

**A PROJECT REPORT**

**ON**

**SMART ALGO TRADING**

*Submitted In Partial Fulfillment of the Requirement for the Award of* **Post Graduate Diploma in Artificial Intelligence (PG-DAI)** Under the Guidance of

**MR. NIMESH DAGUR**

**(Project Guide)**



**Submitted By**

**NAGARJUN HONRAO SANKET PATIL PRASAD KOPARDE**

**PRN: 220320528035 PRN: 220320528036** **PRN: 220320528058**

# CONTENTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INDEX** | **TITLE** | | | **PAGE NUMBER** |
| 1 | Certificate | | | **2** |
| 2 | Acknowledgement | | | **3** |
| 3 | Abstract | | | **4** |
| 4 | Introduction | | | **5-6** |
| 5 | Related Work | | | **6-8** |
| 6 | Design & Implementation | | | **8** |
| 7 | Jupyter | | | **8-9** |
| 8 | RSI & EMA Indicators | | | **9-12** |
| 9 | TradingView | | | **12-13** |
| 10 | Coding Work | **14-18** | | |
| 11 | Visualization & Results | **19-21** | | |
| 12 | Conclusions & Future Scope | **21-22** | | |
| 13 | References & Bibliography | | **23** | |

CDAC, B-30, Institutional Area, Sector-62 Noida (Uttar Pradesh)-201307

# CERTIFICATE CDAC, NOIDA

This is to certify that Report entitled **“SMART ALGO TRADING(SAT BOT)”** which is submitted by Nagarjun Honrao, Sanket Patil and Prasad Koparde in partial fulfillment of the requirement for the award of **Post Graduate Diploma in Artificial Intelligence** (PG-DAI) to **CDAC, Noida** is a record of the candidates own work carried out by them under my supervision.

The documentation embodies results of original work, and studies are carried out by the student themselves and the contents of the report do not from the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**MR. NIMESH DAGUR**

**(Project Guide)**

# ACKNOWLEDGEMENT

We would like to express our best sense of gratitude & endeavor with respect to **Mr. Nimesh Dagur (Project Guide)** CDAC, Noida for suggesting the problems scholarly guidance and expert supervision during the course of this project. Special thanks to **Mr. Ravi Payal (Program Coordinator).**

We are very thankful to **Mr. Nimesh Dagur (Project Guide)** the project guide for constant simulated discussion, encouraging new ideas about this project.

**NAGARJUN HONRAO SANKET PATIL PRASAD KOPARDE**

**PRN: 220320528035 PRN: 220320528036** **PRN: 220320528058**

**ABSTRACT**

The objective of this project is to make a trading bot in python to facilitate trading. The bot will run on its own and try to make accurate trades that will result in an accumulation of currency for the user. The bot is written in python. It makes use of the TradingView API and python's libraries. The trading strategy involves the use of RSI and EMA. The trading bot makes multiple buys and sell calls based on the information that it gets from the RSI and EMA, and tries to maximize profit while limiting risks.

# INTRODUCTION

In the last couple of years, stock market has become the latest fad in investment and money management. Currently, there are hundreds of different stocks that are available to trade. This leads to a lot of volatility, which means that money can be made and lost easily.

Human emotion is usually one of the biggest barriers to successful trading pedigree. The market is sometimes irrational and doesn’t follow expected paths. It is usually our greed that takes over and causes us to lose money and not take profits when they should be taken. In Fenton-O’Creevy journal, *Emotion regulation and trader expertise: heart rate*

*variability on the trading floor,* he argues that,

" Investors who experienced more intense emotional reactions to gain and loss were poorer performers than those with more attenuated emotional responses.” .What this teaches us is that in order to become better traders we must rid ourselves of our emotions. The only thing we can do that with any sense of finality is by getting rid of the human variable. If we can create an environment where we have more control of the final outcome, then we are on the right track towards profit.

In order to get rid of the human variable, we have to automate trades. This project uses a python trading bot to make most of the trades. The strategy is based on two indicators. The indicators are RSI and EMA. The bot will check at these two indicators, and make appropriate moves, and take appropriate strategy in order to maximize profit.

