Final Project

Taiga Hasegawa

2019/4/26

EXECUTIVE SUMMARY

Finacial data is often said to be difficult for analysis becase of its non-stationarity, few available data and difficulty of understanding relationship of each data. Recently, however, accuracy of prediction in financial data is increasing due to development of machine learning, increased computational capability, and quantmental approach. In this paper, quantmental approach and some machine learning models were used and compared. Especially using image of futures chart was the most intersting trial. The model that was developed in this paper achived 62.5% accuracy, which is much higher than random guess, educated guess and majority guess.

I. INTRODUCTION

Futures markets has its origins in the commodities industry. Farmers, oil and gas producers, miners, and others whose business was to produce commodities wanted a way to manage the risk of having to accept an uncertain price for their future production. Futures contracts were the answer, and they met the needs of many market participants.

How a futures contract works is actually fairly simple. Under a futures contract, the contract seller agrees to sell a fixed amount of a certain commodity to the contract buyer on a particular day in the future. Most importantly, the price that the buyer will pay the seller is set based on the prevailing futures market price at the time the two parties enter into the contract.

In this paper, the effect of moving average price, vloumes and other technical indicators on returns was tested. Futhermore, the usefulness of using image of futures chart was also assessed in this paper. It would be great help with get capital gain if we could know the future returns from these models.

II. DATA DESCRIPTION

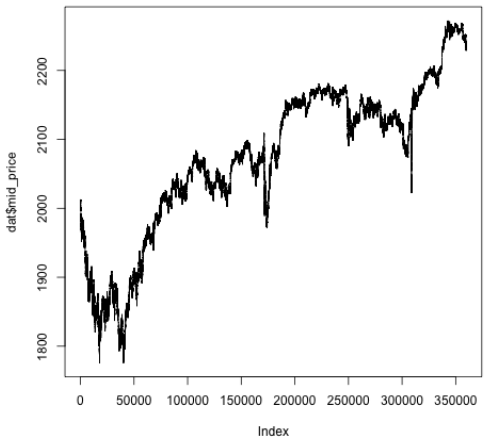
ES (E-Mini S&P 500) was mainly used to model the relationship between previous data and future return. The E-mini S&P 500 is an electronically-traded futures contract on the Chicago Mercantile Exchange representing one-fifth of the value of the standard S&P 500 futures contract. The data was provided by Larry Hua in University of Illinois at Urbana Champaign and license of this data only allowed students who took STAT430: Machine Learning for Financial Data.

This data has 359,887 samples and 26 colums. Each observation had minute-to-minute volume weighted average prices and 10 levels limit order books volume and indicated the type of contract, date, hour and minutes. The description of each variable is like below.

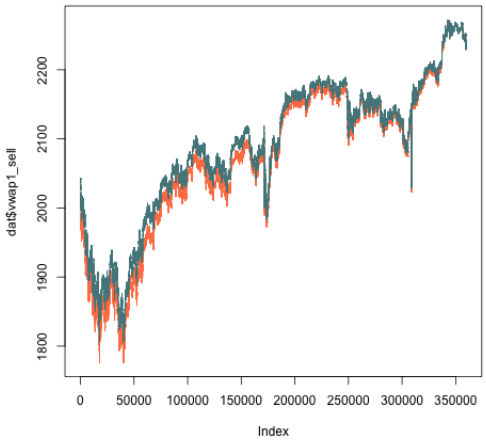
- contract: The name of the contract
- date: Date in "YYYYMMDD" format
- h: Hour
- m: Minute
- vwap1_buy: Volume weighted average prices of buy side in time period
- vwap1_sell: Volume weighted average prices of sell side in time period
- v(Number)_buy: Buy side's (Number)th limit order books volume of ES transacted in time period.
- v(Number)_sell: Sell side's (Number)th limit order books volume of ES transacted in time period.

III. EXPLORATORY ANALYSIS

The following graph is mid_price vs time plot. Mid_price was calculated by taking the mean of vwap1_buy and vwap1_sell. At first glance, it's obvious that the trend went down in the early time of dataset and then it started going up. However, the price drifted and sometiems droped severly while the trend was going up. Thefore, capturing these kinds of movments beforehand is meaningful.



Before starting feature engineering and analysis, there was a problem called "rollover". Rollover was switching from the front month contract that is close to expiration to another contract in a further-out month. This caused price gap before and after rollover and so this gap must be compensated by their price gap. The orange line is the one before considering rollover and blue one is the one after compensating price gap.



IV. FEATURE ENGIEERING

A. Technical indicator

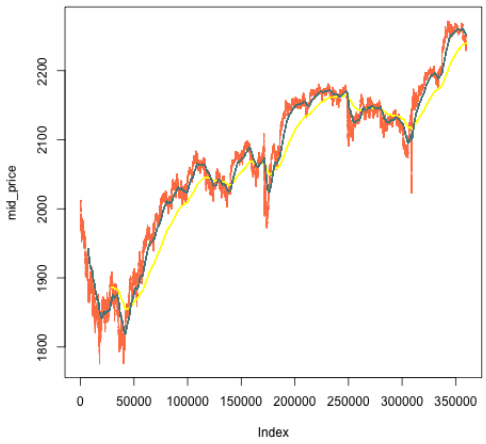
A technical indicator is a mathematical calculation based on historic price, volume or open interest of a security or contract used by traders who follow technical analysis. They were used to predict future price movements in this project. There are two basic types of technical indicators:

- Overlays: Technical indicators that use the same scale as prices are plotted over the top of the prices on a stock chart. Examples include moving averages and Bollinger Bands.
- Oscillators: Technical indicators that oscillate between a local minimum and maximum are plotted above or below a price chart. Examples include the stochastic oscillator, MACD or RSI.

