library(wbstats)

library(dplyr)

library(data.table)

library(rio)

library(stringr)

library(magrittr)

library(purrr)

library(ggplot2)

library(latex2exp)

library(stargazer)

library(glmnet)

library(corrplot)

library(dplyr)

library(lubridate)

library(rio)

library(stringr)

library(tibble)

library(magrittr)

library(purrr)

library(data.table)

library(tidyr)

library(forecast)

library(gridExtra)

library(grid)

library(Amelia)

library(fastDummies)

library(gnm)

library(parallel)

library(xgboost)

library(rbenchmark)

library(gganimate)

#library(plotly)

library(e1071)

library(rlang)

library(caret)

library(DiagrammeR)

library(sophisthse)

library(DiagrammeRsvg)

library(rsvg)

library(neuralnet)

library(ggridges)

library(tseries)

library(xts)

library(DtD)

library(readr)

library(TTR)

library(zoo)

library(seqHMM)

library(hexbin)

library(sophisthse)

library(tseries)

library(caret)

library(sophisthse)

library(tseries)

library(MLmetrics)

library(caret)

library(xts)

library(randomForest)

library(spikeslab)

library(BigVAR)

library(xtable)

library(readxl)

library(httr)

library(ggpubr)

library(shiny)

library(scales)

library(DT)

library(lemon)

library(shinyWidgets)

library(dplyr)

library(multDM)

library(tables)

library(xtable)

library(gifski)

library(png)

library(transformr)

library(magick)

# загрузка данных sophisthse ----

sophist\_series <- c('UNEMPL\_Q\_SH',# безработица

'EMPLDEC\_Q', # заявленная потребность в работниках

'CONSTR\_Q\_NAT', # индекс строительно-монтажных работ

'TRP\_Q\_PASS\_DIRI', # индекс пассажирских перевозок

'WAG\_Q', # зарплата

'CPI\_Q\_CHI',# ипц

'CONI\_Q\_CHI', # индекс цен на строительно-монтажные работы

'CTI\_Q\_CHI', # индекс тарифов на грузовые перевозки

'AGR\_Q\_DIRI', # индекс сельхоз производства

'CNSTR\_Q\_DIRI',# индекс работ в строительстве

'RTRD\_Q\_DIRI', # оборот розничной торговли

'HHI\_Q\_DIRI',# индекс реальных денежных доходов населения

'M0\_Q', # M0

'M2\_Q',# М2

'IR\_Q',# прямые иностранные инвестиции

'ICR\_Q',# валютные резервы ЦБР

'CBREV\_Q',# доходы конс. бюджета

'CBEX\_Q',# расходы конс. бюджета

'FBREV\_Q',# доходы фед. бюджета

'FBEX\_Q',# расходы фед. бюджета

'RDEXRO\_Q',# официальный курс доллара

'RDEXRM\_Q',# курс доллара на ммвб

'RDEXRMA\_Q',# средний за период курс доллара на ммвб

'LIAB\_T\_Q',# кредиторская задолженность в среднем за период

'LIAB\_UNP\_Q',# просроченная кредиторская задолженность в среднем за период

'LIAB\_S\_Q',# кредиторская задолженность поставщикам в среднем за период

'LIAB\_B\_Q',# кредиторская задолженность в бюджет в среднем за период

'DBT\_T\_Q',#дебиторская задолженность в среднем за период

'DBT\_UNP\_Q',#просроченная дебиторская задолженность в среднем за период

'DBT\_P\_Q',# дебиторская задолженность покупателей в среднем за период

'EX\_T\_Q',# экспорт

'EX\_NON.CIS\_Q',# экспорт в не-снг страны

'IM\_T\_Q',# импорт

'IM\_NON.CIS\_Q', # импорт не из стран снг

'INVFC\_Q', # номинальные инвестиции

# методика расчета ввп поменялась в 2004, поэтому два разных ряда не соответствуют друг другу

'GDPEA\_Q\_DIRI',# индексы реального ввп (после 2004)

'GDP\_Q\_DIRI',# (до 2004)

'GDPEA\_C\_Q',# номинальный ввп (после 2004)

'GDP\_Q\_C',# (до 2004)

# методика расчета индекса цен производителей промтоваров тоже поменялась

'PPI\_Q\_CHI', # индекс цен производителей промтоваров (до 2004)

'PPI\_EA\_Q' # (после 2004-01)

