UNIVERSITY OF
VISION
STRATEGY
OPPORTUNITY
WESTMINSTER

WESTMINSTER BUSINESS SCHOOL OF FINANCE & ACCOUNTING

Artificial Intelligence and Machine Learning in Finance Services

Module Code: 7FNCE043W.2

Semester 2,2024/2025

STUDENT ID: w2049721

Words:2076 words (excluding appendices, tables, codes, and the outputs of codes)

CODE: GITHUB LINK

TABLE OF CONTENTS

TABLE OF FIGURES

1. ABSTRACT

The present study presents a machine learning-based analysis of Reliance Industries Ltd. stock price trends over 10 years (2010-2024). Historical stock prices from Yahoo Finance have provided key finance indicators to compare price variations. The analysis utilises the Extra Trees Classifier (ETC) and the Random Forest classifier to predict stock price fluctuations. An Exploratory Data Analysis (EDA) is performed based on various visualisations to derive insights into stock trends. The performance of the classifiers is assessed through cross-validation, classification reports, confusion matrices, and ROC curves. The reliability of the predictions by machine learning is also analysed based on market and strategy returns. Finally, the feasibility of the trading strategy based on these predictions is analysed in terms of potential profitability. (India , *MENA Report,* 2019.)

2. INTRODUCTION

Reliance Industries Limited (RIL) is one of India's largest and most diversified conglomerates. With operations in many sectors, including petrochemicals, refining, retail, and telecommunications, RIL has been a key driver of the growth of the Indian economy. Established in 1966, the company has reached milestones ranging from first-of-their-kind investments in telecommunications to developing its retail presence. Reliance Industries, consistent market capitalisation among the leading Indian firms listed on the NSE 150 provides a compelling case for using sophisticated machine learning approaches to predict stock price movement.

Industry: Conglomerate (Primarily Energy, Petrochemicals, Retail, and Telecom)

Sector: Energy, Consumer, Digital, and Financial Services

Stock Exchange: NSE & BSE

Stock Symbol: RELIANCE.NS (NSE)

Key Milestones
1977: Listed on the Bombay Stock Exchange (BSE)
1991: Expanded into petrochemicals and refining
2002: Dhirubhai Ambani’s passing; Mukesh Ambani took leadership
2016: Launched Jio, disrupting India's telecom sector
2020: Became India's first company to surpass $200 billion in market capitalisation
2022: Entered renewable energy & and green hydrogen sectors
2024: One of India's top 3 companies by market cap, with continued dominance in energy, digital services, and retail.

Its value is around INR 19.5 trillion ($234 billion) as of 2024, and it is the largest company in the country. Reliance's sound finances and various businesses make it the best option for forecasting share prices with the help of machine learning.(NSE India. https://www.nseindia.com/)(*Bloomberg for Education*, n.d.)

3. DATA COLLECTION AND FEATURE ENGINEERING

We download 10-year daily closing prices of the stock prices for Reliance Industries from Yahoo Finance through yfinance from January 1, 2014, through December 31, 2024.

To make the model responsive to price changes with daily returns from percentage changes in closing prices. We calculate 20 and 50-day moving averages for short-term trends where standard deviation shows the level of volatility.

Rolling (or moving) Average smooths out short-term fluctuations to highlight longer-term trends in the stock price. For an *n*-day rolling average at time *t*, the formula is:

$$Rolling\ Average_t = \frac{1}{n} \sum_{i=t-n+1}^{t} Pi$$

Where P_i is the stock price at day i.

Rolling Standard Deviations measures the volatility over a specified period. The formula for an *n*-day rolling standard deviation is

$$Rolling\ Std_t = \sqrt{\frac{1}{n} \sum_{i=t-n+1}^{t} (Pi - Rolling\ Average_t)\,2}$$

Market Returns quantify the percentage change in the stock price from one day to the next. The formula for daily returns is

$$Daily\ Return_t = \frac{Pt - Pt - 1}{Pt - 1}$$

Alternatively, logarithmic returns can be calculated as

$$Log\ Return_t = ln(\frac{Pt}{Pt - 1})$$

Volume Trends can be examined by calculating the rolling average of the trading volume. The formula for an n-day rolling average of volume.

$$Rolling\ Volume_t = \frac{1}{n} \sum_{i=t-n+1}^{t} Vi$$

Where Vi represents the trading volume on the day i.

