Algorithmic Trading

MAJOR PROJECT REPORT

Master of Technology

IN

Information Technology

BY

Shubham Pandey (MIT2020050)

UNDER THE SUPERVISION OF

Profesor Satish Singh

IIIT-ALLAHABAD

***Abstract*—The financial markets runs on information.** *As new information technology was developed, the exchanges quickly adopted them, eager to take advantage of the opportunities provided. The new technology allowed for faster communication, greater accessibility to the markets and the new infrastructure helped decreasing the cost of doing business. Today, a trader, with many accounts to manage with limited time as the information available for analysis is abundant, if not overwhelming. The new features of the electronic exchange (API), mainly on the order book, allows the use of algorithms that can be useful for automating some trading processes.* **This algorithmic trading will improve efficacy of trader to analyse the market. And take discission on it.**

In this report, we provide algorithmic functionality to the Zerodha trading application. We analyzed the requirements and issues of three market-impact algorithms (Candlestick pattern, RSI, and significance) and designed a prototype which we integrated in the Zerodha application. During the development, of the combined(all three algorithms) algorithm, we tested several models for market prediction, obtaining the best results. We further develop our prototype including these findings.

Keywords— Trading , Algorithmic trading, API , BSE, NSE.

I. INTRODUCTION

Algorithmic trading is the well-defined set of rules for the computer programmed to follow a defined set of instructions for placing a trade. The defined set of rules are based on timing, quantity (number of shares), prices (cost of one share) or any mathematical model. It is also known as Quantitative trading, High frequency trading , Automatic Trading and Black- Box trading.

1) We have use the Algorithmic trading because it rules out the human emotion on trading activites.

2) Algorithmic trading is more faster compare to manual trading

3) Algorithmic trading allows us to trade and analyse the multiple stock simultaneously.

4) Algorithmic trading reduce the manual errors in placing order.

II. EXPERIMENTAL DATASET

I have used Kite Zerodha Historical data for this project.

Dataset

III. METHODOLOGY

Pre-trade analysis – We will analysis properties of stocks using of market data or financial news. And we will create a list of stock on the basis of previous day movement of the stock. We preferred the stock which shows more divergence on the previous day. As this stock is more likely to move.

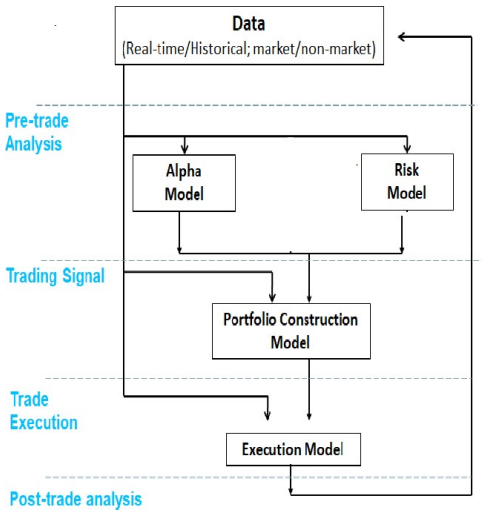

Figure 1: Algorithmic trading system components and their relations

Trading signal – After creating the list we will generate trading signal based on the algorithm that has been pre-determined.
Algorithm for generating trading signals –

1) Significance – It will tells us weather the price of stock is near the support or resistance line as the price of stock moving around this line is more volatile.
2) Pattern – It will tell us the candle stick pattern formed by the stocks . as this pattern are very important to decide the prediction of the stock movement. Pattern used in this project doji bullish , doji bearish , maru bozu bullish , maru bozu bearish , hanging man bearish , hammer bullish , shooting star bearish , harami cross bearish , harami cross bullish , engulfing bearish , engulfing bullish.
3) RSI Value – RSI stand for Relative strength index. If the RSI value greater than 50 and less than 70 indicates stock is bullish and RSI value greater than 70 indicates the stock is overbought and it is more likely to fall. RSI value below 50 and greater than 30 indicates stock is bearish and likely to fall. And RSI value below 30 indicates the stock is oversold and is more likely to rise

$$RSI_{\text{step one}} = 100 - \left[\frac{100}{1+\frac{\text{Average gain}}{\text{Average loss}}}\right]$$

The average gain or loss used in the calculation is the average percentage gain or loss during a look-back period. The formula uses a positive value for the average loss.

$$RSI_{\text{step two}} = 100 - \left[\frac{100}{1+\frac{(\text{Previous Average Gain}\times 13)+\text{Current Gain}}{-((\text{Previous Average Loss}\times 13)+\text{Current Loss})}}\right]$$

.