Following technical indicators were ued as predictors in the following model and were seen if they really had an impact on future price movements.

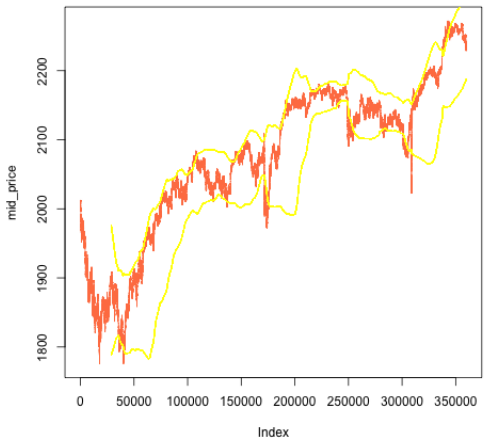
a. Exponential Moving Average

Weighted moving average (WMA) that gives more weighting, or importance, to recent price data than the simple moving average (SMA) does. The EMA responds more quickly to recent price changes than the SMA. In this paper, 5 day and 20 day backward window length was used. The golden cross was also calculated, which is a candlestick pattern that is a bullish signal in which a relatively short-term moving average crosses above a long-term moving average. Crossovers was also calculated. The most basic type of crossover is when the price of an asset moves from one side of a moving average and closes on the other.



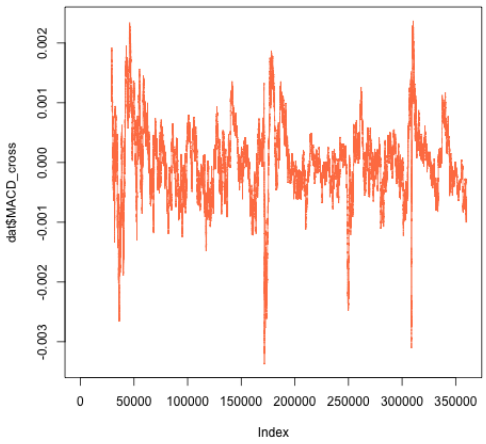
b. Bollinger Band

Bollinger Band is a technical analysis tool defined by a set of lines plotted two standard deviations (positively and negatively) away from a simple moving average (SMA). The squeeze was calculated too, which is when the bands come close together, constricting the moving average. This signals a period of low volatility and is considered by traders to be a potential sign of future increased volatility and possible trading opportunities.



c. Moving Average Convergence and Divergence (MACD)

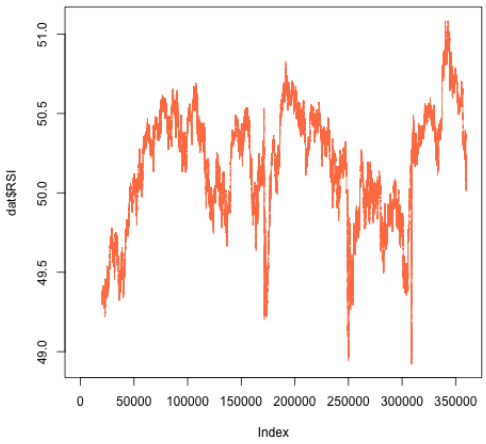
Moving Average Convergence Divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. The MACD is calculated by subtracting the 26-period Exponential Moving Average (EMA) from the 12-period EMA. A nine-day EMA of the MACD, called the "signal line," is then plotted on top of the MACD line, which can function as a trigger for buy and sell signals.



d. Relative Strength Index (RSI)

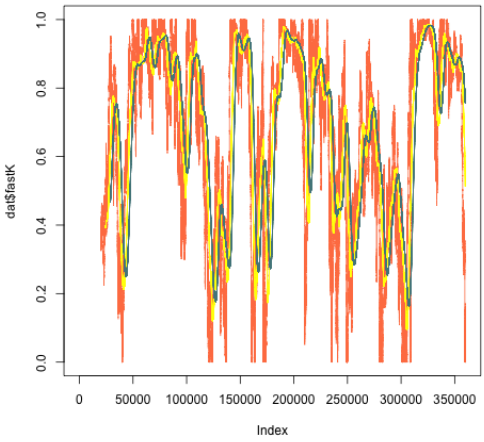
The relative strength index (RSI) is a momentum indicator that measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock or other asset.

$$\text{RSI}=100-\frac{100}{1+\frac{Averagegain}{Averageloss}}$$



e. Stochastic Oscillator

A stochastic oscillator is a momentum indicator comparing a particular price to a range of its prices over a certain period of time. The sensitivity of the oscillator to market movements is reducible by adjusting that time period or by taking a moving average of the result.

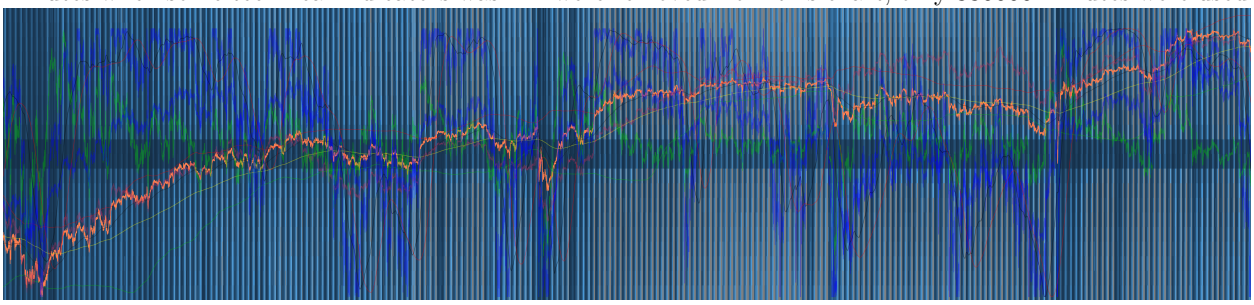


Correlated tickers

Based on the assumption that either positively correlated or negatively correlated tickers can help improve the prediction, the price from Nasdaq 100 E-Mini was also used as predictors. Nasdaq 100 E-Mini and E-Mini S&P 500 are 93.51263% correlated.

