Stock Trading Decision Prediction

TRIVIKRAM TRISHIR RAO PADALA
Roll_No: 23AE10038

BRIEF EXPLANATION:

This Python script uses historical stock price data to predict trading decisions (Buy, Sell, or Hold) using technical indicators such as Simple Moving Averages (SMA), Exponential Moving Averages (EMA), MACD (Moving Average Convergence Divergence), and Bollinger Bands, along with Standard Deviation (STD).

Functionality Overview:

get_stock_data(symbol, start, end):
Retrieves historical stock price data from Yahoo Finance for the specified symbol and date range.
Parameters:
symbol: Ticker symbol of the stock (e.g., 'AAPL' for Apple Inc.).
start: Start date in the format 'YYYY-MM-DD'.
end: End date in the format 'YYYY-MM-DD'.
Returns:
Pandas DataFrame containing the stock price data.

calculate_technical_indicators(data):
Calculates various technical indicators such as Simple Moving Averages (SMA), Exponential Moving Averages (EMA), MACD Line, Signal Line, Bollinger Bands, and Standard Deviation (STD).
Parameters:
data: Pandas DataFrame containing the stock price data.
Returns:
Updated DataFrame with calculated technical indicators.

generate_signals(data):
Generates trading signals (Buy, Sell, or Hold) based on the calculated technical indicators.
Parameters:
data: Pandas DataFrame containing the stock price data with technical indicators.
Returns:
DataFrame with additional 'Signal' column indicating the trading decision.

train_model(data):
Trains a Random Forest classifier using the technical indicators as features and the trading signals as the target variable.
Parameters:
data: Pandas DataFrame containing the stock price data with technical indicators and trading signals.
Prints:
Model accuracy score.

main():
Main function to execute the entire workflow:
Retrieves stock data.
Calculates technical indicators.
Generates trading signals.

Trains the prediction model.
Plots the stock price data with Buy/Sell signals.

Additional Indicator:

Standard Deviation (STD):
Calculates the volatility of the stock price.
Used in the calculation of Bollinger Bands to determine the upper and lower bands.

Dependencies:

numpy
pandas
yfinance
matplotlib
scikit-learn

Usage:
Replace the symbol, start_date, and end_date variables in the main() function with your desired stock symbol and date range.
Run the script to visualize the stock price data with predicted Buy/Sell signals and evaluate the model accuracy.

INTRODUCTION:
In this project I used Simple Moving Average (SMA), Exponential Moving Average (EMA), Moving average convergence/divergence (MACD) and Bollinger Bands (BB) as technical indicators.

DATA ASSORTMENT:
The standard definitions have been applied to the indicators to formulate the data and add to the cumulative dictionary of all data obtained from yfinance.

SIGNAL GENERATION:
I used the weighted average technique to decide the final signal I.e, to Buy, Sell or Hold the asset.
I assumed the momentum of the stock to have a greater impact on the decision while the SMA and EMA have an equal impact and the Volatility shown by the Bollinger Bands to have the least affect.

Although I tried to develop an algorithm to automatically assign the weights based on the data, my inexperience led me to invest large amounts of time to study how to do this and due to the busy schedule during these few weeks I could not further this plan.

LOGISTIC REGRESSION:
The logistic regression function used in the provided code is a binary classifier that predicts the probability of a binary outcome based on one or more predictor variables. Here's a brief overview of the logistic regression function:

Purpose: Logistic regression is commonly used for binary classification problems, where the target variable has two possible outcomes (e.g., yes/no, true/false, 0/1).

Model Representation: The logistic regression model calculates the probability that a given input belongs to a particular class. It models the relationship between the independent variables (features) and the binary dependent variable using the logistic function.

Sigmoid Function: The logistic regression model uses the sigmoid function to map the linear combination of features to a probability value between 0 and 1. The sigmoid function ensures that the predicted probabilities are bounded and interpretable.

Training: The model is trained using an optimization algorithm (such as gradient descent) to find the optimal values for the model parameters (weights and bias). During training, the model adjusts its parameters to minimize the difference between the predicted probabilities and the actual class labels in the training data.

Prediction: Once trained, the logistic regression model can make predictions on new data by computing the probability that each observation belongs to the positive class (e.g., class 1). The model then assigns the observation to the positive class if the predicted probability exceeds a certain threshold (usually 0.5).

Evaluation: The performance of the logistic regression model can be evaluated using various metrics such as accuracy, precision, recall, F1 score, and the receiver operating characteristic (ROC) curve. These metrics help assess how well the model is able to correctly classify instances into the correct classes.

Overall, logistic regression is a versatile and widely used classification algorithm that is especially useful for problems with binary outcomes and interpretable predictions.