

# Stock Trading Day Clustering Project

## Overview

### Background & Concepts

#### What is Clustering?

#### Core Concepts

1. K-Means Clustering
2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
3. Gaussian Mixture Models (GMM)

#### Data Processing (Before Clustering)





##### Feature Engineering

##### Scaling the Data

##### PCA (Principal Component Analysis)

##### What is TA?

### Step-by-Step Analysis Process

1.  Data Collection
2.  Feature Engineering
3.  Scaling the Data
4.  PCA for Dimensionality Reduction
5.  Clustering the Data

### Raw Data

### Visual Outputs

1.  Cluster Distribution
2.  PCA Projection
3.  Feature Pairplot
4.  Heatmap - Normalized
5.  Average Feature Value Per Cluster

### Using the Streamlit App

#### Features

#### How to Run Locally

### Project Folder Structure

### Insights & Learnings

## Overview

This project uses unsupervised machine learning to explore how stock trading days behave. By looking at key features like returns, volatility, and trading volume, we group similar days together using clustering algorithms like K-Means, DBSCAN, and GMM. These clusters help identify patterns, such as “volatile high-volume gain days” or “quiet loss days,” which can help investors and analysts make more informed decisions.

**LINK TO WEB APP:** <https://trade-day-clustering.streamlit.app/>

## Background & Concepts

### What is Clustering?

Clustering is an *unsupervised machine learning* technique used to group similar data points together based on their features, without prior labels. The goal is to partition data such that points in the same group (cluster) are more similar to each other than to those in other groups.

Clustering is like sorting socks after laundry—but you don't know in advance how many types of socks you have. Instead, you look at characteristics (like color, size, material) and let a program group them based on similarity. In this project, each trading day is a "sock", and we group them based on things like return, volatility, and volume.

---

## Core Concepts [↗](#)

### 1. K-Means Clustering [↗](#)

K-Means is a centroid-based clustering algorithm that partitions data into  $K$  clusters by minimizing the sum of squared distances between data points and their assigned cluster center (centroid). It iteratively updates cluster centers until convergence.

Imagine you have a bunch of dots (data points) on a piece of paper. K-Means picks  $K$  centers and pulls each dot toward the closest center. It keeps adjusting the centers until everything is nicely grouped.

---

### 2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [↗](#)

DBSCAN groups together points that are closely packed (points with many nearby neighbors) and marks as outliers the points that lie alone in low-density regions. It uses two parameters: `eps` (neighborhood radius) and `min_samples` (minimum points to form a dense region).

Think of people at a party. DBSCAN groups people standing close together into groups. If someone is standing alone or far from others, they're called "noise" or an outlier.

---

### 3. Gaussian Mixture Models (GMM) [↗](#)

GMM assumes the data is generated from a mixture of several Gaussian distributions. It uses *Expectation-Maximization* to assign each point a probability of belonging to each cluster instead of a hard label.

Instead of assigning each data point strictly to a group, GMM gives each day a probability of belonging to each group.

---

## Data Processing (Before Clustering) [↗](#)

We generate numerical *features* to describe each trading day—these are the inputs for clustering. Think of them as coordinates in a multi-dimensional space representing market behavior.

---

## Feature Engineering [↗](#)

Feature	Formula	What it tells us
<code>daily_return</code>	$(\text{Close} - \text{Open}) / \text{Open}$	Was the price up or down? Measures daily % change.
<code>price_range</code>	$(\text{High} - \text{Low}) / \text{Open}$	How much price moved during the day. Reflects range.
<code>volatility</code>	$\text{Std}([\text{Open}, \text{High}, \text{Low}, \text{Close}])$	A measure of price "bounciness" during the day.
<code>volume_change</code>	$\text{Volume}[t] / \text{Volume}[t-1] - 1$	Volume change vs the previous day.

<code>volume_vs_avg</code>	Volume / 5-day rolling average	Whether today's volume is high or low compared to recent days.
<code>rsi</code>	RSI (14-day)	Momentum indicator. >70 = overbought, <30 = oversold.
<code>macd</code>	MACD (12-26 EMA)	Shows bullish/bearish momentum.
<code>macd_signal</code>	MACD signal line (9 EMA of MACD)	Used for identifying buy/sell triggers.

---

### **Scaling the Data** [↗](#)

Scaling standardizes feature values to have a mean of 0 and a standard deviation of 1 using `StandardScaler`. This ensures that features with larger ranges don't dominate the clustering algorithm.

Like resizing all pictures before printing so they fit equally—no one picture takes over the wall space.

