1 1 1 1 1 ...
```

```

## $ DividendAdj : num 0.617 0.617 0.617 0.617 0.617 ...
## $ Currency : chr "NOK" "NOK" "NOK" "NOK" ...
## $ NumberOfShares : num 22170906 22170906 22170906 22170906 22170906 ...
## $ Exchange : chr "OAX" "OAX" "OAX" "OAX" ...
## $ NOKPerForex : int 1 1 1 1 1 1 1 1 1 1 ...
## $ mktcap : num 1.80e+09 1.80e+09 1.80e+09 1.77e+09 1.75e+09 ...
## $ OSEBXmktshare_prevmth : num NA NA NA NA NA NA NA NA NA NA ...
## $ OSEBXAlpha_prevmth : num NA NA NA NA NA NA NA NA NA NA ...
## $ OSEBxBeta_prevmth : num NA NA NA NA NA NA NA NA NA NA ...
## $ SMB : num -0.000806 -0.007297 0.001962 0.009697 -0.004902 ...
## $ HML : num -0.00519 0.00342 -0.00321 -0.01069 -0.00554 ...
## $ LIQ : num 0.01343 -0.01184 0.00741 0.00646 -0.00242 ...
## $ MOM : num -0.01566 0.00819 -0.00329 0.00108 -0.00118 ...
## $ lnDeltaP : num -0.0244 0 0 -0.0124 -0.0126 ...
## $ lnDeltaOSEBX : num -0.00144 -0.00728 -0.00721 -0.00687 0.00154 ...
## $ lnDeltaOBX : num -0.00106 -0.00916 -0.008 -0.00584 0.00289 ...
## $ NOWA_DayLnrate : num 5.52e-05 5.52e-05 5.48e-05 5.57e-05 5.52e-05 ...
## $ bills_3month_Lnrate : num 5.61e-05 5.64e-05 5.65e-05 5.66e-05 5.67e-05 ...
## $ Sector : chr "Industrials" "Industrials" "Industrials" "Industrials" ...
## $ IN_OSEBX : int 0 0 0 0 0 0 0 0 0 0 ...
## $ Equity : num NA NA NA NA NA NA NA NA NA NA ...
## $ Debt : num NA NA NA NA NA NA NA NA NA NA ...
## $ Earnings : num NA NA NA NA NA NA NA NA NA NA ...
## $ debt_ratio : num NA NA NA NA NA NA NA NA NA NA ...
## $ PE : num NA NA NA NA NA NA NA NA NA NA ...
## $ ID : int 1570527 1437408 1437518 1570897 1571022 1571146 1437961 1571374 1571492 14
38389 ...

```

```

# Replace commas with dots and convert to numeric where needed
data <- data %>%
  mutate(across(c(SMB, HML, LIQ, MOM, Close), ~as.numeric(gsub(",", ".", .x))),
    Date = as.Date(Date, format = "%Y-%m-%d")) # Convert Date to Date type

#write.csv(data, "titlonmoyinready.csv", row.names = FALSE)
data <- read.csv("titlonmoyinready.csv")

str(data, 50)

```

```

## 'data.frame': 1872358 obs. of 63 variables:
## $ Date : chr "2019-07-29" "2019-07-30" "2019-07-31" "2019-08-01" ...

```

```

## $ Internal.code      : int  2014128 2014128 2014128 2014128 2014128 2014128 2014128 2014128 2014128 20
14128 ...
## $ SecurityId         : int  1304857 1304857 1304857 1304857 1304857 1304857 1304857 1304857 1304857 13
04857 ...
## $ CompanyId         : int  12720 12720 12720 12720 12720 12720 12720 12720 12720 12720 ...
## $ Symbol            : chr  "2020" "2020" "2020" "2020" ...
## $ ISIN              : chr  "BMG9156K1018" "BMG9156K1018" "BMG9156K1018" "BMG9156K1018" ...
## $ Name              : chr  "2020 Bulkers" "2020 Bulkers" "2020 Bulkers" "2020 Bulkers" ...
## $ BestBidPrice      : num  76 76 77 70 70.1 ...
## $ BestAskPrice      : num  77 77 78 76 76 69 71 70 75 75.5 ...
## $ Open              : num  76 76 77 78 76 ...
## $ High              : num  78 77 77 78 76 ...
## $ Low               : num  76 76 76 75 75 ...
## $ Close             : num  77 77 77 75 75 ...
## $ NumberOfTrades    : int  NA NA NA NA NA NA NA NA NA NA ...
## $ Volume            : num  2195 100 4324 1522 537 ...
## $ Turnover          : num  169518 7651 332116 115770 40285 ...
## $ VolumeWeightedAveragePrice: num  77.2 76.5 76.8 76.1 75 ...
## $ Price             : num  77 77 77 75 75 ...
## $ AdjustedPrice     : num  47.5 47.5 47.5 46.2 46.2 ...
## $ Dividends         : num  0 0 0 0 0 0 0 0 0 0 ...
## $ LDividends        : num  0 0 0 0 0 0 0 0 0 0 ...
## $ CorpAdj           : num  1 1 1 1 1 1 1 1 1 1 ...
## $ DividendAdj       : num  0.617 0.617 0.617 0.617 0.617 ...
## $ Currency          : chr  "NOK" "NOK" "NOK" "NOK" ...
## $ NumberOfShares    : num  22170906 22170906 22170906 22170906 22170906 ...
## $ Exchange          : chr  "OAX" "OAX" "OAX" "OAX" ...
## $ NOKPerForex       : int  1 1 1 1 1 1 1 1 1 1 ...
## $ mktcap            : num  1.71e+09 1.71e+09 1.71e+09 1.66e+09 1.66e+09 ...
## $ OSEBXmktshare_prevmnth : num  NA NA NA NA NA NA NA NA NA NA ...
## $ OSEBXAlpha_prevmnth : num  NA NA NA NA NA NA NA NA NA NA ...
## $ OSEBXBeta_prevmnth  : num  NA NA NA NA NA NA NA NA NA NA ...
## $ SMB              : num  -0.005101 0.004157 0.000479 -0.001486 0.008561 ...
## $ HML              : num  0.00818 -0.00442 -0.01274 -0.00161 -0.00127 ...
## $ LIQ              : num  0.001633 0.010677 -0.00865 0.000509 0.011394 ...
## $ MOM              : num  -0.00508 -0.00767 0.00282 -0.00175 0.00339 ...
## $ lnDeltaP         : num  -0.00531 0.00013 0 -0.02632 0 ...
## $ lnDelta0SEBX     : num  0.00148 -0.01476 0.00787 0.00479 -0.01129 ...
## $ lnDelta0BX       : num  0.000756 -0.015365 0.008592 0.004494 -0.012291 ...
## $ NOWA_DayLnrate    : num  5.57e-05 5.52e-05 5.52e-05 5.48e-05 5.48e-05 ...
## $ bills_3month_Lnrate : num  5.70e-05 5.70e-05 5.70e-05 5.70e-05 5.71e-05 ...
## $ Sector           : chr  "Industrials" "Industrials" "Industrials" "Industrials" ...

```

```

## $ IN_OSEBX      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Equity        : num  NA NA NA NA NA NA NA NA NA NA ...
## $ Debt          : num  NA NA NA NA NA NA NA NA NA NA ...
## $ Earnings      : num  NA NA NA NA NA NA NA NA NA NA ...
## $ debt_ratio    : num  NA NA NA NA NA NA NA NA NA NA ...
## $ PE            : num  NA NA NA NA NA NA NA NA NA NA ...
## $ ID            : int  1438389 1571848 1438611 1572088 1438840 1572536 1572657 1572890 1573019 14
39748 ...
## $ MA5           : num  370 457 367 341 320 ...
## $ MA10          : num  250 315 310 300 302 ...
## $ MA20          : num  NA NA NA NA NA NA NA NA NA NA ...
## $ BIAS5         : num  -79.2 -83.2 -79 -78 -76.6 ...
## $ BIAS10        : num  -69.2 -75.5 -75.2 -75 -75.1 ...
## $ RSI6          : num  NA NA NA NA NA ...
## $ RSI12         : num  NA NA NA NA NA NA NA NA NA NA ...
## $ Stoch_K       : num  NA NA NA NA NA NA NA NA NA NA ...
## $ Stoch_D       : num  NA NA NA NA NA NA NA NA NA NA ...
## $ MACD          : num  NA NA NA NA NA NA NA NA NA NA ...
## $ Signal        : num  NA NA NA NA NA NA NA NA NA NA ...
## $ WPR           : num  NA NA NA NA NA NA NA NA NA NA ...
## $ VOL1          : num  NA NA NA NA NA NA NA NA NA NA ...
## $ VOL2          : num  NA NA NA NA NA NA NA NA NA NA ...
## $ Delta_MA5     : num  NA 87 -90 -26 -20.5 -115 -107 0.5 0.5 -0.5 ...

```

```
summary(data,50)
```

```

##      Date      Internal.code      SecurityId      CompanyId
## Length:1872358 Min.   :2004834 Min.    :   6000 Min.    : 1007
## Class :character 1st Qu.:2015528 1st Qu.:   6212 1st Qu.: 2237
## Mode  :character Median :2015566 Median :   24518 Median : 5203
##              Mean  :2037838 Mean  :  170204 Mean   : 5307
##              3rd Qu.:2015611 3rd Qu.:   60388 3rd Qu.: 7953
##              Max.   :9067757 Max.   :1305713 Max.   :12811
##              NA's   :812035  NA's   :112444 NA's   :112547
##      Symbol      ISIN      Name      BestBidPrice
## Length:1872358 Length:1872358 Length:1872358 Min.    : -999999999
## Class :character Class :character Class :character 1st Qu.:      9
## Mode  :character Mode  :character Mode  :character Median :     40
##              Mean   :   -372601
##              3rd Qu.:     110
##              Max.   :    24000

```

```

##
## BestAskPrice Open High Low
## Min. :-999999999 Min. : 0.0 Min. : 0.00 Min. : 0.00
## 1st Qu.: 9 1st Qu.: 8.0 1st Qu.: 9.00 1st Qu.: 8.70
## Median : 40 Median : 34.9 Median : 40.00 Median : 39.00
## Mean : -649848 Mean : 76.8 Mean : 89.05 Mean : 87.36
## 3rd Qu.: 110 3rd Qu.: 98.0 3rd Qu.: 111.50 3rd Qu.: 109.50
## Max. : 25000 Max. :17500.0 Max. :25000.00 Max. :25000.00
## NA's :18329 NA's :116694 NA's :11 NA's :11
## Close NumberOfTrades Volume Turnover
## Min. : 0.00 Min. : 0.0 Min. :-1.819e+09 Min. :0.000e+00
## 1st Qu.: 8.90 1st Qu.: 12.0 1st Qu.: 3.144e+03 1st Qu.:1.128e+05
## Median : 39.50 Median : 67.0 Median : 2.600e+04 Median :7.358e+05
## Mean : 88.19 Mean : 451.8 Mean : 7.110e+05 Mean :2.007e+07
## 3rd Qu.: 110.00 3rd Qu.: 371.0 3rd Qu.: 1.852e+05 3rd Qu.:5.240e+06
## Max. :25000.00 Max. :103757.0 Max. : 1.977e+09 Max. :1.632e+10
## NA's :1586306 NA's :77 NA's :121445
## VolumeWeightedAveragePrice Price AdjustedPrice
## Min. : -0.55 Min. : 0.00 Min. : 0
## 1st Qu.: 8.41 1st Qu.: 9.30 1st Qu.: 8
## Median : 35.49 Median : 40.00 Median : 27
## Mean : 77.56 Mean : 88.92 Mean : 20263
## 3rd Qu.: 99.48 3rd Qu.: 111.50 3rd Qu.: 77
## Max. :17500.00 Max. :25000.00 Max. :58612732
## NA's :116811
## Dividends LDividends CorpAdj DividendAdj
## Min. : 0.0000 Min. : 0.0000 Min. : 0 Min. :0.004756
## 1st Qu.: 0.0000 1st Qu.: 0.0000 1st Qu.: 1 1st Qu.:0.690736
## Median : 0.0000 Median : 0.0000 Median : 1 Median :0.920635
## Mean : 0.0121 Mean : 0.0121 Mean : 1636 Mean :0.812673
## 3rd Qu.: 0.0000 3rd Qu.: 0.0000 3rd Qu.: 1 3rd Qu.:1.000000
## Max. :869.3840 Max. :869.3840 Max. :1695150 Max. :1.000000
## NA's :931
## Currency NumberOfShares Exchange NOKPerForex
## Length:1872358 Min. :0.000e+00 Length:1872358 Min. :1
## Class :character 1st Qu.:1.395e+07 Class :character 1st Qu.:1
## Mode :character Median :4.386e+07 Mode :character Median :1
## Mean :1.487e+08 Mean :1
## 3rd Qu.:1.257e+08 3rd Qu.:1
## Max. :2.563e+10 Max. :1
## NA's :13973
## mktcap OSEBXmktshare_prevmnth OSEBXAlpha_prevmnth

```

```

## Min. :0.000e+00 Min. :0.0 Min. : -0.3
## 1st Qu.:3.383e+08 1st Qu.:0.0 1st Qu.: 0.0
## Median :1.103e+09 Median :0.0 Median : 0.0
## Mean :7.323e+09 Mean :0.0 Mean : 0.0
## 3rd Qu.:3.695e+09 3rd Qu.:0.0 3rd Qu.: 0.0
## Max. :1.290e+12 Max. :0.3 Max. : 0.8
## NA's :13973 NA's :1389220 NA's :1389295
## OSEBXBeta_prevmnth SMB HML LIQ
## Min. :-24.7 Min. :-0.1 Min. :-0.2 Min. :-0.2
## 1st Qu.: 0.4 1st Qu.: 0.0 1st Qu.: 0.0 1st Qu.: 0.0
## Median : 0.8 Median : 0.0 Median : 0.0 Median : 0.0
## Mean : 0.9 Mean : 0.0 Mean : 0.0 Mean : 0.0
## 3rd Qu.: 1.3 3rd Qu.: 0.0 3rd Qu.: 0.0 3rd Qu.: 0.0
## Max. : 30.8 Max. : 0.4 Max. : 0.8 Max. : 0.5
## NA's :1389295 NA's :334560 NA's :396392 NA's :334560
## MOM lnDeltaP lnDeltaOSEBX lnDeltaOBX
## Min. :-0.5 Min. :-5.8430 Min. :-0.685 Min. :-0.7
## 1st Qu.: 0.0 1st Qu.: -0.0132 1st Qu.: -0.005 1st Qu.: 0.0
## Median : 0.0 Median : 0.0000 Median : 0.000 Median : 0.0
## Mean : 0.0 Mean : -0.0004 Mean : 0.000 Mean : 0.0
## 3rd Qu.: 0.0 3rd Qu.: 0.0121 3rd Qu.: 0.006 3rd Qu.: 0.0
## Max. : 0.5 Max. : 6.8365 Max. : 1.776 Max. : 1.9
## NA's :334560 NA's :931 NA's :30218 NA's :332031
## NOWA_DayLnrate bills_3month_Lnrate Sector IN_OSEBX
## Min. :-4.445e-07 Min. :0 Length:1872358 Min. :0.000
## 1st Qu.: 6.530e-05 1st Qu.:0 Class :character 1st Qu.:0.000
## Median : 1.232e-04 Median :0 Mode :character Median :0.000
## Mean : 1.861e-04 Mean :0 Mean :0.263
## 3rd Qu.: 2.678e-04 3rd Qu.:0 3rd Qu.:1.000
## Max. : 2.586e-03 Max. :0 Max. :1.000
## NA's :93089
## Equity Debt Earnings debt_ratio
## Min. :-1.270e+11 Min. :0.000e+00 Min. :-6.352e+11 Min. : 0.0
## 1st Qu.: 1.840e+08 1st Qu.:1.606e+08 1st Qu.: -2.806e+07 1st Qu.: 0.4
## Median : 7.990e+08 Median :1.217e+09 Median : 3.722e+07 Median : 0.6
## Mean : 4.791e+11 Mean :5.748e+11 Mean : 2.146e+10 Mean : 0.6
## 3rd Qu.: 3.462e+09 3rd Qu.:7.982e+09 3rd Qu.: 3.418e+08 3rd Qu.: 0.7
## Max. : 1.015e+15 Max. :9.036e+14 Max. : 8.456e+13 Max. :159.1
## NA's :553474 NA's :695665 NA's :696329 NA's :696822
## PE ID MA5 MA10
## Min. :-1883157.9 Min. : 1 Min. : 0.00 Min. : 0.00
## 1st Qu.: -2.2 1st Qu.: 468090 1st Qu.: 31.01 1st Qu.: 39.73

```



```

## Median :      5.9   Median : 936180   Median :  60.60   Median :  64.91
## Mean   :     -7.3   Mean   : 936181   Mean   :  88.19   Mean   :  88.19
## 3rd Qu.:     17.3   3rd Qu.:1404269   3rd Qu.: 105.85   3rd Qu.: 103.63
## Max.   :  374153.9   Max.   :1872368   Max.   :5237.80   Max.   :3184.00
## NA's   :713263
##      MA20      BIAS5      BIAS10
## Min.   :    0.00   Min.   : -3.800e+15   Min.   : -5.142e+16
## 1st Qu.:   45.68   1st Qu.: -8.600e+01   1st Qu.: -8.700e+01
## Median :   67.75   Median : -3.700e+01   Median : -4.300e+01
## Mean   :   88.19   Mean   : -1.794e+12   Mean   :  3.608e+13
## 3rd Qu.:  100.08   3rd Qu.:  1.100e+02   3rd Qu.:  8.000e+01
## Max.   : 1885.15   Max.   :  2.682e+05   Max.   :  2.364e+17
## NA's   :10
##      RSI6      RSI12      Stoch_K      Stoch_D
## Min.   : -164100.00   Min.   : -36.67   Min.   : -6.667   Min.   :  0.00
## 1st Qu.:   30.00   1st Qu.:  36.84   1st Qu.: 22.222   1st Qu.: 25.69
## Median :   50.00   Median :  50.00   Median : 50.000   Median : 50.37
## Mean   :   53.57   Mean   :  49.81   Mean   : 50.379   Mean   : 50.38
## 3rd Qu.:   69.23   3rd Qu.:  62.92   3rd Qu.: 79.487   3rd Qu.: 75.33
## Max.   : 187528.57   Max.   :100.00   Max.   :114.286   Max.   :100.00
## NA's   :6      NA's   :12      NA's   :1595      NA's   :2159
##      MACD      Signal      WPR      VOL1
## Min.   : -100.00000   Min.   : -100.00000   Min.   : -106.67   Min.   :  0.00
## 1st Qu.:  -2.02671   1st Qu.:  -1.96772   1st Qu.:  -78.05   1st Qu.:  1.46
## Median :  -0.04564   Median :  -0.05985   Median :  -50.00   Median :  2.30
## Mean   :  -0.64041   Mean   :  -0.64044   Mean   :  -49.68   Mean   :  3.62
## 3rd Qu.:   1.73382   3rd Qu.:   1.64356   3rd Qu.:  -20.51   3rd Qu.:  3.69
## Max.   : 107.69231   Max.   :   67.88527   Max.   :   14.29   Max.   :311.86
## NA's   :25      NA's   :33      NA's   :1962      NA's   :80076
##      VOL2      Delta_MA5
## Min.   :    0.00   Min.   : -4984.500
## 1st Qu.:   68.84   1st Qu.:  -9.740
## Median : 109.65   Median :  -0.004
## Mean   : 125.58   Mean   :    0.000
## 3rd Qu.: 166.07   3rd Qu.:    9.900
## Max.   : 838.13   Max.   : 4980.500
## NA's   :80846   NA's   :1

```

```

# Load necessary libraries
library(quantmod)
stock_data_xts <- xts(data[, -1], order.by = as.Date(data$Date))

```

```

# Calculate Moving Averages
data$MA5 <- SMA(stock_data_xts$Close, n=5)
data$MA10 <- SMA(stock_data_xts$Close, n=10)
data$MA20 <- SMA(stock_data_xts$Close, n=20)

# Save data frame to a CSV file
#write.csv(data, "700mb.csv", row.names = FALSE)
# Load data frame from the CSV file
#data <- read.csv("700mb.csv")

#Remove rows with NAs in MA5 and MA10
data <- data[!is.na(data$MA5) & !is.na(data$MA10), ]

data$Close <- as.numeric(as.character(data$Close))
data$MA5 <- as.numeric(as.character(data$MA5))
# Check for unique values in 'Close' and 'MA5' columns
print(head(unique(data$Close), 10))

```

```
## [1] 75.30 75.00 74.99 79.79 78.00 77.98 79.00 76.99 77.00 78.80
```

```
print(head(unique(data$MA5), 10))
```

```
## [1] 370.0 457.0 367.0 341.0 320.5 205.5 98.5 99.0 99.5 229.0
```

```

# Now calculate BIAS
data$BIAS5 <- (data$Close - data$MA5) / data$MA5 * 100
data$BIAS10 <- (data$Close - data$MA10) / data$MA10 * 100

# Save data frame to a CSV file
#write.csv(data, "700mb.csv", row.names = FALSE)

```

```

library(zoo)
library(data.table)

```

```

calculate_RSI <- function(prices, n) {
  # Calculate daily changes in price
  deltas <- diff(prices)