Futures Chart

How to incorprorate and use technical indicators was one of the most challenging part. One approach was just using them as numeric variables and the other was to plot these indicators in futures chart and use this image as data. Both approaches should be tested. The latter method might be better becuae it can deal with spacial data that the former method cannot detect. Futures chart of E-Mini S&P 500 is like this. First 29888 minutes when some technical indicators was NA were removed from this chart; only 330000 minutes were used.



The orange line in chart shows the mid_price of E-Mini S&P 500.

The scale of RSI, fastK, fastD, slowD, and mid_price of Nasdaq 100 E-Mini is completely different from mid_price of E-Mini S&P 500. So they were rescaled so that they could fit within one chart. Because volumes of futures might have some explanatory power, they were included in the chart as a background heatmap, where middle in y-axis shows the volume of "v1_buy", "v1_sell" and top part in y-axis shows the volume of "v10_buy" and down part in y-axis shows the volume of "v10_sell". This image had 15000 width, 3500 hight, 3channels. This means that each 1 width has 22 minutes (i.e. 330000/15000=22). In the following part, 15 width(= 330 minutes) was used as one image data.

Labeling

Rolling mean price of previous w (300 for models without image of future charts and 330 for models with future charts) minutes was used for previous price level and the one of future 300 minutes was used for future price level to detect price movements. If the price change percentages were more than threshold, the label was 1, if they were less than threshold, then the label was 0 and otherwise, the label was -1. The threshold was 0.0005 because this made the number of each label almost equal and this threshold was worthy enough for traders.

V. METHODOLOGY AND ANALYSIS

There were three goals in this section:

1. to see whether technical indicator really had an imapct on future price movement
2. to see whether using futures chart as an input was better way or not
3. to see whether machine learning model achived better than random guess, educated guess and majority guess

Four types of machine learning methods: ResNet, LSTM, CNN, and Random Forest were used to check above questions. In each case, I ued first 70% of the dataset as traing data, next 15% as validation dataset, and last 15% as test data. Notice that we can't use k-fold cross validation in time series becuase if we could use future price as training data, it would cause serious data leakage. So train, validation and test data were splited in the time series order.

Model 0 used only mid_price and volume as predictors. Model 1-3 used volumes of ES, technical indicator and mid_price of NQ as predictors but not futures chart image. Only the last model used futures chart image as as input to model.

Model 0 Random Forest + No technical indicator + No futures chart

To see whether technical indicators really had an impact on future price movement, the model without technical indicators was tested as a control. The parameters of this model was ntree=1000, mtry=5, nodesize=5 and samplesize=500

Model 1 Random Forest + Technical indicator + No futures chart

This model used technical indicators. The parameter of random forest was ntree=1000, mtry=6, nodesize=5 and samplesize=500.

Model 2 CNN + Technical indicator + No futures chart

This model still used technical indicator and mid_price of NQ. The model architecture is like the following table.

CNN + Technical indicator + No futures chart

		description of each layer
layer1	conv2d	filters=6, kernel size=(3,3), activation=relu
	max pooling	pool size=(2,2)
layer2	conv2d	filters=8, kernel size=(3,3), activation=relu
	max pooling	pool size=(2,2)
	drop out	drop out rate=0.5
layer3	dense layer	units=16, activation="relu", kernel l1 regularizer=0.001
	drop out	drop out rate=0.5
layer4	output layer	units=3, activation="softmax"

optimizer: rmsprop
batch size=120
epochs=20

Model 3 GRU + Technical indicator + No futures chart

RNN can handle sequence data. Financial data is sequential data and each data has memory of previous data. This is why RNN is thought to be a good tool for financial analysis. Following architecture was used in this report.

GRU

		description of each layer
layer1	separable conv1d	filters=20, kernel size=(9,9), activation=relu
layre2	gru	unit=32, reurn sequence=True
layer3	gru	unit=32
layer4	dense layr	units=16, activation="relu", kernel l1 regularizer=0.001
	drop out	drop out rate=0.5
layer5	output layer	units=3, activation="softmax"

optimizer: rmsprop
batch size=120
epochs=20

GRU and LSTM is one of the RNN methods. They were developed to solve vanishing gradient problem. GRU was used in this model.In addition to GRU, convolution network: separable conv1d was also used because it can turn the long input sequence into much shorter ones of higher-level features.

