Stock Market Data Analysis and Prediction

# *Work Module for Stock Market Data Analysis and Prediction*

**Start Date: (9th November 2024)**

## Single Stock Analysis

1. Extract historical stock data from the source and store it in an Excel file.
2. Clean and preprocess the data.
3. Perform feature engineering.
4. Conduct predictive analysis to assess initial model performance.

## Adding Version controlling system (Git) & Automation of Daily Tasks

Automate the data extraction and prediction processes to run daily, ensuring updated predictions based on the latest data.

## Data Management

Transition data storage from Excel to a cloud database for better scalability and accessibility.

## Multi-Stock Testing

Expand analysis to include 2-3 additional stocks and evaluate the setup's performance.

## Full-Scale Stock Analysis

Extend the setup to include as many stocks as possible within the database and perform predictive analysis on all included stocks.

Goal:

* Inputs:

1. Stock you already have + stock listed in the model.
2. Capital

* Output:

It will tell you the stock to buy with that capital

The model takes the input of already have stocks (if any) also it will have a list of fundamentally good stocks.

Model workflow:

1. Calculate the next-day price of a single stock.
2. Group the stock as buy-stock, sell-stock, hold stock.
3. Check the capital and accordingly suggest which stock to buy.

Points:

1. Once the stock is shown as buy, it should show sell only if it has gained 5% or more.

# Downloading Data

Here’s a rating of the platforms based on **data accuracy** and **ease of automation for data extraction**, with consideration for reliability and ease of integration into Python-based workflows:

| **Platform** | **Data Accuracy** | **Ease of Automation** | **Comments** |
| --- | --- | --- | --- |
| **1. Yahoo Finance (yfinance)** | ⭐⭐⭐⭐ | ⭐⭐⭐⭐⭐ | Highly accurate, broad data for Indian stocks, easy to automate with yfinance library. Limited by occasional data delays. |
| **2. NSE India** | ⭐⭐⭐⭐⭐ | ⭐⭐ | Extremely accurate (official source), but lacks an official API. Requires web scraping or manual downloads, making automation complex. |
| **3. Alpha Vantage** | ⭐⭐⭐⭐ | ⭐⭐⭐⭐ | Accurate and reliable, especially for historical data. API available, but limited requests in the free version. |
| **4. Quandl** | ⭐⭐⭐⭐ | ⭐⭐⭐⭐ | Good data accuracy with free datasets. Limited stock selection but easy to automate with the Quandl API. |
| **5. Investing.com (InvestPy)** | ⭐⭐⭐⭐ | ⭐⭐⭐⭐ | Accurate and wide data range, including indices and commodities. InvestPy library simplifies automation, but some setup is needed for installation. |
| **6. Twelve Data** | ⭐⭐⭐ | ⭐⭐⭐⭐ | Generally accurate with a user-friendly API for automation. Limited real-time data and capped by request quotas. |
| **7. BSE India** | ⭐⭐⭐⭐⭐ | ⭐ | Highly accurate (official source), but automation is challenging due to manual download requirements or complex web scraping. |

**Detailed Breakdown:**

**1. Yahoo Finance (yfinance)**

* **Data Accuracy**: Very reliable for most users, with high historical accuracy for daily data.
* **Automation**: High ease of automation due to the yfinance library, which simplifies data access for daily or historical data. Data can sometimes have slight delays, but it's generally accurate for non-intraday use.

**2. NSE India**

* **Data Accuracy**: As the official source, it provides the most accurate and complete dataset for Indian equities.
* **Automation**: There is no public API, so automation often requires manual scraping or downloading, which can violate NSE’s terms. For accurate and official data, this is a solid source, but it requires more setup for automated extraction.

**3. Alpha Vantage**

* **Data Accuracy**: High accuracy, especially for historical data. Real-time accuracy may lag slightly.
* **Automation**: Provides an easy-to-use API, so automation is straightforward. However, free accounts have limited requests (5 per minute, 500 per day), which may constrain high-frequency data needs.

**4. Quandl**

* **Data Accuracy**: Reliable, though some free datasets may have limited data points.
* **Automation**: The Quandl API is well-documented and allows easy data extraction. However, some datasets require premium access, and the selection of free Indian stocks is somewhat limited.

**5. Investing.com (InvestPy)**

* **Data Accuracy**: Investing.com data is quite accurate for historical data, covering a wide range of stocks and sectors.
* **Automation**: The InvestPy library provides a convenient method for fetching data. Installation and initial configuration can take a bit of time, but it’s straightforward for regular use once set up.

**6. Twelve Data**

* **Data Accuracy**: Generally good for Indian stocks, though occasionally, real-time data may not be as accurate as official sources.
* **Automation**: Provides a simple API with straightforward integration, though free usage limits might restrict data-heavy tasks. Good for lower-frequency or small-volume data extraction.

**7. BSE India**

* **Data Accuracy**: BSE, as the official exchange, offers highly accurate data.
* **Automation**: BSE lacks an API, making automation challenging. Data extraction is typically manual, so it’s best for users looking for very specific data with limited frequency.

**Final Recommendations**

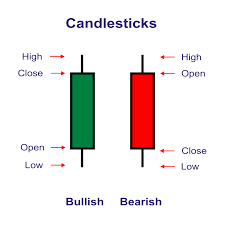
For high **automation** with decent **accuracy**, **Yahoo Finance**, **Alpha Vantage**, and **Investing.com** (using InvestPy) are the most practical choices. If **data accuracy** is your highest priority and automation can be managed manually, **NSE India** and **BSE India** are ideal.

