Best_performer_Analysis_

September 6, 2024

1 Check the stock that out perform in the hypothetical portfolio

1.1 Analysing Best Performer stock in protfolio in last 3 month

1.2 portfolio includes three sector i.e Banking, Pharma, and IT (BPI)

1.3 Sectorial allocation were finalised based on the the proportion of each sector. For example if market cap of sector B,P, I are 30,15,10 respectively then 54% (30/55) in B, 27% in P and remaining 9% in I

1.4 Then Invetesting Equal weight portfolio in each sector by inveting in top 4 company by market cap

(Source https://www.icicidirect.com/equity/index/nse/nifty-pharma/26769)

[124]:
```
#!pip install fyers_apiv3 pyotp plotly
from fyers_apiv3 import fyersModel
import pyotp
import pandas as pd
import requests as req
import datetime as dt
from urllib.parse import parse_qs, urlparse
import time
#import seaborn as sea
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
```

2 Fyers Credential (Sensitive)

[125]:
```
credential = {
    'secret_key': [redacted],
    'client_id': [redacted],
    'redirect_uri': 'https://trade.fyers.in/api-login/redirect-uri/index.html',
    'FY_ID': [redacted],
    'PIN': [redacted],
    'TOTP': [redacted]
}
```

3 Login Function created and used to communicate with

[126]:
```
def login(secret_key, client_id, redirect_uri, fy_id, pin, totp):
    BASE_URL = "https://api-t2.fyers.in/vagator/v2"
    BASE_URL_2 = "https://api-t1.fyers.in/api/v3"
    URL_SEND_LOGIN_OTP = BASE_URL + "/send_login_otp"
    URL_VERIFY_TOTP = BASE_URL + "/verify_otp"
    URL_VERIFY_PIN = BASE_URL + "/verify_pin"
    URL_TOKEN = BASE_URL_2 + "/token"
    URL_VALIDATE_AUTH_CODE = BASE_URL_2 + "/validate-authcode"
    res = req.post(url=URL_SEND_LOGIN_OTP, json={"fy_id": fy_id, "app_id":␣
 ↪"2"}).json()
    if dt.datetime.now().second % 30 > 27: time.sleep(5)
    vari_topt = req.post(url=URL_VERIFY_TOTP,
                         json={"request_key": res['request_key'], "otp": pyotp.
 ↪TOTP(totp).now()}).json()

    ses = req.Session()
    time.sleep(.5)
    vari_pin_payload = {"request_key": vari_topt["request_key"],␣
 ↪"identity_type": "pin", "identifier": [redacted]}

    vari_pin = req.post(url=URL_VERIFY_PIN, json=vari_pin_payload).json()

    ses.headers.update({"authorization": f"Bearer␣
 ↪{vari_pin['data']['access_token']}"})
    token_payload3 = {
        "fyers_id": fy_id,
        "app_id": client_id[:-4],
        "redirect_uri": redirect_uri,
        "appType": "100", "code_challenge": "",
        "state": "None", "scope": "", "nonce": "",
        "response_type": "code", "create_cookie": True
    }

    token_ = ses.post(url=URL_TOKEN, json=token_payload3).json()

    url = token_['Url']
    parsed = urlparse(url)
    auth_code = parse_qs(parsed.query)['auth_code'][0]

    session = fyersModel.SessionModel(
        client_id=client_id,
        secret_key=secret_key,
        redirect_uri=redirect_uri,
        response_type = 'code',
        grant_type = 'authorization_code'
```

```
    )
    session.set_token(auth_code)
    response = session.generate_token()
    access_token = response['access_token']
    fyers = fyersModel.FyersModel(client_id=client_id, is_async=False,␣
 ↪token=access_token, log_path="")
    return access_token, fyers

access_token, fyers = login(
    credential.get("secret_key"),credential.get("client_id"),
    credential.get("redirect_uri"), credential.get("FY_ID"),
    credential.get("PIN"), credential.get("TOTP")
              )
```

