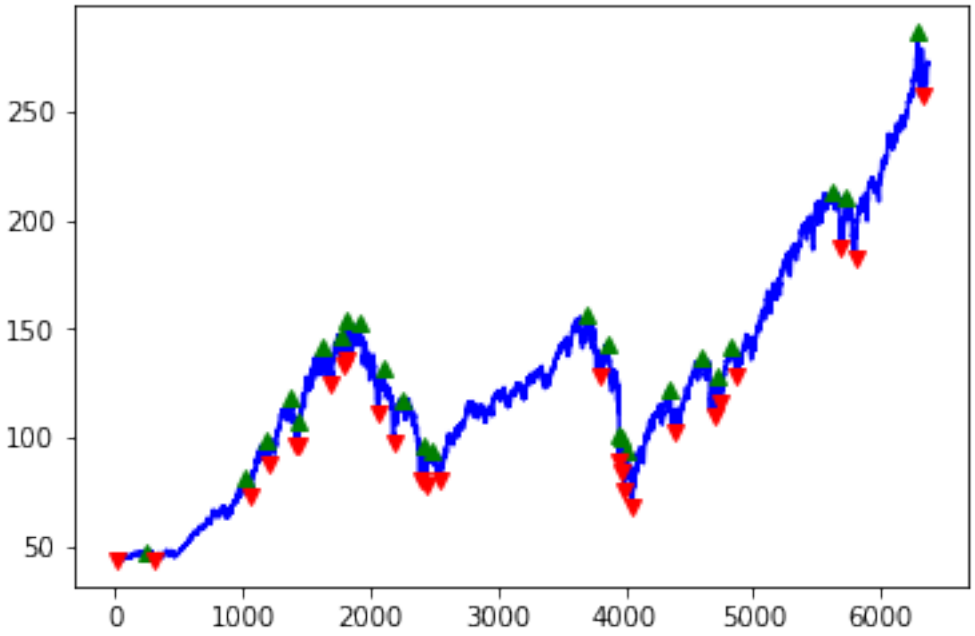
AUTOMATED TECHNICAL INDICATOR

KEY TERMS :

- Perpetually Important Points
- Pattern
- Technical Analysis

➢ Perpetually Important Points

The sets of local maxima and minima for a given time series such that the each minima is atleast R (compression rate) times less than the previous and next maxima. Similarly, each maxima is atleast R (compression rate) times more than the previous and next minima.



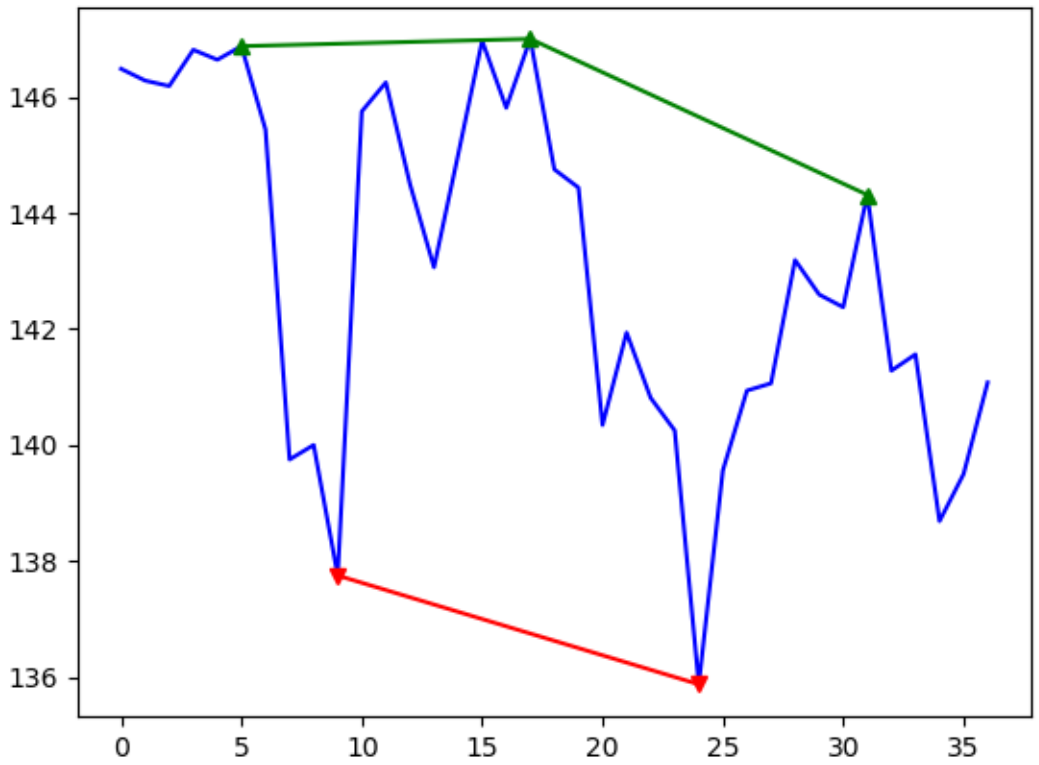
Green : Important Point (Maxima)

