Alioune Beye

7/14/2022

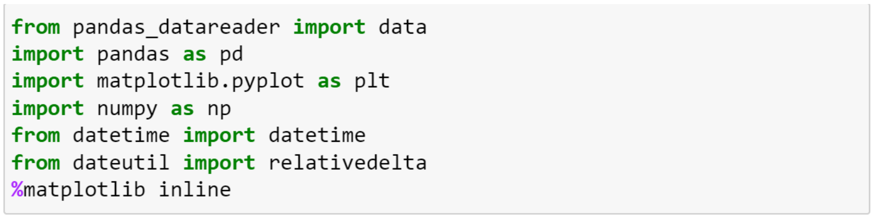
Portfolio Return Analysis

Companies that survive recessions tend to have the basic necessities needed by consumers. The reason for this is the constant consume demand of staples and food. Examples of companies that tend to fare better than the stock market (S&P 500) include the likes of Walmart and General Mills, as opposed to companies whose industries tend to be much more sensitive to economic turmoil, such as Ford and Boeing.

One way to visualize this occurrence is through the use python. Through importing the required libraries such as Pandas and Matplotlib, we can not only create functions to create all the necessary instruments, but we can also plot the returns of specific companies using real time data of their respective stock prices.

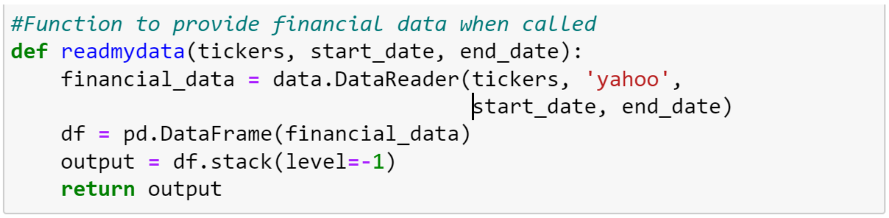
Step-By-Step Process of Portfolio Analysis Using Python:

To do so, we start with the importation of the required libraries:



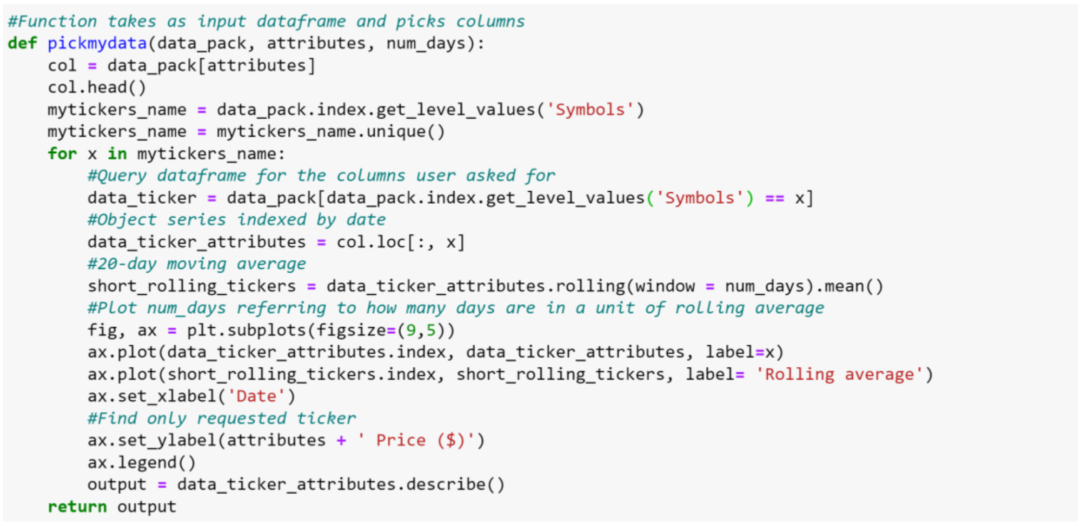
```
from pandas_datareader import data
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
from dateutil import relativedelta
%matplotlib inline
```

We then create a function to import all of the financial data we need in the form of a data frame. We also reformat the data frame to make it easier to read.