**RELATED WORK**

There are a few trading bots in the market; however, they may not be completely the same as the one designed and built in this project. For example, one important trading bot is called Zenbot.It can run on your personal computer or hosted on the cloud. There are a lot of cons including sms, discord and other notifications and paper and live trading however the technical experience needed is high and everything is common to the line based so it wasn’t the right choice for this project. Another popular trading bot is the catalyst which has good documentation of a good community and statistical analysis and machine learning support. Unfortunately, it is difficult to set up. It doesn’t really support live trading and there is a lengthy installation process. Overall the best choice to mimic was Brad Lindblad’s Trading Bot.

In the article, “Build A Cryptocurrency Trading Bot with R” , Brad Lindblad, a data scientist, talks about building his Trading Bot using the R Trading Language. In order to implement his trading bot, Brad Lindblad used the rgdax package which was an R wrapper for the GDAX API. His work is similar to this project but instead, Python wrapper for the Tradingview API is used in this project.For his strategy, Lindblad uses the Ethereum USD pair(Cryptocurrency).

The strategy adopted in uses the Relative Strength Index indicators in order for the bot to find a good place to enter the market. If the indicators told the bot that the market was being oversold, then the bot kicks in and takes a position. The more interesting part takes place during the selling phase. Lindblad’s strategy relies on three limits to sell orders. When he made 1% profit, he sold part of his stock. When he made 4% profit, he sold another part of his stock. Finally, he sold his last part at 7%.

To automate the above strategy, Lindblad used Windows Task Scheduler for this task. He chooses the script to run every 10 minutes. Overall Lindblad added a few other variables, such as, only running the script when the balance of his account was $20 or greater, and when the RSI was 30 in the current period and the previous 3 periods at least once. This tied up all the loose ends in his strategy and made it profitable.

**Design and Implementation**

This chapter discusses the detailed design and implementation of the python trading bot. It goes over all the frameworks. It goes over Jupyter and then discusses other possible frameworks. It then goes into detail about RSI and EMA indicators and it discusses a few charts and how they relate to the project.

**Frameworks - Jupyter**

In this project, the framework used for the trading bot was Jupyter .The reason that it was used was because of the features and capabilities of the framework. The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects.

A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media. In other words: it’s a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable.

Using Notebooks is now a major part of the data science workflow at companies across the globe. If your goal is to work with data, using a Notebook will speed up your workflow and make it easier to communicate and share your results.

**RSI INDICATORS**

Relative Strength Index is what’s called a momentum indicator. It shows you in what direction the market is heading towards. It compares the number of times that the price closed in an upwards trend vs the number of times it closed in a downward trend. From this information, the Relative Strength Index is assigned a score from 0-100.

The Relative Strength Index (RSI) tells you if something is being oversold or overbought. For example, if the RSI score is over 70 then the stock can be thought of as being overbought. This means it would be a good time to sell. However, if the RSI score is below 30 then the stock can be thought of as oversold . In this case, it would be a good time to think about entering into a position.

This is very similar to what the Bollinger Band Indicators say. So both the Bollinger bands and the RSI Indicators can be used in conjunction to determine whether to enter into a trade or to leave it.

If the price is touching the lower Bollinger Band and the RSI is under 30 then the stock is probably oversold. This is where it would be a good idea to make a buy. The opposite is also true. For example, if the RSI is over 70 and the price is touching or approaching the upper band then it is probably under bought. This would be a good opportunity to make a sell.

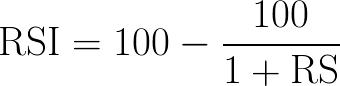
**EMA INDICATORS**

The exponential moving average(EMA) is a technical chart indicator that tracks the price of an investment (like a stock or commodity) over time. The EMA is a type of weighted moving average(WMA) that gives more weighting or importance to recent price data. Like the relative strength index(RSI), the EMA is used to see price trends over time, and watching several EMAs at the same time is easy to do with moving average ribbons. The EMA is designed to improve on the idea of an SMA by giving more weight to the most recent price data, which is considered to be more relevant than older data. Since new data carries greater weight, the EMA responds more quickly to price changes than the SMA does.

**Setting up the RSI calculations**

In order to get the calculation of the RSI indicator, a specific formula is used. Basically, one needs to use the RSI calculations of the previous 30 days. Here, we used the RSI which goes as follows.

**RSI=100-100(1+RS)** where RS is the average gain over the average loss of the last 30 days. The RSI basically relies on the fluctuations in the price of the cryptocurrency. It measures the average gain over the average loss.

