Journal of Business and Management Studies
ISSN: 2709-0876
DOI: 10.32996/jbms
Journal Homepage: www.al-kindipublisher.com/index.php/jbms

JBMS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Algorithmic Trading Strategies: Leveraging Machine Learning Models for Enhanced Performance in the US Stock Market

Nisha Gurung[1], Md Rokibul hasan[2] ✉ Md Sumon Gazi[3] and Md zahidul Islam[4]
[1234]MBA Business Analytics, Gannon University, USA
Corresponding Author: MD Rokibul Hasan, **E-mail**: prorokibulhasanbi@gmail.com

| ABSTRACT

In the recent past, algorithmic trading has become exponentially predominant in the American stock market. The principal objective of this research was to explore the employment of machine learning frameworks in formulating algorithmic trading strategies tailored for the US stock market. For this investigation, an array of software tools was employed, comprising the Pandas library for data manipulation and analysis, the Python programming language, the Scikit-learn library for machine learning algorithms and analysis metrics, and the LIME library for explainable AI. In this study, the researcher gathered an extensive dataset from the Amazon Stock Exchange, spanning from October 19, 2018, to October 16, 2022. The dataset comprised a wide range of parameters related to Amazon's stock data, facilitating a rigorous analysis of its market performance. Five models were subjected to the experiment, notably Ridge Regression, Ada-Boost, Light-GBM, XG-Boost, Linear Regression, and Cat-Boost. From the experiment result, it was evident that the XG-Boost attained the highest R-squared (99.24%) and accuracy (99.23%) among all the algorithms. From the above results, the analyst inferred that the XG-Boost was able to learn a more complex and accurate model of the stock exchange data compared to the other algorithms. XG-Boost algorithm can be utilized to back-test distinct trading strategies on historical data, enabling investors to evaluate their efficiency before risking real capital. By assessing a wide array of factors, the XG-Boost algorithm can assist investors in selecting stocks with a higher probability of outperforming the market.

| KEYWORDS

Algorithmic Strategies; Stock Market; Machine Learning; Python; Ridge Regression; Ada-Boost; Linear Regression; Cat-Boost; Light GBM.

| ARTICLE INFORMATION

ACCEPTED: 01 April 2024 **PUBLISHED:** 20 April 2024 **DOI:** 10.32996/jbms.2024.6.2.13

1. Introduction

As per IJSRCSEIT (2023), the invention of algorithmic trading has transformed the domain of financial markets, facilitating traders to perform large volumes of transactions with exceptional efficiency and speed. Within this spectrum, machine learning algorithms have proven to be instrumental tools for devising trading strategies that can optimize market inefficiencies and generate alpha. By utilizing a large volume of historical data and innovative computational methods, machine learning provides the possibility to uncover sophisticated relationships and patterns that traditional techniques may overlook. The prime focus of this research is to explore the employment of machine learning frameworks in formulating algorithmic trading strategies tailored for the US stock market.

IRJET Journal (2022) indicates that in the recent past, algorithmic trading has become exponentially predominant in the American stock market. Also deemed as black-box trading or automated trading, algorithmic trading entails utilizing computers to make trades premised on machine learning models or predefined quantitative rules. The objectives of algorithmic trading are to utilize advanced data analytics and computing power to pinpoint beneficial trading opportunities and trends that may be too complicated

or rapid for human traders to acknowledge and act upon. While algorithmic trading has prevailed for a few decades now, recent inventions in machine learning and the large volume of financial data available have inspired new opportunities to establish even more complicated automated trading strategies.

2. Literature Review

The scholarly literature comprehensively covers a myriad of stock trading techniques. Nevertheless, the domain of algorithmic trading is quite a recent development, and as a result, the employment of machine learning in algorithmic trading has emerged as a prime focus in modern academic research. Previous research has extensively examined the use of machine learning in financial markets, with a particular focus on algorithmic trading strategies. Various studies have demonstrated the effectiveness of machine learning models in forecasting stock prices, identifying trading signals, and optimizing portfolio allocations. For instance, Salim (2021) employed deep learning techniques to predict stock returns, achieving promising results compared to traditional methods. Similarly, Ghania et al. (2019) utilized reinforcement learning algorithms to tailor adaptive trading strategies that outperformed conventional methods. These findings emphasized the capability of machine learning in terms of consolidating the capabilities of algorithmic trading systems and generating alpha in highly competitive markets.

Nabipour et al. (2020) investigated stock market forecasting utilizing regression methods and recommended a promising regression technique for predicting stock market prices based on market data. The authors indicated that future enhancements in the multiple regression technique could be attained by integrating a greater number of factors. The prime goal of their research was to help stock traders and investors make strategic decisions when investing in the stock market. Provided the sophisticated and dynamic aspect of the stock market, accurate forecasting plays a significant role in this intricate and challenging process.

Research undertaken by Rashid (2022) utilized the 'Random Forest' algorithm to design a predictive algorithm for predicting the 5-day-ahead and 10-day-ahead arrangements of the CROBEX index and chosen stocks. The outcomes of their study illustrate the successful utilization of random forests in designing predictive models for the expected trends in the stock market.

On the other hand, Mohammad Awais (2023) performed research intending to develop and contrast three algorithms for forecasting the trend of movement in the daily Tehran Stock Exchange (TSE) index. The frameworks were grounded on three classification algorithms, notably Random Forest, Decision Tree, and Naïve Bayesian Classifier. The investigators inferred that technical analysis played a more imperative role than fundamental analysis in the decision-making process of stakeholders and brokers.