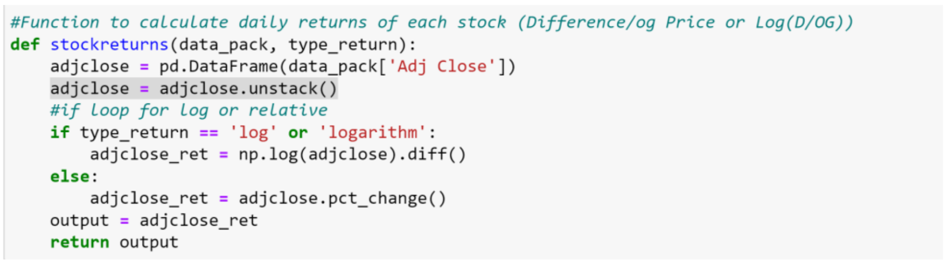
```python
#Function to provide financial data when called
def readmydata(tickers, start_date, end_date):
    financial_data = data.DataReader(tickers, 'yahoo',
                                     start_date, end_date)
    df = pd.DataFrame(financial_data)
    output = df.stack(level=-1)
    return output
```

We then create a ‘pickmydata’ function to select a user requested symbol and date range. The stock prices of the specific symbols are then plotted whenever the ‘pickmydata’ function is called.



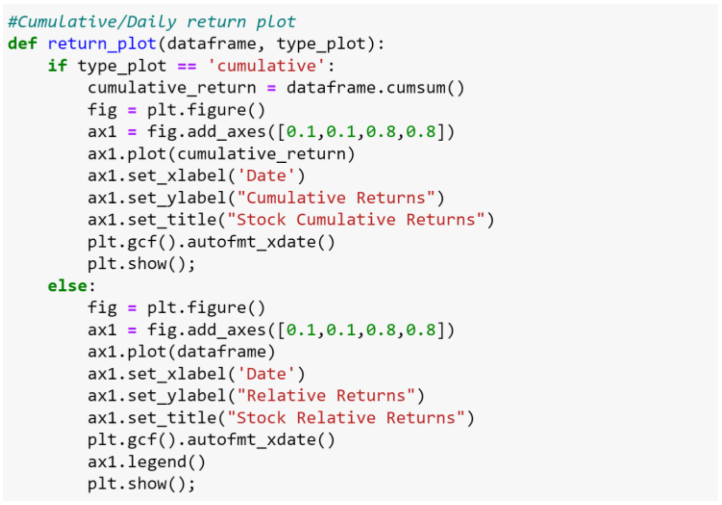
```python
#Function takes as input dataframe and picks columns
def pickmydata(data_pack, attributes, num_days):
    col = data_pack[attributes]
    col.head()
    mytickers_name = data_pack.index.get_level_values('Symbols')
    mytickers_name = mytickers_name.unique()
    for x in mytickers_name:
        #Query dataframe for the columns user asked for
        data_ticker = data_pack[data_pack.index.get_level_values('Symbols') == x]
        #Object series indexed by date
        data_ticker_attributes = col.loc[:, x]
        #20-day moving average
        short_rolling_tickers = data_ticker_attributes.rolling(window = num_days).mean()
        #Plot num_days referring to how many days are in a unit of rolling average
        fig, ax = plt.subplots(figsize=(9,5))
        ax.plot(data_ticker_attributes.index, data_ticker_attributes, label=x)
        ax.plot(short_rolling_tickers.index, short_rolling_tickers, label= 'Rolling average')
        ax.set_xlabel('Date')
        #Find only requested ticker
        ax.set_ylabel(attributes + ' Price ($)')
        ax.legend()
        output = data_ticker_attributes.describe()
    return output
```

We follow this up with a ‘stockreturns’ function, to calculate the daily returns of each stocks two different ways, one using pct_change and the other with log.



```python
#Function to calculate daily returns of each stock (Difference/og Price or Log(D/OG))
def stockreturns(data_pack, type_return):
    adjclose = pd.DataFrame(data_pack['Adj Close'])
    adjclose = adjclose.unstack()
    #if loop for log or relative
    if type_return == 'log' or 'logarithm':
        adjclose_ret = np.log(adjclose).diff()
    else:
        adjclose_ret = adjclose.pct_change()
    output = adjclose_ret
    return output
```

