DMP01 Lecture Summary

Overview

- This lecture will cover the back-testing and analysis of a few strategies. You will learn the following aspects of back-testing in this lecture:
    - To use two different libraries (pandas-DataReader and yfinance) to download stock data.
    - To work with and analyse intraday data while back-testing the moving average strategies (simple and exponential).
    - To work with daily data for two other trading strategies.
    - To download multiple stock data in one go.

A step-wise approach to creating and testing trading strategies

Steps:

1. Ideation of the Strategy:
    a. Come up with a trading/strategy idea. You can use the following sources for the same:
        i. Trader forums
        ii. News
        iii. Blogs
        iv. academic studies
        v. gut instinct
        vi. Analysis and Visualization of the various data points
2. Download the relevant data.
    a. Daily historical data is straightforward to obtain. Some of the sources are yfinance, nsepy, and investpy.
    b. Higher frequency data (5 minutely, minutely, etc.) for a limited period is available through yfinance. For a longer period, check with your broker.
3. Come up with the algorithm (to buy and sell)
    a. Calculate/build the indicators and statistical parameters required in the logic.
    b. Create the rules for entry and exit in the trade.
    c. Calculate returns, P&L, and other performance metrics. We will use pyfolio library to calculate the performance metrics.
    d. Program it stepwise and test each part after completion.
    e. Parameter optimization to improve the performance of the algorithm
4. Steps 1-3 are iterative in nature; you can go back and forth between these steps before you finalize the algorithm.

Directions:

1. Document the strategy logic and annotate your code with comments. It will make a traceability matrix between the logic and the code you have developed. You will not miss out on any part of the logic. And your algorithm will follow the logic. Also, it will be easier for you to understand the code whenever you will revisit the algorithm.
2. Modularize the code. Define the function having repeated code or functionality. For example, create a function to download the price data. Create functions for indicators like sma crossover, rsi etc. You can further group all the functions with similar categories in a single script, class or set of classes (will be covered in coming lectures).
3. Create Contingency plan
    - What if the code to download data automatically doesn't work from one library, the program can try accessing the other possible library to download the data?
    - Download the data manually as a CSV file and use it instead.

Strategies Covered in the lecture:

Strategy # 1: Simple Moving Average *(Long only) *:

- Determine the 12-period simple moving average (referred to as 'SMA12') and compare it with the price at that time. We (subjectively) select 12 since SMA12 would be the average price over one hour.
- There are two conditions which we check.
- If the price is greater than the SMA12, we go long. We continue to stay invested until the sell condition is satisfied.
- When the price becomes less than the SMA12, we square off our long position.
- Our trading rules can be stated as
- Buy when price > SMA12
- Square-off price < SMA12

Strategy # 1: Technical Specifications:

- Import the required libraries with the usual shorthand notations where possible
- Create a variable called end1 for the date 15th May 2020. Use the datetime library.
- Create a variable start1 which is 50 days before end1
- Use the yfinance library to download the data into a variable df for "Nifty" OHLCV data between start1 and end1.
- The data has to be at 5-minute intervals.
- The download should be into a pandas DataFrame called df
- Check the data type, the dimensions, the first few and last few rows of the pandas DataFrame
- If the above step looks fine, create a copy of df called df1a.
- In case you do not have yfinance installed, please use the CSV file and import it into pandas
- Call the pandas DataFrame df
- Check the data type, the dimensions, the first few and last few rows of df
- Create a copy of df called df1a.

Strategy # 2: Big moves on Mondays *(Long only)*

The strategy suggests that we go long on the S&P 500 on a Monday and close out our position on the Friday that week based on certain conditions. We assume that positions can be taken only in periods when markets are open on Monday and Friday in a week and the Friday in the previous week.

We calculate the following indicators and backtest the conditions shown below:

1. Calculate the 25-day average of `relative_range = (High - Low) / Close` and call it `rel_range_ma`.
2. The Monday `Close` must be lower than the previous Friday `Close` by at least 0.25 times of `rel_range_ma`.
3. Create a variable `ibs = (Close - Low) / (High - Low)`. It must be lower than 0.3.
4. If conditions in 2, and 3 are met, go long on Monday `Close`.
5. Square off your position on Friday `Close`.

Strategy # 2: Technical Specifications:

- Create a function called 'download_daily_data' where you use yfinance to automatically download daily data based on three input arguments - ticker, the start date and the end date.
- Create a function called 'compute_daily_returns' where you calculate the log daily returns based on 'Close' prices.
- You pass the pandas DataFrame as an argument to the function.
- Create three variables 'ticker2', 'end2', and 'start2'. Initialize 'ticker' to be "SPY", 'end2' to be today, and 'start2' to be the day 15 years in the past (from today).
- Use 'download_daily_data' to download SPY prices for the last 15 years into df.
- Create a copy of df called df2. We will work with df2 for the rest of the strategy.
- Use 'compute_daily_returns' to calculate daily returns of SPY into df2
- Create a function called 'compute_indicators' where you add additional columns 'day', 'prev_day', 'four_days_after', 'relative_range', 'rel_range_ma', and 'ibs' to df2 and compute them.
- Create a function called 'backtest_strategy' where you work with df2 from the previous step. You can add additional columns 'condition1', 'condition2', 'condition3' and use them to calculate strategy returns.
- Create a function called 'show_backtesting_results' where you use df2 and print the strategy returns and buy-and-hold returns.
- The function also plots the strategy returns, buy-and-hold returns and the positions over time.
- Now run all of the functions one after the other.

Strategy # 3: The Moving Average Crossover Strategy *(Long-short) *

We have two SMA filters viz. the shorter lookback period SMA (henceforth referred to as ‘SMA50') and the longer lookback period SMA (henceforth referred to as 'SMA200'). We go long on Tata Steel *at the first instance* when the SMA50 exceeds the SMA200. Similarly, we go short on it, *at the first instance* when the SMA200 exceeds the SMA50.

Our trading rules can be stated as

- Go long when SMA50 > SMA200 on a given day and SMA50 < SMA200 on the previous day
- Go short when SMA50 < SMA200 on a given day and SMA50 > SMA200 on the previous day

Strategy # 3: Technical Specifications:

- Create a variable called end3 for today. Use the datetime library.
- Create a variable start3 which is 10 years before end3
- Use the yfinance library to download daily data into a variable df for "TATASTEEL.NS" between start3 and end3.
- The download should be into a pandas DataFrame called df
- Check the data type, the dimensions, the first few and last few rows of the pandas DataFrame
- If the above step looks fine, create a copy of df called df3. (We will manipulate and work with the df3 DataFrame.)
- In case you do not have yfinance installed, please use the attached CSV file and import it into pandas
- You can name the pandas DataFrame 'df'
- Check the data type, the dimensions, the first few and last few rows of 'df'
- Create a copy of df called df3. (We will manipulate and work with the df3 DataFrame.)
- Create variables m=50 and n=200 for the shorter and longer lookback period respectively
- Create columns called 'sma50' and 'sma200' which are the moving averages based on the 'Adj Close' price
- Plot the 'sma50, 'sma200' and the 'Adj Close' for the data set
- Create columns 'sma50_prev_day' and 'sma200_prev_day' which are the moving averages shifted for the previous day
- Also periodically check df3 to see that each column is getting populated correctly.