**Install and** **Importing Packages**

#Install Packages

pip install numpy

pip install pandas

pip install seaborn

pip install matpoltlib

pip install termcolor

#Import Packages

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from math import floor

from termcolor import colored as cl

**Reading data using Pandas and describe the data**

appl\_data = pd.read\_csv('appl\_1980\_2014.csv')

appl\_data

#Overview Rows, Columns, Data types (dtypes), dimensions

appl\_data.info()

appl\_data.shape

appl\_data.describe

appl\_data.columns

**Descriptive statistics**

#Descriptive statistics Any statistical parameter and characteristics available to data. (Min, Max, Range, #Mean, Variance, Standard Deviation, Coefficient of Variation,…)

appl\_data.agg({'Open':'max', 'Close':'max','High':'max',

'Low':'max','Volume':'max','Adj\_Close':'max'})

appl\_data.mean()

appl\_data.agg({'Open':'min', 'Close':'min','High':'min',

'Low':'min','Volume':'min','Adj\_Close':'min'})

appl\_data.median()

appl\_data.var()

appl\_data.std()

Coefficient\_of\_Variation= appl\_data.std()/appl\_data.mean()

**Preparation data**

#Converting object to datetime format

appl\_data['Date'] = pd.to\_datetime(appl\_data['Date'])

#Sort list of dates

appl\_data=appl\_data.sort\_values(by=['Date'], ascending= True)

appl\_data

appl\_data= pd.DataFrame(appl\_data)

**Plotting the Trading Lists**

#Trend and Distribution plot (Trend Plot, Histogram, Box and Whiskers) {Trend of the data, as #demonstrated in Figure XXXX, showing an increase for close price}

appl\_data.plot(x="Date", y=["Open","Close"], kind="line")

plt.grid()

plt.show()

appl\_data= pd.DataFrame(appl\_data)

appl\_data.plot(x="Date", y=["Volume"], kind="line")

plt.show()

appl\_data= pd.DataFrame(appl\_data)

appl\_data.plot(x="Date", y=["Adj\_Close"], kind="line")

plt.show()

appl\_data= pd.DataFrame(appl\_data)

appl\_data.plot(x="Date", y=["High","Low"], kind="line")

plt.show()

**Analytics performed (Methods)**

#Analytics performed (Methods) Indicators used to predict and forecast the available time series.

Simple Moving Average (SMA)

#Definition, window size (parameters of indicator), plots, code implementation. Signaling (Strategy for #buy and sell)

#definition function for simple moving average

def moving\_fun(data,m\_range):

i = 0

moving\_averages = []

while i < len(data) - m\_range + 1:

m\_ranges = data[i : i + m\_range]

m\_range\_average = round(sum(m\_ranges) / m\_range, 2)

moving\_averages.append(m\_range\_average)

i += 1

return moving\_averages

# moving average 30 (make, merge, plot)

moving\_30 = moving\_fun(appl\_data["Open"],30)

len(moving\_30),type(moving\_30), len(appl\_data)

appl\_moving\_30=pd.DataFrame(appl\_data["Date"][29:])

appl\_moving\_30["moving\_30"]=moving\_30

appl\_moving\_30

appl\_data\_join= pd.merge(appl\_data, appl\_moving\_30, on='Date', how='left')

appl\_data\_join

appl\_data\_join.plot(x="Date", y=["Open","moving\_30"], kind="line")

# moving average 200 (make, merge, plot)

moving\_200= moving\_fun(appl\_data["Open"],200)

len(moving\_200),type(moving\_200), len(appl\_data)

appl\_moving\_200=pd.DataFrame(appl\_data["Date"][199:])

appl\_moving\_200["moving\_200"]=moving\_200

appl\_moving\_200

appl\_data\_join2= pd.merge(appl\_data\_join, appl\_moving\_200, on='Date', how='left')

appl\_data\_join2

plt.figure(figsize=(9,5))

appl\_data\_join2.plot(x="Date", y=["Close","moving\_200","moving\_30"], kind="line")

plt.title('Plot')

plt.ylabel('Function Vaule')

plt.xlabel('x-axis')

plt.show()