In 2022, Reddy & Sai designed a forecasting algorithm premised on the Backpropagation Neural Network (BPNN) and the K-Nearest Neighbors (KNN) algorithms. The algorithm was employed to forecast the stock prices of Chinese stocks, and the test outcomes illustrated that the average errors noted in the KNN-ANN models were smaller compared to those in the KNN model. Their finding suggested that the forecasting algorithm based on the KNN-ANN models outperforms the KNN algorithm in stock forecasting.

3. Methodology

For this investigation, an array of software tools was employed, comprising the Pandas library for data manipulation and analysis, the Python programming language, the Scikit-learn library for machine learning algorithms and analysis metrics, and the LIME library for explainable AI. Python was selected because of its versatility, simplicity, a rich variety of machine learning libraries, and comprehensive capabilities for data analysis (proAIrokibul, 2024). By contrast, LIME was integrated to elevate the interpretability of machine learning models, facilitating the analyst's gain of insight into the process of prediction generation.

3.1 Dataset

In this study, the researcher gathered an extensive dataset from the Amazon Stock Exchange, spanning from October 19, 2018, to October 16, 2022. The dataset comprised a wide range of parameters related to Amazon's stock data, facilitating a rigorous analysis of its market performance (proAIrokibul, 2024). Amazon stock exchange comprised the following parameters:

1. **Date**: The particular date for the stock trading, facilitating chronological analysis.
2. **Open Price:** Refers to the opening price of Amazon's stock at the start of a trading session.
3. **Close Price:** Denotes the closing price of Amazon's stock at the climax of a trading session.
4. **High Price**: Refer s to the highest price attained by Amazon's stock during a specific trading session.
5. **Low Price:** The lowest price set by Amazon's stock exchange during a particular trading session.
6. **Volume:** The overall number of shares traded during a particular period, presenting insights into investor interest and market liquidity.

7. **Adjusted Close Price**: Denotes the closing price monitored for factors such as stock splits, dividends, and other corporate activities, providing a more accurate representation of stock performance.

3.2 Pre-Processing

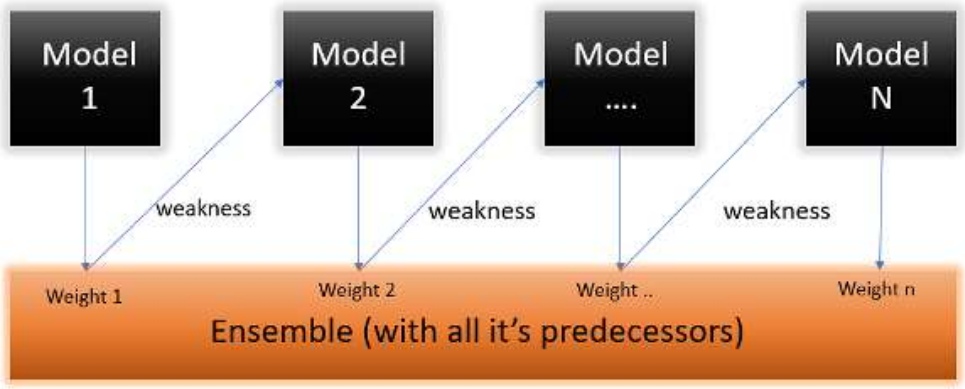
Data preprocessing entailed applying different techniques to cleanse the collected data. In this research, the analyst employed the Min Max-Scaler technique, which is imported from the sci-kit-learn library. The Min Max-Scaler was responsible for converting attributes by scaling each one to a particular range (proAIrokibul, 2024). This approximator operated by independently scaling and changing each attribute to guarantee it falls within the targeted range, typically between zero and one, based on the training set.

3.3 Feature Engineering Selection

As per Pro-AI-Rokibul. (2024), feature selection is a pivotal step in designing a predictive algorithm since it comprises choosing the most pertinent attributes that have the highest effect on the stock price label. This process is targeted to pinpoint the optimal subset of features that will assist in generating a relatively accurate stock price prediction. By carefully choosing the right set of attributes, the researchers affirmed that the algorithm captured the significant relationships and patterns in the data. After the feature selection stage was completed, the subsequent phase comprised fitting the algorithm utilizing the selected attributes.

