WESTMINSTER BUSINESS SCHOOL
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING IN FINANCE SERVICES

ASSESSMENT 1

An Analysis of Stock Price Prediction and Investment Strategy for Amazon.com, Inc

2168 words

MSc FinTech and Business Analytics
Artificial
Module Code: FNCE043W
Module Leader: Dr Yumei Yao
Deadline: 16th November 2023

CONTENT

I. Abstract 3
1. INTRODUCTION 3
2. DATA COLLECTION AND PREPROCESSING 3
a. Amazon.com,Inc Historical Data 3
b. Feature Engineering 3
3. EXPLORATORY DATA ANALYSIS OF FEATURES 4
4. MACHINE LEARNING CLASSIFICATION 8
a. Logistic Regression 8
b. Extra Trees 8
5. CROSS VALIDATION ACCURACY ANALYSIS 8
a. Cross Validation for Logistic Regression 9
b. Cross Validation for Extra Trees 9
6. EVALUATION OF CLASSIFIERS PERFORMANCE/RESULTS 9
a. Classification Report 10
b. Confusion Matrix 10
c. ROC Curve 11
8. MARKET AND STRATEGY RETURNS 11
9. CUMULATIVE RETURNS ANALYSIS 12
10. INTERPRETATION AND DISCUSSION 12
11. CONCLUSION 13
II. References 14
III. Appendices: 15

I. Abstract

This analysis spanning 20% of January 2013 to December 2022 data predicts Amazon's stock movements using the Extra Trees Classifier, achieving 50% accuracy—comparable to random guessing, signalling limited reliability. The observed discrepancy in cumulative market and strategy returns underscores challenges in precise forecasting and debates about challenges machine learning may face in unpredictable financial markets.

.

1. INTRODUCTION

Amazon.com, Inc., is a renowned multinational e-commerce and technology company, founded by Jeff Bezos in 1994. The company started as an online bookshop and increased the range of products and services it provided (Bezos, 2022). Key milestones include launching Kindle, Amazon Prime, AWS, and acquiring Whole Foods in 2017, expanding into supermarkets and physical retail with Amazon Go stores (Bezos, 2022). Economically, Amazon revolutionized retail, altering traditional establishments, consumer behavior, expectations, and industry competition, and integrating robots, AI, and machine learning (Bezos, 2022).

Amazon's strategy diversification to deliver personalized customer experience while simultaneously expanding its market reach and revenue streams by leveraging its extensive data resources, diverse product and service offerings, and technological capabilities have given Amazon the title of one of the most influential and successful companies in the world (Bezos, 2022). The company's stock performance is monitored worldwide, and it is part of major stock market indices, reflecting its influence on the broader market.

2. DATA COLLECTION AND PREPROCESSING

a. Amazon.com,Inc Historical Data

Collection of Amazon.com, Inc daily stock data from Jan 2013 to Dec 2022 was executed from Yahoo Finance, totalintg 2518 rows. Data includes include open, high, low, close prices, adjusted close, and stock volume. Rows with missing or incomplete data were removed.

b. Feature Engineering

Generated features from the processed data include:

i. Daily high-low **('H-L')** and open-close**('O-C')** difference, which capture intraday price fluctuations (Malkiel,1973):

- $H - L = \text{High Price} - \text{Low Price}$
- $O - C = \text{Close Price} - \text{Open Price}$

ii. 3-day, 10-day, and 30-day moving averages ('3d MA', '10d MA' and '10d MA'), revealing short and medium-term trends of the closing price (Witte and Witte,2017):

- $n\text{-Day MA} = \frac{\text{Sum of Closing Prices for the Last } n \text{ Days}}{n}$

iii. The standard deviation of the closing price('Std_dev'), which Depicts daily price volatility with a 5-day rolling window (John,1999):

$$\text{Std\_dev}_t = \sqrt{\frac{\sum_{i=1}^{t}(X_i - X)^2}{t}}$$

Where:

- X_i is each closing price at time i.
- $\bar{X}$ is the mean closing price up to time t.
- The summation is performed over all closing prices up to time t.

iv. 'Price_Rise' which is a binary column that predicts price rises based on historical data:

$$\text{Price\_Rise}_t = \begin{cases} 1 & \text{if } \text{Close}_{t+1} > \text{Close}_t \\ 0 & \text{otherwise} \end{cases}$$

Where:

- Close_{t+1} is the closing price at the next time step.
- Close_t is the closing price at the current time step.
- Price_Rise_t is the binary column indicating whether the price rises (1) or not (0).

The dataset was further refined by eliminating rows with missing values and selecting the most relevant amzndata columns for analysis.

3. EXPLORATORY DATA ANALYSIS OF FEATURES

Employment of techniques, mostly graphical to summarize the main characteristics of the features was executed, which includes plots, histograms, descriptive statistics, scatter plots, correlation analysis, and data type explanation.

The observed closing price fluctuation movement has an inverse relationship with the stock volume, showing that when the closing price decreases volume tends to increase and the inverse is also true (Fig.1).

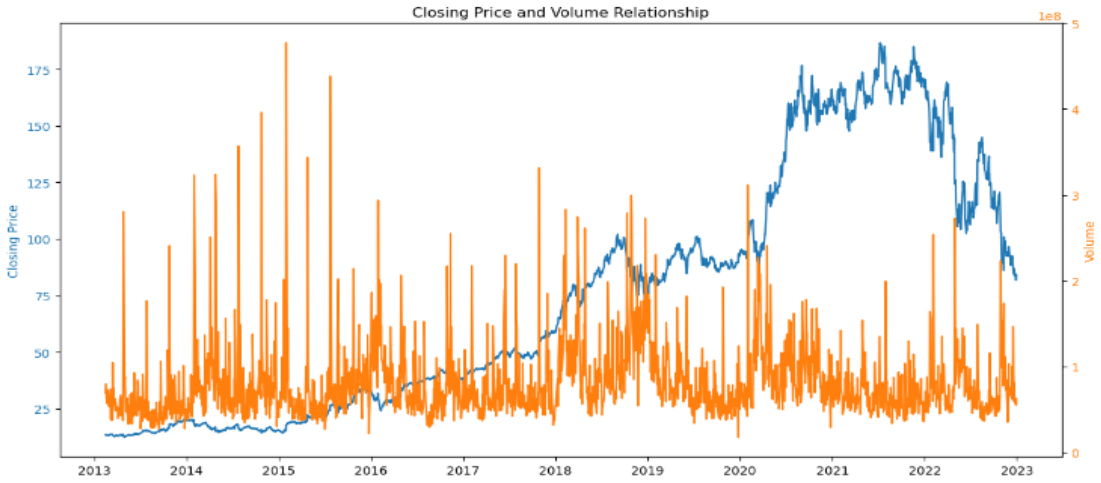


Figure 1:Plots Close Price and Volume

The '3day MA', 10day MA', and '30day MA' moving averages smooth out short-term fluctuations. The moving averages show a similar variation through time **(Fig.1)** and similar densities **(Fig.4)**. As the rolling window increases, a smoother line is observed, and an increase of the closing price is observed for most of the timeframe reaching surpassing 175$.

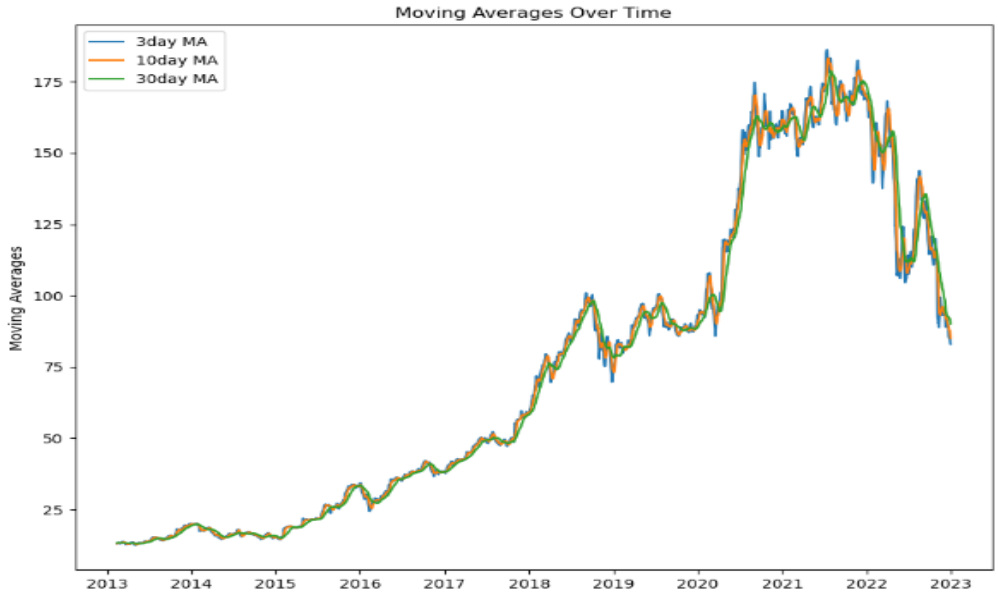


Figure 2:Plot for 3day, 10day,30day MA

The features 'H-L' and 'O-C' reflect information on price movements and market sentiment. The O-C price difference is higher as time increases and the market sentiment in 'H-L' also known as the daily price range show that it generally increases over time **(Fig.3)**. The daily standard deviation has an identical movement as the 'H-L' feature shows that the greater the higher the 'Std_dev' the greater the range of potential price movement range and a higher level of risk.