  # Separate gains and losses
  gains <- pmax(deltas, 0)
  losses <- pmax(-deltas, 0)

  # Calculate the average gains and losses using a rolling mean
  avg_gains <- rollapply(gains, n, mean, fill = NA, align = 'right')
  avg_losses <- rollapply(losses, n, mean, fill = NA, align = 'right')

  # Calculate the RS (Relative Strength)
  rs <- avg_gains / avg_losses

  # Calculate the RSI (Relative Strength Index)
  rsi <- 100 - (100 / (1 + rs))

  return(rsi)
}

#
# Convert 'Close' column to numeric if it's not
#data[, Close := as.numeric(as.character(Close))]

# Calculate RSI for 6 periods

rsi6_values <- calculate_RSI(data$Close, 6)
# Prepend NA to the rsi6_values to match the length of the data
rsi6_values <- c(NA, rsi6_values)

# Now the lengths match and you can assign it to the data table
data$RSI6 <- rsi6_values

# Do the same for RSI12 if needed
rsi12_values <- c(NA, calculate_RSI(data$Close, 12))
data$RSI12 <- rsi12_values

# Save data frame to a CSV file
#write.csv(data, "700mb.csv", row.names = FALSE)

```

```

calculate_stoch <- function(high, low, close, n, smoothK, smoothD) {
  # Calculate %K
  lowest_low <- rollapply(low, n, min, fill = NA, align = 'right')
  highest_high <- rollapply(high, n, max, fill = NA, align = 'right')
  fastK <- (close - lowest_low) / (highest_high - lowest_low) * 100

  # Smooth %K to get the slow %K
  slowK <- rollapply(fastK, smoothK, mean, fill = NA, align = 'right')

  # Calculate %D as a moving average of %K
  fastD <- rollapply(slowK, smoothD, mean, fill = NA, align = 'right')

  return(list(fastK = fastK, fastD = fastD))
}

#
high_prices <- data$High
low_prices <- data$Low
close_prices <- data$Close

# Set the parameters
n <- 14      # The look-back period for %K
smoothK <- 3 # The smoothing period for %K
smoothD <- 3 # The smoothing period for %D

# Calculate stochastic oscillator
stoch_values <- calculate_stoch(high_prices, low_prices, close_prices, n, smoothK, smoothD)

# Add the stochastic values to data frame
data$Stoch_K <- stoch_values$fastK
data$Stoch_D <- stoch_values$fastD

# Save data frame to a CSV file
#write.csv(data, "700mb.csv", row.names = FALSE)

#load the TTR package
library(TTR)

```

```
# Convert the 'Close' column of data to numeric (if it's not already)
data$Close <- as.numeric(as.character(data$Close))

# Calculate MACD
macd_results <- MACD(x = data$Close)
print(head(macd_results, 30) ) # This will show the output structure
```

```
##           macd signal
## [1,]      NA      NA
## [2,]      NA      NA
## [3,]      NA      NA
## [4,]      NA      NA
## [5,]      NA      NA
## [6,]      NA      NA
## [7,]      NA      NA
## [8,]      NA      NA
## [9,]      NA      NA
## [10,]     NA      NA
## [11,]     NA      NA
## [12,]     NA      NA
## [13,]     NA      NA
## [14,]     NA      NA
## [15,]     NA      NA
## [16,]     NA      NA
## [17,]     NA      NA
## [18,]     NA      NA
## [19,]     NA      NA
## [20,]     NA      NA
## [21,]     NA      NA
## [22,]     NA      NA
## [23,]     NA      NA
## [24,]     NA      NA
## [25,]     NA      NA
## [26,]  2.444412     NA
## [27,]  2.467804     NA
## [28,]  2.308527     NA
## [29,]  2.108842     NA
## [30,]  1.976866     NA
```

```
print(class(macd_results)) # Check the class/type of macd_results
```

```
## [1] "matrix" "array"
```

```
data$MACD <- macd_results[, 1]  
data$Signal <- macd_results[, 2]
```

```
# Calculate Williams
```

```
calculate_WPR <- function(high, low, close, n) {  
  highest_high <- rollapply(high, n, max, fill = NA, align = 'right')  
  lowest_low <- rollapply(low, n, min, fill = NA, align = 'right')  
  wpr <- ((highest_high - close) / (highest_high - lowest_low)) * -100  
  return(wpr)  
}
```

```
#
```

```
library(zoo) # for rollapply
```

```
data$WPR <- calculate_WPR(data$High, data$Low, data$Close, 12)
```

```
# Calculate Volatility
```

```
calculate_volatility <- function(series, n) {  
  # Calculate the percentage change  
  pct_change <- diff(log(series)) * 100  
  
  # Calculate the rolling standard deviation (volatility)  
  vol <- rollapply(pct_change, width = n, FUN = sd, fill = NA, align = 'right')  
  
  return(vol)  
}
```

```
#data <- data[, !names(data) %in% c("Volume", "VOL1", "VOL2")]
```

```
colnames(data)
```

```
## [1] "Date" "Internal.code"
## [3] "SecurityId" "CompanyId"
## [5] "Symbol" "ISIN"
## [7] "Name" "BestBidPrice"
## [9] "BestAskPrice" "Open"
## [11] "High" "Low"
## [13] "Close" "NumberOfTrades"
## [15] "Volume" "Turnover"
## [17] "VolumeWeightedAveragePrice" "Price"
## [19] "AdjustedPrice" "Dividends"
## [21] "LDividends" "CorpAdj"
## [23] "DividendAdj" "Currency"
## [25] "NumberOfShares" "Exchange"
## [27] "NOKPerForex" "mktcap"
## [29] "OSEBXmktshare_prevmnth" "OSEBXAlpha_prevmnth"
## [31] "OSEBXBeta_prevmnth" "SMB"
## [33] "HML" "LIQ"
## [35] "MOM" "lnDeltaP"
## [37] "lnDeltaOSEBX" "lnDeltaOBX"
## [39] "NOWA_DayLnrate" "bills_3month_Lnrate"
## [41] "Sector" "IN_OSEBX"
## [43] "Equity" "Debt"
## [45] "Earnings" "debt_ratio"
## [47] "PE" "ID"
## [49] "MA5" "MA10"
## [51] "MA20" "BIAS5"
## [53] "BIAS10" "RSI6"
## [55] "RSI12" "Stoch_K"
## [57] "Stoch_D" "MACD"
## [59] "Signal" "WPR"
## [61] "VOL1" "VOL2"
## [63] "Delta_MA5"
```

```
library(zoo) # for rollapply
```

```
# Specify numeric columns
```

```
numeric_columns <- c("Open", "High", "Low", "Close", "OfficialVolume")
```

```
# Convert 'Close' and 'Volume' to numeric
```

```
data$Close <- as.numeric(as.character(data$Close))
```

```
data$Volume <- as.numeric(as.character(data$Volume))
```

```
# Prepend NA to the volatility vectors
```

```
vol1_values <- c(NA, calculate_volatility(data$Close, 10))
```

```
vol2_values <- c(NA, calculate_volatility(data$Volume, 10))
```

```
## Warning in log(series): NaNs produced
```

```
# Now assign the values to the data table
```

```
data$VOL1 <- vol1_values
```

```
data$VOL2 <- vol2_values
```

```
# Calculate Differential Technical Indicators ( $\Delta$ MA5: the change in MA5 from one period to the next)
```

```
# Add NA to the beginning to align lengths
```

```
data$Delta_MA5 <- c(NA, diff(data$MA5, lag = 1))
```

```
colnames(data)
```

```
## [1] "Date" "Internal.code"
## [3] "SecurityId" "CompanyId"
## [5] "Symbol" "ISIN"
## [7] "Name" "BestBidPrice"
## [9] "BestAskPrice" "Open"
## [11] "High" "Low"
## [13] "Close" "NumberOfTrades"
## [15] "Volume" "Turnover"
## [17] "VolumeWeightedAveragePrice" "Price"
## [19] "AdjustedPrice" "Dividends"
## [21] "LDividends" "CorpAdj"
## [23] "DividendAdj" "Currency"
## [25] "NumberOfShares" "Exchange"
## [27] "NOKPerForex" "mktcap"
## [29] "OSEBXmktshare_prevmnth" "OSEBXAlpha_prevmnth"
## [31] "OSEBXBeta_prevmnth" "SMB"
## [33] "HML" "LIQ"
## [35] "MOM" "lnDeltaP"
## [37] "lnDeltaOSEBX" "lnDeltaOBX"
## [39] "NOWA_DayLnrate" "bills_3month_Lnrate"
```



```
## [41] "Sector" "IN_OSEBX"
## [43] "Equity" "Debt"
## [45] "Earnings" "debt_ratio"
## [47] "PE" "ID"
## [49] "MA5" "MA10"
## [51] "MA20" "BIAS5"
## [53] "BIAS10" "RSI6"
## [55] "RSI12" "Stoch_K"
## [57] "Stoch_D" "MACD"
## [59] "Signal" "WPR"
## [61] "VOL1" "VOL2"
## [63] "Delta_MA5"
```

```
# Corrected list of columns to check for NAs
```

```
columns_to_check <- c("MA5", "MA10", "MA20", "BIAS5", "BIAS10", "RSI6", "RSI12", "Stoch_K", "Stoch_D", "MACD", "Signal", "WPR", "VOL1", "VOL2", "Delta_MA5")
```

```
# Subset the dataframe to keep only rows that are complete cases in the specified columns
```

```
data <- data[complete.cases(data[, columns_to_check]), ]
```

```
# Check the structure of the cleaned data
```

```
#str(data,10)
```

```
#unique(data$CompanyId)
```

```
#summary(data)
```

```
#typeof(data$Date)
```

```
# Defining the columns to keep
```

```
# Updated list of columns to keep
```

```
columns_to_keep <- c("Date", "CompanyId", "Close", "SMB", "HML", "LIQ", "MOM", "MA5", "MA10", "MA20", "BIAS5", "BIAS10", "RSI6", "RSI12",  
                    "Stoch_K", "Stoch_D", "MACD", "Signal", "WPR", "VOL1", "VOL2", "Delta_MA5" )
```

```
library(dplyr)
```

```
# Subset the dataframe to include only specified columns
```

```
data <- select(data, all_of(columns_to_keep))
```

```
# Optionally, check the structure of the new subset
print(head(unique(data$CompanyId), 10))
```

```
## [1] 12720 8380 12440 2017 8548 8408 12348 2407 7811 2202
```

```
summary(data)
```

```
##      Date      CompanyId      Close      SMB
## Length:1790443   Min.   : 1007   Min.   :    0.002   Min.   : -0.1
## Class :character 1st Qu.: 2231   1st Qu.:    9.350   1st Qu.:  0.0
## Mode  :character Median : 5203   Median :   40.810   Median :  0.0
##              Mean  : 5261   Mean  :   89.360   Mean   :  0.0
##              3rd Qu.: 7942   3rd Qu.:  111.500   3rd Qu.:  0.0
##              Max.   :12811   Max.   :25000.000   Max.   :  0.4
##              NA's   :70971   NA's   :321416
##      HML      LIQ      MOM      MA5
## Min.   : -0.2   Min.   : -0.2   Min.   : -0.5   Min.   :    0.00
## 1st Qu.:  0.0   1st Qu.:  0.0   1st Qu.:  0.0   1st Qu.:   31.00
## Median :  0.0   Median :  0.0   Median :  0.0   Median :   60.59
## Mean    :  0.0   Mean    :  0.0   Mean    :  0.0   Mean    :   88.33
## 3rd Qu.:  0.0   3rd Qu.:  0.0   3rd Qu.:  0.0   3rd Qu.:  105.90
## Max.    :  0.8   Max.    :  0.5   Max.    :  0.5   Max.    :5237.80
## NA's    :383123 NA's    :321416 NA's    :321416
##      MA10.Index      MA10.X[[i]]
## Min.   :1980-01-04   Min.   :    0.000
## 1st Qu.:1999-07-06   1st Qu.:   39.712
## Median :2009-05-13   Median :   64.897
## Mean    :2008-01-01   Mean    :   88.318
## 3rd Qu.:2017-10-26   3rd Qu.:  103.650
## Max.    :2024-04-04   Max.    :  3184.000
##
##      MA20.Index      MA20.X[[i]]      BIAS5
## Min.   :1980-01-04   Min.   :    0.0000   Min.   : -3.800e+15
## 1st Qu.:1999-07-06   1st Qu.:   45.6730   1st Qu.: -8.500e+01
## Median :2009-05-13   Median :   67.7370   Median : -3.500e+01
## Mean    :2008-01-01   Mean    :   88.3183   Mean    : -1.454e+12
## 3rd Qu.:2017-10-26   3rd Qu.:  100.0586   3rd Qu.:  1.140e+02
## Max.    :2024-04-04   Max.    :  1885.1500   Max.    :  2.905e+05
##
##      BIAS10.Index      BIAS10.X[[i]]
```

```

## Min.      :1980-01-04      Min.      :-514233130529000000
## 1st Qu.:1999-07-06      1st Qu.:      -87
## Median :2009-05-13      Median :      -41
## Mean    :2008-01-01      Mean     :    13760395599800
## 3rd Qu.:2017-10-26      3rd Qu.:      83
## Max.    :2024-04-04      Max.     :236438980437000000
##
##          RSI6          RSI12          Stoch_K          Stoch_D
## Min.    :-164100.00    Min.    :-36.67    Min.    : -6.667    Min.    :  0.00
## 1st Qu.:   29.41    1st Qu.:  36.36    1st Qu.: 22.059    1st Qu.: 25.00
## Median :   50.00    Median :  50.00    Median : 50.000    Median : 49.21
## Mean    :   52.66    Mean     :  49.89    Mean     : 49.443    Mean     : 49.59
## 3rd Qu.:   69.57    3rd Qu.:  63.46    3rd Qu.: 76.923    3rd Qu.: 74.32
## Max.    : 187528.57    Max.    :100.00    Max.    :114.286    Max.    :100.00
##
##          MACD          Signal          WPR          VOL1
## Min.    :-93.05972    Min.    :-92.71547    Min.    :-106.67    Min.    :  0.000
## 1st Qu.: -1.88572    1st Qu.: -1.80856    1st Qu.: -78.26    1st Qu.:  1.463
## Median : -0.01107    Median : -0.00711    Median : -50.00    Median :  2.300
## Mean    : -0.48776    Mean     : -0.48076    Mean     : -50.64    Mean     :  3.623
## 3rd Qu.:  1.72478    3rd Qu.:  1.66327    3rd Qu.: -23.08    3rd Qu.:  3.692
## Max.    :107.38287    Max.     : 67.88527    Max.     :  14.29    Max.     :283.504
##
##          VOL2          Delta_MA5
## Min.    :  0.00    Min.    :-4984.500
## 1st Qu.: 68.83    1st Qu.:  -9.732
## Median :109.62    Median :  -0.006
## Mean    :125.56    Mean     :  0.000
## 3rd Qu.:166.05    3rd Qu.:   9.900
## Max.    :838.13    Max.     : 4980.500
##

```

```
print(typeof(data$Date))
```

```
## [1] "character"
```

```
str(head(data, 10))
```

```

## 'data.frame':    10 obs. of  22 variables:
## $ Date      : chr  "2019-09-30" "2019-10-01" "2019-10-02" "2019-10-03" ...

```

```

## $ CompanyId: int 12720 12720 12720 12720 12720 12720 12720 12720 12720 12720
## $ Close      : num  83 83 81 80.8 80.8 81.8 82.4 83.4 81.6 82
## $ SMB       : num  -0.01322 0.00532 0.00465 0.01004 -0.01364 ...
## $ HML       : num  -0.00281 -0.00886 0.0235 -0.01351 -0.0041 ...
## $ LIQ       : num  -0.0037 -0.00532 0.00761 -0.00195 0.01229 ...
## $ MOM       : num  -1.01e-04 3.09e-05 1.06e-02 -7.09e-03 -1.02e-03 ...
## $ MA5       : num   92 102 121 128 144 ...
## $ MA10      :An xts object on 1980-01-04 / 1980-01-04 containing:
##   Data:      double [10, 1]
##   Index:     Date [10] (TZ: "UTC")
## $ MA20      :An xts object on 1980-01-04 / 1980-01-04 containing:
##   Data:      double [10, 1]
##   Index:     Date [10] (TZ: "UTC")
## $ BIAS5     : num  -9.78 -18.47 -32.95 -36.88 -43.81 ...
## $ BIAS10    :An xts object on 1980-01-04 / 1980-01-04 containing:
##   Data:      double [10, 1]
##   Index:     Date [10] (TZ: "UTC")
## $ RSI6      : num   59.3 62.1 50 65.6 0 ...
## $ RSI12     : num   60.5 58.5 45.9 45.5 40.2 ...
## $ Stoch_K   : num   75.9 70.2 40 40 40.1 ...
## $ Stoch_D   : num   67.9 74.4 73.3 62.8 50.7 ...
## $ MACD      : num   1.95 1.94 1.72 1.5 1.32 ...
## $ Signal    : num   2.06 2.04 1.97 1.88 1.77 ...
## $ WPR       : num  -33.3 -28.6 -57.1 -59.9 -52.5 ...
## $ VOL1      : num   2.95 2.95 2.98 2.94 2.89 ...
## $ VOL2      : num  130 119 112 120 160 ...
## $ Delta_MA5: num   25 9.8 19 7.2 15.8 ...

```

```
data <- read.csv("titlonmoyinreadyfinalsubset 17.5.2024.csv")
```

```
# Load necessary libraries
```

```
library(quantmod)
```

```
library(keras)
```

```
library(data.table)
```

```
library(keras)
```

```
library(tensorflow)
```

```
# Remove rows with any NA values from cnn_output dataframe
```

```
data <- na.omit(data)
```

```
# Select the columns to be used as features
```

```

features <- data[, c("Close", "SMB", "HML", "LIQ", "MOM", "MA5", "MA10", "MA20", "BIAS5", "BIAS10", "RSI6", "RSI1
2",
                    "Stoch_K", "Stoch_D", "MACD", "Signal", "WPR", "VOL1", "VOL2", "Delta_MA5")]

# Normalize the features
# We will perform min-max scaling here. You can also use other methods like Z-score standardization
min_vals <- sapply(features, min, na.rm = TRUE)
max_vals <- sapply(features, max, na.rm = TRUE)
scaled_features <- as.data.table(scale(features, center = min_vals, scale = max_vals - min_vals))

# Prepare the data for the CNN
# use a sequence length of 'n' days for each prediction
sequence_length <- 10 # using the past 10 days to predict the next day
n_features <- ncol(features)
n_samples <- nrow(features) - sequence_length + 1

# Initialize an array to hold the reshaped data
cnn_input <- array(NA, dim = c(n_samples, sequence_length, n_features))
dates <- data$Date[(11:(nrow(data) + 1))]

# Reshape the data into a 3D array
for(i in 1:n_samples) {
  cnn_input[i,,] <- as.matrix(scaled_features[i:(i + sequence_length - 1),])
}

# Now, cnn_input is ready to be used as an input for the CNN
# It has the shape: (number of samples, sequence length, number of features)
print(dim(cnn_input))

```

```
## [1] 1336811      10      20
```

```

# Prepare the output data (target variable)
# Shift the closing price by one time step to create the target output
cnn_output <- data$Close[(sequence_length + 1):nrow(data)]

# Make sure the length of cnn_output matches the number of samples in cnn_input
# The last value of 'Close' will not have a corresponding future value to predict, so we remove it
cnn_output <- cnn_output[1:n_samples]

# If the CNN model's output layer has one neuron, ensure the target is a matrix with one column

```

```
cnn_output <- matrix(cnn_output, ncol = 1)
print(length(cnn_output)) # Should match n_samples
```

```
## [1] 1336811
```

```
# Convert the list element 'Output' into a dataframe
cnn_output <- data.frame(cnn_output)

# Now add the 'dates' vector as a new column to the dataframe
cnn_output$Date <- dates

### check data
# Apply is.na() and is.infinite() to each element in the data frame
na_or_inf <- sapply(data, function(x) is.na(x) | is.infinite(x))

# Count the number of rows that have any NA or Inf
count_rows_with_na_or_inf <- sum(rowSums(na_or_inf) > 0)

# Print the result
print(paste("Number of rows with NA or Inf:", count_rows_with_na_or_inf))
```

```
## [1] "Number of rows with NA or Inf: 0"
```

```
# Check for NA, NaN, or Inf in each column
columns_with_issues <- sapply(data, function(x) any(is.na(x) | is.infinite(x)))

# Get names of columns with any NA, NaN, or Inf
columns_names_with_issues <- names(data)[columns_with_issues]

# Print the result
print(columns_names_with_issues)
```

```
## character(0)
```

```
# Identified columns with issues
columns_with_issues <- c("CompanyId", "SMB", "HML", "MOM")
```

```

# Count NA, NaN, or Inf in each identified column
count_issues_per_column <- sapply(data[columns_with_issues], function(x) sum(is.na(x) | is.infinite(x)))

