



**INTERNATIONAL JOURNAL OF NOVEL RESEARCH
AND DEVELOPMENT (IJNRD) | IJNRD.ORG**
An International Open Access, Peer-reviewed, Refereed Journal

TRADING VICE VOLUME-II

KSHITIJ SHARMA

Student at Amity University

SAKSHAM SARDANA

Student at Amity University

KRITI CHANDEL

Student at Amity University

DR. NIDHI MISHRA

ASISTANT PROFESSOR (III)

AMITY UNIVERSITY

ABSTRACT

Trading Vice is a trading assistant that helps to analyse market and provides a variety of tools to make profitable trades happen. Trading vice believes in making trades efficient, fast and profitable. We aim exclude any kind of manual error while trading.

Trading vice is beneficial for both beginners and pro traders but we advice people to have prior trading knowledge for to start trading with trading vice. In particular with Trading Vice we provide Scripts majorly in Python and in Pine script format. These are widely used formats in trading industry. This helps

people to directly start trading with their favourite programming friendly trading platform.

As third party platform we help people to get their desired strategy for trading and get efficient trades. People can fetch trading scripts from our website. These scripts are ready to use and generates valuable results if used correctly, at correct time and in correct stock and market domain.

We back all our scripts and methodologies with pure mathematical and logical backgrounds equipped with

knowledge of scripting and coding to automate the cycles and make trading systems run in perfect periodic cycles.

Trading vice is backed with strategies with mathematical logics and are well tested in market by general public and economists. We provide our services for stocks, forex, commodities and crypto markets. We highly recommend our strategies for people who want indulge in trading given that they have prior experience and knowledge.

Investing in markets is subject to market risks. We only hold up for the performance of scripts at correct time and in correct markets. We are not responsible for any losses. A trader to use strategies with own risk and intelligence of the market.

We suggest people to back test the strategy scripts before subjecting them for real money in live markets.

Chapter 1: Introduction

Trading Vice is a trading assistant. It is a third party portal which provides for scripts to run and make trades possible efficiently and profitably without much human interventions. We aim to cut time loss between deciding and placing an order. Placing an order can take a lot of time. While placing an order one has to provide for a lot parameters like quantity, stop-loss, target, trailing stop loss, etc. While filing up these parameters price for single unit share fluctuates and sometimes can make huge differences from the

price one wanted to step into the trade.

In order to remove this human action latency we prepare to come with a solution, Trading Vice. We aim to automate the trades. People want to use the bots, they want to automate trades, want make their trades efficient but have to pay a cost for it. As of now Trading Vice is a totally free platform helping people to get their desired results and take the shots in markets for maximum profits and low latency rates.

We provide Python and Pine trading scripts for people to use and automate their trades. We do so because these languages are very trading industry friendly and beginners tend to understand the logic comfortably. People can have access to these scripts from our web page and can use any programmable trading platforms to apply these scripts and reap maximum benefits.

Our scripts are backed with mathematical, trading and coding logics. These scripts are thoroughly back tested and are widely used by the people in industry for their trading and market capitalisations. We highly recommend people to have prior knowledge before using these scripts and stepping into the trading industry. We only promise to

make trades efficient and latency proof to get profitable trades but if one doesn't know about when and where to use a strategy, we do not take responsibility for any losses.

All of our trading scripts are back tested and then only brought to public domain. We only wish to make the trading experience an easy and efficient one. We promise to improve and bring more products with extended usage of people, collected data and feedback will help us to refine our product.

Chapter 2: Literature Review

Literature review particularly holds all the knowledge we gained while researching on this project. A project is just like a baby and a project to achieve such big motive we had to be really careful with what information we want to choose for our use and what to exclude. We curate this project and our research paper with best of information and resources handpicked from best sources. In this project we had just not researched through the orthodox research paper method but from individual people too, either by talking or by going through their work and speeches.

Let's see and understand key drawings of our understandings,

2.1: Market Analysis

Let's first learn basic things about market, current situations and some strategies to understand gist of it otherwise this subject is another research paper all together.

Particularly when we talk about any market now onwards in this research paper, we will be refereeing to markets especially NSE, Forex and Crypto. These markets are very volatile since past quarter and people are able to bag good trades.

Bank Nifty and Nifty 50 did hit their all time high for last season and recorded some good trades as well. With major volatility amount of buyers and suppliers increases significantly in these markets and high volumes are traded.

A lot of indicators and studies are used to predict market situation in order to bag good trades in specific time frame. Market is also prone on repeating historic trends and with knowledge of history in markets one can really take advantage of stock markets.

Multiple companies are formed everyday and prepare to be listed on Indian stock market list. Share rates of each company depend upon multiple factors such as financial and technical aspects of the company. These factors are the sole reason for market and price fluctuations.

Markets are volatile these days and with great prediction of recession all

the traders will be in a rush to bag good trades everyday and want to make as much money as possible.

2.2: Basic Trading Terms

Relevant to our use we need to know some basic terminologies only. These terminologies will be function of our bot to perform in real time.

- **Sell order:** This means whenever we want to sell a share or a stock of any company for a certain price we put in a sell order.
- **Buy order:** This means whenever we want to buy any share or a stock of any company for a certain price we put a buy order.
- **Stop loss:** This is the amount below or above which we will exit the trade if we are incurring any loss in the trade. Scenarios are different for both selling and buying options.
- **Trigger Point/ Target:** This is the desired amount where we limit and exit our trade after making desired profit. Scenarios are different for both selling and buying options.
- **Short Sell:** First we sell the stock and buy it afterwards.

- **Risk to reward ratio:** Risk to reward ratio is a simple calculation to know what we are risking to get a desired result. E.g., Giving out 10 rupees to earn 11rs is not good instead of giving 5rs to earn 11rs.

These terminologies are enough for us start working on our bot trading vice. Prior to that we just need to understand how market works in brief.

In market there are buyers on one end and sellers on the other. On one end there is always a buyer to buy and a seller to sell from the other end. If a buyer wants to buy some share for a certain point then similarly there should be a seller too selling at the same share at same price. This is how the trading markets work. They make buyers and sellers meet under one roof and carry out the trade business.

2.3: Understanding Psychology

All around the world traders fight on for a price for a respective share which can turn their shots profitable. Everyone only wants profit, obviously. More importantly if someone wins that means someone has to lose. This is basics of economics. Economics means transit of ruling tradable commodity from one to another. Generally this pattern

is mostly evident when people say “poor getting more poor and rich getting richer”. This means most often the money is getting out of the hands of poor and going into the hands of rich people. At trading we see same pattern. People with no knowledge to trade and with less or maybe all the money they have they just pool-in and incur losses. This cycle just keeps on repeating.

Trading Vice is a platform which helps people to execute trades with what knowledge we have and with our product offerings we can help people to at least get good trades and get their money back with some profit. Here we are helping people who don't have time to dive deep and learn but know few basics about trading.

One should never let any emotion to fall in any trade. Trading should be emotion less. There should be no hesitation in getting into a trade, no regret in losing a trade and no greed to have more.

Chapter 3: Technologies Used

Technologies used particularly to develop our Scripts are python and Pine scripts.

3.1: Python

High-level programming language Python is used in our scripts due to the easy comprehension of client with little or no coding experience.

Python is very lucid because it goes hand in hand due to its easy understanding and easy debugging on the fly. Other than that python has numerous libraries to support trading and due its analytics and mathematical libraries coding logics have become so much easy to implement and fetch results in graph and plot formats.

Libraries extensively used in our scripts-

Pre-dominantly we kept our code simple and easy. We use Yfinance, Pandas and NumPy.

3.2: NumPy

NumPy is basically a library which supports multiple heavy duty mathematical expressions for to implement in python. It also helps with multidimensional arrays and matrices. As we know python is a high level language which first computes the code in low level language and then runs the code. Hence the compilation takes time and execution time increases. Thus using NumPy is very handy.

3.3: Pandas

Pandas is a python library helps in providing data structures in fast and flexible manner with both labelled and relational data.

3.4: Yfinance

Yfinance is a python based Yahoo library which provides us with data to back test our strategies and let us know the results for our strategy.

3.5: Pine Scripts

Pine is a programming language specifically developed by Trading View. It is a light weight programming language with major objective to build scripts for trading purposes only. It is not based on any other programming language. With pine scripts it is easier to backtest and evaluate all the strategies.

3.6: Trading View

Trading View is a trading platform but has extended usage of scripting and tools to analyse the charts. It is a terrific platform for trading. No trader is unfamiliar of Trading View. It is one of a kind platform, built to cater its client with the best technology and trading experience.