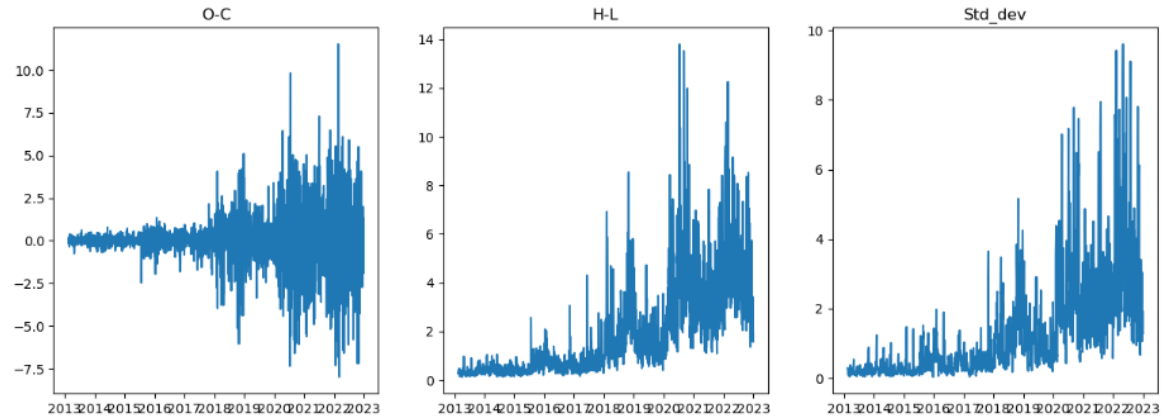


Figure 3:Plots for O-C, H-L, and Std_dev

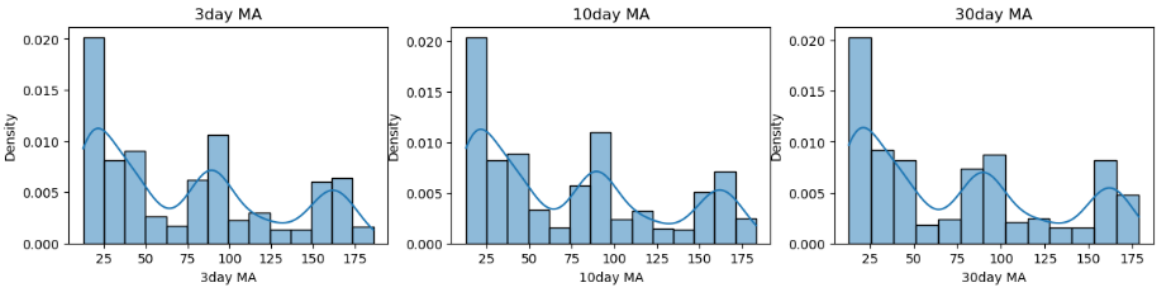


Figure 4: 3day, 10day,30day MA histogram and density

There is not much difference in the count binary column for price rises and falls in our data based on the created feature 'Price_Rise'**(Fig.5)**. Both predictions about closing price and standard deviation have subtle a positive trend and show a strong relationship in some 3 areas where the points seem to be closely packed.

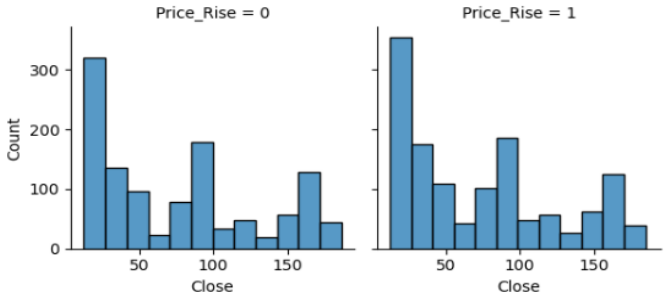


Figure 5:Price Rise & Fall Count Plot and Price Rise & Fall Scatter Plot in Relation to 'Close' and 'Std_dev'

From the descriptive statistics tables **(Fig.6)** a total of 2488 rows are observed due to data reduction techniques (previously 2518 rows). There is a yearly increase in statistical parameters for the closing price and observed 734.316% increase in mean closing price between 2013 to 2022. All years show acceptable counts corresponding to yearly trading days except the first year which was affected by data reduction techniques after computing moving averages.

	Volume	Open	High	Low	Close	H-L	O-C	3day MA	10day MA	30day MA	Std_dev	Pric
count	2.488000e+03	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.
mean	8.025653e+07	74.552471	75.399285	73.601288	74.508956	1.797997	-0.043515	74.452280	74.352092	74.045907	1.303940	0.
std	4.223134e+07	53.249129	53.901596	52.517793	53.192284	1.856050	1.561140	53.203891	53.209464	53.229335	1.466109	0.
min	1.762600e+07	12.447000	12.646500	12.287500	12.411500	0.125500	-7.990494	12.529667	12.747400	13.070567	0.035102	0.
25%	5.364552e+07	24.938375	25.491250	24.634874	24.919250	0.432500	-0.404249	24.938000	25.102738	23.672679	0.279866	0.
50%	6.876000e+07	67.102253	68.581001	65.091499	67.089001	0.975502	-0.004749	65.830332	64.543424	60.951291	0.695143	1.
75%	9.303150e+07	108.757378	112.648251	107.055000	108.927498	2.710249	0.383000	109.952501	110.587412	111.901904	1.818740	1.
max	4.771220e+08	187.199997	188.654007	184.839493	186.570496	13.794998	11.520508	186.154999	183.275400	178.717716	9.604773	1.