Red : Important Point (Minima)

*not all local maxima and minima are marked(only important points)

➢ Pattern

Pattern is a geometric shape formed by a sequence of **important points** which helps in interpreting the movement of stock prices. Each pattern has its own specific characteristics and a well-defined significance.

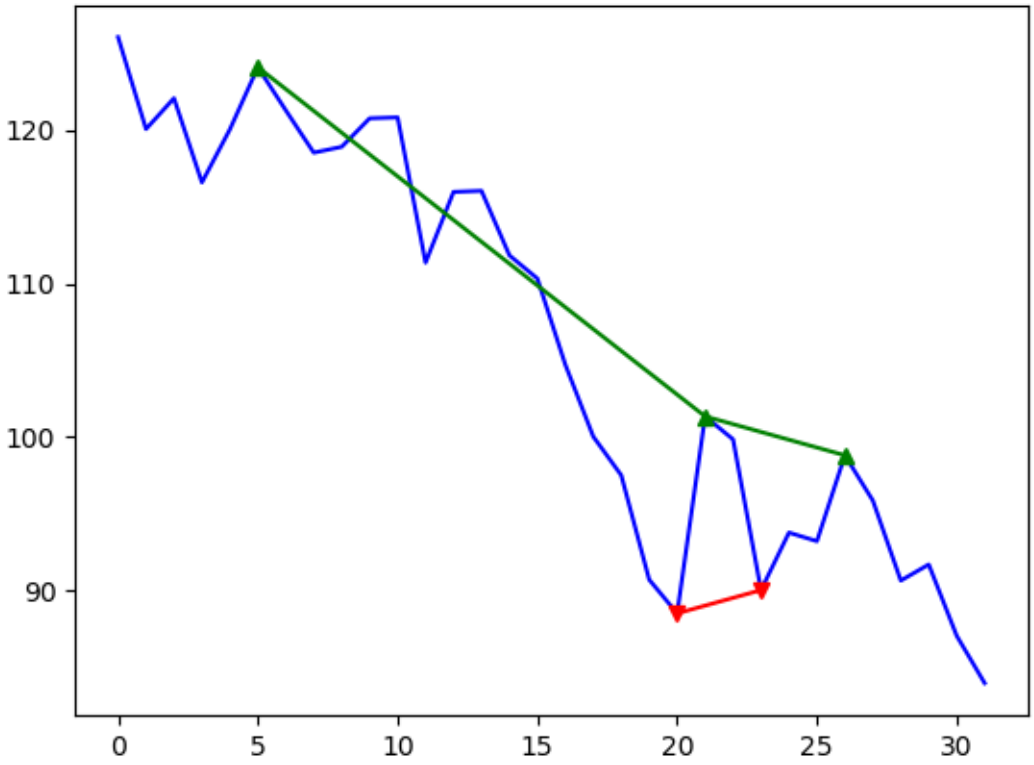


Result of Head-and-Shoulder pattern search

- Technical Analysis

The technical indicators like changepoints, perpetually important points, etc are used to analyze changes in the distribution property of the given data. Using these technical indicators we will analyze and search through the series for existence of different **patterns**.

In the undertaken project, we will also consider the guessing which is basically a approximating method of technical analysis. It works using price of given date, with a bump of one standard deviation, instead of last extrema.



Theory

- Important Point :

We compress a time series by selecting some of its minima and maxima, called important points, and dropping the other points. The intuitive idea is to discard minor fluctuations and keep major minima and maxima. We control the compression rate with a parameter R , which is always greater than one; an increase of R leads to selection of fewer points.