### Conclusion: We will use the data from **Yahoo Finance** for initial setup and later on check for any upgrade.

The data we got from Yahoo Finance looks like below,

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Date | Adj Close | Close | High | Low | Open | Volume |
| 01-01-2024 | 38.5 | 38.5 | 38.6 | 38 | 38.5 | 35078873 |
| 02-01-2024 | 38.25 | 38.25 | 38.8 | 37.1 | 38.7 | 30864924 |
| 03-01-2024 | 37.8 | 37.8 | 38.25 | 37.7 | 38.25 | 21788140 |

Columns (Open, Close, High, Close) are basic features for stock and can be demonstrated as below.



* If a stock goes up for a day, it is represented by Bullish/Upward growth.
* If a stock goes down for a day, it is represented by Bearish/Downward growth.

# Adding Technical indicators to data:

To improve predictive analysis, we will add the technical indicators below,

* 1. **Moving Averages (SMA & EMA)**

**Simple Moving Average (SMA)**

The **SMA** is the average of a stock's closing prices over a specified number of periods (e.g., 10-day or 50-day).

**Exponential Moving Average (EMA)**

The **EMA** gives more weight to recent prices and is more responsive than SMA.

* 1. **Relative Strength Index (RSI)**

The **RSI** measures the speed and change of price movements and indicates overbought or oversold conditions (values >70 or <30).

* 1. **Moving Average Convergence Divergence (MACD)**

The **MACD** is the difference between the 12-day EMA and 26-day EMA, along with a signal line (9-day EMA of MACD).

* 1. **Bollinger Bands**

**Bollinger Bands** consist of three lines: the middle line (SMA), the upper band (SMA + 2 \* standard deviation), and the lower band (SMA - 2 \* standard deviation).

We calculated the indicators and appended them to the initial data as below,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Date | Adj Close | Close | High | Low | Open | Volume | SMA\_10 | SMA\_50 | EMA\_10 | EMA\_50 | RSI | EMA\_12 | EMA\_26 | MACD | MACD\_Signal | SMA\_20 | Bollinger\_Upper | Bollinger\_Lower |
| 13-03-2024 | 37.35 | 37.35 | 39.3 | 37.35 | 38.5 | 38160910 | 41.155 | 43.512 | 40.659 | 42.694 | 28.488 | 41.101 | 42.637 | -1.536 | -0.742 | 43.31 | 49.196 | 37.424 |
| 14-03-2024 | 39.05 | 39.05 | 39.2 | 35.5 | 35.5 | 67013191 | 40.74 | 43.523 | 40.367 | 42.551 | 34.748 | 40.786 | 42.371 | -1.585 | -0.91 | 42.96 | 48.991 | 36.929 |
| 15-03-2024 | 39.4 | 39.4 | 40 | 37.5 | 39 | 1.45E+08 | 40.155 | 43.546 | 40.191 | 42.427 | 35.602 | 40.572 | 42.151 | -1.579 | -1.044 | 42.578 | 48.485 | 36.671 |

## Additional technical indicators can be added in the later version.

* 1. **Market Trends**
* **Indices Correlation**: Track major indices (e.g., NIFTY, S&P 500) and correlate them with your stock data.
* **Sectoral Indicators**: Include performance metrics of the stock's sector (e.g., FMCG index for Nestle).
* **Volatility Index (VIX)**: Use VIX to gauge market fear and sentiment.
  1. **Advance Technical Indicators**
* **ATR (Average True Range)**: Measures market volatility.
* **Stochastic Oscillator**: Compares a stock’s closing price to its price range over time.
* **VWAP (Volume Weighted Average Price)**: Reflects price trends considering trading volume.

# Adding Sentimental indicators to data:

To enhance prediction, we need to add sentiment data, especially for short-term stock price movements, as sentiment from news or social media often influences price volatility.

Currently, we found two ways to collect stock sentiment data from the source and analyze it as below:

1. Twitter API
2. News API

**Both Tweepy (Twitter data) and News API** can provide a more comprehensive view of market sentiment. Here’s a breakdown of the strengths of each source and why combining them might yield the best results:

**Twitter Sentiment (Tweepy)**

* **Real-Time Insights**: Twitter captures immediate reactions, making it ideal for high-frequency trading and short-term sentiment analysis.
* **Broader User Sentiment**: Twitter reflects opinions from individual investors, which can often influence volatile stock movements, especially for stocks with a strong retail trading interest.
* **Frequent Updates**: You can monitor sentiment minute-by-minute, which is helpful for tracking quick shifts in public sentiment during major events or announcements.

**Limitations**:

* Twitter data can be noisy and requires extensive cleaning (e.g., filtering bots, spam, irrelevant tweets).
* It may not capture the in-depth analysis typically seen in news articles.

**News Sentiment (News API)**

* **Detailed Analysis**: News articles provide in-depth information and analysis on company performance, industry trends, and economic factors, making it a solid sentiment source for medium-to-long-term predictions.
* **Credible Sources**: News API pulls data from trusted news sources, reducing the need for extensive filtering of spam or low-quality content.
* **Broader Context**: News often provides contextual information that is harder to get from social media, such as market analysis, expert opinions, and regulatory updates.

**Limitations**:

* News articles may lag behind events, particularly in fast-moving markets.
* Fewer updates compared to Twitter, especially outside of major news events.

**Using Both Twitter and News Sentiment Together**

* **Comprehensive Coverage**: Twitter provides real-time reaction, while news offers in-depth context. This combination can give a well-rounded view of both immediate public opinion and expert analysis.
* **Improved Model Robustness**: Combining both sources can help smooth out sentiment fluctuations. Twitter sentiment is often more volatile, while news sentiment provides a steadier signal, making predictions more reliable.
* **Layered Sentiment Features**: You can create features like short-term sentiment (Twitter) and medium-term sentiment (news) to help the model distinguish between short-lived hype and sustained trends.