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2488 entries, 2013-02-14 to 2022-12-30
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Volume      2488 non-null   int64
 1   Open        2488 non-null   float64
 2   High        2488 non-null   float64
 3   Low         2488 non-null   float64
 4   Close       2488 non-null   float64
 5   H-L         2488 non-null   float64
 6   O-C         2488 non-null   float64
 7   3day MA     2488 non-null   float64
 8   10day MA    2488 non-null   float64
 9   30day MA    2488 non-null   float64
 10  Std_dev     2488 non-null   float64
 11  Price_Rise  2488 non-null   int32
dtypes: float64(10), int32(1), int64(1)
memory usage: 243.0 KB
```

Figure 6: Descriptive Statistics of the Dataset (description and information)

All features except 'Price_Rise' and 'O-C' are positively correlated due to the relationship with the closing price **(Fig.7)**. The 'Price_Rise' and 'O-C' however have a negative correlation with other features. A Higher closing price means less 'O-C' difference and a binary value of 0.

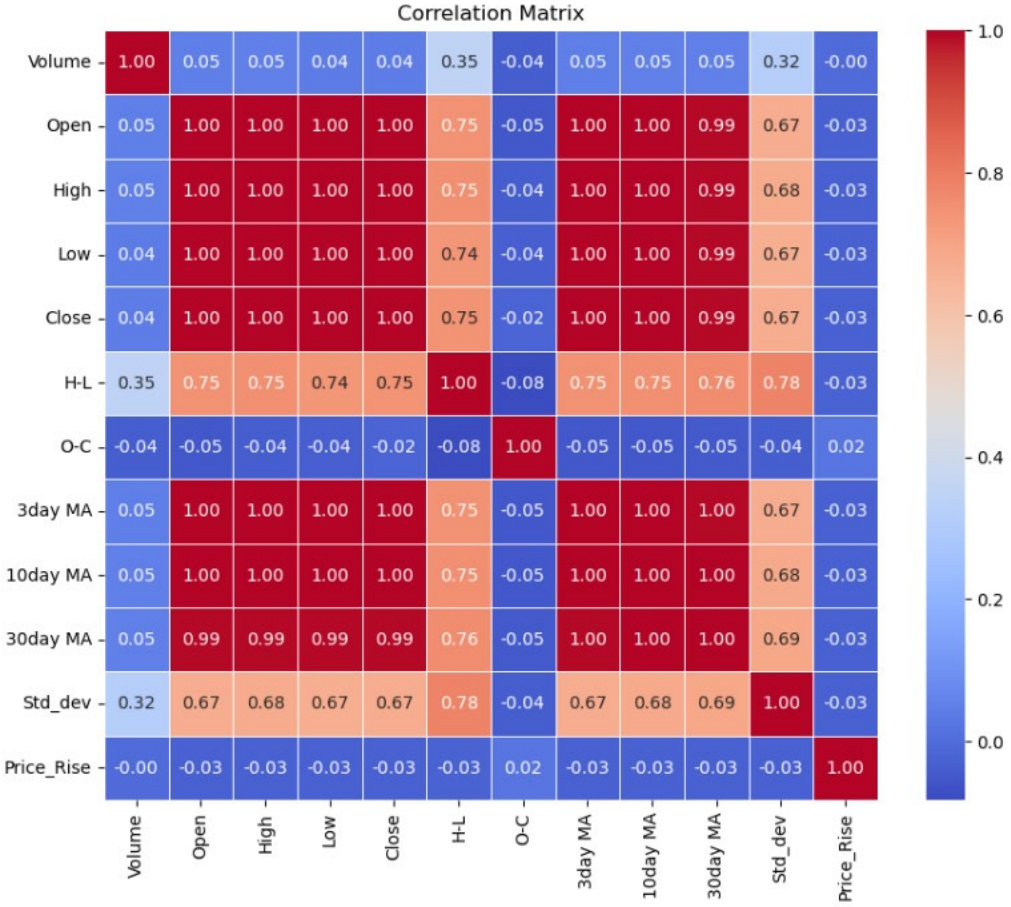


Figure 7: Correlation matrix of data variables

4. MACHINE LEARNING CLASSIFICATION

Evaluation of two machine learning models, Logistic Regression and Extra Trees Classifier for predicting stock price movements (binary classification) using features derived from historical stock data of Amazon.com, Inc was executed.

The data preprocessing process includes a dataset split into features (X) and the target variable (Y), representing the 'Price_Rise' binary column. The features ranged from the 'H-L' column to the 'Std_dev' column. The dataset was further divided into training and testing sets with a test size of 20%, and feature scaling was applied using the StandardScaler.

a. Logistic Regression

Logistic Regression is a statistical model that models the probability of a binary outcome using the logistic function (Muller and Guido,2016). The logistic function transforms a linear combination of input features into a value between 0 and 1, representing the probability of belonging to the positive class (Muller and Guido,2016). Results and evaluation are presented in section 6.

$$P(y = 1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

Where:

- $P(y = 1|x)$ is the probability of the positive class.
- β_0 is the intercept.
- β_1 is the coefficient for the feature x.

b. Extra Trees

Extra Trees (or Extremely Randomized Trees) Classifier is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mode of the classes (classification) or the average prediction (regression) of the individual trees (Muller and Guido,2016). It introduces additional randomness in the tree-building process by selecting random thresholds for each feature, hence promoting diversity among the trees (Geurts, Ernst, & Wehenkel, 2006). Results and evaluation are presented in section 6.

5. CROSS VALIDATION ACCURACY ANALYSIS

Cross-validation is a statistical technique used to assess the performance and generalizability of a machine learning model (Hastie, Tibshirani, Friedman, Friedman, 2009). In k-fold cross-validation, the dataset is divided into k subsets (folds), and the model is trained and evaluated k times, each time using a different fold as the test set and the remaining folds as the training set (Hastie, Tibshirani, Friedman, Friedman, 2009).

$$\text{Mean Accuracy} = \frac{1}{k}\sum_{i=1}^{k} \text{Accuracy}_i$$

where Accuracy_i is the accuracy of the model on the i-th fold.

$$\text{Standard Deviation} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(\text{Accuracy}_i - \text{Mean Accuracy})^2}$$

a. Cross Validation for Logistic Regression

The logistic regression model achieved an accuracy of 52% across the k=5 folds, meaning that the model when applied to the entire data set performs slightly better than random guessing, and a standard deviation of 0.02 suggests that the model's performance is relatively consistent across different subsets.

b. Cross Validation for Extra Trees

The Extra Trees Classifier, when assessed through cross-validation, demonstrated an average accuracy of 47%, indicating performance slightly below that of random guessing. Additionally, the standard deviation of 0.03 implies some variability in performance across different subsets, potentially highlighting the model's sensitivity to different portions of the dataset.

6. EVALUATION OF CLASSIFIERS PERFORMANCE/RESULTS

The logistic regression model for our data achieved an accuracy of 47%, with higher precision and recall for the positive class (1), indicating that the model is better at identifying instances of price rises:

	precision	recall	f1-score	support
0	0.44	0.14	0.21	255
1	0.47	0.82	0.60	243
accuracy			0.47	498
macro avg	0.46	0.48	0.41	498
weighted avg	0.46	0.47	0.40	498

The extra trees Classifier model demonstrated an accuracy of 50%, with balanced precision and recall for both classes:

	precision	recall	f1-score	support
0	0.51	0.42	0.46	255
1	0.49	0.58	0.53	243
accuracy			0.50	498
macro avg	0.50	0.50	0.50	498
weighted avg	0.50	0.50	0.49	498

Using these classifiers on 20% split data the Extra Tree Model outperforms the logistic regression model. Comparatively (section 5), the logistic regression model exhibited a slightly higher mean accuracy and lower standard deviation accuracy than the Extra Tree model in the overall data set:

```
In [137]: accuracy_scores = cross_val_score(modellr, X, Y, cv=5,
                                scoring=make_scorer(accuracy_score))

          # Print mean and standard deviation of accuracy
          print(f"Mean Accuracy: {accuracy_scores.mean():.2f}")
          print(f"Standard Deviation: {accuracy_scores.std():.2f}")

          Mean Accuracy: 0.52
          Standard Deviation: 0.02
```

```
In [138]: accuracy_scores = cross_val_score(modelet, X, Y, cv=5,
                                            scoring=make_scorer(accuracy_score))

          # Mean and standard deviation of accuracy
          print(f"Mean Accuracy: {accuracy_scores.mean():.2f}")
          print(f"Standard Deviation Accuracy: {accuracy_scores.std():.2f}")