# Print the counts
print(count_issues_per_column)

```

```

## CompanyId      SMB      HML      MOM
##           0           0           0           0

```

```

# Function to remove rows with any NaN values
remove_NaN_rows <- function(input, output) {
  nan_rows_input <- apply(is.na(input), c(1), any)
  nan_rows_output <- is.na(output)
  nan_rows <- nan_rows_input | nan_rows_output

  input_clean <- input[!nan_rows, , ]
  output_clean <- output[!nan_rows]

  return(list("input_clean" = input_clean, "output_clean" = output_clean))
}

# Remove NaN values from cnn_input and cnn_output
#clean_data <- remove_NaN_rows(cnn_input, cnn_output)
#cnn_input <- clean_data$input_clean
#cnn_output <- clean_data$output_clean

# Function to check for NaN, Inf, or -Inf values in an array
check_for_bad_values <- function(arr) {
  nan_count <- sum(is.na(arr))
  inf_count <- sum(arr == Inf)
  neg_inf_count <- sum(arr == -Inf)

  return(list("NaN Count" = nan_count, "Inf Count" = inf_count, "-Inf Count" = neg_inf_count))
}

cnn_output <- na.omit(cnn_output)

```

```
# Check for bad values in cnn_input
input_check <- check_for_bad_values(cnn_input)
print(input_check)
```

```
## $`NaN Count`
## [1] 0
##
## $`Inf Count`
## [1] 0
##
## $`-Inf Count`
## [1] 0
```

```
# Check for bad values in cnn_output
output_check <- check_for_bad_values(cnn_output)
print(output_check)
```

```
## $`NaN Count`
## [1] 0
##
## $`Inf Count`
## [1] 0
##
## $`-Inf Count`
## [1] 0
```

```
cnn_input<- cnn_input[1:1336810, , ]
```

```
# Now cnn_output is ready to be used as the target for training the CNN
```

```
# Save cnn_input
```

```
save(cnn_input, file = "cnn_input17.5.2024.RData")
save(cnn_output, file = "cnn_output17.5.2024.RData")
```

```
# Load cnn_input
```

```
load("cnn_input17.5.2024.RData")
load("cnn_output17.5.2024.RData")
```



```
# Now cnn_output is ready to be used as the target for training the CNN
```

```
###model running testing
```

```
# Load cnn_input
```

```
load("cnn_input17.5.2024.RData")
```

```
load("cnn_output17.5.2024.RData")
```

```
summary(cnn_input)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.06211 0.32636 0.36304 0.58998 1.00000
```

```
data <- read.csv("titlonmoyinreadyfinalsubset 17.5.2024.csv")
```

```
features <- data[, c("Close", "SMB", "HML", "LIQ", "MOM", "MA5", "MA10", "MA20", "BIAS5", "BIAS10", "RSI6", "RSI12",
                    "Stoch_K", "Stoch_D", "MACD", "Signal", "WPR", "VOL1", "VOL2", "Delta_MA5")]
```

```
# Prepare the data for the CNN
```

```
# use a sequence length of 'n' days for each prediction
```

```
sequence_length <- 10 # using the past 10 days to predict the next day
```

```
n_features <- ncol(features)
```

```
n_samples <- nrow(features) - sequence_length + 1
```

```
library(keras)
```

```
library(tidyverse)
```

```
## — Attaching core tidyverse packages ————— tidyverse 2.0.0 —
```

```
## ✓ forcats   1.0.0      ✓ readr     2.1.5
```

```
## ✓ ggplot2   3.5.1      ✓ stringr   1.5.1
```

```
## ✓ lubridate 1.9.3      ✓ tibble    3.2.1
```

```
## ✓ purrr     1.0.2      ✓ tidyr     1.3.1
```

```
## — Conflicts ————— tidyverse_conflicts() —
```

```
## ✗ dplyr::between() masks data.table::between()
```

```
## ✗ dplyr::filter()  masks stats::filter()
```

```
## ✗ xts::first()     masks dplyr::first(), data.table::first()
```

```
## ✗ lubridate::hour() masks data.table::hour()
```

```
## ✗ lubridate::isoweek() masks data.table::isoweek()
```

```
## ✖ dplyr::lag()           masks stats::lag()
## ✖ xts::last()           masks dplyr::last(), data.table::last()
## ✖ lubridate::mday()     masks data.table::mday()
## ✖ lubridate::minute()   masks data.table::minute()
## ✖ lubridate::month()    masks data.table::month()
## ✖ lubridate::quarter()  masks data.table::quarter()
## ✖ lubridate::second()   masks data.table::second()
## ✖ purrr::transpose()    masks data.table::transpose()
## ✖ lubridate::wday()     masks data.table::wday()
## ✖ lubridate::week()     masks data.table::week()
## ✖ lubridate::yday()     masks data.table::yday()
## ✖ lubridate::year()     masks data.table::year()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Specify TensorFlow version
install_keras(tensorflow = "2.16.1")
```

```
# Load necessary libraries
```

```
library(quantmod)
```

```
library(keras)
```

```
library(data.table)
```

```
library(tensorflow)
```

```
#install.packages("tensorflow")
```

```
library(keras)
```

```
# Adding 1D Convolutional layers with different kernel sizes as per the paper
```

```
# Note that 'input_shape' is set according to input data's dimensions (sequence_length, n_features)
```

```
# Define the model
```

```
model <- keras_model_sequential()
```

```
model2 <- keras_model_sequential() %>%
```

```
  layer_conv_1d(
```

```
    filters = 32,
```

```
    kernel_size = 3,
```

```
    activation = 'relu',
```

```
    input_shape = c(sequence_length, n_features),
```

```
    padding = "same",
```

```
    kernel_regularizer = regularizer_l2(0.01), # Adding L2 regularization
```

```
    kernel_initializer = initializer_glorot_uniform() # Changing the initializer
```

```

) %>%
layer_max_pooling_1d(pool_size = 2, strides = 2) %>%
layer_conv_1d(
  filters = 64,
  kernel_size = 3,
  activation = 'relu',
  padding = "same",
  kernel_regularizer = regularizer_l2(0.01), # Adding L2 regularization to another layer
  kernel_initializer = initializer_glorot_uniform() # Applying initializer here as well
) %>%
layer_max_pooling_1d(pool_size = 2, strides = 2) %>%
layer_flatten()

model2 %>% compile(
  loss = 'mse', # Mean Squared Error for regression tasks
  optimizer = optimizer_adam(learning_rate = 0.00005, clipnorm = 1), ## learnin rate nesf kardam 0.0001 bood
  metrics = c('mean_absolute_error', 'accuracy') # Combining metrics into one vector
)

# Print the model summary
summary(model2)

```

```

## Model: "sequential_1"
##
## Layer (type)                Output Shape                Param #
## =====
## conv1d_1 (Conv1D)           (None, 10, 32)              1952
## max_pooling1d_1 (MaxPooling1D) (None, 5, 32)              0
## conv1d (Conv1D)             (None, 5, 64)              6208
## max_pooling1d (MaxPooling1D) (None, 2, 64)              0
## flatten (Flatten)           (None, 128)                 0
## =====
## Total params: 8160 (31.88 KB)
## Trainable params: 8160 (31.88 KB)
## Non-trainable params: 0 (0.00 Byte)
##

```

```

# Train the model
history <- model2 %>% fit(

```

```

x = cnn_input,
y = cnn_output$cnn_output,
epochs = 10,
batch_size = 32,
validation_split = 0.2
)

```

```

## Epoch 1/10
## 33421/33421 - 98s - loss: 34143.5742 - mean_absolute_error: 67.7059 - accuracy: 0.0014 - val_loss: 17255.3828
- val_mean_absolute_error: 63.2252 - val_accuracy: 0.0018 - 98s/epoch - 3ms/step
## Epoch 2/10
## 33421/33421 - 136s - loss: 31293.7422 - mean_absolute_error: 65.2233 - accuracy: 0.0013 - val_loss: 15351.7783
- val_mean_absolute_error: 59.3631 - val_accuracy: 0.0017 - 136s/epoch - 4ms/step
## Epoch 3/10
## 33421/33421 - 144s - loss: 24956.6895 - mean_absolute_error: 57.5307 - accuracy: 0.0013 - val_loss: 10672.9082
- val_mean_absolute_error: 49.0417 - val_accuracy: 0.0016 - 144s/epoch - 4ms/step
## Epoch 4/10
## 33421/33421 - 81s - loss: 13519.9033 - mean_absolute_error: 40.8776 - accuracy: 0.0013 - val_loss: 4075.7043 -
val_mean_absolute_error: 28.5433 - val_accuracy: 0.0018 - 81s/epoch - 2ms/step
## Epoch 5/10
## 33421/33421 - 100s - loss: 3231.6775 - mean_absolute_error: 18.0687 - accuracy: 0.0013 - val_loss: 1003.8926 -
val_mean_absolute_error: 9.5267 - val_accuracy: 0.0020 - 100s/epoch - 3ms/step
## Epoch 6/10
## 33421/33421 - 133s - loss: 874.7887 - mean_absolute_error: 6.8794 - accuracy: 0.0015 - val_loss: 775.6360 - va
l_mean_absolute_error: 5.7688 - val_accuracy: 0.0021 - 133s/epoch - 4ms/step
## Epoch 7/10
## 33421/33421 - 141s - loss: 719.8345 - mean_absolute_error: 4.6062 - accuracy: 0.0015 - val_loss: 682.2665 - va
l_mean_absolute_error: 4.5168 - val_accuracy: 0.0017 - 141s/epoch - 4ms/step
## Epoch 8/10
## 33421/33421 - 146s - loss: 649.9177 - mean_absolute_error: 3.7987 - accuracy: 0.0014 - val_loss: 638.8582 - va
l_mean_absolute_error: 4.1336 - val_accuracy: 0.0017 - 146s/epoch - 4ms/step
## Epoch 9/10
## 33421/33421 - 98s - loss: 611.9424 - mean_absolute_error: 3.6009 - accuracy: 0.0014 - val_loss: 613.8934 - val
_mean_absolute_error: 3.9820 - val_accuracy: 0.0017 - 98s/epoch - 3ms/step
## Epoch 10/10
## 33421/33421 - 88s - loss: 589.1505 - mean_absolute_error: 3.5075 - accuracy: 0.0013 - val_loss: 596.9057 - val
_mean_absolute_error: 3.9482 - val_accuracy: 0.0017 - 88s/epoch - 3ms/step

```

Create a feature extraction model that outputs from the second convolutional layer

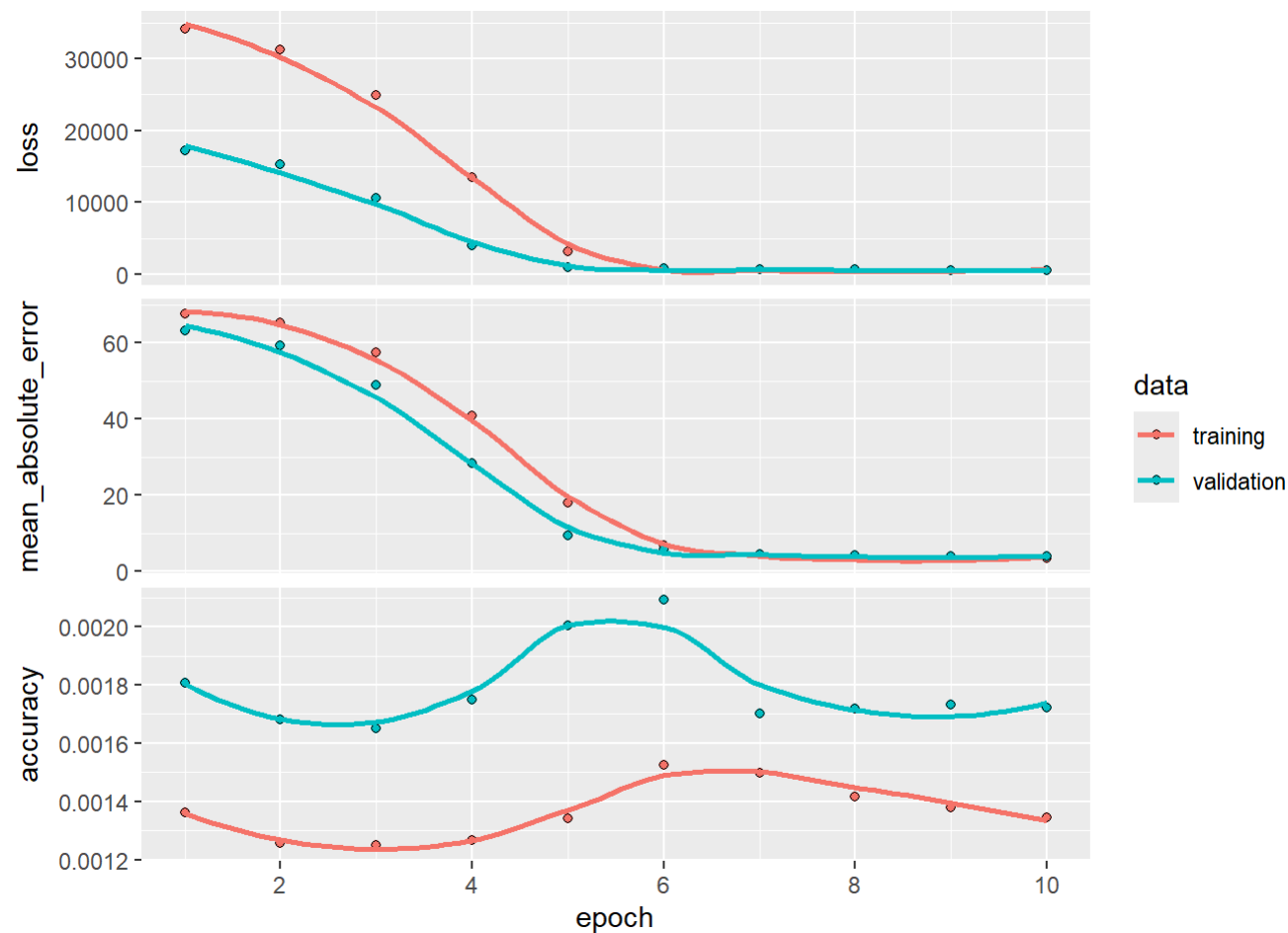
```
feature_extractor <- keras_model(inputs = model2$input, outputs = get_layer(model2, index = 2)$output)
```

```
# Now, use this model to predict features
extracted_features <- predict(feature_extractor, cnn_input)
```

```
## 41776/41776 - 56s - 56s/epoch - 1ms/step
```

```
# Saving the extracted features for later use
saveRDS(extracted_features, file = "extracted_features_medel2.19.5.2024fama frenchfinalgetbetter pca.rds")

# Plot training and validation loss
plot(history)
```



```
# Load the RDS file
extracted_features <- readRDS("extracted_features_medel2.19.5.2024fama frenchfinalgetbetter pca.rds")
```

```
# View the dimensions of the loaded features  
dim(extracted_features)
```

```
## [1] 1336810      5      32
```

```
# Check the summary statistics for the features  
summary(extracted_features)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.  
##      0.000   1.972   2.422   3.145   2.943  31.698
```

```
# If you want to see the first few rows to understand what the features look like  
head(extracted_features)
```

```
## , , 1  
##  
##      [,1]      [,2]      [,3]      [,4]      [,5]  
## [1,] 2.954318 2.902958 2.793275 2.851139 2.940864  
## [2,] 2.902958 2.842877 2.786640 2.940864 2.905286  
## [3,] 2.842877 2.793275 2.851139 2.940864 2.843683  
## [4,] 2.793275 2.786640 2.940864 2.905286 2.774110  
## [5,] 2.734892 2.851139 2.940864 2.843683 2.761003  
## [6,] 2.786640 2.940864 2.905286 2.774110 2.787590  
##  
## , , 2  
##  
##      [,1]      [,2]      [,3]      [,4]      [,5]  
## [1,] 2.319855 2.291306 2.190804 2.231504 2.316408  
## [2,] 2.291306 2.230147 2.184289 2.316408 2.312615  
## [3,] 2.230147 2.190804 2.231504 2.316408 2.278483  
## [4,] 2.190804 2.184289 2.316408 2.312615 2.237279  
## [5,] 2.116596 2.231504 2.316408 2.278483 2.191635  
## [6,] 2.184289 2.316408 2.312615 2.237279 2.210824  
##  
## , , 3
```

```

##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.781562 2.721928 2.652502 2.691606 2.728589
## [2,] 2.721928 2.682832 2.633554 2.728589 2.723169
## [3,] 2.682832 2.652502 2.691606 2.728589 2.633924
## [4,] 2.652502 2.633554 2.728589 2.723169 2.633364
## [5,] 2.587069 2.691606 2.728589 2.633924 2.598199
## [6,] 2.633554 2.728589 2.723169 2.633364 2.626759
##
## , , 4
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.767569 3.715847 3.565057 3.604253 3.722660
## [2,] 3.715847 3.624586 3.544922 3.722660 3.709620
## [3,] 3.624586 3.565057 3.604253 3.722660 3.590407
## [4,] 3.565057 3.544922 3.722660 3.709620 3.516551
## [5,] 3.429116 3.604253 3.722660 3.590407 3.412183
## [6,] 3.544922 3.722660 3.709620 3.516551 3.456850
##
## , , 5
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.848643 2.810842 2.705670 2.720306 2.800961
## [2,] 2.810842 2.754901 2.669765 2.800961 2.784938
## [3,] 2.754901 2.705670 2.720306 2.800961 2.735616
## [4,] 2.705670 2.669765 2.800961 2.784938 2.683870
## [5,] 2.604631 2.720306 2.800961 2.735616 2.633394
## [6,] 2.669765 2.800961 2.784938 2.683870 2.616972
##
## , , 6
##
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    0    0    0    0    0
## [3,]    0    0    0    0    0
## [4,]    0    0    0    0    0
## [5,]    0    0    0    0    0
## [6,]    0    0    0    0    0
##
## , , 7
##
##      [,1]      [,2]      [,3]      [,4]      [,5]

```

```
## [1,] 2.622634 2.567065 2.455908 2.480480 2.557527
## [2,] 2.567065 2.499005 2.424078 2.557527 2.561853
## [3,] 2.499005 2.455908 2.480480 2.561853 2.502497
## [4,] 2.455908 2.424078 2.557527 2.561853 2.450645
## [5,] 2.389630 2.480480 2.561853 2.502497 2.395728
## [6,] 2.424078 2.557527 2.561853 2.450645 2.441107
##
## , , 8
##
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.050714 2.996390 2.893982 2.919470 2.971909
## [2,] 2.996390 2.934685 2.870043 2.971909 2.985106
## [3,] 2.934685 2.893982 2.919470 2.985106 2.924108
## [4,] 2.893982 2.870043 2.971909 2.985106 2.913815
## [5,] 2.812239 2.919470 2.985106 2.924108 2.867736
## [6,] 2.870043 2.971909 2.985106 2.913815 2.896150
##
## , , 9
##
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.6281550 0.6120690 0.5777699 0.5939625 0.6120415
## [2,] 0.6120690 0.5933056 0.5932828 0.6120415 0.5913891
## [3,] 0.5933056 0.5777699 0.5939625 0.6120415 0.5878857
## [4,] 0.5777699 0.5932828 0.6120415 0.5913891 0.5689480
## [5,] 0.5409214 0.5939625 0.6120415 0.5878857 0.5353054
## [6,] 0.5932828 0.6120415 0.5913891 0.5689480 0.5662389
##
## , , 10
##
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.129924 3.078132 2.966232 3.002552 3.111548
## [2,] 3.078132 3.003875 2.966014 3.111548 3.087477
## [3,] 3.003875 2.966232 3.002552 3.111548 2.995544
## [4,] 2.966232 2.966014 3.111548 3.087477 2.954662
## [5,] 2.831478 3.002552 3.111548 2.995544 2.894800
## [6,] 2.966014 3.111548 3.087477 2.954662 2.913656
##
## , , 11
##
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.208476 2.156288 2.093285 2.190548 2.211378
## [2,] 2.156288 2.107503 2.144052 2.211378 2.172805
```



```
## [3,] 2.107503 2.093285 2.190548 2.211378 2.090384
## [4,] 2.093285 2.144052 2.211378 2.172805 2.065701
## [5,] 2.010753 2.190548 2.211378 2.090384 2.064604
## [6,] 2.144052 2.211378 2.172805 2.065701 2.084885
##
## , , 12
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.087142 3.062483 2.960894 2.979788 3.060022
## [2,] 3.062483 3.002443 2.915417 3.060022 3.074085
## [3,] 3.002443 2.960894 2.979788 3.074085 3.010867
## [4,] 2.960894 2.915417 3.060022 3.074085 2.962034
## [5,] 2.876361 2.979788 3.074085 3.010867 2.900420
## [6,] 2.915417 3.060022 3.074085 2.962034 2.883407
##
## , , 13
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.740625 2.671417 2.627221 2.713439 2.699677
## [2,] 2.671417 2.636334 2.659643 2.713439 2.694354
## [3,] 2.636334 2.627221 2.713439 2.699677 2.594595
## [4,] 2.627221 2.659643 2.713439 2.694354 2.566846
## [5,] 2.526364 2.713439 2.699677 2.594595 2.559771
## [6,] 2.659643 2.713439 2.694354 2.566846 2.595991
##
## , , 14
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.641452 2.617517 2.521078 2.516832 2.615083
## [2,] 2.617517 2.565220 2.465686 2.615083 2.623219
## [3,] 2.565220 2.521078 2.516832 2.623219 2.589463
## [4,] 2.521078 2.465686 2.615083 2.623219 2.528605
## [5,] 2.427723 2.516832 2.623219 2.589463 2.490943
## [6,] 2.465686 2.615083 2.623219 2.528605 2.498967
##
## , , 15
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.967037 2.923476 2.792659 2.813413 2.898127
## [2,] 2.923476 2.842431 2.754575 2.898127 2.871614
## [3,] 2.842431 2.792659 2.813413 2.898127 2.831438
## [4,] 2.792659 2.754575 2.898127 2.871614 2.771378
```

