# Order Imbalance Strategy, Machine Learning in Python

Written by Katelyn Schoenberger
Quantitative Researcher Intern, Clear Capital Group, LLC.
Summer 2018

**ABSTRACT**

Originally, this strategy was written in R by Darryl Shen, and while this strategy is indeed durable, the writing itself in the paper was rather hard to follow, given that the author used a series of former authors of similar strategies (which ones worked, which ones didn't work, etc). As the reader, I found this rather confusing, so I decided to filter out the unnecessary details and focus in on just the strategies that would be used in our machine learning model. This edited version will streamline some of the strategies as applied to our new working machine learning model written in Python.

**Objective:**

There are three factors we want to find in the data: ***Volume (VOI)***, ***Order-Imbalance Ratio (OIR)***, and ***Mid-Price Balance (MPB)*** (Equations on page 2).

**Conceptualizing Order-Imbalance:**

Classifying trader orders are either buyer-initiated or seller-initiated.

**Suggested models for machine learning:**

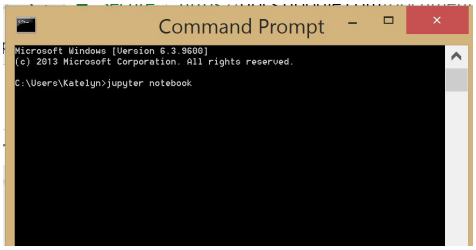
The following models that we will be using in the algorithm are SVM (Support Vector machines), Random Forest classifiers, Logistic Regression, and Linear regression. These were recommended to use by the author, Darrel Shen.

**Steps for writing code in machine learning software platform:**

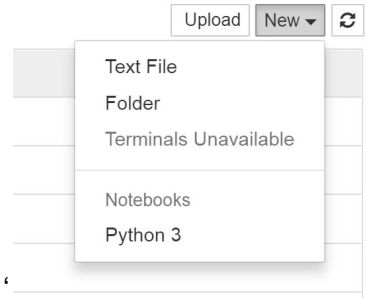
**INSTALLATION GUIDE**

**Jupyter Notebook**

- **Directions:**
  1) Install Anaconda, the package in which Jupyter comes in. For more directions, depending on your OS, follow this link:
     http://jupyter.org/install.html
  2) Open Command line (or Anaconda Prompt) and type in Jupyter notebook in first line. Hit 'Enter' once complete.




3) Choose ‘Python 3’ under Notebooks.




4) Once you’ve opened ‘Python 3’ notebook, get started by first importing your libraries in the first In[ ] cell. Hit ‘Run’ once complete. If compilable, then it will say In [ 1 ] on the left. Then, import your data set. Once the data set is imported correctly, it’ll say Out [ # ] See sample below as a reference:

```
In [1]: import pandas as pd
        import numpy as np

In [2]: pwd
Out[2]: 'C:\\Users\\Katelyn'

In [6]: cd /Users/Katelyn/Documents
        C:\Users\Katelyn\Documents

In [7]: pd.read_csv("B3.csv")
```

Out[7]:

| | Instrument | Date | TimeStamp | Bid | Ask | BidQty | AskQty |
|---|---|---|---|---|---|---|---|
| **0** | DI1F19 | 6/19/2018 | 7:00:32 | 7.13 | 7.2 | 550 | 55 |
| **1** | DI1F19 | 6/19/2018 | 7:00:33 | 7.13 | 7.2 | 550 | 55 |
| **2** | DI1F19 | 6/19/2018 | 7:00:34 | 7.13 | 7.2 | 550 | 55 |
| **3** | DI1F19 | 6/19/2018 | 7:00:35 | 7.13 | 7.2 | 550 | 55 |
| **4** | DI1F19 | 6/19/2018 | 7:00:36 | 7.13 | 7.2 | 550 | 55 |
| **5** | DI1F19 | 6/19/2018 | 7:00:37 | 7.13 | 7.2 | 550 | 55 |
| **6** | DI1F19 | 6/19/2018 | 7:00:38 | 7.13 | 7.2 | 550 | 55 |
| **7** | DI1F19 | 6/19/2018 | 7:00:39 | 7.13 | 7.2 | 550 | 55 |
| **8** | DI1F19 | 6/19/2018 | 7:00:40 | 7.13 | 7.2 | 550 | 55 |
| **9** | DI1F19 | 6/19/2018 | 7:00:41 | 7.13 | 7.2 | 550 | 55 |
| **10** | DI1F19 | 6/19/2018 | 7:00:42 | 7.13 | 7.2 | 550 | 55 |

**DEVELOPER**

Link to the scripts from GitHub is down below:
https://github.com/kschoenberger1/Machine-Learning-Trading-model

**Order Imbalance**

The difference between the **bid** and **ask** volume is referred to as order imbalance. The relationship between order imbalance and mid-price changes are hopefully ways to predict future price changes. For the purpose of this strategy, we are going to enter into a long position when the order imbalance is positive and a short position when order imbalance is negative.