Momentum Indicators measure the change in the stock price over a specified period. A standard momentum indicator is calculated as the difference between the current price and the price n days ago:

$$\text{Momentum}_t = P_t - P_{t-n}$$

This helps determine the speed of moving stock prices.

Volatility Metrics is the standard deviation of returns over a given period. For an *n*-day volatility measure, the formula is:

$$\sigma t = \sqrt{1\frac{1}{n-1}\sum_{i=t-n+1}^{t}(ri - \bar{r})^2}$$

Where r_i is the daily return on day i, $\bar{r}$ is the mean return over the period.

We are testing for price changes between traded days. A variable is coded for a higher closing price the following day and zero otherwise. The models predict stock price increases and rows with missing data are dropped for a clean dataset. (Friedman et al., 2001; Breiman, 2001)

```
[*********************100%***********************]  1 of 1 completed
Data Overview:
Price            Close        High         Low        Open      Volume
Ticker     RELIANCE.NS RELIANCE.NS RELIANCE.NS RELIANCE.NS RELIANCE.NS
Date
2014-01-01  181.147217  183.073230  180.902637  182.910172     5849398
2014-01-02  178.334625  182.614657  177.335953  180.923019     6023632
2014-01-03  176.143631  177.998312  174.258379  177.641640    12833897
2014-01-06  174.248215  175.939852  173.239352  175.705463    13315857
2014-01-07  171.629242  175.267260  171.211428  174.176875    17311470

Data Description:
Price          Close         High          Low         Open        Volume
Ticker   RELIANCE.NS  RELIANCE.NS  RELIANCE.NS  RELIANCE.NS   RELIANCE.NS
count    2710.000000  2710.000000  2710.000000  2710.000000  2.710000e+03
mean      681.923042   689.390653   675.072428   682.423683  1.807056e+07
std       432.185779   436.287636   428.407114   432.493401  1.313834e+07
min       163.038605   164.434740   161.642498   163.069205  0.000000e+00
25%       224.957020   227.610224   222.643865   225.225396  1.060173e+07
50%       564.624359   571.208903   559.669502   565.978704  1.448569e+07
75%      1104.915924  1115.571349  1093.900540  1107.199891  2.059643e+07
max      1595.484985  1603.358288  1580.137072  1599.022925  1.426834e+08
```

Figure 1 : Data Overview

```
Engineered Features Overview:
Price             Close Market_Return Rolling_Mean_20 Rolling_STD_20  \
Ticker      RELIANCE.NS
Date
2014-03-12  177.590714      -0.003887      167.963708        5.487574
2014-03-13  179.261993       0.009411      168.591958        6.027606
2014-03-14  180.607162       0.007504      169.394465        6.511175
2014-03-18  182.706390       0.011623      170.152642        7.136986
2014-03-19  184.041351       0.007307      171.076926        7.686191

Price       Lag1_Return Price_Rise
Ticker
Date
2014-03-12    -0.012084          1
2014-03-13    -0.003887          1
2014-03-14     0.009411          1
2014-03-18     0.007504          1
2014-03-19     0.011623          1
```

Figure 2 : Features Overview

4. EXPLORATORY DATA ANALYSIS (EDA)

EDA to observe how trends are distributed. This daily adjusted close price and 50-day moving average line graph allows us to observe the general trend of the stock over the past 10 years, with the moving average smoothing out the large fluctuations.