****

**RSI FORMULA**

**Tradingview**

TradingView is a platform that lets you customize technical indicators, create charts, and analyze financial assets. These indicators are patterns, lines, and shapes that millions of traders use every day. TradingView is entirely browser-based, with no need to download a client.

TradingView is a browser-based charting platform and screener for cryptocurrencies and other financial assets. Its charting tools are also available to use natively in Binance’s trading UI. Apart from charting, you can also share your trading strategies and live stream your analysis.

When it comes to free options, TradingView is a powerful tool for all experience levels. The basic features should be enough for most traders. However, be careful browsing through other users’ ideas and streams. It’s important to filter out the information that adds value to your strategy rather than blindly following others. For traders who love technical analysis, robust charting tools are essential. TradingView is one option for both amateur and experienced traders. It offers numerous trading and charting tools and also a free membership option. Let’s face it... not everyone has the money or need for a Bloomberg terminal subscription.

If you’ve been trading on Binance, you might have already noticed TradingView’s tools available on the exchange. But without understanding them, it can look a little intimidating. So better start learning.

**CODING WORK**

Importing Libraries

from nsepy import get\_history

#to get publicly available data on the current NSEIndia site by communicating with their REST APIs.

from datetime import date, datetime

# supplies classes for manipulating dates and times.

import os

#provides functions for interacting with the operating system.

from tradingview\_ta import TA\_Handler, Interval, Exchange

#unofficial Python API wrapper to retrieve technical analysis from TradingView

import time

#time module allows to work with time in Python.

from selenium import webdriver

#selenium represents a set of api's that can be used to control a browser programmatically.

#webdriver used for the validation of websites (and web applications)

from selenium.webdriver.chrome.service import Service

#Returns us an instance of Chrome driver through which we will be interacting with Chrome browser.

#service will open chrome browser

from webdriver\_manager.chrome import ChromeDriverManager

from selenium.webdriver.common.by import By

#os.system("figlet -c Smart Algo Trading Bot ")

Today = date.today()

y = Today.strftime("%Y")

m = Today.strftime("%m")

d = Today.strftime("%d")

# d = "30"

#last order

last\_order="sell"

sold\_before = False

bought\_before = False

current\_price = 0

take\_profit = 0.0

take\_loss = 0.0

#load chrome driver

# driver = webdriver.Chrome(executable\_path="/Users/mrm/Downloads/chromedriver")

s=Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=s)

driver.maximize\_window()

driver.get("https://in.tradingview.com/")

time.sleep(60)

#initiating tradingview handler to get the recomendation for sonata software for 15 min interval

ssw = TA\_Handler(

    symbol="RELIANCE",

    screener="india",

    exchange="NSE",

    interval=Interval.INTERVAL\_5\_MINUTES

)

def countdown(t):

    while t:

        mins, secs = divmod(t, 60)

        timer = '{:02d}:{:02d}'.format(mins, secs)

        print(timer, end="\r")

        time.sleep(1)

        t -= 1

while True:

    now = datetime.now()

    current\_time = now.strftime("%H:%M:%S")

    if(current\_time >= "09:30:00" and current\_time < "15:10:00"):

        rec = ssw.get\_analysis()

        RSI = rec.indicators["RSI"]

        # MACD = rec.indicators["MACD.macd"]

        EMA = rec.moving\_averages["COMPUTE"]["EMA10"]

        print("RSI:", RSI, "EMA:", EMA)

        if ( RSI >= 30 and RSI <= 70 and EMA == "BUY" ):

            if (last\_order=="sell"):

                print("Buying 1 stock of RELIANCE")

                last\_order="buy"

                print(last\_order)

                print(sold\_before)

                #buy 1 stock of RELIANCE

                driver.find\_element(By.XPATH,"//div[2]/div[2]/div[1]/div/div/div[1]/div[2]/div/div[2]").click()

                driver.find\_element(By.XPATH,"//div[2]/div[2]/div[1]/div/div/div[1]/div[6]/button/div/span[2]").click()

                current\_price = driver.find\_element(By.XPATH,"/html/body/div[2]/div[1]/div[2]/div[1]/div/table/tr[1]/td[2]/div/div[2]/div[1]/div[2]/div[1]").text

                print(current\_price)

                take\_profit = float(current\_price) + 8

                take\_loss = float(current\_price) - 5

                while True:

                    print("Time left till next call - ")

                    countdown(int(5))

                    rec = ssw.get\_analysis()

