

**Title:****Machine Learning Driven Algorithmic Trading: A Symbol-Model Search Engine Approach**

Sahand Hassanizorgabad / [shassanizorgabad22@ku.edu.tr](mailto:shassanizorgabad22@ku.edu.tr)

**Abstract:**

The financial market is the arena where buyers and sellers engage in transactions to acquire and sell diverse assets, aiming to maximize their Return on Investment (ROI). In the financial market, prices of various assets experience fluctuations based on the fundamental principles of supply and demand. The financial market is nonlinear in nature and prediction of financial market trends is a tedious task because it can easily get affected by a lot of parameters such as stock and company-specific news, company profile, public sentiment, global economy and etc. Over many years, predicting the stock market trend has been a challenging problem for most data scientists.

The core of financial research lies in predicting the future price trend, whether it's bullish or bearish. This project is geared towards forecasting whether an instrument will rise or fall on a daily basis. This is achieved through the implementation of classification models on time-series data, followed by a second classification model to identify which instruments and their models have the potential to be profitable.

Firstly, the project acquires lists of daily instruments data from yfinance library. To define the label up and down, the project compares the daily closing price with its yesterday price. If it is positive, it is recorded as up; if it is negative, it is recorded as down. Multiple classification models are established, each with hyper-parameters selected through the grid search technique to predict the next day label.

The list of profitable models will be selected by comparing all models' evaluation scores based on their accuracy, precision, backtest results, and other evaluation metrics applied to the validation dataset. This approach uses multiple classification models, and the one with higher sum of accuracy and precision will be selected to identify the best model.

The project has the potential for enhancement through the integration of pattern recognition using Convolutional Neural Networks (CNN) and analysis of fundamental

news using Natural Language Processing (NLP). Despite its current unfinished state, the project demonstrates the capability to identify promising opportunities.

## **Keywords:**

Algorithmic Trading , Stock Prediction , Financial Forecasting , Supervised , Machine learning , Stock time series forecasting , Machine learning , Backtesting

## **Introduction:**

Financial data is often characterized by its complexity and volume. It includes a wide range of data types such as market prices, economic indicators, trading volumes, and more. Analyzing and extracting insights from such large and intricate datasets requires advanced data science techniques. One of the use cases of the data science is forecasting which means stating a possibility against future based on current datas. In finance, prediction is a key knowledge that can be extracted from chainsaws of price movment in diffrent time frames like hourly,daily,weekly or mountly. Time series aims at using past data of a series of events to develop an appropriate model which describes the inherent structure of a series of events. This model is then used to predict future values for the series. Algorithmic trading, powered by data science, involves using algorithms to execute trades with speed and precision also enables the execution of orders using a set of rules determined by a computer program. Algorithmic trading is important for several reasons, and its significance has grown significantly in the financial industry. Here are key reasons why algorithmic trading is important:

- **Efficiency and Speed:** Algorithms can execute trades at speeds and frequencies impossible for human traders. This high-speed execution is crucial in markets where prices can change rapidly, allowing for timely responses to market conditions.
- **Market Liquidity:** Algorithmic trading contributes to market liquidity by providing continuous buy and sell orders. This increased liquidity benefits all market participants by reducing bid-ask spreads and minimizing price impact.
- **Reduced Transaction Costs:** Automated trading systems can optimize trade execution, reducing transaction costs such as slippage and market impact. Algorithms can efficiently break down large orders into smaller ones and execute them at opportune moments.

- **Risk Management:** Algorithms can incorporate sophisticated risk management strategies to control exposure and mitigate potential losses. This includes setting predefined risk limits and dynamically adjusting trading parameters based on market conditions.
- **Increased Trading Volumes:** Algorithmic trading can handle a large number of orders simultaneously. This scalability contributes to increased trading volumes, fostering more active and efficient markets.
- **Market Making:** Algorithmic trading facilitates market making, where algorithms continuously provide liquidity by quoting bid and ask prices. Market makers profit from the bid-ask spread and contribute to overall market stability.
- **Statistical Arbitrage and Quantitative Strategies:** Algorithms can analyze vast amounts of historical and real-time data to identify statistical patterns and execute trades based on quantitative strategies. This enables traders to capitalize on market inefficiencies and pricing anomalies.
- **24/7 Market Monitoring:** Algorithms can monitor markets around the clock, responding to news, economic indicators, and other events in real-time. This continuous monitoring is crucial in global markets where trading occurs in different time zones.
- **Adaptability to Market Conditions:** Algorithmic trading systems can adapt to changing market conditions and adjust strategies accordingly. This adaptability is valuable in dynamic markets where different strategies may be more effective at different times.
- **Backtesting and Optimization:** Traders can use historical data to backtest and optimize algorithms before deploying them in live markets. This allows for the refinement of strategies and the identification of potential pitfalls.

Overall, algorithmic trading enhances market efficiency, improves liquidity, and provides traders with tools to implement a wide range of strategies. While it introduces complexities and challenges, the benefits it brings to financial markets make it a crucial component of modern trading operations.

In this project, First, yfinance library used to collect daily historical financial instrument (e.g., stocks, bonds, cryptocurrencies). Then in Preprocessing section, new features such as osilators and labels will be added. Labeling is vital to determine upswing or decline. After incorporating additional features, rows are shifted to create a time series datapoint. In this way, each data point will include precise information from previous

rows, along with additional features. Following Preprocessing, Dataframes will be Partitioned into train, validation and test sets before machine learning training.