4 Sectorial_allocation

[127]:
```
nifty_bank = 4059695.19
pharma =  1750884.26
i_t =   3790322.85
total =sum([nifty_bank, pharma, i_t])
nifty_bank_weight = round(nifty_bank/total,2)
pharma_weight = round(pharma/total,2)
i_t_weight = 1 - sum([nifty_bank_weight,pharma_weight])
initial_capital = 1000000
print("Sectorial Weight are as follow:")
print(f"1. Banking: {nifty_bank_weight}\n2. Pharma: {pharma_weight}\n3. IT:␣
 ↪{i_t_weight}")
```

```
Sectorial Weight are as follow:
1. Banking: 0.42
2. Pharma: 0.18
3. IT: 0.4
```

5 Get the Daily Historical Data of Top 4 firm by market cap in each sector

[128]:
```
from time import sleep

def Historical(symbol:str):
    sleep(1)
    """
    This function is used to extract historical data from Fyers Server
    parameter:
        symbol: ticker name eg:TCS datatype:string
    """
```

```
    today  = dt.date.today() #yyyy-mm-dd
    start_day = today - dt.timedelta(90)

    data = {
        "symbol":f"NSE:{symbol.upper()}-EQ",
        "resolution":"D",
        "date_format":"1",
        "range_from":str(start_day),
        "range_to":str(today),
        "cont_flag":"1"
    }
    try:
        response = fyers.history(data=data)
        #Lets strucutre the data
        data = pd.DataFrame(response['candles'], columns=["Date",␣
 ↪"Open","High","Low","Close", "Volume"])
        data['Date'] = pd.to_datetime(data["Date"], unit='s')
        return data[['Date',"Close"]].rename(columns={'Close':symbol.upper()})
    except:
        #print(response, symbol)
        None


nifty_bank_capital = round(nifty_bank_weight*initial_capital,2)
nifty_bank_portfolio_stock = ['HDFCBank','ICICIBank', 'AXISBANK', 'SBIN'] #top␣
 ↪4 company in banking sector

pharma_capital = round(pharma_weight*initial_capital,2)
pharma_portfolio_stock =  ['SunPharma','DIVISLAB','Cipla','TORNTPHARM' ] #top 4␣
 ↪firm in pharma sector

i_t_capital = round(i_t_weight*initial_capital,2)
i_t_portfolio_stock = ['TCS','INFY','HCLTech','Wipro']#top 4 firm in IT sector
print(f"Capital Allocated in each sector are as follow:\n1. Banking:␣
 ↪{nifty_bank_capital}\n2. Pharma: {pharma_capital}\n3. IT: {i_t_capital}")

#Portfolio Data
bank_portfoli_data = [Historical(a) for a in nifty_bank_portfolio_stock]
pharma_portfoli_data = [Historical(a) for a in pharma_portfolio_stock]
it_portfoli_data = [Historical(a) for a in i_t_portfolio_stock]

print(len(bank_portfoli_data), len(pharma_portfoli_data), len(it_portfoli_data))
```

```
Capital Allocated in each sector are as follow:
1. Banking: 420000.0
2. Pharma: 180000.0
3. IT: 400000.0
```

```
4 4 4
```

[129]:
```
import pprint
security_capital_allocation = {a.upper():100/4/100*nifty_bank_capital for a in␣
 ↪nifty_bank_portfolio_stock}

[security_capital_allocation.update({a.upper(): 100/4/100*pharma_capital}) for␣
 ↪a in pharma_portfolio_stock]
[security_capital_allocation.update({a.upper():100/4/100*i_t_capital}) for a in␣
 ↪i_t_portfolio_stock]

pprint.pprint(f"Capital allocation in each security in portfolio:")
security_capital_allocation
```

```
'Capital allocation in each security in portfolio:'
```

[129]:
```
{'HDFCBANK': 105000.0,
 'ICICIBANK': 105000.0,
 'AXISBANK': 105000.0,
 'SBIN': 105000.0,
 'SUNPHARMA': 45000.0,
 'DIVISLAB': 45000.0,
 'CIPLA': 45000.0,
 'TORNTPHARM': 45000.0,
 'TCS': 100000.0,
 'INFY': 100000.0,
 'HCLTECH': 100000.0,
 'WIPRO': 100000.0}
```