                    RSI = rec.indicators["RSI"]

                    # MACD = rec.indicators["MACD.macd"]

                    EMA = rec.moving\_averages["COMPUTE"]["EMA10"]

                    print("RSI:", RSI, "EMA:", EMA)

                    current\_price = driver.find\_element(By.XPATH,"//div[2]/div[1]/div[2]/div[1]/div/table/tr[1]/td[2]/div/div[2]/div[1]/div[2]/div[1]").text

                    if((RSI >= 30 and EMA == "SELL") or (float(current\_price) >= take\_profit) or (float(current\_price) <= take\_loss)):

                        #sell the stock

                        print("Selling 1 stock of RELIANCE")

                        last\_order="sell"

                        print(last\_order)

                        #sell 1 stock of RELIANCE

                        driver.find\_element(By.XPATH,"//div[2]/div[2]/div[1]/div/div/div[1]/div[2]/div/div[1]").click()

                        time.sleep(2)

                        driver.find\_element(By.XPATH,"//button[1]/div[1]/span[2]").click()

                        break

                    else:

                        print("no adjustment required")

            else:

                print("last order not sold")

        elif( RSI >= 50 and EMA == "SELL" ):

            if ( last\_order == "sell"):

                print("selling stock of  RELIANCE")

                driver.find\_element(By.XPATH,"//div[2]/div[2]/div[1]/div/div/div[1]/div[2]/div/div[1]").click()

                time.sleep(2)

                driver.find\_element(By.XPATH,"//button[1]/div[1]/span[2]").click            ()

                current\_price = driver.find\_element(By.XPATH,"//div[2]/div[1]/div[2]/div[1]/div/table/tr[1]/td[2]/div/div[2]/div[1]/div[2]/div[1]").text

                print(current\_price)

                take\_profit = float(current\_price) - 8

                take\_loss = float(current\_price) + 5

                while True:

                    print("Time left till next call - ")

                    countdown(int(5))

                    rec = ssw.get\_analysis()

                    RSI = rec.indicators["RSI"]

                    EMA = rec.moving\_averages["COMPUTE"]["EMA10"]

                    print("RSI:", RSI, "EMA:", EMA)

                    current\_price = driver.find\_element(By.XPATH,"//div[2]/div[1]/div[2]/div[1]/div/table/tr[1]/td[2]/div/div[2]/div[1]/div[2]/div[1]").text

                    if((RSI <= 30 and EMA == "BUY") or ( float(current\_price) <= take\_profit) or (float(current\_price) >= take\_loss)):

                        #buy the stock

                        print("Buying the stock")

                        driver.find\_element(By.XPATH,"//div[2]/div[2]/div[1]/div/div/div[1]/div[2]/div/div[2]").click()

                        driver.find\_element(By.XPATH,"//div[2]/div[2]/div[1]/div/div/div[1]/div[6]/button/div/span[2]").click()

                        break

                    else:

                        print("no adjustment required")

        else:

            print("condition not favourable..waiting")

    elif(current\_time >= "15:10:00"):

        print("Time to close for the day")

        # #fetch open profit

        open\_profit = driver.find\_element(By.XPATH,"//div[2]/div[7]/div[2]/div[4]/div/div[1]/div/div[2]/div[3]/div[1]").text

        # print(open\_profit)

        # P = "1000"

        print("Calculating profit :",open\_profit)

        break

    else:

        if(current\_time >= "09:15:00" and current\_time < "09:30:00"):

            print("Analysing market","\n\n")

        elif(current\_time < "9:15:00"):

            print("Waiting for market to open")

        else:

            print("No action required")

