Implementation
of
Post Earnings Announcement Drift (PEAD)
in
Indian Markets (Nifty50)

by

Anup Hanamant Itale, IIT Bombay
&
Aniket Niranjan Mishra, IIT Kharagpur

as a part of
Winter Internship 2019
at

Centre for Analytical Finance
Indian School of Business, Hyderabad

o

INDEX

1. Introduction

This report replicates the PEAD phenomenon for Indian markets. In financial economics and accounting research, post–earnings-announcement drift, or PEAD (also named the SUE effect) is the tendency for a stock's cumulative abnormal returns to drift in the direction of an earnings surprise for several weeks (even several months) following an earnings announcement. It is an anomaly resulting from the ability of market participants to predict future abnormal returns by using information that is contained in past earnings announcements. Once a firm's current earnings become known, the information content should be quickly digested by investors and incorporated into the efficient market price. However, it has long been known that this is not exactly what happens since there are a number of inefficiencies in the market. PEAD is an academically well-documented anomaly first discovered by Ball and Brown in 1968. Since then it has been studied and confirmed by countless academics in many international markets. The PEAD anomaly provides investors with opportunities to earn abnormal profits. The phenomenon can be explained with a number of hypotheses. Possible explanations for the PEAD like the delay in price response and misspecification of CAPM have also been covered in this report.

2. Literature review

Historically, the incomes of firms have tended to move together. About half of the variability in the level of an average firm's earnings per share (EPS) could be associated with economy-wide effects [Ball and Brown (1967)]. In light of this evidence, at least part of the change in a firm's income from one year to the next is to be expected. If, in prior years, the income of a firm has been related to the incomes of other firms in a particular way, then knowledge of that past relation, together with knowledge of the incomes of those other firms for the present year, yields a conditional expectation for the present income of the firm. Thus, apart from confirmation effects, the amount of new information conveyed by the present income number can be approximated by the difference between the actual change in income and its conditional expectation. But not all of this difference is necessarily new information. Some changes in income result from financing and other policy decisions made by the firm. If the income forecast error is negative (that is, if the actual change in income is less than its conditional expectation), we define it as bad news and predict that if there is some association between accounting income numbers and stock prices, then release of the income number would result in the return on that firm's

securities being less than would otherwise have been expected. Such a result would be evidenced by negative behaviour in the stock return residuals around the annual report announcement date. The converse should hold for a positive forecast error. Foster, Olsen, and Shevlin [1984] are among the many who have replicated the phenomenon. They show that over the 60 trading days subsequent to an earnings announcement, a long position in stocks with unexpected earnings in the highest decile, combined with a short position in stocks in the lowest decile, yields an annualized "abnormal" return of about 25%, before transactions costs. Victor L. Bernard and Jacob K. Thomas formed portfolios on the basis of SUE (Standardized Unexpected Earnings) and proved that returns can be generated in the above manner. Mendenhall provided explanation as to why unbiased investors do not or cannot eliminate the underreaction to earnings announcement and hence enforce market efficiency. Bhushan, on the other hand provides evidence that the magnitude of the post-earnings announcement drift is positively related to the direct and indirect costs of trading. Runeet et al. show that to avoid look-ahead bias, the deciles should be formed based on the SUE of previous quarter.

3. Data Collection

This study was conducted on the companies listed on the National Stock Exchange, namely stocks that are listed on the Nifty 50 index. NSE was preferred over BSE because of higher daily trading volume which translates to more liquidity. The data used in the study consisted of the EPS and stock prices of the companies. Additionally the returns of Nifty 50 index was also taken to benchmark the performance of our strategy against just investing in the index itself. The companies listed on Nifty 50 announce their earnings regularly and earnings data was easily available for them via CMIE Prowess. To pinpoint the date of earnings announcement, NSE corporate announcements was used. After cleaning the extracted data, it was found that 48 companies had sufficient earnings data. Their earnings and prices in the period starting from quarter ending in Jun 2008 till quarter ending in Sep 2019 (total 46 quarters) have been included in the study.

4. Methodology

The data was extracted for various companies listed on the Bombay Stock Exchange along with their announcement dates. Each company had a unique announcement date in every quarter. Trading position for the company was taken one day after the announcement date, if the announcement is done on a non-trading day (saturday/sunday/national holidays), then the next trading day was taken for taking position. The code developed as part of the exercise can be tweaked to incorporate any number of days in the holding period. For the current analysis, the position was held for 40 trading days after start which translates to approximately 2 months.
For SUE(Standardised unexpected earnings) calculation, expected earnings is required.

Several variants of the method to calculate expected earnings are prevalent:-

1. Take average of same quarter from previous 4 years: This method was not used because it does not generalize well in case of sudden changes in the business environment of the company.
2. Take average of analyst's estimates of earnings: Analyst estimates data for Nifty 50 stocks in the study period was not available from a reliable source.
3. Using a time series model like ARIMA: ARIMA model can be used to predict earnings based on previous earnings, only when the time series formed from previous lagged earnings is stationary(without trend and seasonality). In our case, the series was not stationary, hence ARIMA could not be used.

