Graph-based market contagion detector

Aadhav Vijay

1 · PROBLEM

what and why

Investors usually monitor individual stocks in isolation. However, market shocks tend to propagate through networks of economically connected companies rather than affecting a single company. By the time this is visible, money has already been lost, and much opportunity has already passed. This project tests whether tracking the market as a whole can help mitigate the amount of money lost through the use of correlation-weighted graphs and tracking volatility shocks spread throughout the rest of the market.

2 · APPROACH

model / method, and why this one

The main hypothesis is that a stock's near-term shock risk depends not solely on its recent behavior, but the shock status of a correlated neighbor must also be factored in. 1. First off, the model computes the rolling volatility of each stock and compares it with its own stock's historical percentile threshold. If the volatility exceeds this cutoff, then that day will be labeled a "shock" day. 2. Secondly, instead of just looking at one stock, it looks at the whole market to see which stocks move together on certain days. Additionally, it builds a map to be able to visualize it; the closer the dots and the more lines connecting two stocks mean they move in a similar manner. 3. Thirdly, the model looks at the selected stock's connected neighbors and evaluates how many of them are currently having a shock day on the given day. The closer a stock is, the more heavily it is weighted. This number gives us our "neighbor shock exposure" a. For example: let's say we are analyzing stock A and it currently has three neighbors: stock B with a correlation of 0.8, stock C with a correlation of 0.6, and stock D with a correlation of 0.4. In this scenario, let's say both stock B and C are having a shock day, but D isn't. weighted sum = (1 × 0.8) + (1 × 0.6) + (0 × 0.4) = 1.4 total weight = 0.8 + 0.6 + 0.4 = 1.8 neighbor shock exposure = 1.4 ÷ 1.8 = 0.78 4. Finally, using each stock's own volatility, return, and its neighbor shock exposure today, two logistic regression models are trained to predict whether the stock will have a shock day or an unusually turbulent day five days ahead. One model serves as the baseline using only the stock's own information, such as returns and volatility. The other model factors in the neighbor using its shock exposure to test whether this extra information can actually help improve predictions. To ensure these models avoid "lookahead bias," a very common mistake in financial models, both models are trained on past data but tested on new data it has never seen before. This model will not tell you whether the stock will move; instead, it also uses precise metrics to measure whether the current day is in its top 15% most volatile days. After all training is complete, the model must be trained once across the whole market and can calculate any single stock result. This is expressed as "lift" — how many times more likely a neighbor is to shock following the chosen stock's shock, compared to its normal rate.

3 · DATASET

source, size, preprocessing

The data points that I used are the daily adjusted closing prices of 23 large-cap US equities, which are across 5 different sectors (energy, consumer staples, technology, financial, and healthcare), which were retrieved using the Yahoo Finance API. The time period ranged from January of 2018 to June of 2021, and this was deliberate as it includes the 2020 COVID-19 volatility shock. Preprocessing: daily percentage returns; 20-day rolling volatility; shock labeling at each stock's own 85th volatility percentile; rolling 20-day correlation matrix for graph edges. (edge threshold |r|>=0.35)

4 · METRICS

how you'll know it worked

The main comparison is between the baseline model and the graph-aware model, using recall, precision, and AUC. This was all measured using a test set that the model had never seen before, and this comes directly after training the model. For the investor side, "lift" is used, and this measures how much more likely a connected stock is to shock; this is given that the neighboring stock just shocked, compared to the stock's normal shock rate. To ensure every metric is correct, it's recalculated a second time with the February-June 2020 COVID window excluded entirely. This also answers another question: is the contagion effect a real pattern, or is it just the one giant event across the entire market getting picked up by the model? Preliminary Results This has already been built and has been tested on real data using JPMorgan Chase as a test, and here's what I found: Model Precision Recall AUC Baseline (own stock only) 0.615 0.009 0.847 Graph-aware (comparing with 0.704 0.474 0.870 neighbors) Not only did the recall change drastically, going from 0.9% (meaning it essentially never caught a real shock) to 47.4% (meaning it actually works), but precision also increased by around 9%. Robustness Check Now, the COVID window won't be factored in, as about 67% of JPM's shock days occurred during it, and the model may be picking up a massive market event rather than a relationship. So the lift numbers were recalculated with that window taken out entirely: Neighbor Sector Relation Lift (full data) Lift(excluding COVID) BAC Bank peer 6.44 15.99 WFC Bank peer 6.49 14.99 MS Bank peer 6.23 14.26 JNJ Unrelated sector 2.48 0.23 This is a good sign, as this shows that the bank peers such as BAC, WFC, and MS got a stronger COVID was removed. This means that the relationship holds up even without one dominant crash driving everything. However, for JNJ, the lift dropped to near zero, which indicates it has no economic connection to JPM. This shows that, by excluding COVID, the model picks up a real, repeatable pattern rather than a market-wide event.

5 · COMPUTE

training time, memory, dependencies

No GPU is required here. The entire plan: downloading data, building features, training, and evaluating. It runs in under 2 minutes on a free Google Colab CPU instance. All the tools in use are completely open source: numpy, pandas, yfinance, networkx, scikit-learn, joblib, and matplotlib.

6 · DELIVERABLES

model, paper, demo

and ends with an evaluation; it is all runnable from start to finish in Google Colab. ● In the end, it can generate a full contagion report in plain language for any stock in the basket. ● This written report covers the baseline vs graph aware comparison as well as the COVID robustness check. ● The goal would be to scale this multihop contagion instead of just direct neighbors.

· A trained, graph-aware logistic model with interpretable coefficients. ● A full Python pipeline that starts from raw data

7 · RISK

what breaks first, and plan B

Twenty-three large-caps are not a market. Treat the JPM result as a proof of concept, then rerun on a broader basket before claiming generality.

The lift may be the 2020 crash rather than a repeating relationship. The COVID-exclusion check is the current evidence against that: same-sector peers got stronger, JNJ fell to near zero.

Exact lift numbers on this basket are noisy. Use them as a direction, not a trading signal.