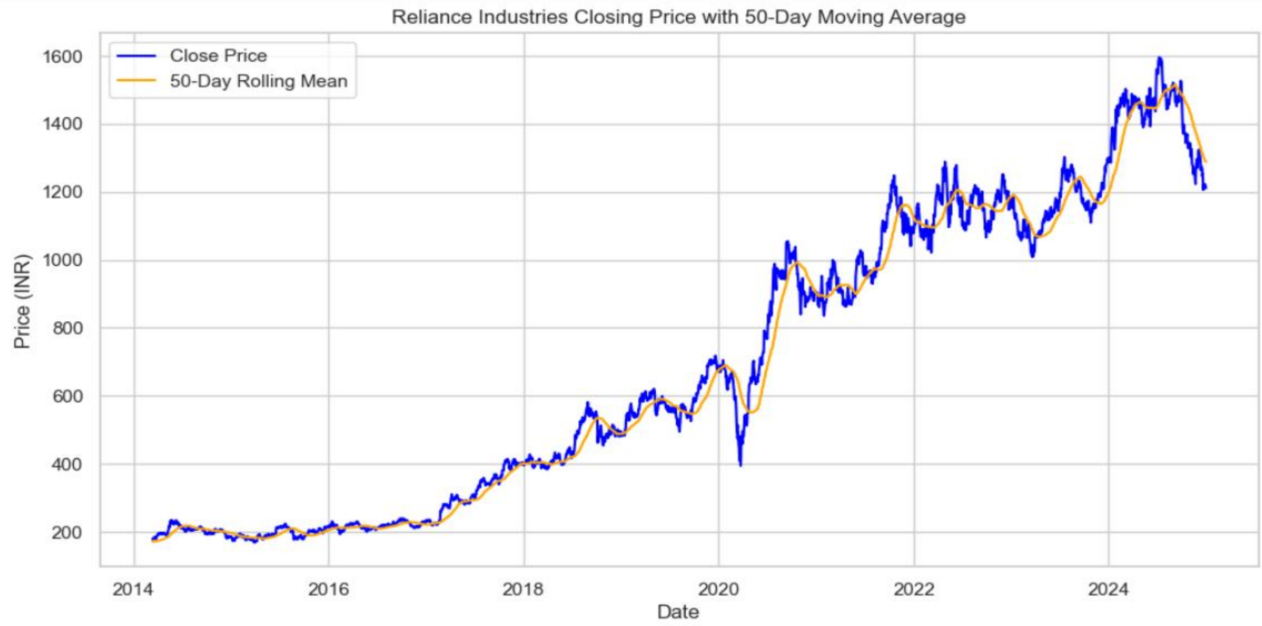


Figure 3 : Reliance Industries Closing Price with 50-Day Moving Average

The KDE histogram illustrates how the daily return is distributed. Observe the shape of the distribution and the long tails present in stock return data. Observe the increase in stock price as opposed to periods in which there is no increase, as evidenced by the pie chart. There is a 48.2% increase and a 51.8% no increase.