As the above methods could not be used, in the present study, we used linear regression to predict the expected earnings based on earnings of past 16 quarters. Clearly, for the first 16 quarters of available data (Jun 2008 to Mar 2012), expected earnings was not available. Hence trading was started from Jun 2012.
The expected EPS was subtracted from the actual EPS announced and then divided by the standard deviation of the 16 earlier earnings used for prediction to get the SUE.

SUE = (Actual Earnings - Expected Earnings)/Standard Deviation

The expected EPS was subtracted from the actual EPS announced and then divided by the standard deviation of the 16 earlier earnings used for prediction to get the SUE.
For starting the trading from earnings announcement date of quarter q, the SUE values of quarter q-1 were arranged in deciles and cutoff for top and bottom decile was noted. This cutoff was used to filter companies on their announcement date. Say, if SUE cutoff for top

10 percentile is x and the same for bottom 10 percentile is y. So, when earnings announcement for a company for quarter occurs, we compare the SUE with x and y. If SUE > x, then the company is accepted in LONG portfolio. If SUE < y, the company is put in the SHORT portfolio.

The actual returns of the portfolios and the market was taken for the 40 trading days following the announcement date. These 40 days were used for back testing(judging the performance of the portfolio). The percentage return of a stock is given by:

% return = (Final price - Initial price)/Initial price

This return is positive when the price of the stock increases and is negative when it drops The return of the overall portfolio is taken as the sum of all LONG returns minus the sum of all SHORT returns.

Also, in the implementation of the strategy, we have implemented both STOP LOSS and TAKE PROFIT. If the profit from a stock crosses a certain TP threshold (here we had 10%), then the position is cleared (stock is sold) and the profit is booked. This enables us to ensure that a good profit once achieved, is booked. On the other hand if the loss hits 5% (SL threshold), the position is cleared and the investor is shielded from heavy losses. These have been implemented in each stock traded, both in the LONG and SHORT portfolios.

Assumptions taken:-

1. Short selling is freely allowed in Indian markets.
2. There are no market friction forces like transaction costs.

5. Results and Conclusions

Based on the performance parameters reported in the Appendix (i), it is clear that the strategy performs best when SL is -10% and TP is 20%.

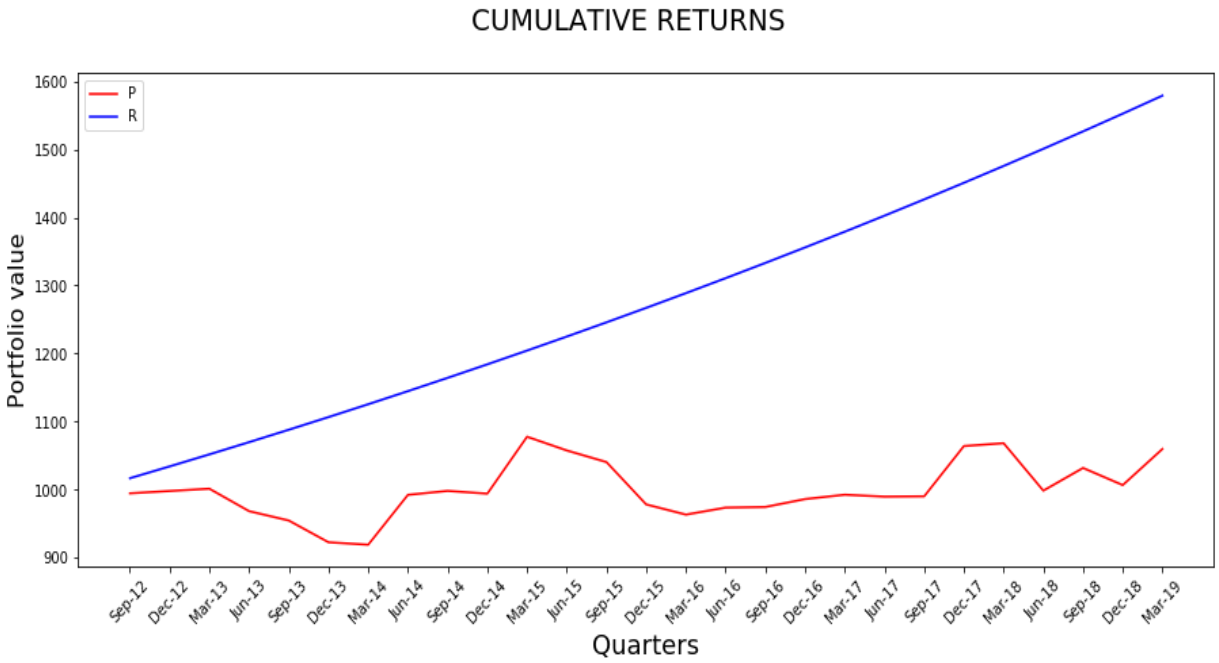
Some insights into the performance of the strategy:-

1. The portfolio formed outperforms the market(Nifty) when Nifty goes down, more than 60% of the time.
2. Over the long term the strategy **fails** to realise significant amount of gains in its current form.
3. Further, transaction costs have to be considered, which will show that the performance is even worse.