**Implementation Strategy**

1. **Real-Time Events**: Use Twitter for immediate reactions to events (e.g., earnings reports, new product releases).
2. **Broader Market Sentiment**: Use news sentiment for analyzing longer-term market or industry trends.
3. **Combine and Aggregate**: Include both Twitter and news sentiment as separate features and test their impact on your model’s accuracy.

\*\*14th November Update:\*\* We attempted to work on sentiment analysis; however, we encountered a 'Too Many Requests' issue while trying to access Tweepy. Additionally, we had difficulty finding relevant news articles using the News API. Both files are saved as ‘SentimentAnalysis\_Tweepy.ipynb’ and ‘NewsApi\_sentiments.ipynb’ in the designated folder. We will consider this as a future enhancement and plan to revisit it in later versions.

# Adding Fundamental Indicators to data:

We will also include some fundamental indicators like PE ratio and PB ratio to make our prediction fundamentally strong.

**P/E Ratio (Price-to-Earnings Ratio)**: Measures how much investors are willing to pay per dollar of earnings. A high P/E might suggest high growth expectations, while a low P/E could indicate undervaluation or lower growth prospects.

**P/B Ratio (Price-to-Book Ratio)**: Compares the market price of a stock to its book value. A low P/B could suggest the stock is undervalued.

There are some other indicators, such as the debt-to-equity ratio, ROE, current ratio, EPS, revenue and net income, and quarterly earnings report. We can add this at a later stage.

We calculated the indicators and appended them as below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Date | Adj Close | Close | High | Low | Open | Volume | PE\_ratio | PB\_ratio | SMA\_10 | SMA\_50 | EMA\_10 | EMA\_50 | RSI | EMA\_12 | EMA\_26 | MACD | MACD\_Signal | SMA\_20 | Bollinger\_Upper | Bollinger\_Lower |
| 13-03-2024 | 37.35 | 37.35 | 39.3 | 37.35 | 38.5 | 38160910 | 52.606 | 11.09 | 41.155 | 43.512 | 40.659 | 42.694 | 28.488 | 41.101 | 42.637 | -1.536 | -0.742 | 43.31 | 49.196 | 37.424 |
| 14-03-2024 | 39.05 | 39.05 | 39.2 | 35.5 | 35.5 | 67013191 | 55 | 11.594 | 40.74 | 43.523 | 40.367 | 42.551 | 34.748 | 40.786 | 42.371 | -1.585 | -0.91 | 42.96 | 48.991 | 36.929 |
| 15-03-2024 | 39.4 | 39.4 | 40 | 37.5 | 39 | 144957453 | 55.493 | 11.698 | 40.155 | 43.546 | 40.191 | 42.427 | 35.602 | 40.572 | 42.151 | -1.579 | -1.044 | 42.578 | 48.485 | 36.671 |

### \*\* Improvements for data collection model\*\*

1. **Error Handling**:
   * Wrap API calls and file operations in try-except blocks.
   * Handle cases like invalid ticker symbols, API limits, or empty datasets.
2. **Logging**:
   * Use the logging library to replace print statements.
   * Enable detailed logs with timestamps for easy debugging.
3. **Enhanced Flexibility**:
   * Automate data downloads for multiple stocks using a list of tickers.
   * Add optional arguments for additional intervals or regions.
4. **Data Validation**:
   * Check for missing or invalid values in the downloaded data.
   * Add fallback mechanisms (e.g., retry on failures).
5. **Optimization**:
   * Modularize the code into smaller functions for readability.
   * Optimize saving logic to avoid overwriting files unintentionally.
6. **Documentation**:
   * Include clear docstrings and inline comments.

In the later versions of the model, we will focus on the areas of improvement mentioned above.

# Analysis Data:

**Key Insights from the Data**

1. **Fundamental Indicators:**
   * **PE\_ratio** and **PB\_ratio** are well-calculated and included.
   * These provide strong financial insights for your predictions.
2. **Technical Indicators:**
   * **Moving Averages (SMA\_10, EMA\_10, EMA\_50):**
     + Capture short- and long-term trends.
   * **Relative Strength Index (RSI):**
     + Indicates stock momentum and overbought/oversold conditions.
   * **MACD & MACD Signal:**
     + Highlight trend strength and reversals.
   * **Bollinger Bands (Upper/Lower):**
     + Define volatility and potential breakout points.
3. **Integration:**
   * The combination of fundamental (PE/PB) and technical indicators (SMA, RSI, Bollinger Bands) is an excellent setup for predictive modeling.

**Strengths of Current Dataset**

* **Comprehensive Coverage:** Both fundamental and technical factors are included, offering a robust basis for predictive analysis.
* **Data Preprocessing:** Indicators like SMA, EMA, and Bollinger Bands are pre-calculated, saving computational time for modeling.
* **Balanced Approach:** Fundamental indicators (e.g., PE, PB) complement technical ones, capturing both intrinsic value and market behavior.

**Identifying the best combination of data for swing trading prediction:**

To identify the best single or combination of columns for swing stock price analysis, it's essential to focus on indicators that capture price trends, momentum, volatility, and support/resistance levels. Below is an evaluation of the relevance of each column in your dataset, along with recommended sets for swing trading.

**Key Indicators for Swing Trading**

1. Price Trends

* SMA\_10, SMA\_50, EMA\_10, EMA\_50:
  + Used to identify short-term and medium-term trends.
  + Crossovers (e.g., SMA\_10 crossing above SMA\_50) are strong buy/sell signals.
* MACD, MACD\_Signal:
  + Indicates trend direction and strength.
  + A bullish crossover (MACD > MACD\_Signal) is a buy signal, while a bearish crossover (MACD < MACD\_Signal) is a sell signal.

2. Momentum

* RSI:
  + Identifies overbought (>70) and oversold (<30) conditions, signaling potential reversals.