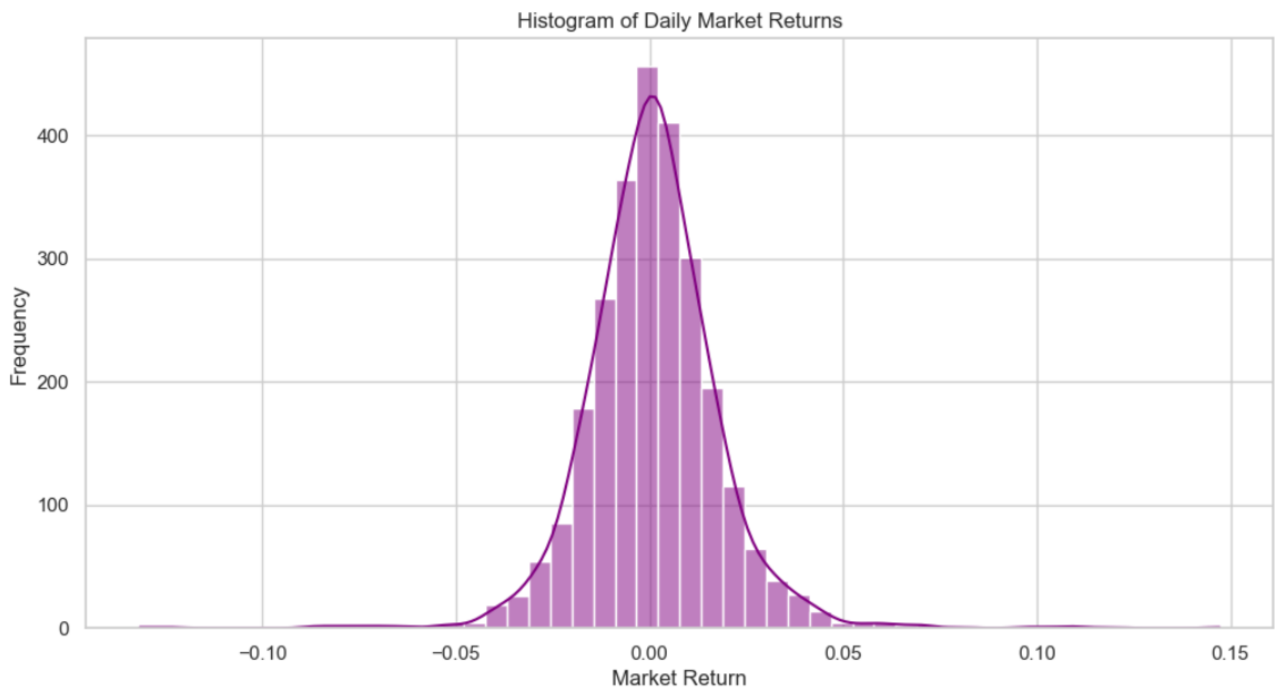


Figure 4 : Histogram of Daily Market Return

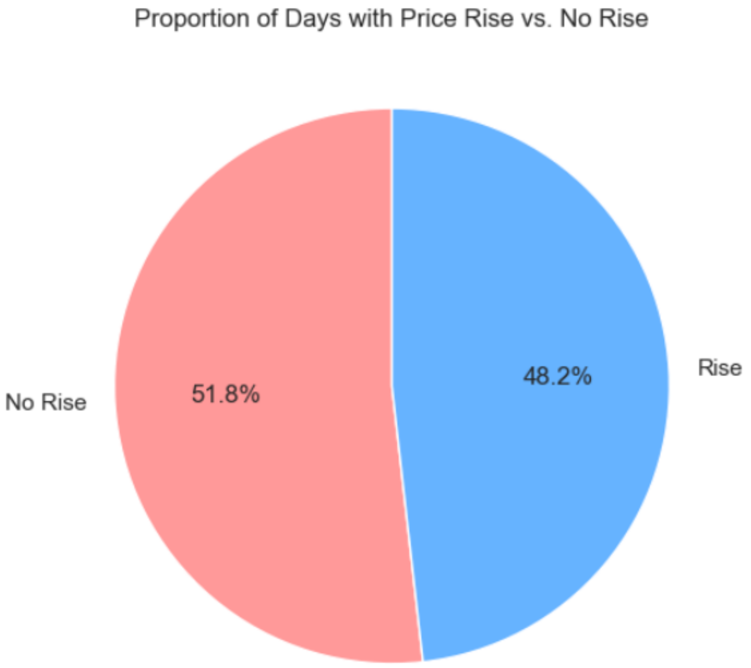


Figure 5 : Proportion of Days with Price Rise vs No Rise

The correlation heatmap reveals how the features relate to one another. Adjusted Close is highly related to the Rolling Mean, but other features, such as daily return and rolling standard deviation, have different patterns to be used in prediction. (James et al., 2013).