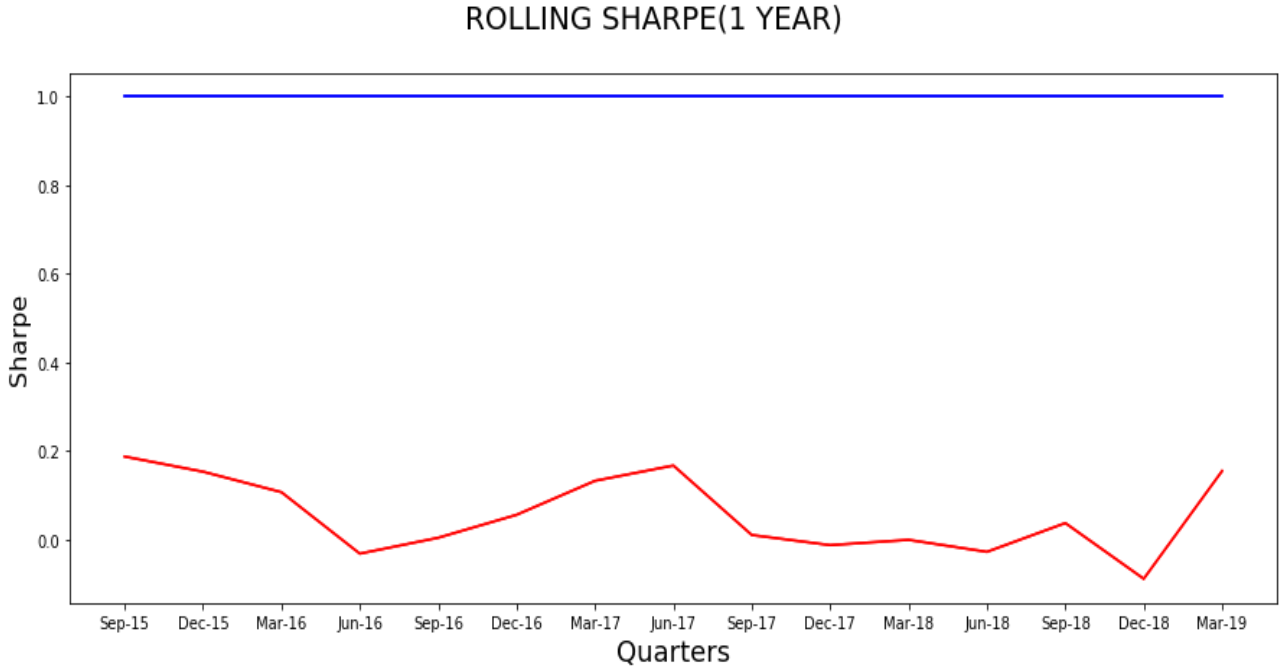
4. Based on the above observations, we conclude that **PEAD** in the way we implemented it **does not work well in Nifty 50 universe stocks in the period 2008-2019**.

The further graphs are presented taking the best possible combination of SL and TP:-

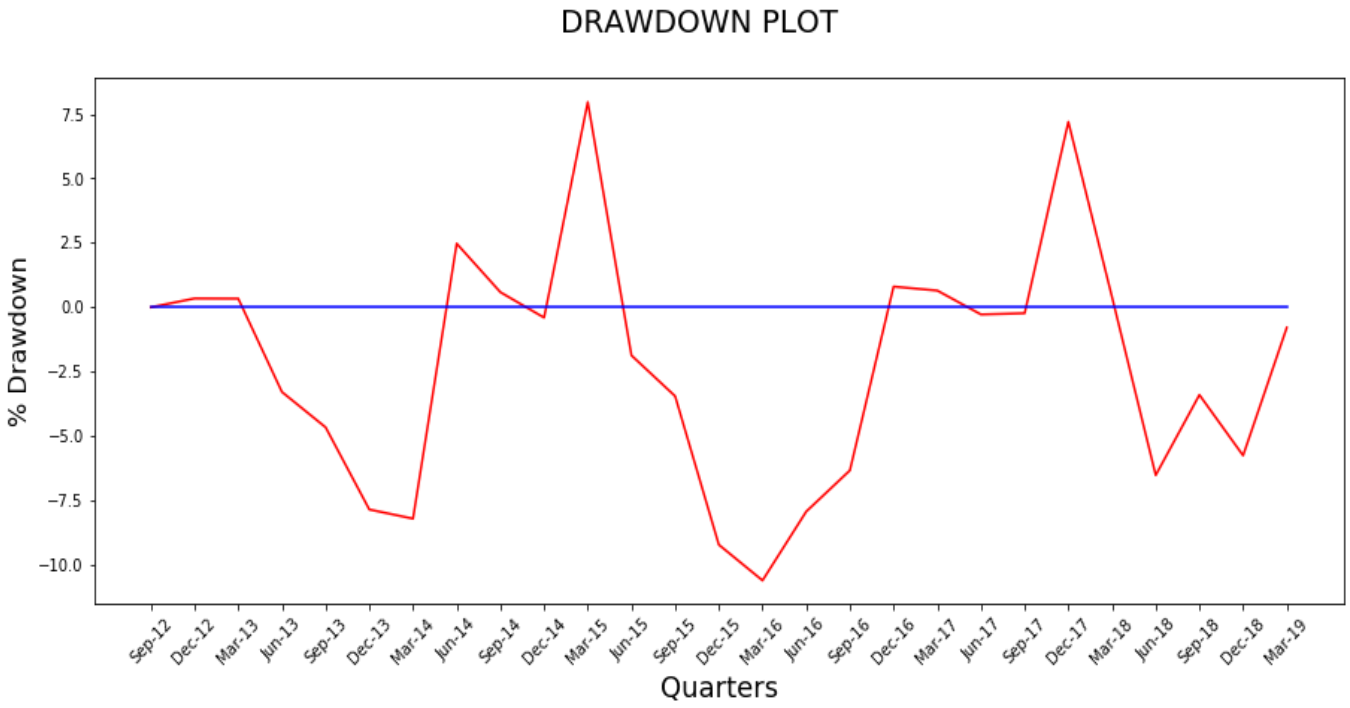
1. **Cumulative returns** graph (Portfolio in red and Risk free in blue (investing 1000 Rs. in both))