---

### **PCA (Principal Component Analysis)** [↗](#)

PCA is a dimensionality reduction technique that transforms the original variables into a new set of linearly uncorrelated variables (principal components), preserving as much variance as possible.

It's like showing a 3D object from the best angle in 2D—so we capture the key patterns without the complexity.

---

### **What is TA?** [↗](#)

TA (Technical Analysis) is a Python library used to compute indicators derived from historical price and volume data, such as RSI, MACD, and moving averages.

Think of it as adding smart labels to trading days, like “momentum up” or “volume spike,” helping us better group days by behavior.

Indicator	What it Means	Why it's Useful
RSI	Measures momentum—how strongly price is moving up/down	>70 overbought, <30 oversold
MACD	Difference between short and long-term moving averages	Detect trend reversals and momentum shifts
MACD Signal	Smoothed MACD used as a signal line	Triggers buy/sell decisions

### **Step-by-Step Analysis Process** [↗](#)

This analysis process was done for NMR stock ticker using K-Means clustering with 5 clusters.

## 1. 📥 Data Collection [🔗](#)

We used Yahoo Finance to download daily stock data for NMR (Nomura Holdings). This included Open, Close, High, Low, and Volume for each trading day from 2023-01-01 to 2024-12-30.

## 2. 📊 Feature Engineering [🔗](#)

We calculated eight key features to describe trading behavior.

## 3. ⚖️ Scaling the Data [🔗](#)

We standardized all features using StandardScaler so they have the same scale. This prevents large-value features like Volume from overpowering smaller ones like Return.

## 4. 📈 PCA for Dimensionality Reduction [🔗](#)

PCA reduces complex, multi-dimensional data down to 2D for easy visualization. This helps in plotting clusters on a simple chart.

## 5. 🔗 Clustering the Data [🔗](#)

Based on the selected method (KMeans, DBSCAN, or GMM), trading days are grouped into clusters:

- KMeans: Divides days into a set number of clusters (e.g., 3 or 5).
- DBSCAN: Finds dense groups and labels isolated points as noise.
- GMM: Assigns probabilities to each day for each cluster.

Each cluster gets a human-readable label based on average volume, return, volatility, etc., e.g., 'Volume Trend Up, High Volume, High Volatility, Moderate Gain'.

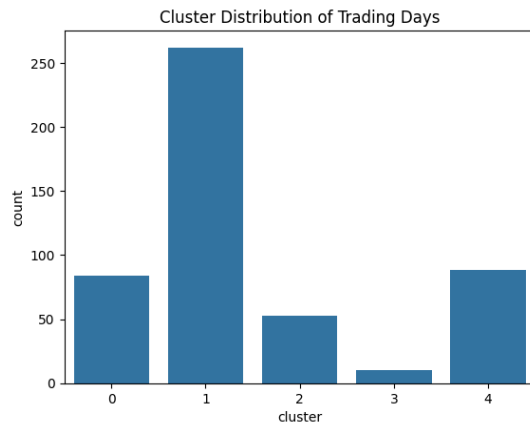
## 📊 Raw Data [🔗](#)

	Date	Close	High	Low	Open	Volume	daily_return	price_range	volatility	volume_change	volume_5day_avg	volume_vs_avg
0	2023-01-09	3.79	3.84	3.78	3.83	523800	-0.010444	0.015666	0.029439	-0.726704	1158340.0	0.452199
1	2023-01-10	3.75	3.76	3.72	3.75	979700	0.000000	0.010667	0.017320	0.870370	1165300.0	0.840728
2	2023-01-11	3.79	3.80	3.77	3.79	758800	0.000000	0.007916	0.012583	-0.225477	1038700.0	0.730529
3	2023-01-12	3.87	3.89	3.81	3.82	765800	0.013089	0.020942	0.038622	0.009225	988940.0	0.774364
4	2023-01-13	3.97	3.97	3.89	3.89	725100	0.020566	0.020566	0.046188	-0.053147	750640.0	0.965976

## 📊 Visual Outputs [🔗](#)

### 1. 📈 Cluster Distribution [🔗](#)

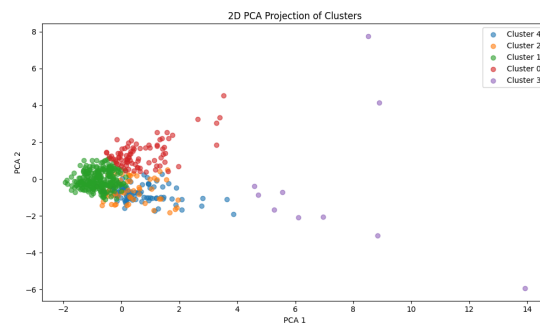
Shows how many trading days belong to each cluster.