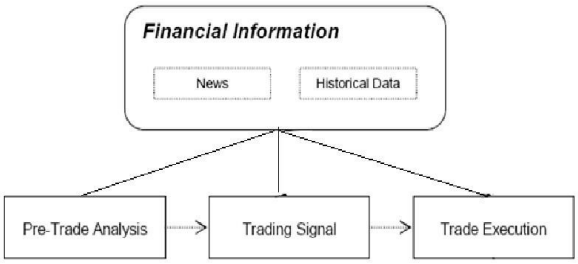

Figure 2 – Trade flow chart

Trade execution – executing orders for the selected stock depends on these parameter

a) Significance
b) Pattern
c) RSI value

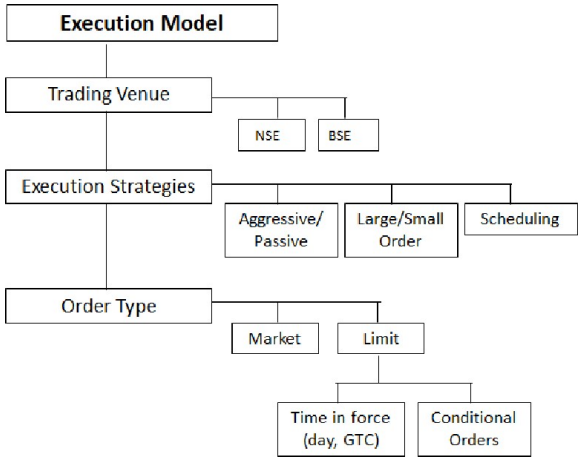

Figure 3 Execution Model

If Significance is high and it detect any candlestick pattern (Bullish) and RSI value is greater than 50 it will automatically execute buy order from NSE .

If Significance is high and it detect any candlestick pattern (Bearish) and RSI value is below 50 it will automatically execute sell order from NSE.

Order placement

In India there are two trading venue available to trade on stock BSE and NSE.

There is different order type –
Market Order – Order will be placed on the market price.

Limit Order – we restrict the price of stock if it hits the price then only it will execute the order. Otherwise, it wont.

IV. Results and discussion

I have performed my model in the Live Market to analyse the efficacy and behavior of my model. It shows a promising result 50-65% efficacy.

The experiment is done on only 6 trading days. I would like to perform this trading model in the live market for the period of5-6 months to come to any conclusion.

For performing this model, I need Kite Zerodha API and Kite Zerodha Historic API. Zerodha API will be used for executing my order in Zerodha trading platform. Where as Zerodha Historic API will be used to analyse the market based on historic data. This will cost 4000 Rs per month apart from that I need base amount for trading approx. 4000Rs per month. I request Prof Satish Singh to allocate 50,000 Rs to perform this experiment.

V. Conclusion and Future Scope

I studied the financial market (specifically stock trading) in order to obtain a better understanding of the problems associated with algorithmic trading. I then design the general architecture/flow of trading algorithms and study the most commonly used, trading methods. I study some possible model for order placement execution. Finally, I implement the algorithms and integrate them on the Kite Zerodha application, I develop the model for candlestick pattern recognition.

I used the market data for quick simulation and testing on real time.
I would like to implement this model for long period (5-6 months) to analyse the efficacy of my model. So that our model is prominent for using in real life scenario.

ACKNOWLEDGMENT

I would like to thanks Prof Satish Singh for guiding me through out this project. His valuable suggestion helps me to improve this project.

I would like to thanks Kite Zerodha for providing platform.