```
## [5,] 2.700878 2.813413 2.898127 2.831438 2.705415
## [6,] 2.754575 2.898127 2.871614 2.771378 2.688599
##
## , , 16
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.345538 3.304990 3.204113 3.254920 3.354094
## [2,] 3.304990 3.251389 3.179498 3.354094 3.321936
## [3,] 3.251389 3.204113 3.254920 3.354094 3.241472
## [4,] 3.204113 3.179498 3.354094 3.321936 3.211694
## [5,] 3.123771 3.254920 3.354094 3.241472 3.179174
## [6,] 3.179498 3.354094 3.321936 3.211694 3.219393
##
## , , 17
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.056042 3.017359 2.893963 2.954183 3.041197
## [2,] 3.017359 2.936780 2.892976 3.041197 3.017299
## [3,] 2.936780 2.893963 2.954183 3.041197 2.967087
## [4,] 2.893963 2.892976 3.041197 3.017299 2.915750
## [5,] 2.828662 2.954183 3.041197 2.967087 2.870609
## [6,] 2.892976 3.041197 3.017299 2.915750 2.902232
##
## , , 18
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.251759 3.197350 3.118888 3.119411 3.178431
## [2,] 3.197350 3.147736 3.061527 3.178431 3.168839
## [3,] 3.147736 3.118888 3.119411 3.178431 3.119441
## [4,] 3.118888 3.061527 3.178431 3.168839 3.077954
## [5,] 3.044368 3.119411 3.178431 3.119441 3.077577
## [6,] 3.061527 3.178431 3.168839 3.077954 3.065518
##
## , , 19
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.873053 2.843155 2.719352 2.771043 2.870885
## [2,] 2.843155 2.763168 2.734197 2.870885 2.819052
## [3,] 2.763168 2.719352 2.771043 2.870885 2.773652
## [4,] 2.719352 2.734197 2.870885 2.819052 2.722332
## [5,] 2.640645 2.771043 2.870885 2.773652 2.643290
## [6,] 2.734197 2.870885 2.819052 2.722332 2.688293
```

```
##
## , , 20
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.527283 1.493188 1.419245 1.431216 1.490720
## [2,] 1.493188 1.450546 1.396144 1.490720 1.479255
## [3,] 1.450546 1.419245 1.431216 1.490720 1.444307
## [4,] 1.419245 1.396144 1.490720 1.479255 1.403357
## [5,] 1.335613 1.431216 1.490720 1.444307 1.332538
## [6,] 1.396144 1.490720 1.479255 1.403357 1.317446
##
## , , 21
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.6173271 0.5984175 0.5602492 0.5740771 0.5869714
## [2,] 0.5984175 0.5719535 0.5740771 0.5869714 0.5929846
## [3,] 0.5719535 0.5602492 0.5740771 0.5929846 0.5765064
## [4,] 0.5602492 0.5740771 0.5869714 0.5929846 0.5503516
## [5,] 0.5377978 0.5740771 0.5929846 0.5765064 0.5348529
## [6,] 0.5740771 0.5869714 0.5929846 0.5503516 0.5501267
##
## , , 22
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.141166 3.113188 3.000761 3.011733 3.101300
## [2,] 3.113188 3.055427 2.973862 3.101300 3.096423
## [3,] 3.055427 3.000761 3.011733 3.101300 3.066905
## [4,] 3.000761 2.973862 3.101300 3.096423 2.975310
## [5,] 2.902640 3.011733 3.101300 3.066905 2.900775
## [6,] 2.973862 3.101300 3.096423 2.975310 2.898582
##
## , , 23
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.973756 2.938513 2.828700 2.846973 2.920516
## [2,] 2.938513 2.881334 2.801356 2.920516 2.929999
## [3,] 2.881334 2.828700 2.846973 2.929999 2.884977
## [4,] 2.828700 2.801356 2.920516 2.929999 2.833885
## [5,] 2.735139 2.846973 2.929999 2.884977 2.768606
## [6,] 2.801356 2.920516 2.929999 2.833885 2.806856
##
## , , 24
```

```
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 16.91631 16.76044 15.97400 16.04786 16.66906
## [2,] 16.76044 16.33565 15.75999 16.66906 16.59522
## [3,] 16.33565 15.97400 16.04786 16.66906 16.36987
## [4,] 15.97400 15.75999 16.66906 16.59522 16.04740
## [5,] 15.57182 16.04786 16.66906 16.36987 15.63823
## [6,] 15.75999 16.66906 16.59522 16.04740 15.65354
##
## , , 25
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 3.207378 3.185500 3.056593 3.116370 3.208179
## [2,] 3.185500 3.107914 3.038446 3.208179 3.208334
## [3,] 3.107914 3.056593 3.116370 3.208334 3.147545
## [4,] 3.056593 3.038446 3.208179 3.208334 3.082471
## [5,] 2.943789 3.116370 3.208334 3.147545 3.003130
## [6,] 3.038446 3.208179 3.208334 3.082471 3.024259
##
## , , 26
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.986640 1.966478 1.906163 1.918697 1.970094
## [2,] 1.966478 1.929324 1.892851 1.970094 1.959148
## [3,] 1.929324 1.906163 1.918697 1.970094 1.932972
## [4,] 1.906163 1.892851 1.970094 1.959148 1.899286
## [5,] 1.867426 1.918697 1.970094 1.932972 1.864228
## [6,] 1.892851 1.970094 1.959148 1.899286 1.874582
##
## , , 27
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.622403 2.588847 2.483687 2.510450 2.580549
## [2,] 2.588847 2.521481 2.472068 2.580549 2.573039
## [3,] 2.521481 2.483687 2.510450 2.580549 2.523389
## [4,] 2.483687 2.472068 2.580549 2.573039 2.471047
## [5,] 2.425348 2.510450 2.580549 2.523389 2.432238
## [6,] 2.472068 2.580549 2.573039 2.471047 2.427258
##
## , , 28
##
##      [,1]      [,2]      [,3]      [,4]      [,5]
```

```
## [1,] 3.557818 3.509318 3.392532 3.453823 3.524165
## [2,] 3.509318 3.451707 3.402643 3.524165 3.515938
## [3,] 3.451707 3.392532 3.453823 3.524165 3.444752
## [4,] 3.392532 3.402643 3.524165 3.515938 3.386801
## [5,] 3.290368 3.453823 3.524165 3.444752 3.316172
## [6,] 3.402643 3.524165 3.515938 3.386801 3.320003
##
## , , 29
##
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.812152 2.782400 2.695077 2.693002 2.757360
## [2,] 2.782400 2.735908 2.665402 2.757360 2.745985
## [3,] 2.735908 2.695077 2.693002 2.757360 2.702783
## [4,] 2.695077 2.665402 2.757360 2.745985 2.659160
## [5,] 2.624502 2.693002 2.757360 2.702783 2.614398
## [6,] 2.665402 2.757360 2.745985 2.659160 2.628008
##
## , , 30
##
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 16.58248 16.42806 15.62566 15.67463 16.31684
## [2,] 16.42806 15.99365 15.37027 16.31684 16.24078
## [3,] 15.99365 15.62566 15.67463 16.31684 16.01666
## [4,] 15.62566 15.37027 16.31684 16.24078 15.67296
## [5,] 15.20001 15.67463 16.31684 16.01666 15.22410
## [6,] 15.37027 16.31684 16.24078 15.67296 15.21288
##
## , , 31
##
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.918725 2.873864 2.743254 2.755913 2.849358
## [2,] 2.873864 2.805976 2.721694 2.849358 2.854462
## [3,] 2.805976 2.743254 2.755913 2.854462 2.795372
## [4,] 2.743254 2.721694 2.849358 2.854462 2.712779
## [5,] 2.670018 2.755913 2.854462 2.795372 2.631255
## [6,] 2.721694 2.849358 2.854462 2.712779 2.669034
##
## , , 32
##
##          [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 2.883401 2.850788 2.747334 2.824714 2.895210
## [2,] 2.850788 2.791325 2.769894 2.895210 2.865869
```

```
## [3,] 2.791325 2.747334 2.824714 2.895210 2.797762
## [4,] 2.747334 2.769894 2.895210 2.865869 2.758550
## [5,] 2.690532 2.824714 2.895210 2.797762 2.714154
## [6,] 2.769894 2.895210 2.865869 2.758550 2.730249
```

```
#####PCA TEST #####
```

```
# Flatten the array into a 2D matrix
```

```
flattened_data <- array(extracted_features, dim = c(dim(extracted_features)[1], dim(extracted_features)[2] * dim(extracted_features)[3]))
```