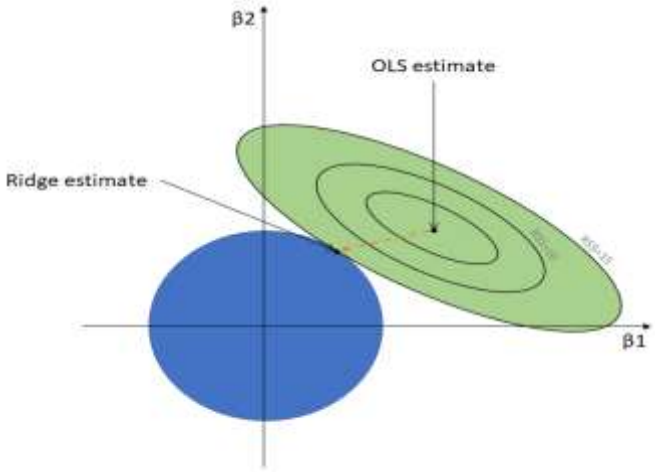
3.4 Models and Metrics

Ada-boost Model



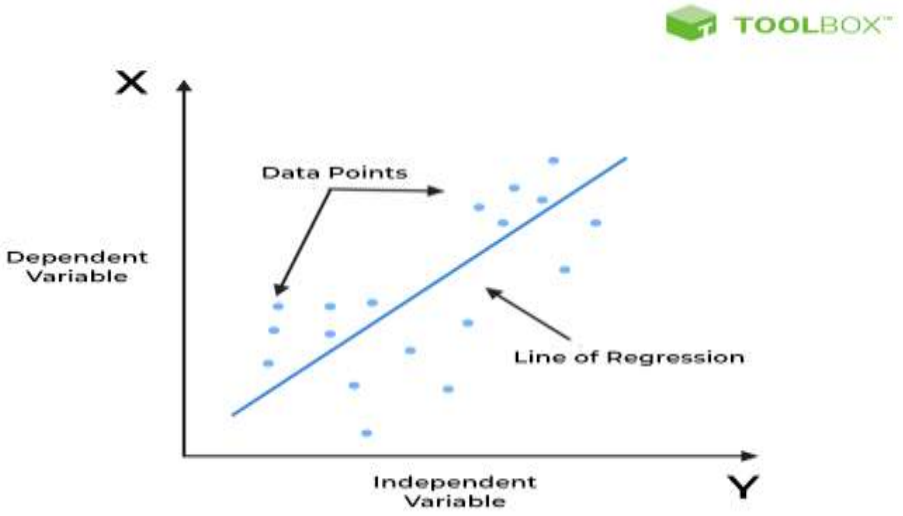
The AdaBoost model is a method that transforms weak learners into solid ones by employing a specialized boosting protocol termed an ensemble model. Ada-Boosting targets to reinforce the accuracy of less-willed learners by sequentially reconsidering their past forecasting (Saifan, 2020). This meta-predictor framework fits an algorithm to the principal dataset and then utilizes that algorithm to fit supplementary copies of itself to the dataset. By modifying sample weights based on actual prediction error, the training process facilitates the model to concentrate on the most challenging data points.

Ridge Regression Model



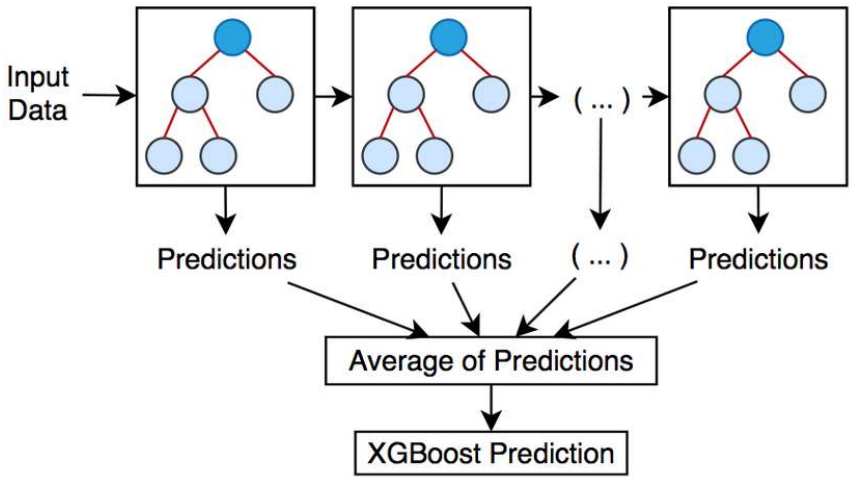
Ridge Regression is a model of approximating the coefficients of multiple regression frameworks in incidents where the independent variables are greatly correlated. Ridge regression is specifically instrumental for reducing the challenge of multicollinearity in linear regression, which mostly happens in algorithms with large quantities of parameters. The technique offers enhanced efficiency in parameter approximation challenges in exchange for a tolerable volume of bias (Sanyal, 2022). The ridge approximator is the resolution to the least square challenges that are vulnerable to the restraint that the sum of the squares of the coefficients is less than a constant. The ridge parameter, which regulates the intensity of the penalty term, is normally selected as the peer of the heuristic criterion.

Linear Regression Model



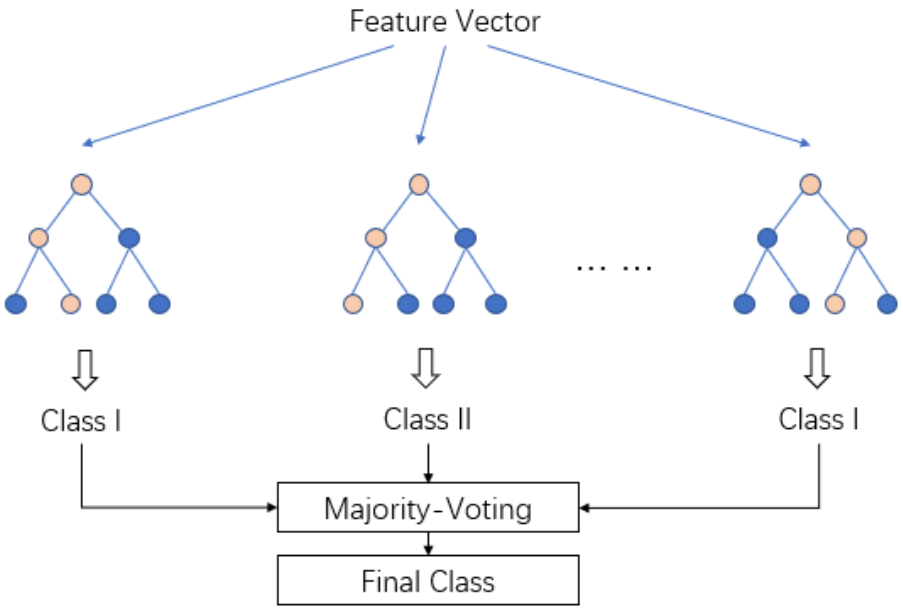
Linear regression is a statistical technique utilized in machine learning and data science for predictive analysis. It is a model that offers a linear association between an independent variable (explanatory and predictor variable) and a dependent variable (outcome or response variable) that remains static because of the alteration in other variables (Umer, 2019). The regression algorithms predict the value of the dependent variable, which is the outcome or response variable being analyzed or studied.