REFERENCES

[1] Udemy Jyoti Bansal (Jan ,2021) Technical Analysis Master class: Trading by Technical Analyss.
https://www.udemy.com/course/technical_analysis/

[2] Udemy Mohsen Hassan, bloom team (Feb , 2021) The complete fundamental Stock trading course.
https://www.udemy.com/course/foundation-course/

[3] Mario Andre Ferreira Pinto (2014) Design and implementation of an algorithmic trading system for the Sifox application

[4] Udemy Jose Portilla (feb ,2021) Python for financial analysis and Algorithmic Trading
https://www.udemy.com/course/python-for-finance-and-trading-algorithms/

APPENDIX

#Autologin

```
from kiteconnect import KiteConnect
from selenium import webdriver
import time
import os
cwd = os.chdir("D:\\AlgorithmicTrading")



def autologin():
    token_path = "api_key.txt"
    key_secret = open(token_path,'r').read().split()
    kite = KiteConnect(api_key=key_secret[0])
    service = webdriver.chrome.service.Service('./chromedriver')
    service.start()
    options = webdriver.ChromeOptions()


    options = options.to_capabilities()
    driver = webdriver.Remote(service.service_url, options)


    driver.get(kite.login_url())
    driver.implicitly_wait(10)


    username =
driver.find_element_by_xpath('/html/body/div[1]/div/div[2]/div[1]/div/div/div[2]/form/div[1]/in
put')
    password =
driver.find_element_by_xpath('/html/body/div[1]/div/div[2]/div[1]/div/div/div[2]/form/div[2]/in
put')
```

```
    username.send_keys(key_secret[2])
    password.send_keys(key_secret[3])

driver.find_element_by_xpath('/html/body/div[1]/div/div[2]/div[1]/div/div/div[2]/form/div[4]/button').click()

    pin = driver.find_element_by_xpath('/html/body/div[1]/div/div[2]/div[1]/div/div/div[2]/form/div[2]/div/input')
    pin.send_keys(key_secret[4])

driver.find_element_by_xpath('/html/body/div[1]/div/div[2]/div[1]/div/div/div[2]/form/div[3]/button').click()
    time.sleep(10)

    request_token = driver.current_url.split('=')[1].split('&action')[0]
    with open('request_token.txt','w') as the_file:
        the_file.write(request_token)
    driver.quit()

autologin()
#Generating the access token-- Valid till 6am next morni
request_token = open('request_token.txt','r').read()
key_secret = open("api_key.txt",'r').read().split()
kite = KiteConnect(api_key=key_secret[0])
data = kite.generate_session(request_token, api_secret=key_secret[1])
with open('access_token.txt','w') as file:
    file.write(data["access_token"])
```