```
summary(flattened_data)## dataha too ranega an scaleled ok
```

```
##      V1      V2      V3      V4
## Min.   : 1.545 Min.   : 1.550 Min.   : 1.550 Min.   : 1.550
## 1st Qu.: 2.265 1st Qu.: 2.290 1st Qu.: 2.290 1st Qu.: 2.290
## Median : 2.483 Median : 2.506 Median : 2.506 Median : 2.506
## Mean   : 2.676 Mean   : 2.700 Mean   : 2.700 Mean   : 2.700
## 3rd Qu.: 2.806 3rd Qu.: 2.830 3rd Qu.: 2.830 3rd Qu.: 2.830
## Max.   :30.945 Max.   :30.945 Max.   :30.945 Max.   :30.945
##      V5      V6      V7      V8
## Min.   : 1.545 Min.   : 1.032 Min.   : 1.070 Min.   : 1.070
## 1st Qu.: 2.265 1st Qu.: 1.608 1st Qu.: 1.635 1st Qu.: 1.635
## Median : 2.483 Median : 1.862 Median : 1.887 Median : 1.887
## Mean   : 2.676 Mean   : 2.039 Mean   : 2.064 Mean   : 2.064
## 3rd Qu.: 2.806 3rd Qu.: 2.188 3rd Qu.: 2.211 3rd Qu.: 2.211
## Max.   :30.945 Max.   :30.287 Max.   :30.287 Max.   :30.287
##      V9      V10     V11     V12
## Min.   : 1.070 Min.   : 1.032 Min.   : 1.572 Min.   : 1.575
## 1st Qu.: 1.635 1st Qu.: 1.608 1st Qu.: 2.089 1st Qu.: 2.110
## Median : 1.887 Median : 1.862 Median : 2.289 Median : 2.309
## Mean   : 2.064 Mean   : 2.039 Mean   : 2.496 Mean   : 2.516
## 3rd Qu.: 2.211 3rd Qu.: 2.188 3rd Qu.: 2.615 3rd Qu.: 2.636
## Max.   :30.287 Max.   :30.287 Max.   :31.147 Max.   :31.147
##      V13     V14     V15     V16
## Min.   : 1.575 Min.   : 1.575 Min.   : 1.572 Min.   : 2.182
## 1st Qu.: 2.110 1st Qu.: 2.110 1st Qu.: 2.089 1st Qu.: 2.904
## Median : 2.309 Median : 2.309 Median : 2.289 Median : 3.219
```

##	Mean	: 2.516	Mean	: 2.516	Mean	: 2.496	Mean	: 3.375				
##	3rd Qu.:	2.636	3rd Qu.:	2.636	3rd Qu.:	2.615	3rd Qu.:	3.573				
##	Max.	:31.147	Max.	:31.147	Max.	:31.147	Max.	:31.698				
##	V17		V18		V19		V20					
##	Min.	: 2.186	Min.	: 2.186	Min.	: 2.186	Min.	: 2.182				
##	1st Qu.:	2.942	1st Qu.:	2.942	1st Qu.:	2.942	1st Qu.:	2.904				
##	Median	: 3.257	Median	: 3.257	Median	: 3.257	Median	: 3.219				
##	Mean	: 3.410	Mean	: 3.410	Mean	: 3.410	Mean	: 3.375				
##	3rd Qu.:	3.605	3rd Qu.:	3.605	3rd Qu.:	3.605	3rd Qu.:	3.573				
##	Max.	:31.698	Max.	:31.698	Max.	:31.698	Max.	:31.698				
##	V21		V22		V23		V24					
##	Min.	: 1.310	Min.	: 1.314	Min.	: 1.314	Min.	: 1.314				
##	1st Qu.:	2.044	1st Qu.:	2.074	1st Qu.:	2.074	1st Qu.:	2.074				
##	Median	: 2.351	Median	: 2.381	Median	: 2.381	Median	: 2.381				
##	Mean	: 2.514	Mean	: 2.543	Mean	: 2.543	Mean	: 2.543				
##	3rd Qu.:	2.707	3rd Qu.:	2.733	3rd Qu.:	2.733	3rd Qu.:	2.733				
##	Max.	:30.948	Max.	:30.948	Max.	:30.948	Max.	:30.948				
##	V25		V26		V27		V28		V29		V30	
##	Min.	: 1.310	Min.	:0	Min.	:0	Min.	:0	Min.	:0	Min.	:0
##	1st Qu.:	2.044	1st Qu.:	:0	1st Qu.:	:0	1st Qu.:	:0	1st Qu.:	:0	1st Qu.:	:0
##	Median	: 2.351	Median	:0	Median	:0	Median	:0	Median	:0	Median	:0
##	Mean	: 2.514	Mean	:0	Mean	:0	Mean	:0	Mean	:0	Mean	:0
##	3rd Qu.:	2.707	3rd Qu.:	:0	3rd Qu.:	:0	3rd Qu.:	:0	3rd Qu.:	:0	3rd Qu.:	:0
##	Max.	:30.948	Max.	:0	Max.	:0	Max.	:0	Max.	:0	Max.	:0
##	V31		V32		V33		V34					
##	Min.	: 1.250	Min.	: 1.274	Min.	: 1.274	Min.	: 1.274				
##	1st Qu.:	1.847	1st Qu.:	1.874	1st Qu.:	1.874	1st Qu.:	1.874				
##	Median	: 2.109	Median	: 2.135	Median	: 2.135	Median	: 2.135				
##	Mean	: 2.286	Mean	: 2.312	Mean	: 2.312	Mean	: 2.312				
##	3rd Qu.:	2.442	3rd Qu.:	2.467	3rd Qu.:	2.467	3rd Qu.:	2.467				
##	Max.	:30.819	Max.	:30.819	Max.	:30.819	Max.	:30.819				
##	V35		V36		V37		V38					
##	Min.	: 1.250	Min.	: 1.258	Min.	: 1.273	Min.	: 1.273				
##	1st Qu.:	1.847	1st Qu.:	2.188	1st Qu.:	2.221	1st Qu.:	2.221				
##	Median	: 2.109	Median	: 2.509	Median	: 2.541	Median	: 2.541				
##	Mean	: 2.286	Mean	: 2.665	Mean	: 2.695	Mean	: 2.695				
##	3rd Qu.:	2.442	3rd Qu.:	2.871	3rd Qu.:	2.899	3rd Qu.:	2.899				
##	Max.	:30.819	Max.	:31.167	Max.	:31.167	Max.	:31.167				
##	V39		V40		V41		V42					
##	Min.	: 1.273	Min.	: 1.258	Min.	: 0.0000	Min.	: 0.0000				
##	1st Qu.:	2.221	1st Qu.:	2.188	1st Qu.:	0.1014	1st Qu.:	0.1105				
##	Median	: 2.541	Median	: 2.509	Median	: 0.2445	Median	: 0.2543				

##	Mean	: 2.695	Mean	: 2.665	Mean	: 0.4946	Mean	: 0.5045
##	3rd Qu.:	2.899	3rd Qu.:	2.871	3rd Qu.:	0.5923	3rd Qu.:	0.6034
##	Max.	:31.167	Max.	:31.167	Max.	:29.0143	Max.	:29.0143
##	V43		V44		V45		V46	
##	Min.	: 0.0000	Min.	: 0.0000	Min.	: 0.0000	Min.	: 1.517
##	1st Qu.:	0.1105	1st Qu.:	0.1105	1st Qu.:	0.1010	1st Qu.:	2.287
##	Median	: 0.2543	Median	: 0.2543	Median	: 0.2444	Median	: 2.603
##	Mean	: 0.5045	Mean	: 0.5045	Mean	: 0.4943	Mean	: 2.764
##	3rd Qu.:	0.6034	3rd Qu.:	0.6034	3rd Qu.:	0.5922	3rd Qu.:	2.961
##	Max.	:29.0143	Max.	:29.0143	Max.	:29.0143	Max.	:31.517
##	V47		V48		V49		V50	
##	Min.	: 1.530	Min.	: 1.530	Min.	: 1.530	Min.	: 1.517
##	1st Qu.:	2.324	1st Qu.:	2.324	1st Qu.:	2.324	1st Qu.:	2.287
##	Median	: 2.639	Median	: 2.639	Median	: 2.639	Median	: 2.603
##	Mean	: 2.798	Mean	: 2.798	Mean	: 2.798	Mean	: 2.764
##	3rd Qu.:	2.992	3rd Qu.:	2.992	3rd Qu.:	2.992	3rd Qu.:	2.961
##	Max.	:31.517	Max.	:31.517	Max.	:31.517	Max.	:31.517
##	V51		V52		V53		V54	
##	Min.	: 0.6825	Min.	: 0.6872	Min.	: 0.6872	Min.	: 0.6872
##	1st Qu.:	1.5748	1st Qu.:	1.5986	1st Qu.:	1.5986	1st Qu.:	1.5986
##	Median	: 1.7743	Median	: 1.7946	Median	: 1.7946	Median	: 1.7946
##	Mean	: 1.9826	Mean	: 2.0048	Mean	: 2.0048	Mean	: 2.0048
##	3rd Qu.:	2.1049	3rd Qu.:	2.1269	3rd Qu.:	2.1269	3rd Qu.:	2.1269
##	Max.	:30.5721	Max.	:30.5721	Max.	:30.5721	Max.	:30.5721
##	V55		V56		V57		V58	
##	Min.	: 0.6825	Min.	: 1.614	Min.	: 1.616	Min.	: 1.616
##	1st Qu.:	1.5748	1st Qu.:	2.324	1st Qu.:	2.353	1st Qu.:	2.353
##	Median	: 1.7743	Median	: 2.613	Median	: 2.641	Median	: 2.641
##	Mean	: 1.9826	Mean	: 2.776	Mean	: 2.803	Mean	: 2.803
##	3rd Qu.:	2.1049	3rd Qu.:	2.951	3rd Qu.:	2.976	3rd Qu.:	2.976
##	Max.	:30.5721	Max.	:30.770	Max.	:30.770	Max.	:30.770
##	V59		V60		V61		V62	
##	Min.	: 1.616	Min.	: 1.614	Min.	: 1.294	Min.	: 1.302
##	1st Qu.:	2.353	1st Qu.:	2.324	1st Qu.:	2.078	1st Qu.:	2.105
##	Median	: 2.641	Median	: 2.613	Median	: 2.293	Median	: 2.317
##	Mean	: 2.803	Mean	: 2.776	Mean	: 2.490	Mean	: 2.515
##	3rd Qu.:	2.976	3rd Qu.:	2.951	3rd Qu.:	2.619	3rd Qu.:	2.644
##	Max.	:30.770	Max.	:30.770	Max.	:30.842	Max.	:30.842
##	V63		V64		V65		V66	
##	Min.	: 1.302	Min.	: 1.302	Min.	: 1.294	Min.	: 1.197
##	1st Qu.:	2.105	1st Qu.:	2.105	1st Qu.:	2.078	1st Qu.:	1.858
##	Median	: 2.317	Median	: 2.317	Median	: 2.293	Median	: 2.159

##	Mean	: 2.515	Mean	: 2.515	Mean	: 2.490	Mean	: 2.322
##	3rd Qu.:	2.644	3rd Qu.:	2.644	3rd Qu.:	2.619	3rd Qu.:	2.508
##	Max.	:30.842	Max.	:30.842	Max.	:30.842	Max.	:30.849
##	V67		V68		V69		V70	
##	Min.	: 1.232	Min.	: 1.232	Min.	: 1.232	Min.	: 1.197
##	1st Qu.:	1.888	1st Qu.:	1.888	1st Qu.:	1.888	1st Qu.:	1.858
##	Median	: 2.189	Median	: 2.189	Median	: 2.189	Median	: 2.159
##	Mean	: 2.350	Mean	: 2.350	Mean	: 2.350	Mean	: 2.322
##	3rd Qu.:	2.533	3rd Qu.:	2.533	3rd Qu.:	2.533	3rd Qu.:	2.508
##	Max.	:30.849	Max.	:30.849	Max.	:30.849	Max.	:30.849
##	V71		V72		V73		V74	
##	Min.	: 1.427	Min.	: 1.432	Min.	: 1.432	Min.	: 1.432
##	1st Qu.:	2.121	1st Qu.:	2.153	1st Qu.:	2.153	1st Qu.:	2.153
##	Median	: 2.430	Median	: 2.461	Median	: 2.461	Median	: 2.461
##	Mean	: 2.592	Mean	: 2.622	Mean	: 2.622	Mean	: 2.622
##	3rd Qu.:	2.787	3rd Qu.:	2.814	3rd Qu.:	2.814	3rd Qu.:	2.814
##	Max.	:31.179	Max.	:31.179	Max.	:31.179	Max.	:31.179
##	V75		V76		V77		V78	
##	Min.	: 1.427	Min.	: 2.012	Min.	: 2.018	Min.	: 2.018
##	1st Qu.:	2.121	1st Qu.:	2.647	1st Qu.:	2.674	1st Qu.:	2.674
##	Median	: 2.430	Median	: 2.883	Median	: 2.909	Median	: 2.909
##	Mean	: 2.592	Mean	: 3.064	Mean	: 3.090	Mean	: 3.090
##	3rd Qu.:	2.787	3rd Qu.:	3.202	3rd Qu.:	3.227	3rd Qu.:	3.227
##	Max.	:31.179	Max.	:31.124	Max.	:31.124	Max.	:31.124
##	V79		V80		V81		V82	
##	Min.	: 2.018	Min.	: 2.012	Min.	: 1.714	Min.	: 1.722
##	1st Qu.:	2.674	1st Qu.:	2.647	1st Qu.:	2.341	1st Qu.:	2.368
##	Median	: 2.909	Median	: 2.883	Median	: 2.579	Median	: 2.604
##	Mean	: 3.090	Mean	: 3.064	Mean	: 2.763	Mean	: 2.788
##	3rd Qu.:	3.227	3rd Qu.:	3.202	3rd Qu.:	2.902	3rd Qu.:	2.926
##	Max.	:31.124	Max.	:31.124	Max.	:30.970	Max.	:30.970
##	V83		V84		V85		V86	
##	Min.	: 1.722	Min.	: 1.722	Min.	: 1.714	Min.	: 1.712
##	1st Qu.:	2.368	1st Qu.:	2.368	1st Qu.:	2.341	1st Qu.:	2.472
##	Median	: 2.604	Median	: 2.604	Median	: 2.579	Median	: 2.731
##	Mean	: 2.788	Mean	: 2.788	Mean	: 2.763	Mean	: 2.909
##	3rd Qu.:	2.926	3rd Qu.:	2.926	3rd Qu.:	2.902	3rd Qu.:	3.068
##	Max.	:30.970	Max.	:30.970	Max.	:30.970	Max.	:31.394
##	V87		V88		V89		V90	
##	Min.	: 1.760	Min.	: 1.760	Min.	: 1.760	Min.	: 1.712
##	1st Qu.:	2.497	1st Qu.:	2.497	1st Qu.:	2.497	1st Qu.:	2.472
##	Median	: 2.756	Median	: 2.756	Median	: 2.756	Median	: 2.731

##	Mean : 2.933	Mean : 2.933	Mean : 2.933	Mean : 2.909
##	3rd Qu.: 3.091	3rd Qu.: 3.091	3rd Qu.: 3.091	3rd Qu.: 3.068
##	Max. :31.394	Max. :31.394	Max. :31.394	Max. :31.394
##	V91	V92	V93	V94
##	Min. : 1.451	Min. : 1.453	Min. : 1.453	Min. : 1.453
##	1st Qu.: 2.139	1st Qu.: 2.167	1st Qu.: 2.167	1st Qu.: 2.167
##	Median : 2.399	Median : 2.426	Median : 2.426	Median : 2.426
##	Mean : 2.574	Mean : 2.600	Mean : 2.600	Mean : 2.600
##	3rd Qu.: 2.728	3rd Qu.: 2.752	3rd Qu.: 2.752	3rd Qu.: 2.752
##	Max. :30.862	Max. :30.862	Max. :30.862	Max. :30.862
##	V95	V96	V97	V98
##	Min. : 1.451	Min. : 0.1324	Min. : 0.1730	Min. : 0.1730
##	1st Qu.: 2.139	1st Qu.: 0.8280	1st Qu.: 0.8512	1st Qu.: 0.8512
##	Median : 2.399	Median : 1.0691	Median : 1.0912	Median : 1.0912
##	Mean : 2.574	Mean : 1.2589	Mean : 1.2811	Mean : 1.2811
##	3rd Qu.: 2.728	3rd Qu.: 1.4022	3rd Qu.: 1.4241	3rd Qu.: 1.4241
##	Max. :30.862	Max. :29.8339	Max. :29.8339	Max. :29.8339
##	V99	V100	V101	V102
##	Min. : 0.1730	Min. : 0.173	Min. : 0.00000	Min. : 0.0000
##	1st Qu.: 0.8512	1st Qu.: 0.828	1st Qu.: 0.09587	1st Qu.: 0.1038
##	Median : 1.0912	Median : 1.069	Median : 0.23625	Median : 0.2447
##	Mean : 1.2811	Mean : 1.259	Mean : 0.48458	Mean : 0.4935
##	3rd Qu.: 1.4241	3rd Qu.: 1.402	3rd Qu.: 0.58040	3rd Qu.: 0.5900
##	Max. :29.8339	Max. :29.834	Max. :28.75524	Max. :28.7552
##	V103	V104	V105	V106
##	Min. : 0.0000	Min. : 0.0000	Min. : 0.00000	Min. : 1.480
##	1st Qu.: 0.1038	1st Qu.: 0.1038	1st Qu.: 0.09589	1st Qu.: 2.332
##	Median : 0.2447	Median : 0.2447	Median : 0.23625	Median : 2.643
##	Mean : 0.4935	Mean : 0.4935	Mean : 0.48465	Mean : 2.803
##	3rd Qu.: 0.5900	3rd Qu.: 0.5900	3rd Qu.: 0.58035	3rd Qu.: 2.997
##	Max. :28.7552	Max. :28.7552	Max. :28.75524	Max. :31.328
##	V107	V108	V109	V110
##	Min. : 1.496	Min. : 1.496	Min. : 1.496	Min. : 1.480
##	1st Qu.: 2.363	1st Qu.: 2.363	1st Qu.: 2.363	1st Qu.: 2.332
##	Median : 2.673	Median : 2.673	Median : 2.673	Median : 2.643
##	Mean : 2.832	Mean : 2.832	Mean : 2.832	Mean : 2.803
##	3rd Qu.: 3.024	3rd Qu.: 3.024	3rd Qu.: 3.024	3rd Qu.: 2.997
##	Max. :31.328	Max. :31.328	Max. :31.328	Max. :31.328
##	V111	V112	V113	V114
##	Min. : 1.591	Min. : 1.596	Min. : 1.596	Min. : 1.596
##	1st Qu.: 2.228	1st Qu.: 2.255	1st Qu.: 2.255	1st Qu.: 2.255
##	Median : 2.492	Median : 2.519	Median : 2.519	Median : 2.519

##	Mean	: 2.667	Mean	: 2.693	Mean	: 2.693	Mean	: 2.693
##	3rd Qu.:	2.825	3rd Qu.:	2.849	3rd Qu.:	2.849	3rd Qu.:	2.849
##	Max.	:31.080	Max.	:31.080	Max.	:31.080	Max.	:31.080
##	V115		V116		V117		V118	
##	Min.	: 1.591	Min.	: 0.00	Min.	: 0.00	Min.	: 0.00
##	1st Qu.:	2.228	1st Qu.:	13.99	1st Qu.:	14.16	1st Qu.:	14.16
##	Median	: 2.492	Median	:15.26	Median	:15.46	Median	:15.46
##	Mean	: 2.667	Mean	:15.22	Mean	:15.38	Mean	:15.38
##	3rd Qu.:	2.825	3rd Qu.:	16.55	3rd Qu.:	16.71	3rd Qu.:	16.71
##	Max.	:31.080	Max.	:21.18	Max.	:21.18	Max.	:21.18
##	V119		V120		V121		V122	
##	Min.	: 0.00	Min.	: 0.00	Min.	: 1.766	Min.	: 1.784
##	1st Qu.:	14.16	1st Qu.:	13.99	1st Qu.:	2.408	1st Qu.:	2.443
##	Median	:15.46	Median	:15.26	Median	: 2.727	Median	: 2.762
##	Mean	:15.38	Mean	:15.22	Mean	: 2.883	Mean	: 2.916
##	3rd Qu.:	16.71	3rd Qu.:	16.55	3rd Qu.:	3.079	3rd Qu.:	3.109
##	Max.	:21.18	Max.	:21.18	Max.	:31.381	Max.	:31.381
##	V123		V124		V125		V126	
##	Min.	: 1.784	Min.	: 1.784	Min.	: 1.766	Min.	: 0.9736
##	1st Qu.:	2.443	1st Qu.:	2.443	1st Qu.:	2.408	1st Qu.:	1.4141
##	Median	: 2.762	Median	: 2.762	Median	: 2.727	Median	: 1.5857
##	Mean	: 2.916	Mean	: 2.916	Mean	: 2.883	Mean	: 1.7983
##	3rd Qu.:	3.109	3rd Qu.:	3.109	3rd Qu.:	3.079	3rd Qu.:	1.9024
##	Max.	:31.381	Max.	:31.381	Max.	:31.381	Max.	:29.5175
##	V127		V128		V129		V130	
##	Min.	: 0.975	Min.	: 0.975	Min.	: 0.975	Min.	: 0.9736
##	1st Qu.:	1.428	1st Qu.:	1.428	1st Qu.:	1.428	1st Qu.:	1.4141
##	Median	: 1.598	Median	: 1.598	Median	: 1.598	Median	: 1.5857
##	Mean	: 1.812	Mean	: 1.812	Mean	: 1.812	Mean	: 1.7983
##	3rd Qu.:	1.917	3rd Qu.:	1.917	3rd Qu.:	1.917	3rd Qu.:	1.9023
##	Max.	:29.518	Max.	:29.518	Max.	:29.518	Max.	:29.5175
##	V131		V132		V133		V134	
##	Min.	: 1.175	Min.	: 1.190	Min.	: 1.190	Min.	: 1.190
##	1st Qu.:	1.916	1st Qu.:	1.939	1st Qu.:	1.939	1st Qu.:	1.939
##	Median	: 2.148	Median	: 2.168	Median	: 2.168	Median	: 2.168
##	Mean	: 2.342	Mean	: 2.363	Mean	: 2.363	Mean	: 2.363
##	3rd Qu.:	2.479	3rd Qu.:	2.500	3rd Qu.:	2.500	3rd Qu.:	2.500
##	Max.	:31.248	Max.	:31.248	Max.	:31.248	Max.	:31.248
##	V135		V136		V137		V138	
##	Min.	: 1.175	Min.	: 2.083	Min.	: 2.093	Min.	: 2.093
##	1st Qu.:	1.916	1st Qu.:	2.748	1st Qu.:	2.781	1st Qu.:	2.781
##	Median	: 2.148	Median	: 3.054	Median	: 3.087	Median	: 3.087

##	Mean	: 2.342	Mean	: 3.214	Mean	: 3.245	Mean	: 3.245
##	3rd Qu.:	2.479	3rd Qu.:	3.403	3rd Qu.:	3.432	3rd Qu.:	3.432
##	Max.	:31.248	Max.	:31.436	Max.	:31.436	Max.	:31.436
##	V139		V140		V141		V142	
##	Min.	: 2.093	Min.	: 2.083	Min.	: 1.452	Min.	: 1.462
##	1st Qu.:	2.781	1st Qu.:	2.748	1st Qu.:	2.107	1st Qu.:	2.131
##	Median	: 3.087	Median	: 3.054	Median	: 2.340	Median	: 2.362
##	Mean	: 3.245	Mean	: 3.214	Mean	: 2.524	Mean	: 2.547
##	3rd Qu.:	3.432	3rd Qu.:	3.403	3rd Qu.:	2.663	3rd Qu.:	2.685
##	Max.	:31.436	Max.	:31.436	Max.	:30.691	Max.	:30.691
##	V143		V144		V145		V146	
##	Min.	: 1.462	Min.	: 1.462	Min.	: 1.452	Min.	: 0.00
##	1st Qu.:	2.131	1st Qu.:	2.131	1st Qu.:	2.107	1st Qu.:	13.52
##	Median	: 2.362	Median	: 2.362	Median	: 2.340	Median	:14.86
##	Mean	: 2.547	Mean	: 2.547	Mean	: 2.524	Mean	:14.82
##	3rd Qu.:	2.685	3rd Qu.:	2.685	3rd Qu.:	2.663	3rd Qu.:	16.22
##	Max.	:30.691	Max.	:30.691	Max.	:30.691	Max.	:20.73
##	V147		V148		V149		V150	
##	Min.	: 0.00	Min.	: 0.00	Min.	: 0.00	Min.	: 0.00
##	1st Qu.:	13.70	1st Qu.:	13.70	1st Qu.:	13.70	1st Qu.:	13.52
##	Median	:15.06	Median	:15.06	Median	:15.06	Median	:14.86
##	Mean	:14.99	Mean	:14.99	Mean	:14.99	Mean	:14.82
##	3rd Qu.:	16.39	3rd Qu.:	16.39	3rd Qu.:	16.39	3rd Qu.:	16.22
##	Max.	:20.73	Max.	:20.73	Max.	:20.73	Max.	:20.73
##	V151		V152		V153		V154	
##	Min.	: 1.517	Min.	: 1.524	Min.	: 1.524	Min.	: 1.524
##	1st Qu.:	2.131	1st Qu.:	2.160	1st Qu.:	2.160	1st Qu.:	2.160
##	Median	: 2.404	Median	: 2.432	Median	: 2.432	Median	: 2.432
##	Mean	: 2.576	Mean	: 2.604	Mean	: 2.604	Mean	: 2.604
##	3rd Qu.:	2.740	3rd Qu.:	2.766	3rd Qu.:	2.766	3rd Qu.:	2.766
##	Max.	:31.077	Max.	:31.077	Max.	:31.077	Max.	:31.077
##	V155		V156		V157		V158	
##	Min.	: 1.517	Min.	: 1.306	Min.	: 1.314	Min.	: 1.314
##	1st Qu.:	2.131	1st Qu.:	2.237	1st Qu.:	2.259	1st Qu.:	2.259
##	Median	: 2.404	Median	: 2.441	Median	: 2.461	Median	: 2.461
##	Mean	: 2.576	Mean	: 2.641	Mean	: 2.662	Mean	: 2.662
##	3rd Qu.:	2.740	3rd Qu.:	2.764	3rd Qu.:	2.785	3rd Qu.:	2.785
##	Max.	:31.077	Max.	:30.892	Max.	:30.892	Max.	:30.892
##	V159		V160					
##	Min.	: 1.314	Min.	: 1.306				
##	1st Qu.:	2.259	1st Qu.:	2.237				
##	Median	: 2.461	Median	: 2.441				

```
## Mean : 2.662 Mean : 2.641
## 3rd Qu.: 2.785 3rd Qu.: 2.764
## Max. :30.892 Max. :30.892
```