#Visualization of signal

appl\_data\_join2=appl\_data\_join2.set\_index('Date')

fig, ax = plt.subplots(figsize=(14,8))

ax.plot(appl\_data\_join2['Close'],linewidth=0.5, color='blue', alpha = 0.9)

ax.plot(appl\_data\_join2['moving\_30'], label = 'moving\_30', alpha = 0.85)

ax.plot(appl\_data\_join2['moving\_200'], label = 'moving\_200' , alpha = 0.85)

ax.scatter(appl\_data\_join2.index , appl\_data\_join/2['Buy\_Signal\_price'] , label = 'Buy' , marker = '^', color = 'green',alpha =1 )

ax.scatter(appl\_data\_join2.index , appl\_data\_join2['Sell\_Signal\_price'] , label = 'Sell' , marker = 'v', color = 'red',alpha =1 )

ax.set\_title(stocksymbols[0] + " Price History with buy and sell signals",fontsize=10, backgroundcolor='blue', color='white')

ax.set\_xlabel(f'{startdate} - {end\_date}' ,fontsize=18)

ax.set\_ylabel('Close Price INR (₨)' , fontsize=18)

legend = ax.legend()

ax.grid()

plt.tight\_layout()

plt.show()

#Function for buy and sell signal

def buy\_sell(data):

signalBuy = []

signalSell = []

position = False

for i in range(len(data)):

if data['moving\_30'][i] > data['moving\_200'][i]:

if position == False :

signalBuy.append(data['Close'][i])

signalSell.append(np.nan)

position = True

else:

signalBuy.append(np.nan)

signalSell.append(np.nan)

elif data['moving\_30'][i] < data['moving\_200'][i]:

if position == True:

signalBuy.append(np.nan)

signalSell.append(data['Close'][i])

position = False

else:

signalBuy.append(np.nan)

signalSell.append(np.nan)

else:

signalBuy.append(np.nan)

signalSell.append(np.nan)

return pd.Series([signalBuy, signalSell])

appl\_data\_join2['Buy\_Signal\_price'], appl\_data\_join2['Sell\_Signal\_price'] = buy\_sell(appl\_data\_join2)

appl\_data\_join2['Sell\_Signal\_price'].unique()

Exponential Moving Average (EMA)

#Definition, window size (parameters of indicator), plots, code implementation. Signaling (Strategy for #buy and sell)

#definition function for Exponential moving average

def moving\_average\_EMA(Data,alpha):

numbers\_series = pd.Series(Data)

moving\_averages = round(numbers\_series.ewm(

alpha=alpha, adjust=False).mean(), 2)

moving\_averages\_list = moving\_averages.tolist()

return moving\_averages\_list

moving\_average\_Exponential =moving\_average\_EMA(appl\_data["Open"],0.5)

len(moving\_average\_Exponential)

appl\_moving\_Exponential=pd.DataFrame(appl\_data["Date"])

appl\_moving\_Exponential["moving\_Exponential"]=moving\_average\_Exponential

appl\_moving\_Exponential

appl\_data\_join= pd.merge(appl\_data, appl\_moving\_Exponential, on='Date', how='left')

appl\_data\_join

plt.figure(figsize=(5,5))

appl\_data\_join.plot(x="Date", y=["Open","moving\_Exponential"], kind="line")

plt.title('Plot')

plt.ylabel('Function Vaule')

plt.xlabel('x-axis')

plt.show()

#Function for buy and sell signal

def buy\_sell(data):

signalBuy = []

signalSell = []

position = False

for i in range(len(data)):

if data['moving\_Exponential'][i] > data['moving\_Exponentia2'][i]:

if position == False :

signalBuy.append(data['Close'][i])

signalSell.append(np.nan)

position = True

else:

signalBuy.append(np.nan)

signalSell.append(np.nan)

elif data['moving\_Exponential'][i] < data['moving\_Exponentia2'][i]:

if position == True:

signalBuy.append(np.nan)

signalSell.append(data['Close'][i])

position = False

else:

signalBuy.append(np.nan)

signalSell.append(np.nan)

else:

signalBuy.append(np.nan)

signalSell.append(np.nan)

return pd.Series([signalBuy, signalSell])

appl\_data\_join2['Buy\_Signal\_price'], appl\_data\_join2['Sell\_Signal\_price'] = buy\_sell(appl\_data\_join2)

appl\_data\_join2['Sell\_Signal\_price'].unique()

Moving Average Convergence Divergence (MACD)

#Definition, window size (parameters of indicator), plots, code implementation. Signaling (Strategy for #buy and sell)

Steps for calculate MACD

1. Calculate a short-period EMA of the price for the chosen time period.
2. Calculate a long-period EMA of the price for the chosen time period.
3. Subtract the long-period EMA from the short-period EMA.
4. Calculate a given period EMA of the result obtained from step 3.

