



Pairs Trading of VISA and MasterCard

Dhyey Mavani (ddm2149)

Visiting Student at Columbia University, Spring 2023

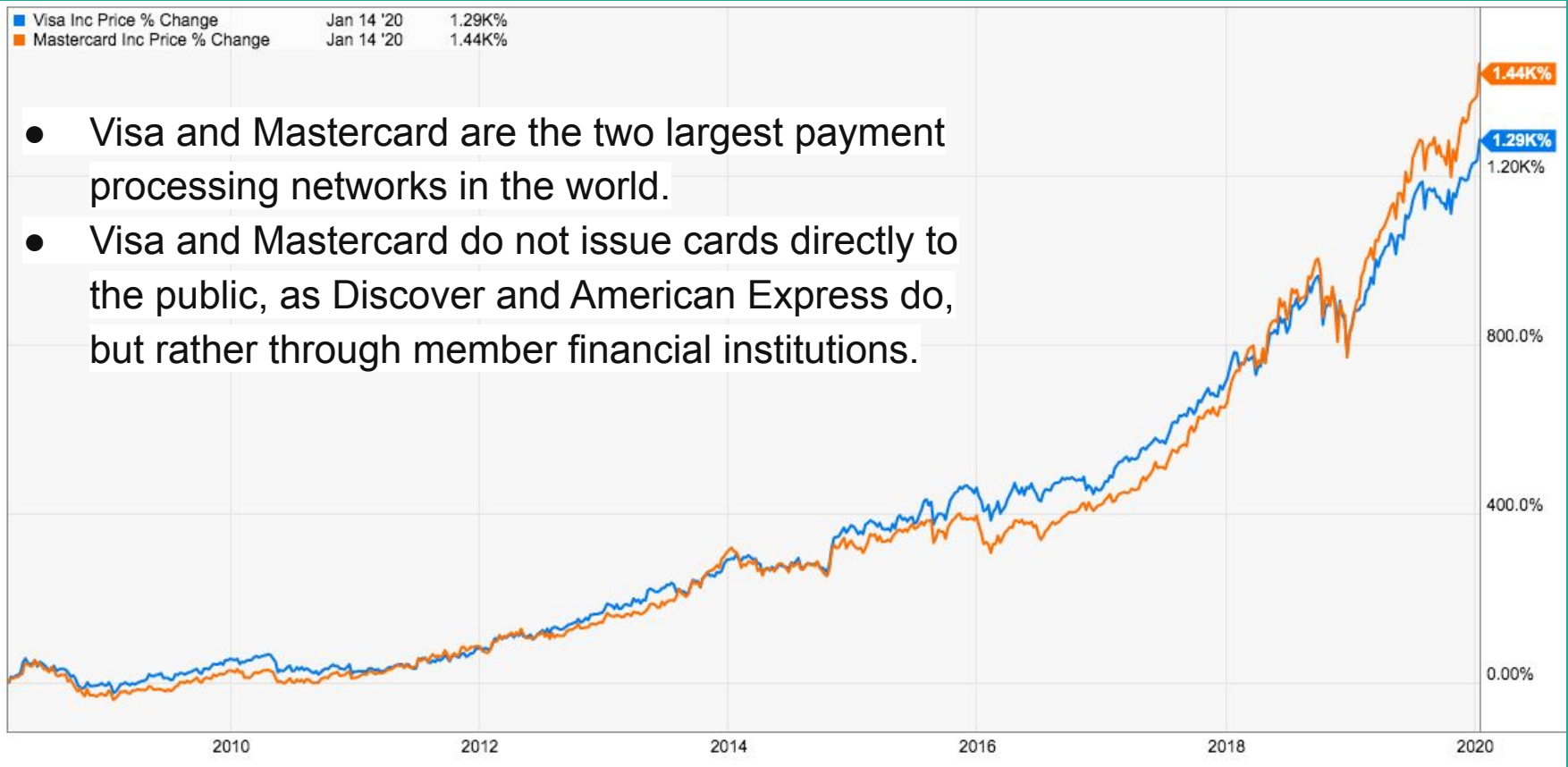
(Sophomore triple majoring in Math, CS and Stats @ Amherst College)

Why VISA and MasterCard?


■ Visa Inc Price % Change Jan 14 '20 1.29K%

■ Mastercard Inc Price % Change Jan 14 '20 1.44K%

- Visa and Mastercard are the two largest payment processing networks in the world.
- Visa and Mastercard do not issue cards directly to the public, as Discover and American Express do, but rather through member financial institutions.