```
summary(flattened_data)## dataha too ranega an scaleled ok
```

```
##      V1      V2      V3      V4
## Min.   : 1.545 Min.   : 1.550 Min.   : 1.550 Min.   : 1.550
## 1st Qu.: 2.265 1st Qu.: 2.290 1st Qu.: 2.290 1st Qu.: 2.290
## Median : 2.483 Median : 2.506 Median : 2.506 Median : 2.506
## Mean   : 2.676 Mean   : 2.700 Mean   : 2.700 Mean   : 2.700
## 3rd Qu.: 2.806 3rd Qu.: 2.830 3rd Qu.: 2.830 3rd Qu.: 2.830
## Max.   :30.945 Max.   :30.945 Max.   :30.945 Max.   :30.945
##      V5      V6      V7      V8
## Min.   : 1.545 Min.   : 1.032 Min.   : 1.070 Min.   : 1.070
## 1st Qu.: 2.265 1st Qu.: 1.608 1st Qu.: 1.635 1st Qu.: 1.635
## Median : 2.483 Median : 1.862 Median : 1.887 Median : 1.887
## Mean   : 2.676 Mean   : 2.039 Mean   : 2.064 Mean   : 2.064
## 3rd Qu.: 2.806 3rd Qu.: 2.188 3rd Qu.: 2.211 3rd Qu.: 2.211
## Max.   :30.945 Max.   :30.287 Max.   :30.287 Max.   :30.287
##      V9      V10     V11     V12
## Min.   : 1.070 Min.   : 1.032 Min.   : 1.572 Min.   : 1.575
## 1st Qu.: 1.635 1st Qu.: 1.608 1st Qu.: 2.089 1st Qu.: 2.110
## Median : 1.887 Median : 1.862 Median : 2.289 Median : 2.309
## Mean   : 2.064 Mean   : 2.039 Mean   : 2.496 Mean   : 2.516
## 3rd Qu.: 2.211 3rd Qu.: 2.188 3rd Qu.: 2.615 3rd Qu.: 2.636
## Max.   :30.287 Max.   :30.287 Max.   :31.147 Max.   :31.147
##      V13     V14     V15     V16
## Min.   : 1.575 Min.   : 1.575 Min.   : 1.572 Min.   : 2.182
## 1st Qu.: 2.110 1st Qu.: 2.110 1st Qu.: 2.089 1st Qu.: 2.904
## Median : 2.309 Median : 2.309 Median : 2.289 Median : 3.219
## Mean   : 2.516 Mean   : 2.516 Mean   : 2.496 Mean   : 3.375
## 3rd Qu.: 2.636 3rd Qu.: 2.636 3rd Qu.: 2.615 3rd Qu.: 3.573
## Max.   :31.147 Max.   :31.147 Max.   :31.147 Max.   :31.698
##      V17     V18     V19     V20
## Min.   : 2.186 Min.   : 2.186 Min.   : 2.186 Min.   : 2.182
## 1st Qu.: 2.942 1st Qu.: 2.942 1st Qu.: 2.942 1st Qu.: 2.904
## Median : 3.257 Median : 3.257 Median : 3.257 Median : 3.219
## Mean   : 3.410 Mean   : 3.410 Mean   : 3.410 Mean   : 3.375
## 3rd Qu.: 3.605 3rd Qu.: 3.605 3rd Qu.: 3.605 3rd Qu.: 3.573
## Max.   :31.698 Max.   :31.698 Max.   :31.698 Max.   :31.698
```

##	V21	V22	V23	V24	
##	Min. : 1.310	Min. : 1.314	Min. : 1.314	Min. : 1.314	
##	1st Qu.: 2.044	1st Qu.: 2.074	1st Qu.: 2.074	1st Qu.: 2.074	
##	Median : 2.351	Median : 2.381	Median : 2.381	Median : 2.381	
##	Mean : 2.514	Mean : 2.543	Mean : 2.543	Mean : 2.543	
##	3rd Qu.: 2.707	3rd Qu.: 2.733	3rd Qu.: 2.733	3rd Qu.: 2.733	
##	Max. :30.948	Max. :30.948	Max. :30.948	Max. :30.948	
##	V25	V26	V27	V28	V29
##	Min. : 1.310	Min. :0	Min. :0	Min. :0	Min. :0
##	1st Qu.: 2.044	1st Qu.:0	1st Qu.:0	1st Qu.:0	1st Qu.:0
##	Median : 2.351	Median :0	Median :0	Median :0	Median :0
##	Mean : 2.514	Mean :0	Mean :0	Mean :0	Mean :0
##	3rd Qu.: 2.707	3rd Qu.:0	3rd Qu.:0	3rd Qu.:0	3rd Qu.:0
##	Max. :30.948	Max. :0	Max. :0	Max. :0	Max. :0
##	V31	V32	V33	V34	
##	Min. : 1.250	Min. : 1.274	Min. : 1.274	Min. : 1.274	
##	1st Qu.: 1.847	1st Qu.: 1.874	1st Qu.: 1.874	1st Qu.: 1.874	
##	Median : 2.109	Median : 2.135	Median : 2.135	Median : 2.135	
##	Mean : 2.286	Mean : 2.312	Mean : 2.312	Mean : 2.312	
##	3rd Qu.: 2.442	3rd Qu.: 2.467	3rd Qu.: 2.467	3rd Qu.: 2.467	
##	Max. :30.819	Max. :30.819	Max. :30.819	Max. :30.819	
##	V35	V36	V37	V38	
##	Min. : 1.250	Min. : 1.258	Min. : 1.273	Min. : 1.273	
##	1st Qu.: 1.847	1st Qu.: 2.188	1st Qu.: 2.221	1st Qu.: 2.221	
##	Median : 2.109	Median : 2.509	Median : 2.541	Median : 2.541	
##	Mean : 2.286	Mean : 2.665	Mean : 2.695	Mean : 2.695	
##	3rd Qu.: 2.442	3rd Qu.: 2.871	3rd Qu.: 2.899	3rd Qu.: 2.899	
##	Max. :30.819	Max. :31.167	Max. :31.167	Max. :31.167	
##	V39	V40	V41	V42	
##	Min. : 1.273	Min. : 1.258	Min. : 0.0000	Min. : 0.0000	
##	1st Qu.: 2.221	1st Qu.: 2.188	1st Qu.: 0.1014	1st Qu.: 0.1105	
##	Median : 2.541	Median : 2.509	Median : 0.2445	Median : 0.2543	
##	Mean : 2.695	Mean : 2.665	Mean : 0.4946	Mean : 0.5045	
##	3rd Qu.: 2.899	3rd Qu.: 2.871	3rd Qu.: 0.5923	3rd Qu.: 0.6034	
##	Max. :31.167	Max. :31.167	Max. :29.0143	Max. :29.0143	
##	V43	V44	V45	V46	
##	Min. : 0.0000	Min. : 0.0000	Min. : 0.0000	Min. : 1.517	
##	1st Qu.: 0.1105	1st Qu.: 0.1105	1st Qu.: 0.1010	1st Qu.: 2.287	
##	Median : 0.2543	Median : 0.2543	Median : 0.2444	Median : 2.603	
##	Mean : 0.5045	Mean : 0.5045	Mean : 0.4943	Mean : 2.764	
##	3rd Qu.: 0.6034	3rd Qu.: 0.6034	3rd Qu.: 0.5922	3rd Qu.: 2.961	
##	Max. :29.0143	Max. :29.0143	Max. :29.0143	Max. :31.517	

##	V47	V48	V49	V50
##	Min. : 1.530	Min. : 1.530	Min. : 1.530	Min. : 1.517
##	1st Qu.: 2.324	1st Qu.: 2.324	1st Qu.: 2.324	1st Qu.: 2.287
##	Median : 2.639	Median : 2.639	Median : 2.639	Median : 2.603
##	Mean : 2.798	Mean : 2.798	Mean : 2.798	Mean : 2.764
##	3rd Qu.: 2.992	3rd Qu.: 2.992	3rd Qu.: 2.992	3rd Qu.: 2.961
##	Max. :31.517	Max. :31.517	Max. :31.517	Max. :31.517
##	V51	V52	V53	V54
##	Min. : 0.6825	Min. : 0.6872	Min. : 0.6872	Min. : 0.6872
##	1st Qu.: 1.5748	1st Qu.: 1.5986	1st Qu.: 1.5986	1st Qu.: 1.5986
##	Median : 1.7743	Median : 1.7946	Median : 1.7946	Median : 1.7946
##	Mean : 1.9826	Mean : 2.0048	Mean : 2.0048	Mean : 2.0048
##	3rd Qu.: 2.1049	3rd Qu.: 2.1269	3rd Qu.: 2.1269	3rd Qu.: 2.1269
##	Max. :30.5721	Max. :30.5721	Max. :30.5721	Max. :30.5721
##	V55	V56	V57	V58
##	Min. : 0.6825	Min. : 1.614	Min. : 1.616	Min. : 1.616
##	1st Qu.: 1.5748	1st Qu.: 2.324	1st Qu.: 2.353	1st Qu.: 2.353
##	Median : 1.7743	Median : 2.613	Median : 2.641	Median : 2.641
##	Mean : 1.9826	Mean : 2.776	Mean : 2.803	Mean : 2.803
##	3rd Qu.: 2.1049	3rd Qu.: 2.951	3rd Qu.: 2.976	3rd Qu.: 2.976
##	Max. :30.5721	Max. :30.770	Max. :30.770	Max. :30.770
##	V59	V60	V61	V62
##	Min. : 1.616	Min. : 1.614	Min. : 1.294	Min. : 1.302
##	1st Qu.: 2.353	1st Qu.: 2.324	1st Qu.: 2.078	1st Qu.: 2.105
##	Median : 2.641	Median : 2.613	Median : 2.293	Median : 2.317
##	Mean : 2.803	Mean : 2.776	Mean : 2.490	Mean : 2.515
##	3rd Qu.: 2.976	3rd Qu.: 2.951	3rd Qu.: 2.619	3rd Qu.: 2.644
##	Max. :30.770	Max. :30.770	Max. :30.842	Max. :30.842
##	V63	V64	V65	V66
##	Min. : 1.302	Min. : 1.302	Min. : 1.294	Min. : 1.197
##	1st Qu.: 2.105	1st Qu.: 2.105	1st Qu.: 2.078	1st Qu.: 1.858
##	Median : 2.317	Median : 2.317	Median : 2.293	Median : 2.159
##	Mean : 2.515	Mean : 2.515	Mean : 2.490	Mean : 2.322
##	3rd Qu.: 2.644	3rd Qu.: 2.644	3rd Qu.: 2.619	3rd Qu.: 2.508
##	Max. :30.842	Max. :30.842	Max. :30.842	Max. :30.849
##	V67	V68	V69	V70
##	Min. : 1.232	Min. : 1.232	Min. : 1.232	Min. : 1.197
##	1st Qu.: 1.888	1st Qu.: 1.888	1st Qu.: 1.888	1st Qu.: 1.858
##	Median : 2.189	Median : 2.189	Median : 2.189	Median : 2.159
##	Mean : 2.350	Mean : 2.350	Mean : 2.350	Mean : 2.322
##	3rd Qu.: 2.533	3rd Qu.: 2.533	3rd Qu.: 2.533	3rd Qu.: 2.508
##	Max. :30.849	Max. :30.849	Max. :30.849	Max. :30.849

##	V71	V72	V73	V74
##	Min. : 1.427	Min. : 1.432	Min. : 1.432	Min. : 1.432
##	1st Qu.: 2.121	1st Qu.: 2.153	1st Qu.: 2.153	1st Qu.: 2.153
##	Median : 2.430	Median : 2.461	Median : 2.461	Median : 2.461
##	Mean : 2.592	Mean : 2.622	Mean : 2.622	Mean : 2.622
##	3rd Qu.: 2.787	3rd Qu.: 2.814	3rd Qu.: 2.814	3rd Qu.: 2.814
##	Max. :31.179	Max. :31.179	Max. :31.179	Max. :31.179
##	V75	V76	V77	V78
##	Min. : 1.427	Min. : 2.012	Min. : 2.018	Min. : 2.018
##	1st Qu.: 2.121	1st Qu.: 2.647	1st Qu.: 2.674	1st Qu.: 2.674
##	Median : 2.430	Median : 2.883	Median : 2.909	Median : 2.909
##	Mean : 2.592	Mean : 3.064	Mean : 3.090	Mean : 3.090
##	3rd Qu.: 2.787	3rd Qu.: 3.202	3rd Qu.: 3.227	3rd Qu.: 3.227
##	Max. :31.179	Max. :31.124	Max. :31.124	Max. :31.124
##	V79	V80	V81	V82
##	Min. : 2.018	Min. : 2.012	Min. : 1.714	Min. : 1.722
##	1st Qu.: 2.674	1st Qu.: 2.647	1st Qu.: 2.341	1st Qu.: 2.368
##	Median : 2.909	Median : 2.883	Median : 2.579	Median : 2.604
##	Mean : 3.090	Mean : 3.064	Mean : 2.763	Mean : 2.788
##	3rd Qu.: 3.227	3rd Qu.: 3.202	3rd Qu.: 2.902	3rd Qu.: 2.926
##	Max. :31.124	Max. :31.124	Max. :30.970	Max. :30.970
##	V83	V84	V85	V86
##	Min. : 1.722	Min. : 1.722	Min. : 1.714	Min. : 1.712
##	1st Qu.: 2.368	1st Qu.: 2.368	1st Qu.: 2.341	1st Qu.: 2.472
##	Median : 2.604	Median : 2.604	Median : 2.579	Median : 2.731
##	Mean : 2.788	Mean : 2.788	Mean : 2.763	Mean : 2.909
##	3rd Qu.: 2.926	3rd Qu.: 2.926	3rd Qu.: 2.902	3rd Qu.: 3.068
##	Max. :30.970	Max. :30.970	Max. :30.970	Max. :31.394
##	V87	V88	V89	V90
##	Min. : 1.760	Min. : 1.760	Min. : 1.760	Min. : 1.712
##	1st Qu.: 2.497	1st Qu.: 2.497	1st Qu.: 2.497	1st Qu.: 2.472
##	Median : 2.756	Median : 2.756	Median : 2.756	Median : 2.731
##	Mean : 2.933	Mean : 2.933	Mean : 2.933	Mean : 2.909
##	3rd Qu.: 3.091	3rd Qu.: 3.091	3rd Qu.: 3.091	3rd Qu.: 3.068
##	Max. :31.394	Max. :31.394	Max. :31.394	Max. :31.394
##	V91	V92	V93	V94
##	Min. : 1.451	Min. : 1.453	Min. : 1.453	Min. : 1.453
##	1st Qu.: 2.139	1st Qu.: 2.167	1st Qu.: 2.167	1st Qu.: 2.167
##	Median : 2.399	Median : 2.426	Median : 2.426	Median : 2.426
##	Mean : 2.574	Mean : 2.600	Mean : 2.600	Mean : 2.600
##	3rd Qu.: 2.728	3rd Qu.: 2.752	3rd Qu.: 2.752	3rd Qu.: 2.752
##	Max. :30.862	Max. :30.862	Max. :30.862	Max. :30.862

##	V95	V96	V97	V98
##	Min. : 1.451	Min. : 0.1324	Min. : 0.1730	Min. : 0.1730
##	1st Qu.: 2.139	1st Qu.: 0.8280	1st Qu.: 0.8512	1st Qu.: 0.8512
##	Median : 2.399	Median : 1.0691	Median : 1.0912	Median : 1.0912
##	Mean : 2.574	Mean : 1.2589	Mean : 1.2811	Mean : 1.2811
##	3rd Qu.: 2.728	3rd Qu.: 1.4022	3rd Qu.: 1.4241	3rd Qu.: 1.4241
##	Max. :30.862	Max. :29.8339	Max. :29.8339	Max. :29.8339
##	V99	V100	V101	V102
##	Min. : 0.1730	Min. : 0.173	Min. : 0.00000	Min. : 0.0000
##	1st Qu.: 0.8512	1st Qu.: 0.828	1st Qu.: 0.09587	1st Qu.: 0.1038
##	Median : 1.0912	Median : 1.069	Median : 0.23625	Median : 0.2447
##	Mean : 1.2811	Mean : 1.259	Mean : 0.48458	Mean : 0.4935
##	3rd Qu.: 1.4241	3rd Qu.: 1.402	3rd Qu.: 0.58040	3rd Qu.: 0.5900
##	Max. :29.8339	Max. :29.834	Max. :28.75524	Max. :28.7552
##	V103	V104	V105	V106
##	Min. : 0.0000	Min. : 0.0000	Min. : 0.00000	Min. : 1.480
##	1st Qu.: 0.1038	1st Qu.: 0.1038	1st Qu.: 0.09589	1st Qu.: 2.332
##	Median : 0.2447	Median : 0.2447	Median : 0.23625	Median : 2.643
##	Mean : 0.4935	Mean : 0.4935	Mean : 0.48465	Mean : 2.803
##	3rd Qu.: 0.5900	3rd Qu.: 0.5900	3rd Qu.: 0.58035	3rd Qu.: 2.997
##	Max. :28.7552	Max. :28.7552	Max. :28.75524	Max. :31.328
##	V107	V108	V109	V110
##	Min. : 1.496	Min. : 1.496	Min. : 1.496	Min. : 1.480
##	1st Qu.: 2.363	1st Qu.: 2.363	1st Qu.: 2.363	1st Qu.: 2.332
##	Median : 2.673	Median : 2.673	Median : 2.673	Median : 2.643
##	Mean : 2.832	Mean : 2.832	Mean : 2.832	Mean : 2.803
##	3rd Qu.: 3.024	3rd Qu.: 3.024	3rd Qu.: 3.024	3rd Qu.: 2.997
##	Max. :31.328	Max. :31.328	Max. :31.328	Max. :31.328
##	V111	V112	V113	V114
##	Min. : 1.591	Min. : 1.596	Min. : 1.596	Min. : 1.596
##	1st Qu.: 2.228	1st Qu.: 2.255	1st Qu.: 2.255	1st Qu.: 2.255
##	Median : 2.492	Median : 2.519	Median : 2.519	Median : 2.519
##	Mean : 2.667	Mean : 2.693	Mean : 2.693	Mean : 2.693
##	3rd Qu.: 2.825	3rd Qu.: 2.849	3rd Qu.: 2.849	3rd Qu.: 2.849
##	Max. :31.080	Max. :31.080	Max. :31.080	Max. :31.080
##	V115	V116	V117	V118
##	Min. : 1.591	Min. : 0.00	Min. : 0.00	Min. : 0.00
##	1st Qu.: 2.228	1st Qu.:13.99	1st Qu.:14.16	1st Qu.:14.16
##	Median : 2.492	Median :15.26	Median :15.46	Median :15.46
##	Mean : 2.667	Mean :15.22	Mean :15.38	Mean :15.38
##	3rd Qu.: 2.825	3rd Qu.:16.55	3rd Qu.:16.71	3rd Qu.:16.71
##	Max. :31.080	Max. :21.18	Max. :21.18	Max. :21.18

##	V119	V120	V121	V122
##	Min. : 0.00	Min. : 0.00	Min. : 1.766	Min. : 1.784
##	1st Qu.:14.16	1st Qu.:13.99	1st Qu.: 2.408	1st Qu.: 2.443
##	Median :15.46	Median :15.26	Median : 2.727	Median : 2.762
##	Mean :15.38	Mean :15.22	Mean : 2.883	Mean : 2.916
##	3rd Qu.:16.71	3rd Qu.:16.55	3rd Qu.: 3.079	3rd Qu.: 3.109
##	Max. :21.18	Max. :21.18	Max. :31.381	Max. :31.381
##	V123	V124	V125	V126
##	Min. : 1.784	Min. : 1.784	Min. : 1.766	Min. : 0.9736
##	1st Qu.: 2.443	1st Qu.: 2.443	1st Qu.: 2.408	1st Qu.: 1.4141
##	Median : 2.762	Median : 2.762	Median : 2.727	Median : 1.5857
##	Mean : 2.916	Mean : 2.916	Mean : 2.883	Mean : 1.7983
##	3rd Qu.: 3.109	3rd Qu.: 3.109	3rd Qu.: 3.079	3rd Qu.: 1.9024
##	Max. :31.381	Max. :31.381	Max. :31.381	Max. :29.5175
##	V127	V128	V129	V130
##	Min. : 0.975	Min. : 0.975	Min. : 0.975	Min. : 0.9736
##	1st Qu.: 1.428	1st Qu.: 1.428	1st Qu.: 1.428	1st Qu.: 1.4141
##	Median : 1.598	Median : 1.598	Median : 1.598	Median : 1.5857
##	Mean : 1.812	Mean : 1.812	Mean : 1.812	Mean : 1.7983
##	3rd Qu.: 1.917	3rd Qu.: 1.917	3rd Qu.: 1.917	3rd Qu.: 1.9023
##	Max. :29.518	Max. :29.518	Max. :29.518	Max. :29.5175
##	V131	V132	V133	V134
##	Min. : 1.175	Min. : 1.190	Min. : 1.190	Min. : 1.190
##	1st Qu.: 1.916	1st Qu.: 1.939	1st Qu.: 1.939	1st Qu.: 1.939
##	Median : 2.148	Median : 2.168	Median : 2.168	Median : 2.168
##	Mean : 2.342	Mean : 2.363	Mean : 2.363	Mean : 2.363
##	3rd Qu.: 2.479	3rd Qu.: 2.500	3rd Qu.: 2.500	3rd Qu.: 2.500
##	Max. :31.248	Max. :31.248	Max. :31.248	Max. :31.248
##	V135	V136	V137	V138
##	Min. : 1.175	Min. : 2.083	Min. : 2.093	Min. : 2.093
##	1st Qu.: 1.916	1st Qu.: 2.748	1st Qu.: 2.781	1st Qu.: 2.781
##	Median : 2.148	Median : 3.054	Median : 3.087	Median : 3.087
##	Mean : 2.342	Mean : 3.214	Mean : 3.245	Mean : 3.245
##	3rd Qu.: 2.479	3rd Qu.: 3.403	3rd Qu.: 3.432	3rd Qu.: 3.432
##	Max. :31.248	Max. :31.436	Max. :31.436	Max. :31.436
##	V139	V140	V141	V142
##	Min. : 2.093	Min. : 2.083	Min. : 1.452	Min. : 1.462
##	1st Qu.: 2.781	1st Qu.: 2.748	1st Qu.: 2.107	1st Qu.: 2.131
##	Median : 3.087	Median : 3.054	Median : 2.340	Median : 2.362
##	Mean : 3.245	Mean : 3.214	Mean : 2.524	Mean : 2.547
##	3rd Qu.: 3.432	3rd Qu.: 3.403	3rd Qu.: 2.663	3rd Qu.: 2.685
##	Max. :31.436	Max. :31.436	Max. :30.691	Max. :30.691

	V143	V144	V145	V146
## Min.	: 1.462	Min. : 1.462	Min. : 1.452	Min. : 0.00
## 1st Qu.:	2.131	1st Qu.: 2.131	1st Qu.: 2.107	1st Qu.:13.52
## Median :	2.362	Median : 2.362	Median : 2.340	Median :14.86
## Mean :	2.547	Mean : 2.547	Mean : 2.524	Mean :14.82
## 3rd Qu.:	2.685	3rd Qu.: 2.685	3rd Qu.: 2.663	3rd Qu.:16.22
## Max. :	30.691	Max. :30.691	Max. :30.691	Max. :20.73
##	V147	V148	V149	V150
## Min.	: 0.00	Min. : 0.00	Min. : 0.00	Min. : 0.00
## 1st Qu.:	13.70	1st Qu.:13.70	1st Qu.:13.70	1st Qu.:13.52
## Median :	15.06	Median :15.06	Median :15.06	Median :14.86
## Mean :	14.99	Mean :14.99	Mean :14.99	Mean :14.82
## 3rd Qu.:	16.39	3rd Qu.:16.39	3rd Qu.:16.39	3rd Qu.:16.22
## Max. :	20.73	Max. :20.73	Max. :20.73	Max. :20.73
##	V151	V152	V153	V154
## Min.	: 1.517	Min. : 1.524	Min. : 1.524	Min. : 1.524
## 1st Qu.:	2.131	1st Qu.: 2.160	1st Qu.: 2.160	1st Qu.: 2.160
## Median :	2.404	Median : 2.432	Median : 2.432	Median : 2.432
## Mean :	2.576	Mean : 2.604	Mean : 2.604	Mean : 2.604
## 3rd Qu.:	2.740	3rd Qu.: 2.766	3rd Qu.: 2.766	3rd Qu.: 2.766
## Max. :	31.077	Max. :31.077	Max. :31.077	Max. :31.077
##	V155	V156	V157	V158
## Min.	: 1.517	Min. : 1.306	Min. : 1.314	Min. : 1.314
## 1st Qu.:	2.131	1st Qu.: 2.237	1st Qu.: 2.259	1st Qu.: 2.259
## Median :	2.404	Median : 2.441	Median : 2.461	Median : 2.461
## Mean :	2.576	Mean : 2.641	Mean : 2.662	Mean : 2.662
## 3rd Qu.:	2.740	3rd Qu.: 2.764	3rd Qu.: 2.785	3rd Qu.: 2.785
## Max. :	31.077	Max. :30.892	Max. :30.892	Max. :30.892
##	V159	V160		
## Min.	: 1.314	Min. : 1.306		
## 1st Qu.:	2.259	1st Qu.: 2.237		
## Median :	2.461	Median : 2.441		
## Mean :	2.662	Mean : 2.641		
## 3rd Qu.:	2.785	3rd Qu.: 2.764		
## Max. :	30.892	Max. :30.892		

```
# Identify columns where all rows are zero
columns_all_zeros <- apply(flattened_data, 2, function(x) all(x == 0))

# Get the names of these columns (if the matrix has column names)
columns_with_all_zeros <- names(columns_all_zeros)[columns_all_zeros]
```

```
# Print the names of columns with all zeros  
print(columns_with_all_zeros)
```

```
## NULL
```

```
# Remove these columns from extracted_features  
flattened_data <- flattened_data[, !columns_all_zeros]  
  
# Display the dimensions of the cleaned extracted_features  
print(dim(flattened_data))
```

```
## [1] 1336810    155
```

```
# Step 1: Identify columns that contain at least one zero  
#columns_with_zeros <- apply(flattened_data, 2, function(x) any(x == 0))
```

```
# Step 2: Filter out these columns from the matrix  
#flattened_data <- flattened_data[, !columns_with_zeros]
```

```
# Show dimensions or summary of the filtered data  
#print(dim(filtered_data))  
#summary(filtered_data)
```

```
library(stats)
```

```
#  
pca <- prcomp(flattened_data, scale. = TRUE, center = TRUE)
```

```
# Extracting the importance of components  
explained_variance <- summary(pca)$importance[2,]  
cumulative_variance <- cumsum(explained_variance)  
num_components <- which(cumulative_variance >= 0.999)[1]
```

```
# Using the number of components to reduce dimensions  
reduced_data <- pca$x[, 1:num_components]
```