)

sophistdata <- sophisthse(series.name = sophist\_series,

output = 'zoo') %>% as.xts

sophistdata <- sophistdata[, which(colnames(sophistdata) %in% sophist\_series)]

# заменяем значения в колонках для индекса реального ввп и для номинального ввп

sophistdata$GDPEA\_Q\_DIRI[which(is.na(sophistdata$GDPEA\_Q\_DIRI))] <-

sophistdata$GDP\_Q\_DIRI[which(is.na(sophistdata$GDPEA\_Q\_DIRI))]/1.1515

sophistdata$GDPEA\_C\_Q[which(is.na(sophistdata$GDPEA\_C\_Q))] <-

sophistdata$GDP\_Q\_C[which(is.na(sophistdata$GDPEA\_C\_Q))]

# удаляем лишние столбцы с ввп

sophistdata$GDP\_Q\_C <- sophistdata$GDP\_Q\_DIRI <- NULL

# заменяем значения для индекса цен производителей промтоваров

sophistdata$PPI\_EA\_Q[which(is.na(sophistdata$PPI\_EA\_Q))] <-

sophistdata$PPI\_Q\_CHI[which(is.na(sophistdata$PPI\_EA\_Q))]

# удаляем лишний столбец (данные до 2004)

sophistdata$PPI\_Q\_CHI <- NULL

# загрузка индекса реальных инвестиций ----

# первая часть данных -- с 1995-01 по 2008-12

url <- 'http://www.gks.ru/free\_doc/new\_site/vvp/kv/tab29arh.xls'

GET(url, write\_disk(tf <- tempfile(fileext = ".xls")))

inv1 <- read\_excel(tf, 1L,skip=14, col\_names = FALSE) %>%

.[1,-1] %>%

t %>%

as.numeric %>%

xts(x = .,

order.by = seq(as.Date('1995-01-01'),

by = 'quarter',

length.out = length(.)) %>% as.yearqtr) %>%

set\_colnames('investment')

# вторая часть данных -- с 2003-01 по 2011-12

# внутри есть примечания вида 2)

# их надо удалить

url <- 'http://www.gks.ru/free\_doc/new\_site/vvp/kv/tab29.xls'

GET(url, write\_disk(tf <- tempfile(fileext = ".xls")))

inv2 <- read\_excel(tf, 1L,skip=13, col\_names = FALSE) %>%

.[1,-1] %>%

t %>%

gsub(pattern = "2)",

replacement = "",

x = .) %>%

gsub(pattern = ",",

replacement = ".",

x = .) %>%

as.numeric %>%

xts(x = .,

order.by = seq(as.Date('2003-01-01'),

by = 'quarter',

length.out = length(.)) %>% as.yearqtr) %>%

set\_colnames('investment')

# 3 часть данных -- с 2011-01 по 2016-12

url <- 'http://www.gks.ru/free\_doc/new\_site/vvp/kv/tab29a.xls'

GET(url, write\_disk(tf <- tempfile(fileext = ".xls")))

inv3 <- read\_excel(tf, 1L,skip=12, col\_names = FALSE) %>%

.[1,-1] %>%

t %>%

as.numeric %>%

xts(x = .,

order.by = seq(as.Date('2011-01-01'),

by = 'quarter',

length.out = length(.)) %>% as.yearqtr) %>%

set\_colnames('investment')

# 4 часть данных -- с 2014-01

url <- 'http://www.gks.ru/free\_doc/new\_site/vvp/kv/tab29b.xls'

GET(url, write\_disk(tf <- tempfile(fileext = ".xls")))

inv4 <- read\_excel(tf, 1L,skip=12, col\_names = FALSE) %>%

.[1,-1] %>%

t %>%

as.numeric %>%

xts(x = .,

order.by = seq(as.Date('2014-01-01'),

by = 'quarter',

length.out = length(.)) %>% as.yearqtr) %>%

set\_colnames('investment')

# привести в соответствие со последующим индексом

inv3[,1] <- (inv3[,1])\*(inv4['2014-01',1] %>% as.numeric)/(inv3['2014-01',1] %>% as.numeric)

inv2[,1] <- (inv2[,1])\*(inv3['2011-01',1] %>% as.numeric)/(inv2['2011-01',1] %>% as.numeric)

inv1[,1] <- (inv1[,1])\*(inv2['2003-01',1] %>% as.numeric)/(inv1['2003-01',1] %>% as.numeric)

inv <- rbind(inv1['1995-01/2002-12'], inv2['2003-01/2010-12'],inv3['2011-01/2013-12'],

inv4['2014-01/2020-1'])

inv <- inv\*100/(inv %>% first %>% as.numeric)