6 Merging

[130]:
```
portfoli_data = pd.concat(bank_portfoli_data+␣
 ↪pharma_portfoli_data+it_portfoli_data, axis=1)
portfoli_data = portfoli_data.transpose()
portfoli_data = portfoli_data.drop_duplicates().transpose()
```

[130]:
```

```

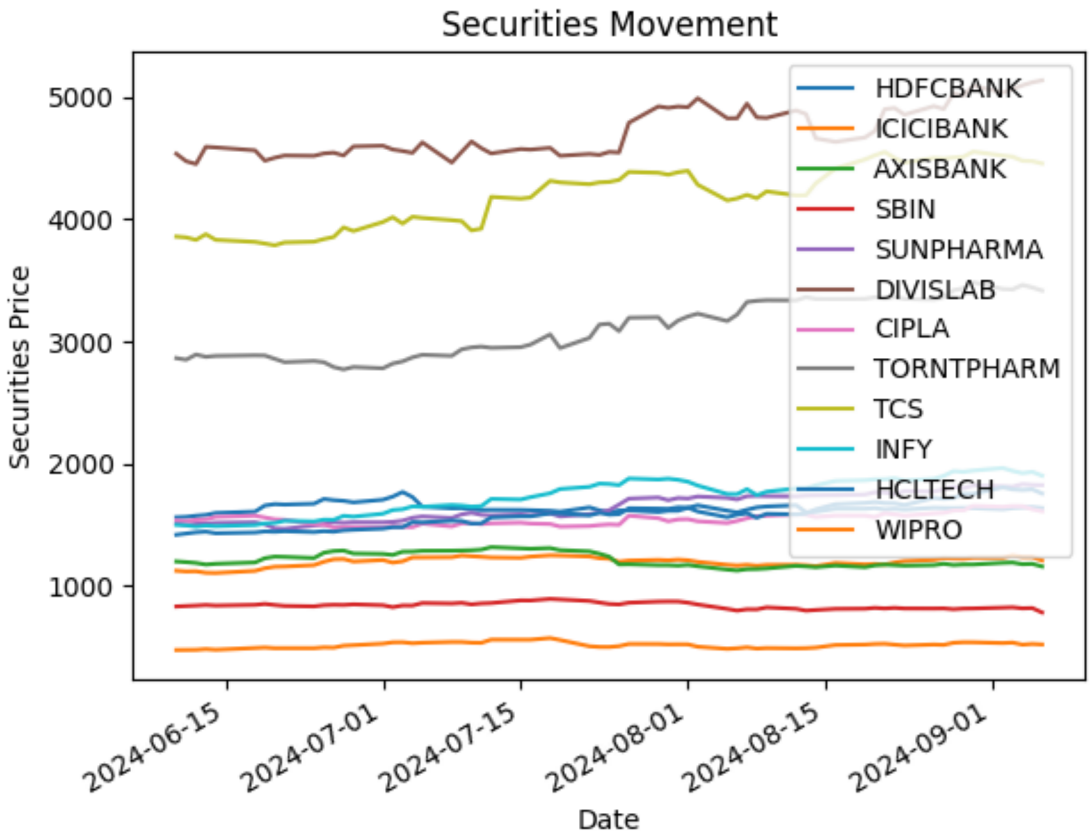
7 Few more processing

[131]:
```
portfoli_data = pd.concat(bank_portfoli_data+␣
 ↪pharma_portfoli_data+it_portfoli_data, axis=1)
portfoli_data = portfoli_data.transpose()
portfoli_data = portfoli_data.drop_duplicates().transpose()
portfoli_data.set_index(portfoli_data['Date'], inplace=True)
```

```
portfoli_data.drop(columns="Date", inplace=True)
print(portfoli_data.tail(10))
print(portfoli_data.plot(title="Securities Movement", ylabel="Securities␣
 ↪Price"))
```

```
            HDFCBANK ICICIBANK AXISBANK    SBIN SUNPHARMA DIVISLAB    CIPLA  \
Date
2024-08-26  1639.95    1213.3   1170.3  815.05   1772.45  4926.25  1593.95
2024-08-27  1637.75   1226.35  1181.25   815.9    1789.4   4902.5  1598.05
2024-08-28   1637.1   1223.85  1170.95   809.4   1811.85   5030.7   1618.2
2024-08-29  1638.55    1221.9   1175.4   814.5    1799.2  5012.55  1618.55
2024-08-30   1636.9    1229.2  1175.25   815.6   1821.65   5093.9   1654.9
2024-09-02  1626.95   1229.95   1188.8  822.15   1815.95  5036.85  1646.65
2024-09-03  1637.35    1247.7   1191.6   824.8    1811.5   5066.0   1653.2
2024-09-04   1641.8   1236.35   1177.7   816.5   1832.85   5096.6   1651.9
2024-09-05  1645.45   1235.95  1180.55  818.75    1826.5   5120.9  1627.75
