CHAITANYA KEDIA **180040027**

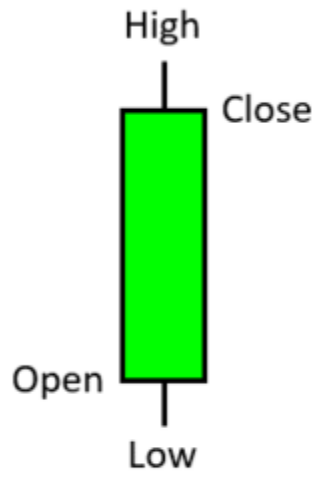
PATTERN ANALYSIS IN STOCK CHARTS USING DIGITAL IMAGE PROCESSING

INTRODUCTION

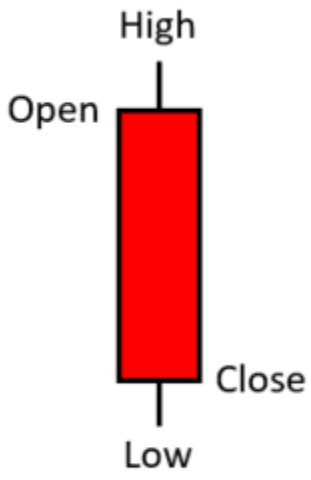
Trading in stock market is based on two major schools of market analysis, Fundamental Analysis and Technical Analysis. Technical analysis is a method based on historical price movement in financial markets in a trend or repetitive pattern. Traders develop their system based on identification of previous market patterns and make a trade on prediction of future price movement based on those patterns.

Technical analysis, or using charts to identify trading signals and price patterns shows the window to market psychology to identify opportunities to profit.

Stock prices are plotted as candlesticks on the chart where it consists of wicks and a body and the Open, High, Low and Close (OHLC) data of the stock during a time interval is plotted on the graph. The color of the candlestick shows the type of trading action which took place in the given interval of time where green or bullish candle denotes that buyers were dominant in the period and price closed higher whereas red or bearish candle shows bearish or selling sentiments and price hence closed lower.



Bullish candlestick

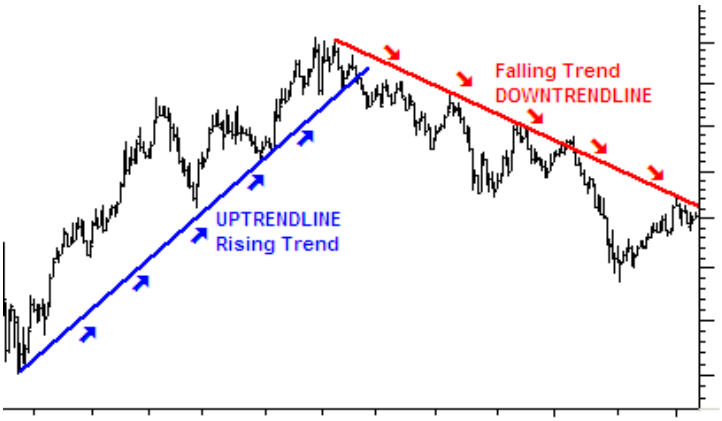


Bearish candlestick

Chart Patterns are an integral aspect of technical analysis, but they require proper identification and caters to trader's psychology on making a trade based on the patterns.

Support and Resistances are very important levels in trading which determines the future potential of a trade and helps trader determine the potential risk of the trade and also to selectively choose the opportunities with a better Risk to Reward. Support and Resistance in itself do not has a meaning, but just an area of interest and price decides what the level is going to be at the point of contact.

Trendlines are also a very important tool and can only be drawn when the market is trending in a direction! Market can only move in three ways- Uptrend, Downtrend, Sideways. So drawing a trend line shows the balance between the demand and supply and shows the interest of the market participants in the stock and can help a trader calibrate its entry and exit positions to optimize the risk to reward and reduce the risk involved and improve the accuracy of the trade.



Head & Shoulders Pattern

It is a chart pattern in which a large peak has a slightly smaller peak on either side of it. Traders look at head and shoulders pattern to predict a trend reversal.