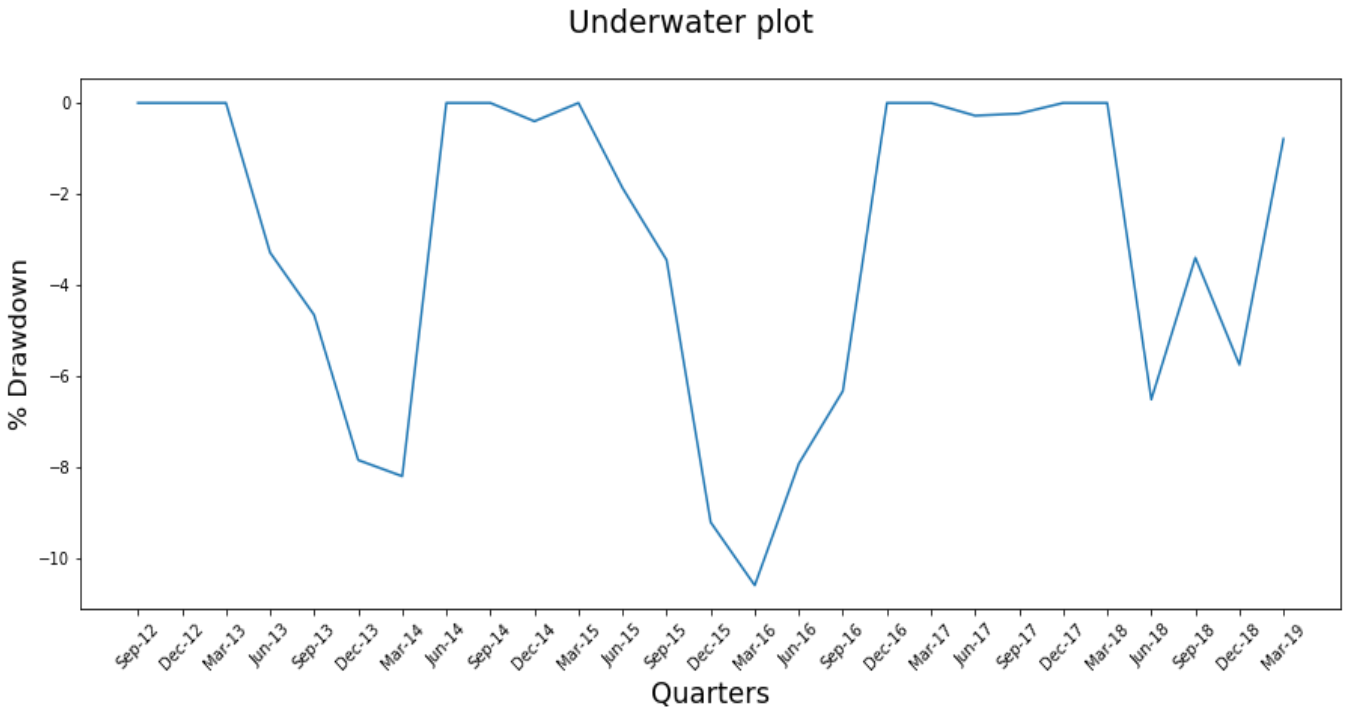
2. **1 Year Rolling Sharpe**, the overall sharpe is **-0.39**. (Red - Portfolio, Blue - Risk Free)



3. **Drawdown** (Red - Portfolio, Blue - 0% drawdown line)



4. **Underwater plot**



6. Appendix

(i) The performance of the strategy on different combination of Stop loss(SL) and Take profit(TP) are given below (The SL is varied from -5% to -15%, and the TP is varied from 10% to 20%) :-