With our ‘return_plot’ function, we can finally plot our return dataframe. We have two different plot layouts, one with the stock prices being cumulative and the reading the stock prices independent of each other.



```python
#Cumulative/Daily return plot
def return_plot(dataframe, type_plot):
    if type_plot == 'cumulative':
        cumulative_return = dataframe.cumsum()
        fig = plt.figure()
        ax1 = fig.add_axes([0.1,0.1,0.8,0.8])
        ax1.plot(cumulative_return)
        ax1.set_xlabel('Date')
        ax1.set_ylabel("Cumulative Returns")
        ax1.set_title("Stock Cumulative Returns")
        plt.gcf().autofmt_xdate()
        plt.show();
    else:
        fig = plt.figure()
        ax1 = fig.add_axes([0.1,0.1,0.8,0.8])
        ax1.plot(dataframe)
        ax1.set_xlabel('Date')
        ax1.set_ylabel("Relative Returns")
        ax1.set_title("Stock Relative Returns")
        plt.gcf().autofmt_xdate()
        ax1.legend()
        plt.show();
```

The portfolio_ret function is used to calculate the different metrics that are useful in interpreting the performance of a particular portfolio.

```python
def portfolio_ret(dataframe, dataframe1, start_date = sd, end_date = ed ):
    #Plotting Portfolio Returns
    start_price = [1]
    weighted_df = dataframe
    ret_total = weighted_df.sum(axis=1) + start_price
    fig = plt.figure()
    ax = fig.add_axes([0.1,0.1,0.8,0.8])
    ax.plot(ret_total)
    ax.set_xlabel('Date')
    ax.set_ylabel("Portfolio Returns")
    ax.set_title("Stock Relative Returns")
    plt.gcf().autofmt_xdate()
    plt.show();
    
    #Relative Returns
    df = dataframe.stack(level=-1)
    df = df.mean(axis=0)
    df = start_price + df
    df = round(df, 6)
    value = df.values
    ret = ','.join([str(i) for i in value])

    #Annualized Returns
    d1 = sd
    d2 = ed
    start_date = datetime.strptime(d1, "%Y-%m-%d")
    end_date = datetime.strptime(d2, "%Y-%m-%d")
    delta = relativedelta.relativedelta(end_date, start_date)
    months = delta.months
    ann_ret = (([1] + df) ** (12 / months)) - 1
    ann_ret = round(ann_ret, 6)
    value = ann_ret.values
    ann_ret = ','.join([str(i) for i in value])

    #Volatility
    dif = ret_total - ret_total.mean(axis=0)
    square = dif**2
    sum = square.sum() / len(square)
    standard_dev = np.sqrt(sum)

    #Sharpe
    adjclose = data_pack['Adj Close']
    adjclose_ret = np.log(adjclose).diff()
    sharpe = data_pack['Adj Close'].mean() / data_pack['Adj Close'].std()
```

```python
    #Drawdown
    adjclose = pd.DataFrame(data_pack['Adj Close'])
    adjclose = adjclose.unstack()
    rolling_max = adjclose.cummax()
    df = (adjclose - rolling_max) / rolling_max
    df = df.stack(level=-1)
    df = df.mean(axis=0)
    drawdowns = round(df, 6)
    value = drawdowns.values
    drawdown = ','.join([str(i) for i in value])

    #Dataframe
    data = [['# of Securities', 6], ['Relative Return', ret],
           ['Annualized return', ann_ret], ['Volatility', standard_dev], ['Sharpe', sharpe],
           ['Drawdown', drawdown]]
    output = pd.DataFrame(data, columns=['Portfolio', "John Doe's Portfolio"]) #,ret_total

    return output
```

Looking at the returns of these companies, we notice that they tent to fare much better during tough financial times compared to their counterparts.

Calculating returns and visualizing the data. We are able to see a a much more consistent return in the companies deemed resilient than those whose stock price was deemed more volatile.

Looking for trends, we can see that retail company stocks would hit a dip during the recession, but the precentage of the dip would be less extreme than that of the S&P.

Companies that tend to perform terribly are those who depend on consumers being able to afford big ticket item, such as cars, housing and plane tickets. The reasons depend on the industry, for example, though the car industry is quickly becoming a necessity, are quite expensive and that alone makes the demand for them fall during economic downturns.

In order to understand which future companies are at a greater risk than average during economical downturns, it is important to look back in history and see which industries tend to suffer the most during such events.

These companies have got a dip that is either as sharp or worse than the S&P 500. Unlike the discount retail companies, the industries that these commpanies belong to rely on consumer spending large amounts of money, a feature that is stressed during economic downturns. Of course it is extremely difficult to predict what type of disasted could impact the stock market tomorrow, it is still a great exercise to understand which buisness tend to weather harsh economical storms versus those who don't.