          Mean Accuracy: 0.47
          Standard Deviation Accuracy: 0.03
```

7. PREDICTION OF PRICE USING X_TEST DATA: EXTRA TREE

The Decision to continue with the Extra Tee Model (section 5) is based on the outperformance of the model with the split data which outperforms the logistic regression model.

a. Classification Report

The classification report evaluates metric for the binary for the model, such as how many instances predicted as a class was correspondent to the class (precision), the percentage of how the actual class was correctly predicted (recall), the harmonic mean of precision and recall for the class (f1-score), and the number of instances in the data set for each class (support) (Müller and Guido, 2016).

	precision	recall	f1-score	support
0	0.51	0.42	0.46	255
1	0.49	0.58	0.53	243
accuracy			0.50	498
macro avg	0.50	0.50	0.50	498
weighted avg	0.50	0.50	0.49	498

b. Confusion Matrix

The confusion matrix describes the performance of a classification model on a set of test data for which the true values are known (Fawcett, 2006). 107 instances that are positive vs 141 instances that are negative are correctly classified by the model, 102 instances that are actually negative vs 148 instance that are actually positive are incorrectly classified as the opposite by the model **(Fig.8)**.

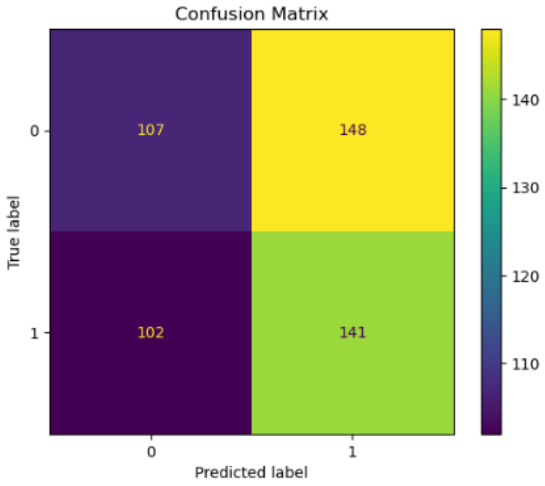


Figure 8:Confusion Matrix of the modelet (Extra Tree classification model)

c. ROC Curve

A Receiver Operating Characteristic (ROC) curve graphically represents binary classification performance across discrimination thresholds (Fawcett, 2006). The Area Under the Curve (AUC) of an ROC curve is also commonly used as a summary metric for the model's performance, which is 51% for our data set, close to our accuracy **(Fig.9)**.

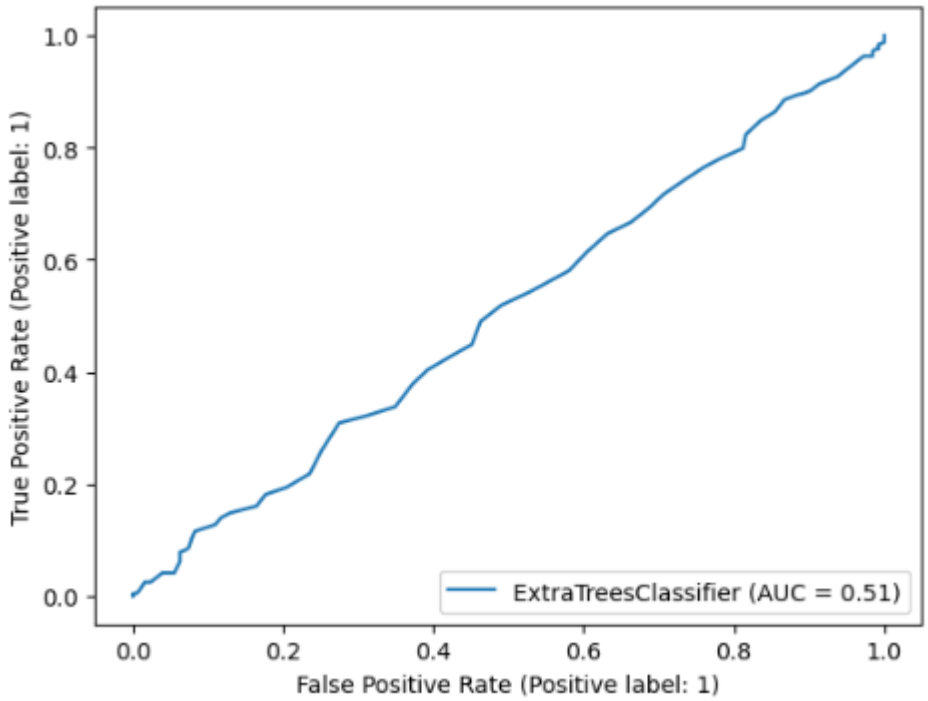


Figure 9: ROC Curve of the modelet(Extra Tree classification mode

8. MARKET AND STRATEGY RETURNS

Preprocessing data for market and strategy returns using the Extra Tree model (section 6), involves integrating predicted values into a novel 'Y_predicted' column and handling potential NaN values.

Market returns are computed through a 'Tomorrows Returns' column in the trade dataset, initialized at 0 in decimal notation. This column captures the logarithmic returns of the current day, calculated as the logarithm of the present closing price divided by the closing price of the preceding day. The shift of these values by one position ensures the alignment of tomorrow's returns with today's prices (Yao, 2023).

$$\text{Market Return} = \frac{\text{Closing Price}_{\text{today}} - \text{Closing Price}_{\text{yesterday}}}{\text{Closing Price}_{\text{yesterday}}}$$

Strategy returns are then determined, assuming long positions for true predicted 'Y' values and embracing short positions for false predictions. The introduction of a 'Strategy Returns' column, initialized at 0 for floating-point values, serves as the repository for 'Tomorrows Returns' values. If 'Y_predicted' indicates a long position, 'Tomorrows Returns' values are stored in the 'Strategy Returns' column (Yao, 2023). Conversely, the negative of 'Tomorrows Returns' is preserved for short positions(Yao, 2023).

9. CUMULATIVE RETURNS ANALYSIS

This method establishes the foundation for evaluating the chosen model's efficacy in shaping trading strategies. Cumulative returns for both the market and the strategy were computed using the cumsum () function.

Considering our data set up until the beginning of the second quarter of 2022 (2022-04) Amazon strategies seem to behave differently with market strategy returns mostly outperforming strategy returns. After that time frame the strategies have a similar trend, however, this time strategy returns outperform market returns. Another distinction is that strategy returns are positive while market returns are negative.

Figure 10: Plot of cumulative market and strategy returns based on Y_prediction

10. INTERPRETATION AND DISCUSSION

The analysis of Amazon stock prices, particularly utilizing the Extra Trees Classifier, offers valuable insights into the intersection of machine learning and stock price prediction. Considering the assumption - Can machine learning predict stock price rises? - the analysis results, with an observed accuracy of 50%, indicate a mixed capability (No). While the machine learning model exhibits some predictive power, it falls short of achieving consistently accurate predictions, acknowledging the inherent complexity of stock market dynamics.

Implications suggest while capturing patterns, machine learning may face challenges in the unpredictable financial markets. Caution is warranted due to factors like market sentiment and unforeseen events (El Hajj and Hammoud 2023).

Debate based on Literature and Theories: The efficient market hypothesis, a cornerstone in financial economics, posits that stock prices already reflect all relevant information, challenging the notion of consistent predictability. Fama (1970) argues that, in an efficient market, it is difficult to achieve consistently superior returns based on historical information. Behavioural finance theories (Shiller, 1981) emphasize the influence of psychological factors, introducing unpredictability.

The Adaptive Market Hypothesis (Lo, 2004) acknowledges market dynamics evolve, making predictions challenging. Additionally, Malkiel (2003) advocates for the random walk theory, suggesting that stock prices move randomly, challenging the feasibility of consistent prediction strategies. Thoroughly acknowledging these theories provides a nuanced perspective on the role of machine learning in stock price forecasting, aligning with the observed variability in model performance.

11. CONCLUSION

While the data shows that the Extra Trees Classifier can partially predict Amazon stock values significant limitations remain. Despite positive outcomes, machine learning algorithms struggle with unforeseen events and potential overfitting. Generalizing to new market conditions is challenging, and presuming historical trends repeat oversimplifies the intricate financial landscape. While offering valuable insights, it's crucial to acknowledge market volatility and model constraints. A balanced integration of machine learning with traditional approaches is advised, underscoring the need for prudent decision-making. Future research should focus on refining models to adapt to diverse market situations, ensuring ongoing development and reliability.

II. References

- Bezos, J. (2022) The Evolution of Amazon: From Online Bookstore to Global E-commerce Giant. Business Insights. Available at: https://www.businessinsights.com/amazon-evolution (Accessed: 16 November 2023).
- El Hajj, M. and Hammoud, J., 2023. Unveiling the Influence of Artificial Intelligence and Machine Learning on Financial Markets: A Comprehensive Analysis of AI Applications in Trading, Risk Management, and Financial Operations. Journal of Risk and Financial Management, 16(10), p.434.
- Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. The Journal of Finance, 25(2), 383-417.
- Fawcett, T., 2006. An introduction to ROC analysis. Pattern recognition letters, 27(8), pp.861-874.
- Flach, P. A. (2015). Precision-Recall and ROC Curves: A Tutorial. In LNAI: Machine Learning and Knowledge Discovery in Databases, 2015, pp. 37-52.
- Geurts, P., Ernst, D., & Wehenkel, L. (2006) 'Extremely randomized trees,' Machine learning, 63(1), 3-42.
- Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: Springer.
- John, J., 1999. Technical analysis of the financial markets.
- Lo, A. W. (2004). The Adaptive Markets Hypothesis. The Journal of Portfolio Management, 30(5), 15-29.
- Malkiel, B. G. (2003). The Efficient Market Hypothesis and Its Critics. Journal of Economic Perspectives, 17(1), 59-82
- Malkiel, B.G., 1973. A random walk down Wall Street [by] Burton G. Malkiel. Norton.
- Müller, A.C. and Guido, S., 2016. Introduction to machine learning with Python: a guide for data scientists. " O'Reilly Media, Inc.".
- Shiller, R. J. (1981). Do Stock Prices Move Too Much to Be Justified by Subsequent Changes in Dividends? The American Economic Review, 71(3), 421-436.
- Witte, R.S. and Witte, J.S., 2017. Statistics. John Wiley & Sons.
- Yao, Y, 2023. Seminar 6.2 Case-CLF-TradingStrategies-Full. Module FNCE043W Class Notes. Available at:https://learning.westminster.ac.uk/bbcswebdav/pid-4777008-dt-content-rid-43364841_1/xid-43364841_1 (Accessed: 16 November 2023).

III. Appendices:

The appendices section follows the same order as the table content with properly identified figures, analysis, predictions, and reports.

2.Data Collection and Preprocessing

Import library, metrics, and exclude warnings

In [117]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [118]:
```
#Import libraries and metrics
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import accuracy_score,roc_curve, classification_report
from sklearn.metrics import confusion_matrix, make_scorer, roc_auc_score
from sklearn.metrics import RocCurveDisplay, ConfusionMatrixDisplay
```

In [119]:
```
pd.options.mode.chained_assignment = None
```

2.a.Amazon.com,Inc Historical Data

Download data from Yahoo Finance

In [120]:
```
amzndata = yf.download('AMZN ', start="2013-01-01", end='2022-12-31')

