# Trading Challenge

## Challenge Overview

This challenge is about creating and testing trading strategies for a group of 20 stocks. You'll develop algorithms to decide which strategy to use each day.

## Challenge Description

You are given the OHLC market data of 20 stocks. There are 4 tasks. Do try to attempt as many as you can as there is partial marking for each one of them.

You are provided with training data for the first 3500 days. You can test your work on the cross-validation dataset which contains the data for the next 500 days.(Please note that each number in the Date column represents a business day) We will test your work on the future unseen data. **Make sure that in no case you are using the future data anywhere to determine the weights else that submission will not be evaluated. You can only use the data till date D to determine weights for the date D+1.**

We have attached a notebook.py file with the question which must be followed. Make sure that you refer to the sample output csv formats given and follow them exactly.

## Task 1

Create these 5 strategies to assign weights to each stock each day. Make sure you exactly follow the given steps and examples in your code. In the given notebook there are functions for each strategy which you must complete. After that run the function ‘task1’ which will calculate the performances of all the strategies and generate the output file. Make sure you follow the sample weights dataframe format to be able to use the given backtester correctly.

1) **Strategy 1:**

We're looking to calculate the average weekly returns for each stock over the past year. Here's how we approach it:

- Consider a full year of trading consists of 50 weeks, and each week contains 5 business days.
- For each week, we calculate the return as follows:
  Weekly Returns = (Closing price on last day of current week - Closing price on last day of previous week) / Closing price on day day of previous week
  (Assume closing price for week 0 to be 1)
  For example, given this 11-day closing price sequence: 2,2,2,3,3,4,4,5,5,6,7
  Week 1 return (days 1-5): (3-1)/1 = 200%
  Week 2 return (days 6-10): (6-3)/3 = 100%
  Week 3 is incomplete, so we ignore it

- We perform this calculation for all 50 latest completed weeks for each stock. Then, we take the average of these 50 weekly returns to get the mean historical return for each stock.

(Each week has the same weekly return value across all its trading days. Weeks run from day [5k+1] to day [5k+5]. For instance, if you're looking at dates 306-310, the system analyzes returns from day 56 through 305 (the previous 50 weeks). Similarly, for dates 2441-2445, it examines returns from day 2191 through 2440.)

Weight Assignment:

- We rank the stocks based on their mean historical returns. (decreasing order)
- We assign equal negative weights to the top 6 performing stocks, such that their sum is -1.
- We assign equal positive weights to the bottom 6 performing stocks such that their sum is 1.
- All other stocks in between are assigned a weight of 0.

Code Function: task1_Strategy1

**2) Strategy 2:**

We're looking to compare the long-term trends to the short-term trends assuming that stocks will revert to their long term average. The steps are as follows:

- Long-term moving average (LMA): Average of the closing price of last 30 business days
- Short-term moving average (SMA): Average of the closing price of last 5 business days
- Find the relative position of the SMA with respect to LMA.
  Relative position = (SMA-LMA)/LMA
    - Example: If LMA = 5, SMA = 6, The relative position would be (6-5)/5 = +20%
    - Example: If LMA = 100, SMA = 99, The relative position would be (100-99)/100 = -1%

Weight Assignment:

- We rank the stocks based on their relative position of the SMA with respect to LMA (decreasing order)
- We assign equal positive weights to the bottom 5 performing stocks, such that their sum is 1.
- We assign equal negative weights to the top 5 performing stocks such that their sum is -1.
- All other stocks in between are assigned a weight of 0.

Code Function: task1_Strategy2

**3) Strategy 3:**

We're looking to capture the trends which have reached their peaks or lows to determine the weights. Follow these steps:

- Calculate the ROC = 100 * (Latest Closing Price – Closing Price 7 business days ago) / (Closing Price 7 business days ago)
    - Example: The closing price of a stock over the days is 1,2,3,4,5,6,7,8,9
      Therefore ROC = (9 – 2) / 2 = +3.5
    - Example: The closing price of a stock over the days is 9,8,7,6,5,4,3,2,1
      Therefore ROC = (1 – 8) / 8 = -0.875

Weight Assignment:

- We rank the stocks based on their calculated ROC (decreasing order)

- We assign equal positive weights to the bottom 4 performing stocks, such that their sum is 1.
- We assign equal negative weights to the top 4 performing stocks such that their sum is -1.
- All other stocks in between are assigned a weight of 0.