# загрузка квартального дефлятора ----

# росстат представляет данные в % по отношению к соответствующему кварталу прошлого года

# первая часть данных -- с 1996-01 по 2011-12

url1 <- 'http://www.gks.ru/free\_doc/new\_site/vvp/kv/tab9.xls'

GET(url1, write\_disk(tf1 <- tempfile(fileext = ".xls")))

def1 <- read\_excel(tf1, 1L, skip = 5, col\_names = FALSE) %>%

.[1,] %>%

t %>%

as.numeric

# вторая часть данных -- с 2012-01 по последний день

url2 <- 'http://www.gks.ru/free\_doc/new\_site/vvp/kv/tab9a.xls'

GET(url2, write\_disk(tf2 <- tempfile(fileext = ".xls")))

def2 <- read\_excel(tf2, 1L, skip = 5, col\_names = FALSE)%>%

.[1,] %>%

t %>%

as.numeric

# склеиваем дефлятор

def <- xts(c(def1, def2),

order.by = seq(as.Date('1996-01-01'),

by = 'quarter',

length.out = length(c(def1, def2))) %>% as.yearqtr) %>%

set\_colnames("deflator")

# Загрузка ставок мбр ----

# Показатели ставок межбанковского рынка (статистика ЦБ РФ)

# 03.01.1996 - 01.08.2000:

# http://www.cbr.ru/hd\_base/mkr/mkr\_base\_old/

# уточнение: после загрузки в csv файлах ','->'.' and ';'->',' and '-'>''

# Фактические ставки по предоставленным кредитам

# (MIACR - Moscow InterBank Actual Credit Rate)

# (в процентах годовых для рублевых кредитов)

# -Средневзвешанные фактические

mkr1 <- import('data/mkr\_old.csv') %>%

set\_colnames(c("date", '1d', '3d', '7d','14d', '21d', '30d', '60d', '90d')) %>%

mutate(date=as.Date(date,format = '%d.%m.%Y')) %>%

xts(x = .[,-1],order.by = .[,1]) %>%

# в данных за 1996--2000 есть повторы

.[!duplicated(time(.))]

# без серьезных пропусков есть только данные:

# 1d, 7d, 14d, 30d

# c 01.08.2000

# https://www.cbr.ru/hd\_base/mkr/mkr\_base/

# Средневзвешенные фактические ставки по кредитам,

# предоставленным московскими банками (MIACR) с 01.08.2000

mkr2 <- import('data/mkr.csv') %>%

set\_colnames(c("date", '1d', '2-7d', '8-30d','31-90d', '91-180d', '181-365d')) %>%

mutate(date=as.Date(date,format = '%d.%m.%Y')) %>%

xts(x = .[,-1],order.by = .[,1])

# без серьезных пропусков есть только данные:

# 1d, 2-7d,

# от 8 до 365 данные обновляются очень редко (или почти никогда)

# совмещать имеет смысл только по 1d -- 1d и 7d -- 2-7d

mkr <- merge(mkr1[,c('1d', '7d')],mkr2[,c('1d', '2-7d')]) %>%

set\_colnames(c('1d', '7d', '1d\_old', '2-7\_old'))

# склейка

mkr$`1d`[which(is.na(mkr$`1d`))] <- mkr$`1d\_old`[which(is.na(mkr$`1d`))]

mkr$`7d`[which(is.na(mkr$`7d`))] <- mkr$`2-7\_old`[which(is.na(mkr$`7d`))]

# удаление лишних колонок

mkr$`1d\_old` <- mkr$`2-7\_old` <- NULL

# необходимо перевести дневные данные в квартальные

# первый вариант - средние за квартал (реализован)

# второй вариант - на последнюю дату квартала

mkr %<>%

as.data.frame %>%

rownames\_to\_column('date') %>%

group\_by(date %>% as.Date %>% as.yearqtr) %>%

summarise(mkr\_1d = mean(`1d`, na.rm=TRUE),

mkr\_7d = mean(`7d`, na.rm=TRUE)) %>%

as.data.frame %>%

xts(x=.[,-1],order.by=.[,1])