**VOI, Volume formula:**

$$OI_t = \delta V_t^B - \delta V_t^A$$

where

$$\delta V_t^B = \begin{cases} 0, & P_t^B < P_{t-1}^B \\ V_t^B - V_{t-1}^B, & P_t^B = P_{t-1}^B \\ V_t^B, & P_t^B > P_{t-1}^B \end{cases}, \qquad \delta V_t^A = \begin{cases} V_t^A, & P_t^A < P_{t-1}^A \\ V_t^A - V_{t-1}^A, & P_t^A = P_{t-1}^A \\ 0, & P_t^A > P_{t-1}^A \end{cases}$$

Order of Imbalance of time = Volume of Bid over time - Volume of Ask over time

***Where:***

Volume of Bid over time = 0 if the Price of the bid over time < Price of bid over time - second ago

Volume of Bid over time = Volume of bid over time - Volume of bid over time - second ago,
Price of bid over time = Price of Bid over time - 1
Volume of Bid over time = Volume of bid over time, Price of Bid over time > Price of Bid over time - 1

AND:

Volume of Ask over time = 0 if the Price of the Ask over time < Price of Ask over time - second ago

Volume of Ask over time = Volume of Ask over time - Volume of Ask over time - second ago,
Price of Ask over time = Price of Ask over time - 1
Volume of Ask over time = Volume of Ask over time, Price of Ask over time > Price of Ask over time - 1

## Choosing the data

Build non-linear model using previous business day's data and use forecasting window (*k*) to project how long into the future we want to forecast. In this case, we are using previous day's data to predict the mid-price change for the current trading day.

Given the following equations, we want to factor these in when sorting through the data:

**Ratio (VOR):**

$$\rho_t = \frac{V_t^B - V_t^A}{V_t^B + V_t^A}$$

**Price over time** = Volume of Bid over time - Volume of Ask over time
Volume of Bid over time + Volume of Ask over time

**Calculating the mid-price in mean reversion strategy (MP)**

$$\overline{TP}_t = \begin{cases} M_1, & t = 1 \\ \frac{1}{300}\frac{T_t - T_{t-1}}{V_t - V_{t-1}}, & V_t \neq V_{t-1} \\ \overline{TP}_{t-1}, & V_t = V_{t-1} \end{cases}$$

***Where***

Time x Price over time = Mid-price1, time = 1
1/300 Time (time) - Time (time) - 1 / Volume over time - Volume (time) - 1,
Volume (t) doesn't equal Volume (time) - 1

Time x Price (time-1), Volume(time) = Volume (t-1)

After loading the data into the dataframe (Df.read_csv("File") format,

**Composite features:**

- Confusion matrix: measures the performance of the classifier models and their accuracy in correspondence to the data w/ their classifiers
- Heat mapping: represents data as colors

**Classification models (Supervised Learning):**

- SVM
    - Analyzes data for regression analysis and classification
- Logistic Regression
    - Statistical model used to apply to a binary dependent variable
- Linear Regression
    - Measures relationship in scalar responses and other variables
- Random Forest
    - fits a number of decision tree classifiers and uses averages to improve the accuracy and prevent over-fitting
- Decision tree
    - Uses tree-like graphs to measure potential outcomes
- Confusion matrix
    - Describes the performance of the classification model being utilized

When deciding which classifiers to use, the most important thing to keep in mind is to make sure the data you are using is correct. Before you do anything else, make sure your findings are as accurate as possible and that you have put the data into a dataframe library. Here we are using pandas (pd) and numpy (np), as well as sklearn. One you've split the data into a testing and a training set (80-20 ratio, 50/50, 70/30); this doesn't matter so much in terms of how you split the data, but it's important to split it to prevent overfitting.

When choosing which classification models to run it through, be sure that you are choosing the ones that end with the highest accuracy rates. We are aiming to have this model at 90-100% accuracy rate to ensure it runs efficiently and is able to cooperate with the data properly.