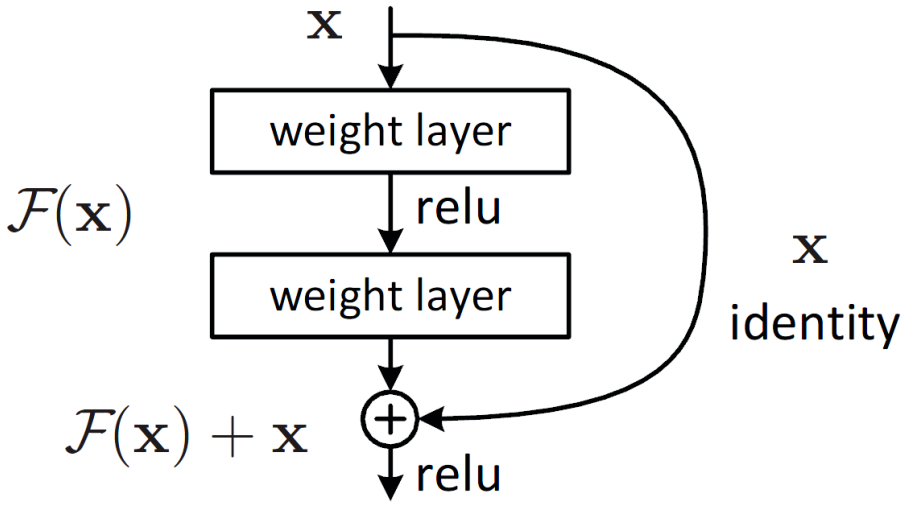
Model 4 ResNet + Technical indicator + Future chart

Finally the usefulness of futures chart was tested. The model architecture was like this table.

ResNet + technical indicator + futures chart

		description of each layer
layer1	conv2d	filters=16, kernel size=(8,1), activation=relu
	batch normalization	pool size=(2,2)
layer2	block1	number of input=16, number of output=16, number of basic block=6, stride=1
layer3	block2	number of input=16, number of output=32, number of basic block=6, stride=2
layer4	block3	number of input=32, number of output=32, number of basic block=6, stride=2
	average pool 2d	pool size=(2,2)
	drop out	drop out rate=0.5
layer5	dense layer	units=16, activation="relu", kernel l1 regularizer=0.001
	drop out	drop out rate=0.5
layer6	output layer	units=3, activation = "softmax"
optimizer: adam		
batch size=120		
epochs=20		

Deep convolution network suffers from vanishing gradient descent. Residual Network trys to mitigate this problem by introducing a so-called "identity shortcut connection" that skips one or more layers, as shown in the following figure.



This block is called basic block. Block1, Block2 and Block3 in the model architecture was composed by some basic blocks. Usually resnet outperforms CNN.

VI. Evaluations

5 statistical models were tested to get better accuracy for future price movement of E-Mini S&P 500. Accuracy was calculated by taking average of number of match of grand truth and predicted label.

	ACC
Mean random guess	33.33611
Mean educated guess	35.26678
Acc majority guess	40.52568
Random Forest without technical indicator	58.92055
Random Forest with technical indicator	62.5053
LSTM with technical indicator	40.99167
CNN with technial indicator	40.48333
alpha=0.0005	
w=300	

As you can see in the table, random forest with technical indicator performs best. Its accuracy was 62.5053. On the other hand, the accuracy of random forest without technical indicator was 58.92055. Conditions of both cases was all the same except technical indicator. That is to say, technical indicator had an influence on the price movement.

LSTM and CNN with technical indicator got almost the same accuracy as ACC majority guess and they outperformed mean random guess and mean educated guess.

	ACC
Mean random guess	33.32138
Mean educated guess	33.33605
Acc majority guess	37.53375
Futures Chart	38.4375
alpha=0.0005	
w=300	

How about ResNet with futures chart? Its accuracy was 38.4375% while the accuracy of mean random guess was 33.32138%, the one of mean educated guess was 33.33605% and the one of acc majority guess was 37.53375%. ResNet with futures chart outperformed these guess models but it was stil not as good as random forest.

VII. Conclusion

Question of analysis was:

1. to see whether technical indicator really had an imapct on future price movement
2. to see whether using futures chart as an input is better way or not

3. to see whether machine learning model achived better than mean random guess, mean educated guess and acc majority guess

The answer was:

1. technical indicator had a significantly big influence on price movement of futures as you can know from the VI. Evaluations part.
2. using futures chart was better way than mean random guess, mean educated guess and acc majority guess but it was not beter than random forest. I thought model using futures chart would be better because it might be able to catch some spacial data like the gap and cross of each variable. However, it did not go well maybe because of image resolution, color of each plot, few number of samples or misassumption.
3. in almost all case, machine learning model was better than mean random guess, mean educated guess and acc majority guess. Especially random forest performed much better.

In this project, the target return rate (e.g. 0.0005, 0.0008) and target window length (e.g. w=300,500) was not considered well. I tried a few other cases like target return rate = 0.0008, 0.001 but the 0.0005 performed better. The accuracy would increase if they are tested more by grid search.

Model using futures chart can be improved more. In this paper we could get only 15000 sample images at total for this model, while the other model had 359288 samples. The number of samples can be increased by using high resolution because the number of samples is the same with the width of image.

Appendix: Code

Data Loading

```
data_folder <- "algoseek_ES_M1"
data_folder_out <- file.path(data_folder)

data_range <- stringr::str_replace_all(as.character(as.Date(as.Date("2016-01-01"):as.Date("2016-12-30")
                      origin = "1970-01-01")), "-", "")
pre_roll_dates <- c("20160310","20160609","20160915","20161208")
roll_dates <- c("20160311","20160610","20160916","20161209")

files_out <- list.files(data_folder_out)
```