A point a_m of a series a_1 .... a_n is an important minimum if there are indices i and j ,where i<m<j , such that

- a_m is the minimum among a_i ..... a_j ,and
- $a_i/a_m > R$ and $a_j/a_m > R$

A point a_m of a series a_1 ,..., a_n is an important maximun if there are indices i and j ,where i<m<j , such that

- a_m is the maximum among a_i ..... a_j ,and
- $a_m/a_i > R$ and $a_m/a_j > R$

➢ Pattern

Head-and-Shoulders

Head-and-shoulders (HS) and inverted head-and-shoulders (IHS) patterns are characterized by a sequence of five consecutive local extrema E_1 , . . . , E_5 such that

$$
\text{HS} = \begin{cases} E_1 \text{ is a maximum} \\ \\ E_3 > E_1 \text{ , } E_3 > E_5 \\ \\ E_1 \text{ and } E_5 \text{ are within 1.5 percent of their average} \\ \\ E_2 \text{ and } E_4 \text{ are within 1.5 percent of their average} \end{cases}
$$

$$
\text{IHS} = \begin{cases} E_1 \text{ is a minimum} \\ \\ E_3 < E_1 \text{ , } E_3 < E_5 \\ \\ E_1 \text{ and } E_5 \text{ are within 1.5 percent of their average} \\ \\ E_2 \text{ and } E_4 \text{ are within 1.5 percent of their average} \end{cases}
$$

Broadening

Broadening tops (BTOP) and bottoms (BBOT) are characterized by a sequence of five consecutive local extrema E_1 , . . . , E_5 such that

$$
\text{BTOP} = \begin{cases} E_1 \text{ is a maximum} \\ \\ E_1 < E_3 < E_5 \\ \\ E_2 > E_4 \end{cases}
$$

$$
\text{BBOT} = \begin{cases} E_1 \text{ is a minimum} \\ \\ E_1 > E_3 > E_5 \\ \\ E_2 < E_4 \end{cases}
$$

Triangle

Triangle tops (TTOP) and bottoms (TBOT) are characterized by a sequence of five consecutive local extrema $E_1, \ldots, E_5$ such that

$$
\text{TTOP} = \begin{cases} E_1 \text{ is a maximum} \\ \\ E_1 > E_3 > E_5 \\ \\ E_2 < E_4 \end{cases}
$$

$$
\text{TBOT} = \begin{cases} E_1 \text{ is a minimum} \\ \\ E_1 < E_3 < E_5 \\ \\ E_2 > E_4 \end{cases}
$$

Rectangle

Rectangle tops (RTOP) and bottoms (RBOT) are characterized by a sequence of five consecutive local extrema E_1 , . . . , E_5 such that

$$\text{RTOP} = \begin{cases} E_1 \text{ is a maximum} \\ \\ \text{tops are within 0.75 percent of their average} \\ \\ \text{bottoms are within 0.75 percent of their average} \\ \\ \text{lowest top} > \text{highest bottom} \end{cases}$$

$$\text{RTOP} = \begin{cases} E_1 \text{ is a minimum} \\ \\ \text{tops are within 0.75 percent of their average} \\ \\ \text{bottoms are within 0.75 percent of their average} \\ \\ \text{lowest top} > \text{highest bottom} \end{cases}$$

Double

Double tops (DTOP) and bottoms (DBOT) are characterized by an initial local extremum E_1 and subsequent local extrema E_a and E_b such that

$$E_a \equiv \sup\{P^*_{t_k} : t^*_k > t^*_1, k = 2, \ldots, n\}$$

$$E_b \equiv \inf\{P^*_{t_k} : t^*_k > t^*_1, k = 2, \ldots, n\}$$

$$
\text{DTOP} = \begin{cases} E_1 \text{ is a maximum} \\ \\ E_1 \text{ and } E_a \text{ are within 1.5 percent of their average} \\ \\ t^*_a - t^*_1 > 22 \end{cases}
$$

$$
\text{DBOT} = \begin{cases} E_1 \text{ is a minimum} \\ \\ E_1 \text{ and } E_b \text{ are within 1.5 percent of their average} \\ \\ t^*_a - t^*_1 > 22 \end{cases}
$$

- Technical Analysis

Technical analysis, also known as "charting," has been a part of financial practice for many decades, but this discipline has not received the same level of academic scrutiny and acceptance as more traditional approaches such as fundamental analysis. One of the main obstacles is the highly subjective nature of technical analysis-the presence of geometric shapes in historical price charts is often in the eyes of the beholder. We will build a systematic and automated code to technical pattern recognition using technical indicators such as head-and-shoulders or double-bottoms.