Trading view has its own editor to type the code in which integrates beautifully with the live trading charts. Plus it also enhances the experience with its tools to test the strategies and it gives results in very analytical manner with proper graphs and testing metrics.

Chapter 4: Trading Vice

Trading Vice is a digital platform which is offering digital products for efficient and profitable market trading. We choose to offer our products in form of scripts in python and pine script languages which are widely used in trading industry. We did all our research and through testing while finalising our scripts and bringing our product to finesse.

Other than the scripts we have a recommender system which recommends about if we need to buy a stock or not. This recommender system is universal in nature and can give feedback on any stock trading in any market around the globe.

We package all of our products in a web app. All the scripts can be copied from there and the recommender system can be accessed from the same portal itself too.

Let's discuss how we made our web application.

Technologies Used-

HTML:

Hypertext mark-up language is a basis of web application. With the help of it we can give structure and body to our web application. HTML is useful and an integral part of all web applications. It help the developer and to make the applications well structured and oriented.

Node.js is a backend language which helps up to take care of all the get and post requests. With Node.js we can show the world about what we have made. Best thing about Node.js is, is that it uses JavaScript only for all of its coding so it becomes very handy for developer and development time decreases significantly.

CSS Style Sheets:

Cascading style sheets, is format of file in which we can code to provide our HTML structures with styling and options of modifications to enhance overall appearance of whole web application.

Bootstrap:

It is a library which helps to beautify and modify our web application. Bootstrap has a lot of elements do choose from and to modify from. With this we can render our web application on fly without wasting much effort on front-end.

JavaScript:

JavaScript is the language which provides the functionality to our bare HTML code. With HMTL and CSS, a web application can only stand and look good but will be static and unresponsive in nature. JavaScript adds functionality and gives a dynamic as well as responsive edge to the web application. With JavaScript we can enable horizon of abilities a web application can ever have.

Chapter 5: Trading Vice product offerings

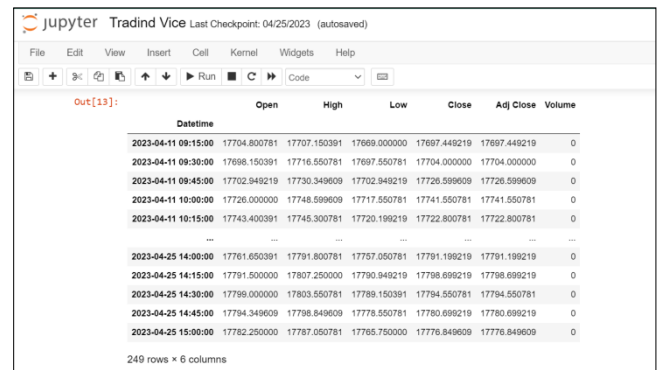
Trading vice has three main strategies to be executed for obtaining good trades in market. As of now we are having one trading script in python and 2 scripts in pine scripts.

Let's discuss first Python based script for trading.

Node.js:

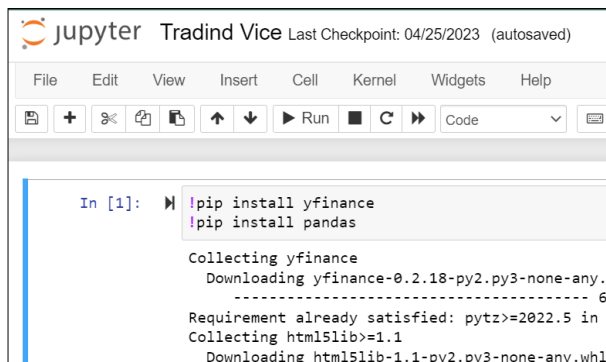
5.1: Beginner's Script (Recommender Script)

This is a python script which call as beginner's script because this script analysis and give signals based on the stock price one chooses. It is particularly for beginners as it tells about how many buy and sell opportunities are there for profitable trades where big volatilities were recorded.



	Open	High	Low	Close	Adj Close	Volume
2023-04-11 09:15:00	17704.800781	17707.150391	17669.000000	17697.449219	17697.449219	0
2023-04-11 09:30:00	17698.150391	17716.550781	17697.550781	17704.000000	17704.000000	0
2023-04-11 09:45:00	17702.949219	17730.349609	17702.949219	17726.599609	17726.599609	0
2023-04-11 10:00:00	17726.000000	17748.599609	17717.550781	17741.550781	17741.550781	0
2023-04-11 10:15:00	17743.400391	17745.300781	17720.199219	17722.800781	17722.800781	0
...
2023-04-25 14:00:00	17761.650391	17791.800781	17757.050781	17791.199219	17791.199219	0
2023-04-25 14:15:00	17791.500000	17807.250000	17790.949219	17798.699219	17798.699219	0
2023-04-25 14:30:00	17799.000000	17803.550781	17789.150391	17784.550781	17784.550781	0
2023-04-25 14:45:00	17794.349609	17798.849609	17778.550781	17780.699219	17780.699219	0
2023-04-25 15:00:00	17782.250000	17787.050781	17765.750000	17778.849609	17778.849609	0

Figure 5.3



```
In [1]: !pip install yfinance
!pip install pandas

Collecting yfinance
  Downloading yfinance-0.2.18-py2.py3-none-any.
  ----- 6
Requirement already satisfied: pytz>=2022.5 in
Collecting html5lib>=1.1
  Downloading html5lib-1.1-py2.py3-none-any.whl
```

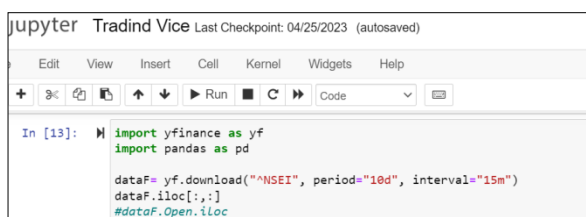
Figure 5.1

Here we are importing Yfinance and pandas. Then we are fetching data from yfinance API to test our strategy. Here we have chosen NSE Index for period of last 10 days and with intervals of 15 mins. Here with 15 mins we refer to the every candle that is generated in every 15 mins. Here is an example:



Figure 5.4

First we install our libraries. Here we are installing pandas and Yfinance. Pandas help us with visualising data in table formats. Yfinance provide us with data. We use historic data for testing and generating results and hence we fetch data from yahoo finance API.

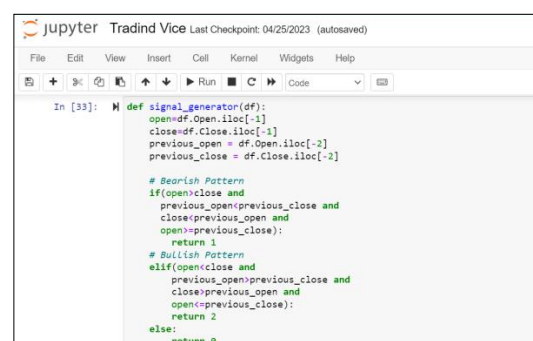


```
In [13]: import yfinance as yf
import pandas as pd

dataF= yf.download("^NSEI", period="10d", interval="15m")
dataF.iloc[:,:]
#dataF.Open.iloc
```

Figure 5.2

Coming back to the code:



```
In [33]: def signal_generator(df):
open=df.Open.iloc[-1]
close=df.Close.iloc[-1]
previous_open = df.Open.iloc[-2]
previous_close = df.Close.iloc[-2]

# Bearish Pattern
if(open>close and
previous_open>previous_close and
close<previous_open and
open<previous_close):
return 1
# Bullish Pattern
elif(open<close and
previous_open<previous_close and
close>previous_open and
open>previous_close):
return 2
else:
return 0
```


Figure 5.5

Here we want to generate signals for our reference.

Here we are first recording open and close prices and then we are referring to open and close prices of from previous day. Then we want to establish bearish and bullish patterns. Bearish pattern is the one when market is going down and bullish pattern is when market is going up.

To get a signal for bullish pattern we want opening price to be smaller than closing price. Additionally we want previous opening price to be bigger than previous closing price with closing price to be bigger than previous opening price and opening price to be smaller than or equal to previous close.

For bearish pattern we just perform the above strategy vice versa.

```

signal = []
signal.append(0)
for i in range(1,len(dataF)):
    df = dataF[i-1:i+1]
    signal.append(signal_generator(df))

#signal_generator(data)
dataF["signal"] = signal
dataF.signal.value_counts()
#dataF.iloc[:,:]

```

```

Out[33]: 0      219
         2       19
         1       11
         Name: signal, dtype: int64

```

Figure 5.6

Now we call our function and we want to return values such as 0 for no trade, 1 for bearish trades and 2 for bullish trades.