3. Volatility

* Bollinger\_Upper, Bollinger\_Lower, SMA\_20:
  + Bollinger Bands help identify overextension in price movements. Price touching the upper band signals overbought conditions, while the lower band signals oversold conditions.

4. Support/Resistance

* High, Low, Open, Close:
  + Helps establish support/resistance levels to refine entry/exit points.

5. Volume

* Volume:
  + Confirms the strength of price moves. A price breakout with high volume is more reliable than one with low volume.

6. Valuation Ratios

* PE\_ratio, PB\_ratio:
  + Useful for fundamental analysis but less significant for short-term swing trading.

Recommended Column Sets

Set 1: Trend and Momentum

* SMA\_10, SMA\_50, RSI
  + Combines trend-following with momentum to identify entry/exit points.
  + Example:
    - Buy: SMA\_10 crosses above SMA\_50, and RSI is below 50.
    - Sell: SMA\_10 crosses below SMA\_50, and RSI is above 50.

Set 2: Trend and MACD

* EMA\_10, EMA\_50, MACD, MACD\_Signal
  + MACD captures momentum shifts, while EMAs confirm trend strength.
  + Example:
    - Buy: MACD > MACD\_Signal and EMA\_10 > EMA\_50.
    - Sell: MACD < MACD\_Signal and EMA\_10 < EMA\_50.

Set 3: Momentum and Volatility

* RSI, Bollinger\_Upper, Bollinger\_Lower
  + Ideal for identifying overbought/oversold conditions.
  + Example:
    - Buy: RSI < 30 and price near Bollinger\_Lower.
    - Sell: RSI > 70 and price near Bollinger\_Upper.

Set 4: Comprehensive Swing Strategy

* SMA\_20, Bollinger\_Upper, Bollinger\_Lower, RSI, Volume
  + Combines trend, volatility, and confirmation with volume.
  + Example:
    - Buy: Price crosses above SMA\_20 from below with RSI < 40 and increasing volume.
    - Sell: Price crosses below SMA\_20 from above with RSI > 60 and decreasing volume.

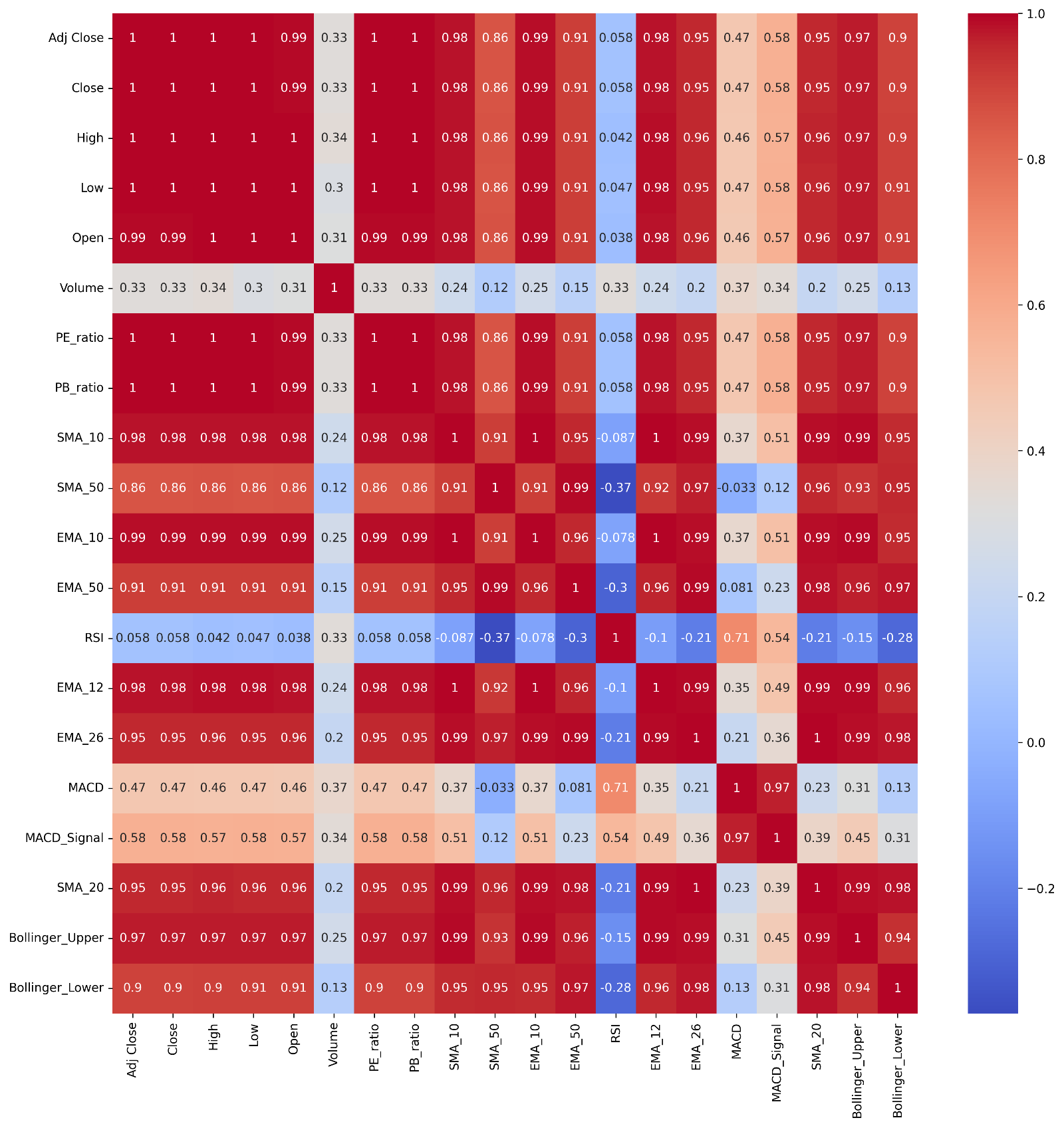
Set 5: Fundamental + Technical

* PE\_ratio, PB\_ratio, RSI, SMA\_10
  + For swing trades with a fundamental filter.
  + Example:
    - Buy: PE < Industry Average, RSI < 30, and price crosses above SMA\_10.
    - Sell: PE > Industry Average, RSI > 70, and price crosses below SMA\_10.