2024-09-06  1636.95   1208.15  1158.75   782.5   1824.55   5137.8  1611.05

            TORNTPHARM      TCS     INFY  HCLTECH   WIPRO
Date
2024-08-26     3342.8  4502.45  1876.15  1719.45   520.0
2024-08-27     3362.0  4497.15   1900.1   1711.6  517.15
2024-08-28    3419.45  4506.05   1939.1  1719.45   534.6
2024-08-29    3434.55   4511.8  1933.35  1751.85   538.7
2024-08-30    3485.15  4553.75   1943.7  1753.25   538.4
2024-09-02    3429.75  4521.05   1964.5  1806.65  532.45
2024-09-03     3426.1  4512.35  1941.25  1790.45  536.05
2024-09-04    3462.85  4479.25  1922.45  1785.25  519.15
2024-09-05     3442.5  4475.95  1933.15  1790.55  524.85
2024-09-06     3416.7  4456.75  1901.85   1756.1   520.6
Axes(0.125,0.2;0.775x0.68)
```



8 Getting Cummulative return

[132]:
```
cummulative_return_data = pd.DataFrame()

for a in portfoli_data.columns:
    cummulative_return_data[a] = portfoli_data[a].pct_change()
    cummulative_return_data[f"{a}"] = cummulative_return_data[a].cumsum()+1

    #cummulative_return_data.drop(columns=a, inplace=True)


print(cummulative_return_data)
```

```
            HDFCBANK  ICICIBANK  AXISBANK      SBIN  SUNPHARMA  DIVISLAB  \
Date
2024-06-10       NaN        NaN       NaN       NaN        NaN       NaN
2024-06-11  1.002242   0.995017  0.995500  1.004508   0.991177  0.986597
2024-06-12  1.008217   0.996224  0.989891  1.008757   0.995911  0.981435
2024-06-13  1.012410   0.985327  0.978737  1.014477   0.998533  1.013138
```