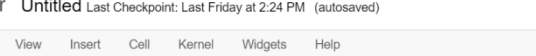
Here we can see that in last 10 days we had 219 instances with no trade options but 11 selling opportunities and 19 buying opportunities. With a total of 30 opportunities, NSE proves to be good index to be traded.

5.2: API Assisted Recommender:

Api assisted recommender is basically based on an API call which uses its own algo to recommend buying or selling decision for any particular stock. We have made this available because we want to be sure and double check our recommender with the ones which are available on web.

So to make this recommender system we used python script to analyse market and stock. Here we use trading view api and its recommender system to ensure our trades.

Let's code it.



The screenshot shows the JupyterLab interface. At the top, the title bar reads "upyter Untitled Last Checkpoint: Last Friday at 2:24 PM (autosaved)". Below the title bar is a menu bar with options: Edit, View, Insert, Cell, Kernel, Widgets, and Help. Under the "Kernel" menu, the "Run" option is highlighted. Below the menu bar is a toolbar with icons for file operations (new, open, save, close), cell navigation (up, down), execution (run, interrupt), and a dropdown menu currently set to "Code". The main area is a code editor with a light blue background. It contains two input cells. The first cell, labeled "In [4]:", contains the following Python code:

```
from tradingview_ta import TA_Handler, Interval, Exchange
import tradingview_ta
```

The second cell, labeled "In [5]:", contains the following Python code:

```
print(tradingview_ta.__version__)
```

The output of the second cell is displayed below the code:

```
3.3.0
```

Figure 5.7

First we have to import Trading analysis handler with interval and exchange parameters too from tradingview trading analysis and for all of this we need to import trading view analysis first.

Then just as a precautionary it is a good practice to check the version first before coding any further.



```
In [6]: M handler = TA_Handler(
        symbol="tesla",
        exchange="nse",
        screener="america",
        interval="10d",
        timeout=None
    )

In [7]: M from tradingview_ta import *
        analysis = get_multiple_analysis(screener="america", interval=Interval.INTERVAL_1_DAY, symbols=["nadaq:tsla", "nyse:docn"],
        )

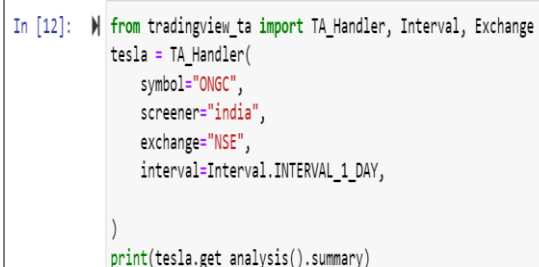
In [8]: M print(analysis)

{'NYSE:DOCN': <tradingview_ta.main.Analysis object at 0x0000024F073678B0>, 'NASDAQ:AAPL': <tradingview_ta.main.Analysis object at 0x0000024F0B53C440>, 'NASDAQ:TSLA': <tradingview_ta.main.Analysis object at 0x0000024F0B53C2B0>}
```

Figure 5.8

Then we need to type out an object for reference. Then we try to get the analysis for our desired stock of tesla and here we can see that we get a reply in form of JSON and it is very difficult to read it that ways.

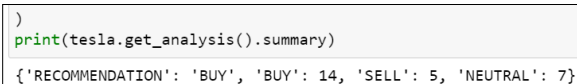
So we try to get a summary for analysis:



```
In [12]: M from tradingview_ta import TA_Handler, Interval, Exchange
        tesla = TA_Handler(
            symbol="ONGC",
            screener="india",
            exchange="NSE",
            interval=Interval.INTERVAL_1_DAY,
        )
        print(tesla.get_analysis().summary)
```

Figure 5.9

Here we first put our object that we made and filled in the desired information. Here we wanted to know recommendation for ONGC and now let's see the result.



```
)
print(tesla.get_analysis().summary)

{'RECOMMENDATION': 'BUY', 'BUY': 14, 'SELL': 5, 'NEUTRAL': 7}
```

Figure 5.10

Here we see that our recommendation is to 'BUY' which is very similar to our recommender. This is because we checked for NSE index and here ONGC is part of NSE index too. The only difference is off result ratios because our recommender works on 10 days of historic data for better and accurate results and this one works on historic data of previous 24 hours.

Let's discuss our flagship scripts now. We have these scripts in Pine language. Pine language is very helpful because pine language has its inspiration python and makes coding really easy.

5.3: Trading Vice Flagship Script

This script is totally for beginners. This is a very simple strategy which suits daily traders as well.

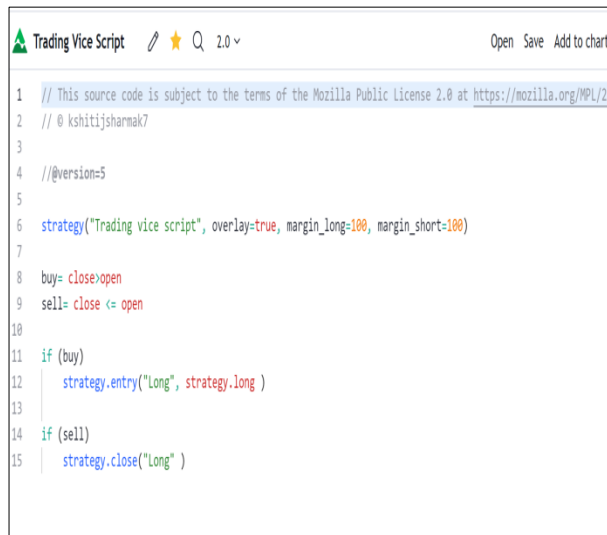


Figure 5.11

In this strategy first we are defining couple of things like margins and overlays. Here margin means the amount we have for trading. So if we exhaust the entire amount in our wallet then we will not be able purchase or sell any other stock. Hence margin is set to 100. 100 mean 100% of the value we have.

If the closing price of the preceding candle was higher than the starting price of the next candle in this case, we wish to initiate a buy call.

Similarly, if the closing price of the preceding candle was lower than or equal to the starting price of the succeeding candle, we wish to initiate a sell call.

So now let's test out this strategy on NSE index.



Figure 5.12

Here is the result.

This candle stick graph tells us on which intervals our script kicked in and made the calls for buy and sell orders.

Now let's see some metrics, whether we made any profit or not.



Figure 5.13

It is very evident that even though in a slow moving market we were able to make profit and our profit factor was 1.071 and profit percentage of 38.15%.

Similarly we have tested same strategy on BSE index too.



Figure 5.14



Figure 5.15

Here it is quite evident that our net profit is much better and profit factor is at 1.11 and profit percentage is at 38.01%.

Now let's move on to next strategy. This strategy will be more technically advanced and is meant for basically intermediate or pro traders.

5.4: Pull Back Strategy

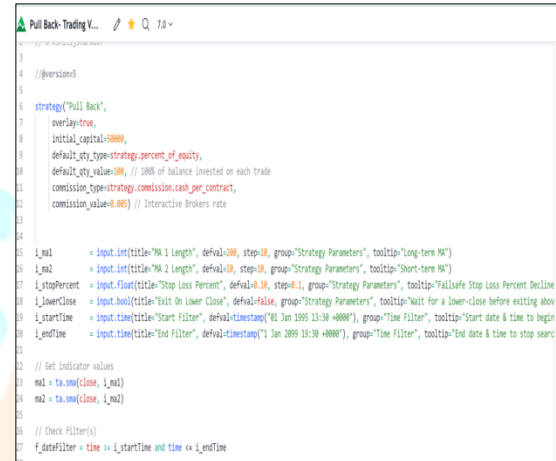


Figure 5.16

Here we will be first declaring our capital and commission that a broker will ask for.

Then we are declaring values for margins, starting time and end time and for stoploss and configurations for lower close.

Then we will get indicator values and put time filter.

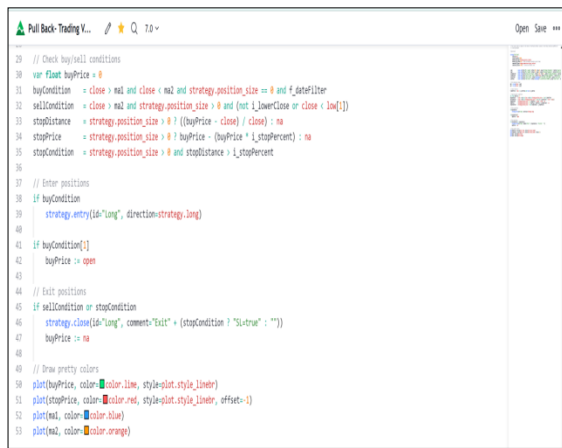


Figure 5.17



Figure 5.19

Now we will specifying buy and sell conditions with all the necessary stop conditions. After this we will put in our configurations for entry positions and exit positions.