Best Single Indicators

* RSI: Simple and effective for momentum-based reversals.
* MACD: Useful for momentum shifts in trends.
* SMA\_10/SMA\_50: Essential for trend following.

**As these indicators are correlated with each other,** **multicollinearity and redundancy will occur.**

****

To avoid this problem, we have included all indicators to calculate a single upward/downward probability column.

The “Upward\_Downward\_Probability” column will calculate the probability considering all the features listed above considering their weights.

We have used **Log Odds Transformation** for calculation of the probability.

Here’s how the calculation is done,

|  |  |  |
| --- | --- | --- |
| **Indicator** | **Up Score** | **Down Score** |
| SMA\_10 > SMA\_50 | Increase | 0 |
| SMA\_10 < SMA\_50 | 0 | Increase |
| EMA\_10 > EMA\_50 | Increase | 0 |
| EMA\_10 < EMA\_50 | 0 | Increase |
| MACD > MACD\_Signal | Increase | 0 |
| MACD < MACD\_Signal | 0 | Increase |
| RSI > 70 | 0 | Increase |
| RSI < 30 | Increase | 0 |
| 30 < RSI < 70 | 0 | 0 |
| Close > Bollinger\_Upper | 0 | Increase |
| Close < Bollinger\_Upper | Increase | 0 |
| Close > SMA\_20 | 0 | Increase |
| Close < SMA\_20 | Increase | 0 |
| Volume > Avg\_Volume | Increase | Increase |
| PE\_ratio > 20 or PB\_ratio > 3 | 0 | Increase |
| PE\_ratio < 10 or PB\_ratio < 1 | Increase | 0 |

Also, adding weights to these indicators,

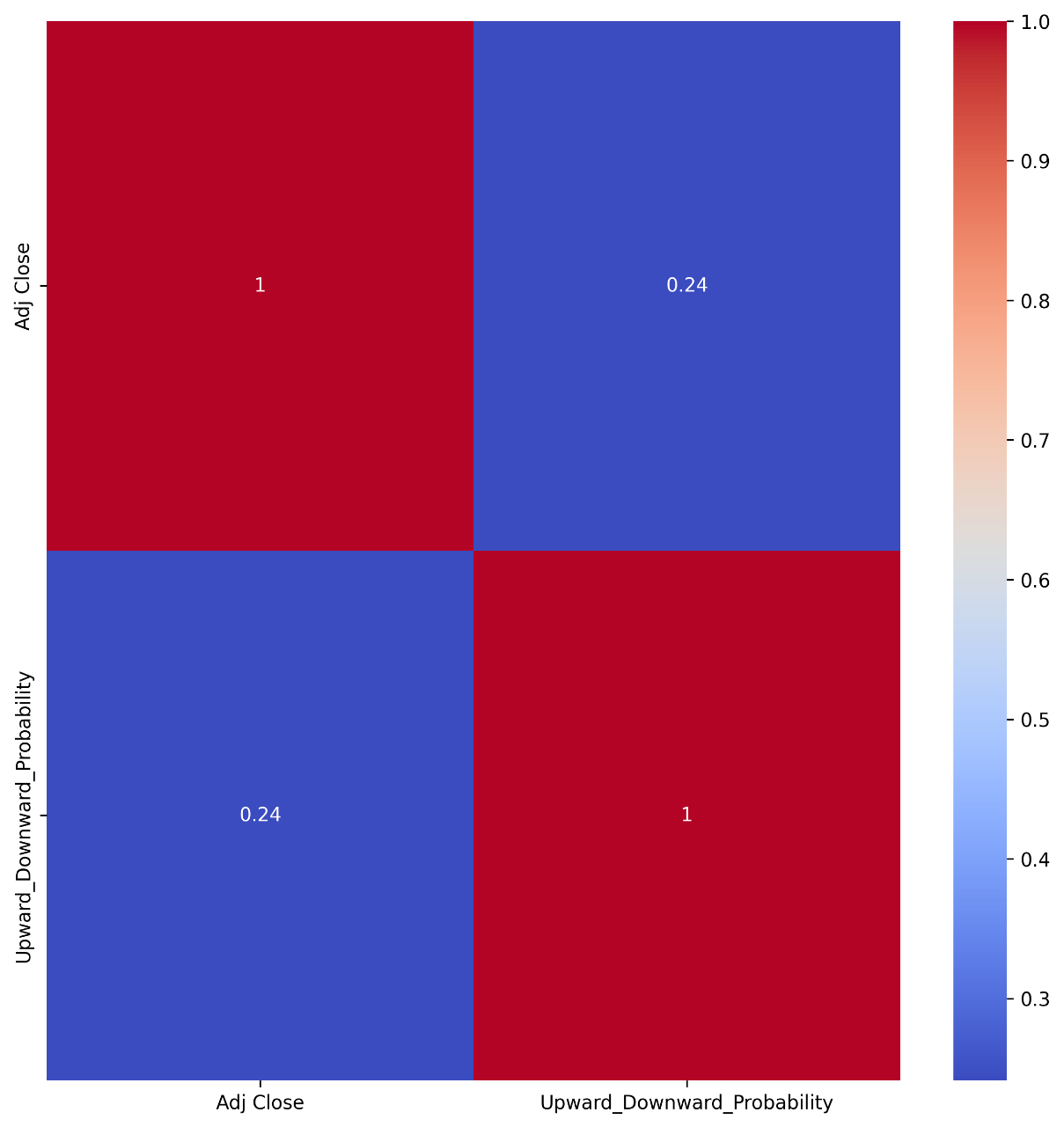
|  |  |  |
| --- | --- | --- |
| **Indicator** | **Weight** | **Rationale** |
| SMA (10 vs. 50) | 20% | Significant trend indicator for short- to medium-term price movements. |
| EMA (10 vs. 50) | 20% | More sensitive trend indicator, useful for recent price action. |
| MACD vs. Signal | 15% | Key momentum indicator for determining bullish or bearish trends. |
| RSI | 10% | Identifies overbought/oversold conditions, important but secondary to trend. |
| Bollinger Bands | 10% | Highlights extreme price deviations and reversal opportunities. |
| SMA\_20 vs. Close | 10% | A commonly used support/resistance level. |
| Volume | 5% | Confirms trend strength but may not always signal direction. |
| PE and PB Ratios | 10% | Provides a valuation perspective to balance technical signals. |