1. When SL = -5% and TP = 10 %
    a. % of times our strategy beats the index when index goes up = 37.3%
    b. % of times our strategy beats the index when index goes down = 63.6%
    c. % of times loss is encountered on LONG trades = 50.33%
    d. % of times loss is encountered on SHORT trades = 60.13%
    e. No. of times profit less than SL = 155
    f. No. of times profit more than TP = 85
2. When SL = -10% and TP = 10 %
    a. % of times our strategy beats the index when index goes up = 39.5%
    b. % of times our strategy beats the index when index goes down = 63.6%
    c. % of times loss is encountered on LONG trades = 40.9%
    d. % of times loss is encountered on SHORT trades = 55.2%
    e. No. of times profit less than SL = 94
    f. No. of times profit more than TP = 90
3. When SL = -15% and TP = 10 %
    a. % of times our strategy beats the index when index goes up = 40.7%
    b. % of times our strategy beats the index when index goes down = 67.3%
    c. % of times loss is encountered on LONG trades = 40.3%
    d. % of times loss is encountered on SHORT trades = 53.8%
    e. No. of times profit less than SL = 49
    f. No. of times profit more than TP = 92
4. When SL = -5% and TP = 15 %
    a. % of times our strategy beats the index when index goes up = 34.6%
    b. % of times our strategy beats the index when index goes down = 60.9%
    c. % of times loss is encountered on LONG trades = 52.3%
    d. % of times loss is encountered on SHORT trades = 64.3%
    e. No. of times profit less than SL = 162
    f. No. of times profit more than TP = 41
5. When SL = -10% and TP = 15 %
    a. % of times our strategy beats the index when index goes up = 36.8%
    b. % of times our strategy beats the index when index goes down = 60.9%
    c. % of times loss is encountered on LONG trades = 42.2%

   d. % of times loss is encountered on SHORT trades = 58.7%
   e. No. of times profit less than SL = 97
   f. No. of times profit more than TP = 44
6. When SL = -15% and TP = 15 %
   a. % of times our strategy beats the index when index goes up = 37.9%
   b. % of times our strategy beats the index when index goes down = 64.5%
   c. % of times loss is encountered on LONG trades = 41.6%
   d. % of times loss is encountered on SHORT trades = 57.3%
   e. No. of times profit less than SL = 52
   f. No. of times profit more than TP = 45
7. When SL = -5% and TP = 20 %
   a. % of times our strategy beats the index when index goes up = 34.1%
   b. % of times our strategy beats the index when index goes down = 60.9%
   c. % of times loss is encountered on LONG trades = 52.3%
   d. % of times loss is encountered on SHORT trades = 64.3%
   e. No. of times profit less than SL = 162
   f. No. of times profit more than TP = 22
8. When SL = -10% and TP = 20 %
   a. % of times our strategy beats the index when index goes up = 39.5%
   b. % of times our strategy beats the index when index goes down = 63.6%
   c. % of times loss is encountered on LONG trades = 40.9%
   d. % of times loss is encountered on SHORT trades = 55.2%
   e. No. of times profit less than SL = 97
   f. No. of times profit more than TP = 23
9. When SL = -15% and TP = 20 %
   a. % of times our strategy beats the index when index goes up = 37.4%
   b. % of times our strategy beats the index when index goes down = 64.5%
   c. % of times loss is encountered on LONG trades = 41.6%
   d. % of times loss is encountered on SHORT trades = 57.3%
   e. No. of times profit less than SL = 52
   f. No. of times profit more than TP = 23

(ii) Python code for the strategy - Github link

Importing necessary libraries

```
import pandas as pd
import numpy as np
import datetime
import warnings
import matplotlib.pyplot as plt
%matplotlib inline
import math
plt.rcParams['figure.figsize']=[15,6]
from statistics import mean,stdev
```

Reading in the data

```
cutoff = pd.read_csv('sue_cutoffs.csv')
sue = pd.read_csv('sue.csv')
ann_dates = pd.read_csv('eps_nifty50_announcement_dates - Sheet1.csv')
prices = pd.read_csv('Prices_nifty50.csv')
nifty = pd.read_csv("nifty_prices.csv")
```

Dropping unnecessary columns

```
sue.drop('Unnamed: 0',inplace=True,axis=1)
sue_ticker_index = sue.copy()
sue_ticker_index.set_index('Ticker');
cutoff.rename(columns = {'Unnamed: 0':'QtrEnd'}, inplace = True)
ann_dates.drop('Company',inplace=True,axis=1)
```

Converting to standard datetime format

```
for i in ann_dates.columns.tolist()[1:]:
    #print(i)
    ann_dates[i]= pd.to_datetime(ann_dates[i],dayfirst=True)

nifty['Date']= pd.to_datetime(nifty['Date'],dayfirst=True)
```

Creating a list of quarter ends

```
qtrend_list = []
qtrend_list = cutoff['QtrEnd'].tolist()
qtrend_list.insert(0, 'Sep-19')
qtrend_list.insert(0, 'Ticker')
#qtrend_list
prices.drop(['PX_VOLUME','Name'],inplace=True,axis=1)
prices['date']= pd.to_datetime(prices['date'],dayfirst=True)
```

Removing the useless dates

```
mod_prices=prices.copy()
start=datetime.datetime(2012, 7, 1)
end=datetime.datetime(2019, 8, 31)
```

```
mod_prices = mod_prices[mod_prices['date']>=start]
mod_prices = mod_prices[mod_prices['date']<=end]

mod_prices.reset_index(inplace=True)
mod_prices.drop('index',axis=1,inplace=True)
```