# загрузка данных по гособлигациям ----

# данные не по всем облигациям есть с 1995

# зато для того времени есть данные по ГКО (месячные)

gko <- sophisthse('GKO\_M', 'zoo')[,'GKO\_M'] %>%

as.xts %>%

na.locf %>%

set\_colnames('GKO')

gko %<>% .[mod(month(time(.)), 3)==0,]

time(gko) <- as.yearqtr(time(gko))

# с 2004 года вместо ГКО в ряду используется доходность ОБР (можно удалить)

gko['2004-01/2019-12'] <- NA

# данные по доходности 6-месячных облигаций есть с 1995-03

# https://ru.investing.com/rates-bonds/russia-6-month-bond-yield-historical-data

# (скачать недельные (из-за дат). Порядок изменения файлов csv (через блокнот):

# 1. "," -> ;

# 2. , -> .

# 3. ; -> ,

# 4. " ->

# 5. % ->

gov\_6m <- import('data/gov\_6m.csv') %>%

set\_colnames(c("date", 'close', 'open', 'max','min', 'change')) %>%

mutate(date = as.Date(date, format = "%d.%m.%Y")) %>%

as.data.frame %>%

group\_by(as.yearqtr(date)) %>%

filter(row\_number() == min(row\_number())) %>%

as.data.frame %>%

xts(x=.[,2],order.by=.[,1] %>% as.yearqtr) %>%

set\_colnames('gov\_6m')

# данные по доходности годовых облигаций --- с 2001-08

# https://ru.investing.com/rates-bonds/russia-1-year-bond-yield-historical-data

# изменить файл аналогично предыдущей инструкции

# 1. "," -> ;

# 2. , -> .

# 3. ; -> ,

# 4. " ->

# 5. % ->

gov\_1y <- import('data/gov\_1y.csv') %>%

set\_colnames(c("date", 'close', 'open', 'max','min', 'change')) %>%

mutate(date = as.Date(date, format = "%d.%m.%Y")) %>%

as.data.frame %>%

group\_by(as.yearqtr(date)) %>%

filter(row\_number() == min(row\_number())) %>%

as.data.frame %>%

xts(x=.[,2],order.by=.[,1] %>% as.yearqtr) %>%

set\_colnames('gov\_1y')

# 3-летних --- с 2001-03

# https://ru.investing.com/rates-bonds/russia-3-year-bond-yield-historical-data

# изменить файл аналогично предыдущей инструкции

# 1. "," -> ;

# 2. , -> .

# 3. ; -> ,

# 4. " ->

# 5. % ->

gov\_3y <- import('data/gov\_3y.csv') %>%

set\_colnames(c("date", 'close', 'open', 'max','min', 'change')) %>%

mutate(date = as.Date(date, format = "%d.%m.%Y")) %>%

as.data.frame %>%

group\_by(as.yearqtr(date)) %>%

filter(row\_number() == min(row\_number())) %>%

as.data.frame %>%

xts(x=.[,2],order.by=.[,1] %>% as.yearqtr) %>%

set\_colnames('gov\_3y')

gov <- merge.xts(gov\_6m, gov\_1y, gov\_3y)

# загрузка данных из bloomberg (цена нефти, эффективный обменный курс и индекс RTS) ----

# Квартальные данные

# файл bloomberg.csv

# Russia Real Effective Exchange Rate Broad (BISBUR)

# Russia Nominal Effecive Exchange Rate (BISBRUN)

# CO1 Comdty (Oil Brent) last price

# RTS last price

# сохранить excel как csv

# действия:

# 1. , -> .

# 2. ; -> ,

bloomberg <- import('data/bloomberg.csv') %>%

set\_colnames(c("date", 'reer', 'neer', 'oil','rts')) %>%

mutate(date = as.Date(date, format = "%d.%m.%Y") %>% as.yearqtr) %>%

xts(x=.[,-1],order.by=.[,1])

# конечная склейка данных ----

rawdata <- merge.xts(inv,sophistdata, def, mkr, gov,gko, bloomberg)

save(rawdata, file = "data/raw.RData")

rio::export(rawdata, 'data/raw.csv', 'csv', row.names = TRUE)

rm(list=ls())

source('lib.R')

load('data/raw.Rdata')