```
summary(reduced_data)
```

```
##          PC1          PC2          PC3          PC4
## Min.    :-321.156 Min.    :-10.43790 Min.    :-22.3097 Min.    :-22.60498
## 1st Qu.: -1.464 1st Qu.: -2.20609 1st Qu.: -1.0018 1st Qu.: -0.42571
## Median :  2.298 Median : -0.04554 Median :  0.0193 Median : -0.01867
## Mean   :  0.000 Mean   :  0.00000 Mean   :  0.0000 Mean   :  0.00000
## 3rd Qu.:  4.691 3rd Qu.:  2.13991 3rd Qu.:  1.0148 3rd Qu.:  0.40148
## Max.    : 11.300 Max.    : 14.30648 Max.    : 87.2442 Max.    : 45.61329
##          PC5          PC6          PC7
## Min.    :-22.32528 Min.    :-41.33976 Min.    :-9.51994
## 1st Qu.: -0.09744 1st Qu.: -0.19908 1st Qu.: -0.11711
## Median : -0.00279 Median :  0.00184 Median :  0.02676
## Mean   :  0.00000 Mean   :  0.00000 Mean   :  0.00000
## 3rd Qu.:  0.09110 3rd Qu.:  0.20026 3rd Qu.:  0.14409
## Max.    : 97.42868 Max.    : 18.73958 Max.    :11.17001
##          PC8
## Min.    :-14.486692
## 1st Qu.: -0.128074
## Median : -0.008422
## Mean   :  0.000000
## 3rd Qu.:  0.119418
## Max.    : 15.148364
```

```
# Combine df1 and df2 side by side
combined_df <- cbind(cnn_output, reduced_data)
data <- rename(combined_df, c("close"= "cnn_output" ))
combined_df <- rename(combined_df, c("close"= "cnn_output" ))

# Display the combined data frame
print(head(combined_df,100))
```

```
##      close      Date      PC1      PC2      PC3      PC4
## 1    83.0 2019-10-01 -2.26918747 -2.24243821  0.43747432 -1.104238831
## 2    81.0 2019-10-02 -2.16304539 -2.15324518 -0.08013630 -1.050565237
## 3    80.8 2019-10-03 -1.95001609 -1.92132539 -0.29606686 -0.463170987
## 4    80.8 2019-10-04 -1.86825572 -1.83262084 -0.33459373  0.282201038
## 5    81.8 2019-10-07 -1.58824028 -1.56645507 -0.12940610  0.772491511
## 6    82.4 2019-10-08 -1.78946263 -1.70874145  0.42550469  0.547143425
## 7    83.4 2019-10-09 -1.82946503 -1.66088641  0.53428766 -0.090016863
## 8    81.6 2019-10-10 -2.11481635 -2.04738572  0.33184996 -0.939907068
## 9    82.0 2019-10-11 -2.07436920 -2.04127037 -0.13550107 -1.124886587
```

##	10	81.0	2019-10-14	-1.97948701	-2.02827851	-0.50641104	-0.741481434
##	11	81.6	2019-10-15	-1.80047321	-1.81618493	-0.46024136	0.185197214
##	12	87.6	2019-10-16	-1.76901564	-1.77161515	-0.41581906	0.609646282
##	13	86.0	2019-10-17	-1.87868331	-1.76201342	0.07385843	0.515288868
##	14	86.6	2019-10-18	-2.10050495	-1.92921059	0.21190047	0.150690565
##	15	85.8	2019-10-21	-2.20685552	-2.07602461	0.29518447	-0.776919697
##	16	86.8	2019-10-22	-2.27141868	-2.09924641	0.28924172	-0.699691975
##	17	87.0	2019-10-23	-2.18334795	-2.01829409	-0.10177895	-0.746829583
##	18	88.6	2019-10-24	-2.15653411	-1.89323013	-0.53335582	-0.181463880
##	19	88.8	2019-10-25	-2.30962010	-2.02295434	-0.79206654	-0.237598123
##	20	89.8	2019-10-28	-2.66426537	-2.33916804	-0.84058779	-0.317825640
##	21	91.4	2019-10-29	-2.92668286	-2.68046668	-1.07369855	-0.650816495
##	22	91.0	2019-10-30	-3.29315936	-3.17567513	-1.16222373	-0.606267353
##	23	91.0	2019-10-31	-3.37337779	-3.29751806	-1.37422738	-0.380515571
##	24	92.0	2019-11-01	-3.63663800	-3.63010951	-0.98148336	0.042730214
##	25	91.0	2019-11-04	-3.73465371	-3.75081117	-0.83116012	0.183157775
##	26	91.2	2019-11-05	-3.91955986	-3.93417642	-0.23427501	0.135159727
##	27	89.0	2019-11-06	-3.98161425	-4.00562427	0.01379722	0.059753506
##	28	89.0	2019-11-07	-3.87294512	-3.85707105	0.73846120	-0.061387691
##	29	89.2	2019-11-08	-3.72909426	-3.66463604	1.16968084	0.078936839
##	30	86.0	2019-11-11	-3.33243600	-3.10848591	1.72403860	0.192704833
##	31	87.0	2019-11-12	-2.87638596	-2.52515861	2.17714491	0.587525197
##	32	85.4	2019-11-13	-2.39332372	-1.81263032	2.61381184	0.368351782
##	33	86.0	2019-11-14	-1.77956855	-1.02668148	3.23907562	0.359880257
##	34	87.4	2019-11-15	-1.26497139	-0.26610007	3.26517231	-0.055107899
##	35	85.8	2019-11-18	-0.52280041	0.71823961	3.02501991	-0.336243617
##	36	85.2	2019-11-19	-0.09023679	1.40460439	2.48001842	-0.483996844
##	37	84.4	2019-11-20	0.45635335	2.12760951	1.74166844	-0.799698865
##	38	83.6	2019-11-21	0.87482352	2.62209541	1.20154216	-0.478076466
##	39	87.0	2019-11-22	1.20327072	3.09246417	0.67047704	-0.285909803
##	40	86.4	2019-11-25	1.35318697	3.30749223	-0.01152339	-0.092111429
##	41	86.0	2019-11-26	1.14720923	3.11910735	-0.57666816	-0.194393359
##	42	86.4	2019-11-27	0.94388597	2.78888918	-1.10479067	-0.507064954
##	43	85.2	2019-11-28	0.62189102	2.32931943	-1.43435315	-0.365269347
##	44	85.6	2019-11-29	0.47738895	2.01454628	-1.58243290	-0.138680874
##	45	84.2	2019-12-02	0.23293777	1.63606990	-1.68120761	0.330478804
##	46	82.8	2019-12-03	0.15391679	1.43555132	-1.33933134	0.810697552
##	47	84.2	2019-12-04	0.01876250	1.20325274	-0.56237682	0.971086736
##	48	84.0	2019-12-05	-0.02107456	1.18774465	0.20783168	0.969179411
##	49	85.0	2019-12-06	0.01450782	1.17293836	0.84275167	0.679366112
##	50	84.4	2019-12-09	0.01762329	1.28972747	1.03043637	0.112660807
##	51	83.2	2019-12-10	0.11542934	1.37624439	0.98425327	-0.272169573

##	52	83.4	2019-12-11	0.26615968	1.61627506	0.88782469	-0.622480677
##	53	83.4	2019-12-12	0.51356272	1.88672586	0.50406445	-0.327883133
##	54	82.8	2019-12-13	0.75900077	2.19233444	0.11681619	-0.029833737
##	55	82.6	2019-12-16	0.83400864	2.36889111	0.07329674	0.258093102
##	56	85.8	2019-12-17	0.92906705	2.46946898	0.14609029	0.373480242
##	57	85.2	2019-12-18	0.78245465	2.38274336	0.14741609	-0.041326022
##	58	85.4	2019-12-19	0.53306824	2.10467578	-0.34754527	-0.482614749
##	59	85.0	2019-12-20	0.23287265	1.63441119	-1.15958310	-0.895492059
##	60	84.0	2019-12-23	-0.09779604	1.14122410	-1.85565896	-0.850526558
##	61	87.0	2019-12-27	-0.28346359	0.72917229	-2.06735890	-0.226290421
##	62	87.6	2019-12-30	-0.66891284	0.28269481	-2.25734285	0.209396382
##	63	87.6	2020-01-02	-1.11323531	-0.31347470	-2.31085956	0.555689525
##	64	84.0	2020-01-03	-1.72371125	-1.08398979	-2.33611884	0.114666835
##	65	83.8	2020-01-06	-2.11238379	-1.79270314	-1.75966699	0.272889441
##	66	84.4	2020-01-07	-2.22009538	-1.94850152	-0.89139994	0.408579784
##	67	84.4	2020-01-08	-2.24236975	-2.11336836	0.11329525	1.035544290
##	68	84.4	2020-01-09	-2.04604772	-1.84212607	0.59485296	1.173235725
##	69	84.2	2020-01-10	-1.95655812	-1.66850839	1.58798871	0.829599053
##	70	83.0	2020-01-13	-1.72624064	-1.24175594	2.07499213	0.370984052
##	71	81.0	2020-01-14	-1.31559180	-0.73348698	2.80414401	-0.318379287
##	72	81.0	2020-01-15	-0.66054831	0.04305778	2.71230048	-0.120367278
##	73	81.0	2020-01-16	0.06563980	1.08417256	2.53196712	0.159249101
##	74	81.0	2020-01-17	0.60346976	1.90411462	2.36195092	0.387907350
##	75	78.0	2020-01-20	1.09075665	2.61072184	2.42356341	0.136101809
##	76	80.0	2020-01-21	1.56672970	3.24070682	2.38529710	-0.008106613
##	77	78.4	2020-01-22	1.92520227	3.80375131	1.84951267	-0.391420629
##	78	76.0	2020-01-23	2.31934503	4.26581536	1.21299249	-0.372528585
##	79	78.0	2020-01-24	2.56938630	4.54775068	0.47073261	-0.436803313
##	80	74.4	2020-01-27	2.62747031	4.62253889	-0.21173473	-0.305143424
##	81	76.6	2020-01-28	2.70834951	4.60612541	-0.59001329	-0.085686334
##	82	79.0	2020-01-29	2.62762207	4.42004029	-0.90444363	0.104059824
##	83	78.2	2020-01-30	2.47608353	4.20224665	-1.24639174	0.319798799
##	84	78.0	2020-01-31	2.10343449	3.69577548	-1.38231535	0.015302841
##	85	74.0	2020-02-03	1.80554927	3.16784492	-1.56436564	0.086048304
##	86	77.8	2020-02-04	1.74657402	2.95518311	-1.18594897	0.347736430
##	87	77.6	2020-02-05	1.26429845	2.31615660	-1.40774483	0.172177092
##	88	75.2	2020-02-06	0.88152992	1.72414631	-1.85344725	0.001693554
##	89	73.0	2020-02-07	0.32767077	0.88627101	-2.19550138	-0.363349166
##	90	71.4	2020-02-10	0.06980170	0.39623438	-2.00618404	0.084444200
##	91	72.8	2020-02-11	-0.12779380	0.06737101	-1.42574091	0.511518919
##	92	73.6	2020-02-12	-0.16407265	-0.18082997	-0.69886679	1.105975068
##	93	73.0	2020-02-13	-0.09677602	-0.12498061	-0.43274396	1.395513722

##	94	74.0	2020-02-14	-0.28590765	-0.31754288	0.50731135	0.749812637
##	95	73.2	2020-02-17	-0.44930305	-0.47699315	0.69068682	-0.161874456
##	96	75.0	2020-02-18	-0.46837405	-0.57997864	0.78599037	-0.817567536
##	97	75.0	2020-02-19	-0.33857288	-0.31401934	0.18442645	-0.847124431
##	98	75.0	2020-02-20	-0.32389911	-0.30869688	-0.41466695	-0.465512601
##	99	75.0	2020-02-21	-0.30662499	-0.26444237	-0.92489160	-0.307376431
##	100	68.8	2020-02-24	-0.56643942	-0.63766213	-1.03317867	-0.126647457
##			PC5	PC6	PC7	PC8	
##	1	8.517397e-03	-0.284581165	0.007407972	3.353055e-02		
##	2	1.677207e-01	0.265566009	-0.128344815	-1.647790e-01		
##	3	2.053564e-01	0.440556055	-0.030381288	-2.725254e-02		
##	4	1.646317e-01	0.491063123	0.148170749	2.400937e-01		
##	5	-1.502566e-02	0.108259543	0.023184723	8.987586e-02		
##	6	-8.638432e-02	-0.365059557	-0.058743162	9.835542e-03		
##	7	-1.253019e-01	-0.611404957	-0.004865333	1.526114e-01		
##	8	9.887654e-03	-0.375972021	-0.072734596	1.467222e-01		
##	9	2.079814e-01	0.119801335	-0.306438724	-1.424219e-01		
##	10	1.506840e-01	0.287459420	-0.147113714	9.410234e-02		
##	11	1.832769e-01	0.628713174	-0.008370578	3.412481e-01		
##	12	7.081935e-02	0.296412269	-0.159156002	1.318477e-01		
##	13	-1.917867e-01	-0.475953430	-0.125695162	2.217364e-01		
##	14	-2.319938e-01	-0.491413720	-0.106255208	2.595702e-01		
##	15	-5.431455e-02	-0.309531677	-0.261308339	5.272896e-02		
##	16	-6.582975e-03	0.068847437	-0.357656349	-1.348573e-01		
##	17	1.890932e-02	0.382858672	-0.040207129	2.779280e-01		
##	18	-4.990710e-02	0.134569491	0.075953174	3.273594e-01		
##	19	-1.003020e-01	-0.200273902	0.023851145	2.477119e-01		
##	20	2.389758e-02	-0.272231447	-0.093346835	-3.465474e-02		
##	21	4.669441e-02	-0.255405004	-0.030026279	4.609111e-02		
##	22	1.328698e-01	0.107678006	0.037540276	7.095434e-02		
##	23	1.037875e-01	0.065225953	0.053421690	5.140690e-02		
##	24	4.393296e-02	0.106554185	0.105053873	5.505759e-02		
##	25	7.607837e-03	0.028745028	0.094649359	3.158046e-02		
##	26	-6.049942e-02	-0.209970124	0.081181460	1.169791e-02		
##	27	-6.947103e-02	-0.232489647	0.056029536	1.584096e-02		
##	28	5.132763e-02	0.096968482	-0.052002975	-1.284680e-01		
##	29	6.207960e-02	0.344149457	-0.132150644	-1.903670e-01		
##	30	4.076205e-02	0.444937330	0.014417772	4.730249e-02		
##	31	4.168247e-05	0.373427742	0.015094523	8.092872e-02		
##	32	-7.018407e-02	0.212956544	-0.054341012	1.784656e-02		
##	33	-8.348701e-02	0.098026645	-0.148018840	-4.487345e-02		
##	34	-7.207112e-02	0.252805476	-0.103421742	5.406426e-02		

## 35	-1.947466e-01	-0.055405594	-0.052351910	1.683910e-01
## 36	-1.273214e-01	-0.059309854	-0.015347044	2.017028e-01
## 37	-2.419564e-02	0.015511986	-0.120021610	5.928372e-02
## 38	1.591198e-02	0.149799470	-0.183569591	-6.671301e-02
## 39	8.503666e-02	0.407005828	-0.051750250	1.188840e-01
## 40	-9.673922e-02	-0.060286661	0.094376797	2.920221e-01
## 41	-6.955279e-02	-0.158403253	0.023791907	1.908882e-01
## 42	7.376006e-02	-0.243925505	-0.147536105	-8.866026e-02
## 43	1.461536e-01	0.020740501	-0.141505949	-1.101538e-01
## 44	1.721157e-01	0.311075848	0.034796206	1.351050e-01
## 45	7.425095e-02	0.022987242	0.113044271	2.102178e-01
## 46	1.959392e-02	-0.035718290	-0.070866710	-5.004652e-02
## 47	6.575522e-02	-0.014394352	-0.218851591	-2.980175e-01
## 48	-3.996943e-02	-0.019342938	-0.108359134	-1.247968e-01
## 49	-1.161588e-01	-0.055800055	0.066855965	1.477166e-01
## 50	-1.842379e-01	-0.355326056	0.059277804	1.578028e-01
## 51	-1.268252e-01	-0.279151263	-0.134911661	-9.620873e-02
## 52	2.915828e-02	0.049692799	-0.143072609	-9.269454e-02
## 53	-7.703258e-03	0.266844227	-0.028176354	7.802570e-02
## 54	-1.049974e-02	0.291267427	0.048043598	1.796428e-01
## 55	-2.120695e-02	0.022664761	-0.019483121	7.205170e-02
## 56	-4.249978e-02	-0.055642685	-0.069129198	9.525828e-03
## 57	-7.703507e-02	-0.189109524	0.025638981	1.214819e-01
## 58	-1.115470e-01	-0.311219437	0.036787991	1.521099e-01
## 59	3.718694e-02	-0.181553271	-0.058693495	-9.568511e-03
## 60	2.208069e-01	0.070212064	-0.103510369	-1.564288e-01
## 61	3.076344e-01	0.435459080	-0.012370880	-1.873864e-02
## 62	1.201825e-01	0.142210323	0.248053253	3.153517e-01
## 63	-5.407765e-02	-0.299408719	0.141412724	1.796844e-01
## 64	-4.806465e-02	-0.772963606	-0.022268253	-9.454094e-02
## 65	1.968327e-01	-0.167525136	-0.297241606	-4.766055e-01
## 66	2.372226e-01	0.443978164	-0.197245013	-3.320669e-01
## 67	8.386687e-02	0.586545307	0.092426654	7.991099e-02
## 68	-1.237245e-01	0.127533583	0.145294493	2.142717e-01
## 69	-2.494414e-01	-0.534763365	0.018138244	-9.590652e-03
## 70	-1.587178e-01	-0.442243593	-0.153283842	-2.316268e-01
## 71	-1.511841e-02	0.223875818	-0.178572450	-2.907394e-01
## 72	-9.494606e-02	0.512494402	-0.117489375	-1.215420e-01
## 73	-1.090639e-01	0.532415665	0.081902169	1.133144e-01
## 74	-1.358326e-01	0.168084219	0.061410118	7.518041e-02
## 75	-1.482617e-01	0.003071744	0.034472396	-5.428643e-05
## 76	-1.007418e-01	0.026855282	-0.033496326	-8.803027e-02

```
## 77 -1.014240e-01 0.004046392 0.044726220 1.770356e-02
## 78 -6.294254e-02 0.031345538 0.033423309 3.195920e-03
## 79 3.746224e-02 0.018624673 0.072581359 2.492040e-02
## 80 7.907437e-02 0.033583321 0.050377526 -1.926971e-02
## 81 1.573986e-01 0.166293035 0.018331931 -5.577653e-02
## 82 1.325887e-01 0.004140257 0.108116146 6.267537e-02
## 83 1.566766e-02 -0.126622876 0.087564914 6.308358e-02
## 84 1.923811e-02 -0.282987968 0.044745717 8.156926e-03
## 85 7.366669e-02 -0.181118088 -0.096913754 -1.418189e-01
## 86 1.413849e-01 0.221458151 -0.134193749 -2.026747e-01
## 87 4.555657e-02 -0.112695835 0.079798874 2.020417e-01
## 88 2.325527e-02 -0.311094007 0.042739334 2.696694e-01
## 89 1.733329e-01 -0.431375663 -0.218436131 -6.770686e-02
## 90 3.577631e-01 0.081944631 -0.438120584 -2.997086e-01
## 91 3.636392e-01 0.506839208 -0.238546073 6.656466e-02
## 92 1.443857e-01 0.364112131 -0.089753920 4.244045e-01
## 93 -1.570128e-01 -0.399100916 -0.249346976 2.590960e-01
## 94 -2.496885e-01 -0.701296673 -0.448498144 4.375985e-02
## 95 -1.363792e-01 -0.688052785 -0.430170716 1.067822e-01
## 96 9.365824e-02 0.031906446 -0.412011590 1.313277e-01
## 97 5.022116e-02 0.050704413 -0.339130562 2.479776e-01
## 98 4.099108e-02 0.181005502 -0.176246652 3.676044e-01
## 99 6.941029e-02 0.015793282 -0.200227650 3.146651e-01
## 100 5.373250e-02 -0.134789751 -0.224100815 1.505442e-01
```

```
write.csv(combined_df, file = "final.feature.feed_to_other_models19.05.2024better_pca_+++.csv", row.names = FALSE, na = "NA")
```

```
save(reduced_data, file = "datatabnetwithhpcafinal+19.5.2024.RData")
```

```
##### pca histograms
```

```
# Load necessary libraries
```

```
library(ggplot2)
```

```
library(gridExtra)
```

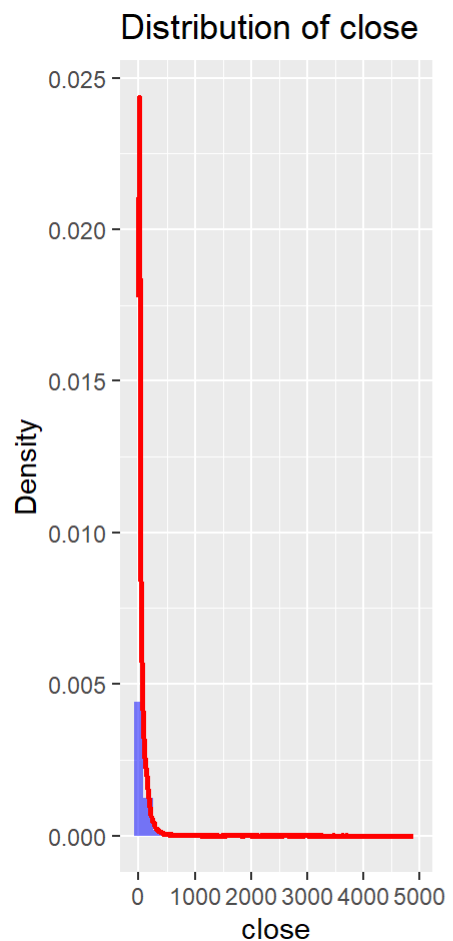
```
##  
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':  
##  
##      combine
```