METHOD (technical analysis) :

Repeat for $t = 1$ to $T- l - d +1$

- Identify all of the local extrema in the window $[t, t+l+d-1]$
- Proceed to check for the presence of the various technical patterns.

METHOD (guessing) :

- Identify all of the local extrema in the window **[t-l-d+1, t-1], t = current date**
- If no. of local extrema < total_req_extrema - 1 to define pattern, then we don't have sufficient extrema.
- Final pattern being all the identified extrema plus [price(t) + bump]

◉ Code

➢ Important Point :

The important points is a function taking **a** (time series with 'price element') and **R** (compression rate) as input. It returns a minmax list consisting of consecutive important points.

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt


def IMPORTANT_POINTS(a, R):
    minmax_list = []
    temp_1, temp_2, i = FIND_FIRST_TWO(a, R)
    minmax_list.append(temp_1)
    minmax_list.append(temp_2)
    n = len(a)
    if i < n and a[i-1] > a[0]:
        temp, i = FIND_MINIMUM(a, R, i)
        minmax_list.append(temp)
    while i < n:
        temp1, i = FIND_MAXIMUM(a, R, i)
        minmax_list.append(temp1)
        if i < n:
            temp2, i = FIND_MINIMUM(a, R, i)
            minmax_list.append(temp2)
    return minmax_list
```

Uses three functions:

- FIND_FIRST_TWO : To determine first two important points
- FIND_MINIMUM : To determine next important points(minima)
- FIND_MAXIMA : To determine next important points(maxima)

➢ Pattern :

- **Pattern.py :** A file defining class Pattern and attributes of the class.
- **pattern_library.py :** A file containing all the methods to modify patterns.
- **patterns.py :** A file with the methods to recognize patterns using IMPORTANT_POINTS() and conditions & logics in Pattern.

- Pattern.py :

It is a class file defining all the attributes. Apart from the the **__init__()** the class also has a bunch of simple but useful functions used for retrieving the information of these attributes. The main thing to built an automated technical analyzer, we'll need a function to read the input as string but evaluate the variables as if it is code line. We will use **eval()** method predefined in the python package. After evaluating the function compares the conditions and logics to recognize pattern.

```
def pattern_recog(self, data):
    condition = list(self.condition)                                    #condition = self.condition //results in referencing the list
    tolerance = self.tolerance
    j=0
    while j < self.no_of_condition:
        condition[j] = eval(condition[j])
        j += 1
    i=0
    while i < self.no_of_logic:
        if type(eval(self.logic[i])[0]) is np.bool_:                    #Output of eval(self.logic[i])[0] is of numpy.bool_ type
            compare = bool(eval(self.logic[i])[0])
        else:
            compare = bool(reduce(lambda x, y: x*y, eval(self.logic[i])[0]))
        if compare:                                                     #if compare i.e, logic is true then it should return the mentioned output
            return bool(eval(self.logic[i])[1])
        else:
            i += 1
            continue
    return False
```

- pattern_library.py :

The file contains all the very simple functions needed to modify patterns. The included functions are load_patterns(), save_patterns(), create_pattern(), delete_pattern(), select_pattern()

and display_pattern(). These functions will basically modify the pattern_list, a list of class Pattern objects available.

- **patterns.py :**

The file contain a function patterns() which uses pricing data and IMPORTANT_POINTS() to find the consecutive extrema and check for the pattern using pattern_recog() from class Patttern. The functions itself calls Plotting() to plot the identified patterns. Plotting() is visualizing function resulting in a well-marked plot, well-marked in sense it shows the extrema points and pattern lines in the graph. The function guess_pattern() is defined same as patterns(), the only difference is it has been adjusted to guessing part instead of technical analysis.

- **technical analysis :**

- **technical_analysis.py :** It runs the console based interaction with other back-end files. File provides the list of options which has been provided to the user to perform different tasks. The options include technical analysis, guessing, pattern info, add pattern, save pattern, delete pattern and exit.

- **options.py :** These file contains the two methods of technica_analysis.py. These methods are technical_analysis() and guessing(). Other options are related to pattern modification and hence, included in pattern_library.py