Data



[Help](#) [Sponsors](#) [Log in](#) [Register](#)

yfinance 0.2.18


`pip install yfinance`


 [Latest version](#)


Released: Apr 16, 2023

Download market data from Yahoo! Finance API


Navigation

 [Project description](#)

 [Release history](#)


 [Download files](#)


Project links


 [Homepage](#)


Statistics

GitHub statistics:

 Stars: 9369

 Forks: 1872

 Open issues: 200

 Open PRs: 17



Project description

Download market data from Yahoo! Finance's API

*** IMPORTANT LEGAL DISCLAIMER ***

Yahoo!, Y!Finance, and Yahoo! finance are registered trademarks of Yahoo, Inc. yfinance is not affiliated, endorsed, or vetted by Yahoo, Inc. It's an open-source tool that uses Yahoo's publicly available APIs, and is intended for research and educational purposes. You should refer to Yahoo!'s terms of use ([here](#), [here](#), and [here](#)) for details on your rights to use the actual data downloaded. Remember - the Yahoo! finance API is intended for personal use only.

python 2.7, 3.6+ pypi v0.2.18 status beta installs 644k/month build not found codefactor -

 Star 9.4k  Follow

yfinance offers a threaded and Pythonic way to download market data from [Yahoo!® finance](#).

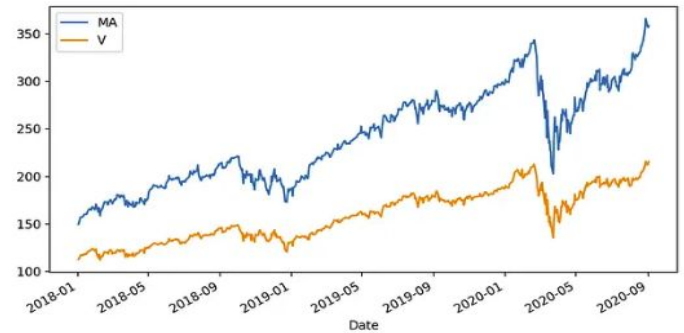
→ Check out this [Blog post](#) for a detailed tutorial with code examples.

[Changelog »](#)

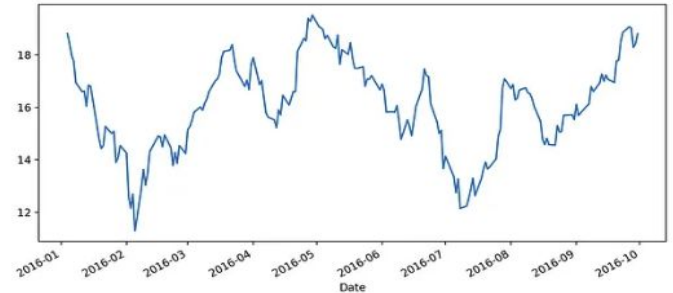
Pairs Trading

Reference: Medium Article

<https://medium.com/@mikayil.majidov/pairs-trading-python-template-strategy-review-e22e8def539>



You don't need to calculate the correlation coefficient here to understand that the assets are co-moving. Now, how do we understand that the co-movement is broken? Spread! We look at the spread between the stocks and expect it to be reverting around a certain mean value!



Spread between Mastercard and Visa

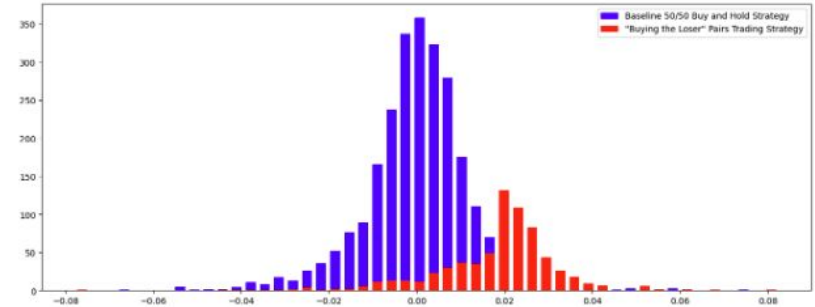
Methodologies Explored

Please note that there are many more methods that can be used! I was just unable to explore a lot because of the time-constraints.