OBJECTIVES

- Drawing chart patterns or levels on a chart is a very subjective process and can vary from person to person, so to make it quantified I am performing statistical analysis on the stock chart to obtain the levels.

- The process is manual and is time taking and there are hundreds of stocks which monitoring manually is almost impossible, so to create an automated process where I just have to run my script and I will get all the levels drawn on the stock data is the target to simplify and reduce the manual work but focus more on a trading system and also get rid of any fake trading opportunities.
- The script also identifies a very useful stock chart pattern which is head and shoulders which is a very reputable pattern and is highly profitable when traded. The pattern also could predict change of trend.
- To generate a status finally that whether the stock is trending or not or is sideways.

DATASETS USED

- The images are obtained from TradingView.com and the image is cleaned from the stock charts after removing the unnecessary charting facilities available for real time trading.

METHODOLOGY

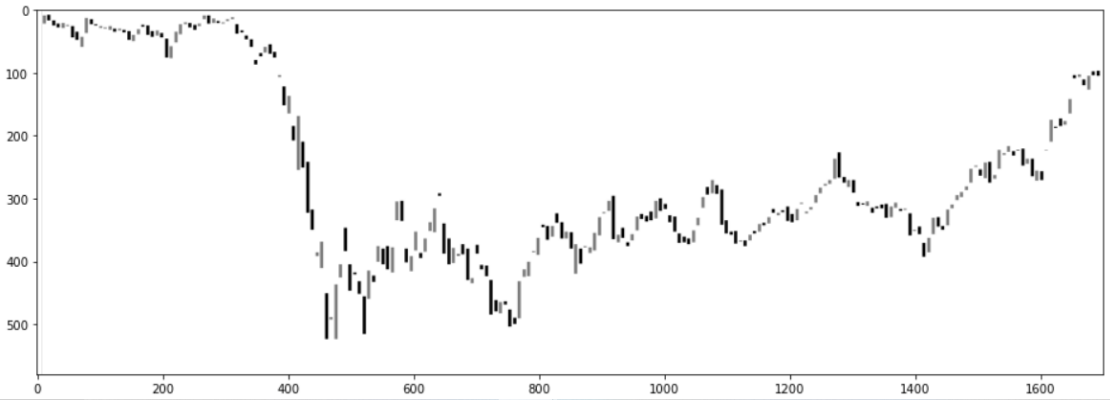
1. Obtain the images from TradingView.com and store it in a directory.
2. Import the libraries- pandas, numpy, matplotlib, PIL, sklearn
3. Crop the image using the crop method available for an image imported using PIL (where crop region is calculated manually and is valid for all the images).
4. Extract only one band from the image to make the calculations easy and remove the redundant information by visualization I found the Green data to be taking all the details.
5. Plot the initial figure using matplotlib to check that the data is propely imported or not.