Lastly we want some feedback from the plot so we specify different parameters with different colours.

Let's test out this strategy on Bitcoin/US Dollar Chart.

It is quite evident that we are able retain a profit percentage of 60.94% which is pretty huge and a profit factor of 1.148.

Let's check this strategy and make sure it runs good on Indian Indices too.



Figure 5.18

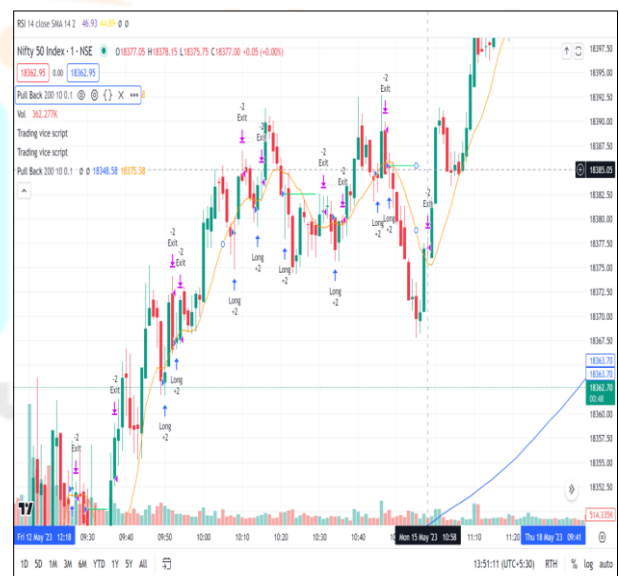


Figure 5.20



Figure 5.21

This strategy has proven to be a boon and for our Indian Index too. With net profit over 1207.82rs, this tradeshot has been very successful. Profit percent is also high at 76.18% with profit factor 1.392, we are very sure that we have booked in some good trades.

Chapter 6: Web Deployment

For people to access our product we will be developing a GUI. More importantly in this section we will discuss all the technologies used for our scripts and web app.

So let's first discuss for our scripts.

PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics, and it was developed by Guido van

Rossum. Python is used for everything, including developing websites, testing software, and machine learning. It may be used by both developers and non-developers. Python has attracted a lot of interest in the areas of modern software development, infrastructure management, and notably in data science and artificial intelligence. Python is a widely used language that may be used in a variety of applications and can operate on virtually all system architectures.

Python-based open-source libraries are widely used nowadays, and each one has its own root source. A collection of previously assembled code scripts that can be used again to save time is known as a library. It's a collection of syntax, tokens, semantics. It includes the standard Python distribution.

For instance, you only need one line of code to produce visualizations. To create a chart from an item without the aid of a library like this would require writing a significant amount of code. Python is a well-liked option for data analysis because to its extensive toolbox for modifying, visualizing, and training machine-learning models. Here are some well-known Python libraries:

- Tensorflow
- Numpy
- Pandas
- keras

and many more



Figure 6.1

6.1: TensorFlow

TensorFlow, an open source, end-to-end platform for building machine learning applications. To execute machine learning & deep learning apps and other statistical and predictive data workloads, Google researchers built Tensor-Flow. It is a symbolic math toolbox that performs several dataflow and differentiable programming-based operations aimed towards deep learning training and inference. For users like data scientists, statisticians, and predictive modellers, it is designed to ease the use of process of creating and implementing advanced analytics applications.

Tensor & flow are the essential components of the term "tensorflow". "Tensors are multidimensional

arrays, and flow refers to the movement of data during an operation. In essence, it lets you design dataflow graphs and structures to describe the path that data takes through a graph. With the output displaying at the other end, it allows you to diagram the operations that may be performed on these inputs.

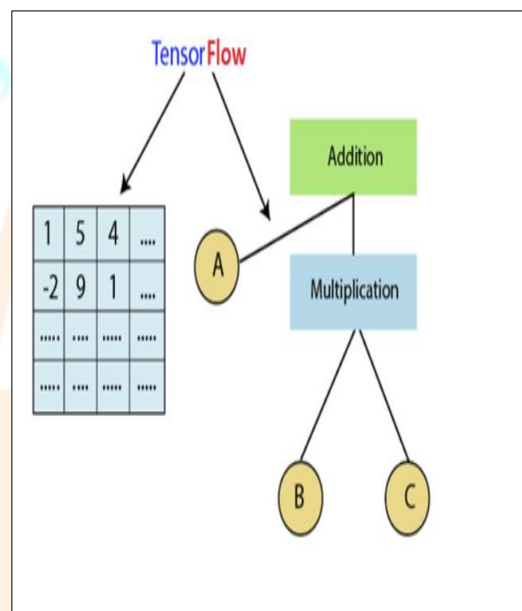


Figure 6.2

Example of TensorFlow: A user can get better refined search whenever a user try to search a particular word then google automatically suggest what could be the next things with the help of AI.

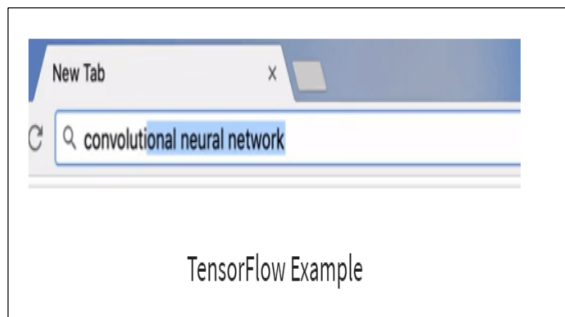


Figure 6.3

With help of machine learning, google uses their datasets to provide best and good experience to the user.

There are many use cases of tensorflow:

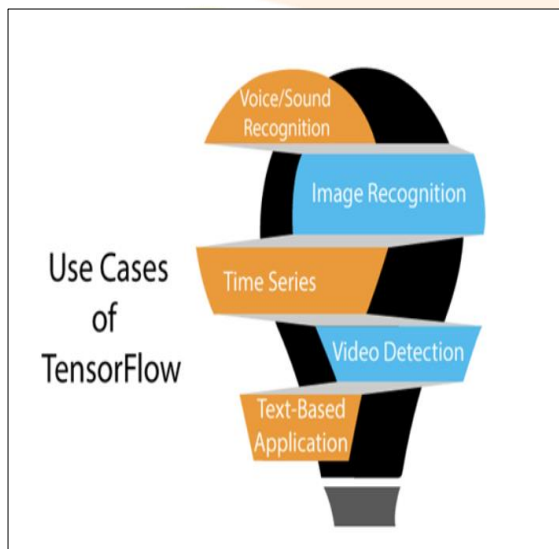


Figure 6.4

When compared to other well-known deep learning frameworks, TensorFlow offers incredible features and services. A multi-layered, large-scale neural network is built using TensorFlow.

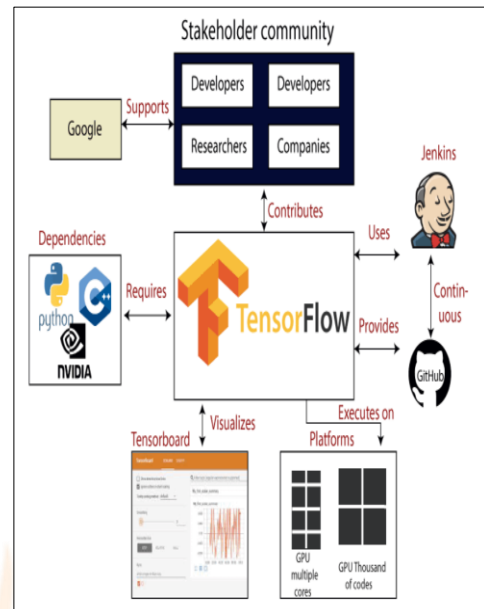


Figure 6.5

The majority of deep learning application cases include image recognition, speech recognition, video detection, and time series (to identify significant data). It is typically used to address categorization, perception, comprehending, discovering predictions, and creation problems in deep learning or machine learning.