XG-Boost Model



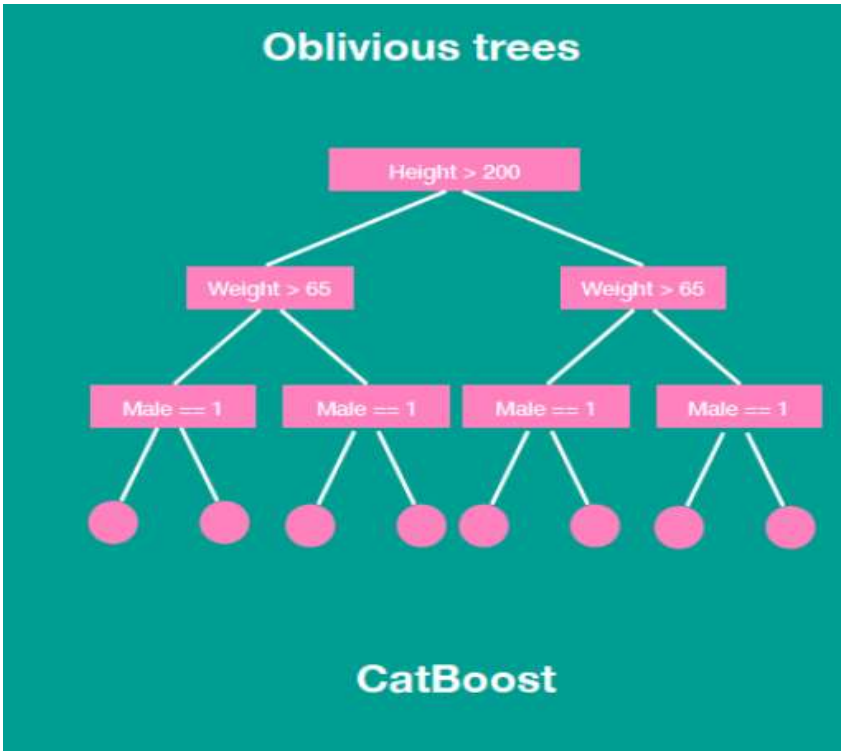
XG-Boost is a comprehensive and efficient model of the gradient boosting model for regression forecasting modeling. It is an open-source library that offers an effective and efficient deployment of the gradient boosting algorithm, which is a form of ensemble machine learning framework that can be used for classification or regression predictive modeling (Sanyal, 2022).

Light-GBM Model



Light-GBM is a distributed and high-performance gradient-boosting model utilized for distinct machine-learning tasks. It is renowned for its efficiency, speed, and capability to manage large datasets efficiently (Saifan, 2020). Light-GBM employs histogram-based learning to escalate the training procedure while ensuring accuracy, making it a popular selection among data analysts worldwide for tasks like regression, classification, and ranking.

Cat-boost Model



Cat-Boost is a renowned machine-learning model that can handle both numerical and categorical features efficiently, mitigate missing values, and combat overfitting through various methods (Salim, 2021). Its GPU-powered version and dynamic classifier class make it a prominent choice for regression and classification tasks with large datasets.

3.5 Experimentation

Importing Libraries

In [1]:
```
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
```

In [2]:
```
df = pd.read_csv('FinancialDataCSV - FinancialData (2).csv')
df
```

Output

Out[2]:

	Stock ID	8:00:00	8:01:00	8:02:00	8:03:00	8:04:00	8:05:00	8:06:00	8:07:00	8:08:00	...	1M - <3M Vol	3M - <5M Vol	5M - <10M Vol	10M - 20M Vol	>2
0	0	1.483636	1.501818	1.483091	1.490364	1.487273	1.505455	1.505455	1.505673	1.530909	...	1	0	0	0	
1	1	1.300000	1.316000	1.312000	1.304000	1.296000	1.296000	1.304000	1.304000	1.304000	...	1	0	0	0	
2	2	1.016249	1.016249	1.016249	1.016249	1.016249	1.016249	1.016249	1.016249	1.016249	...	0	0	0	0	
3	3	1.168163	1.026939	1.144490	1.167347	1.167347	1.142857	1.167347	1.182857	1.182857	...	0	0	1	0	
4	4	1.834043	1.825532	1.814894	1.812766	1.808511	1.810638	1.840426	1.834043	1.793617	...	0	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2543	2543	1.102724	1.101946	1.050584	1.102724	1.070039	1.105058	1.104280	1.105058	1.128405	...	0	0	1	0	
2544	2544	1.036496	1.080292	1.026764	1.026764	1.026764	1.000000	1.000000	1.216545	1.618005	...	0	0	0	1	
2545	2545	1.228387	1.224516	1.227097	1.201290	1.200000	1.223226	1.219355	1.183226	1.190968	...	0	1	0	0	
2546	2546	1.312217	1.312217	1.248869	1.226244	1.206637	1.220211	1.259427	1.279035	1.306184	...	0	0	0	0	
2547	2547	3.981711	3.628319	3.746313	4.041298	4.070796	4.719764	4.483776	4.159292	4.277286	...	0	0	1	0	

Concerning the data frame, every row in the command was modified to portray the distribution of the price of the stock for a specific month in the stock exchange. Particularly, for every month, the row displayed the range price of every stock from the lowest as per the Amazon Stock Exchange. By organizing the data in that format, the analyst obtained insights concerning the performance of stock price distributions within each month.