Removing companies that have been traded for less number of days.

```
df_s=mod_prices.groupby('Ticker').count()
df_s.reset_index(inplace=True)

max=0
for i in range (len(df_s)):
    if df_s.iloc[i]['date']>max:
        max=df_s.iloc[i]['date']

less_dates=[]
for i in range (len(df_s)):
    if df_s.iloc[i]['date']!=max:
        #print (df_s.iloc[i]['Ticker'])
        #print (df_s.iloc[i]['date'])
        less_dates.append(df_s.iloc[i]['Ticker'])
print(less_dates)

for j in less_dates:
    mod_prices = mod_prices[mod_prices['Ticker']!=j]

for i in less_dates:
    #print (i)
    sue = sue[sue['Ticker']!=i]
```

Formatting the ann_dates

```
ann_dates.drop(ann_dates.columns.tolist()[31:],axis=1,inplace=True)
ann_dates.columns = qtrend_list
```

Forming long and short portfolio

```
#qtr end based on which to form the portfolio, example: Jun-19, Mar-19, Dec-18, Sep-18
#percentile is the long cutoff (number, not a string)


def port_form (qtr_end,percentile):
    long_cutoff = 0
    short_cutoff = 0
    #print (cutoff.iloc[2]['QtrEnd'])
    #print(qtr_end)
    for i in range(len(cutoff)):
        if cutoff.iloc[i]['QtrEnd']==qtr_end:
            long_cutoff = cutoff.iloc[i-1][str(percentile)]
            short_cutoff = cutoff.iloc[i-1][str(100-percentile)]
    long_st=[]
    short_st=[]
    for i in range(len(sue)):
        if sue.iloc[i][qtr_end]>=long_cutoff:
            long_st.append(sue.iloc[i]['Ticker'])
```

```
        elif sue.iloc[i][qtr_end]<=short_cutoff:
            short_st.append(sue.iloc[i]['Ticker'])
    return long_st, short_st
```

Function that returns the announcement date

```
def announce_date (ticker,qtr_end):
    ann_date = 0
    temp = ann_dates[ann_dates['Ticker']==ticker]
    temp.reset_index(inplace=True)
    ann_date=temp.iloc[0][qtr_end]
    return ann_date
```

Function that does the trading for a given SL and TP

```
def trade (ticker,ann_date,SL=-0.1,TP=0.2,ndays=40,posn='long'):
    #posn = position.copy()
    #print (ticker)
    nifty_start_price=0
    nifty_end_price=0
    per_return_nifty=0
    price = mod_prices[mod_prices['Ticker']==ticker]
    price.reset_index(inplace=True)

    for i in range (len(price)):
        if price.iloc[i]['date']>=ann_date:
            break
    start = price.iloc[i]['date']
    #print(start)
    end = price.iloc[i+ int(ndays)]['date']
    #print(end)

    for i in range(len(nifty)):
        if nifty.iloc[i]['Date']==start:
            nifty_start_price = nifty.iloc[i]['PX_LAST']
        elif nifty.iloc[i]['Date']==end:
            nifty_end_price = nifty.iloc[i]['PX_LAST']
    per_return_nifty = (nifty_end_price - nifty_start_price)/nifty_start_price

    price = price[price['date']>=start]
    price = price[price['date']<end]

    price.reset_index(inplace=True)
    price.drop('index',axis=1,inplace=True)

    buy_price = price.iloc[0]['PX_LAST']
    trade_end = ''
    for i in range(1,len(price)):
        if posn == 'long':
            if (price.iloc[i]['PX_LAST']-buy_price)/buy_price<=SL:
                sell_price = price.iloc[i]['PX_LAST']
                posn = 'squared off'
                trade_end = 'SLH'
                profit = (sell_price-buy_price)/buy_price
                break
            elif (price.iloc[i]['PX_LAST']-buy_price)/buy_price>=TP:
                sell_price = price.iloc[i]['PX_LAST']
                posn = 'squared off'
```

```
                trade_end = 'PB'
                profit = (sell_price-buy_price)/buy_price
                break
        if posn == 'short':
            if (-(price.iloc[i]['PX_LAST']-buy_price))/buy_price<=SL:
                sell_price = price.iloc[i]['PX_LAST']
                posn = 'squared off'
                trade_end = 'SLH'
                profit = (-(sell_price-buy_price))/buy_price
                break
            elif (-(price.iloc[i]['PX_LAST']-buy_price))/buy_price>=TP:
                sell_price = price.iloc[i]['PX_LAST']
                posn = 'squared off'
                trade_end = 'PB'
                profit = (-(sell_price-buy_price))/buy_price
                break

        if posn!='squared off':
            sell_price = price.iloc[len(price)-1]['PX_LAST']
            trade_end = 'PSO'
            if posn=='long':
                profit = (sell_price-buy_price)/buy_price
            else:
                profit = (-(sell_price-buy_price))/buy_price

    return buy_price, sell_price, profit, trade_end, per_return_nifty

qtr_end_list=ann_dates.columns.tolist()[3:-1]
long=[]
short=[]
for i in qtr_end_list:
    lo,sho = port_form (i,90)
    long.append(lo)
    short.append(sho)

long_port = dict(zip(qtr_end_list, long))
short_port = dict(zip(qtr_end_list, short))
```