Example of TensorFlow in python:


```
python
import tensorflow as tf
from tensorflow import keras

# Load the MNIST dataset
mnist = keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the input images
x_train = x_train / 255.0
x_test = x_test / 255.0
```

Figure 6.6

Basically in this we are training a cnn for img. Classification. Then we import some libraries and load required datasets.

```
python
# Define the model architecture
model = keras.models.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(10, activation='softmax')
])
```

Figure 6.7

Next, we define the CNN's architecture using Keras, a high-level TensorFlow API. To extract significant characteristics from the input photos, we build a sequential model and add layers like convolutional & pooling layers.

```
python
# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Figure 6.8

Once the model has been created, we can compile it by defining the evaluation metric, optimizer, and loss function.

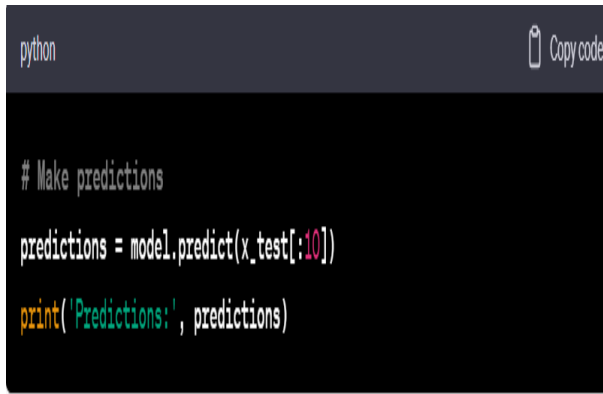
```
python
# Train the model
model.fit(x_train, y_train, epochs=5, batch_size=64)

# Evaluate the model
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

Figure 6.9

Then, using test data, the model's performance is evaluated after it has

been trained using training data.



```
python

# Make predictions

predictions = model.predict(x_test[:10])

print('Predictions:', predictions)
```

Figure 6.10

In the end, we can use the trained model to predict outcomes from fresh, untested data.

6.2: NumPy

For array manipulation, use the Python library "NumPy". NumPy offers a quick and efficient way to handle the massive amount of data. NumPy also makes it exceedingly easy to multiply matrices and manipulate data. NumPy is speedy, thus it makes sense to work with a lot of data. It also offers functions for working in the field of linear algebra as well as matrices and the Fourier transform. The following tasks may be completed by a developer using NumPy:

- Logical and mathematical operations on arrays.

- Fourier transforms and shape-manipulation techniques.
- Operations involving linear algebra.
- In NumPy, there are pre-built linear algebra and random number generation methods.



Figure 6.11

Numpy is very much popular. It is different from core python as shown below:

Core Python	Numpy
Python is an open-source interpreted high-level, general-purpose dynamically typed programming language developed by Guido van Rossum.	NumPy is an open-source library for the Python programming language developed by Travis Oliphant.
Object-oriented, imperative, procedural, and functional programming paradigms are supported by python. It has built-in containers which are lists, tuples, dictionaries, and sets.	Numpy supports built-in mathematical functions to operate on large multidimensional arrays and matrices.
Simple python list program : list=[1,2,"hi"] Print(list) Output: [1,2,'hi']	Simple Numpy program : import numpy as np arr=np.array([1,2,3]) print(arr) Output: [1,2,3]

Table 6.1

Series:

In the fields of data-science and artificial-intelligence, it is particularly well-liked and frequently utilized. Numpy is utilized by numerous libraries like Scipy, Matplotlib, Tkinter, and Pandas.

The Pandas Series is a type of labelled array that is just one dimension. They are capable of transporting any type of data, including Python objects, text, floats, and integers. The term "indexes" refers to all of the axis labels together.

6.3: Pandas*Figure 6.12*

ser=pd.Series([Name])	ser=pd.Series([Team])	ser=pd.Series([Number])
Name	Team	Number
0 Avery Bradley	a Boston Celtics	A100 0.0
1 John Holland	b Boston Celtics	B101 30.0
2 Jonas Jerebko	c Boston Celtics	C103 8.0
3 Jordan Mickey	d Boston Celtics	D104 NaN
4 Terry Rozier	e Boston Celtics	E105 12.0
5 Jared Sullinger	f Boston Celtics	F106 7.0
6 Evan Turner	g Boston Celtics	G107 11.0

Table 6.2

Working with "relational" or "labelled" data should be straightforward and natural thanks to a Python package/module called pandas, which provides rapid, adaptive, and expressive data structures. It aims to serve as the fundamental, high-level building block for utilizing Python for real, practical data analysis. Given that Pandas is an open source library, anyone can access its source code and provide feedback through pull requests.

In general, Pandas offers two data structures for data manipulation, namely: Series & Data Frame.

DataFrame:

A "Pandas DataFrame" is a two-dimensional, size-mutable, possibly heterogeneous tabular data structure with named axes (rows and columns). A data frame is a two-dimensional data structure where data is organised in rows and columns.

	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Table 6.3

Large data sets may be examined with the help of Pandas, and conclusions based on statistical principles can be reached. Disorganised data sets may be organised with Pandas, which makes them understandable and valuable. Relevant data is essential in data science.

Data of all types, including unlabeled data and both ordered and unordered time series, are ideally suited for Python pandas.

- Additional observational or statistical data sets of any kind

Due to its extensive use in numerous domains and integration with other scientific libraries, NumPy is crucial for accelerating procedures for data analysis, scientific computing, and machine learning.

Chapter 7: Pine Script

An Overview of the Scripting Language Used by TradingView

7.1: Introduction

TradingView, a well-known digital graphing and trading system, established the exclusive language for scripting known as Pine Script. It was created with TradingView's unique metrics, techniques, and signals in consideration. We shall examine Pine Script's main attributes, grammar, and functionalities in the following section.

7.2: Salient Features of Pine Script

- Easily comprehensible Pine Script's structure is comparable to JavaScript's and is extremely simple to comprehend and study, especially for complete novices. The technical evaluation is performed using a variety of integrated features, managers, and factors, and unique indications are produced.
- Pine Script has a broad variety of integrated markers and algorithms that may be employed to analyze market information, compute moving

averages, create trend patterns, and more. The procedure of putting sophisticated trading methods and signals into practice is made simpler by these integrated characteristics.

- Visualization that can be customized by setting graph forms, colors, and patterns is possible with Pine Script. Utilizing their choice of visual appeal, dealers may use this tool to illustrate their indicator setups and tactics on the TradingView interface.
- Alerts and reminders: Pine Script enables the development of personalized notifications depending on particular circumstances or occurrences. Whenever particular requirements are satisfied, including a trend line crossing or a breakaway sequence, dealers can configure indicators to get information by text message, email, or via their TradingView portal.

7.3: Syntax and Example

Pine Script's syntax is comparable to that used in different coding languages. Let's examine a straightforward average movement bridging example:

```
//@version=4
study(title="Moving
Average Crossover",
shorttitle="MA Crossover",
overlay=true)
```

```
// Define the input
parameters
fast_length = input(9, "Fast
MA Length")
slow_length = input(21,
"Slow MA Length")
```

```
// Calculate the moving
averages
fast_ma = sma(close,
fast_length)
slow_ma = sma(close,
slow_length)
```

```
// Plot the moving averages
on the chart
plot(fast_ma,
color=color.blue, title="Fast
MA")
plot(slow_ma,
color=color.red, title="Slow
MA")
```

```
// Generate a buy signal
when the fast MA crosses
above the slow MA
buy_signal =
crossover(fast_ma, slow_ma)
```

```
// Generate a sell signal when
the fast MA crosses below
the slow MA
sell_signal =
crossunder(fast_ma,
slow_ma)
```

```
// Plot buy and sell signals on
the chart
plotshape(buy_signal,
title="Buy          Signal",
location=location.belowbar,
color=color.green,
style=shape.triangleup)
plotshape(sell_signal,
title="Sell          Signal",
location=location.abovebar,
color=color.red,
style=shape.triangledown)
```

The sma() tool is used to determine the moving averages within this example's average movement cross approach, as well as the crossover() and crossunder() methods are used to produce signals for buying and selling. Moving averages are displayed on a graph using the plot() operation, and signals for buying and selling are plotted as forms using the plotshape() method.

Now let's dive deeper into web application.

Chapter 8: JavaScript

A Potent Web Development Language

8.1: Introduction

One of the foundational elements of the World Wide Web, JavaScript, is a programming language that is employed by 98% of websites on the client side to control how their webpage behaves.

The main application of the dynamic programming language JavaScript is the creation of websites. It enables programmers to include interactive components, change the content of web pages, and produce engaging user interfaces. We will examine JavaScript's main features and functions, its use in web development, and some recommended practices for producing effective and clean JavaScript code in this paper.

8.2: Overview of JavaScript

Brendan Eich developed JavaScript, commonly referred to as JS, in 1995. All contemporary web browsers are compatible with this high-level, interpreted programming language. Because it is lightweight and simple to use, JS is a preferred option for client-side scripting on web sites.