Sequentially, the core function of this project involves selecting the best classification machine learning model among:

GradientBoostingClassifier, LogisticRegression, GaussianNB, DecisionTreeClassifier, KNeighborsClassifier, RandomForestClassifier, MLPClassifier, SupportVectorClassifier and LinearSVC. This stage involves training all the mentioned models with lists of selected instruments and evaluating them with validation datasets. In the subsequent validation of models, in addition to evaluation scores based on their accuracy, precision, F1, recall, and AUC metrics, a backtesting phase is also utilized. In the backtesting phase, the normal investment return on validation dataset for the considered instrument and the daily trading return with the selected model will be compared to determine how this algorithm can enhance the investment plan. In the final stage, the second classification machine learning model will identify profitable models across all symbols, considering their validation metrics and backtesting results.

The hypothesis behind selecting evaluation metrics is akin to the analogy of flipping a coin. If someone or a company can consistently predict the outcome of a coin flip correctly 51% of the time, they stand to win their bets over a significant number of coin flips. Achieving a prediction accuracy of 51% and above is considered favorable, and leveraging such predictions on a large scale is a crucial factor in obtaining positive results. This analogy illustrates the significance of accuracy in predictions and its application on a broader scale within the context of this project. However, in trading, accuracy alone may not be sufficient for measurement. This is because each data point has different returns. Additionally, if an instrument predominantly has a 70% buy or sell label within a time period, models that consistently predict the majority label can achieve a high accuracy, potentially leading to overfitting. Therefore, other evaluation metrics, such as precision, recall, F1, AUC and backtesting scores, are employed to provide a more comprehensive assessment in addition to accuracy.

Each algorithmic trading system consists of several general components, including fetching data, preprocessing, machine learning, backtesting, presenting and implementing results. The distinctiveness of this project is in its machine learning structure, highlighting its generality. This suggests that the trading system can be applied across various instruments, enhancing any regular investment strategy. It acts as a baseline for other projects and functions as a search engine to identify potential instruments and models.

In the future, should someone desire to develop a specific algorithm for a particular instrument, they can execute this code in a general manner and simultaneously run a distinct algorithm in parallel.

## Literature Review:

The diversity of metrics is one of the reasons why these studies are so difficult to compare with each other.

Related work-1:

In "**Liang, M., Wu, S., Wang, X., & Chen, Q. (2022). A stock time series forecasting approach incorporating candlestick patterns and sequence similarity. Expert Systems with Applications, 205, 117595**", The paper presents a multivariate financial time series stock forecasting model based on sequential pattern mining and sequence similarity. The model improves the accuracy of stock trend forecasting by applying K-line pattern mining to multivariate time series data. The traditional sequential pattern mining is improved based on the morphological characteristics of the K-line and combined with empirical data analysis to improve the poor performance of the classical forecasting model on certain stocks. The sequence similarity proposed in this work is compatible with financial market volatility under the influence of various factors and can effectively locate similar volatility. The model's validity was verified through experiments and analysis of financial time series data of the constituents of the CSI 300 and the CSI 500 indices. The proposed hybrid model could gain 56.05% and 55.56% average accuracy on two datasets, while the SVM model was 51.83%, 51.32%, and the LSTM model was 50.71%, 50.68%. It has gained significant improvements in overall performance. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. Although the hybrid model proposed by paper still could not cover all the stocks in the market.

The current project shares similarities with Related Work-1, encompassing data types, certain evaluation metrics, and the approach output. However preprocessing stage and Machine Learning models are different.

In the preprocessing stage, Related Work-1 proposes Sequential Pattern Mining based on K-line patterns. This involves encoding the price of a data point into Japanese candlestick patterns, assigning labels to them. For example, a sequence like {a, b, e, e} might be considered as indicating a downtrend. In contrast, current project processes the instruments using oscillators utilized by professional traders. This approach provides

machine learning models with more features for training, especially considering that financial datasets have few features.

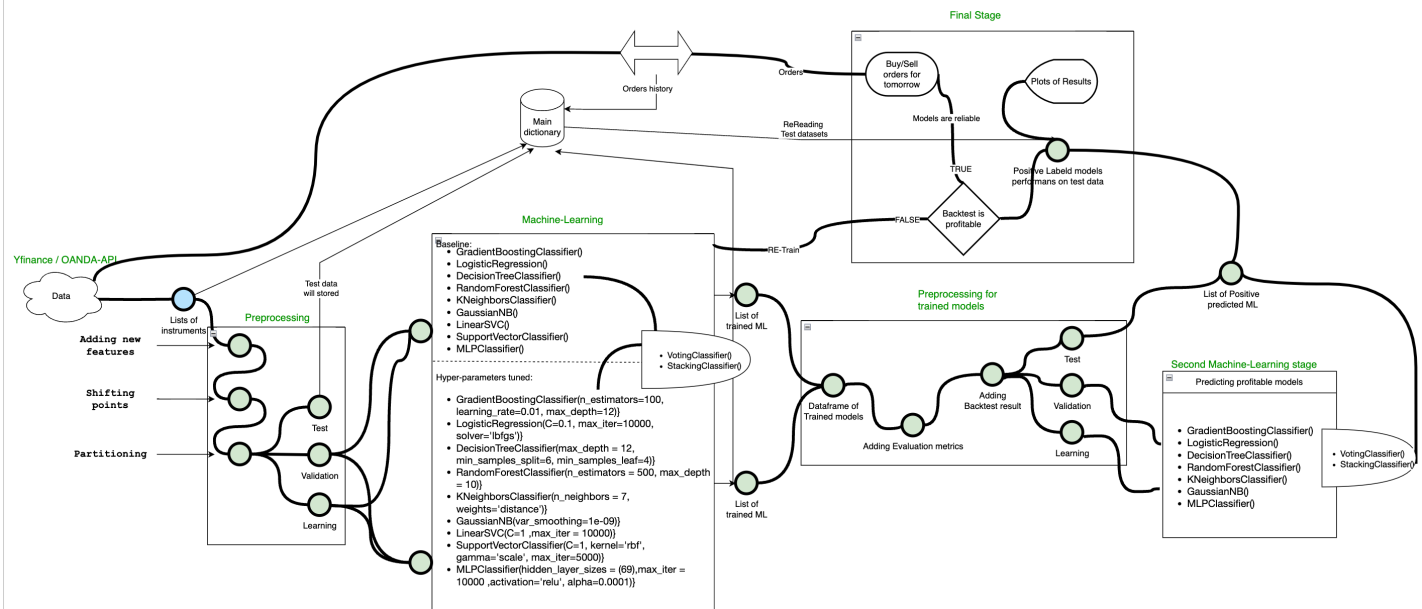
In the evaluation stage, Related work-1 compares the average accuracy of their models by contrasting it with SVM and LSTM models. However, they missed comparing with other models because, as different models yield better results for each instrument. So, a single model can only satisfy a certain instrument for a period of time.