Code for strategy

```
quarter = []
comp =[]
position =[]
bp=[]
nifty_return=[]
sp=[]
pnl =[]
end=[]
i=0
j=0
for qtr in long_port:
    i=i+1
    for cpy in long_port[qtr]:
        quarter.append(qtr)
        comp.append(cpy)
        a,b,c,d,e= trade (cpy,announce_date(cpy,qtr))
        bp.append(a)
        sp.append(b)
        pnl.append(c)
```

```
        end.append(d)
        nifty_return.append(e)
        position.append('long')
    print ('No. of quarters done:', i)

print ('long portfolio done')

for qtr in short_port:
    j=j+1
    for cpy in short_port[qtr]:
        quarter.append(qtr)
        comp.append(cpy)
        a,b,c,d,e= trade(cpy,announce_date(cpy,qtr),posn='short')
        bp.append(a)
        sp.append(b)
        pnl.append(c)
        end.append(d)
        nifty_return.append(e)
        position.append('short')
    print ('No. of quarters done:', j)

Final = pd.DataFrame(list(zip(quarter, comp, position, bp, sp, pnl, end, nifty_return)), columns =['Quarter', 'Ticker', 'Position', 'Pos. Open', 'Pos. Close', 'Profit', 'Pos. Status','Nifty return'])
```

Multiplying the returns by 100 to get percentage and then rounding them off to two decimal places

```
analysis = Final.copy()
analysis['Profit']= analysis['Profit']*100
analysis['Nifty return']= analysis['Nifty return']*100

analysis['Profit']= round(analysis['Profit'],2)
analysis['Nifty return']= round(analysis['Nifty return'],2)
```

Performance metrics for the strategy:-
1. when market goes up how many times strategy beats market.
2. when market goes down how many times strategy beats market.
3. when we go long, how many times we see loss
4. when we go short, how many times we see loss

```
warnings.filterwarnings('ignore')
print(len(analysis[analysis['Nifty return']>0][analysis['Profit']>analysis['Nifty return']])/len(analysis[analysis['Nifty return']>0]))
print(len(analysis[analysis['Nifty return']<0][analysis['Profit']>analysis['Nifty return']])/len(analysis[analysis['Nifty return']<0]))
print(len(analysis[analysis['Position']=='long'][analysis['Profit']<0])/len(analysis[analysis['Position']=='long']))
print(len(analysis[analysis['Position']=='short'][analysis['Profit']<0])/len(analysis[analysis['Position']=='short']))
```

Analysing very heavy losses and very great profits

```
analysis[analysis['Profit']<-10]
analysis[analysis['Profit']>20]

analysis[analysis['Nifty return']>20]
analysis[analysis['Nifty return']<-10]
```

Implementing the strategy and printing the performance metrics by varying the SL and TP.

```
stoploss = [-0.05,-0.1,-0.15]
takeprofit = [0.10,0.15,0.20]
for tp in takeprofit:
    print(tp,'done')
    TP = tp
    for sl in stoploss:
        SL = sl
        #########
        quarter = []
        comp =[]
        position =[]
        bp=[]
        nifty_return=[]
        sp=[]
        pnl =[]
        end=[]
        i=0
        j=0
        for qtr in long_port:
            i=i+1
            for cpy in long_port[qtr]:
                quarter.append(qtr)
                comp.append(cpy)
                a,b,c,d,e= trade (cpy,announce_date(cpy,qtr),SL,TP)
                bp.append(a)
                sp.append(b)
                pnl.append(c)
                end.append(d)
                nifty_return.append(e)
                position.append('long')
            if(i%5==0):
                print ('No. of quarters done:', i)

        print ('long portfolio done')

        for qtr in short_port:
            j=j+1
            for cpy in short_port[qtr]:
                quarter.append(qtr)
                comp.append(cpy)
                a,b,c,d,e= trade(cpy,announce_date(cpy,qtr),SL,TP,posn='short')
                bp.append(a)
                sp.append(b)
                pnl.append(c)
                end.append(d)
                nifty_return.append(e)
                position.append('short')
            if(j%5==0):
                print ('No. of quarters done:', j)
        print ('short portfolio done')
        dfname = pd.DataFrame()
          dfname = pd.DataFrame(list(zip(quarter, comp, position, bp, sp, pnl, end, nifty_return)), columns
=['Quarter', 'Ticker', 'Position', 'Pos. Open', 'Pos. Close', 'Profit', 'Pos. Status','Nifty return'])
```

```
                print(len(dfname[dfname['Nifty return']>0][dfname['Profit']>dfname['Nifty return']])/len(dfname[dfname['Nifty return']>0]))
                print(len(dfname[dfname['Nifty return']<0][dfname['Profit']>dfname['Nifty return']])/len(dfname[dfname['Nifty return']<0]))

print(len(dfname[dfname['Position']=='long'][dfname['Profit']<0])/len(dfname[dfname['Position']=='long']))

print(len(dfname[dfname['Position']=='short'][dfname['Profit']<0])/len(dfname[dfname['Position']=='short']))
    print(len(dfname[dfname['Profit']<sl]))
    print(len(dfname[dfname['Profit']>tp]))


analysis_long = analysis[analysis['Position']=='long']
analysis_short = analysis[analysis['Position']=='short']
```