This bar chart shows how many trading days fall into each cluster. We observe that **Cluster 1** dominates, suggesting a “normal” or typical market behavior that occurs most often. Smaller clusters like **Cluster 3** may represent outliers or unusual market conditions worth deeper investigation.

## 2. 🌐 PCA Projection 🔗

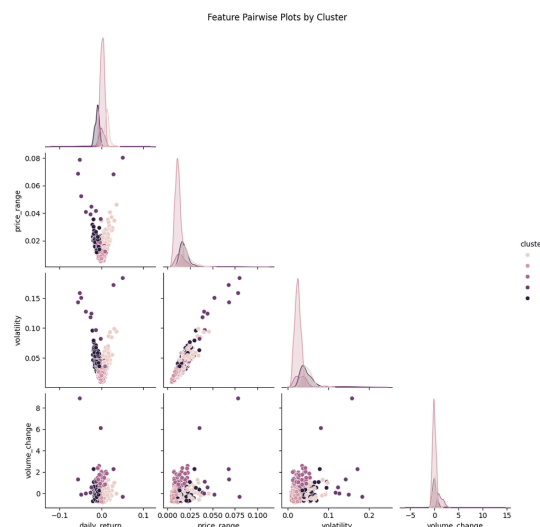
Plots trading days in a 2D space to visualize cluster separation.



The PCA plot reduces our high-dimensional features into two components, allowing us to visually inspect how trading days are grouped. Distinct cluster separation (e.g., Cluster 3 being far apart) implies that those days have significantly different behaviors —potentially tied to earnings releases, economic events, or extreme volume surges.

## 3. 🔄 Feature Pairplot 🔗

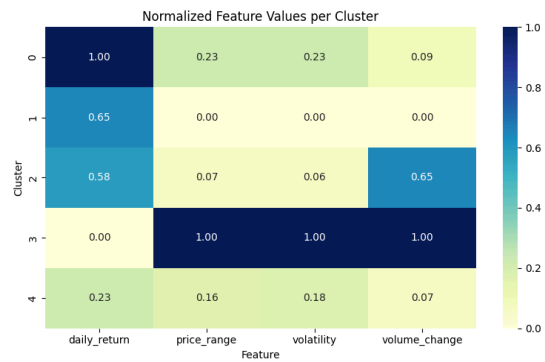
Shows how features relate to each other within clusters.



These pairplots show how features interact within and across clusters. For instance, higher **volatility** often aligns with wider **price ranges** and sometimes larger **volume changes**. Clusters with tighter, less varied distributions (e.g., Cluster 1) reflect consistent, less volatile market days, while scattered patterns (e.g., Cluster 4) reveal erratic behaviors.

#### 4. 🔥 Heatmap - Normalized [🔗](#)

Normalized to 0–1 scale to highlight differences across features and clusters.



The normalized heatmap brings all features to a common scale (0–1) for easy comparison. Here, **Cluster 0** shows the highest normalized **daily return**, while **Cluster 3** dominates in volatility and volume change. This view makes it easier to identify each cluster’s defining characteristics regardless of raw magnitude.

#### 5. 📈 Average Feature Value Per Cluster [🔗](#)

		price_range	volatility	volume_change
	daily_return			
cluster				
0	0.013	0.021	0.049	0.089
1	0.002	0.011	0.024	-0.091
2	0.000	0.014	0.031	1.250
3	-0.018	0.055	0.135	1.980
4	-0.011	0.018	0.044	0.046

### 💻 Using the Streamlit App [🔗](#)

#### 🚀 Features [🔗](#)

- Select stock ticker, date range, and clustering method.
- Set parameters for KMeans, DBSCAN, or GMM.
- View and download charts.
- Toggle raw data visibility.
- Export clustered data as CSV.

## How to Run Locally

1. Install dependencies:

```
1 pip install -r requirements.txt
```

2. Launch the app:

```
1 streamlit run app.py
```

The app will open at <http://localhost:8501>

## Project Folder Structure

```
1 .
2 |─ app/app.py           # Main Streamlit application
3 |─ requirements.txt      # Dependencies
4 |─ notebooks/clustening_model.ipynb # Jupyter notebook to rerun analysis
5 |─ clustening_model_results.pdf    # Jupyter Notebook snapshots
```

## Insights & Learnings

- Unsupervised learning can uncover hidden market patterns.
- Volume and return are especially helpful in clustering trading behaviors.
- Visual tools like PCA and heatmaps aid in explaining results to non-technical stakeholders.
- Streamlit enables an interactive, low-code way to share results.