#STOCK SCANNER AND TRADE EXECUTE

```
from kiteconnect import KiteConnect
import pandas as pd
import datetime as dt
import os
import time
import numpy as np
import requests

cwd = os.chdir("D:\\AlgorithmicTrading")

#generate trading session
access_token = open("access_token.txt",'r').read()
key_secret = open("api_key.txt",'r').read().split()
kite = KiteConnect(api_key=key_secret[0])
kite.set_access_token(access_token)

#get dump of all NSE instruments
instrument_dump = kite.instruments("NSE")
instrument_df = pd.DataFrame(instrument_dump)

def instrumentLookup(instrument_df,symbol):
    """Looks up instrument token for a given script from instrument dump"""
    try:
        return instrument_df[instrument_df.tradingsymbol==symbol].instrument_token.values[0]
    except:
        return -1
```

```
def fetchOHLC(ticker,interval,duration):
    """extracts historical data and outputs in the form of dataframe"""
    instrument = instrumentLookup(instrument_df,ticker)
    data = pd.DataFrame(kite.historical_data(instrument,dt.date.today()-dt.timedelta(duration),
dt.date.today(),interval))
    data.set_index("date",inplace=True)
    return data


def doji(ohlc_df):
    """returns dataframe with doji candle column"""
    df = ohlc_df.copy()
    avg_candle_size = abs(df["close"] - df["open"]).median()
    df["doji"] = abs(df["close"] - df["open"]) <= (0.05 * avg_candle_size)
    return df


def maru_bozu(ohlc_df):
    """returns dataframe with maru bozu candle column"""
    df = ohlc_df.copy()
    avg_candle_size = abs(df["close"] - df["open"]).median()
    df["h-c"] = df["high"]-df["close"]
    df["l-o"] = df["low"]-df["open"]
    df["h-o"] = df["high"]-df["open"]
    df["l-c"] = df["low"]-df["close"]
    df["maru_bozu"] = np.where((df["close"] - df["open"] > 2*avg_candle_size) & \
                      (df[["h-c","l-o"]].max(axis=1) <
0.005*avg_candle_size),"maru_bozu_green",
                      np.where((df["open"] - df["close"] > 2*avg_candle_size) & \
                      (abs(df[["h-o","l-c"]]).max(axis=1) <
0.005*avg_candle_size),"maru_bozu_red",False))
```

```
    df.drop(["h-c","l-o","h-o","l-c"],axis=1,inplace=True)
    return df


def hammer(ohlc_df):
    """returns dataframe with hammer candle column"""
    df = ohlc_df.copy()
    df["hammer"] = (((df["high"] - df["low"])>3*(df["open"] - df["close"])) & \
                    ((df["close"] - df["low"])/(.001 + df["high"] - df["low"]) > 0.6) & \
                    ((df["open"] - df["low"])/(.001 + df["high"] - df["low"]) > 0.6)) & \
                    (abs(df["close"] - df["open"]) > 0.1* (df["high"] - df["low"]))
    return df



def shooting_star(ohlc_df):
    """returns dataframe with shooting star candle column"""
    df = ohlc_df.copy()
    df["sstar"] = (((df["high"] - df["low"])>3*(df["open"] - df["close"])) & \
                    ((df["high"] - df["close"])/(.001 + df["high"] - df["low"]) > 0.6) & \
                    ((df["high"] - df["open"])/(.001 + df["high"] - df["low"]) > 0.6)) & \
                    (abs(df["close"] - df["open"]) > 0.1* (df["high"] - df["low"]))
    return df


def levels(ohlc_day):
    """returns pivot point and support/resistance levels"""
    high = round(ohlc_day["high"][-1],2)
    low = round(ohlc_day["low"][-1],2)
    close = round(ohlc_day["close"][-1],2)
    pivot = round((high + low + close)/3,2)
```

```
    r1 = round((2*pivot - low),2)
    r2 = round((pivot + (high - low)),2)
    r3 = round((high + 2*(pivot - low)),2)
    s1 = round((2*pivot - high),2)
    s2 = round((pivot - (high - low)),2)
    s3 = round((low - 2*(high - pivot)),2)
    return (pivot,r1,r2,r3,s1,s2,s3)

def trend(ohlc_df,n):
    "function to assess the trend by analyzing each candle"
    df = ohlc_df.copy()