# # доходность ртс, процентные ставки на межбанковском рынке, спреды по облигациям,

# темпы роста номинального эффективного курса, реального эффективного курса, прирост цен на нефть,

# 4 разность логарифма ВВП и ИПЦ, отношение номинальных инвестиций к номинальному ввп + лаги

series <- c('investment',

'mkr\_1d',

'mkr\_7d',

'gov\_6m',

'GKO',

'reer',

'neer',

'oil',

'rts',

'CPI\_Q\_CHI',

'GDPEA\_Q\_DIRI',

'GDPEA\_C\_Q',

'INVFC\_Q',

'EMPLDEC\_Q',

)

df <- rawdata[,series]

# отношение номинальных инвестиций к номинальному ВВП

df$invest2gdp <- df$INVFC\_Q/df$GDPEA\_C\_Q

# удалить лишнее

df$GDPEA\_C\_Q <- df$INVFC\_Q <- NULL

# вставить gko в 6m gov bond yield

df$gov\_6m[which(is.na(df$gov\_6m))] <-

df$GKO[which(is.na(df$gov\_6m))]

df$GKO <- NULL

# нестационарные ряды без сезонности (разность) ----

for(i in c('reer','neer','oil','rts')){

df[,i] %<>% diff.xts(log=TRUE)

}

# нестационарные ряды с сезонностью (4-ая разность) ----

for(i in c('investment', 'CPI\_Q\_CHI',

'invest2gdp',

'GDPEA\_Q\_DIRI',

'EMPLDEC\_Q')){

df[,i] %<>% diff.xts(lag = 4, log=TRUE)

}

save(df, file = 'data/stationary\_data.RData')

# extended series ----

eseries <- c('investment', 'mkr\_1d','mkr\_7d','gov\_6m','GKO',

'reer', 'neer', 'oil', 'rts',

'GDPEA\_C\_Q', 'INVFC\_Q', # служебные для invest2gdp

'CPI\_Q\_CHI',# ипц

'GDPEA\_Q\_DIRI', # ввп

'EMPLDEC\_Q', # заявленная потребность в работника

'UNEMPL\_Q\_SH',# безработица

'CONSTR\_Q\_NAT', # индекс строительно-монтажных работ,

'WAG\_Q', # зарплата

'CONI\_Q\_CHI', # индекс цен на строительно-монтажные работы

'CTI\_Q\_CHI', # индекс тарифов на грузовые перевозки

'AGR\_Q\_DIRI', # индекс сельхоз производства

#'CNSTR\_Q\_DIRI',# индекс работ в строительстве

'RTRD\_Q\_DIRI', # оборот розничной торговли

'HHI\_Q\_DIRI',# индекс реальных денежных доходов населения

'M0\_Q', # M0

'M2\_Q',# М2

'CBREV\_Q',# доходы конс. бюджета

'CBEX\_Q',# расходы конс. бюджета

'FBREV\_Q',# доходы фед. бюджета

'FBEX\_Q',# расходы фед. бюджета

'RDEXRO\_Q',# официальный курс доллара

'RDEXRM\_Q',# курс доллара на ммвб

'LIAB\_T\_Q',# кредиторская задолженность в среднем за период

'LIAB\_UNP\_Q',# просроченная кредиторская задолженность в среднем за период

'LIAB\_S\_Q',# кредиторская задолженность поставщикам в среднем за период

'LIAB\_B\_Q',# кредиторская задолженность в бюджет в среднем за период

'DBT\_T\_Q',#дебиторская задолженность в среднем за период

'DBT\_UNP\_Q',#просроченная дебиторская задолженность в среднем за период

# 'DBT\_P\_Q',# дебиторская задолженность покупателей в среднем за период

'EX\_T\_Q',# экспорт

'IM\_T\_Q',# импорт

'PPI\_EA\_Q' # (после 2004-01)

)

df <- rawdata[,eseries]

# отношение номинальных инвестиций к номинальному ВВП

df$invest2gdp <- df$INVFC\_Q/df$GDPEA\_C\_Q

# удалить лишнее

df$GDPEA\_C\_Q <- df$INVFC\_Q <- NULL

# вставить gko в 6m gov bond yield

df$gov\_6m[which(is.na(df$gov\_6m))] <-

df$GKO[which(is.na(df$gov\_6m))]

df$GKO <- NULL

# нестационарные ряды без сезонности (разность) ----

for(i in c('reer','neer','oil','rts')){

df[,i] %<>% diff.xts(log=TRUE)