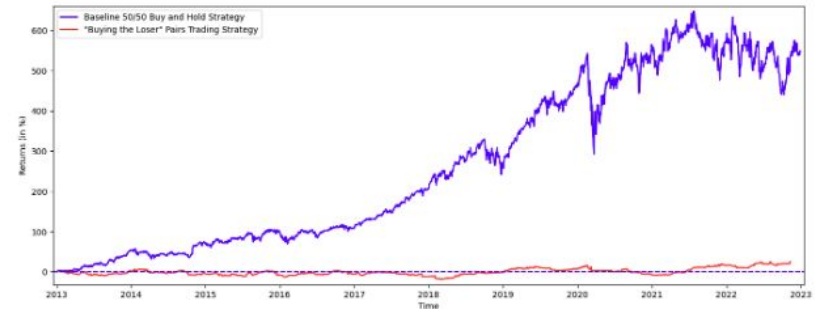
- Just Buy and Hold Baseline without Pairs Trading
 - Buying the Loser
 - Logistic Regression
 - Decision Trees
 - Further Work
-

“Buying the Loser” Strategy

1. Whenever the companies depart from their mutual equilibrium in terms of spread, they should catch up in the next period.
2. Company which is currently valued less than expected (according to z-score band of spread) should be valued more next day.
3. Simple “Buy low sell high” logic
4. If demand for credit cards increase, stocks of both companies go up.



```
##### Summary: Returns using "Buying the Loser" Pairs Trading Strategy #####  
Mean return = 0.04 %  
Standard deviation of the return = 1.48 %  
Minimum return = -9.89 %  
Maximum return = 6.65 %  
Lower quantile of return = -0.5 %  
Median return of return = 0.27 %  
Upper quantile of return = 0.8 %  
#####
```



Logistic Regression Strategy

1. Calculate the spread and normalize using z-scores
2. Train a Logistic Regression model on if it increases or decreases in past if spread is greater than the mean spread.
3. Trade accordingly by being Long on one and Short on the other.
4. Calculate the PnLs

```
import yfinance as yf
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Downloading historical data
symbol1 = yf.Ticker('V')
symbol2 = yf.Ticker('MA')

df1 = symbol1.history(period='10y')
df2 = symbol2.history(period='10y')

# Creating features and labels
spread = df1['Close'] - df2['Close']
mean_spread = spread.mean()
std_spread = spread.std()
z_score = (spread - mean_spread) / std_spread

features = []
labels = []

for i in range(1, len(df1)):
    feature = [z_score[i - 1]]
    label = 0

    if spread[i] > mean_spread:
        label = 1

    features.append(feature)
    labels.append(label)

# Training the model
X = pd.DataFrame(features, columns=['feature_{}'.format(i) for i in range(len(features[0]))])
y = np.array(labels)

model = LogisticRegression(max_iter = 10000)
model.fit(X, y)

# Making predictions
feature_today = [z_score[-1]]
X_today = pd.DataFrame([feature_today], columns=['feature_{}'.format(i) for i in range(len(feature_today))])

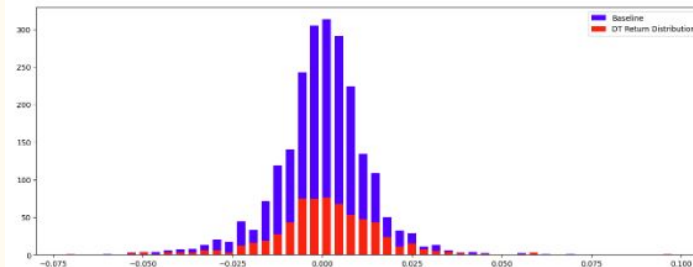
prediction = model.predict(X_today)[0]
probabilities = model.predict_proba(X_today)[0]

# Calculating P&L
symbol1_shares = 100
symbol2_shares = -prediction * (symbol1_shares * df1['Close'][-1]) / (df2['Close'][-1])
pnl = (symbol1_shares * (df1['Close'][-1] - df1['Close'][-2])) + (symbol2_shares * (df2['Close'][-1] - df2['Close'][-2]))
```