Futures rollover

```
####################
# Futures rollover #
####################


roll_gap_container = rep(0,length(files_out))

futures_rollover=function(pre_roll_date,roll_date){
  dat_roll <- read.csv( file.path( data_folder_out, paste0(roll_date, "_M1.csv")) )
  dat_roll_pre <- read.csv( file.path( data_folder_out, paste0(pre_roll_date, "_M1.csv")))
  gap_roll <- (dat_roll$vwap1_buy[1] + dat_roll$vwap1_sell[1])/2 -
    mean(c(dat_roll_pre$vwap1_buy[nx <- nrow(dat_roll_pre)], dat_roll_pre$vwap1_sell[nx]),na.rm = TRUE)
```

```
  files_out <- list.files(data_folder_out)
  inx_add_gap <- as.numeric(stringr::str_sub(files_out, 1, 8)) < as.numeric(roll_date)
  roll_gap_container[inx_add_gap]=roll_gap_container[inx_add_gap]+gap_roll
  return(roll_gap_container)
}

roll_gap_container=futures_rollover(pre_roll_dates[1],roll_dates[1])
roll_gap_container=futures_rollover(pre_roll_dates[2],roll_dates[2])
roll_gap_container=futures_rollover(pre_roll_dates[3],roll_dates[3])
roll_gap_container=futures_rollover(pre_roll_dates[4],roll_dates[4])

dat=NULL
for(k in 1:length(files_out)){
    tmp <- read.csv( file.path(data_folder_out, files_out[k]) )
    tmp[,c(5,6)] <- tmp[,c(5,6)] + roll_gap_container[k]
    dat <- rbind(dat, tmp)
}

dat_without_rollover=NULL
for(k in 1:length(files_out)){
  tmp <- read.csv( file.path(data_folder_out, files_out[k]) )
  dat_without_rollover <- rbind(dat_without_rollover, tmp)
}
dat_without_rollover$mid_price <- apply(dat_without_rollover[,c(5,6)],1,FUN = mean, na.rm=TRUE)
write.csv(dat_without_rollover,"dat_without_rollover")
write.csv(dat,"preprocessed_data.csv")
```

Mid price

```
raw_data=read.csv("preprocessed_data.csv")[,-1]
dat=raw_data
dat$mid_price <- apply(dat[,c(5,6)],1,FUN = mean, na.rm=TRUE)
library(stringr)
dat$month=as.numeric(str_replace(dat$date,pattern =
                                   "^[:digit:]{4}([:digit:]{2})[:digit:]{2}$",replacement = "\\1"))
dat$day=as.numeric(str_replace(dat$date,pattern =
                                 "^[:digit:]{4}[:digit:]{2}([:digit:]{2})$",replacement = "\\1"))
```

Technical indicator

```
#Exponential moving average
library(pracma)
dat$movavg20=c(rep(NA,28800),movavg(dat$mid_price, 28800,
                                    type="e")[-c(1:28800)])
dat$movavg5=c(rep(NA,7200),movavg(dat$mid_price, 7200,
                                  type="e")[-c(1:7200)])
dat$cross=dat$movavg5-dat$movavg20
dat$diff_from_movavg20=(dat$mid_price-dat$movavg20)/dat$movavg20
dat$diff_from_movavg5=dat$mid_price-dat$movavg5/dat$movavg5
library(TTR)
#Bollinger Band
```

```
bbands=BBands(dat$mid_price,n=28800)
dat$pctB=bbands[,"pctB"]
dat$up=bbands[,"up"]
dat$down=bbands[,"dn"]
dat$squeeze=bbands[,"up"]-bbands[,"dn"]
#Moving Average Convergence and Divergence
MACD=MACD(dat$mid_price,nFast=7200, nSlow=28800)
dat$MACD_cross=MACD[,"macd"]-MACD[,"signal"]
#RSI
dat$RSI=RSI(dat$mid_price,n=20160)
#Stochastic Oscillator
stoch=stoch(dat$mid_price, nFastK=20160, nFastD=4320, nSlowD=4320)
dat$fastK=stoch[,"fastK"]
dat$fastD=stoch[,"fastD"]
dat$slowD=stoch[,"slowD"]
dat_NQ=read.csv("NQpreprocessed_data.csv")[,-1]
dat_NQ$mid_price_NQ <- apply(dat_NQ[,c(5,6)],1,FUN = mean, na.rm=TRUE)
dat=dplyr::left_join(dat,dat_NQ[,c(2,3,4,27)],by=c("date","h","m"))
write.csv(dat,"dat.csv")
```

Rescale for Futures chart

```
dat=read.csv("dat.csv")[,-1]
library(scales)
library(reshape2)
dat_for_chart=dat[29888:nrow(dat),]
dat_for_chart$index=1:nrow(dat_for_chart)
scale.range=range(dat_for_chart$mid_price)
dat_for_chart$MACD_cross=rescale(dat_for_chart$MACD_cross,to = scale.range)
dat_for_chart$RSI=rescale(dat_for_chart$RSI,to = scale.range)
dat_for_chart$fastK=rescale(dat_for_chart$fastK,to = scale.range)
dat_for_chart$fastD=rescale(dat_for_chart$fastD,to = scale.range)
dat_for_chart$slowD=rescale(dat_for_chart$slowD,to = scale.range)
dat_for_chart$mid_price_NQ=rescale(dat_for_chart$mid_price_NQ,to=scale.range)
```

Change the volume to long data for heatmap

```
#Volume
dat_1=dat_for_chart[,7:26]
rownames(dat_1)=dat_for_chart$index
library(tidyverse)
dat2 <- dat_1 %>%
    tbl_df()%>%
    rownames_to_column('Var1') %>%
    gather(Var2, value, -Var1)
dat2$Var2=factor(dat2$Var2,levels=
                   c("v10_buy",  "v9_buy", "v8_buy", "v7_buy", "v6_buy", "v5_buy",
                     "v4_buy", "v3_buy", "v2_buy", "v1_buy", "v1_sell",  "v2_sell",
                     "v3_sell", "v4_sell",  "v5_sell",  "v6_sell",  "v7_sell",
                     "v8_sell",  "v9_sell",  "v10_sell"))
```

```
levels(dat2$Var2)=as.character(seq(min(dat$down),max(dat$up),length.out = 20))
dat2$Var2=as.numeric(as.character(dat2$Var2))
dat2$Var1=as.numeric(dat2$Var1)
write.csv(dat2,"dat_volume.csv")
```