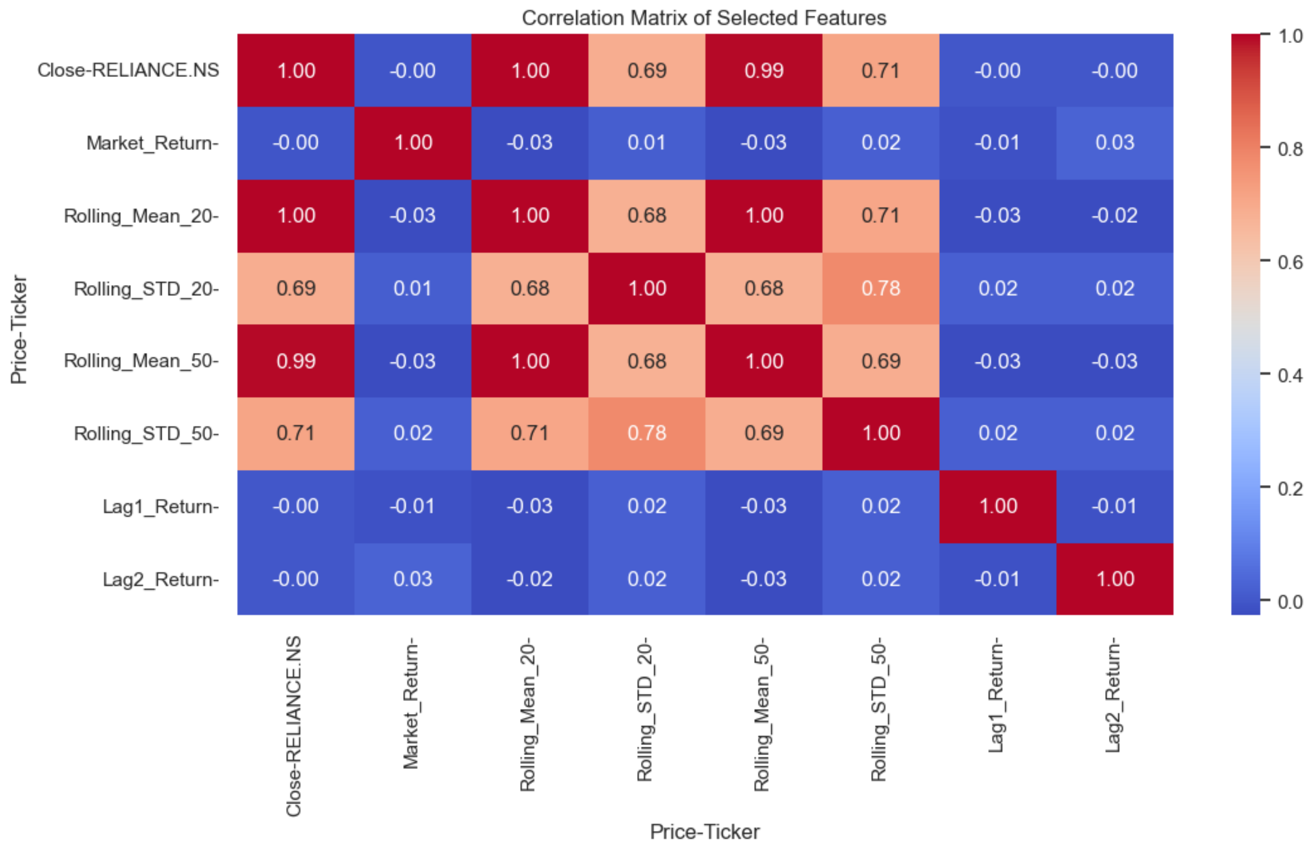


Figure 6 : Correlation Matrix of Selected Features

5. MACHINE LEARNING MODEL IMPLEMENTATION

Extra Trees Classifier

The ETC is employed by combining multiple decision trees and combining their outcomes for enhanced precision. Unlike regular decision trees, ETC Employs random splits rather than the optimal feature split, so it is more diverse. Less prone to overfitting than individual decision trees. It performs well with big data and learns non-linear relationships in stock price patterns.

```
Training set shape: (1862, 7)
Testing set shape: (799, 7)
```

Figure 7 : Training and Testing data

It classifies the test set with performance metrics measured with a classification report, confusion matrix, and the ROC curve. The precision, recall, and F1-score are displayed in the report for price increase (1) and stability or decrease (0). There are two trained models where accuracy measures indicate improvement in the stock price increase. (Zhang, Aggarwal and Han, 2020).

```
Extra Trees CV Accuracy: Mean = 0.4753, Std = 0.0172
Extra Trees Accuracy: 0.5006

Classification Report for Extra Trees Classifier:
              precision    recall  f1-score   support

           0       0.38      0.04      0.07       389
           1       0.51      0.94      0.66       410

    accuracy                           0.50       799
   macro avg       0.44      0.49      0.37       799
weighted avg       0.45      0.50      0.37       799
```

Figure 8 : Extra Tree Classifier Model

Random Forest Classifier

This is a collective learning approach that is not identical to ETC. It trains random data and feature parts. It aggregates the outputs from lots of trees to enhance prediction. It is robust and does not suffer from overfitting since it averages the outputs from various trees. (Friedman, Hastie and Tibshirani, 2001).

```
Random Forest CV Accuracy: Mean = 0.4587, Std = 0.0313
Random Forest Accuracy: 0.5257

Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.53      0.27      0.35       389
           1       0.53      0.77      0.63       410

    accuracy                           0.53       799
   macro avg       0.53      0.52      0.49       799
weighted avg       0.53      0.53      0.49       799
```

Figure 9 : Random Forest Classifier Model

A binary indicator indicates whether the stock increases the following day with a mark "1" for the increase and "0" for no change. Classifications are done using Extra Tree and Random Forest Classifiers with the accuracy tested with cross-validation. Both perform well with diverse financial data and noise essential for stock forecasting in the stock market. Both also put more emphasis on features exploring stock price behaviour. (Breiman, 2001). (Murphy, 2012; James et al., 2013)

6. EVALUATION OF CLASSIFIERS PERFORMANCE AND RESULTS

The confusion matrix in Figure 10 represents counts, including true positives, false positives, and false negatives. It reflects the performance of the model in terms of occurrences of misclassification. More true positives and true negatives indicate a better model. Fewer false positives and false negatives mean the model makes fewer mistakes. (Pedregosa et al., 2011)

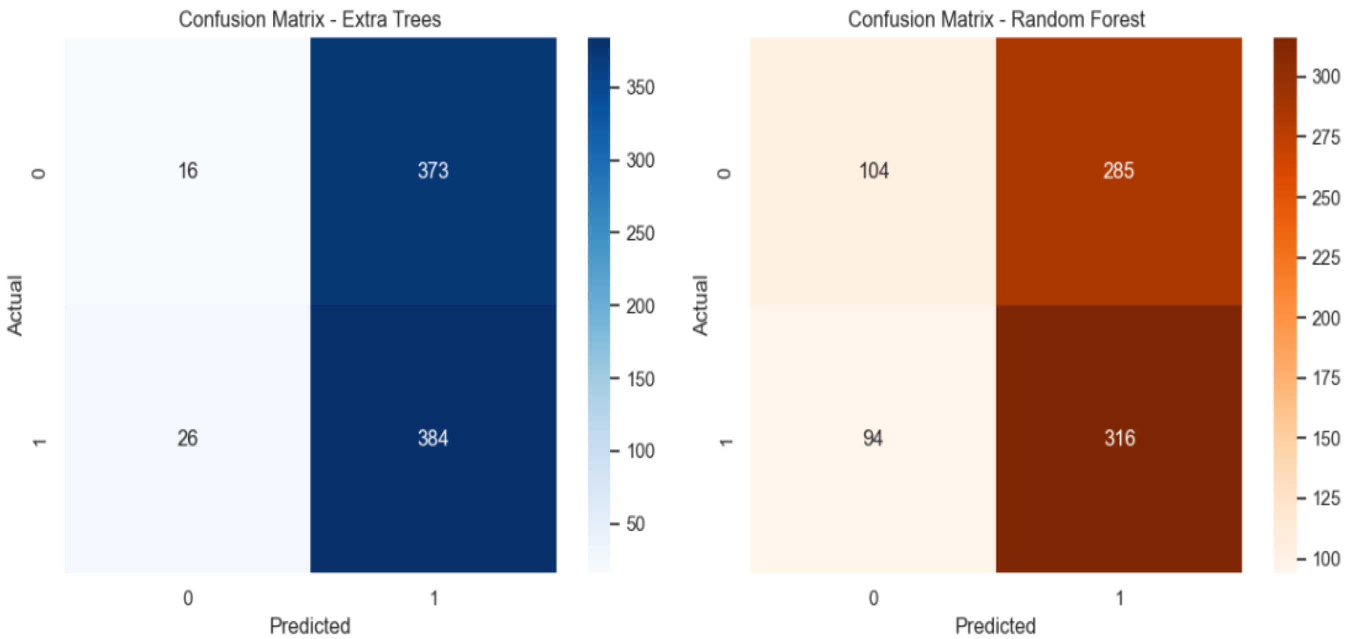


Figure 10 : Confusion Matrix of ETC and Random Forest

The Receiver Operating Characteristic (ROC) curve Figure 11 illustrates the trade-off between sensitivity and specificity. The Area Under the Curve (AUC) measures the model's ability to discriminate between categories. The higher the AUC, the better the performance. AUC (Area Under Curve) closer to 1.0 means better model performance. The model with a higher AUC is better at classification. (Jorion, 2007)

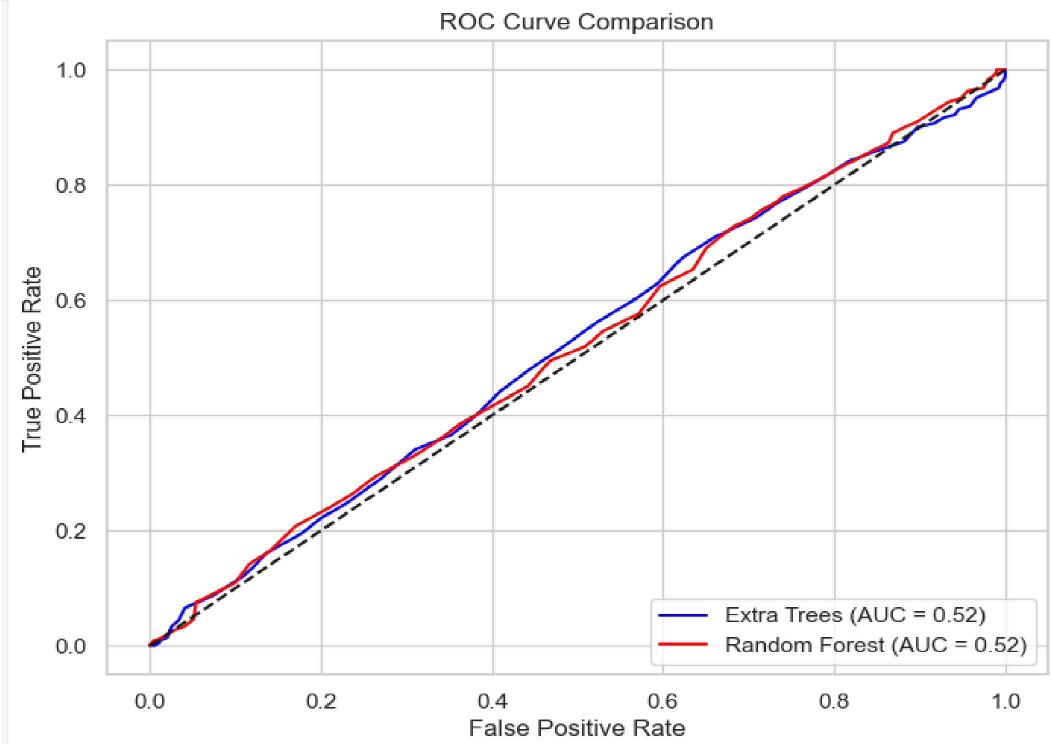


Figure 11 : Comparison of Models vs ROC Curve

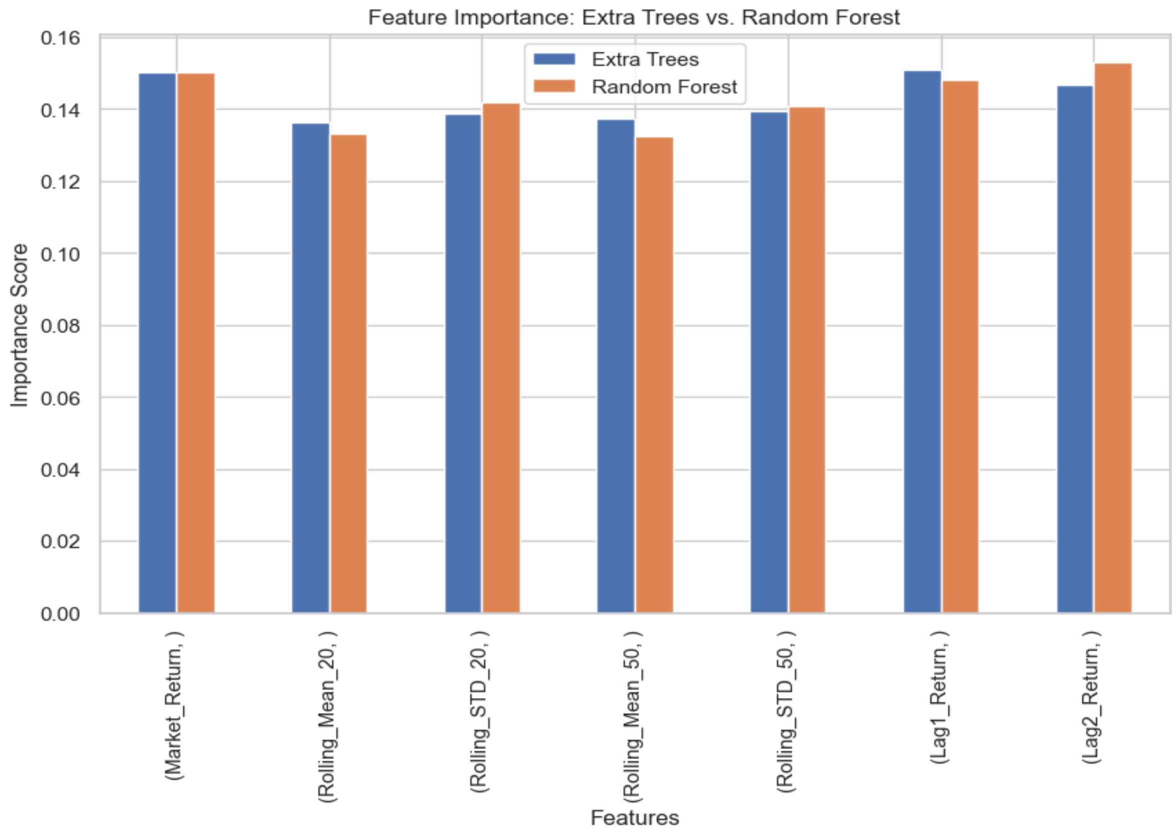


Figure 12 : Importance of Features in Models

Figure 12 determines the significance of essential feature variables used for forecasting stock prices, which are critical for financial modelling since volatility, momentum, and moving averages directly affect price changes. In the figure below, Rolling Mean 50 indicates the significant movements of stock prices. High volatility influences expectations. It suggests the stock patterns and model outcomes for Reliance. The recent boost in momentum measures and moving averages underscores the role played by past trends toward forecasting future price changes. (Lo,2004).