The initial name of JavaScript was LiveScript, but Netscape changed it to JavaScript, maybe in reaction to the excitement that Java was generating. The language's core is present in several browsers, including Netscape, IE from Microsoft, and others. Values are manipulated by JavaScript programmes, and each value is a member of a type. Seven primitive types are provided by JavaScript:

- **Number:** used for all integer and floating point number values, excluding very large numbers.
- **BigInt:** used for integers of any size.
- Text is stored in **strings**.
- **Boolean** values of true and false are frequently employed in conditional reasoning.
- **Symbol:** used to generate distinct, non-overlapping identities.
- **Undefined:** a sign that no value has been given to a variable.
- **Null:** signifying a purposeful non-value.

The *numerical operators* available in JavaScript include +, -, *, /, % (remainder), and ** (exponentiation). = is used to assign values. Compound assignment counterparts for each binary operator exist as well, such as += and -=, which go all the way to the x = x operator y.

The C family's grammar is quite similar to JavaScript's. A few things to be pointed out:

- Identifiers are not permitted to be reserved words, but they

may contain Unicode characters.

- Comments are typically written in the form of // or /* */.
- In JavaScript, semicolons are not mandatory; the language will automatically insert them as necessary. Since semicolons are still used in the syntax, unlike Python, there are some restrictions to be aware of.

8.3: JavaScript's Function in Web Development

JavaScript is a vital component of web development because it gives programmers the ability to improve the functionality and interactivity of web sites. It has numerous capabilities and APIs that enable many different things, like real-time updates, form validation, data manipulation, and more. Here are a few typical situations when JavaScript is used in web development:

- **DOM Manipulation:**

A web page's HTML elements are shown as a hierarchical tree structure in the DOM. Every component of the tree is referred to as a node, and JavaScript has strong methods to access and modify these nodes. Accessing and changing different DOM components, such as element

properties, attributes, and content, is known as DOM manipulation. The following are some typical methods for DOM manipulation in JavaScript:

- **Accessing Elements:**

A number of ways are offered by JavaScript to choose elements within the DOM, such as:

- Gets an element by its specific identifier using the **getElementById** command.
- **getElementsByName:** Returns a group of elements that belong to a given class.
- **getElementsByTagName:** Returns a group of elements that have the specified tag name.
- **querySelector:** Chooses the first element that a CSS selector matches.

- **Changing element content:**

The ability to dynamically change an element's content is provided by JavaScript once it has been selected.

Finally, JavaScript's DOM manipulation features give programmers the ability to design dynamic, interesting, and interactive web sites. Any web developer who

wants to fully utilize JavaScript when creating contemporary online apps must understand and grasp the concepts of DOM manipulation.

- **Event Handling:**

A key component of JavaScript programming is event handling, which enables programmers to create interactive and responsive web applications. Events are deeds or happenings that take place on a web page, such as mouse clicks, keyboard inputs, or touch movements from users. JavaScript offers reliable means for recording and reacting to these events, allowing programmers to design dynamic and interesting user interfaces.

To ensure the best performance, it is crucial to manage events effectively. Avoid using too many event listeners or complicated event handling logic if you don't want to affect performance. Furthermore, managing event handling in big apps can be facilitated by comprehension of event propagation and delegation.

- **Form Validation:**

In order to guarantee that data submitting by users is precise, full, and in the desired format, form validation is a crucial component of web development. JavaScript offers robust facilities for form input

validation and fast feedback from users. Developers of JavaScript may improve the user's experience, reduce mistakes, and increase accuracy of data by integrating form validation.

JavaScript programmers can design reliable and easy to use online forms by including form validation. It's crucial to remember that validation on the server must constantly be used in addition to client-side validation. As an insurance, server-side validation stops any possibility of manipulation or circumvention of validation on the client's side.

- **AJAX and Fetch:**

Both the Fetch API and Ajax (Asynchronous JavaScript and XML), two potent JavaScript capabilities that allow asynchronous interaction with server infrastructure, allow programmers to obtain and transmit data without necessitating a web page to refresh. These methods have transformed the creation of websites by improving the customer experience and producing dynamic, engaging web apps.

descriptive; they should simply explain what is happening.

- It is a terrible idea to use global names for variables. This is due to the fact that every JavaScript script on the page only runs within a single scope. If our programme contains global variables or functions, they will be overwritten if any scripts contained after ours use the identical variable and function names.
- Code that is clear and acceptable is simpler to pass on to new programmers, less complicated to amend, and more secure.
- Avoid unnecessary comments to make the code page less crowded and neater.
- Try not to use too much nesting so as to keep the code simple and easy to follow for new developers.

8.4: JavaScript Coding Best Practices

- Appropriate variable names ought not to be more or less

In conclusion, the building of fluid, interactive internet pages has been revolutionized by JavaScript, a strong language. Because of its

adaptability and broad compatibility throughout internet browsers, programmers must be proficient in it. Programmers may fully utilize JavaScript and build compelling online experiences by adhering to best practices and composing clear, concise code.

loop, it can effectively manage several requests at once. It makes use of unrestricted input/output (I/O) processes, enabling it to execute a number of requests concurrently with no pausing for each individual to finish. This enhances both speed and capacity.

Chapter 9: Node.js

An Introduction to Server-Side JavaScript

9.1: Introduction:

Programmers can run JavaScript scripts on the server-side using Node.js, a robust freely available JavaScript runtime framework. Due to its effectiveness, adaptability, and capacity for managing multiple simultaneous demands, it has been quite successful over the past few years. We will examine the main benefits, characteristics, and applications of Node.js in this paper.

- The V8 JavaScript engine that is additionally utilized by the Chrome browser is the foundation upon which Node.js is based. V8 turns JavaScript to machine-readable code so that Node.js can run scripts at breakneck rates.

- Node.js includes NPM, a capable package manager that gives users access to a sizable infrastructure of open-source libraries and packages. NPM makes it easier to include external libraries within Node.js programs enabling programmers to use pre-existing technologies and hasten the building cycle.

9.2: Salient Features of Node.js:

- Asynchronous and Non-Blocking: Because Node.js uses a single-threaded event
- Node.js has an event-based design in which actions or occurrences call off particular functions or responses. This method enables programmers

to create code that successfully finishes asynchronous tasks and reacts to events.

9.3: Basic Node.js functionality

- Simple HTTP Server:

```
const http = require('http');

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World!');
});

server.listen(3000, 'localhost', () => {
  console.log('Server running at http://localhost:3000/');
});
```

Figure 9.1

```
const fs = require('fs');

// Reading a file
fs.readFile('file.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});

// Writing to a file
fs.writeFile('file.txt', 'Hello, World!', (err) => {
  if (err) throw err;
  console.log('File written successfully!');
});
```

Figure 9.2

- Express.js Web Application Framework:

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.send('Hello, World!');
});

app.listen(3000, () => {
  console.log('Server running at http://localhost:3000/');
});
```

Figure 9.3

- File System Operations:
- Asynchronous Function with Promises:

```
function delay(ms) {
  return new Promise((resolve) => {
    setTimeout(resolve, ms);
  });
}

async function asyncFunction() {
  console.log('Before delay');
  await delay(2000);
  console.log('After delay');
}

asyncFunction();
```

Figure 9.4

```
const fs = require('fs');

function processFile(file) {
  return new Promise((resolve, reject) => {
    fs.readFile(file, 'utf8', (err, data) => {
      if (err) {
        reject(err);
      } else {
        // Perform desired operations with the file data
        // ...
        resolve(data);
      }
    });
  });
}

async function processFiles(files) {
  for (const file of files) {
    try {
      const data = await processFile(file);
      console.log('File processed:', data);
    } catch (error) {
      console.error('Error processing file:', error);
    }
  }
}

const files = ['file1.txt', 'file2.txt', 'file3.txt'];
processFiles(files);
```

Figure 9.6

- Handling HTTP POST Requests with Express.js:

```
const express = require('express');
const app = express();

app.use(express.json());

app.post('/api/users', (req, res) => {
  // Process the received JSON data
  const { name, email } = req.body;

  // Save the user to the database or perform any desired action
  // ...