# WORKING

# 1. Run SAT BOT.py at 9:00 AM on open market days.

# 2. Bot tries to read the RELIANCE current value.

# 3. Bot runs forever, starting from 9:00AM till 03:00 PM in the interval of 5 mins.

# 4. Bot used strategies based on RSI and EMA.

# STEPS TO RUN

# 1. execute – SAT BOT.py - this will open a browser, login to your tradeview account and connect to paper trading.

# 2. Open the trading box by clicking on the icon shown in the image below –

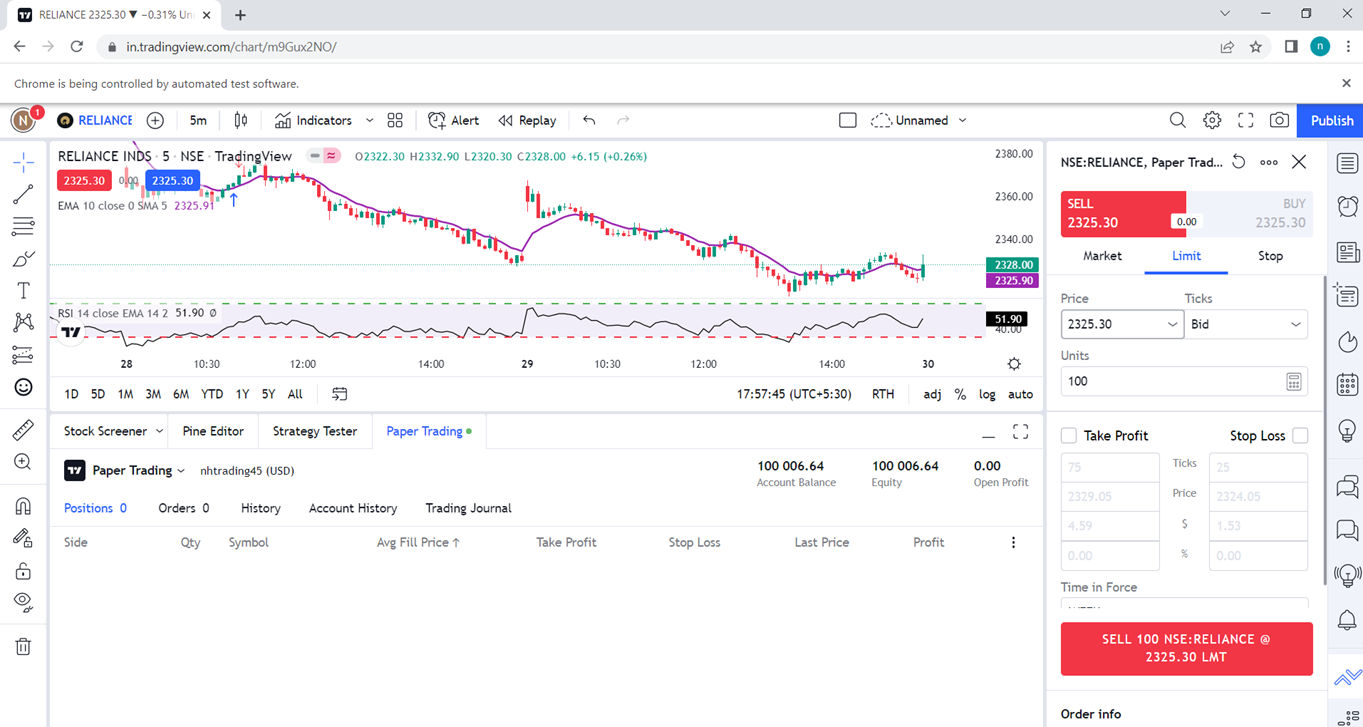
# 

# 

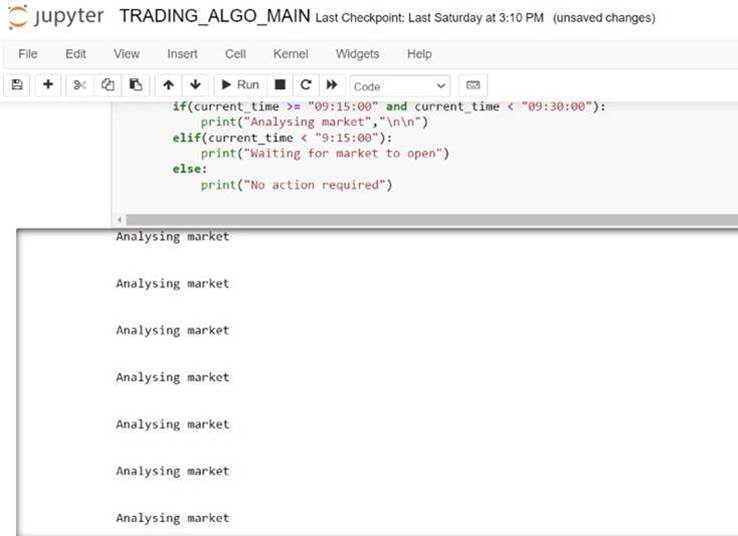
# 3. Let the code run. It will close at 3:00PM and sums up your profit for the day.