In the system structure section, Related Work-1 created a model to predict the next candle direction, a goal shared by this project. However, their measurement using only accuracy may not be reliable for algorithmic trading, as candles vary in returns, some have almost no return, while others can have over 10% return. On the other hand, this project measures models with more than 10 features. Moreover, this project is more generalized than Related Work-1. While the output of Related Work-1 can be utilized in the preprocessing of second model, functioning as a search engine to determine profitable models.

Other related works:

- RINGMU, H. S., & OUMAR, S. B. (2022). Forecasting stock prices in the New York stock exchange. *Journal of Economics Bibliography*, 9(1), 1-20.
- Rahman, T., Akhter, R., Lawal, K., Mazumder, S. A., Afroz, T., & Rahman, A. (2021). Forecasting and Pattern Analysis of Dhaka Stock Market using LSTM and Phropheet Algorithm. Search in.
- Parente, M., Rizzuti, L., & Trerotola, M. (2024). A profitable trading algorithm for cryptocurrencies using a Neural Network model. *Expert Systems with Applications*, 238, 121806.
- Maheronnaghsh, M. J., Gheidi, M. M., Younesi, A., & Fazli, M. A. (2023). Machine Learning Methods in Algorithmic Trading: An Experimental Evaluation of Supervised Learning Techniques for Stock Price.

In this section, all approaches utilized for inputs, outputs, and the main function of this project are introduced:



**a. Data representation and description:**

The time series data for this project is sourced from the yfinance library and OANDA broker APIs, encompassing price data for supported symbols. Each dataset comprises six quantitative features: Open, High, Low, Close, Adj Close, and Volume.

Open: The opening price of the financial instrument at a specific time period.

High: The highest price reached by the financial instrument during the same time period.

Low: The lowest price reached by the financial instrument during the same time period.

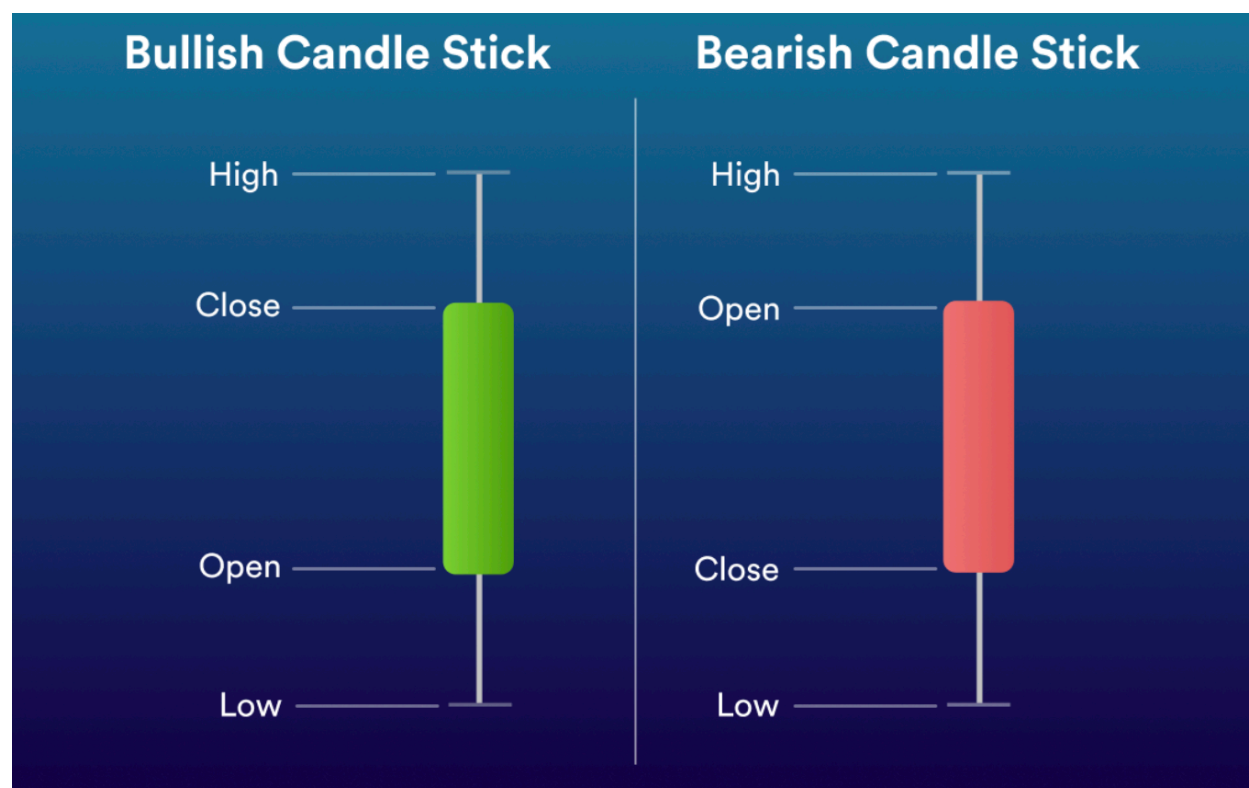
Close: The closing price of the financial instrument at the end of the time period.

Adj Close: The adjusted closing price, which accounts for factors such as dividends and stock splits, providing a more accurate representation of the instrument's value.

Volume: The trading volume, representing the total number of shares or contracts traded during the given time period.

	Open	High	Low	Close	Adj Close	Volume
Date						
2013-12-24	1199.800049	1205.599976	1197.699951	1205.099976	1205.099976	184
2013-12-26	1207.099976	1215.900024	1207.099976	1214.099976	1214.099976	140
2013-12-27	1213.400024	1218.500000	1212.900024	1216.099976	1216.099976	278
2013-12-30	1215.000000	1215.000000	1194.400024	1203.099976	1203.099976	351
2013-12-31	1196.400024	1212.400024	1182.000000	1201.900024	1201.900024	124
...	...	...	...	...	...	...

Traditionally, candlesticks are best used on a daily basis, the idea being that each candle captures a full day's worth of news, data, and price action. This suggests that candles are more useful to longer-term or swing traders.





## **b. Pre-processing and Partitioning steps:**

In the preprocessing step, new features are introduced, including return, label, and indicators. In finance, "return" denotes the profit or loss generated on an investment relative to the invested amount. It serves as a metric for assessing the profitability of an investment and is typically expressed as a percentage.

$$\text{Return} = (\text{Close} - \text{Open}) / \text{Open} \times 100$$

Labels are categorical values representing the project output, taking on values of 0 and 1. A label is assigned the value of 1 if the return on the given day is positive and 0 otherwise.

An indicator refers to a statistical calculation or measurement used to assess and analyze various aspects of financial markets, assets, or economic conditions. Indicators are often derived from financial data and are employed by traders, analysts, and investors to gain insights into market trends, potential price movements, and overall economic health. These indicators can cover a wide range of metrics, including price levels, trading volumes, volatility, and other relevant factors. In this project, the following indicators are utilized:

- Moving Averages:

Moving averages are commonly used to smooth out price data and identify trends over a specific time period. The Simple Moving Average (SMA) is calculated by summing up a set of prices over a specified period and dividing by the number of data points. Formula:

$$\text{SMA} = \text{Number of Periods} / \text{Sum of Prices}$$

- Relative Strength Index:

RSI is a momentum oscillator that measures the speed and change of price movements. RSI values range from 0 to 100 and are often used to identify overbought or oversold conditions.

$$\text{RSI} = 100 - ( 100 / 1 + \text{RS} )$$

where RS (Relative Strength) is the average of 'n' days' up closes divided by the average of 'n' days' down closes.

- Moving Average Convergence Divergence:

MACD is a trend-following momentum indicator that shows the relationship between two moving averages of an asset's price. It consists of the MACD line, Signal line, and Histogram.

MACD Line:  $(12\text{-dayEMA}) - (26\text{-dayEMA})$

Signal Line:  $9\text{-dayEMA of MACD Line}$

Histogram:  $MACD Line - Signal Line$

After adding additional features, each data point will be shifted based on its previous point, creating a time series data structure similar to the one generated by the TimeseriesGenerator function, which will be employed for Keras preprocessing.

Following the collection and preprocessing steps, the dataset will undergo a partitioning process to facilitate model development and evaluation. The data will be divided into training and test sets using an 95-5 ratio, with 95% designated for training purposes which itself will be divided into learning and validation sets using 90-10 ratio, and the remaining 5% reserved exclusively for the final evaluation of the model.

### **c. Learning Approach, the System Structure and hyper-parameter selection:**

The system structure consists of a learning phase, validation evaluation, and the detection of successful models among all trained models for all instruments.

In the learning phase, a dictionary of classification models is employed, including KNeighborsClassifier, LogisticRegression, GaussianNB, DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, SupportVectorClassifier, LinearSVC, and MLPClassifier, are trained both with hyperparameters and without them as a baseline for evaluation purposes. Additionally, VotingClassifier and StackingClassifier are implemented using the mentioned models.

The hyper-parameters of the models are meticulously selected through the implementation of the gridsearchCV technique, a robust approach aimed at optimizing their performance. Consequently, most of the successful models exhibit notable improvements with the inclusion of hyper-parameters. The following hyperparameters are being considered:

- GradientBoostingClassifier( $n\_estimators=100$ ,  $learning\_rate=0.01$ ,  $max\_depth=12$ )}

- `LogisticRegression(C=0.1, max_iter=10000, solver='lbfgs')`
- `DecisionTreeClassifier(max_depth = 12, min_samples_split=6, min_samples_leaf=4)`
- `RandomForestClassifier(n_estimators = 500, max_depth = 10)`
- `KNeighborsClassifier(n_neighbors = 7, weights='distance')`
- `GaussianNB(var_smoothing=1e-09)`
- `LinearSVC(C=1 ,max_iter = 10000)`
- `SupportVectorClassifier(C=1, kernel='rbf', gamma='scale', max_iter=5000)`
- `MLPClassifier(hidden_layer_sizes = (69),max_iter = 10000 ,activation='relu', alpha=0.0001)`

Following the learning phase, the models undergo validation with dedicated datasets, where their performance is assessed using various metrics such as accuracy, backtest, nnp, NormalizedAcc, precision, recall, f1, and auc. These metrics play a crucial role in evaluating the effectiveness of the models and guiding the training of new machine learning model to identify a successful classification model and its instrument to be utilized.

All metrics are sourced from `sklearn.metrics`, except for backtest, nnp, and NormalizedAcc. The backtest metric represents the output of a function that displays the return of the validation dataset using the considered model. Nnp signifies the normal return profit without any machine learning assistance. NormalizedAcc represents normalized accuracy calculated using the following formula:

$$\text{NormalizeAcc} = \text{counts}[1] / (\text{counts}[0] + \text{counts}[1])$$

Certainly, considering the potential issue with models setting all outputs to 0 or 1 to achieve higher accuracy in the presence of imbalanced labels, the formula for NormalizedAcc can be adapted to address this challenge. NormalizedAcc provides a more balanced evaluation, considering the baseline accuracy that could be achieved by random guessing.

After identifying successful models, test datasets are employed to evaluate the models, and their outputs are visualized through plots.

In the implementation part, which will be discussed further, the output, which consists of potential successful models, will be implemented on the OANDA API. These models will predict the direction of their linked instruments for the following day.

#### **d. Baseline:**

The baseline has the same structure as discussed in the system structure without hyper-parameters for these models:

- GradientBoostingClassifier()
- LogisticRegression()
- DecisionTreeClassifier()
- RandomForestClassifier()
- KNeighborsClassifier()
- GaussianNB()
- LinearSVC()
- SupportVectorClassifier()
- MLPClassifier()

#### **Implementation Details:**

This section encompasses the implementation of the project with the OANDA API, consolidated into a single cell. The script is designed to be implemented in a cloud environment, running continuously 24/7. It incorporates a sleep function after code execution, allowing it to rest for a day. Additionally, the script operates exclusively between 22:00 and 23:59 hours. The following steps explain the implementation:

- **Fetching Selected Instruments from Oanda API:**

Retrieve the selected financial instruments data from the Oanda API, including historical prices and relevant market information.

- **Main Function Execution:**

In the main function:

-First, the fetched data undergoes preprocessing, which involves adding oscillators, shifting, labeling, and partitioning, as discussed in the approach section.

-Second, the machine learning steps take place, encompassing model training, hyperparameter tuning, and the collection of model metric scores from the validation dataset.

-Finally, the main function returns test datasets along with trained machine learning models and their corresponding evaluation metric scores.

- **DataFrame of Model Performance Creation:**

Generate a DataFrame that illustrates the performance of each trained model on validation datasets specific to their linked instruments. Include metrics such as accuracy, precision, backtest results, and other relevant evaluation metrics.

- **Data Splitting for Profitable Model Identification: <\b>**

Divide the performance DataFrame into training, validation, and test datasets. Train the models on the training dataset to identify profitable symbol-model pairs.

- **Plotting Performance and Comparison:**

Plot the sum of performance traded with the chosen models and compare it with the sum of normal returns for chosen models instruments. Visualize the comparison to assess the effectiveness of the selected models in outperforming the market by plotting normal returns alongside model returns.

- **Real Market Order Placement in OANDA:**

The project yields a potentially profitable model paired with its corresponding instrument. This allows for the labeling of tomorrow's direction, enabling manual order placement or the use of the Oanda place order API function.

OANDA API Connection guide:

The script employs a demo account to test the project in a real-world setting. All instructions for utilizing the Oanda API are provided below:

**1):** Install required packages:

- --> !conda install ujson

- --> !pip install v20
- --> !pip install pyyaml
- --> !pip install --upgrade git+<https://github.com/yhilpisch/tpqoa.git>
- --> !pip install tpqoa

**2):** Import the tpqoa library from OANDA.

**3):** Fill in the "oanda.cfg" file as shown below:

- [oanda]
- account\_id = 101-011-27967879-001
- access\_token =  
366cbe0fda34665f63346c536ded7e67-78d4abd23dd495ed412a36ecef5a2de
- account\_type = practice

Note: The provided account ID and access token are for educational purposes from my demo account. For your own account, register at <https://www.oanda.com>, obtain your account ID and API access token through token generation.

**4):** Establish a connection to your OANDA account and utilize the OANDA API with the following code:

```
"api = tpqoa.tpqoa("oanda.cfg")"
```

This allows for connectivity to the OANDA account, enabling the use of the OANDA API.

**5):** For more information, please visit the link below:

<https://developer.oanda.com/rest-live-v20/introduction/>

## Results:

To obtain the project's results, first, the following instruments are employed:

- AAPL - Apple Inc.
- GOOGL - Alphabet Inc. (Google)
- AMZN - Amazon.com Inc.
- TSLA - Tesla, Inc.
- META - Meta Platforms, Inc. (formerly Facebook)
- CSCO - Cisco Systems, Inc.
- NVDA - NVIDIA Corporation
- NFLX - Netflix Inc.
- JPM - JPMorgan Chase & Co.
- IBM - International Business Machines Corporation
- BTC-USD - Bitcoin to US Dollar
- ETH-USD - Ethereum to US Dollar
- GC=F - Gold

Secondly, the following instruments are utilized in the implementation part:

As a result of the initial phase of training, a dataframe is generated that illustrates the effectiveness of models for each symbol which is 13 symbols, 11models ,both in baseline and hyper-parameter versions(286 rows):

	model	Valacc	ValNormalizedAcc	Valprecision	...	Testnnp	TestProfit %	ValProfit %	Label
AAPL N	(DecisionTreeRegressor(criterion='friedman_ms...	0.502520	0.476776	0.572954	...	13990.150509	67.855008	8.321195	0
AAPL N	LogisticRegression()	0.519078	0.492486	0.560150	...	13990.150509	55.346114	33.240932	0
AAPL N	DecisionTreeClassifier()	0.504680	0.478825	0.553140	...	13990.150509	69.003508	22.933613	0
AAPL N	(DecisionTreeClassifier(max_features='sqrt', r...	0.504680	0.478825	0.575342	...	13990.150509	85.175299	18.824472	0
AAPL N	KNeighborsClassifier()	0.501800	0.476093	0.528329	...	13990.150509	60.707518	26.885541	0
...	...	...	...	...	...	...	...	...	...
GC=F P	LinearSVC(C=1, max_iter=10000)	0.465032	0.440574	0.467219	...	10663.355380	98.470347	84.378250	0
GC=F P	SVC(C=1, max_iter=5000)	0.532084	0.504098	0.520000	...	10663.355380	85.613717	48.792148	0
GC=F P	MLPClassifier(hidden_layer_sizes=69, max_iter=...	0.488825	0.463115	0.471992	...	10663.355380	97.048925	53.192160	0
GC=F P	VotingClassifier(estimators=[('GBC',\n ...	0.518385	0.491120	0.487572	...	10663.355380	102.214038	69.970820	1
GC=F P	StackingClassifier(estimators=[('GBC',\n ...	0.533526	0.505464	0.531746	...	10663.355380	93.621316	48.751927	0

286 rows x 14 columns

In the second phase, the application of instruments and their associated models with high potential for profitability is determined based on the outputs from the training and validation datasets. Subsequently, the trained models will be employed to make predictions on the previously unused test dataset, revealing the final results:

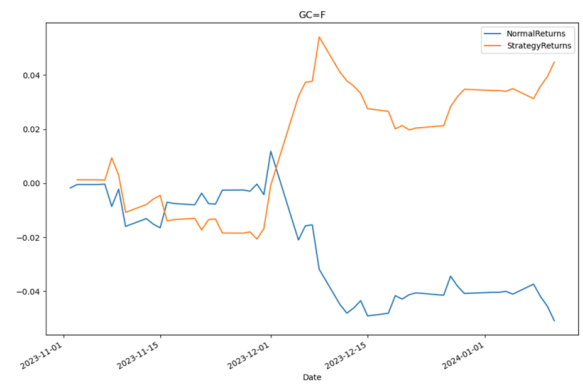
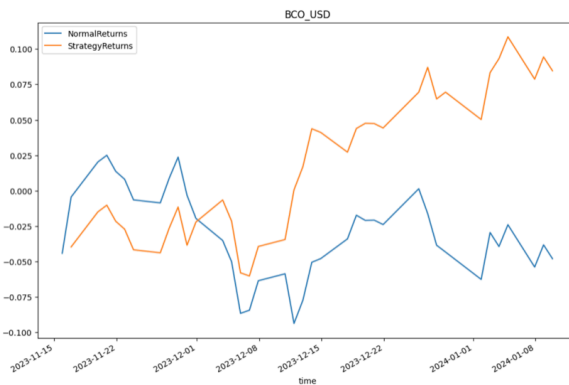
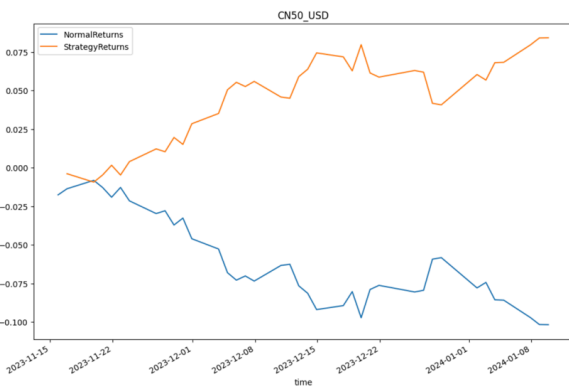
	Valacc	ValNormalizedAcc	Valprecision	Valrecall	...	model	Testnnp	Label	PLabel
<b>GC=F N</b>	0.464311	0.439891	0.461739	0.810687	...	MLPClassifier()	10663.355380	1	0
<b>GOOGL N</b>	0.521933	0.501429	0.524214	0.881429	...	(DecisionTreeRegressor(criterion='friedman_ms...	14507.866542	1	0
<b>GC=F N</b>	0.520548	0.493169	0.488372	0.320611	...	VotingClassifier(estimators=[('GBC', GradientB...	10663.355380	1	0
<b>IBM P</b>	0.505400	0.501429	0.500832	0.873730	...	VotingClassifier(estimators=[('GBC', \n ...	11644.014940	1	0
<b>NFLX N</b>	0.503960	0.500715	0.500546	0.663768	...	VotingClassifier(estimators=[('GBC', GradientB...	14188.421167	1	1
<b>CSCO P</b>	0.519798	0.494521	0.523666	0.954795	...	GaussianNB()	10537.477464	1	1
<b>META N</b>	0.458122	0.439173	0.476744	0.399027	...	MLPClassifier()	12666.830645	1	0
<b>BTC-USD P</b>	0.498375	0.484211	0.491731	0.995536	...	(DecisionTreeClassifier(max_depth=10, max_feat...	14594.376013	1	1
<b>GC=F N</b>	0.529200	0.501366	0.503333	0.230534	...	LogisticRegression()	10663.355380	1	0
<b>GC=F P</b>	0.471521	0.446721	0.471861	0.998473	...	GaussianNB()	10663.355380	1	1
<b>BTC-USD N</b>	0.485374	0.471579	0.485278	0.993304	...	LinearSVC()	14594.376013	1	0
<b>CSCO P</b>	0.503960	0.479452	0.531587	0.472603	...	(DecisionTreeRegressor(criterion='friedman_ms...	10537.477464	1	0
<b>BTC-USD N</b>	0.490791	0.476842	0.487723	0.975446	...	SVC()	14594.376013	1	1

13 rows x 15 columns

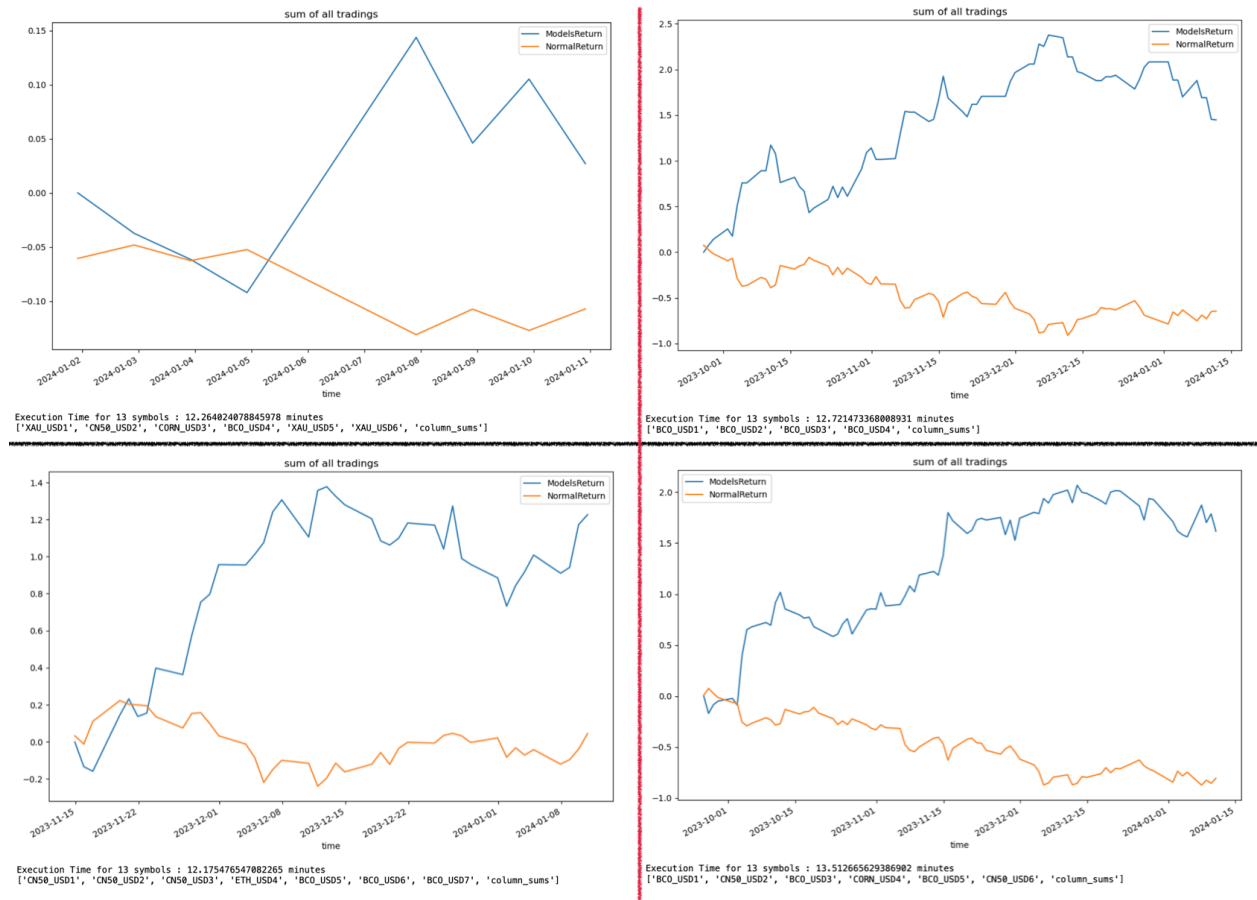
Among all the selected symbol-model pairs, those chosen by the best-model-finder are going to be tested on the test dataset. The best-model-finder employs two selection methods: one for choosing the best symbol-model, and the other for selecting lists of symbol-model pairs predicted to be profitable. This method is set manually.



In the following chart, four subcharts exist, each trained at different times. This adjustment is made by selecting the best symbol-model for each specific time period. The "StrategyReturns," depicted by the green line, represents the return of selected symbols enhanced by their chosen machine learning models. On the other hand, the "NormalReturns," illustrated by the blue line, signifies the return of selected symbols without employing any machine learning techniques.



In the chart below, there are four subcharts, each trained at different times with various instruments. This adjustment is made by selecting lists of symbol-models for each specific time period. The "ModelsReturn," depicted by the blue line, represents the return of selected symbols enhanced by their chosen machine learning models. On the other hand, the "NormalReturn," illustrated by the green line, signifies the return of selected symbols without employing any machine learning techniques.



## Conclusion:


The experimental results suggest that the proposed method could be efficient. However, to confirm its effectiveness, the project should be tested with real-world and real-time data provided by brokers. For now, the project proves itself by being tested on real-world data provided by brokers, but it should also be tested on a real-time system for comprehensive validation.

This project holds high potential for improvement, Because the project itself operates akin to a search engine, seeking better opportunities, it primarily requires a continuous feed of data to enhance its functionality. Enhancing the project involves enriching it with superior data sources such as news, oscillators, reliable model outputs, or their correlations. Nevertheless, even with these limited data, the models can identify promising opportunities, paving the way for future enhancements.

The future goals of this project primarily focus on enhancing the preprocessing section for both machine learning steps through the addition or removal of features. Some ideas for improvement include:

- **Implementing CNN Model for Pattern Recognition:** Utilizing a Convolutional Neural Network (CNN) model to recognize and identify bullish and bearish patterns.
- **Incorporating Financial News API with NLP:** Integrating a Financial News API and employing Natural Language Processing (NLP) techniques for labeling news data.
- **Utilizing RNN and Regression Models for Outputs:** Implementing Recurrent Neural Networks (RNN) and regression models for labeling their outputs.
- **Adding MACD Oscillator:** Introducing the MACD oscillator to the preprocessing stage.

Additionally, exploring avenues to enhance the overall aspects of the project, including execution speed, result accuracy, trading frequency, and code functionality.



## Reference:

udemy :

Algorithmic Trading A-Z with Python, Machine Learning & AWS:

<https://www.udemy.com/share/103Kga3@mu2CYr5xN2DcSkAxCRWM6loQhheGWja18Ap7eLGrhXeiun5juADFyawvMn1GJnt0/>

Algorithmic Trading In MQL5: Code Robots & Free Up Your Time:

[https://www.udemy.com/share/1069ka3@bSXklLjqBomW\\_-0DcVsSFpvgTOmF8Wnnat-DhRVrxeUzr9CNBXe7f-C6G4j7LsOz/](https://www.udemy.com/share/1069ka3@bSXklLjqBomW_-0DcVsSFpvgTOmF8Wnnat-DhRVrxeUzr9CNBXe7f-C6G4j7LsOz/)

Investment Analysis with Natural Language Processing NLP:

[https://www.udemy.com/share/103FvX3@WmWngsEJNrLJU81mRG1TR-8PHvF6vxYKiz\\_vVXTsGb5R6DuvUDSILCJp v2YofDy5/](https://www.udemy.com/share/103FvX3@WmWngsEJNrLJU81mRG1TR-8PHvF6vxYKiz_vVXTsGb5R6DuvUDSILCJp v2YofDy5/)

books :

John Murphy , 1999 ,Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications

Steve Nison , 2001 ,Japanese Candlestick Charting Techniques

researches :

RINGMU, H. S., & OUMAR, S. B. (2022). Forecasting stock prices in the New York stock exchange. Journal of Economics Bibliography, 9(1), 1-20.

Rahman, T., Akhter, R., Lawal, K., Mazumder, S. A., Afroz, T., & Rahman, A. (2021). Forecasting and Pattern Analysis of Dhaka Stock Market using LSTM and Phrophet Algorithm. Search in.

Liang, M., Wu, S., Wang, X., & Chen, Q. (2022). A stock time series forecasting approach incorporating candlestick patterns and sequence similarity. Expert Systems with Applications, 205, 117595.

Parente, M., Rizzuti, L., & Trerotola, M. (2024). A profitable trading algorithm for cryptocurrencies using a Neural Network model. Expert Systems with Applications, 238, 121806.

Maheronnaghsh, M. J., Gheidi, M. M., Younesi, A., & Fazli, M. A. (2023). Machine Learning Methods in Algorithmic Trading: An Experimental Evaluation of Supervised Learning Techniques for Stock Price.

youtube:

Recurrent Neural Networks | LSTM Price Movement Predictions For Trading Algorithms:  
[https://youtu.be/hpfQE0bTeA4?si=e4BzbiiS\\_0TEVgVI](https://youtu.be/hpfQE0bTeA4?si=e4BzbiiS_0TEVgVI)

Create, Package & Publish your OWN Python Library:  
[https://youtu.be/zhpl6Yhz9\\_4?si=EAoieIgBM88oxZpK](https://youtu.be/zhpl6Yhz9_4?si=EAoieIgBM88oxZpK)

websites:

<https://www.investopedia.com/articles/active-trading/092315/5-most-powerful-candlestick-patterns.asp>

<https://blog.dhan.co/how-to-read-candlestick-charts-for-day-trading/>

<https://www.dynotrading.com/10-best-macd-settings-for-effective-trading/>

<https://developer.oanda.com/rest-live-v20/introduction/>