In [3]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2548 entries, 0 to 2547
Columns: 115 entries, Stock ID to ActualOutcome
dtypes: float64(90), int64(25)
memory usage: 2.2 MB
None
```

In [4]:
```
from scipy import stats

# Identify numerical columns
numerical_cols = df.select_dtypes(include=['int', 'float']).columns

# Calculate Z-scores for each data point in the numerical columns
z_scores = stats.zscore(df[numerical_cols])

# Define a threshold for outlier detection (e.g., 3 standard deviations)
threshold = 3

# Check if any Z-score exceeds the threshold in any numerical column
outliers_exist = (z_scores > threshold).any().any()

if outliers_exist:
    print("Outliers exist in the dataset.")
else:
    print("No outliers found in the dataset.")
```

```
Outliers exist in the dataset.
```

In [5]:
```
# Identify numerical and categorical columns
numerical_cols = df.select_dtypes(include=['int']).columns
categorical_cols = df.select_dtypes(include=['object']).columns

# Outlier detection for numerical columns using Z-score
z_scores = stats.zscore(df[numerical_cols])
outliers_numerical = (z_scores > 3) | (z_scores < -3)
outliers_numerical = pd.DataFrame(outliers_numerical, columns=numerical_cols)

# Outlier detection for categorical columns based on frequency
outliers_categorical = pd.DataFrame()
for col in categorical_cols:
    counts = df[col].value_counts()
    outliers_categorical[col] = df[col].isin(counts[counts < 10].index)

# Combine outliers from numerical and categorical columns
outliers_combined = pd.concat([outliers_numerical, outliers_categorical], axis=1).any(axis=1)

# Remove outliers from the dataset
df = df[~outliers_combined]

# Display or save the cleaned dataset
print(df)
```

Output:

	Stock ID	8:00:00	8:01:00	8:02:00	8:03:00	8:04:00	8:05:00	\
1	1	1.300000	1.316000	1.312000	1.304000	1.296000	1.296000	
3	3	1.168163	1.026939	1.144490	1.167347	1.167347	1.142857	
7	7	1.112903	1.112903	1.112903	1.112903	1.112903	1.112903	
10	10	1.101562	1.109375	1.109375	1.234375	1.210938	1.273438	
11	11	1.000000	0.976331	0.976331	0.977988	1.005917	1.017751	
...	...	...	...	...	...	...	...	...
2541	2541	1.548333	1.466667	1.479583	1.444583	1.444583	1.368333	
2542	2542	1.436127	1.436127	1.522376	1.501221	1.505289	1.505289	
2543	2543	1.102724	1.101946	1.050584	1.102724	1.070039	1.105058	
2544	2544	1.036496	1.080292	1.026764	1.026764	1.026764	1.000000	
2547	2547	3.981711	3.628319	3.746313	4.041298	4.070796	4.719764	

	8:06:00	8:07:00	8:08:00	...	1M - <3M Vol	3M - <5M Vol	\
1	1.304000	1.304000	1.304000	...	1	0	
3	1.167347	1.182857	1.182857	...	0	0	
7	1.112903	1.112903	1.112903	...	0	0	
10	1.250000	1.226562	1.289062	...	0	0	
11	1.017751	1.017751	1.017751	...	1	0	
...	...	...	...	...	...	...	
2541	1.397500	1.391250	1.425000	...	0	0	
2542	1.480815	1.472742	1.432859	...	0	0	
2543	1.104280	1.105058	1.128405	...	0	0	
2544	1.000000	1.216545	1.618005	...	0	0	
2547	4.483776	4.159292	4.277286	...	0	0	

	5M - <10M Vol	10M - 20M Vol	>20M Vol	IsTuesday	IsWednesday	\
1	0	0	0	0	0	
3	1	0	0	0	0	
7	1	0	0	0	0	
10	1	0	0	0	0	
11	0	0	0	1	0	
...	...	...	...	...	...	
2541	0	1	0	0	0	
2542	0	1	0	0	0	
2543	1	0	0	0	1	
2544	0	1	0	0	1	
2547	1	0	0	0	0	

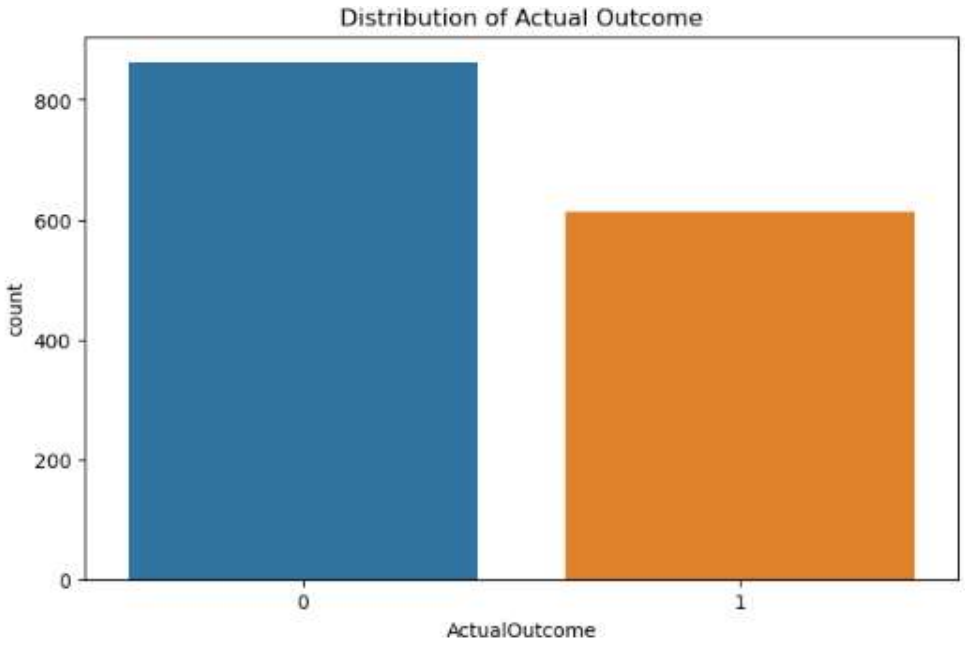
	IsThuraday	Isfriday	ActualOutcome
1	0	1	0
3	1	0	0
7	1	0	1
10	0	1	0
11	0	0	1
...	...	...	...
2541	0	1	0
2542	0	1	1
2543	0	0	1
2544	0	0	0
2547	0	0	1

[1474 rows x 115 columns]

To visualize the distribution of the targeted variables, a code snippet was applied to generate a histogram chart of the targeted variables:

In [7]:
```
# Visualize the distribution of the target variable
plt.figure(figsize=(8, 5))
sns.countplot(x='ActualOutcome', data=df)
plt.title('Distribution of Actual Outcome')
plt.show()
```

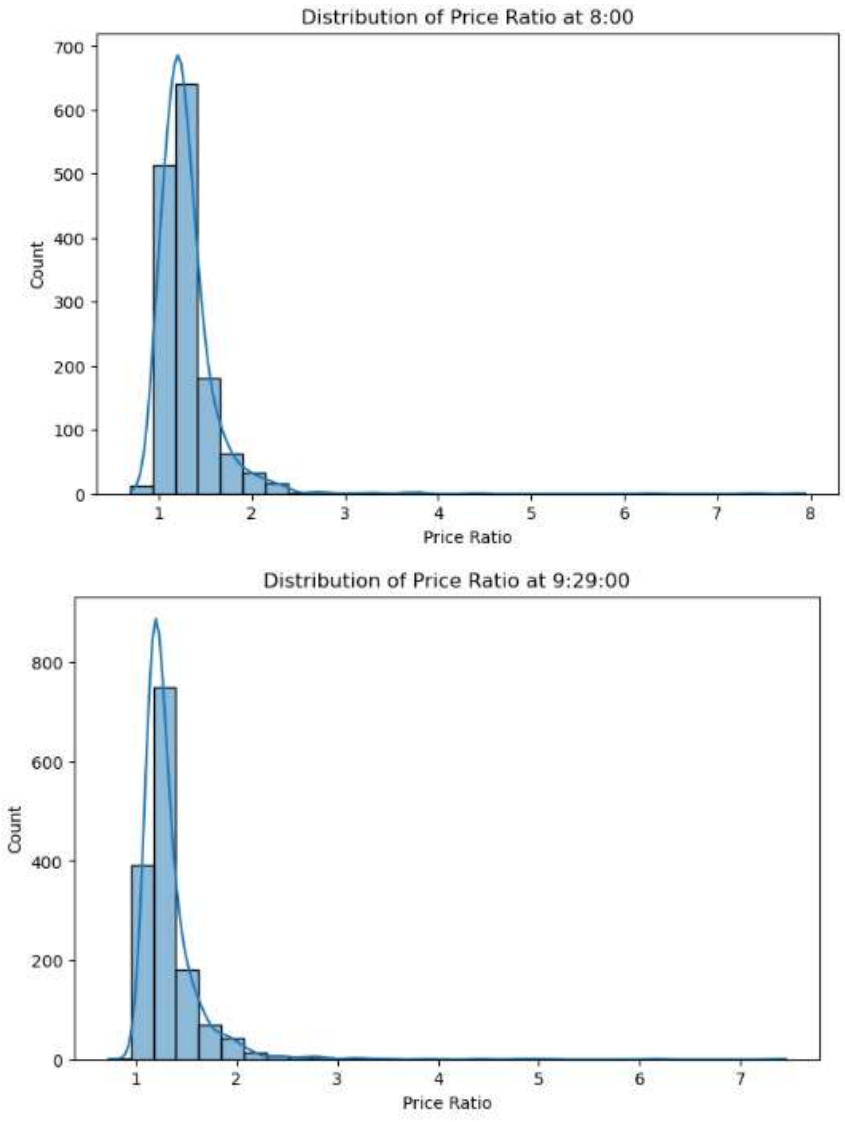
Output:



Apart from the distribution of the targeted variable, the analyst equally tried to generate the price ratio distribution as showcased below:

```
In [9]: # Visualize the distribution of a specific feature
        plt.figure(figsize=(8, 5))
        sns.histplot(df['8:01:00'], bins=30, kde=True)
        plt.title('Distribution of Price Ratio at 8:00')
        plt.xlabel('Price Ratio')
        plt.show()
```

Output:



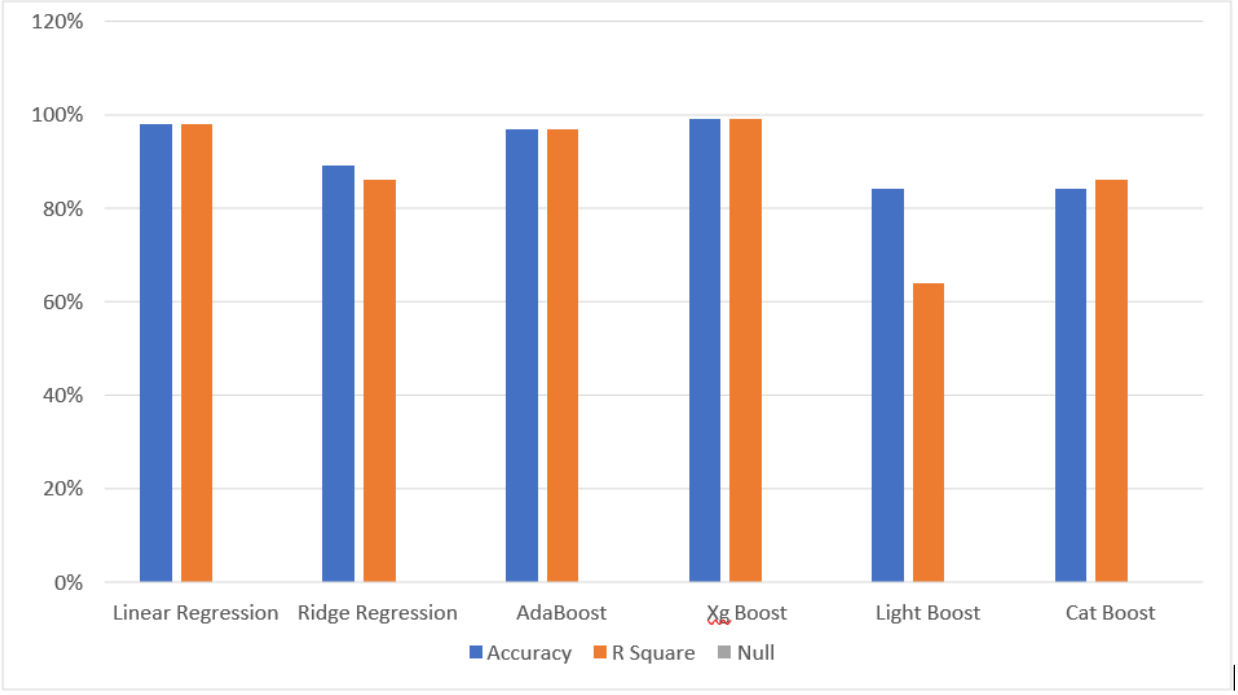
3.6 Performance Metrics

R-squared (R^2) was utilized in this research, which revolves around a statistical calculation that determines the measure of the variance in the dependent variable (Spending Score) that can be conveyed by the independent variables in the regression model. R-squared ranges from 0 to 1, with higher values indicating a better fit of the framework to the data. R-squared can expressed as follows:

$$R^2 = 1 - (\Sigma (y_i - y\_hat\_i)^2) / (\Sigma (y_i - y\_bar)^2)$$

Results/Outcomes of the Machine Learning Algorithms

Algorithm	R-squared	Accuracy
Ridge Regression	86.14%	89.23%
Linear Regression	98.02%	98.04
Ada-Boost	97.11%	95.23%
XG-Boost	99.24%	99.23%
Light Boost	64%	85%
Cat Boost	83%	88%



3.7 Interpretation

From the above table and chart, it was evident that the XG-Boost attained the highest R-squared (99.24%) and accuracy (99.23%) among all the algorithms. From the above results, the analyst inferred that the XG-Boost was able to learn a more complex and accurate model of the stock exchange data compared to the other algorithms. Besides, the linear regression came second, where it achieved R-squared (98.02%) and accuracy (98.04%). Overall, the results indicated that XG-Boost was the best-performing algorithm on this particular dataset.

4. Business Impact

4.1 Benefits for the Stock Investors

1. **Enhanced Stock Price Signal Identification:** XG-Boost's accuracy in pinpointing sophisticated patterns within large-volume data can help discover subtle signals in financial indicators, historical stock price data, and news sentiment. Consequently, this can assist investors in pinpointing prospective profitable opportunities that might be missed by simpler analysis methods.
2. **Enhanced Stock Selection**: By assessing a wide array of factors, the XG-Boost algorithm can assist investors in selecting stocks with a higher probability of outperforming the market. This can entail considering mainstream financial ratios, organizational news, social media sentiment, and other data points to develop a more robust picture of a stock's potential.
3. **Back-testing and Scenario Planning**: The XG-Boost algorithm can be utilized to back-test distinct trading strategies on historical data, enabling investors to evaluate their efficiency before risking real capital. Furthermore, the algorithm can be utilized for scenario planning, simulating possible market movements under various economic conditions.

4.2 Benefits for the USA Economy

1. **Enhanced Market Efficiency**: By facilitating investment firms, hedge funds, and government investors with better risk assessment and stock selection through XG-Boost algorithms, the overall market efficiency can be streamlined. Government

investors can be more informed on investment decisions, which, in turn, can result in a more accurate allocation of capital for valuable organizations.

2. **Diminished Market Volatility**: XG-Boost's capability to pinpoint hidden associations and forecast future patterns can assist investors in expecting and responding more strategically to market fluctuations. Consequently, this could lead to enhanced overall market movements and possibly reduce the effect of flash crashes or irrational exuberance.
3. **Increased Investor Confidence**: As XG-Boost enlightens investors with supreme decision-making mechanisms, foreign investors might be empowered to invest in the market. This could lead to more long-term investments and potentially fuel US economic growth.

4.3 How to Use the Model

1. **Step 1- Data Collection and Preprocessing:** Prospective investors should begin by first collecting historical stock data. This could comprise price data, economic indicators, company financials, trading volume, as well as social media sentiment. Subsequently, they should preprocess the data by cleaning it and formatting it for XG-Boost. This might comprise countering missing values, transforming categorical data, and potentially scaling numerical features.
2. **Step 2-Feature Engineering and Selection:** The investor should strive to pinpoint relevant features; they should transcend beyond just price data. Be keen for elements that might impact stock prices, such as organizational news, analyst ratings, sector performance, and economic data.
3. **Step 3-Algorithm Building and Training**: The analyst should select a Python library. Subsequently, the analyst should define the algorithm parameters: XG-Boost has an array of hyperparameters that can substantially influence performance. Afterwards, the analyst should train the algorithm by Splitting the data into training and testing sets.