Futures chart

```
library(cowplot)
dat_sub=dat_for_chart
dat2_sub=dat2
library(ggplot2)
gplot=ggplot(NULL)
gplot=gplot+geom_tile(data=dat2_sub, aes(Var1, Var2,fill = value)) +
  scale_fill_gradient(limits=c(0, 800))+
  theme_nothing() +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0))
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=mid_price),
                      color="coral")
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=movavg20),
                      color="yellow", alpha=0.4)
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=movavg5),
                      color="gray9", alpha=0.4)
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=up),
                      color="red", alpha=0.4)
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y = down),
                      color="green3", alpha=0.4)
gplot=gplot+geom_line(data= dat_sub, aes(x=index,y=mid_price_NQ),
                      col="maroon", alpha=0.4)
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=MACD_cross),
                      color=619,alpha=0.4)
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=RSI),
                      color=380,alpha=0.4)
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=fastK),
                      color=452,alpha=0.4)
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=fastD),
                      color=657,alpha=0.4)
gplot=gplot+geom_line(data = dat_sub,aes(x=index,y=slowD),
                      color=266,alpha=0.4)
ggsave("gplot.png",dpi=500,width = 30, height = 7)
```

```
library(png)
img=readPNG("gplot.png")
```

Case 1 Random Forest

Labeling

```
dat=read.csv("dat.csv")[,-1]
w <- 300
```

```
avgMprice <- c(zoo::rollmean(dat$mid_price, k=w, align="left"))

dat <- dat[-c(((nrow(dat)-w+1):nrow(dat))), ] # remove last w observations
dat$preMP <- avgMprice[1:nrow(dat)]
dat$postMP <- avgMprice[(w+1):(nrow(dat)+w)]
dat <- dat[-(1:(w-1)),] # remove first (w-1) observations

## a: threshold of price change percentages for labeling the direction
a <- 0.0008
chg <- dat$postMP / dat$preMP - 1

## direction of price movement
dat$direction <- -1 # stable, excluded label
dat$direction[chg > a] <- 1 # increase
dat$direction[chg < -a] <- 0 # decrease
table(dat$direction)
```

Train, val, test split

```
# train | validation | test splits: 3|1|1
dat=dat[,-c(1,2)]

data_train <- dat[(1:floor(nrow(dat)/10*7)),][,1:44]
data_val <- dat[((floor(nrow(dat)/10*1.5)+1):floor(nrow(dat)/5*4)),][,1:44]
data_test <- dat[((floor(nrow(dat)/10*1.5)+1):nrow(dat)),][,1:44]
#Model 0
data_train=train[,1:29]
data_val=val[,1:29]
data_test=test[,1:29]
Y_data_train = train[,30]
Y_data_val = val[,30]
Y_data_test =test[,30]

#Model 1
#data_train=train[,1:44]
#data_val=val[,1:44]
#data_test=test[,1:44]
#Y_data_train = train[,45]
#Y_data_val = val[,45]
#Y_data_test =test[,45]
```

Rescale

```
#col_volume <- (1:81)
nCol=ncol(data_train)
me_train <- apply(as.matrix(data_train), 2, mean, na.rm=TRUE)
sd_train <- apply(as.matrix(data_train), 2, sd, na.rm=TRUE)
# rescale train data
for(i in 1:nCol) data_train[,i] <- scale(data_train[,i],
                                         center = me_train[i], scale = sd_train[i])
```

```
# rescale validation data (using train mean and sd)
for(i in 1:nCol) data_val[,i] <- scale(data_val[,i],
                                       center = me_train[i], scale = sd_train[i])

# rescale test data (using train mean and sd)
for(i in 1:nCol) data_test[,i] <- scale(data_test[,i],
                                        center = me_train[i], scale = sd_train[i])
```

Random Forest

```
library(randomForest)
fit=randomForest(as.matrix(data_train),as.factor(Y_data_train),
                 ntree = 1000, mtry = 6, nodesize = 5, sampsize = 500)
pre <- predict(fit, newdata = data_test)
acc <- mean(Y_data_test==pre)
acc
```

Case 2 LSTM

Data generator

```
sampling_generator <- function(X_data, Y_data, batch_size, w=300)
{
  function()
  {
    index<- w:dim(X_data)[1]

    rows <- sample( index, batch_size, replace = TRUE )

    Y <- X <- NULL
    Xlist <- list()
    for(i in rows)
    {
      Xlist[[i]] <- X_data[(i-w+1):i,]
      Y <- c(Y, Y_data[i])
    }
    X <- array(abind::abind(Xlist, along = 0), c(batch_size, w, nCol))
    list(X, to_categorical(Y,num_classes = 3))
  }
}
```