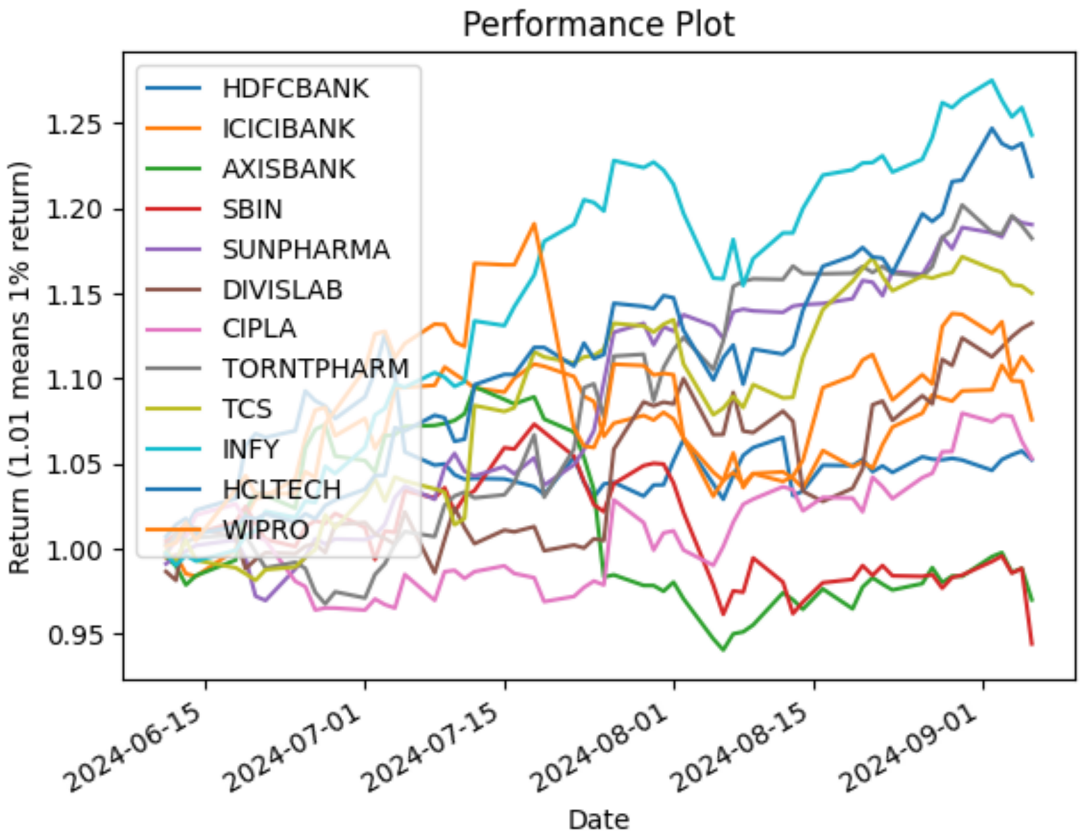
```
2024-06-14  1.022626   0.983792  0.984186  1.008908   1.001974  1.012071
…              …         …        …         …          …         …
2024-09-02  1.046034   1.093319  0.995366  0.992518   1.185539  1.112625
2024-09-03  1.052426   1.107750  0.997722  0.995741   1.183089  1.118413
2024-09-04  1.055144   1.098653  0.986057  0.985678   1.194875  1.124453
2024-09-05  1.057367   1.098330  0.988477  0.988434   1.191410  1.129221
2024-09-06  1.052201   1.075837  0.970011  0.944159   1.190342  1.132521

               CIPLA  TORNTPHARM       TCS      INFY   HCLTECH     WIPRO
Date
2024-06-10       NaN         NaN       NaN       NaN       NaN       NaN
2024-06-11  0.997393    0.995478  0.998290  0.997333  1.007084  1.001683
2024-06-12  1.004385    1.010385  0.992981  0.990280  1.014048  1.003469
2024-06-13  1.006721    1.003957  1.005117  0.996171  1.017801  1.015421
2024-06-14  1.019800    1.006462  0.993229  0.992791  1.008730  1.004853
…              …          …        …         …         …         …
2024-09-02  1.074606    1.185968  1.164316  1.275088  1.246887  1.126521
2024-09-03  1.078584    1.184904  1.162392  1.263252  1.237921  1.133282
2024-09-04  1.077798    1.195631  1.155057  1.253568  1.235016  1.101756
2024-09-05  1.063178    1.189754  1.154320  1.259134  1.237985  1.112735
2024-09-06  1.052919    1.182259  1.150030  1.242943  1.218745  1.104638

[62 rows x 12 columns]
```

[133]:
```
cummulative_return_data.plot(title = 'Performance Plot', ylabel='Return (1.01␣
↪means 1% return)')
```

[133]:
```
<Axes: title={'center': 'Performance Plot'}, xlabel='Date', ylabel='Return (1.01
means 1% return)'>
```



[134]:
```
cum_return = cummulative_return_data.iloc[-1].to_dict()
ending_portfolio_value = 0
performace = {}
for  a in cummulative_return_data.iloc[-1].to_dict().keys():



    ending_portfolio_value+= cum_return.get(a)*security_capital_allocation.
 ↪get(a)
    performace.update( {a:cum_return.get(a)*security_capital_allocation.get(a)})

ending_portfolio_value
```

[134]: 1101179.279304006

[135]:
```
performace_df = pd.DataFrame([list(performace.keys()), list(performace.
 ↪values())])
performace_df = performace_df.transpose()
performace_df.rename(columns={0:"Securities in Portfolio", 1:"Value of␣
 ↪Securities in portfiolio"}, inplace=True)
```

```
px.bar(perforamce_df, x='Securities in Portfolio', y="Value of Securities in portfiolio", title="Best Performcer")
```

[136]:

[137]: