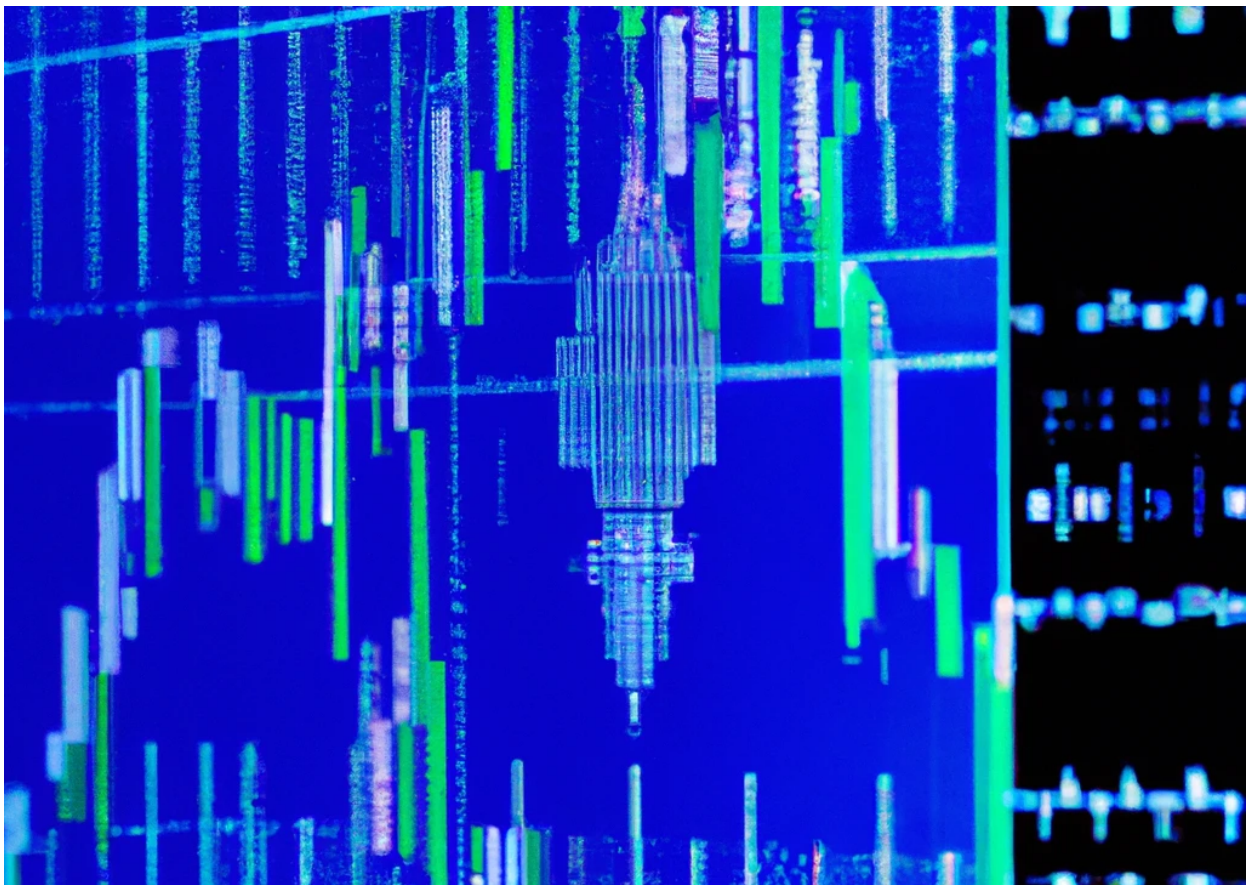
USE OF MACHINE LEARNING FOR AUTOMATED TRADING

CS354N: Minor Project



200001023 Gaurav Jain

200001079 Vihaan Thora

INTRODUCTION

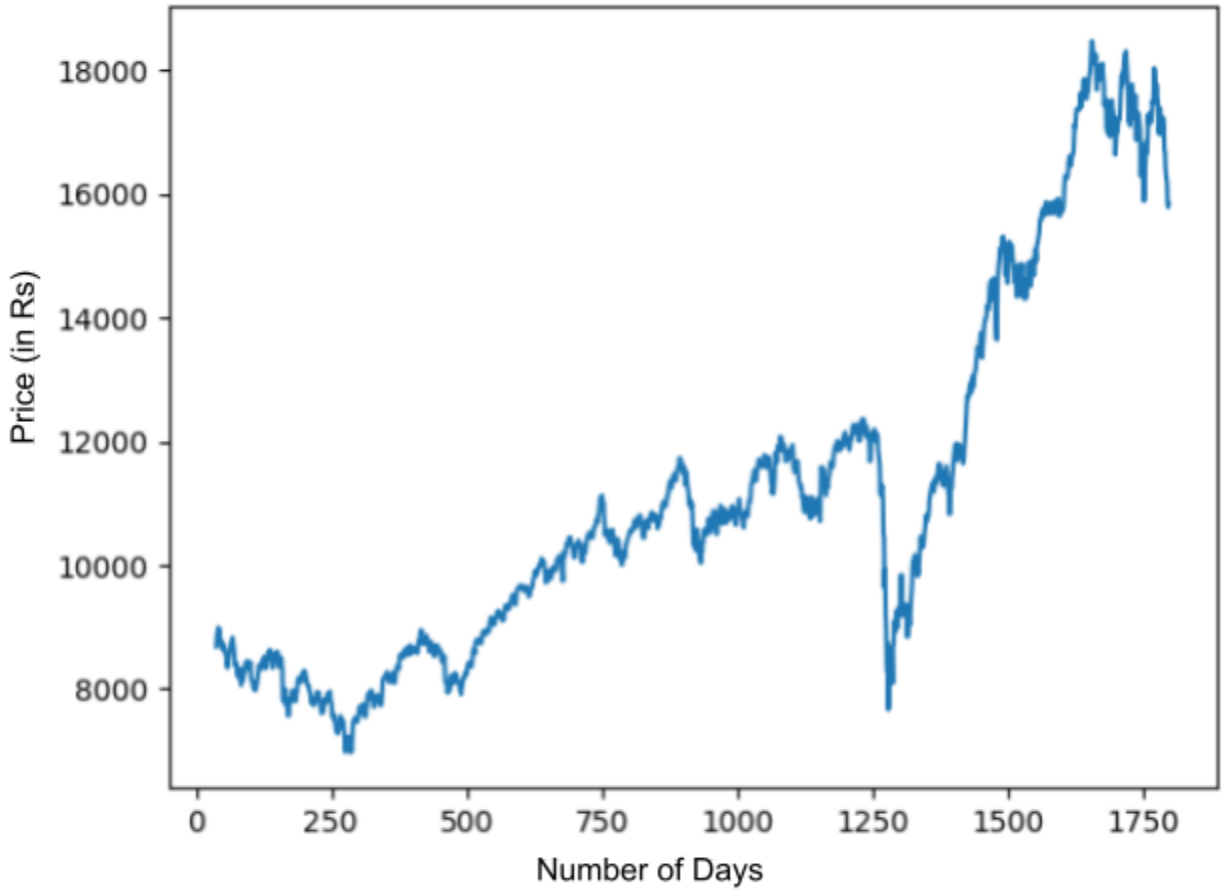
Automated trading has been gaining popularity in recent years, especially with the advancements in machine learning. The use of machine learning algorithms in trading has opened up a new dimension in the field of finance. In this project, we have leveraged the power of machine learning to predict stock prices and identify profitable trades. Our approach involves using technical analyzers as the input features for our machine learning models.

In our project, we have used various algorithms such as Random Forest, Extra Trees Classifier, and Gradient Boosters to predict price trends. These models have been trained on a dataset that comprises historical stock prices and their corresponding technical analyzers. Our aim is to use these models to achieve accurate predictions that can guide us towards profitable trading decisions.

Dataset

The success of our machine learning models heavily depends on the quality and relevance of the data used to train them. In our project, we have used a dataset that comprises both input data features and synthesized data features.

To train our machine learning models, we have used a dataset comprising historical stock prices and their technical analyzers. The dataset includes a range of technical indicators such as Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), and Bollinger Bands, among others. We have obtained the data from reliable sources and have ensured that it is up-to-date and accurate.



Input Data Features

The input data features used in our dataset include Open, Close, High, Low, and Volume (optional) of the stock prices. These features provide essential information about the stock prices, such as the opening and closing prices, the highest and lowest prices during the trading period, and the trading volume.

Synthesized Data Features:

In addition to the input data features, we have also generated synthesized data features to provide additional information that can improve the accuracy of our machine learning models. These synthesized data features include:

Simple Moving Averages (SMA):

SMA is a commonly used technical indicator that helps smooth out price data by creating a constantly updated average price over a specific time period.

Exponential Moving Averages (EMA)

EMA is similar to SMA, but it puts more weight on the most recent prices, making it more responsive to recent price changes.

Bollinger Bands

Bollinger Bands are a volatility indicator that consists of three lines. The middle line is a simple moving average, while the upper and lower lines are the standard deviations of the moving average.

Moving Average Convergence Divergence (MACD)

MACD is a trend-following momentum indicator that shows the relationship between two moving averages of the stock price.

Rate of Change (ROC)

ROC is a momentum oscillator that measures the percentage change in price between the current price and the price n periods ago.

Relative Strength Index (RSI)

RSI is a momentum oscillator that measures the strength of the stock price by comparing the average of the gains to the average of the losses over a specific time period.

Stochastic Oscillators

Stochastic Oscillators are a momentum indicator that compares the closing price of a stock to its price range over a specific time period.

STOCHK and STOCHD

STOCHK and STOCHD are derived from Stochastic Oscillators and are used to identify overbought and oversold conditions.

Average True Range (ATR)

ATR is a volatility indicator that measures the average range of price movements in a given time period.

Kaufman's Adaptive Moving Average (KAMA)

KAMA is a moving average that adapts to the volatility of the stock price.

Vortex Indicators

Vortex Indicators are two lines that measure positive and negative trend movements.

VIm and VIp

VIm and VIp are derived from Vortex Indicators and provide a normalized measure of the trend strength.

Data Pipeline

In Machine Learning projects, it is essential to follow a data pipeline to prepare and preprocess the data for training the models accurately, we followed a data pipeline that involved the following steps:

Data Split

We first split our dataset into training and testing sets. The training set was used to train the models, and the testing set was used to evaluate the models' performance. We used an 80-20 split, where 80% of the data was used for training, and 20% was used for testing.

Feature Generation

We generated synthesized data features from the input data features to provide additional information to our models.

Label Generation

We generated labels for our dataset by comparing the price of the stock at a given time with its price at a future time. If the price increased, we labeled the data as "1", indicating that the stock price went up. If the price decreased, we labeled the data as "0", indicating that the stock price went down.

Smoothening

We used smoothening techniques to remove any noise present in the data. Smoothening helps in removing any sudden spikes or drops in the stock price and makes the data more consistent. We used exponential smoothing to smooth out the input data features and the synthesized data features.

Feature Scaling

We scaled the features to ensure that they had similar ranges and that no feature dominated the others. Feature scaling is necessary because it ensures that the machine learning models give equal importance to all features. We used Min-Max scaling to scale our features, which scales the values between 0 and 1.

By following this data pipeline, we ensured that our dataset was well-prepared and processed, and that our machine learning models would be trained on high-quality and relevant data, leading to accurate predictions.

Model

Our project uses two machine learning models for predicting stock trends: Random Forest and Extra Trees Classifier.

Random Forest Classifier

Random Forest Classifier is a supervised learning algorithm used for classification tasks. It is an ensemble method that creates multiple decision trees and combines their predictions to obtain a final prediction. The process for creating the individual decision trees involves selecting a random subset of features and a random subset of training data for each tree. Each tree is created using a different random subset of features and training data. The trees are grown using recursive binary splitting to find the feature and threshold that best separates the data.

The individual decision trees are used to make predictions. A new data point is passed through each decision tree in the forest, and each tree makes a prediction. The predictions from all trees are combined to obtain a final prediction, which can be done using a simple majority vote. Evaluation can be done using metrics such as accuracy, precision, recall, and F1 score. Cross-validation ensures the model is not overfitting to the training data. Feature importance can be calculated to understand which features are most important for making accurate predictions.

Extra Trees Classifier

Extra Trees Classifier is similar to Random Forest Classifier in that it also involves creating multiple decision trees to make a final prediction. However, Extra Trees Classifier takes randomization to the next level. It randomly selects features and split points and trains multiple trees with randomized splits. The trees are then combined to make predictions. The idea behind this is that the more randomization, the less likely the trees will overfit to the training data. Extra Trees Classifier also provides feature importance, which can be used to understand which features are most important for making accurate predictions.

Both Random Forest and Extra Trees Classifier have their strengths and weaknesses, and their effectiveness may vary depending on the dataset and the problem being solved. For our project, we experimented with both models to see which one performed better on our dataset.

RESULTS

Random Forest	Extra trees	Gradient Boosting	Bagging	Stacking
66.9%	69.9%	63%	61%	67.8%

PERFORMANCE METRICS

Returns

The Annual Return measures the total percentage change in an investment's value over a year, including capital gains and losses. It is calculated by dividing the current value of the investment by its original value, subtracting 1, and multiplying by 100 to get the percentage return.

Sharpe Ratio

The Sharpe Ratio is a measure of risk-adjusted return. It is calculated by subtracting the risk-free rate of return from the investment's average return and dividing the result by the standard deviation of the investment's return. The higher the Sharpe ratio, the better the risk-adjusted performance of the investment.

Maximum Drawdown

Maximum Drawdown is the maximum percentage decline in an investment's value from a previous high. It is a measure of the investment's risk and is used to calculate the Calmar ratio.

Calmar Ratio

The Calmar Ratio measures risk-adjusted return that compares an investment's average annual return to its maximum drawdown, the largest percentage decline in its value from a previous high. The higher the Calmar ratio, the better the risk-adjusted performance of the investment.

Algorithm

To train our machine learning models, we have used an algorithm that involves splitting the dataset into training and testing sets. We have used the training set to train our models and the testing set to evaluate their performance. We have also used cross-validation to ensure that our models are not overfitting the training data.

Training

To ensure a fair and accurate fitting result, we have used a sliding window approach to training and testing.

The default window size taken is of 40 days, reflecting two financial months of the dataset. We have used Extra Trees Classifier as a model to predict the labels.

The default lookahead of the labels is taken as 10 days (2 weeks), which means that our model will be trained to predict whether the stock will go up or down after two weeks.

We have evaluated the performance of our model using different metrics, including Random Forest, Extra Trees, Gradient Boosting, Bagging, and Stacking Ensemble.

Trading Strategy

We have used the Extra Trees Classifier's trained model as an input to our algorithm. The dataset that is passed is unlabeled but contains all the synthesized features. The principal amount denotes the amount of money available to purchase the shares. The trading window size denotes the days after which the predicted label is obeyed. Maximum volume is set such that any major fault in the price would not lead to a large drawdown value.

Confidence is an essential indicator that determines the stock volume that needs to be traded in a position. We have updated the confidence level using the following rule:

$$confidence = (1 - \alpha) \times confidence + \alpha \times result$$

Here, the α parameter denotes the weightage given to the current prediction, and the $result$ signifies if the current prediction was correct or not. This approach ensures that the confidence level is updated dynamically, based on the algorithm's accuracy.

The max shares that can be traded on a day are determined by the rule:

$$max\_shares = volume \times confidence^{n}$$

Here, the n parameter denotes the number of days since the last trading activity. Hence, the number of shares that can be bought or sold is determined by the balance available and the above value.

Conclusion

In conclusion, our algorithm is designed to make profitable trading decisions automatically, based on the predicted labels obtained from the Extra Trees Classifier. Our training approach and evaluation metrics ensure that the model is accurate and robust, and the trading strategy is based on sound principles of money management. However, it is essential to note that the financial market is unpredictable and subject to various risks, and our algorithm is not guaranteed to achieve profits in all situations. Nonetheless, we believe that our algorithm can provide valuable insights and increase the chances of achieving higher profits in this dynamic and complex domain.

Back Testing

To validate the performance of our machine learning models, we have conducted backtesting. Backtesting involves testing a trading strategy on historical data to see how it would have performed in the past. We have used our models to generate trading signals based on their predictions and have tested these signals on historical data to see how profitable they would have been.

Back testing is an important step in evaluating the effectiveness of a trading strategy. It involves testing the strategy on historical data that the model has not been trained on to assess its performance and potential profitability.

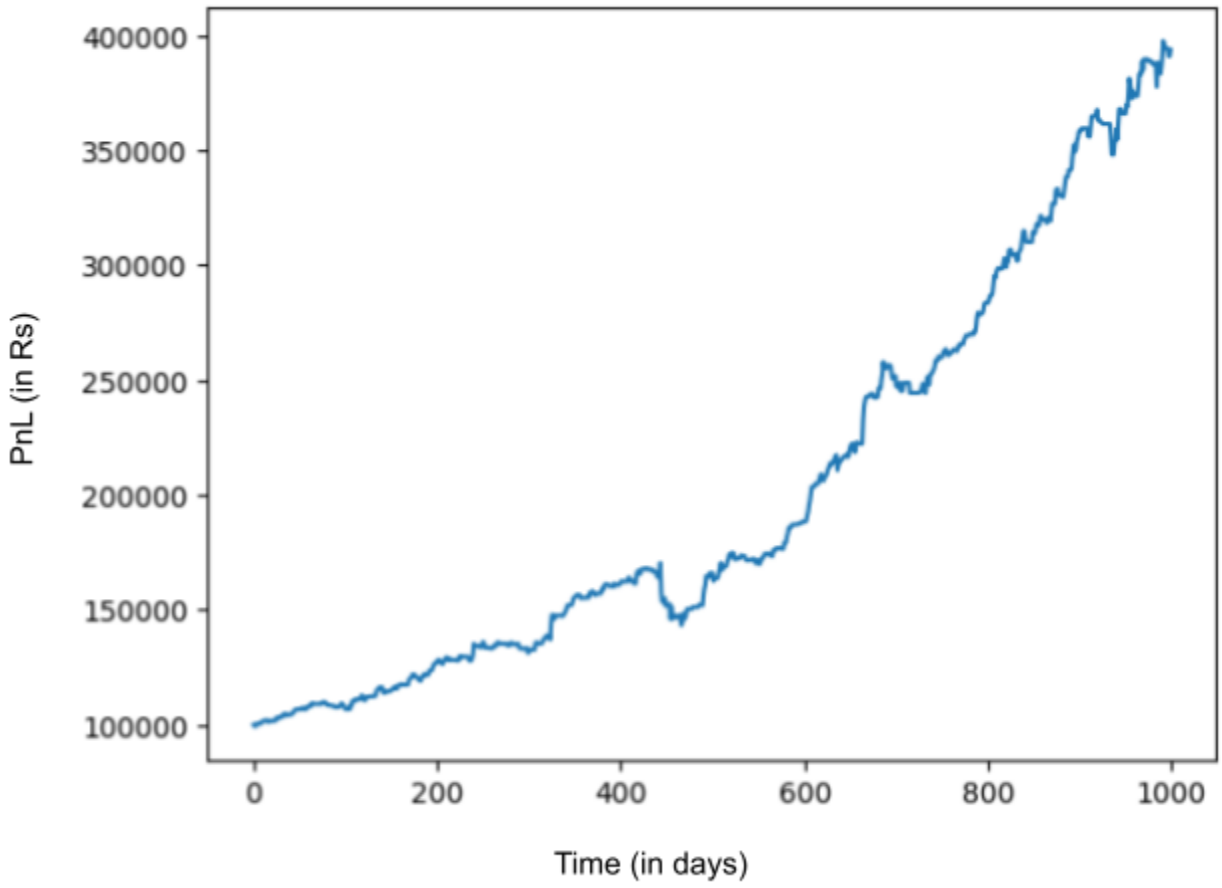
In our case, we back test our trading strategy on four years of unseen data using the Extra Trees Classifier model. The strategy is trained on 18 years of data (2000-18) to make predictions on the unseen data.

As an example test case, we consider an investment amount of INR 100000, a maximum volume of 70 shares, a window size of 5 days, a confidence rank of 0.7, and an n value of 2.4.

We also take into account a risk-free rate of 15% while calculating the Sharpe Ratio. This rate represents the average increment in the NIFTY 50 index price annually and serves as a benchmark to evaluate the performance of our trading strategy.

By testing our strategy on unseen data, we can obtain a more accurate estimate of its potential profitability and risk-adjusted performance. This helps us identify any weaknesses or areas for improvement in our model and trading strategy.

RESULTS



Annual Returns	Cumulative Returns	Sharpe Ratio	Max Drawdown
41.26%	293.85%	1.72	-15.87%

Further Scope

Our project has focused on predicting stock prices using machine learning. However, there is a lot of scope for further research in this area. One area of future research could be to explore the use of deep learning models for stock price prediction. Deep learning models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks have shown promising results in other domains and could be explored for stock price prediction as well.

Another area of future research could be to explore the use of alternative data sources such as social media data and news articles for stock price prediction. These alternative data sources could provide valuable insights that could be used to improve the accuracy of our models.

Machine Learning Perspective

Feature Correlation

Identifying the degree of correlation between features is important as it can help in avoiding the issue of multicollinearity in models. Multicollinearity refers to the correlation between predictor variables that can lead to issues such as unstable model coefficients and reduced predictive power. By analyzing the correlation between features, we can eliminate redundant features and ensure that the model is not overfitting.

Backwards Elimination

Another technique that can help in reducing model complexity is backwards elimination. This technique involves iteratively eliminating features based on their p-value, which determines their statistical significance in the model. The feature with the least significance is eliminated in each iteration until the desired level of significance is achieved.

Feature Normalization

Scaling features to a common range is important to avoid bias towards features with higher magnitudes. This can be done using techniques such as min-max scaling or standardization. Normalizing the features can help in achieving better model performance and can also make the model more interpretable.

Finance Perspective

Neutralization

Taking a neutral position with respect to a sector can help in diversifying the portfolio and reducing the impact of unexpected market events. By balancing the portfolio across different sectors, investors can reduce their exposure to the risks associated with individual sectors.

Shorting

Shorting stocks can be used to improve returns and Sharpe ratio. Short selling involves borrowing shares from a broker and selling them with the expectation that the price will decline, allowing the shares to be bought back at a lower price and the profit to be pocketed. Short selling can be a risky strategy, but it can also provide opportunities for investors to profit in bearish markets.

Volume Function

Calculating share volume using polynomial regression is a useful technique that can help in optimizing resource allocation. The polynomial takes confidence as an argument, which determines the volume of shares that should be traded. The coefficients of the polynomial can be optimized using techniques such as grid search to maximize returns and minimize risk. By optimizing the volume function, investors can ensure that they are making the most efficient use of their resources.

References

- https://towardsdatascience.com/predicting-future-stock-market-trends-with-python-machine-learning-2bf3f1633b3c
- https://towardsdatascience.com/deep-reinforcement-learning-for-automated-stock-trading-f1dad0126a02
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html
- https://blog.quantinsti.com/mean-reversion-time-series/
- https://irep.ntu.ac.uk/id/eprint/32787/1/PubSub10294_702a_McGinnity.pdf