LSTM

```
library(keras)
k_clear_session()

# we have to use the call back function to store the best model that
#is to be used for evaludation of test set
checkPoint <- callback_model_checkpoint(filepath = file.path("ES-LSTM"),
```

```
                                        monitor = "val_acc", save_best_only = TRUE)
reduceLr <- callback_reduce_lr_on_plateau(monitor = "val_acc",
                                          factor = 0.1, patience = 3)
logger <- callback_csv_logger(file.path("logger.csv"))

model <- keras_model_sequential() %>%
  layer_separable_conv_1d(filters = 20, kernel_size = 9, activation = "relu",
                          input_shape = list(NULL, 20)) %>%
  layer_cudnn_gru(unit=32, return_sequences = T) %>%
  layer_cudnn_gru(unit=32) %>%
  layer_dense(units = 16, activation = "relu", kernel_regularizer =
                regularizer_l1(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 3, activation = "sigmoid")

model %>% compile(
loss = "categorical_crossentropy",
optimizer = optimizer_rmsprop(lr = 1e-4),
metrics = c("acc")
)
batch_size <- 120
```

Run the model

```
his <- model %>%
  fit_generator(sampling_generator(data_train, Y_data_train, batch_size = batch_size),
                                epochs = 14,
                                steps_per_epoch = 100,
                                callbacks = list(checkPoint, reduceLr, logger),
                                validation_data =
                  sampling_generator(data_val, Y_data_val, batch_size = batch_size),
                                validation_steps = 100)
plot(his)
```

Evaluation

```
fitted <- load_model_hdf5(file.path("ES-LSTM"))
results <- fitted %>%
evaluate_generator(
  sampling_generator(data_test, Y_data_test, batch_size =batch_size, w=w),
  steps = floor(length(Y_data_test)/batch_size))
tb_Y_train=table(Y_data_train)
tb_Y_test=table(Y_data_test)
library(fmlr)
fmlr::acc_lucky(as.vector(tb_Y_train), as.vector(tb_Y_test), results$acc)
```

Case3 CNN

Data generator

```
sampling_generator <- function(X_data, Y_data, batch_size, w=300)
{
  function()
  {
    index<- w:dim(X_data)[1]

    rows <- sample( index, batch_size, replace = TRUE )

    Y <- X <- NULL
    Xlist <- list()
    for(i in rows)
    {
      Xlist[[i]] <- X_data[(i-w+1):i,]
      Y <- c(Y, Y_data[i])
    }
    X <- array(abind::abind(Xlist, along = 0), c(batch_size, w, nCol, 1))
    list(X, to_categorical(Y,num_classes = 3))
  }
}
```

CNN

```
library(keras)
k_clear_session()

# we have to use the call back function to store the best model that is to be used for evaludation of t
checkPoint <- callback_model_checkpoint(filepath = file.path("ES-LSTM"),
                                        monitor = "val_acc", save_best_only = TRUE)
reduceLr <- callback_reduce_lr_on_plateau(monitor = "val_acc",
                                          factor = 0.1, patience = 3)
logger <- callback_csv_logger(file.path("logger.csv"))

model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 6, kernel_size = c(3, 3),
                activation = "relu", input_shape = c(w, nCol, 1)) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 8, kernel_size = c(3, 3), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_flatten() %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 16, activation = "relu",
              kernel_regularizer = regularizer_l1(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 3, activation = "softmax")

model %>% compile(
loss = "categorical_crossentropy",
optimizer = optimizer_rmsprop(lr = 1e-4),
```

```
metrics = c("acc")
)
batch_size <- 120
```

Run the model

```
his <- model %>%
  fit_generator(sampling_generator(data_train, Y_data_train, batch_size = batch_size),
                                epochs = 20,
                                steps_per_epoch = 100,
                                callbacks = list(checkPoint, reduceLr, logger),
                                validation_data =
                  sampling_generator(data_val, Y_data_val, batch_size = batch_size),
                                validation_steps = 100)
plot(his)
```

```
fitted <- load_model_hdf5(file.path("ES-LSTM"))
results <- fitted %>%
evaluate_generator(
  sampling_generator(data_test, Y_data_test, batch_size =batch_size, w=w),
  steps = 100)
tb_Y_train=table(Y_data_train)
tb_Y_test=table(Y_data_test)
fmlr::acc_lucky(as.vector(tb_Y_train), as.vector(tb_Y_test), results$acc)
```

Case 4 Futures chart

Labeling

```
#####################################################
# use SPY next minute price direction as labels #
#####################################################

w <- 330
preMprice <- c(zoo::rollmean(dat_for_chart$mid_price, k=w, align="left"))
postMprice <- c(zoo::rollmean(dat_for_chart$mid_price, k=300, align="left"))
preMP <- preMprice[w:(nrow(dat_for_chart)-300)]
postMP = postMprice[(w+300):nrow(dat_for_chart)]
preMP <- preMP[(1:(nrow(dat_for_chart)-w-300+1))%%22==1]
postMP = postMP[(1:(nrow(dat_for_chart)-w-300+1))%%22==1]

## a: threshold of price change percentages for labeling the direction
a <- 0.0005
chg <- postMP/preMP - 1
## direction of price movement
Y <- rep(-1,length(chg)) # stable, excluded label
Y[chg > a] <- 1 # increase
Y[chg < -a] <- 0 # decrease
table(Y)
```

Train, val, test split

```
data_train=img[,1:10500,]
data_val=img[,10501:12750,]
data_test=img[,12751:14972,]
Y_train=Y[1:10500]
Y_val=Y[10501:12750]
Y_test=Y[12751:14972]
```

Data generator

```
sampling_generator <- function(X_data, Y_data, batch_size, w=30)
{
  function()
  {
    index<- w:dim(X_data)[2]

    rows <- sample( index, batch_size, replace = TRUE )

    Y <- X <- NULL
    Xlist <- list()
    for(i in rows)
    {
      Xlist[[i]] <- X_data[,(i-w+1):i,]
      Y <- c(Y, Y_data[i])
    }
    X <- array(abind::abind(Xlist, along = 0), c(batch_size, 3500, w, 3))
    # add one axis of dimension of 1
    list(X, to_categorical(Y,num_classes = 3))
  }
}
```

