Pairs Trading of VISA and MasterCard Project Summary for ML for Finance by Dhyey Mavani

April 23, 2023

1 Pairs Trading of VISA (V) and MasterCard (MA)

Dhyey Dharmendrakumar Mavani, ddm2149, Visiting Student at Columbia University in Spring 2023

This is an individual project done for Machine Learning for Finance course at Columbia University during Spring 2023 semester under the guidance of Professor Renzo Silva.

1.0.1 1. Introduction and Scope

I have been really intrigued by the pairs trading strategy implemented by various hedge funds in the early 1990's. In this project, I would like to explore how I can implement this Pairs Trading strategy for VISA (Ticker Symbol: "V") and MasterCard (Ticker Symbol: "MA") in Python using the elements of Machine Learning for Finance that I have learned throughout this course.

The project aims to implement a pairs trading strategy in Python using a combination of techniques such as simple "buying the loser", logistic regression, and decision trees. The project's ultimate goal is to develop a robust and profitable pairs trading strategy that can be used to generate consistent returns in the financial markets.

As a quantitative finance enthusiast and a budding quantitative analyst, I have always been interested in developing and implementing trading strategies that can generate consistent returns in the financial markets. Pairs trading is a popular and widely used strategy that has been shown to be profitable in various market conditions. By pursuing this project, I aim to develop a deeper understanding of pairs trading and its underlying principles, and to develop a profitable and robust pairs trading strategy that can be used in real-world trading scenarios atleast for inspiration if not for deployment into the real-time financial markets.

1.0.2 2. Methodology Summary

Algorithms used: simple "buying the loser", logistic regression and decision trees.

The project will focus on developing and testing a pairs trading strategy using historical market data from a range of time frames for tickers V (VISA) and MA (MasterCard). The project will involve the following steps for each strategy:

1. **Data collection:** Collecting historical market data for VISA and MasterCard using the yfinance API in Python.

2. **Data cleaning and preprocessing:** Cleaning and preprocessing the data to remove any missing or erroneous data points, and preparing the data for analysis.
3. **Feature engineering:** Selecting and engineering a range of features that can be used to identify pairs of financial instruments that are suitable for pairs trading.
4. **Model development:** Developing and testing various pairs trading models, including models that use buying the loser and logistic regression to identify trading signals.
5. **Backtesting:** Backtesting the pairs trading strategy on historical market data to evaluate its performance and profitability.

More detailed summary of the approach undertaken for model development can be found in the "Project Results" section. Also, please feel free to refer to the code in the Jupyter notebook separately uploaded, which runs all the analysis step by step and has extensive documentation associated to it.

1.0.3 3. Description of the Data

The yfinance API in Python is a powerful and easy-to-use tool for accessing financial data from Yahoo Finance. This API allows users to retrieve a wide range of market data, including historical price data, current market data, and fundamental data for various types of financial instruments, such as stocks, ETFs, mutual funds, and currencies.

To use the yfinance API, Python developers can simply install the yfinance package and import it into their code. They can then use a range of functions and methods provided by the API to retrieve the data they need. For example, they can use the "download" function to retrieve historical price data for a given stock or other financial instrument, specifying the start and end dates for the data they want to retrieve. Similarly, they can use the "info" function to retrieve fundamental data for a given stock, such as its market capitalization, earnings per share, and dividend yield.

The yfinance API also supports a range of other features and options, such as the ability to retrieve data for multiple stocks at once, to specify the frequency of the data (e.g. daily, weekly, monthly), and to adjust the data for factors such as stock splits and dividends.

Overall, the yfinance API in Python provides a flexible and powerful way to access financial data from Yahoo Finance, making it a valuable tool for a wide range of financial analysis and trading applications. By using this API, developers can easily integrate financial data into their Python projects and build sophisticated financial models and algorithms. For this project, we will be extracting the price data of the tickers "V" and "MA" for the purposes of this project.

1.0.4 4. Project Results

a) Simple Buying the Loser Strategy I started with initial testing of this strategy of buying the loser, and got to know that it is not as profitable as I thought, but it can be treated as a decent strategy satistying the pairs trading line of thought.

This made me pursue more rigorous testing of this strategy with and without transactional costs in mind. After backtesting, I realized that the "Buying the Loser" strategy lead to a mean return of 0.04% with a standard deviation of 1.48% and a median return of 0.27%.

When I plotted the histogram of returns, this looked convincing compared to the "Baseline" strategy of Buying and holding 50/50 percentage of both V and MA. But, as soon as I plotted the cumulative returns of these strategies, I realized that the returns for “Buying the Loser” strategy stays around 0, unlike the Baseline strategy of 50/50 Buy and Hold Strategy.

b) Logistic Regression Strategy After the initial testing of the Logistic Regression Strategy, I noticed that this strategy lead to a really low PnL even without the features such as carefully curated signals and transactional cost models are incorporated, so I decided to not implement this model in depth as it would perform worse than a simple model which we already have.

c) Decision Tree Regressors Strategy Because pairs trading at its core is exploiting the information of correlation to make predictions of the future movements of a pair of tickers in the same or opposite direction, I decided to use decision trees to achieve this goal. After implementing one decision tree, I quickly realized that I should try to implement an ensemble of decision trees for this strategy.

I implemented this strategy by engineering some features from the historical data so that we can use them to extract specific signals from them with the help of the implemented decision trees. Since this was the case of time-series data, we cannot randomly select our training and testing entries, so I decided to run training and testing in batches where I train for 189 days (= 3/4 of a trading year of 252 days) before testing for 63 days (= 1/4 of a trading year of 252 days.)

The results indicated that the Baseline returns had a mean of around 0.013%, standard deviation of around 1.3% and median of 0.067%. On the other hand, the Pairs Trading Algorithm implemented with the underlying model of a set of Decision Trees lead to returns with mean around 0.1%, standard deviation of 1.5% and median of 0.107%.

Finally, even though the histogram of Baseline and Decision Trees Pairs Trading strategy look nearly the same, we can see that the the cumulative returns for the Decision Trees pairs trading strategy are positive for the most of the time, unlike that of the Baseline strategy, which goes to be heavily negative.

1.0.5 5. Conclusion and Future Work

Although we can see that the pairs trading algorithm with a set of Decision Trees as underlying model produced best returns, we did not get returns much superior in mean, standard deviation or median compared to other strategies including the baseline strategy and “Buying the Loser” strategy. Please note that further significant improvements to the Decision Trees Pairs Trading strategy can be done by fine tuning the hyperparameters, which can be done as an extension to this project.

In conclusion, I would say it makes sense why finishing in the money in today’s financial markets is so difficult. We can see the returns from the various strategies we have tried above are very low (<10%). I would love to explore some more complex ML techniques such as Neural Networks, Reinforcement Learning and other Ensemble techniques because I think they might give us better returns while backtesting. Please note that it is very different to deploy a trading strategy compared to backtesting it on the data at hand because there can be a lot of factors that may cause us friction and additional investment along the process.

I think hedge funds and quantitative trading firms are investing heavily into maintaining and building the infrastrcuture to support the successful and reliable deployment of these kinds of trading strategies. So, I think overall being successful on certain parts of the financial markets can be reduced to a race of quantitative modeling and technology among the competitors. It may very well be the case that we are not getting good return results even though our strategy is reasonable because this strategy is well known by this time and hence we are facing the classical problem of Adverse Selection.