amzndata
```

```
[**********100%**********]  1 of 1 completed
```

Out[120]:

Date	Open	High	Low	Close	Adj Close	Volume
2013-01-02	12.804000	12.905000	12.663000	12.865500	12.865500	65420000
2013-01-03	12.863500	13.044000	12.818500	12.924000	12.924000	55018000
2013-01-04	12.879000	12.990000	12.832500	12.957500	12.957500	37484000
2013-01-07	13.148500	13.486500	13.133500	13.423000	13.423000	98200000
2013-01-08	13.353500	13.449000	13.178500	13.319000	13.319000	60214000
...	...	...	...	...	...	...
2022-12-23	83.250000	85.779999	82.930000	85.250000	85.250000	57433700
2022-12-27	84.970001	85.349998	83.000000	83.040001	83.040001	57284000
2022-12-28	82.800003	83.480003	81.690002	81.820000	81.820000	58228800
2022-12-29	82.870003	84.550003	82.550003	84.180000	84.180000	54995900
2022-12-30	83.120003	84.050003	82.470001	84.000000	84.000000	62401200

2518 rows × 6 columns

2.b.Feature Engineering

Data Reduction

In [121]:
```
amzndata = amzndata.dropna()
amzndata = amzndata[['Volume','Open', 'High', 'Low', 'Close']]
amzndata
```

Out[121]:

Date	Volume	Open	High	Low	Close
2013-01-02	65420000	12.804000	12.905000	12.663000	12.865500
2013-01-03	55018000	12.863500	13.044000	12.818500	12.924000
2013-01-04	37484000	12.879000	12.990000	12.832500	12.957500
2013-01-07	98200000	13.148500	13.486500	13.133500	13.423000
2013-01-08	60214000	13.353500	13.449000	13.178500	13.319000
...	...	...	...	...	...
2022-12-23	57433700	83.250000	85.779999	82.930000	85.250000
2022-12-27	57284000	84.970001	85.349998	83.000000	83.040001
2022-12-28	58228600	82.800003	83.480003	81.690002	81.820000
2022-12-29	54995900	82.870003	84.550003	82.550003	84.180000
2022-12-30	62401200	83.120003	84.050003	82.470001	84.000000

2518 rows × 5 columns

Future Creation

In [122]:
```
amzndata['H-L'] = amzndata['High'] - amzndata['Low']
amzndata['O-C'] = amzndata['Close'] - amzndata['Open']

amzndata['3day MA'] = amzndata['Close'].shift(1).rolling(window = 3).mean()
amzndata['10day MA'] = amzndata['Close'].shift(1).rolling(window = 10).mean()
amzndata['30day MA'] = amzndata['Close'].shift(1).rolling(window = 30).mean()

amzndata['Std_dev']= amzndata['Close'].rolling(5).std()

amzndata['Price_Rise'] = np.where(amzndata['Close'].shift(-1) >amzndata['Close'], 1, 0)

amzndata = amzndata.dropna()
amzndata.head(10)
```

Out[122]:

Date	Volume	Open	High	Low	Close	H-L	O-C	3day MA	10day MA	30day MA	Std_dev	Price_Rise
2013-02-14	69260000	13.3685	13.5325	13.2700	13.4620	0.262500	0.0935	13.089667	13.13575	13.329367	0.288771	0
2013-02-15	79598000	13.3815	13.4460	13.1555	13.2545	0.290500	-0.1270	13.290167	13.15445	13.349250	0.288036	1
2013-02-19	57084000	13.2955	13.5055	13.2250	13.4875	0.280499	0.1920	13.396667	13.15490	13.360267	0.236784	0
2013-02-20	70578000	13.5100	13.7150	13.3185	13.3205	0.396501	-0.1895	13.401333	13.20375	13.377933	0.105347	0
2013-02-21	72748000	13.2560	13.4740	13.1625	13.2970	0.311500	0.0410	13.354167	13.20135	13.374517	0.103956	0
2013-02-22	62496000	13.3310	13.3555	13.0805	13.2710	0.275001	-0.0600	13.368333	13.21995	13.373783	0.093663	0
2013-02-25	60648000	13.3470	13.4345	12.9825	12.9935	0.452000	-0.3535	13.296167	13.24590	13.372233	0.178159	0
2013-02-26	66972000	13.0445	13.1020	12.7865	12.9680	0.315500	-0.0765	13.187167	13.23550	13.363117	0.173880	1
2013-02-27	58162000	12.9700	13.2915	12.8430	13.1625	0.448500	0.1925	13.077500	13.24625	13.348817	0.152766	1
2013-02-28	53344000	13.0905	13.3500	13.0315	13.2135	0.318501	0.1230	13.041334	13.26900	13.333017	0.134575	1

3- EDA of Created Features

Figure 1:Plot for Closing Price and Volume

In [123]:
```
fig, ax1 = plt.subplots(figsize=(12, 6))

color = 'tab:blue'
ax1.set_ylabel('Closing Price', color=color)
ax1.plot(amzndata['Close'], label='Close', color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()
color = 'tab:orange'
ax2.set_ylabel('Volume', color=color)
ax2.plot(amzndata['Volume'], label='Volume', color=color)
ax2.tick_params(axis='y', labelcolor=color)


plt.title('Closing Price and Volume Relationship')
fig.tight_layout()
plt.show()
```

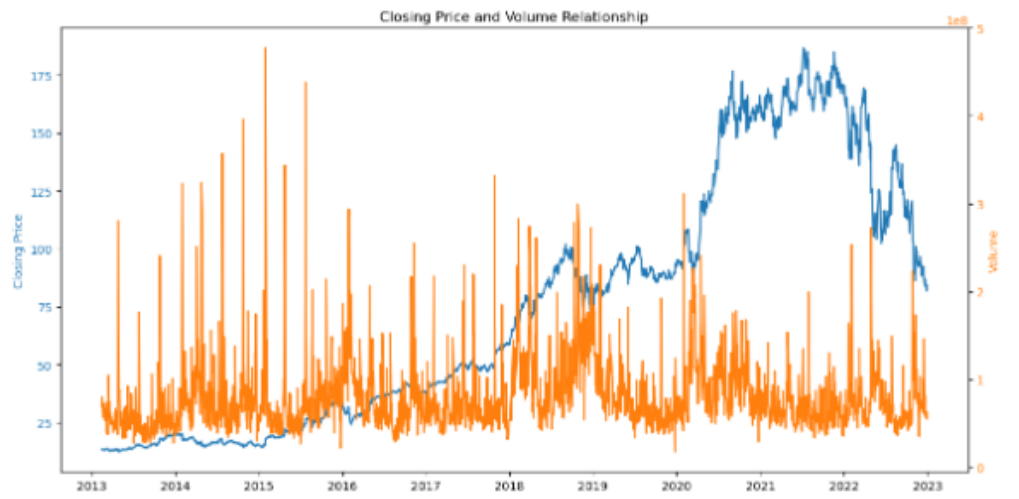


Figure 2: Plot for 3day, 10day,30day MA

In [124]:
```
plt.figure(figsize=(10,8 ))

plt.plot(amzndata['3day MA'], label='3day MA')
plt.plot(amzndata['10day MA'], label='10day MA')
plt.plot(amzndata['30day MA'], label='30day MA')

plt.title('Moving Averages Over Time')
plt.ylabel('Moving Averages')
plt.legend()
plt.show()
```

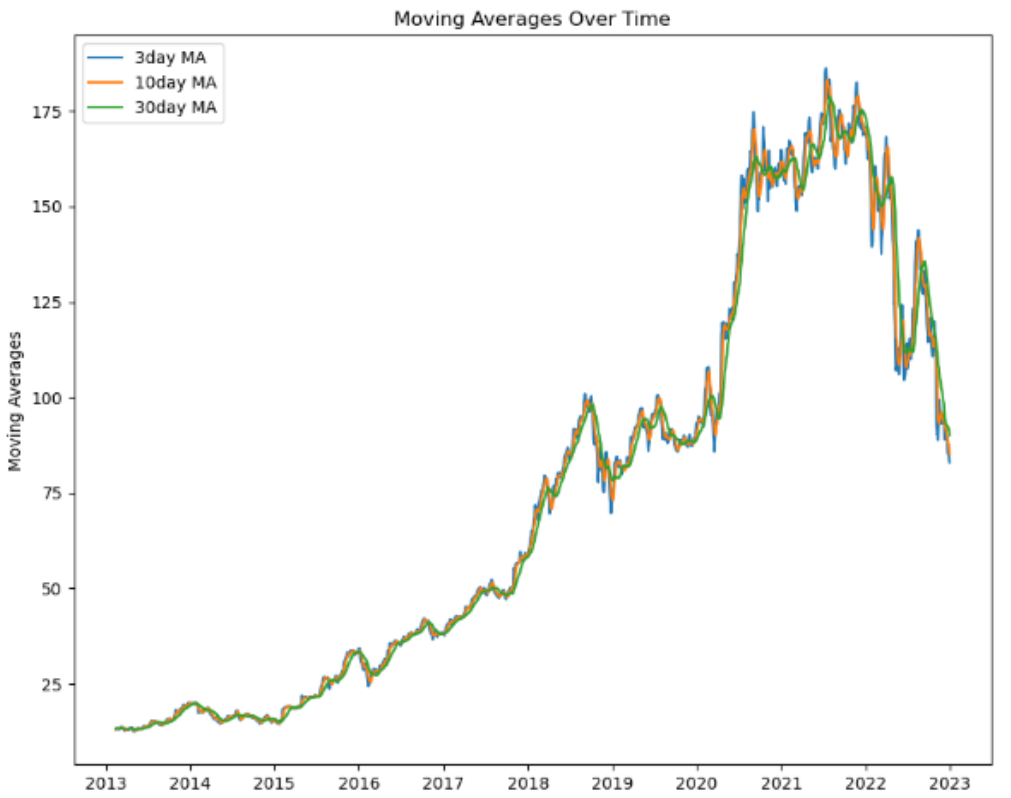


Figure 3: Plots for O-C,H-L and Std_dev

In [125]:
```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