    df["up"] = np.where(df["low"]>=df["low"].shift(1),1,0)
    df["dn"] = np.where(df["high"]<=df["high"].shift(1),1,0)
    if df["close"][-1] > df["open"][-1]:
        if df["up"][-1*n:].sum() >= 0.7*n:
            return "uptrend"
    elif df["open"][-1] > df["close"][-1]:
        if df["dn"][-1*n:].sum() >= 0.7*n:
            return "downtrend"
    else:
        return None

def res_sup(ohlc_df,ohlc_day):
    """calculates closest resistance and support levels for a given candle"""
    level = ((ohlc_df["close"][-1] + ohlc_df["open"][-1])/2 + (ohlc_df["high"][-1] +
ohlc_df["low"][-1])/2)/2
    p,r1,r2,r3,s1,s2,s3 = levels(ohlc_day)
    l_r1=level-r1
    l_r2=level-r2
```

```
    l_r3=level-r3
    l_p=level-p
    l_s1=level-s1
    l_s2=level-s2
    l_s3=level-s3
    lev_ser =
pd.Series([l_p,l_r1,l_r2,l_r3,l_s1,l_s2,l_s3],index=["p","r1","r2","r3","s1","s2","s3"])
    sup = lev_ser[lev_ser>0].idxmin()
    res = lev_ser[lev_ser<0].idxmax()
    return (eval('{}'.format(res)), eval('{}'.format(sup)))


def candle_type(ohlc_df):
    """returns the candle type of the last candle of an OHLC DF"""
    candle = None
    if doji(ohlc_df)["doji"][-1] == True:
        candle = "doji"
    if maru_bozu(ohlc_df)["maru_bozu"][-1] == "maru_bozu_green":
        candle = "maru_bozu_green"
    if maru_bozu(ohlc_df)["maru_bozu"][-1] == "maru_bozu_red":
        candle = "maru_bozu_red"
    if shooting_star(ohlc_df)["sstar"][-1] == True:
        candle = "shooting_star"
    if hammer(ohlc_df)["hammer"][-1] == True:
        candle = "hammer"
    return candle



def rsi(df, n):
    "function to calculate RSI"
```

```
    delta = df["close"].diff().dropna()
    u = delta * 0
    d = u.copy()
    u[delta > 0] = delta[delta > 0]
    d[delta < 0] = -delta[delta < 0]
    u[u.index[n-1]] = np.mean( u[:n]) # first value is average of gains
    u = u.drop(u.index[:(n-1)])
    d[d.index[n-1]] = np.mean( d[:n]) # first value is average of losses
    d = d.drop(d.index[:(n-1)])
    rs = u.ewm(com=n,min_periods=n).mean()/d.ewm(com=n,min_periods=n).mean()
    return 100 - 100 / (1+rs)

def simple_moving(df,n):
    sma = df['close'].rolling(window=n).mean()
    return sma

def st_dir_refresh(ohlc,ticker):
    """function to check for supertrend reversal"""
    global st_dir
    if ohlc["st1"][-1] > ohlc["close"][-1] and ohlc["st1"][-2] < ohlc["close"][-2]:
        st_dir[ticker][0] = "red"
    if ohlc["st2"][-1] > ohlc["close"][-1] and ohlc["st2"][-2] < ohlc["close"][-2]:
        st_dir[ticker][1] = "red"
    if ohlc["st3"][-1] > ohlc["close"][-1] and ohlc["st3"][-2] < ohlc["close"][-2]:
        st_dir[ticker][2] = "red"
    if ohlc["st1"][-1] < ohlc["close"][-1] and ohlc["st1"][-2] > ohlc["close"][-2]:
        st_dir[ticker][0] = "green"
    if ohlc["st2"][-1] < ohlc["close"][-1] and ohlc["st2"][-2] > ohlc["close"][-2]:
```

```
        st_dir[ticker][1] = "green"
    if ohlc["st3"][-1] < ohlc["close"][-1] and ohlc["st3"][-2] > ohlc["close"][-2]:
        st_dir[ticker][2] = "green"


def sl_price(ohlc):
    """function to calculate stop loss based on supertrends"""
    st = ohlc.iloc[-1,[-3,-2,-1]]
    if st.min() > ohlc["close"][-1]:
        sl = (0.6*st.sort_values(ascending = True)[0]) + (0.4*st.sort_values(ascending = True)[1])
    elif st.max() < ohlc["close"][-1]:
        sl = (0.6*st.sort_values(ascending = False)[0]) + (0.4*st.sort_values(ascending = False)[1])
    else:
        sl = st.mean()
    return round(sl,1)


def placeSLOrder(symbol,buy_sell,quantity,sl_price):
    # Place an intraday stop loss order on NSE - handles market orders converted to limit orders
    if buy_sell == "buy":
        t_type=kite.TRANSACTION_TYPE_BUY
        t_type_sl=kite.TRANSACTION_TYPE_SELL
    elif buy_sell == "sell":
        t_type=kite.TRANSACTION_TYPE_SELL