```
#install.packages("gridExtra")  
# Load data  
# Load necessary libraries  
library(ggplot2)  
library(gridExtra)  
library(dplyr)  
  
# Load data  
#data <- read.csv("final.feature.feed_to_other_models17.05.2024+++.csv")  
  
# List of PCA components  
pca_columns <- c( "close", "PC1", "PC2", "PC3", "PC4", "PC5", "PC6", "PC7")  
pca_columns <- c( "close")  
  
# Create a list to hold the plots  
plots <- list()  
  
# Generate histograms using tidy evaluation  
for (pca in pca_columns) {  
  p <- ggplot(data, aes(x = .data[[pca]])) +  
    geom_histogram(aes(y = ..density..), bins = 30, fill = "blue", alpha = 0.5) +  
    geom_density(color = "red", linewidth = 1) +  
    labs(title = paste("Distribution of", pca), x = pca, y = "Density")  
  plots[[pca]] <- p  
}  
  
# Arrange the plots in a grid  
do.call(grid.arrange, c(plots, ncol = 3))
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.  
## i Please use `after_stat(density)` instead.  
## This warning is displayed once every 8 hours.
```

```
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was  
## generated.
```



```
summary(data)
```

```
##      close      Date      PC1      PC2  
## Min.   :  0.013  Length:1336810  Min.   :-321.156  Min.   :-10.43790  
## 1st Qu.:  6.950  Class :character  1st Qu.: -1.464  1st Qu.: -2.20609  
## Median : 29.800  Mode  :character  Median :  2.298  Median : -0.04554  
## Mean   : 72.798                Mean   :  0.000  Mean   :  0.00000  
## 3rd Qu.: 86.000                3rd Qu.:  4.691  3rd Qu.:  2.13991  
## Max.   :4900.000                Max.    : 11.300  Max.    : 14.30648  
##      PC3      PC4      PC5      PC6  
## Min.   :-22.3097  Min.   :-22.60498  Min.   :-22.32528  Min.   :-41.33976  
## 1st Qu.: -1.0018  1st Qu.: -0.42571  1st Qu.: -0.09744  1st Qu.: -0.19908
```

```
## Median : 0.0193 Median : -0.01867 Median : -0.00279 Median : 0.00184
## Mean : 0.0000 Mean : 0.00000 Mean : 0.00000 Mean : 0.00000
## 3rd Qu.: 1.0148 3rd Qu.: 0.40148 3rd Qu.: 0.09110 3rd Qu.: 0.20026
## Max. : 87.2442 Max. : 45.61329 Max. : 97.42868 Max. : 18.73958
## PC7 PC8
## Min. : -9.51994 Min. : -14.486692
## 1st Qu.: -0.11711 1st Qu.: -0.128074
## Median : 0.02676 Median : -0.008422
## Mean : 0.00000 Mean : 0.000000
## 3rd Qu.: 0.14409 3rd Qu.: 0.119418
## Max. : 11.17001 Max. : 15.148364
```

```
#install.packages("DescTools")
# Load necessary libraries
library(DescTools)
```

```
##
## Attaching package: 'DescTools'
```

```
## The following object is masked from 'package:data.table':
##
## %like%
```

```
library(ggplot2)
library(gridExtra)

# Load data
#data <- read.csv("final.feature.feed_to_other_models17.05.2024+++.csv")

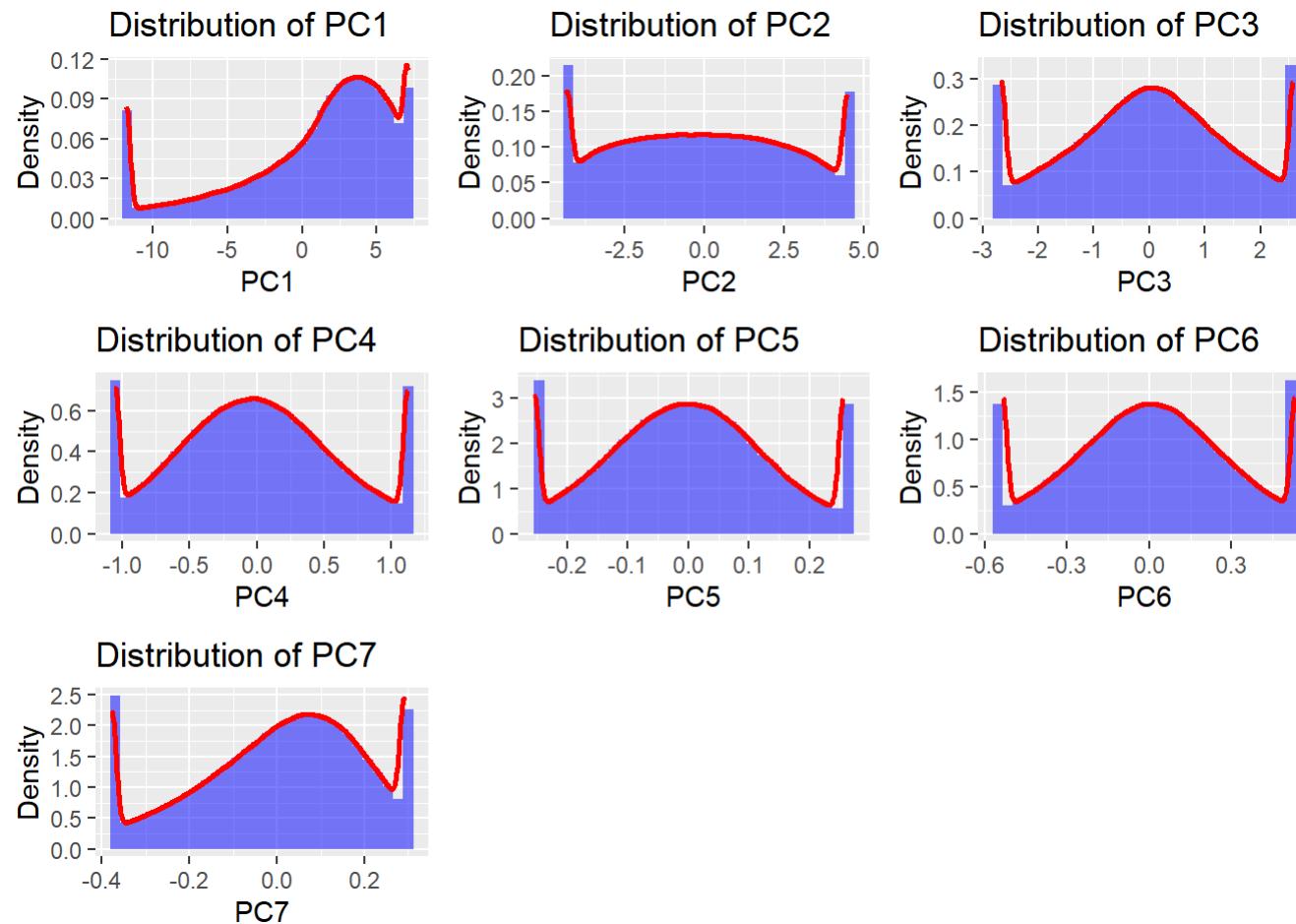
# List of PCA components
pca_columns <- c("PC1", "PC2", "PC3", "PC4", "PC5", "PC6", "PC7")

# Apply winsorization to each PCA component
for (pca in pca_columns) {
  data[[pca]] <- Winsorize(data[[pca]], probs = c(0.05, 0.95))
}

# Create a list to hold the plots
plots <- list()
```

```
# Generate histograms using tidy evaluation
for (pca in pca_columns) {
  p <- ggplot(data, aes(x = .data[[pca]])) +
    geom_histogram(aes(y = ..density..), bins = 30, fill = "blue", alpha = 0.5) +
    geom_density(color = "red", linewidth = 1) +
    labs(title = paste("Distribution of", pca), x = pca, y = "Density")
  plots[[pca]] <- p
}

# Arrange the plots in a grid
do.call(grid.arrange, c(plots, ncol = 3))
```



```
# Load necessary libraries
library(DescTools)
library(dplyr)
```

```

# Load data
#data <- read.csv("final.feature.feed_to_other_models17.05.2024+++.csv")

# Ensure the date column is in datetime format
data$Date <- as.Date(data$Date)

# List of PCA components and other features to winsorize
pca_columns <- c("PC1", "PC2", "PC3", "PC4", "PC5", "PC6", "PC7")

# Store the number of rows before winsorization
num_rows_before <- nrow(data)

# Apply winsorization to each PCA component
for (pca in pca_columns) {
  data[[pca]] <- Winsorize(data[[pca]], probs = c(0.05, 0.95))
}

# Store the number of rows after winsorization
num_rows_after <- nrow(data)

# Print the number of rows before and after winsorization to verify they are the same
cat("Number of rows before winsorization:", num_rows_before, "\n")

```

```
## Number of rows before winsorization: 1336810
```

```
cat("Number of rows after winsorization:", num_rows_after, "\n")
```

```
## Number of rows after winsorization: 1336810
```

```

# Function to check for NA, NaN, and infinite values in each column
check_missing_values <- function(df) {
  sapply(df, function(x) {
    list(
      `NA` = sum(is.na(x)),
      `NaN` = sum(is.nan(x)),
      `PosInf` = sum(is.infinite(x) & x > 0),
      `NegInf` = sum(is.infinite(x) & x < 0)
    )
  })
}

```

```

}

# Check for missing values in the dataset
missing_values_summary <- check_missing_values(data)
print(missing_values_summary)

```

```

##           close Date PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8
## NA         0      0    0    0    0    0    0    0    0
## NaN        0      0    0    0    0    0    0    0    0
## PosInf 0      0      0    0    0    0    0    0    0
## NegInf 0      0      0    0    0    0    0    0    0

```

```

# Save the updated data frame if needed
write.csv(data, "winsorized_datapca final 18.5.2024.csv", row.names = FALSE)

# Check the structure of the updated data
str(data)

```

```

## 'data.frame':   1336810 obs. of  10 variables:
## $ close: num  83 81 80.8 80.8 81.8 82.4 83.4 81.6 82 81 ...
## $ Date : Date, format: "2019-10-01" "2019-10-02" ...
## $ PC1 : num -2.27 -2.16 -1.95 -1.87 -1.59 ...
## $ PC2 : num -2.24 -2.15 -1.92 -1.83 -1.57 ...
## $ PC3 : num  0.4375 -0.0801 -0.2961 -0.3346 -0.1294 ...
## $ PC4 : num -1.058 -1.051 -0.463 0.282 0.772 ...
## $ PC5 : num  0.00852 0.16772 0.20536 0.16463 -0.01503 ...
## $ PC6 : num -0.285 0.266 0.441 0.491 0.108 ...
## $ PC7 : num  0.00741 -0.12834 -0.03038 0.14817 0.02318 ...
## $ PC8 : num  0.0335 -0.1648 -0.0273 0.2401 0.0899 ...

```

```

##### close prices assessing

```

```

# Load necessary libraries
library(dplyr)

```

```

# Define the ranges
ranges <- list(
  "0-50" = c(0, 50),
  "50-100" = c(50, 100),

```



```

"100-500" = c(100, 500),
"500-1000" = c(500, 1000),
"1000+" = c(1000, Inf)
)

# Function to calculate percentage of values in each range
calculate_percentage <- function(data, column, ranges) {
  total_count <- nrow(data)
  percentages <- sapply(ranges, function(range) {
    count <- sum(data[[column]] >= range[1] & data[[column]] < range[2])
    percentage <- (count / total_count) * 100
    return(percentage)
  })
  return(percentages)
}

# Calculate percentages for 'close' prices
percentages <- calculate_percentage(data, "close", ranges)

# Print the results
percentages

```

```

##           0-50      50-100     100-500    500-1000      1000+
## 62.0204816 16.3770469 20.4624442  0.6977057  0.4423216

```

```

# Load necessary libraries
library(ggplot2)

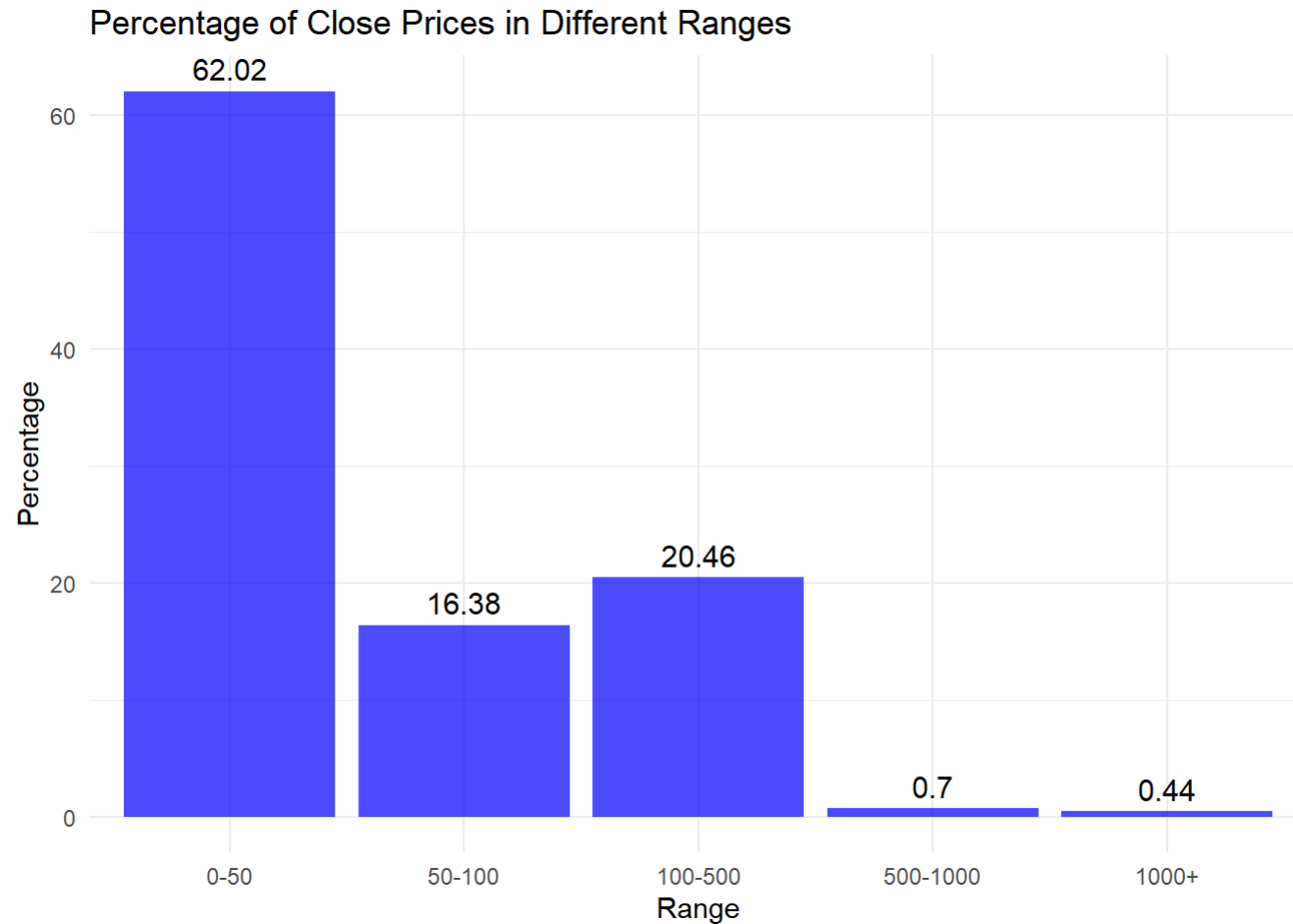
# Define the ranges and percentages
ranges <- c("0-50", "50-100", "100-500", "500-1000", "1000+")
percentages <- c(62.02, 16.38, 20.46, 0.70, 0.44)

# Create a data frame for plotting
percentage_data <- data.frame(
  Range = factor(ranges, levels = ranges),
  Percentage = percentages
)

# Plot the histogram
ggplot(percentage_data, aes(x = Range, y = Percentage)) +
  geom_bar(stat = "identity", fill = "blue", alpha = 0.7) +

```

```
geom_text(aes(label=round(Percentage, 2)), vjust = -0.5, size = 4) +
labs(title = "Percentage of Close Prices in Different Ranges",
      x = "Range",
      y = "Percentage") +
theme_minimal()
```



```
library(dplyr)
library(DescTools)

# Winsorize the 'close' column
data$close <- winsorize(data$close, probs = c(0.05, 0.95))

# Check the summary of the winsorized 'close' column
summary(data$close)
```

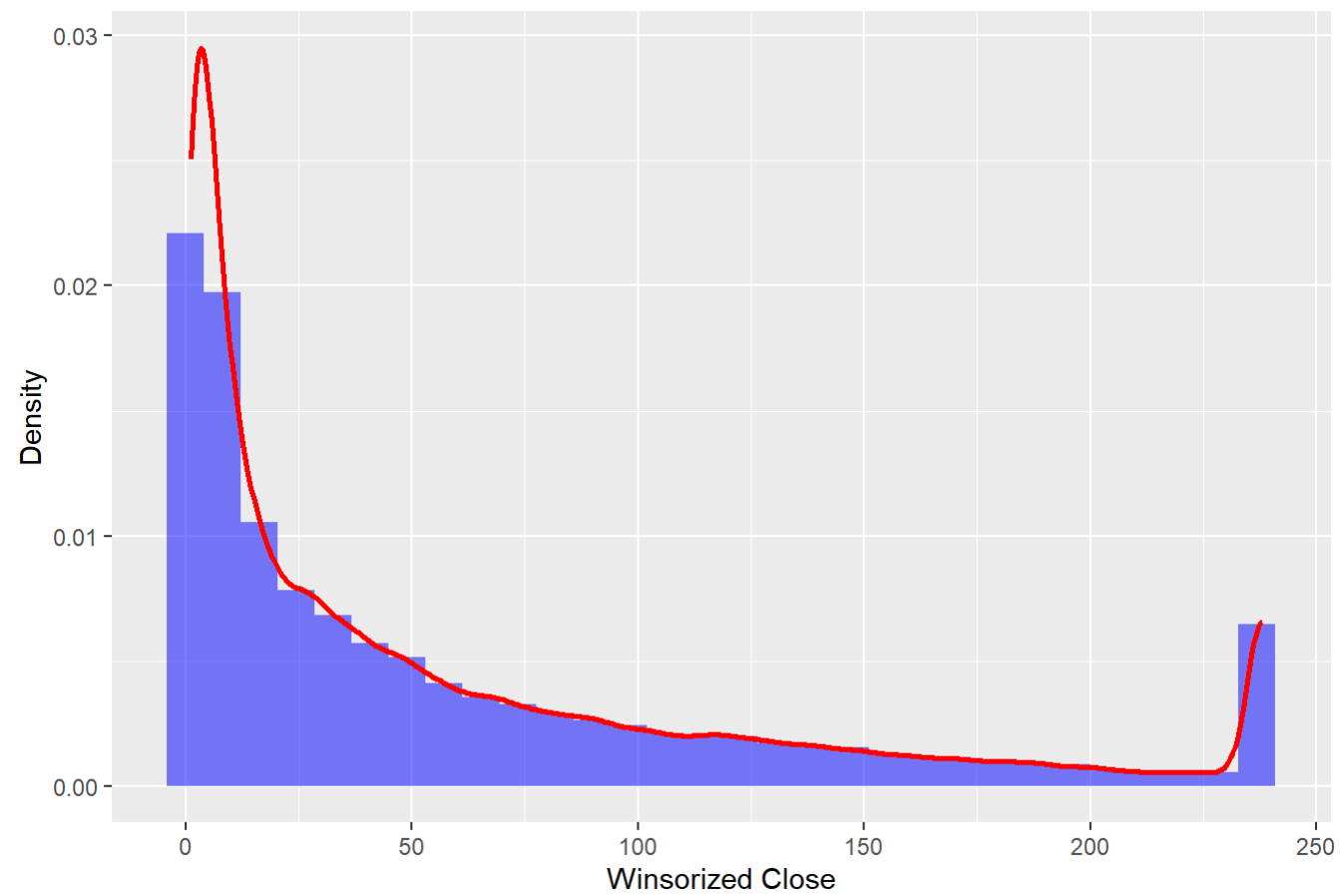
```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.098   6.950   29.800   58.159   86.000  238.000
```

```
# Plot distribution of winsorized 'close' prices
```

```
ggplot(data, aes(x = close)) +  
  geom_histogram(aes(y = ..density..), bins = 30, fill = "blue", alpha = 0.5) +  
  geom_density(color = "red", size = 1) +  
  labs(title = "Distribution of Winsorized Close Prices", x = "Winsorized Close", y = "Density")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## i Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was  
## generated.
```

Distribution of Winsorized Close Prices



```
# Save the updated data frame if needed  
write.csv(data, "winsorized_datapca and clsoe prices final 18.5.2024.csv", row.names = FALSE)
```