axes[0].plot(amzndata['O-C'])
axes[0].set_title('O-C')
axes[1].plot(amzndata['H-L'])
axes[1].set_title('H-L')
axes[2].plot(amzndata['Std_dev'])
axes[2].set_title('Std_dev')

plt.show()
```

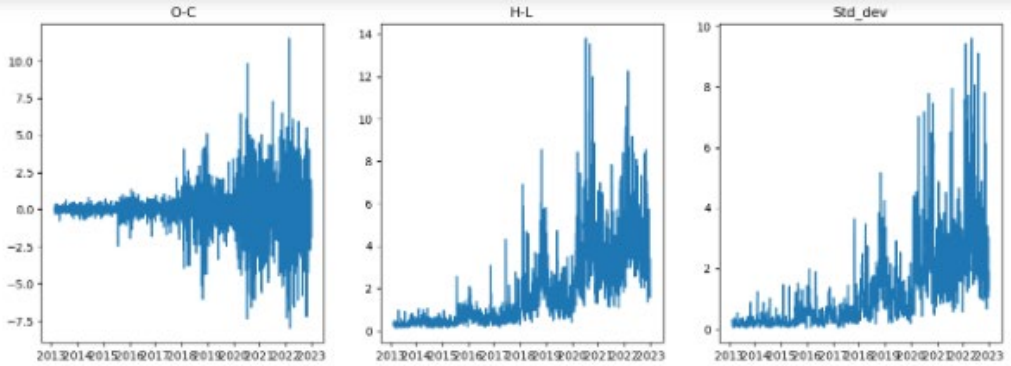


Figure 4: 3day, 10day,30day MA histogram and density

In [126]:
```
fig, axes = plt.subplots(1, 3, figsize=(15,3))  # 1 row, 3 columns

sns.histplot(data=amzndata, x="3day MA", kde=True, stat="density", ax=axes[0])
axes[0].set_title('3day MA')

sns.histplot(data=amzndata, x="10day MA", kde=True, stat="density", ax=axes[1])
axes[1].set_title('10day MA')

sns.histplot(data=amzndata, x="30day MA", kde=True, stat="density", ax=axes[2])
axes[2].set_title('30day MA')

plt.show()
```

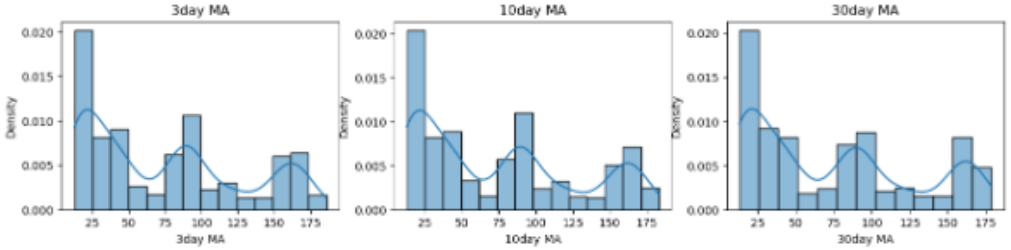


Figure 5: Price Rise & Fall Count Plot and Price Rise & Fall Scatter Plot in Relation to 'Close' and 'Std_dev'

In [127]:
```
chart = sns.FacetGrid(amzndata, col='Price_Rise')
chart.map(sns.histplot, 'Close')

chart = sns.FacetGrid(amzndata, col='Price_Rise')
chart.map(plt.scatter, 'Close','Std_dev')
```

Out[127]: <seaborn.axisgrid.FacetGrid at 0x1a8c3d699a0>

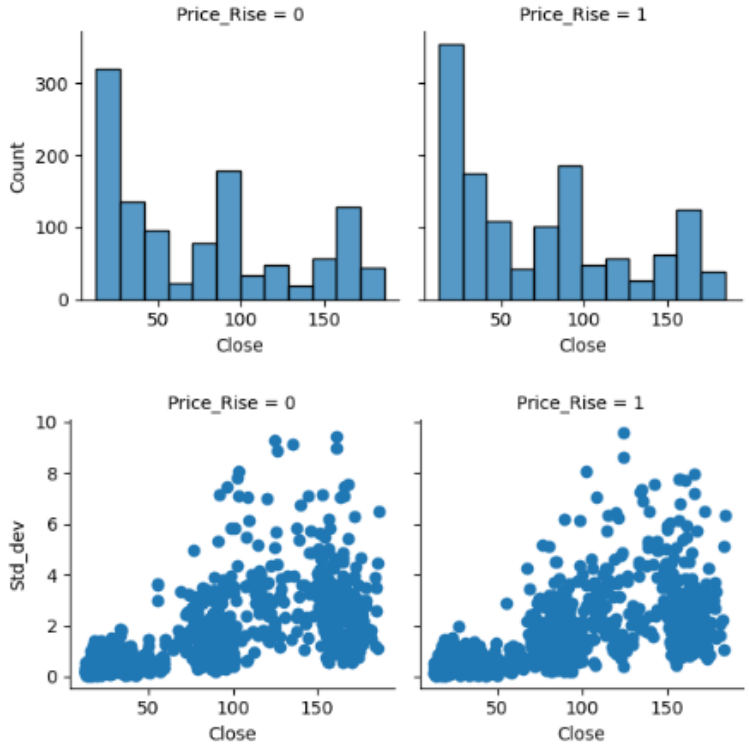


Fig 6: Descriptive Statistics of the Data set (description and information)

```
In [128]: amzndata.describe()
```

Out[128]:

	Volume	Open	High	Low	Close	H-L	O-C	3day MA	10day MA
count	2.488000e+03	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000	2488.000000
mean	8.025653e+07	74.552471	75.399285	73.601288	74.508956	1.797997	-0.043515	74.452280	74.352092
std	4.223134e+07	53.249129	53.901596	52.517793	53.192284	1.856050	1.561140	53.203891	53.209464
min	1.762600e+07	12.447000	12.646500	12.287500	12.411500	0.125500	-7.990494	12.529667	12.747400
25%	5.364552e+07	24.938375	25.491250	24.634874	24.919250	0.432500	-0.404249	24.938000	25.102738
50%	6.876000e+07	67.102253	68.581001	65.091499	67.089001	0.975502	-0.004749	65.830332	64.543424
75%	9.303150e+07	108.757378	112.648251	107.055000	108.927498	2.710249	0.383000	109.952501	110.587412
max	4.771220e+08	187.199997	188.654007	184.839493	186.570496	13.794998	11.520508	186.154999	183.275400

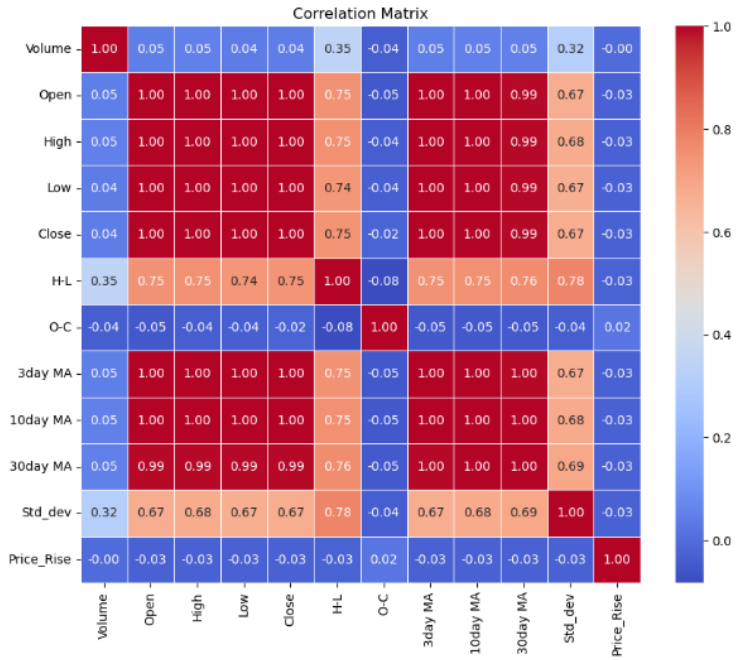
```
In [129]: amzndata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2488 entries, 2013-02-14 to 2022-12-30
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Volume      2488 non-null   int64
 1   Open        2488 non-null   float64
 2   High        2488 non-null   float64
 3   Low         2488 non-null   float64
 4   Close       2488 non-null   float64
 5   H-L         2488 non-null   float64
 6   O-C         2488 non-null   float64
 7   3day MA     2488 non-null   float64
 8   10day MA    2488 non-null   float64
 9   30day MA    2488 non-null   float64
 10  Std_dev     2488 non-null   float64
 11  Price_Rise  2488 non-null   int32
dtypes: float64(10), int32(1), int64(1)
memory usage: 243.0 KB
```

Fig 7: Correlation matrix of data variables