        t_type_sl=kite.TRANSACTION_TYPE_BUY
    market_order = kite.place_order(tradingsymbol=symbol,
                exchange=kite.EXCHANGE_NSE,
                transaction_type=t_type,
                quantity=quantity,
                order_type=kite.ORDER_TYPE_MARKET,
```

```
                product=kite.PRODUCT_MIS,
                variety=kite.VARIETY_REGULAR)
    a = 0
    while a < 10:
        try:
            order_list = kite.orders()
            break
        except:
            print("can't get orders..retrying")
            a+=1
    for order in order_list:
        if order["order_id"]==market_order:
            if order["status"]=="COMPLETE":
                kite.place_order(tradingsymbol=symbol,
                                 exchange=kite.EXCHANGE_NSE,
                                 transaction_type=t_type_sl,
                                 quantity=quantity,
                                 order_type=kite.ORDER_TYPE_SL,
                                 price=sl_price,
                                 trigger_price = sl_price,
                                 product=kite.PRODUCT_MIS,
                                 variety=kite.VARIETY_REGULAR)
            else:
                kite.cancel_order(order_id=market_order,variety=kite.VARIETY_REGULAR)



def ModifyOrder(order_id,price):
```

```
    # Modify order given order id
    kite.modify_order(order_id=order_id,
                      price=price,
                      trigger_price=price,
                      order_type=kite.ORDER_TYPE_SL,
                      variety=kite.VARIETY_REGULAR)




def candle_pattern(ohlc_df,ohlc_day,ticker):
    """returns the candle pattern identified"""
    pattern = None
    signi = "low"
    avg_candle_size = abs(ohlc_df["close"] - ohlc_df["open"]).median()
    sup, res = res_sup(ohlc_df,ohlc_day)



    rs = rsi(ohlc_df,14)
    r = rs.iloc[-1]



    if (sup - 1.5*avg_candle_size) < ohlc_df["close"][-1] < (sup + 1.5*avg_candle_size):
        signi = "HIGH"

    if (res - 1.5*avg_candle_size) < ohlc_df["close"][-1] < (res + 1.5*avg_candle_size):
        signi = "HIGH"
```

```
if candle_type(ohlc_df) == 'doji' \
    and ohlc_df["close"][-1] > ohlc_df["close"][-2] \
    and ohlc_df["close"][-1] > ohlc_df["open"][-1]:
        pattern = "doji_bullish"

if candle_type(ohlc_df) == 'doji' \
    and ohlc_df["close"][-1] < ohlc_df["close"][-2] \
    and ohlc_df["close"][-1] < ohlc_df["open"][-1]:
        pattern = "doji_bearish"

if candle_type(ohlc_df) == "maru_bozu_green":
    pattern = "maru_bozu_bullish"

if candle_type(ohlc_df) == "maru_bozu_red":
    pattern = "maru_bozu_bearish"

if trend(ohlc_df.iloc[:-1,:],7) == "uptrend" and candle_type(ohlc_df) == "hammer":
    pattern = "hanging_man_bearish"

if trend(ohlc_df.iloc[:-1,:],7) == "downtrend" and candle_type(ohlc_df) == "hammer":
    pattern = "hammer_bullish"

if trend(ohlc_df.iloc[:-1,:],7) == "uptrend" and candle_type(ohlc_df) == "shooting_star":
    pattern = "shooting_star_bearish"

if trend(ohlc_df.iloc[:-1,:],7) == "uptrend" \
    and candle_type(ohlc_df) == "doji" \
```

```
    and ohlc_df["high"][-1] < ohlc_df["close"][-2] \
    and ohlc_df["low"][-1] > ohlc_df["open"][-2]:
    pattern = "harami_cross_bearish"

if trend(ohlc_df.iloc[:-1,:],7) == "downtrend" \
    and candle_type(ohlc_df) == "doji" \
    and ohlc_df["high"][-1] < ohlc_df["open"][-2] \
    and ohlc_df["low"][-1] > ohlc_df["close"][-2]:
    pattern = "harami_cross_bullish"

if trend(ohlc_df.iloc[:-1,:],7) == "uptrend" \
    and candle_type(ohlc_df) != "doji" \
    and ohlc_df["open"][-1] > ohlc_df["high"][-2] \
    and ohlc_df["close"][-1] < ohlc_df["low"][-2]:
    pattern = "engulfing_bearish"

if trend(ohlc_df.iloc[:-1,:],7) == "downtrend" \
    and candle_type(ohlc_df) != "doji" \