Code Function: task1_Strategy3

4) **Strategy 4:**

We are looking for Support and Resistance values. Follow the steps below.

- Resistance: 21 day Simple Moving Average of closing prices + 3 * Std Dev
- Support: 21 day Simple Moving Average of closing prices - 3 * Std Dev
- Calculate the proximity of latest closing price with the support and resistance values
  Proximity to Resistance = (Latest Closing Price - Resistance) / Resistance
  Proximity to Support = (Latest Closing Price - Support) / Support
    - Example: If Resistance = 5, Support=2, latest closing price = 4
    - Proximity to Resistance = (4-5)/5 = -20%
    - Proximity to Support = (4-2)/2 = 100%

Weights:

- We rank the stocks based on their proximity to Support (increasing order)
- We assign equal positive weights to the top 4 stocks.
- Then we rank the remaining stocks based on their proximity to Resistance (decreasing order)
- We assign equal negative weights to the top 4 stocks.
- Remains stocks are assigned 0 weights

Code Function: task1_Strategy4

5) **Strategy 5:**

Follow the steps below:

- Calculate a metric %K = 100 * (Closing Price – 14 day Low) / (14 day High – 14 day Low).
- Assign weights based on this %K value

Weights:

- Find 3 stocks with the highest %K value and assign them equal negative weights
- Find 3 stocks with the lowest %K value and assign them equal positive weights

Code Function: task1_Strategy5

## Task 2

Now you have these 5 strategies which you must combine to conquer the market. But there is a constraint. You can only use exactly one of these for any given day. Prepare an algorithm to decide which strategy to use for each day (using only the data till the previous day to prevent lookahead bias), and then

calculate the performance of your ensembled strategy. You can consider the transaction costs to be 0. Take note that there is daily compounding of notional.

Output:

- A csv of weights in the given format (generated by you)
- A csv of performance of the overall strategy (generated by backtester)
- A .pkl file for the trained model (if any) which generates these weights. This model should be plugged into your code for automated test results generation.

Code Function: task2

### Task 3

To make your ensembled strategy work in real market, we need to add transaction costs. Create an ensemble algorithm which selects which strategy to use each day considering transactions costs and calculate its performance.

Note the following changes:

- To make the problem simpler, we will use turnover. So, the more changes you make to the weights, the more you will be penalized.
- Turnover = Sum [ Abs(change of weights for each stock each day) ]
- The daily returns will be calculated like before, and total cost will be subtracted from the final PnL at the end.
- A transaction cost value of 1% will be charged per unit of turnover.
- Net returns = Returns – 0.01*Turnover

Code Functions: task3

### Documentation

- Explain your methodology, thinking and approach in detail for both the ensemble strategies
- Include comprehensive performance metrics and show how you avoided "overfitting" (making your strategy work well only on past data)
- Provide clear, well-commented Python code. Your code should be reproducible and include clear visualizations supporting your analysis.

### Submission

Submit a .zip file containing all the files (results: csv, code: py, documentation: pdf, models(if any): pkl)

### Evaluation Framework

Submissions with good out-sample score will be evaluated across six key dimensions:

a) Task-1 (15%)

b) Task-2 (20%)

c) Task-3 (20%)

e) Documentation and idea basis for ensemble strategies (25%)

f) Coding methods and clarity (10%)

g) Overall innovation and creativity (10%).

**For each task make note of the following:**

- The total of all the positive weights should be 1 for any given date.
- The total of all the negative weights should be -1 for any given date.
- Do not change the format of the csv, else it won't be evaluated.