We have hardcoded the weights and limits, but in a later version, we will improve the code logic for Upward\_Downward\_Probability, using ML techniques to make the weights/limits more dynamic.

One approach to combine both methods could be:

* 1. Start with hardcoded values: This can give you a basic starting point and initial performance.
  2. Use ML for feature optimization: After implementing the hardcoded logic, you can use ML to dynamically adjust the weights of these indicators based on historical data. This would ensure your model can still take advantage of traditional indicators, while optimizing them for better performance.

Through the above step, we have removed the multicollinearity improving the correlation between features.



### Implementation of Machine Learning model:

**Objective:**

1. To include a daily predictive indicator for Buy, Sell, and Hold options.
2. To predict the second day closing price.

**Workflow/ Approach:**

* **To include a daily predictive indicator for Buy, Sell, and Hold options.**

As in the stock dataset, we don’t have a predefined target output (Buy, Sell, Hold), so we will have to use an **unsupervised ML technique** here.

Step 1: **Clustering and Anomaly Detection**:

* **Clustering** is used to group similar stock market behaviors based on historical data, identifying different types of market conditions (e.g., bullish, bearish, neutral).
* **Anomaly Detection** helps identify unusual or rare market events that may indicate buy or sell opportunities (e.g., sudden price spikes, drops, or unusual patterns in Up/Down probabilities).
* **Dimensionality Reduction** (e.g., PCA, t-SNE) can be used to reduce the complexity of the data and highlight the most important features or trends. This helps in better visualizing and identifying patterns for clustering and anomaly detection.

Step 2: **Reinforcement Learning**:

* In a **Reinforcement Learning (RL)** approach, an agent interacts with an environment (the stock market) and learns optimal trading strategies (buy, sell, hold) based on the rewards or penalties it receives.
  1. The environment represents the stock market and the agent’s interactions with it (buy, sell, hold).
  2. The agent observes stock data (e.g., prices, probabilities, indicators) and takes actions.
  3. After taking an action, the environment provides a reward or penalty (e.g., profit/loss).
* The **RL agent** uses the output of the clustering/anomaly detection as a feature for decision-making. For example:
  + The **cluster ID** or **anomaly score** can become part of the state representation.
  + The RL model then learns how to take actions (buy, sell, hold) based on the current market conditions as identified by clustering and anomaly detection.

**Benefits of This Approach:**

1. **Data-Driven Market Understanding**:
   * Clustering and anomaly detection provide a way to understand market trends and unusual behaviors without needing predefined labels (i.e., buy/sell/hold).
2. **Dynamic Adaptation**:
   * The RL agent learns over time and adapts its strategy based on both historical trends and real-time market behavior, leading to a more flexible, dynamic trading strategy.
3. **Unsupervised Insights**:
   * By using unsupervised techniques like clustering and anomaly detection, the system can identify patterns and outliers that traditional models might miss.

**Potential Challenges:**

* **Complexity of Integration**: Combining clustering, anomaly detection, and RL requires careful design, as each model may provide different forms of data that need to be integrated effectively.
* **Exploration vs. Exploitation**: The RL agent needs to balance exploring new actions (exploration) and sticking with profitable ones (exploitation), which can be challenging in a volatile market.

While working with reinforcement learning techniques, we noticed that this approach does not utilize input data as thoroughly as supervised machine learning does.

1. Our current inputs include: ['Temporal\_Features', 'Price\_Features', 'Upward\_Downward\_Probability', 'Cluster', 'Anomaly'].
2. The reinforcement learning model considers all features as a single set of float values, typically with five to six decimal places. The likelihood of encountering the same set of inputs is very low. As a result, the model generates a new state for each new day.
3. The model performs better when there are similar input sets that allow it to compare current outputs with previous outputs. However, this is not the case here, and it takes considerable time for the model to create a new state and apply the algorithm.
4. Additionally, we discovered that we can streamline the model's objective to focus on just one goal.
5. Regardless, we still want to determine whether stock prices will go up or down, which can be efficiently calculated using supervised machine learning techniques.
6. We can later apply the reinforcement learning model to the stock selection process based on stocks previously added to the portfolio.

* **To predict the second day closing price.**

We can use the next day's close price as a target output here, so we will use a **supervised ML** technique.

Models to use:

**1. Traditional Machine Learning Models**

Best for tabular data and when you want quick results with high interpretability.

**a) Linear Regression:**

* **Use Case**: Baseline model to identify linear relationships between features and the target.
* **Consideration**: Works well if relationships are approximately linear.

**b) Random Forest Regressor:**

* **Use Case**: Handles non-linear patterns and works well with mixed data types.
* **Consideration**: Robust to overfitting with proper hyperparameter tuning (e.g., max\_depth).