4. **Step 4- Back-testing and Refinement:** The trader should use the XG-Boost algorithm to simulate trades on historical data and evaluate its efficiency. Based on back-testing outcomes, the trader might need to modify the features, hyperparameters, or even the model architecture itself.
5. **Step 5-Trading Implementation**: The trader should consolidate the algorithm with the trading forum. In particular, they can link their algorithm to the stock exchange trading platform to automate trade execution based on its signals.
6. **Step 6- Model Evaluation:** After implementing the algorithm, investors should consistently monitor and evaluate the algorithm's performance in real-world scenarios. Collect feedback from shareholders and users to pinpoint aspects of improvement and refine the model accordingly.

5. Conclusion

The prime focus of this research was to explore the employment of machine learning frameworks in formulating algorithmic trading strategies tailored for the US stock market. For this investigation, an array of software tools was employed, comprising the Pandas library for data manipulation and analysis, the Python programming language, the Scikit-learn library for machine learning algorithms and analysis metrics, and the LIME library for explainable AI. In this study, the researcher gathered an extensive dataset from the Amazon Stock Exchange, spanning from October 19, 2018, to October 16, 2022. The dataset comprised a wide range of parameters related to Amazon's stock data, facilitating a rigorous analysis of its market performance. Five models were subjected to the experiment, notably Ridge Regression, Ada-Boost, XG-Boost, Linear Regression, and Cat-Boost. From the experiment result, it was evident that the XG-Boost attained the highest R-squared (99.24%) and accuracy (99.23%) among all the algorithms. From the above results, the analyst inferred that the XG-Boost was able to learn a more complex and accurate model of the stock exchange data compared to the other algorithms. XG-Boost algorithm can be utilized to back-test distinct trading strategies on historical data, enabling investors to evaluate their efficiency before risking real capital. By assessing a wide array of factors, the XG-Boost algorithm can assist investors in selecting stocks with a higher probability of outperforming the market.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Awais, M. (2019). Stock market prediction using Machine Learning (ML)Algorithms. *www.academia.edu*. https://www.academia.edu/84023752/Stock_Market_Prediction_Using_Machine_Learning_ML_Algorithms?sm=b

[2] Ghania, M. U., Awaisa, M., & Muzammula, M. (2019). Stock market prediction using machine learning (ML) algorithms. ADCAIJ: Adv Distrib Comput Artif Intell, 8(4), 97-116.

[3] IJSRCSEIT. (2022). Stock prediction using machine learning algorithms. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*. *Technoscienceacademy*. https://www.academia.edu/91042041/Stock_Prediction_Using_Machine_Learning_Algorithms?sm=b

[4] Nabipour, M., Nayyeri, P., Jabani, H., Shahab, S., & Mosavi, A. (2020). Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. *Ieee Access*, *8*, 150199-150212.

[5] Rashid, M., (2022). Predicting stock market using machine learning algorithms. *International Journal of Scientific Research in Science, Engineering and Technology*. *Technoscienceacademy*. https://www.academia.edu/91050428/Predicting_Stock_Market_Using_Machine_Learning_Algorithms?sm=b
[6] Reddy, V. K. S., & Sai, K. (2018). Stock market prediction using machine learning. *International Research Journal of Engineering and Technology (IRJET)*, *5*(10), 1033-1035.
[7] Salim, S., (2021). PREDICTING STOCK MARKET USING MACHINE LEARNING ALGORITHMS. *www.academia.edu*. https://www.academia.edu/44896464/PREDICTING_STOCK_MARKET_USING_MACHINE_LEARNING_ALGORITHMS?sm=b
[8] IRJET Journal (2019). Prediction of Stock Market using Machine Learning Algorithms. *Irjet*. https://www.academia.edu/40043469/IRJET_Prediction_of_Stock_Market_using_Machine_Learning_Algorithms?sm=b
[9] IRJET Journal, (2022). STOCK MARKET PREDICTION AND ANALYSIS USING MACHINE LEARNING ALGORITHMS. *Irjet*. https://www.academia.edu/86783381/STOCK_MARKET_PREDICTION_AND_ANALYSIS_USING_MACHINE_LEARNING_ALGORITHMS?sm=b
[10] proAIrokibul. (n.d.). *Stock-Price-Analysis-And-Prediction/Model/main.ipynb at main · proAIrokibul/Stock-Price-Analysis-And-Prediction*. GitHub. https://github.com/proAIrokibul/Stock-Price-Analysis-And-Prediction/blob/main/Model/main.ipynb
[11] Saifan, R. (2020). Investigating Algorithmic Stock Market Trading using Ensemble Machine Learning Methods. *www.academia.edu*. https://www.academia.edu/81513651/Investigating_Algorithmic_Stock_Market_Trading_using_Ensemble_Machine_Learning_Methods?sm=b
[12] Sanyal, S. (2022). Stock Market Prediction using Machine Learning Algorithms. *www.academia.edu*. https://www.academia.edu/68861133/Stock_Market_Prediction_using_Machine_Learning_Algorithms?sm=b
[13] Umer, M. (2019). Stock market prediction using Machine Learning(ML)Algorithms. *www.academia.edu*. https://www.academia.edu/77318461/Stock_Market_Prediction_Using_Machine_Learning_ML_Algorithms?sm=b