# VISUALISATION & RESULTS

# After executing the code successfully the bot automates a new window as shown below:-



**Afterwards, the bot will start analyzing the market.**

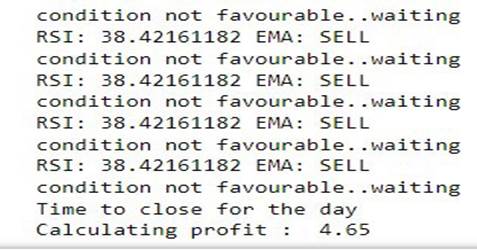


**When the strategy meets the requirements,the Bot then starts Buying and Selling the** **stocks. For Ex.RELIANCE**

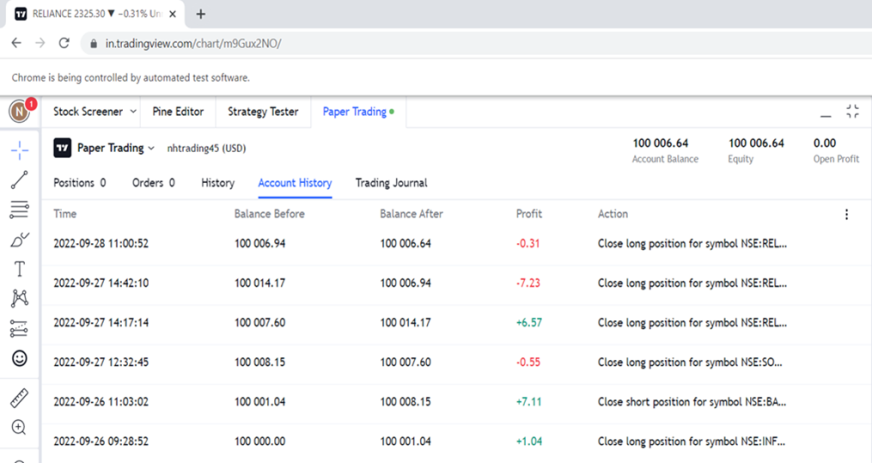


**After using RSI and EMA strategies the bot will run until the market closes and finally calculates the Profit/Loss.**

**Here we have earned Profit in a Single Day/Intraday.**



**Final Results for various stocks in a single day.**



**CONCLUSION AND FUTURE SCOPE**

Overall the trading bot did end up making a profit however the profits were small. In order to improve on this design, a few things had to be done to improve profit. Many more trades could have been set up in order to increase profit. The parameters for this trading bot were to sell every time it made a profit at 1%. This is a relatively small amount of profit. However, if many trades are done, for example, if it is scaled to 100s of trades then profits will also be scaled. This can be done by trying to day trade in smaller time zones. If we go to the minute time frames, then we can set the profit margin at an ever-lower ceiling. For example, we can exit a trade when it is about .05% in profit. This makes it easier to take advantage of volatility, therefore, making it easier to exit and enter trades. In a future implementation, additional technical indicators can be adopted such as a madcow indicator.

# REFERENCES & BIBLIOGRAPHY

* Venkataraman, D., Parameswaran, N. S. "Extraction of Facial Features for Depression Detection among Students." International Journal of Pure and Applied Mathematics, International Conference on Advances in Computer Science, Engineering and Technology, pp. 455-462, 2018.
* Owayjan, Michel, Roger Achkar, and MoussaIskandar. "Face Detection with Expression Recognition using Artificial Neural Networks." In Biomedical Engineering (MECBME), 3rd Middle East Conference on, pp. 115-119. IEEE, 2016

Articles:

Reference Document -

1. chromedriver - <https://sites.google.com/a/chromium.org/chromedriver/getting-started>
2. Tradingview sample code - <https://python-tradingview-ta.readthedocs.io/en/latest/overview.html>
3. Data Fetched From - <https://www.nseindia.com>
4. Libraries - <https://pypi.org/project/selenium/>

[https://python-tradingview-ta.readthedocs.io/en/latest](%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20https://python-tradingview-ta.readthedocs.io/en/latest)