□──────□──────□──────□──────□  
**1. Data Collection**:  
□──────□──────□──────□──────□

* We will collect stock data using the **Yahoo Finance API** with the yfinance library.
* Focus on daily data for a set universe of stocks (Example: S&P 500 or small-cap stocks).
* **Notes: Column Details**
  + **Data returned after setting**tickers start\_date and end\_date will include the following columns…
    - **Adj Close (Adjusted Close)**: The closing price adjusted for corporate actions like stock splits and dividends. It’s the most accurate reflection of the stock’s value for historical analysis.
    - **Close**: The price of the stock at the end of the trading day. This does not account for any adjustments, unlike Adj Close.
    - **High**: The highest price the stock reached during the trading day.
    - **Low**: The lowest price the stock reached during the trading day.
    - **Open**: The price at which the stock began trading at the start of the day.
    - **Volume**: The total number of shares traded during the day. It gives insight into the level of activity or interest in the stock.

□──────□──────□──────□──────□  
**2. Data Preprocessing**:  
□──────□──────□──────□──────□

* **Goal**: Ensure that data is clean, complete, and ready for model training.
* **Actions to Take**:
  1. **Fill missing values**: Handle NaN values by forward-filling and backward-filling to make sure no missing data impacts the analysis.
  2. **Replace infinite values**: Ensure that any extreme values (like inf) are replaced with valid values (e.g., NaN) and are properly filled.
  3. **Align data**: Align the lengths of all calculated indicators (volatility, RSI, SMAs, Bollinger Bands, support/resistance) by filling forward and backward where necessary.
  4. **Create Actions column**: Create function to calculate best action if we knew tomorrows closing price.

□──────□──────□──────□──────□  
**3. Exploratory Data Analysis (EDA)**:  
□──────□──────□──────□──────□

* Analyze the stock data to identify patterns, trends, and insights that can inform model development and feature selection.

□──────□──────□──────□──────□  
**4. Feature Engineering**:  
□──────□──────□──────□──────□

* Create features that can improve the model’s ability to predict stock prices. These features are based on technical indicators used in stock analysis.
* Create a binary target variable to indicate whether the stock will move up (bullish) or down (bearish).

**Features X**:

1. **Volatility**:

* **Why**: Volatility measures the rate at which a stock’s price increases or decreases for a given set of returns. It helps gauge the risk of the stock.
* **How**: Calculate the percentage change of the stock prices.
* **Handling missing/infinite values**: Replace any inf values with NaN and fill missing values forward and backward to align data.

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

* **Why**: RSI measures the speed and change of price movements, helping to identify overbought or oversold conditions.
* **How**: Apply an RSI calculation function to the data.

1. **SMA (Simple Moving Averages)**:

* **Why**: SMAs help smooth out price data to identify trends over specific periods (50, 100, 200 days).
* **How**: Compute the rolling averages for the respective time windows.

1. **Bollinger Bands**:

* **Why**: Bollinger Bands are used to measure the high and low of price volatility in relation to moving averages, helping to identify overbought and oversold conditions.
* **How**: Use a predefined function to calculate the upper and lower Bollinger Bands.

1. **Support and Resistance**:

* **Why**: These levels indicate where the price tends to find support as it falls or resistance as it rises, important for predicting price reversals.
* **How**: Use rolling windows to find the minimum (support) and maximum (resistance) prices over a 50-day period.

**Target y (Action signals):**

* **Why**: The model needs a target variable to learn from. In this case, we aim to predict whether the stock will move up or down and based on that either buy, sell, short, or hold our position.
* **How**:
* Generate a binary target variable: If the stock’s closing price for the next period is higher than the current period’s, the target will be 1 buy or if already bought 2 hold. If the predicted price is lower we will 3 sell, 4 short the market
* This approach simplifies the prediction task to a classification problem, focusing on the direction of movement rather than the exact price.

□──────□──────□──────□──────□  
**5. Data Splitting**:  
□──────□──────□──────□──────□

* Split the data into training, validation, and test sets.
* Ensure no look-ahead bias by correctly handling time-based data.

**Important Notes:**

1. **Handling Time-Based Data**:
   * Stock market data is inherently time-dependent, so we can’t shuffle it randomly like in typical machine learning tasks.
   * **How**: The split should be **sequential** based on the dates.
     + We train the model on past data, validate it on more recent data, and test it on the most recent data to simulate real-world conditions.
2. **Avoiding Look-Ahead Bias**:
   * Ensure that the training set only contains data up to a certain point in time, and the validation and test sets are strictly later in time.
   * The model **only learns from past data** and never sees future information, making predictions based solely on historical trends.

□──────□──────□──────□──────□  
**6. Model Selection**:  
□──────□──────□──────□──────□

* To choose the appropriate classification model that can effectively predict stock price trends (bullish or bearish) based on the features and indicators generated in the previous steps.
* **Possible Options:**
  + Logistic Regression
  + Decision Trees
  + Random Forest
  + Support Vector Machines (SVM)
  + Gradient Boosting Machines (GBM) or XGBoost

□──────□──────□──────□──────□  
**7. Model Training**:  
□──────□──────□──────□──────□

* Train the selected model using historical stock data, focusing on predicting future bull/bear movements.

□──────□──────□──────□──────□  
**8. Model Evaluation**:  
□──────□──────□──────□──────□

* Evaluate model performance using validation data and metrics like accuracy, precision, recall, and F1 score.
* Aim for a win rate of above 70% ensures that the model is sufficiently reliable for making trading decisions.

□──────□──────□──────□──────□  
**9. Hyperparameter Tuning**:  
□──────□──────□──────□──────□

* Fine-tune model parameters (e.g., regularization strength, tree depth) to optimize prediction performance.

□──────□──────□──────□──────□  
**10. Model Deployment**:  
□──────□──────□──────□──────□

* Implement a bot that makes trading decisions (buy or sell) based on the model’s predictions.

□──────□──────□──────□──────□  
**11. Monitoring and Maintenance**:  
□──────□──────□──────□──────□

* Monitor the model’s performance, adjusting for market changes or new data to keep the bot effective.