In [181]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from PIL import Image
# from scipy.ndimage import filters
from sklearn.linear_model import LinearRegression
```

In [275]:
```
img=Image.open(r'ICICIBANK.png')
img=img.crop((0,120,1700,700))
arr=np.asarray(img)
n_arr=255-arr[:,:,1]
```

In [276]:
```
plt.figure(figsize=(16,6))
plt.imshow(255-n_arr,cmap="gray")
# White is red and Grey is Green
```

Out[276]:
```
<matplotlib.image.AxesImage at 0x11a3d85baf0>
```



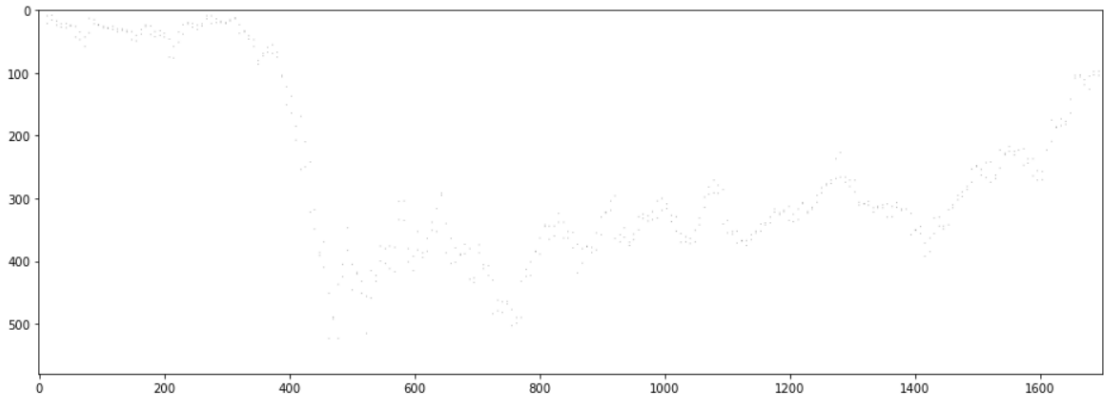
6. Apply a statistical analysis which I developed to reduce the data from candles to just topmost and bottommost point by making all those pixels whose top and right neighborhood are zero to be 255 and those whose bottom and right neighborhood are 0 to 255 and rest all to be 0.
7. By doing the above transformation my data gets reduced to a sequence of dots.
8. Now, plot the transformed array to see how the transformation works.

In [277]:
```
# northward = np.array([[1,2,1],[0,0,0],[-1,-2,-1]])
# fillow1 = filters.convolve(n_arr,northward)
# plt.figure(figsize=(16,6))
# plt.imshow(fillow1,cmap="gray")
ana_arr=np.zeros(n_arr.shape)
```

In [278]:
```
for i in range(1,n_arr.shape[0]-1):
    for j in range(n_arr.shape[1]):
        if n_arr[i-1,j]==0 and n_arr[i,j]!=0 and n_arr[i,j+1]==0:
            ana_arr[i,j]=255
        elif n_arr[i,j]!=0 and n_arr[i+1,j]==0 and n_arr[i,j+1]==0:
            ana_arr[i,j]=255
        else:
            ana_arr[i,j]=0
```

In [279]:
```
plt.figure(figsize=(16,6))
plt.imshow(255-ana_arr,cmap="gray")
```

Out[279]:
```
<matplotlib.image.AxesImage at 0x11a3b54c970>
```



9. Now, to see what is the real trend in the picture, I have performed regression with a window where the left and right boundaries are variable and are iterated using a for loop, and I will take that model which will give the maximum value of correlation.
10. Now I will generate the (x,y) coordinates of lines to be plot in the image by predicting using the linear model used.

In [280]:
```
x,y=np.where(ana_arr==255)
```

In [281]:
```
x=np.reshape(x,(x.shape[0],1))
y=np.reshape(y,(x.shape[0],1))
vali=0
valj=0
m = 0
for i in range(int(2*x.shape[0]/15)):
    for j in range(int(x.shape[0]/15)):
        x_=x[5*i:x.shape[0]-5*j]
        y_=y[5*i:x.shape[0]-5*j]
        reg = LinearRegression().fit(x_, y_)
        s =reg.score(x_,y_)
        if s>m:
            m=s
            vali = i
            valj = j
x_=x[5*vali:x.shape[0]-5*valj]
y_ = y[5*vali:x.shape[0]-5*valj]
reg = LinearRegression().fit(x_,y_)
```

In [282]:
```
m
print(reg.coef_)
print(reg.intercept_)
```

```
[[3.0865087]]
[266.35542463]
```

In [283]:
```
x_plot=np.linspace(0,n_arr.shape[0]-1, n_arr.shape[0]).astype('int')
y_plot=np.asarray(reg.predict(np.reshape(x_plot,(x_plot.shape[0],1))),int)
```

In [284]:
```
for i in range(n_arr.shape[0]):
        if y_plot[i]>0 and y_plot[i]<n_arr.shape[1]:
            n_arr[x_plot[i],y_plot[i]]=255
```

11. Now, I created a way to calculate the horizontal lines by iterating vertically on the image by creating window of size to be 0.2*image height—this no. is obtained intuitively and by hit and trial method, finally to get the horizontal line in a window I take weighted average of the rows (i.e. summing all the values in a row to obtain the row value and then using its y coordinate to assign a weight to the row and finally taking weighted average of all the rows in that window)

```
In [285]: # Horizontal Lines calculation

          h_val = []
          period=int(img.size[1]/5)
          for i in range(int(n_arr.shape[0]/period)):
              h_arr = ana_arr[period*i:period*(1+i)]
              sum_=0
              for j in range(period):
                  sum_= sum_ + j*h_arr[j].sum()
              sum_ = sum_/(255*period) + period*i
              h_val.append(sum_)
```

12. For trend line creation, I again decided a window of size=8 to be iterated left to right on the image to get the points corresponding to minimum and maximum x coordinate in the window and finally append them in an array to store the values.