```
In [130]: corr_matrix = amzndata.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix')
plt.show()
```



4.Machine Learning Classification Methods

Data Preprocessing

In [131]:
```
#Set target variable y(Price_Rise 0 or 1 )and features x
#(from H-L column to Std_dev column)
X = amzndata.iloc[:, 5:-1]
Y = amzndata.iloc[:, -1]
```

In [132]:
```
X
```

Out[132]:

Date	H-L	O-C	3day MA	10day MA	30day MA	Std_dev
2013-02-14	0.262500	0.093500	13.089667	13.135750	13.329367	0.288771
2013-02-15	0.290500	-0.127000	13.290167	13.154450	13.349250	0.288036
2013-02-19	0.280499	0.192000	13.396667	13.154900	13.360267	0.236784
2013-02-20	0.396501	-0.189500	13.401333	13.203750	13.377933	0.105347
2013-02-21	0.311500	0.041000	13.354167	13.201350	13.374517	0.103956
...	...	...	...	...	...	...
2022-12-23	2.849998	2.000000	85.250000	88.068999	92.181999	1.064508
2022-12-27	2.349998	-1.930000	85.269999	87.685000	91.802666	1.445066
2022-12-28	1.790001	-0.980003	84.026667	86.934000	91.211000	1.927674
2022-12-29	2.000000	1.309998	83.370000	85.867000	90.655333	1.282158
2022-12-30	1.580002	0.879997	83.013334	85.127000	90.163333	1.292679

2488 rows × 6 columns

In [133]:
```
# Split the data into training and testing sets

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
                                   shuffle=False)
X_test
```

Out[133]:

Date	H-L	O-C	3day MA	10day MA	30day MA	Std_dev
2021-01-11	2.319000	-1.690002	158.054001	161.049051	159.645550	2.008823
2021-01-12	2.806992	0.041504	157.651164	160.756650	159.527450	1.432383
2021-01-13	3.393494	1.872498	156.962331	159.941000	159.403267	1.499564
2021-01-14	2.870499	-2.002502	156.682165	159.160449	159.399684	1.512147
2021-01-15	2.369003	-0.938507	156.903168	158.368550	159.245334	1.180774
...	...	...	...	...	...	...
2022-12-23	2.849998	2.000000	85.250000	88.068999	92.181999	1.064508
2022-12-27	2.349998	-1.930000	85.269999	87.685000	91.802666	1.445066
2022-12-28	1.790001	-0.980003	84.026667	86.934000	91.211000	1.927674
2022-12-29	2.000000	1.309998	83.370000	85.867000	90.655333	1.282158
2022-12-30	1.580002	0.879997	83.013334	85.127000	90.163333	1.292679

498 rows × 6 columns

In [134]:
```
# Standardize the features (optional but can be beneficial for
#Logistic regression)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

4.a.Logistic Regression

In [135]:
```
# Model
modellr = LogisticRegression(random_state=101)
# Train the model
modellr.fit(X_train, Y_train)
# Predict on the test set
Y_pred = modellr.predict(X_test)

print (classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.44      0.14      0.21       255
           1       0.47      0.82      0.60       243

    accuracy                           0.47       498
   macro avg       0.46      0.48      0.41       498
weighted avg       0.46      0.47      0.40       498
```

4.b.Extra Trees

In [136]:
```
#Model
modelet = ExtraTreesClassifier(random_state=101)
# Train the model
modelet.fit(X_train, Y_train)
# Predict on the test set
Y_pred = modelet.predict(X_test)

print (classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.51      0.42      0.46       255
           1       0.49      0.58      0.53       243

    accuracy                           0.50       498
   macro avg       0.50      0.50      0.50       498
weighted avg       0.50      0.50      0.49       498
```

5.Cross Validation

5.a.Cross Validation for Logistic Regression

In [137]:
```
accuracy_scores = cross_val_score(modellr, X, Y, cv=5,
                  scoring=make_scorer(accuracy_score))

# Print mean and standard deviation of accuracy
print(f"Mean Accuracy: {accuracy_scores.mean():.2f}")
print(f"Standard Deviation: {accuracy_scores.std():.2f}")
```

```
Mean Accuracy: 0.52
Standard Deviation: 0.02
```

5.a.Cross Validation for Extra Tree

In [138]:
```
accuracy_scores = cross_val_score(modelet, X, Y, cv=5,
                  scoring=make_scorer(accuracy_score))

# Mean and standard deviation of accuracy
print(f"Mean Accuracy: {accuracy_scores.mean():.2f}")
print(f"Standard Deviation Accuracy: {accuracy_scores.std():.2f}")
```

```
Mean Accuracy: 0.47
Standard Deviation Accuracy: 0.03
```

7.Prediction of Price Rise Using Extra Trees on X_test Data

Model (repeated)

In [139]:
```
modelet = ExtraTreesClassifier(random_state=101)
modelet.fit(X_train, Y_train)
Y_pred = modelet.predict(X_test)

classification_rep = classification_report(Y_test, Y_pred)
print("Classification Report:\n", classification_rep)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.51      0.42      0.46       255
           1       0.49      0.58      0.53       243

    accuracy                           0.50       498
   macro avg       0.50      0.50      0.50       498
weighted avg       0.50      0.50      0.49       498
```

Fig 8: Confusion Matrix of the modelet(Extra Tree classification model)

In [140]:
```
# Confusion Matrix
conf_matrix = confusion_matrix(Y_test, Y_pred)

# Evaluate the modelet by means of a Confusion Matrix
matrix = ConfusionMatrixDisplay.from_estimator(modelet, X_test, Y_test)
plt.title('Confusion Matrix')
plt.show(matrix)
plt.show()
```

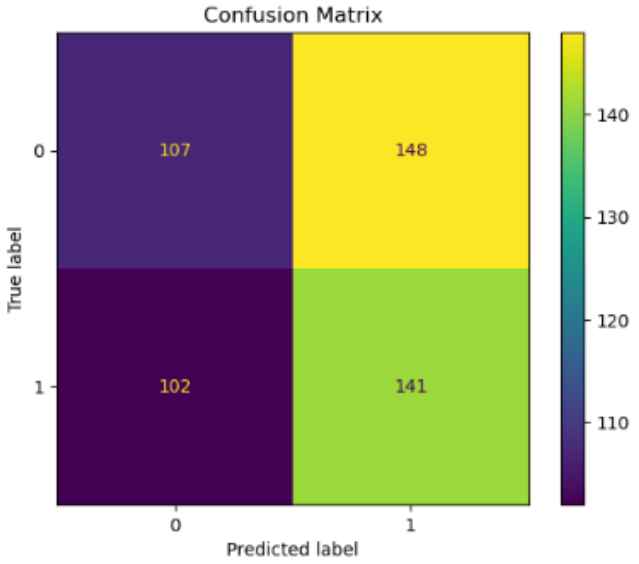


Fig 9: Roc Curve of the modelet(Extra Tree classification model)

In [141]:
```
# ROC Curve
log_disp = RocCurveDisplay.from_estimator(modelet, X_test, Y_test)
```