**c) Gradient Boosting Models:**

* **Models**: **XGBoost, LightGBM, CatBoost**.
* **Use Case**: Excellent for capturing complex, non-linear relationships and handling feature interactions.
* **Consideration**: Performs well with your mixed temporal and price-based features.

**d) Support Vector Regression (SVR):**

* **Use Case**: Suitable for small datasets with non-linear decision boundaries.
* **Consideration**: Requires feature scaling and can be computationally expensive for large datasets.

**2. Time-Series-Specific Models**

If you want to explicitly model the sequential nature of the data.

**a) ARIMA/Seasonal ARIMA (SARIMA):**

* **Use Case**: Great for capturing trends and seasonality in time-series data.
* **Consideration**: Requires stationary data and works only with price features (doesn’t use other predictors well).

**b) Prophet:**

* **Use Case**: Ideal for handling seasonality, holidays, and trends with minimal parameter tuning.
* **Consideration**: Works best with temporal features but doesn’t handle lagged features directly.

**3. Deep Learning Models**

For complex relationships and when you have a large dataset.

**a) Long Short-Term Memory (LSTM):**

* **Use Case**: Designed for sequential data, capturing long-term dependencies between lagged prices and temporal features.
* **Consideration**: Requires a significant amount of data and proper tuning.

**b) Gated Recurrent Unit (GRU):**

* **Use Case**: Similar to LSTM but computationally lighter.
* **Consideration**: Use it if your data isn’t too complex but still requires sequential modeling.

**c) Hybrid Models (CNN-LSTM):**

* **Use Case**: Combines CNN for feature extraction from time-series data and LSTM for sequence learning.
* **Consideration**: Useful for highly complex stock data.

**4. Hybrid and Ensemble Models**

Combine multiple techniques to leverage the strengths of each.

**a) Stacking Models:**

* Combine predictions from models like Random Forest, XGBoost, and SVR for better performance.

**b) Neural Networks with Feature Engineering:**

* Use dense (feed-forward) neural networks after transforming features like close\_lag and Upward\_Downward\_Probability into engineered inputs.

**Recommended Approach:**

1. Start with **Random Forest** or **XGBoost** for quick and interpretable results.
2. Move to **LSTM** or **GRU** if you want to leverage time dependencies.
3. For long-term patterns, experiment with **ARIMA** or **Prophet**.
4. Test ensembles like stacking for the best performance.

**Metrics for Evaluation:**

* **MAE, RMSE**: For continuous target (price prediction).
* **R² Score**: For overall fit.

While working on the supervised learning methods, we made the following changes to the data:

1. Previously, the Temporal Feature was calculated based on the variables Day, Weekday, Month, and Year. Since our model aims to predict for swing trading, we realized that the Day and Year features are not necessary, as they are more focused on intraday and positional trading. Therefore, we removed both the Day and Year features.

2. The previous Temporal Feature was derived using PCA reduction of the aforementioned features, which did not account for the fact that the day after the 31st is the 1st or that January follows December. To address this, we implemented Cyclical Encoding. This method transforms the features into sinusoidal functions (sine and cosine), ensuring:

a. A continuous transition between the start and end of the cycle.

b. The closeness of adjacent values is preserved (for example, December and January remain close in the encoded space).

3. We then combined the sine and cosine components of each feature into a single value using trigonometric relationships, such as the arctangent. This value was subsequently reduced using PCA to form the Temporal Feature.

4. We discovered that the lagged features Adj\_close\_lag\_1, 2, and 3 were not significant. Additionally, the lags up to 10, 50, and 200 had already been accounted for while analyzing Upward and Downward probabilities through SMA and EMA. As a result, we removed the price features associated with these lags.

Different ways to handle the models for multiple stocks:

A common challenge when working with multiple stocks, as different stocks can exhibit distinct patterns, requiring different models or hyperparameters. Here’s how you can tackle it:

**1. Meta-Modeling (Model Selection per Stock)**

* **Approach**: Train multiple models (e.g., Random Forest, XGBoost, LSTM) for each stock during an offline phase, and dynamically select the best-performing model for each stock during prediction.
* **Implementation**:
  1. Maintain a **mapping of stocks to models** based on historical performance.
     + Example: Stock A → XGBoost, Stock B → LSTM.
  2. During prediction, load and use the best model for the stock.

**2. Hyperparameter Tuning by Stock**

* **Approach**: Use automated tools like **Optuna** or **Bayesian Optimization** to tune hyperparameters for each stock and save configurations.
* **Implementation**:
  1. Optimize hyperparameters per stock offline.
  2. Store the tuned hyperparameters in a database or configuration file.
  3. During prediction, load the stock-specific hyperparameters dynamically.

**3. Clustering Stocks by Behavior**

* **Approach**: Group stocks with similar behaviors (e.g., based on volatility, sector, or technical indicators) and use a single model for each cluster.
* **Implementation**:
  1. Cluster stocks using **K-Means, DBSCAN**, or other clustering algorithms.
  2. Train one model per cluster and use it for all stocks in that cluster.
  3. Assign new stocks to clusters dynamically based on their features.

**4. Ensemble of Models**

* **Approach**: Use an ensemble of models for each stock and take a weighted average of their predictions.
* **Implementation**:
  1. Train multiple models (e.g., Random Forest, LSTM, XGBoost) for each stock.
  2. Combine their predictions using weights derived from validation performance.

**5. Universal Model with Feature Scaling**

* **Approach**: Train a single model for all stocks but add stock-specific features to account for differences.
* **Implementation**:
  1. Add stock-specific identifiers as categorical features (e.g., one-hot encode stock names or sectors).
  2. Use features like volatility, sector, or market cap to help the model generalize.