  res.status(201).json({ message: 'User created successfully' });
});

app.listen(3000, () => {
  console.log('Server running at http://localhost:3000/');
});
```

Figure 9.5

- Performing Asynchronous Operations in a Loop:

9.4: Advantages of Node.js

- Superior Efficiency: Node.js's event-based design and unrestricted input/output approach both greatly enhance its efficiency. Developing real-time apps that need immediate data changes, like messaging apps or teamwork tools, is where it excels.
- Ability to scale: Node.js is extremely scalable due to its capacity to manage multiple queries at once. It is a great option for programmes with significant traffic because it efficiently handles a lot of links while not using up a lot of the computer's resources.

- Node.js allows programmers to utilize JavaScript both on the client's and the server-side. This makes it easier to move across languages, allowing for script sharing, and speeds up the software creation process.
- Vast and Dynamic Ecosystem: Node.js has a thriving programmer community that proactively contributes to its development. The community ensures ongoing advancement and creativity by offering a multitude of materials, lessons, and assistance.

good fit for it because of its small footprint, capacity, and capability to interact with different databases and online resources.

- Command-Line Utilities: Node.js is frequently employed to create conveniences and command-line tools. It is a well-liked option for process automation, connecting with computer resources, and developing programs due to its simplicity of use, multi-platform interoperability, and huge network of libraries and extensions.

9.5: Use Cases of Node.js

- Node.js is a popular programming language for creating apps for the web. It is most suited for applications that need constant revisions, including social networking platforms, applications for teamwork, and services for streaming, because to its event-based design and capacity for handling multiple simultaneous queries.
- Node.js is a great option for creating microservices and APIs. Developing strong backend infrastructure is a

- Internet of Things (IoT): Node.js is suited for creating IoT apps due to its effectiveness and minimal weight. It is a favored framework for creating robust and dynamic IoT applications because of its unrestricted input/output style, which allows for smooth actual time connection with IoT gadgets.

9.6: Options for Hosting and Deploying Node.js Applications

Developers have a wide range of alternatives for deploying and hosting Node.js apps. The decision is based on variables such needed

flexibility, implementation difficulty, and financial constraints. Let's examine some popular Node.js project hosting and deploying choices.

- **Self-Hosting:**

Establishing and maintaining your individual server framework is part of self-hosting. This choice offers full oversight of the implementation environment and enables modification in accordance with unique requirements. It is appropriate for businesses with the means and know-how to supervise and maintain servers. Traditional servers or cloud-based services like AWS, Microsoft Azure, or GCP can be used for this purpose.

- **Platform-as-a-Service (PaaS):**

By abstracting off handling server duties, PaaS enables engineers to concentrate on the creation of applications. Such systems frequently include expanding, load distribution, and tracking capabilities integrated in. PaaS is appropriate for programs that need simple implementation, expansion, and an efficient hosting environment.

- **Containerization:**

Applications written in Node.js are increasingly being deployed using containerization technologies like Docker. Applications and their dependencies are contained within vessels, guaranteeing compatibility between different types of environments. Technologies for container automation, like Kubernetes, offer sophisticated tools for expanding controlling, and observing containers in a distributed framework. Application deployment is facilitated by containerization's adaptability and portability spanning a range of hosting settings, encompassing cloud-based services and physical infrastructure.

- **Content Delivery Networks (CDNs):**

Node.js applications can leverage CDNs like Cloudflare, Akamai, and Fastly to temporarily store and deliver static data including JavaScript on the client side scripts, the Hypertext Markup and CSS. The speed and accessibility of online applications are enhanced by CDNs by delivering static material from edge sites all over the globe. A sustainable and internationally dispersed architecture can be created by combining CDNs with various deploying methods.

Chapter 10: GUI Coding

Graphical user interface is really very important.

We will be developing our GUI with help of technologies listed above. Considering that we are not making a web application, this web application has not intent to be front-end heavy. We will be coding for minimal frontend and robust back end.

So let's start with our HTML first. Here's a twist, we will be using ejs because we need templates for our web application so let's do that first.

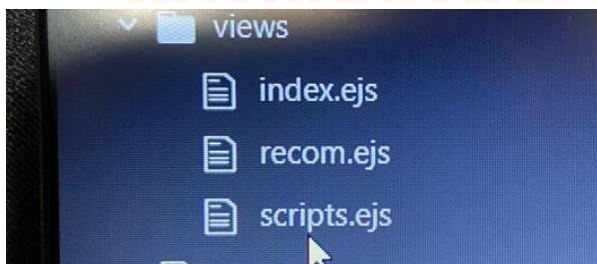


Figure 10.1

We have to make all the ejs files inside views folder. It is normal convention. Let's dive into details of this code.

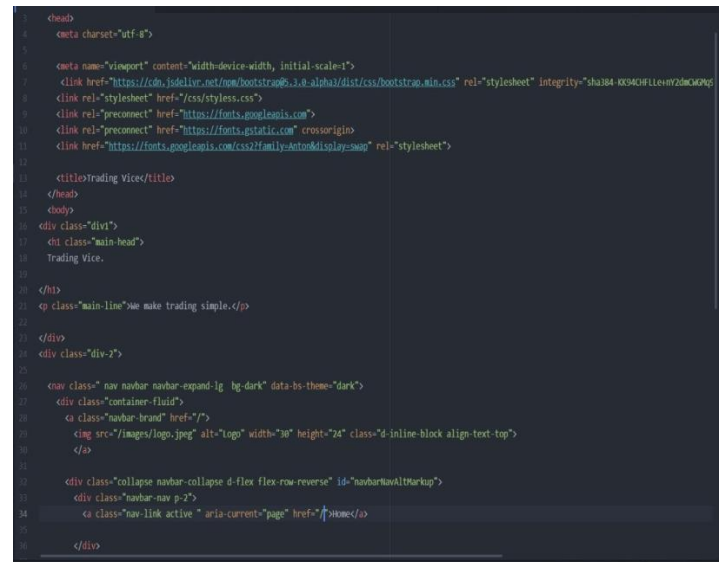


Figure 10.2

Here we can see first we importing all the CDNs through links like bootstrap, google fonts and our style sheets.

Then we have started the main functional code, first we have placed a div with a big heading and then we putting a navbar and underneath that we our links for our recommender systems and Trading Scripts.


```

<nav class="nav navbar navbar-expand-lg bg-dark" data-bs-theme="dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="/">
      
    </a>

    <div class="collapse navbar-collapse d-flex flex-row-reverse" id="navbarSupportedContent">
      <div class="navbar-nav p-2">
        <a class="nav-link active" aria-current="page" href="/">Home</a>

        </div>
      </div>
    </div>
  </nav>

  <div id="trading">
    <h3>Trading Vice Flagship Strategy</h3>
    <p></p>
  </div>

  <div id="pull">
    <h3>Pull Back Strategy</h3>
    <p></p>
  </div>

  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-4B00820047535924827b8530E331672821716527267d9676082417f863044c"
    </script>
</html>

```

Figure 10.5

```

1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3   <head>
4     <meta charset="utf-8">
5
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7
8     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-rc3G9gyAsnp83W0Y/Cl0x+3588+JXWw9pLb0KG9cH5850qFl+GyL3x9nOp7h92">
9     <link rel="stylesheet" href="/css/style.css">
10    <link rel="preconnect" href="https://fonts.googleapis.com">
11    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
12    <link href="https://fonts.googleapis.com/css?family=Anton&display=swap" rel="stylesheet">
13
14    <title>Trading Vice</title>
15  </head>
16  <body>
17    <div class="div1">
18      <div class="main-head">
19        Stock Recommender System
20      </div>
21
22      <div>
23        <div class="main-line">Xoed for better.</div>
24
25        <div>
26          <div class="div2">
27
28            <div class="nav navbar navbar-expand-lg bg-dark data-bs-theme="dark">
29              <div class="container-fluid">
30                <a class="navbar-brand" href="/">
31
32                  
33                  </a>
34
35                <div class="collapse navbar-collapse d-flex flex-row-reverse justify-content-between">
36                  <div class="navbar-nav p-2">
37                    <a class="nav-link active" aria-current="page" href="/">Home</a>
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Figure 10.6

Here we can see that clearly.

Now here we will be coding our recommender system page.

Here we have done exactly the same thing as we have done in `script.ejs`.

Here we can see that-

```

25
26 <nav class="nav navbar navbar-expand-lg bg-dark data-bs-theme="dark">
27   <div class="container-fluid">
28     <a class="navbar-brand" href="/">
29       
30     </a>
31
32     <div class="collapse navbar-collapse d-flex flex-row-reverse" id="navbarNavAltMarkup">
33       <div class="navbar-nav p-2">
34         <a class="nav-link active" aria-current="page" href="/">Home</a>
35
36         </div>
37       </div>
38     </div>
39   </nav>
40
41   <div><div>Python Recommender</div>
42   <p></p>
43   </div>
44
45   <div><div>Trading View Recommender</div>
46   <p></p>
47   </div>
48
49   </div>
50
51
52
53 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-Elj48x38S38Vwz33vwIQozPk6R3GeyBAnDj9763cY7A6c2Di50BBAb28T9r"
54
55
56 </body>
57 </html>
58

```

Figure 10.7

We have started the main functional code, first we have placed a div with a big heading and then putting a navbar and underneath that we our scripts for recommender systems.