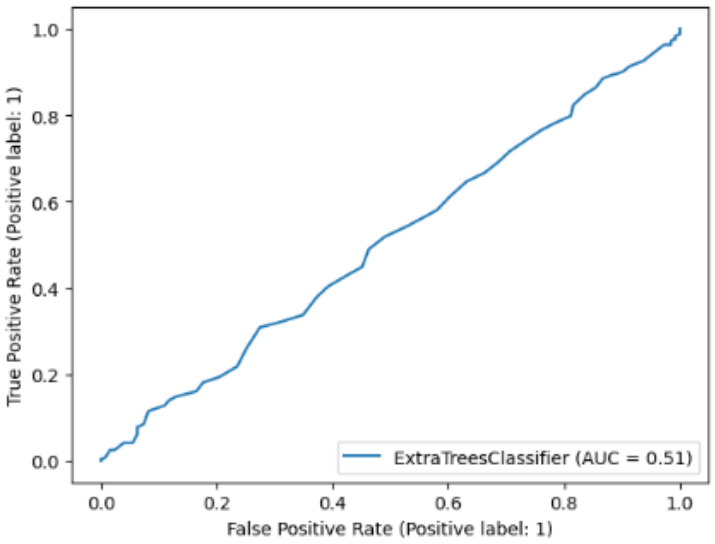
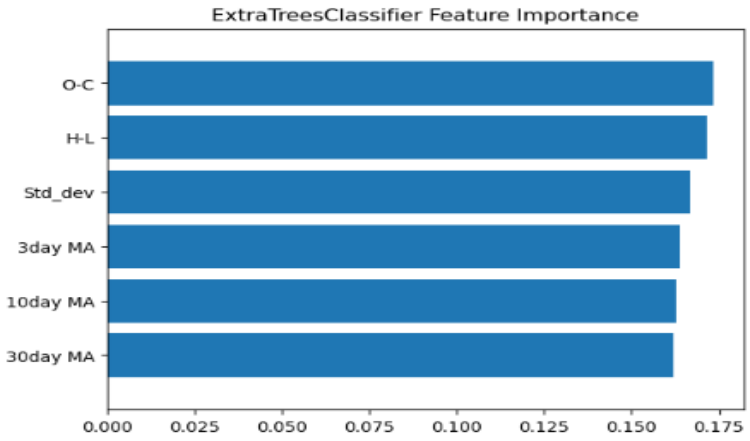


Fig 10: Importance of classifiers in modelet

In [142]:
```
#Importance of classifiers
feature_names=X.columns
importance = modelet.feature_importances_
indices = np.argsort(importance)
range1 = range(len(importance[indices]))
plt.figure()
plt.title("ExtraTreesClassifier Feature Importance")
plt.barh(range1,importance[indices])
plt.yticks(range1, feature_names[indices])
plt.ylim([-1, len(range1)])
plt.show()
```



8.Market and Return Strategies

Data Preprocessing

In [143]:
```
#Create new column Y_pred
amzndata['Y_pred'] = np.NaN
amzndata.iloc[(len(amzndata) - len(Y_pred)):,-1] = Y_pred
trade_amzndata = amzndata.dropna()
trade_amzndata
```

Out[143]:

pen	High	Low	Close	H-L	O-C	3day MA	10day MA	30day MA	Std_dev	Price_Rise	Y_pred
497	157.819000	155.500000	155.710495	2.319000	-1.690002	158.054001	161.049051	159.645550	2.008823	1	1.0
000	157.106995	154.300003	156.041504	2.806992	0.041504	157.651164	160.756650	159.527450	1.432383	1	0.0
997	159.497498	156.104004	158.294495	3.393494	1.872498	156.962331	159.941000	159.403267	1.499564	0	1.0
007	158.899994	156.029495	156.373505	2.870499	-2.002502	156.682165	159.160449	159.399684	1.512147	0	1.0
001	157.127502	154.758499	155.212494	2.369003	-0.938507	156.903168	158.368550	159.245334	1.180774	1	0.0
...	...	...	...	...	...	...	...	...	...	...	...
000	85.779999	82.930000	85.250000	2.849998	2.000000	85.250000	88.068999	92.181999	1.064508	0	1.0
001	85.349998	83.000000	83.040001	2.349998	-1.930000	85.269999	87.685000	91.802666	1.445066	0	1.0
003	83.480003	81.690002	81.820000	1.790001	-0.980003	84.026667	86.934000	91.211000	1.927674	1	1.0
003	84.550003	82.550003	84.180000	2.000000	1.309998	83.370000	85.867000	90.655333	1.282158	0	1.0
003	84.050003	82.470001	84.000000	1.580002	0.879997	83.013334	85.127000	90.163333	1.292679	0	1.0

Computation of market returns

In [144]:
```
trade_amzndata['Tomorrows Returns'] = 0.
trade_amzndata['Tomorrows Returns'] = np.log(trade_amzndata['Close']/trade_amzndata['Close'].shift(1))
trade_amzndata['Tomorrows Returns'] = trade_amzndata['Tomorrows Returns'].shift(-1)
trade_amzndata
```

Out[144]:

igh	Low	Close	H-L	O-C	3day MA	10day MA	30day MA	Std_dev	Price_Rise	Y_pred	Tomorrows Returns
000	155.500000	155.710495	2.319000	-1.690002	158.054001	161.049051	159.645550	2.008823	1	1.0	0.002124
995	154.300003	156.041504	2.806992	0.041504	157.651164	160.756650	159.527450	1.432383	1	0.0	0.014335
498	156.104004	158.294495	3.393494	1.872498	156.962331	159.941000	159.403267	1.499564	0	1.0	-0.012210
994	156.029495	156.373505	2.870499	-2.002502	156.682165	159.160449	159.399684	1.512147	0	1.0	-0.007452
502	154.758499	155.212494	2.369003	-0.938507	156.903168	158.368550	159.245334	1.180774	1	0.0	0.005304
...	...	...	...	...	...	...	...	...	...	...	...
999	82.930000	85.250000	2.849998	2.000000	85.250000	88.068999	92.181999	1.064508	0	1.0	-0.026266
998	83.000000	83.040001	2.349998	-1.930000	85.269999	87.685000	91.802666	1.445066	0	1.0	-0.014801
003	81.690002	81.820000	1.790001	-0.980003	84.026667	86.934000	91.211000	1.927674	1	1.0	0.028436
003	82.550003	84.180000	2.000000	1.309998	83.370000	85.867000	90.655333	1.282158	0	1.0	-0.002141
003	82.470001	84.000000	1.580002	0.879997	83.013334	85.127000	90.163333	1.292679	0	1.0	NaN

Computation of strategy returns

In [32]:
```
#Strategy Returns based on Y_Pred
trade_amzndata['Strategy Returns'] = 0.
trade_amzndata['Strategy Returns'] = np.where(trade_amzndata['Y_pred'] == True,
                                    trade_amzndata['Tomorrows Returns'], - trade_amzndata['Tomorrows Returns'])
trade_amzndata
```

Out[32]:

	Open	High	Low	Close	H-L	O-C	3day MA	10day MA	30day MA	Std_dev	Price_Rise	Y_pred	Tomorrows Returns	Strategy Returns
)	157.400497	157.819000	155.500000	155.710495	2.319000	-1.690002	158.054001	161.049051	159.645550	2.008823	1	1.0	0.002124	0.002124
)	156.000000	157.106995	154.300003	156.041504	2.806992	0.041504	157.651164	160.756650	159.527450	1.432383	1	0.0	0.014335	-0.014335
)	156.421997	159.497498	156.104004	158.294495	3.393494	1.872498	156.962331	159.941000	159.403267	1.499564	0	1.0	-0.012210	-0.012210
)	158.376007	158.899994	156.029495	156.373505	2.870499	-2.002502	156.682165	159.160449	159.399684	1.512147	0	1.0	-0.007452	-0.007452
)	156.151001	157.127502	154.758499	155.212494	2.369003	-0.938507	156.903168	158.368550	159.245334	1.180774	1	0.0	0.005304	-0.005304
.	...	...	...	...	...	...	...	...	...	...	...	...	...	...
)	83.250000	85.779999	82.930000	85.250000	2.849998	2.000000	85.250000	88.068999	92.181999	1.064508	0	1.0	-0.026266	-0.026266
)	84.970001	85.349998	83.000000	83.040001	2.349998	-1.930000	85.269999	87.685000	91.802666	1.445066	0	1.0	-0.014801	-0.014801
)	82.800003	83.480003	81.690002	81.820000	1.790001	-0.980003	84.026667	86.934000	91.211000	1.927674	1	1.0	0.028436	0.028436
)	82.870003	84.550003	82.550003	84.180000	2.000000	1.309998	83.370000	85.867000	90.655333	1.282158	0	1.0	-0.002141	-0.002141
)	83.120003	84.050003	82.470001	84.000000	1.580002	0.879997	83.013334	85.127000	90.163333	1.292679	0	1.0	NaN	NaN

olumns

9.Cummulative Market and Strategies Returns

Computation of cummulative market and strategy returns

In [33]:
```
trade_amzndata['Cumulative Market Returns'] = np.cumsum(trade_amzndata['Tomorrows Returns'])
trade_amzndata['Cumulative Strategy Returns'] = np.cumsum(trade_amzndata['Strategy Returns'])
```

Fig 11: Plot of cummulative market and strategy returns based on Y_prediction

In [41]:
```
plt.figure(figsize=(10,3))
plt.plot(trade_amzndata['Cumulative Market Returns'], color='r', label='Market Returns')
plt.plot(trade_amzndata['Cumulative Strategy Returns'], color='g', label='Strategy Returns')
plt.legend()
plt.show()
```

Github Pyhton Code Link Address: https://github.com/AlexiaLukangu/AI-and-ML-Coursework/blob/main/w19284242%20AI%20%26%20ML-Assessment%201%20Final%20Draft.ipynb