**6. Online Learning Models**

* **Approach**: Use online learning models (e.g., incremental learning in XGBoost, online gradient descent) to update the model for each stock dynamically.
* **Implementation**:
  1. Train a base model offline.
  2. Update it incrementally with new data for each stock during runtime.

**7. Model Orchestration Framework**

* **Approach**: Build an orchestration system to dynamically load models and hyperparameters for each stock.
* **Implementation**:
  1. **Store Models and Hyperparameters**:
     + Use a database to store pre-trained models and configurations for each stock.
  2. **Dynamic Loading**:
     + Load the required model and hyperparameters for each stock when needed.
     + Use libraries like **MLflow** or **TensorFlow Serving** for model versioning and deployment.

**Workflow Example:**

1. **Offline Phase**:
   * Train multiple models or tune hyperparameters per stock/cluster.
   * Save models and configurations in a database.
2. **Online Phase**:
   * For each stock:
     + Identify the stock or its cluster.
     + Load the appropriate model and hyperparameters.
     + Predict and repeat.

**Tools to Help Automate:**

* **MLflow**: Manage multiple models and their versions.
* **Optuna/Hyperopt**: Automate hyperparameter tuning.
* **Docker**: Package stock-specific models into containers for fast deployment.
* **Ray Serve**: Deploy and scale model inference across multiple stocks.

**We are going to add the below features in the ML model:**

1. Time Features (Date breakdown):  
Time features like day, month, year, and weekday provide seasonality and trend insights, which are crucial for stock market predictions:

* Seasonality: Stock prices often show patterns based on time, such as month-end effects, weekday effects (e.g., higher volatility on Mondays), or quarterly earnings announcements.
* Holidays and Cycles: Breaking down the date helps the model understand patterns tied to specific time frames (e.g., December rally or tax seasons).
* Yearly Trends: Incorporating the year helps capture long-term growth or decline trends in the market.

Why useful? It gives the model a temporal context to make predictions more accurate.

2. Lag Features (e.g., Close\_Lag\_1, Close\_Lag\_2):  
Lag features use past values of stock prices or indicators to inform predictions. They are important because:

* Trend Continuity: Stock prices often follow trends influenced by momentum, so yesterday's price is highly indicative of today's price.
* Autocorrelation: Historical prices have a natural correlation with future prices, which the model learns.
* Momentum and Mean Reversion: Lagged features help capture momentum effects (rising prices tend to continue rising) or mean-reversion effects (prices returning to average).

Why useful? It helps the model learn the sequential nature of stock prices and how past trends influence the future. Without lag features, the model might miss key historical patterns.

**In the next step, we will proceed with the selection of features,**

We have 3 feature sets to test in unsupervised machine learning to determine Buy, Sell, and Hold as follows,

1. Adj Close, upward\_downward\_probability

2. Adj Close, Open, high, low, upward\_downward\_probability

3. Adj Close, Day, weekday, month, year, Adj\_Close\_lag\_1, Adj\_Close\_lag\_2, Adj\_Close\_Lag\_3, upward\_downward\_probability

**Evaluation of Feature Sets:**

**Feature Set 1: Adj Close, upward\_downward\_probability**

* **Advantages:**
  + Simple and less likely to create noisy clusters.
  + upward\_downward\_probability is already a signal combining multiple indicators, which can drive meaningful clustering.
* **Disadvantages:**
  + Lack of additional context (e.g., historical prices, volatility) may lead to oversimplified clusters.
  + Clusters may fail to capture nuanced patterns like momentum or seasonal trends.

**Feature Set 2: Adj Close, Open, High, Low, upward\_downward\_probability**

* **Advantages:**
  + Includes key price-related features (Open, High, Low), which can help distinguish between volatile and stable trading days.
  + Captures more intraday price dynamics compared to Feature Set 1.
* **Disadvantages:**
  + **High collinearity** among Adj Close, Open, High, and Low could skew clustering results.
  + Requires feature engineering (e.g., ratios or differences) or dimensionality reduction (e.g., PCA) to mitigate redundancy.

**Feature Set 3: Adj Close, Day, Weekday, Month, Year, Adj\_Close\_Lag\_1, Adj\_Close\_Lag\_2, Adj\_Close\_Lag\_3, upward\_downward\_probability**

* **Advantages:**
  + Combines temporal trends with historical context (lagged prices) and upward\_downward\_probability.
  + Lagged features can capture momentum, making clusters more representative of real market patterns.
  + Less collinearity compared to Feature Set 2.
* **Disadvantages:**
  + Temporal features like Day, Weekday, and Month might not add much value for unsupervised models unless strong seasonal patterns exist.
  + Higher dimensionality could dilute the impact of critical features (Adj Close, upward\_downward\_probability) unless dimensionality reduction is applied.

**Recommendation for Unsupervised Models**

**Best Option:** **Feature Set 3 (with preprocessing)**

* + Lagged features (Adj\_Close\_Lag\_1, Lag\_2, Lag\_3) combined with upward\_downward\_probability allow the model to identify momentum-based patterns (e.g., trends leading to Buy/Sell signals).
  + Temporal features (Day, Weekday, Month, Year) add broader patterns (e.g., weekday effects, end-of-month rallies).
  + **Preprocessing:** Standardize all features and reduce dimensionality (e.g., PCA or t-SNE) to avoid noise from temporal data.

**Next Steps**

* Use Feature Set 3, but preprocess the data:
  1. **Standardize Features:** Scale features like Adj Close, upward\_downward\_probability, and Adj\_Close\_Lag\_\*.
  2. **Reduce Dimensionality:** Use PCA to combine collinear features or t-SNE for visualization.
  3. **Test Clusters:** Visualize clusters in 2D/3D and interpret the results (e.g., mapping clusters to Buy/Sell/Hold).