```
In [286]: # Trend Lines calculation

          t_lines_up=[]
          # t_lines_down=[]
          period= 8
          for i in range(period):
              x_min=n_arr.shape[0]
              y_min=n_arr.shape[1]
              x_max=0
              y_max=0

          #     x_min_down=n_arr.shape[0]
          #     y_min_down=n_arr.shape[1]
          #     x_max_down=0
          #     y_max_down=0

              p_arr=ana_arr[:,int(n_arr.shape[1]*i/period):int(n_arr.shape[1]*(i+1)/period)]
              for j in range(p_arr.shape[1]):
                  if 255 in p_arr[:,j]:
          #           z_x=int((list(p_arr[:,j]).index(255)+list(p_arr[:,j]).index(255,list(p_arr[:,j]).index(255)+1))/2)

                      z_x = list(p_arr[:,j]).index(255)
          #           z_x_down = list(p_arr[:,j]).index(255,list(p_arr[:,j]).index(255)+1)
                      z_y=j

                      if z_x>x_max:
                          x_max=z_x
                          y_max=z_y+int(n_arr.shape[1]*i/period)
                      if z_x<x_min:
                          x_min = z_x
                          y_min = z_y+int(n_arr.shape[1]*i/period)
```

```
#         if p_arr[:,j].sum()>255:

#             z_x_down = int((list(p_arr[:,j]).index(255)+list(p_arr[:,j]).index(255,list(p_arr[:,j]).index(255)+1))/2)
# #              z_x_down = list(p_arr[:,j]).index(255,list(p_arr[:,j]).index(255)+1)
#             z_y=j

#             if z_x_down>x_max_down:
#                 x_max_down=z_x_down
#                 y_max_down = z_y + int(n_arr.shape[1]*i/period)

#             if z_x_down< x_min_down:
#                 x_min_down = z_x_down
#                 y_min_down = z_y+int(n_arr.shape[1]*i/period)

    t_lines_up.append([(x_min,y_min),(x_max,y_max)])
#     t_lines_down.append([(x_min_down,y_min_down),(x_max_down,y_max_down)])
```

13. For Head and Shoulders calculation I decided a period intuitively of size=20 and iterated from left to right on the image to check for the pattern by seeing the top most point and bottom most point in the window and then if three consecutive window follow a pattern such that the first window's highest point is lower than the second window and the third window's highest point is lower than the second window then my condition is satisfied and it may be a head and shoulders pattern.

```
In [287]: # Head and Shoulders calculation

period= 20
hs=[]
for i in range(period):
    x_l_min = 0
    y_l_min = 0
    x_r_min = 0
    y_r_min = 0
    x_top = n_arr.shape[0]
    y_top = n_arr.shape[1]
    hs_arr=ana_arr[:,int(n_arr.shape[1]*i/period):int(n_arr.shape[1]*(i+1)/period)]
    for j in range(hs_arr.shape[1]):
        if 255 in hs_arr[:,j]:
            z_x = list(hs_arr[:,j]).index(255)
            z_y=j

            if z_x<x_top:
                x_top = z_x
                y_top = z_y+int(n_arr.shape[1]*i/period)

    for j in range(hs_arr.shape[1]):
        if 255 in hs_arr[:,j]:
            z_x = list(hs_arr[:,j]).index(255)
            if hs_arr[:,j].sum()>255:
                z_x = int((list(hs_arr[:,j]).index(255)+list(hs_arr[:,j]).index(255,list(hs_arr[:,j]).index(255)+1))/2)
            z_y=j

            if z_y+int(n_arr.shape[1]*i/period)<y_top and z_x>x_l_min:
                x_l_min = z_x
                y_l_min = z_y+int(n_arr.shape[1]*i/period)
            elif z_y+int(n_arr.shape[1]*i/period)>=y_top and z_x>x_r_min:
                x_r_min = z_x
                y_r_min = z_y + int(n_arr.shape[1]*i/period)
```