Creating long and short quarterwise list

```
qtrwise_long = analysis_long.groupby('Quarter',sort=False).mean()
qtrwise_short = analysis_short.groupby('Quarter',sort=False).mean()



qtrwise_long = qtrwise_long.iloc[::-1]
qtrwise_short = qtrwise_short.iloc[::-1]
qtrwise = (qtrwise_long + qtrwise_short)/2



qtrwise.reset_index(inplace=True)
qtrwise.drop(['Pos. Open','Pos. Close','Nifty return'],axis=1,inplace=True)
qtrwise.rename(columns={"index": "date", "Profit": "amount_ret"},inplace=True)
qtrwise['amount_ret']=(qtrwise['amount_ret']+100)/100
```

Creating a column holding portfolio values over time

```
status_1000 = []
for i in range (len(qtrwise)):
  if i==0:
    status_1000.append(1000*qtrwise.iloc[i]['amount_ret'])
  else:
    status_1000.append(status_1000[i-1]*qtrwise.iloc[i]['amount_ret'])
qtrwise['status_1000']=status_1000
```

Risk free

```
rf_pa = 7
rf_pq = (1+rf_pa/100)**(0.25)-1
status_1000_rf=[]
```

```
for i in range (len(qtrwise)):
    if i==0:
        status_1000_rf.append(1000*(1+rf_pq))
    else:
        status_1000_rf.append(status_1000_rf[i-1]*(1+rf_pq))
qtrwise['status_1000_rf']=status_1000_rf
qtrwise.reset_index(inplace=True)


fig = plt.figure(figsize=(15,6))
ax = fig.add_subplot()
quarters = qtrwise['Quarter'].tolist()
portfolio = qtrwise['status_1000']
riskfree = qtrwise['status_1000_rf']
ax.plot(quarters, portfolio, 'r')
ax.plot(quarters, riskfree, 'b')
fig.suptitle('CUMULATIVE RETURNS', fontsize=20)
ax.legend('PR')
plt.xlabel('Quarters', fontsize=18)
plt.ylabel('Portfolio value', fontsize=16)
ax.tick_params(axis ='x', rotation = 45)

temp=qtrwise['amount_ret'].tolist()
sharpe=[]
quarters=[]
rollmax=[]
for i in range (len(qtrwise)):
    if i<12:
        continue
    else:
        ret_list=temp[i-12:i-1]
        ret_list=[j*100 for j in ret_list]
        ret_list=[k-100 for k in ret_list]
        avg=mean(ret_list)
        std=stdev(ret_list)
        #print(std)
        rf_pq = (1+rf_pa/100)**(1/4)-1
        sharpe.append((avg-rf_pq)/std)
        #print(final.iloc[i]['date'])
        quarters.append(qtrwise.iloc[i]['Quarter'])

one=[1]*len(quarters)
```

Rolling Sharpe

```
fig = plt.figure(figsize=(15,6))
ax = fig.add_subplot()
ax.plot(quarters, sharpe, 'r')
ax.plot(quarters, one, 'b')
fig.suptitle('ROLLING SHARPE(1 YEAR)', fontsize=20)
plt.xlabel('Quarters', fontsize=18)
plt.ylabel('Sharpe', fontsize=16)
#ax.tick_params(axis ='x', rotation = 45)
```

```
plt.plot(quarters, sharpe, 'r')
plt.plot(quarters,one,'b')
plt.show()
```

Overall sharpe

```
total_amt_ret=qtrwise['amount_ret'].tolist()
total_amt_ret=[j*100 for j in total_amt_ret]
total_amt_ret=[k-100 for k in total_amt_ret]
avg=mean(total_amt_ret)
std=stdev(total_amt_ret)
overall_sharpe = (avg-(rf_pq*100))/std
print(overall_sharpe)

temp=qtrwise['status_1000'].tolist()
rollmax=[]
#rollmax=rollmax.tolist()
for i in range(len(qtrwise)):
    if i==0:
        rollmax.append(temp[0])
        #print(rollmax)
    elif i<4:
        rollmax.append(np.amax(np.asarray(temp[0:i])))
        #print(rollmax)
    else:
        rollmax.append(np.amax(np.asarray(temp[i-4:i])))

Quarterly_Drawdown = (qtrwise['status_1000']/rollmax) - 1.0

Quarterly_Drawdown = Quarterly_Drawdown.tolist()
Quarterly_Drawdown=[j*100 for j in Quarterly_Drawdown]
dates1=qtrwise['Quarter'].tolist()
zero=[0]*len(dates1)


fig = plt.figure(figsize=(15,6))
ax = fig.add_subplot()
ax.plot(dates1, Quarterly_Drawdown, 'r')
ax.plot(dates1, zero, 'b')
fig.suptitle('DRAWDOWN PLOT', fontsize=20)
plt.xlabel('Quarters', fontsize=18)
plt.ylabel('% Drawdown', fontsize=16)
ax.tick_params(axis ='x', rotation = 45)



UnderWater = Quarterly_Drawdown.copy()
for i in range (len(Quarterly_Drawdown)):
    if UnderWater[i]>0:
        UnderWater[i]=0

fig = plt.figure(figsize=(15,6))
ax = fig.add_subplot()
ax.plot(dates1, UnderWater)
```

```
fig.suptitle('Underwater plot', fontsize=20)
plt.xlabel('Quarters', fontsize=18)
plt.ylabel('% Drawdown', fontsize=16)
ax.tick_params(axis ='x', rotation = 45)
```