Create CNN for Futures Chart model

```
library(keras)
k_clear_session()

# we have to use the call back function to store the best model
#that is to be used for evaludation of test set
checkPoint <- callback_model_checkpoint(filepath =
              file.path("ES-Chart"),monitor = "val_acc", save_best_only = TRUE)
reduceLr <- callback_reduce_lr_on_plateau(monitor = "val_acc",
                                          factor = 0.1, patience = 3)
logger <- callback_csv_logger(file.path("logger.csv"))

model <- keras_model_sequential() %>%
  layer_conv_2d(filters = 8, kernel_size = c(8, 1),
                activation = "relu", input_shape = c(3500, 30, 3)) %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  layer_conv_2d(filters = 8, kernel_size = c(8, 1), activation = "relu") %>%
  layer_max_pooling_2d(pool_size = c(2, 2)) %>%
```

```
  layer_flatten() %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 16, activation = "relu",
              kernel_regularizer = regularizer_l1(0.001)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 3, activation = "softmax")

model %>% compile(
loss = "categorical_crossentropy",
optimizer = optimizer_adam(lr = 1e-4),
metrics = c("acc")
)
batch_size <- 120
```

Run the model

```
his <- model %>%
  fit_generator(sampling_generator(data_train, Y_train, batch_size = batch_size),
                                epochs = 15,
                                steps_per_epoch = 50,
                                callbacks = list(checkPoint, reduceLr, logger),
                                validation_data =
                  sampling_generator(data_val, Y_val, batch_size = batch_size),
                                validation_steps = 50)
plot(his)
```

Evaluation

```
fitted <- load_model_hdf5(file.path("ES-Chart"))
results <- fitted %>%
evaluate_generator(
  sampling_generator(data_test, Y_test, batch_size =batch_size, w=30),
  steps = 100)
tb_Y_train=table(Y_train)
tb_Y_test=table(Y_test)
fmlr::acc_lucky(as.vector(tb_Y_train), as.vector(tb_Y_test), results$acc)
```

ResNet

```
library(dplyr) # For the %>% operator

# Functions to build the model
k_clear_session()
layer_shortcut <- function(object, ninput, noutput, stride) {
    if(ninput == noutput)
        object %>%
            layer_lambda(function(x) x)
    else{
        object <- object %>%
```

```
            layer_average_pooling_2d(1, stride)

        a <- object %>%
            layer_lambda(function(x) x)

        b <- object %>%
            layer_lambda(., function(x) k_zeros_like(x))

        layer_concatenate(c(a, b))
    }
}

layer_basic_block <- function(object, ninput, noutput, stride) {
    a <- object %>%
        layer_conv_2d(noutput, c(8,1), stride, 'same',
                      kernel_initializer = 'lecun_normal') %>%
        layer_batch_normalization() %>%
        layer_activation('relu') %>%
        layer_conv_2d(noutput, c(8,1), 1, 'same',
                      kernel_initializer = 'lecun_normal') %>%
        layer_batch_normalization()

    b <- object %>%
        layer_shortcut(ninput, noutput, stride)

    layer_add(c(a, b))
}


build_block <- function(object, ninput, noutput, count, stride) {
    for(i in 1:count)
        object <- object %>%
            layer_basic_block(if(i == 1) ninput else noutput,
                              noutput,
                              if(i == 1) stride else 1
                              )
    object
}

build_resnet <- function(depth = 26) {
    n <- (depth - 2) / 6

    input <- layer_input(shape=c(3500,30,3))

    output <- input %>%
        layer_conv_2d(16, c(8,1), 1, 'same',
                      kernel_initializer = 'lecun_normal') %>%
        layer_batch_normalization() %>%
        layer_activation('relu') %>%
        build_block(16, 16, n, 1) %>% # Primer conjunto
        build_block(16, 32, n, 2) %>% # Segundo conjunto
        build_block(32, 64, n, 2) %>% # Tercer conjunto
        layer_average_pooling_2d(2, 2) %>%
```

```
        layer_flatten() %>%
      layer_dropout(rate = 0.5) %>%
    layer_dense(units = 16, activation = "relu",
                kernel_regularizer = regularizer_l1(0.001)) %>%
    layer_dropout(rate = 0.5) %>%
        layer_dense(3, activation = "softmax")

    keras_model(input, output)
}
```

```
model <- build_resnet(20)
# Compiling and training the model
model %>% compile(
loss = "categorical_crossentropy",
optimizer = optimizer_adam(lr = 1e-4),
metrics = c("acc")
)
batch_size <- 16
##################
# run the model #
##################
# This part takes time, so I only run it before kniting the Rmarkdown,
# and then save the results and the plots for later use.
reduceLr <- callback_reduce_lr_on_plateau(monitor = "val_acc", factor = 0.1, patience = 3)

his <- model %>%
  fit_generator(sampling_generator(data_train, Y_train, batch_size = batch_size),
                                epochs = 20,
                                steps_per_epoch = 50,
                                callbacks = list(reduceLr),
                                validation_data =
                  sampling_generator(data_val, Y_val, batch_size = batch_size),
                                validation_steps = 50)
plot(his)
```

```
results <- model %>%
evaluate_generator(
  sampling_generator(data_test, Y_test, batch_size =batch_size, w=30),
  steps = 100)
results
tb_Y_train=table(Y_train)
tb_Y_test=table(Y_test)
fmlr::acc_lucky(as.vector(tb_Y_train), as.vector(tb_Y_test), results$acc)
```