```
    if x_r_min==0:
        x_r_min=x_top
        y_r_min=y_top
    if x_l_min==0:
        x_l_min=x_top
        y_l_min=y_top
    hs.append([(x_l_min,y_l_min),(x_top,y_top),(x_r_min,y_r_min)])

hns=[]
for i in range(1,len(hs)-1):
    if hs[i-1][1][0]-hs[i][1][0]>10 and hs[i+1][1][0]-hs[i][1][0]>10:
        hns.append(hs[i-1])
        hns.append(hs[i])
        hns.append(hs[i+1])
```

```
# Head and Shoulders
hns_plot=[]
for i in range(len(hns)):
    if i==0:
        hns_plot.append([hns[i][0],hns[i][1],max(hns[i][2],hns[i+1][0])])
    elif i==len(hns)-1:
        hns_plot.append([max(hns[i][0],hns[i-1][2]),hns[i][1],hns[i][2]])
    else:
        hns_plot.append([max(hns[i][0],hns[i-1][2]),hns[i][1],max(hns[i+1][0],hns[i][2])])
# Neck line for head and shoulders
neck_hns=[]
for i in range(len(hns_plot)):
    if i==0:
        neck_hns.append([hns_plot[i][0],hns_plot[i][2]])
    elif i==len(hns_plot) -1 :
        neck_hns.append([hns_plot[i][0],hns_plot[i][2]])
    else:
        neck_hns.append([hns_plot[i][0],hns_plot[i][2]])
```

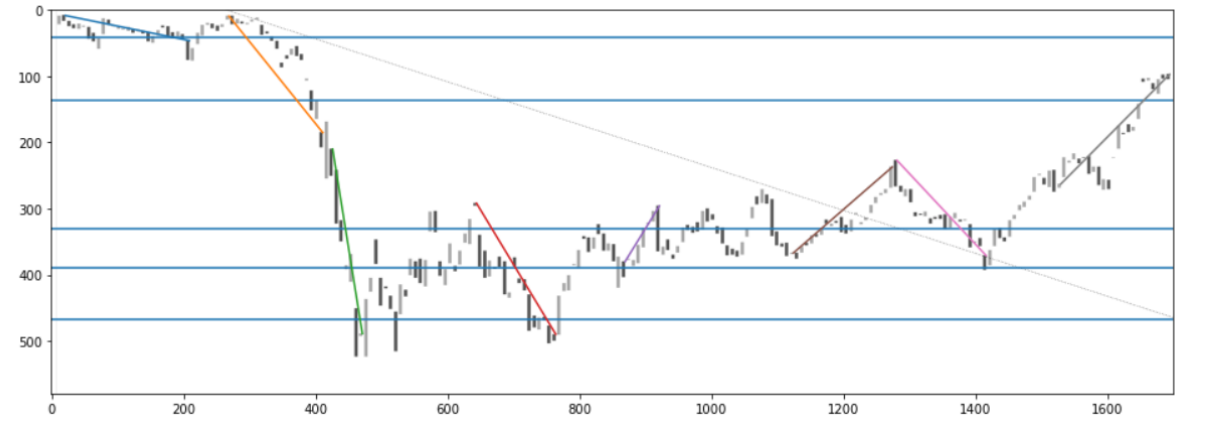
14. Finally, I plotted all the lines on the plot.
15. Then, lastly I created a methodology based on trend lines to find the slope of the trendline corresponding to the last window and if that slope is >0.5(0.5 is obtained by

hit and trial and some raw logic) then the stock is uptrending, else if it is less than -0.5 it is downtrending and else it is consolidating.