}

# нестационарные ряды с сезонностью (4-ая разность) ----

for(i in c('investment', 'CPI\_Q\_CHI',

'invest2gdp',

# 'deflator', только с 1996

'GDPEA\_Q\_DIRI',

'EMPLDEC\_Q',

'UNEMPL\_Q\_SH',

'CONSTR\_Q\_NAT',

###### 'TRP\_Q\_PASS\_DIRI',

'WAG\_Q',

'CONI\_Q\_CHI',

'CTI\_Q\_CHI',

'AGR\_Q\_DIRI',

'RTRD\_Q\_DIRI',

'HHI\_Q\_DIRI',

'M0\_Q',

'M2\_Q',

#### 'IR\_Q',

#### 'ICR\_Q',

'CBREV\_Q',

'CBEX\_Q',

'FBREV\_Q',

'FBEX\_Q',

'RDEXRO\_Q',# официальный курс доллара

'RDEXRM\_Q',# курс доллара на ммвб

'LIAB\_T\_Q',# кредиторская задолженность в среднем за период

'LIAB\_UNP\_Q',# просроченная кредиторская задолженность в среднем за период

'LIAB\_S\_Q',# кредиторская задолженность поставщикам в среднем за период

'LIAB\_B\_Q',# кредиторская задолженность в бюджет в среднем за период

'DBT\_T\_Q',#дебиторская задолженность в среднем за период

'DBT\_UNP\_Q',#просроченная дебиторская задолженность в среднем за период

########## 'DBT\_P\_Q',# дебиторская задолженность покупателей в среднем за период

'EX\_T\_Q',# экспорт

'IM\_T\_Q',# импорт

'PPI\_EA\_Q' # (после 2004-01)

)){

print(i)

df[,i] %<>% diff.xts(lag = 4, log=TRUE)

}

#df['1996-01/2019-06'] %>% missmap()

# lags

# df$gdplag <- lag.xts(df$GDPEA\_Q\_DIRI, k = 1)

# df$investmentlag <- lag.xts(df$investment, k = 1)

# df$invest2gdplag <- lag.xts(df$invest2gdp, k = 1)

save(df, file = 'data/stationary\_data\_ext.RData')

# oil data

load('data/stationary\_data\_ext.RData')

df <- df[,c('investment', 'oil')]

save(df, file = 'data/stationary\_data\_oil.RData')

train.model <- function(startdt= as.Date('1996-01-01'),

enddt = as.Date('2016-10-01'),

model,

# series parameter

# only for regularisation and machine learning models

series='e',

lag = 0L,

h = 1L,

target = c('investment','RTRD\_Q\_DIRI','GDPEA\_Q\_DIRI','UNEMPL\_Q\_SH', 'CPI\_Q\_CHI'),

# parameters for durable evaluations with function arguments from expand.grid table

i = NULL,

N = NULL

){

message(paste0(i, '/', N))

# import df

# seed

set.seed(2019)

if(series == ''){

load('~/investment\_forecasting/data/stationary\_data.RData')

} else if(series == 'e') {

load('~/investment\_forecasting/data/stationary\_data\_ext.RData')

} else if(series == 'oil'){

load('~/investment\_forecasting/data/stationary\_data\_oil.RData')

}

target <- match.arg(target)

if(!model %in% c('arima', 'rw')){

if(!series %in% c('', 'e', 'oil')){

df %<>% df[, c(target,

series)]

}

# create lag

if(lag != 0 |! model %in% c('arima', 'rw')){

df %<>% create\_lag(lag)

}

if(!is.integer(h)){

message('h must be integer')

return(NULL)

}

df$y <- lag.xts(df[,target], k = -h)

if(h == 0){

if(target == 'investment'){

df$investment <-df$invest2gdp <- NULL

} else{

df %<>% .[, colnames(.) != target]

}

}

df %<>% .[rowSums(is.na(.[,colnames(.)!='y']))==0,]

# проверка на start и end

if(startdt >= enddt){

message('start must be greater then end')

return(NULL)

}

startdt <- max(startdt %>% as.yearqtr,

first(time(df)) %>% as.yearqtr) %>%

as.Date

enddt <- min(enddt %>% as.yearqtr,

last(time(df)) %>% as.yearqtr) %>%

as.Date

if(df[paste0(startdt, "/", enddt)] %>% nrow < 48){

message('train set length must be greater then 48 quarters')

return(NULL)