#definition function for Moving Average Convergence Divergence

def get\_macd(price, slow, fast, smooth):

exp1 = price.ewm(span = fast, adjust = False).mean()

exp2 = price.ewm(span = slow, adjust = False).mean()

macd = pd.DataFrame(exp1 - exp2).rename(columns = {'Close':'macd'})

signal = pd.DataFrame(macd.ewm(span = smooth, adjust = False).mean()).rename(columns = {'macd':'signal'})

hist = pd.DataFrame(macd['macd'] - signal['signal']).rename(columns = {0:'hist'})

frames = [macd, signal, hist]

df = pd.concat(frames, join = 'inner', axis = 1)

return df

data\_macd = get\_macd(appl\_data["Close"], 26, 12, 9)

data\_macd

#plot\_MACD

def plot\_macd(data, macd, signal, hist):

ax1 = plt.subplot2grid((8,1), (0,0), rowspan = 5, colspan = 1)

ax2 = plt.subplot2grid((8,1), (5,0), rowspan = 3, colspan = 1)

ax1.plot(data)

ax2.plot(macd, color = 'grey', linewidth = 1.5, label = 'MACD')

ax2.plot(signal, color = 'skyblue', linewidth = 1.5, label = 'SIGNAL')

for i in range(len(data)):

if str(hist[i])[0] == '-':

ax2.bar(data.index[i], hist[i], color = '#ef5350')

else:

ax2.bar(data.index[i], hist[i], color = '#26a69a')

plt.legend(loc = 'lower right')

plt.figure(figsize=(9,5))

plot\_macd(appl\_data["Close"], data\_macd['macd'], data\_macd['signal'], data\_macd['hist'])

#code implementation

def implement\_macd\_strategy(data, data\_macd):

buy\_price = []

sell\_price = []

macd\_signal = []

signal = 0

for i in range(len(data\_macd)):

if data\_macd['macd'][i] > data\_macd['signal'][i]:

if signal != 1:

buy\_price.append(data[i])

sell\_price.append(np.nan)

signal = 1

macd\_signal.append(signal)

else:

buy\_price.append(np.nan)

sell\_price.append(np.nan)

macd\_signal.append(0)

elif data\_macd['macd'][i] < data\_macd['signal'][i]:

if signal != -1:

buy\_price.append(np.nan)

sell\_price.append(data[i])

signal = -1

macd\_signal.append(signal)

else:

buy\_price.append(np.nan)

sell\_price.append(np.nan)

macd\_signal.append(0)

else:

buy\_price.append(np.nan)

sell\_price.append(np.nan)

macd\_signal.append(0)

return buy\_price, sell\_price, macd\_signal

buy\_price, sell\_price, macd\_signal = implement\_macd\_strategy(appl\_data['Close'], data\_macd)

# Signaling (Strategy for #buy and sell)

position = []

for i in range(len(macd\_signal)):

if macd\_signal[i] > 1:

position.append(0)

else:

position.append(1)

for i in range(len(appl\_data['Close'])):

if macd\_signal[i] == 1:

position[i] = 1

elif macd\_signal[i] == -1:

position[i] = 0

else:

position[i] = position[i-1]

macd = data\_macd['macd']

signal = data\_macd['signal']

close\_price = appl\_data['Close']

macd\_signal = pd.DataFrame(macd\_signal).rename(columns = {0:'macd\_signal'}).set\_index(appl\_data.index)

position = pd.DataFrame(position).rename(columns = {0:'macd\_position'}).set\_index(appl\_data.index)

frames = [close\_price, macd, signal, macd\_signal, position]

strategy = pd.concat(frames, join = 'inner', axis = 1)

strategy

SMA PROFIT SMA

appl\_data\_join2['Buy\_Signal\_price']

appl\_data\_join2['Sell\_Signal\_price']

position = []

for i in range(len(appl\_data\_join2)):

if appl\_data\_join2['Buy\_Signal\_price'][i] > 1 or

appl\_data\_join2['Sell\_Signal\_price'][i] > 1:

position1.append(0)

else:

position.append(1)

appl\_ret = pd.DataFrame(np.diff(appl\_data['close'])).rename(columns = {0:'returns'})

sma\_strategy\_ret = []

for i in range(len(appl\_ret)):

try:

returns = appl\_ret['returns'][i]\*strategy['position'][i]

sma\_strategy\_ret.append(returns)

except:

pass

sma\_strategy\_ret\_df = pd.DataFrame(sma\_strategy\_ret).rename(columns = {0:'sma\_returns'})

investment\_value = 100000

number\_of\_stocks = floor(investment\_value/appl\_data['close'][0])