```
In [291]: # Finding that whether the stock is trending or not recently.

t_lines=[]
period= 5
for i in range(period-1,period):
    x_min=n_arr.shape[0]
    y_min=n_arr.shape[1]
    x_max=0
    y_max=0

    p_arr=ana_arr[:,int(n_arr.shape[1]*i/period):int(n_arr.shape[1]*(i+1)/period)]
    for j in range(p_arr.shape[1]):
        if 255 in p_arr[:,j]:

            z_x = list(p_arr[:,j]).index(255)
            z_y=j

            if z_x>x_max:
                x_max=z_x
                y_max=z_y+int(n_arr.shape[1]*i/period)
            if z_x<x_min:
                x_min = z_x
                y_min = z_y+int(n_arr.shape[1]*i/period)

    t_lines.append([(x_min,y_min),(x_max,y_max)])

# slope of last trendline
slope = -(t_lines[0][0][0]-t_lines[0][1][0])/(t_lines[0][0][1]-t_lines[0][1][1])
if slope>0.5:
    print("Uptrending")
elif slope<0.5:
    print("Downtrending")
else:
    print("Sideways")

Uptrending
```

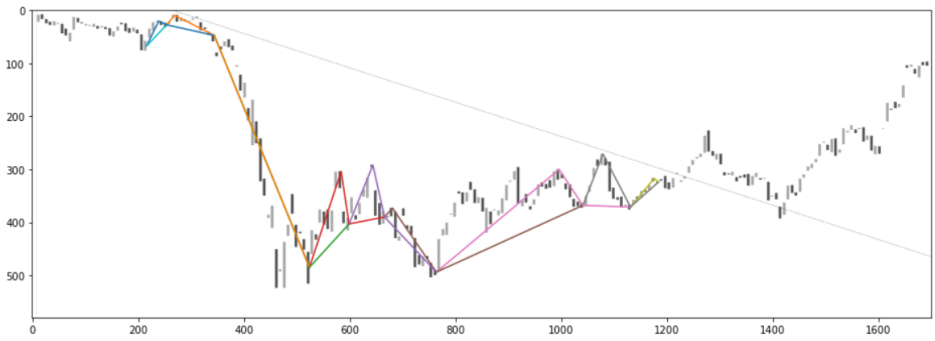
RESULTS AND DISCUSSION

ICICIBANK-1D TIMEFRAME



- In the above diagram, you can see the blue lines are the horizontal lines and these are the area of support and resistances.
- The long gray line is the line generated using linear regression and it shows a general demand=supply line.
- The colored lines are the trendlines in that timeperiod which shows the direction of trend.

```
In [290]: plt.figure(figsize=(16,6))
          plt.imshow(255-n_arr,cmap="gray")

          # Plot head and shoulders

          for u in hns_plot:
              plt.plot([u[0][1],u[1][1],u[2][1]],[u[0][0],u[1][0],u[2][0]])
          for a in neck_hns:
              plt.plot([a[0][1],a[1][1]],[a[0][0],a[1][0]])
```



- The group of three triangles starting at y=450 to y=800 shows the formation of head and shoulders pattern which broke as you can see in the picture that the neckline was broken and hence one could initiate a trade.
- The group of three triangles starting at y=800 to y=1200 shows the formation of head and shoulders pattern which did not broke if broken a trade could be initiated

SUMMARY

- No, such analysis has been successfully performed (at least on Google and there are only few research papers that uses CNN). Through this project I wanted to confirm about the feasibility of the project and try the statistical tools rather than moving to machine learning algorithms which would introduce modelling bias, etc in the analysis.
- The statistical analysis in my code quantifies the levels(support/resistance/trendlines) and also detects the “head and shoulders pattern”.
- The code gives the plot of these levels and the chart pattern if present to be an output and also tells whether the stock is trending and in which direction
- The regression applied on the data is very effective (validated visually) and is respected a lot by the stock and can be of serious consideration while taking a trade.
- The benefit of working on charts rather than raw data is I can zoom out my charts according to my choice and then take a picture which leads to more datapoints in an image.
- Also, my algorithm can capture all the data and is independent of the price magnitude of the stock or any other mathematical manipulations which is the case in traditional algorithms where they use raw data.
- Also, in the raw data you need to iterate all your calculations step by step on all the data points but here I just have to iterate row wise or column wise(and one row contains all the required knowledge about all the datapoints at any time at the same price)