}

if(which((time(df) %>% as.Date())==(enddt%>% as.yearqtr%>% as.Date))+1>length(df$y %>% na.omit)){

#message('enddt must not be greater then last date when investment data is avaliable minus h quarters')

return(NULL)

}

train\_n <- which(time(df) %>% as.Date()==(startdt %>% as.yearqtr %>% as.Date)):

which((time(df) %>% as.Date())==

(enddt %>% as.yearqtr%>% as.Date))

test\_n <- (which((time(df) %>% as.Date())==(enddt%>% as.yearqtr%>% as.Date))+1):nrow(df)

df$y %<>% na.fill(0)

X.matrix <- model.matrix(y~0+., data = df)

X.train <- X.matrix[train\_n,]

X.test <- X.matrix[test\_n,]

y.train <- df$y[train\_n] %>% as.numeric

y.test <- df$y[test\_n] %>% as.numeric

# tc <- trainControl(method = 'cv', number = 10)

tc <- trainControl(method = "timeslice",

initialWindow = 40,

horizon = 1,

skip=0,

fixedWindow = TRUE)

lambda <- seq(-8,-1,length = 200) %>% exp()

if(model == 'lasso'){

train.out <- train(x=X.train,

y=y.train,

method = "glmnet",

metric = "RMSE",

trControl = tc,

tuneGrid =

expand.grid(.alpha = c(1),

.lambda = lambda))

model\_fit <- glmnet(X.train,

y.train,

alpha = 1,

lambda = train.out$bestTune[1,2])

pred <- predict(model\_fit,

newx = rbind(X.train, X.test)) %>% as.numeric

}

else if(model == 'elnet'){

train.out <- train(x=X.train,

y=y.train,

method = "glmnet",

metric = "RMSE",

trControl = tc,

tuneGrid =

expand.grid(.alpha = c(0.5),

# new

.lambda = lambda))

model\_fit <- glmnet(X.train,

y.train,

alpha = 0.5,

lambda = train.out$bestTune[1,2])

pred <- predict(model\_fit,

newx = rbind(X.train, X.test)) %>% as.numeric

}

else if(model == 'ridge'){

train.out <- train(x=X.train,

y=y.train,

method = "glmnet",

metric = "RMSE",trControl = tc,

tuneGrid = expand.grid(.alpha = 0,

.lambda = lambda))

model\_fit <- glmnet(X.train,

y.train,

alpha = 0,

lambda = train.out$bestTune[1,2])

pred <- predict(model\_fit,newx = rbind(X.train, X.test)) %>% as.numeric

}

else if (model == 'adalasso'){

train.ridge <- train(x=X.train,

y=y.train,

method = "glmnet",

metric = "RMSE",trControl = tc,

tuneGrid = expand.grid(.alpha = 0,.lambda = lambda))

m\_ridge <- glmnet(X.train,

y.train,

alpha = 0,

lambda = train.ridge$bestTune[1,2])

w3 <- 1/abs(as.numeric(coef(m\_ridge))

[1:(ncol(X.train))] )^0.5 ## Using gamma = 1

w3[which(w3 == Inf)] <- 999999999 ## Replacing values estimated as Infinite for 999999999

train.out <- train(x=X.train,

y=y.train,

penalty.factor = w3,

method = "glmnet",

metric = "RMSE",

trControl = tc,

tuneGrid = expand.grid(.alpha = 1,.lambda = lambda))

model\_fit <- glmnet(X.train,

y.train,

alpha = 1,

lambda = train.out$bestTune[1,2],

penalty.factor = w3)

pred <- predict(model\_fit,newx = rbind(X.train, X.test)) %>%

as.numeric

} else if (model == 'postlasso'){

train.out <- train(x=X.train,

y=y.train,

method = "glmnet",

metric = "RMSE",trControl = tc,

tuneGrid = expand.grid(.alpha = c(1),.lambda = lambda))

model\_lasso <- glmnet(X.train,

y.train,

alpha = 1,

lambda = train.out$bestTune[1,2])

nzvars <- predict(model\_lasso, type = "nonzero") %>% .[[1]]

train\_post <- as.data.frame(X.train) %>%

select(nzvars) %>%

mutate(y = y.train)

if(class(X.test)=='numeric'){

X.test %<>% as.tibble %>% t %>% set\_colnames(colnames(X.train))