    and ohlc_df["close"][-1] > ohlc_df["high"][-2] \
    and ohlc_df["open"][-1] < ohlc_df["low"][-2]:
    pattern = "engulfing_bullish"



# print(signi)
 #print(pattern)
 #print(r)
 quantity = int(2000/ohlc_df["close"][-1])
```

```
#    print(quantity)


   if signi == "HIGH":
       if(r>=50):
           if(pattern == "engulfing_bullish" or pattern == "harami_cross_bullish" or pattern == "hammer_bullish" or pattern == "maru_bozu_bullish" or pattern == "doji_bullish"):
               # placeSLOrder(ticker,"buy",quantity,sl_price(ohlc_df))
               #placeSLOrder(ticker,"buy",quantity,sl_price(ohlc_df))


               t_type=kite.TRANSACTION_TYPE_BUY
               kite.place_order(tradingsymbol=ticker,
                   exchange=kite.EXCHANGE_NSE,
                   transaction_type=t_type,
                   quantity=quantity,
                   order_type=kite.ORDER_TYPE_MARKET,
                   product=kite.PRODUCT_MIS,
                   variety=kite.VARIETY_REGULAR)


               requests.post('https://textbelt.com/text', {
                     'phone': '+919792836413',
                     'message': 'Hello world BUY ORDER ',
                     'key': 'textbelt',
                     })


               print("BUY ORDER PLACED")
       else:
           if(pattern == "engulfing_bearish" or pattern == "harami_cross_bearish" or pattern == "shooting_star_bearish" or pattern == "hanging_man_bearish" or pattern == "maru_bozu_bearish" or pattern == "doji_bearish"):
```

```
            t_type=kite.TRANSACTION_TYPE_SELL
            kite.place_order(tradingsymbol=ticker,
                exchange=kite.EXCHANGE_NSE,
                transaction_type=t_type,
                quantity=quantity,
                order_type=kite.ORDER_TYPE_MARKET,
                product=kite.PRODUCT_MIS,
                variety=kite.VARIETY_REGULAR)



            requests.post('https://textbelt.com/text', {
                  'phone': '+919792836413',
                  'message': 'Hello world SELL ORDER ',
                  'key': 'textbelt',
                  })

            # placeSLOrder(ticker,"sell",quantity,sl_price(ohlc_df))
            print("SELL ORDER PLACED")


    return "Significance - {}, Pattern - {} , RSI value {}".format(signi,pattern,r)


tickers = ["ZEEL", "WIPRO", "VEDL", "ULTRACEMCO", "UPL", "TITAN", "TECHM",
"TATASTEEL", "TATAMOTORS" , "TCS", "SUNPHARMA", "SBIN", "SHREECEM",
"RELIANCE", "POWERGRID", "ONGC", "NESTLEIND", "MARUTI", "M&M", "LT",
"KOTAKBANK", "JSWSTEEL", "INFY", "INDUSINDBK", "ITC", "ICICIBANK",
"HDFC", "HINDUNILVR", "HINDALCO", "HEROMOTOCO", "HDFCBANK","HCLTECH",
"GRASIM", "GAIL", "EICHERMOT", "DRREDDY", "COALINDIA", "CIPLA",
“BRITANNIA", "INFRATEL", "BHARTIARTL", "BPCL","BAJAJFINSV", "BAJFINANCE",
"BAJAJ-AUTO", "AXISBANK", "ASIANPAINT", "ADANIPORTS", "MCDOWELL-N",
"UBL", "NIACL", "SIEM ENS" ,"SRTRANSFIN", "SBILIFE", "PNB", ]
```

```
def main():


    for ticker in tickers:
        try:
            ohlc = fetchOHLC(ticker, '3minute',5)
            ohlc_day = fetchOHLC(ticker, 'day',30)
            ohlc_day = ohlc_day.iloc[:-1,:]
            cp = candle_pattern(ohlc,ohlc_day,ticker)
            print(ticker, ": ",cp)


        except:
            print("skipping for ",ticker)



# Continuous execution
starttime=time.time()
timeout = time.time() + 60*1*1 # 60 seconds times 60 meaning the script will run for 1 hr
while time.time() <= timeout:
    try:
        print("passthrough at ",time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(time.time())))
         main()
        time.sleep(180 - ((time.time() - starttime) % 180.0)) # 300 second interval between each
new execution
    except KeyboardInterrupt:
        print('\n\nKeyboard exception received. Exiting.')
        exit()
```