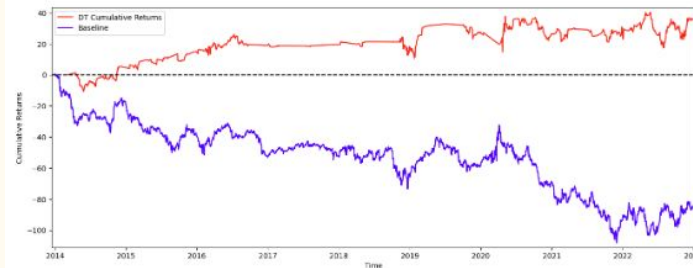
Decision Trees Strategy

1. Feature Engineering: Is_V (was V yesterday's "loser"?), ATR (Average True Range), V_Open and MA_Open.
2. 10 x (0.75/0.25 yr training/testing) Decision Trees, maintain returns array.
3. Selected Hyperparameters at Random for now!
4. Plot the resulting returns distribution, cumulative returns and summary statistics.
5. Can be significantly improved if more effort into feature engineering using alternative datasets and creativity of quant researchers.

```
##### Summary of Baseline Returns #####
count      2268.000000
mean        0.000128
std         0.013081
min         -0.071808
25%         -0.006122
50%         0.000673
75%         0.006893
max         0.097953
Name: Baseline, dtype: float64
#####
```

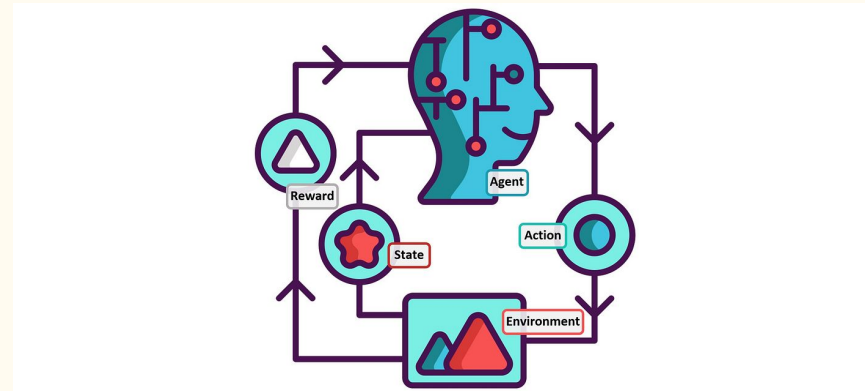
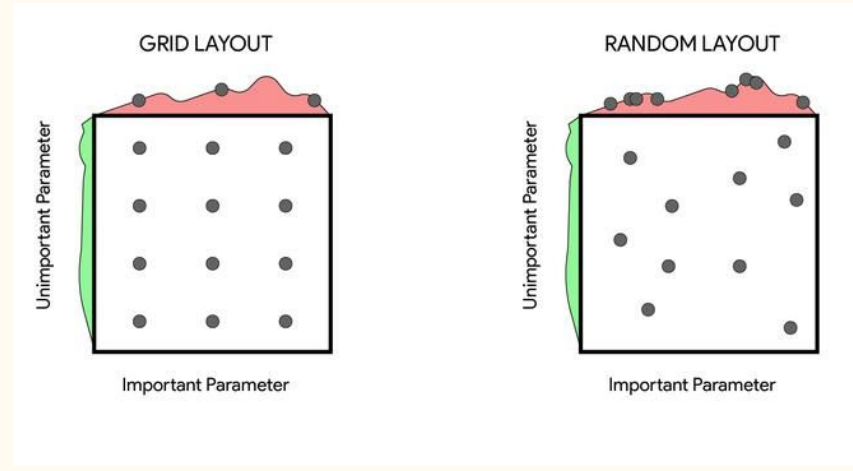


```
##### Summary of Decision Tree Returns #####
count      659.000000
mean        0.001043
std         0.015633
min         -0.071808
25%         -0.006514
50%         0.001074
75%         0.009661
max         0.097953
Name: DT, dtype: float64
#####
```



Further Work

1. Hyperparameter tuning can be done on Decision Trees.
2. Neural Networks
3. Reinforcement Learning
4. Ensemble methods
5. Tests like Cointegration
6. Using Kalman Filters



Reference: GeeksForGeeks and Towards Data Science

Final Thoughts

Some problems that we might face if we want to deploy this strategy...

- Adverse Selection
- Transactional Costs Accounting
- Imperfect Execution of orders unless you are a designated market-maker

The End + [BONUS] Meme

Thank you so much for your attention to this presentation!

I look forward to exploring this topic further over the summer in my free time!

Here is a Bonus Meme just for fun! →

Reference: <https://imgflip.com/i/1ezuu0>