The most import thing we have to do is initiate npm and install NPM packages. With this being done we can initiate our project.

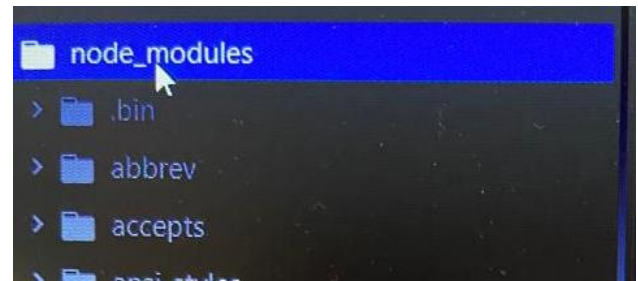


Figure 10.10

```

1 {
2   "name": "y",
3   "version": "1.0.0",
4   "lockfileVersion": 3,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "y",
9       "version": "1.0.0",
10      "license": "ISC",
11      "dependencies": {
12        "body-parser": "^1.20.2",
13        "ejs": "^3.1.9",
14        "express": "^4.18.2",
15        "nodemon": "^2.0.22"
16      },
17    },
18    "node_modules/abbrev": {
19      "version": "1.1.1",
20      "resolved": "https://registry.npmjs.org/abbrev/-/abbrev-1.1.1.tgz",
21      "integrity": "sha512-nne9/IiQ/hzthV6pDnBt7DjPtkrV00P/zvPSm5pOfkl6xuGnXn/
22    },
23    "node_modules/accepts": {
24      "version": "1.3.8",
25      "resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.8.tgz",
26      "integrity": "sha512-9a7a67a7f7a709eb71a6c36fda9bf1a5eef3a512-
27      "dependencies": {
28        "mime-types": "~2.1.34",
29        "negotiator": "0.6.3"
30      },
31    },
32    "engines": {
33      "node": ">= 0.6"
34    }
35  },

```

Figure 10.8

Now we will add our stylesheet for all the styling we require.

```

1 body{
2   background-color: #212A3E;
3   overflow-x: hidden;
4 }
5
6
7 .div1{
8   text-align: center;
9   height:300px;
10  background-color: black;
11  opacity: 75%;
12  border-radius: 3rem;
13  font-family: 'Anton', sans-serif;
14  margin-top:50px;
15 }
16
17 .main-head{
18
19  color: white;
20  font-size: 100px;
21  opacity: 100%;
22  position: relative;
23  top:60px;
24
25 }
26 .main-line{
27   font-size:30px;
28   color:black;
29   position: relative;
30   top:200px;
31 }
32
33 .div-2{
34   position:relative;

```

Figure 10.11

```

1 {
2   "name": "y",
3   "version": "1.0.0",
4   "description": "",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "",
10  "license": "ISC",
11  "dependencies": {
12    "body-parser": "^1.20.2",
13    "ejs": "^3.1.9",
14    "express": "^4.18.2",
15    "nodemon": "^2.0.22"
16  },
17  "engines": {
18    "node": ">= 0.6"
19  }
20 }

```

Figure 10.9

```

23 top:60px;
24
25 }
26 .main-line{
27   font-size:30px;
28   color:black;
29   position: relative;
30   top:200px;
31 }
32
33 .div-2{
34   position:relative;
35   top:150px;
36
37   height:200px;
38   background-color: black;
39   text-align: center;
40
41 }
42
43 button{
44   color:white;
45 }
46
47 .products{
48   margin: 10px 0 10px 0;
49   color:white;
50 }
51
52 h3{
53   color:white;
54   font-family: 'Anton', sans-serif;
55 }
56

```

Figure 10.12

```

1 const express= require("express");
2 const bodyparser = require("body-parser");
3 const ejs= require("ejs");
4
5 const app = express();
6
7 app.set ('view engine', 'ejs');
8
9 app.use(bodyparser.urlencoded({
10   extended:true
11 }));
12
13 app.use(express.static("public"));
14
15
16 app.get("/", function(req,res){
17   res.render("index")
18 })
19
20 app.get("/script", function(req, res){
21   res.render("scripts")
22 })
23
24 app.get("/recom", function(req, res){
25   res.render("recom");
26 })
27
28
29
30
31 app.listen(3000, function(){
32   console.log("server is up and hot")
33 });
34

```

Figure 10.13

Now it's time to code the brains of the app. We will be naming this file as App.js

Here we have to import npm packages from node modules folder. We will be importing ejs, express and body-parser as of now. Express helps us to connect with server. Ejs helps us with templating and body-parser will help us with manage all get and post requests.

Here we can see that-

Here we can see that we have set up the port first for our localhost server. And here we are getting all the get requests for our web application.

And with a simple command of “node app.js”, our app is up and running on port 3000.

Chapter 11: Web Results

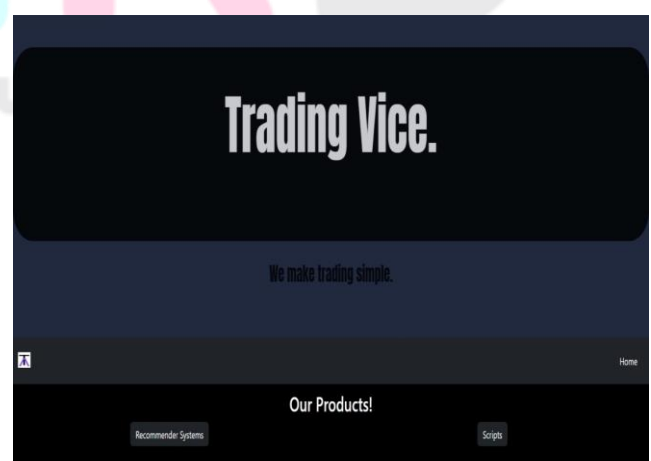
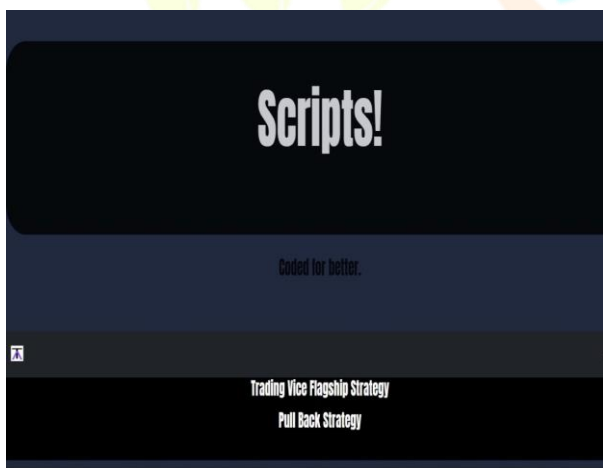


Figure 11.1*Figure 11.2**Figure 11.3*

Chapter 12: Results

Trading Vice has proven to be a really good recommender system and a script source for profitable trading. Our results are promising, and we can indulge in active trading with our clients.

Chapter 13: Conclusion

Trading Vice is now a full tech platform driven and maintained by engineers trying to pioneer new skills in the domain of trading. With simple and automated trading systems people can allow themselves to indulge in intelligent and profitable trading methods. We provide solutions to both, to a rookie and to a pro player. We are delighted to put up our solutions and help people.

Trading Vice, We make trading simple.

Chapter 14: References

Mathur, Medha & Mhadalekar, Satyam & Mhatre, Sahil & Mane, Vanita. (2021). Algorithmic Trading Bot. ITM Web of Conferences. 40. 03041.

10.1051/itmconf/20214003041.

Çubukçu Çerasi, Ceren & KIR, Tunahan & INCE, Doguhan. (2022). Proposition of a Trading Bot for Cryptocurrency Market (KRİPTO PARA PİYASASI İÇİN ALIM SATIM BOTU ÖNERİSİ). SOCIAL SCIENCE DEVELOPMENT JOURNAL. 7. 237-243. 10.31567/ssd.726.

Roy, Gregory & Fiaidhi, Jinan & Mohammed, Sabah. (2022). Multi-Timeframe Algorithmic Trading Bots Using Thick Data Heuristics with Deep Reinforcement Learning.

Artificial Intelligence Evolution. 107-159. 10.37256/aie.3220221722.

Peng, Aisha & Ang, Sau & Lim, Chia. (2022). Automated Cryptocurrency Trading Bot Implementing DRL. Pertanika Journal of Science and Technology. 30. 2683-2705. 10.47836/pjst.30.4.22.