}

test\_post <- as.data.frame(X.test) %>%

select(nzvars) %>%

mutate(y = 0)

model\_fit <- lm(y~., data = train\_post)

pred <- predict(model\_fit,newdata = rbind(train\_post,test\_post ))

} else if (model == 'rf'){

train.out <- NULL

model\_fit <- randomForest(x = X.train, y = y.train, ntree = 100)

pred <- predict(model\_fit, newdata = rbind(X.train, X.test)) %>%

as.numeric

} else if (model =='ss'){

train.out <- NULL

model\_fit <- spikeslab(x = X.train,

y = y.train,

n.iter2 = 500,

bigp.smalln = ncol(X.train)>=nrow(X.train),

intercept = TRUE)

pred <- predict(model\_fit, newdata = rbind(X.train, X.test))$yhat.gnet

} else if (model == 'boost'){

tune\_grid <- expand.grid(nrounds = 100,

max\_depth = c(5),

eta = c(0.2),

gamma = 0,

colsample\_bytree = 0.3,

min\_child\_weight = 1,

subsample = 1)

train.out <- NULL

model\_fit <- train(x = X.train,

y = y.train,

method = "xgbTree",

metric = "RMSE",

tuneGrid = tune\_grid)

pred <- predict(model\_fit, newdata = rbind(X.train, X.test)) %>%

as.numeric

}

} else

{

if(!is.integer(h)){

message('h must be integer')

return(NULL)

}

df %<>% .[,'investment']

df$y <- df$investment

df %<>% na.omit

# проверка на start и end

if(startdt >= enddt){

message('start must be greater then end')

return(NULL)

}

startdt <- max(startdt %>% as.yearqtr,

first(time(df)) %>% as.yearqtr) %>%

as.Date

enddt <- min(enddt %>% as.yearqtr,

last(time(df)) %>% as.yearqtr) %>%

as.Date

if(df[paste0(startdt, "/", enddt)] %>% nrow < 48){

message('train set length must be greater then 48 quarters')

return(NULL)

}

train\_n <- which(time(df) %>% as.Date()==(startdt %>% as.yearqtr %>% as.Date)):

which((time(df) %>% as.Date())==

(enddt %>% as.yearqtr%>% as.Date))

test\_n <- (which((time(df) %>% as.Date())==(enddt%>% as.yearqtr%>% as.Date))+1):nrow(df)

y.train <- df$y[train\_n] %>% as.numeric

y.test <- df$y[test\_n] %>% as.numeric

y.full <- c(y.train, y.test)

if (model == 'arima'){

train.out <- NULL

model\_fit <- auto.arima(y.train,seasonal = FALSE, stationary = TRUE)

maxord <- arimaorder(model\_fit) %>% sum

if(h == 0){

pred <- Arima(y.full, model=model\_fit) %>% fitted() %>% as.numeric()

} else {

if(maxord == 1){

pred <- c()

} else{

pred <- rep(NA, maxord-1)

}

for(i in maxord:length(y.full)){

fit <- Arima(y.full[1:i],

model=model\_fit)

pred <- c(pred, forecast(fit, h) %>% as.data.frame %>% .[h,1])

}

}

} else if (model == 'rw'){

if(h == 0){

pred <- lag.xts(y.full)

} else{

pred <- y.full

}

train.out <- model\_fit <- NULL

}

}

list(

model = model,

series = series,

lag = lag,

startdt = startdt,

enddt = enddt,

date = time(df)[c(train\_n,test\_n)] %>% as.Date,

h = h,

pred = pred,

train.out = train.out,

model\_fit = model\_fit

)

}

out\_rf <- expand.grid(startdt = c(as.Date('1996-01-01'), as.Date('2000-01-01')),

enddt = seq(as.Date('2012-10-01'), as.Date('2018-10-01'), by = 'quarter'),

lag = c(0L),

h=c(0L:8L),

model = c(‘rf’)

) %T>%

(function(x) {assign('N', nrow(x), envir = globalenv())}) %>%

split(seq(1:nrow(.))) %>%

imap(function(x, i){

train.model(startdt=x$startdt,

enddt=x$enddt,

model = x$model,

lag=x$lag,

h=x$h,

i = i,

N = N

)

})

save(out\_rf,

file = 'out/full/out\_rf.RData')