sma\_investment\_ret = []

for i in range(len(sma\_strategy\_ret\_df['position'])):

returns = number\_of\_stocks\*sma\_strategy\_ret\_df['position'][i]

sma\_investment\_ret.append(returns)

sma\_investment\_ret\_df = pd.DataFrame(sma\_investment\_ret).rename(columns = {0:'investment\_returns'})

total\_investment\_ret = round(sum(sma\_investment\_ret\_df['investment\_returns']), 2)

profit\_percentage = floor((total\_investment\_ret/investment\_value)\*100)

print(cl('Profit gained from the SMA strategy by investing $100k in APPLE : {}'.format(total\_investment\_ret), attrs = ['bold']))

print(cl('Profit percentage of the SMA strategy : {}%'.format(profit\_percentage), attrs = ['bold']))

EMA Profit EMA

appl\_data\_join2['Buy\_Signal\_price']

appl\_data\_join2['Sell\_Signal\_price']

position = []

for i in range(len(appl\_data\_join2)):

if appl\_data\_join2['Buy\_Signal\_price'][i] > 1 or

appl\_data\_join2['Sell\_Signal\_price'][i] > 1:

position1.append(0)

else:

position.append(1)

appl\_ret = pd.DataFrame(np.diff(appl\_data['close'])).rename(columns = {0:'returns'})

ema\_strategy\_ret = []

for i in range(len(appl\_ret)):

try:

returns = appl\_ret['returns'][i]\*strategy['position'][i]

ema\_strategy\_ret.append(returns)

except:

pass

ema\_strategy\_ret\_df = pd.DataFrame(ema\_strategy\_ret).rename(columns = {0:'ema\_returns'})

investment\_value = 100000

number\_of\_stocks = floor(investment\_value/appl\_data['close'][0])

ema\_investment\_ret = []

for i in range(len(ema\_strategy\_ret\_df['position'])):

returns = number\_of\_stocks\*ema\_strategy\_ret\_df['position'][i]

ema\_investment\_ret.append(returns)

ema\_investment\_ret\_df = pd.DataFrame(ema\_investment\_ret).rename(columns = {0:'investment\_returns'})

total\_investment\_ret = round(sum(ema\_investment\_ret\_df['investment\_returns']), 2)

profit\_percentage = floor((total\_investment\_ret/investment\_value)\*100)

print(cl('Profit gained from the EMA strategy by investing $100k in APPLE : {}'.format(total\_investment\_ret), attrs = ['bold']))

print(cl('Profit percentage of the EMA strategy : {}%'.format(profit\_percentage), attrs = ['bold']))

MACD Profit MACD

apple\_ret = pd.DataFrame(np.diff(appl\_data['Close'])).rename(columns = {0:'returns'})

macd\_strategy\_ret = []

for i in range(len(apple\_ret)):

try:

returns = apple\_ret['returns'][i]\*strategy['macd\_position'][i]

macd\_strategy\_ret.append(returns)

except:

pass

macd\_strategy\_ret\_df = pd.DataFrame(macd\_strategy\_ret).rename(columns = {0:'macd\_returns'})

investment\_value = 100000

number\_of\_stocks = math.floor(investment\_value/appl\_data['Close'][0])

macd\_investment\_ret = []

for i in range(len(macd\_strategy\_ret\_df['macd\_returns'])):

returns = number\_of\_stocks\*macd\_strategy\_ret\_df['macd\_returns'][i]

macd\_investment\_ret.append(returns)

macd\_investment\_ret\_df = pd.DataFrame(macd\_investment\_ret).rename(columns = {0:'investment\_returns'})

total\_investment\_ret = round(sum(macd\_investment\_ret\_df['investment\_returns']), 2)

profit\_percentage = math.floor((total\_investment\_ret/investment\_value)\*100)

print(cl('Profit gained from the MACD strategy by investing $100k in apple : {}'.format(total\_investment\_ret), attrs = ['bold']))

print(cl('Profit percentage of the MACD strategy : {}%'.format(profit\_percentage), attrs = ['bold']))

Conclusion

Result of SMA

**Profit gained from the MACD strategy by investing $100k in apple : -10575**

**Profit percentage of the MACD strategy : %-10**

Result of EMA

**Profit gained from the MACD strategy by investing $100k in apple : -15879**

**Profit percentage of the MACD strategy : %-15**

Result of MACD

**Profit gained from the MACD strategy by investing $100k in apple : 34675**

**Profit percentage of the MACD strategy : %-34**