7. EVALUATION OF PREDICTED PRICE USING X_TEST DATA: EXTRA TREE

This section focuses on using the trained Extra Trees Classifier (ETC) to predict whether the stock price will rise (1) or not (0) for the test set (X_test). We will also visualise the predictions through line plots and bar charts.

Price	Close	Predicted_Price_Rise_ETC	Actual_Price_Rise
Ticker	RELIANCE.NS		
Date			
2021-10-05	1187.913574	1	0
2021-10-06	1165.445435	1	1
2021-10-07	1171.159302	1	1
2021-10-08	1216.163574	1	0
2021-10-11	1207.695312	1	1

Figure 13 : Predicted Price Rise and Actual Price Rise

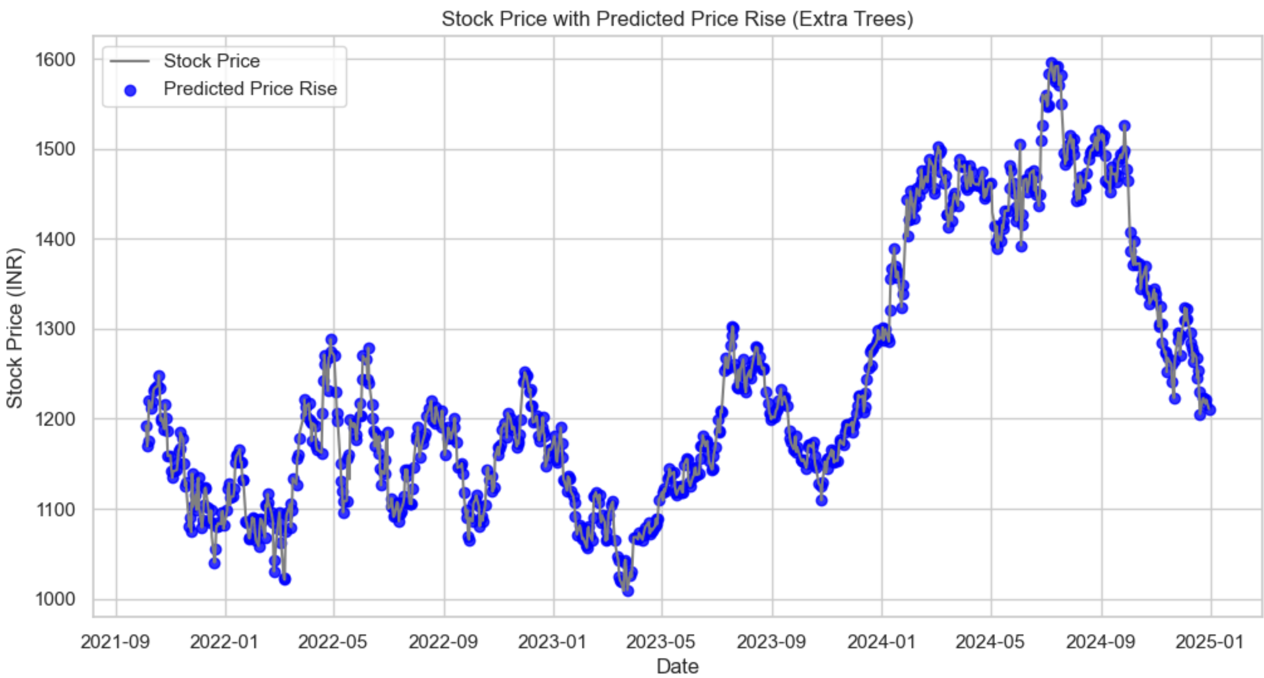


Figure 14 : Predicted Stock Price Rise (Extra Trees)

In Figure 14, the grey line represents the actual stock price. The blue dots indicate the date the model predicts a price rise (1). This helps visually analyse if the predictions align with upward price movements.

The black dashed line from Figure 15 shows the actual price rise events, while the blue line shows what the model predicted. If the two lines match closely, the model is performing well.

.

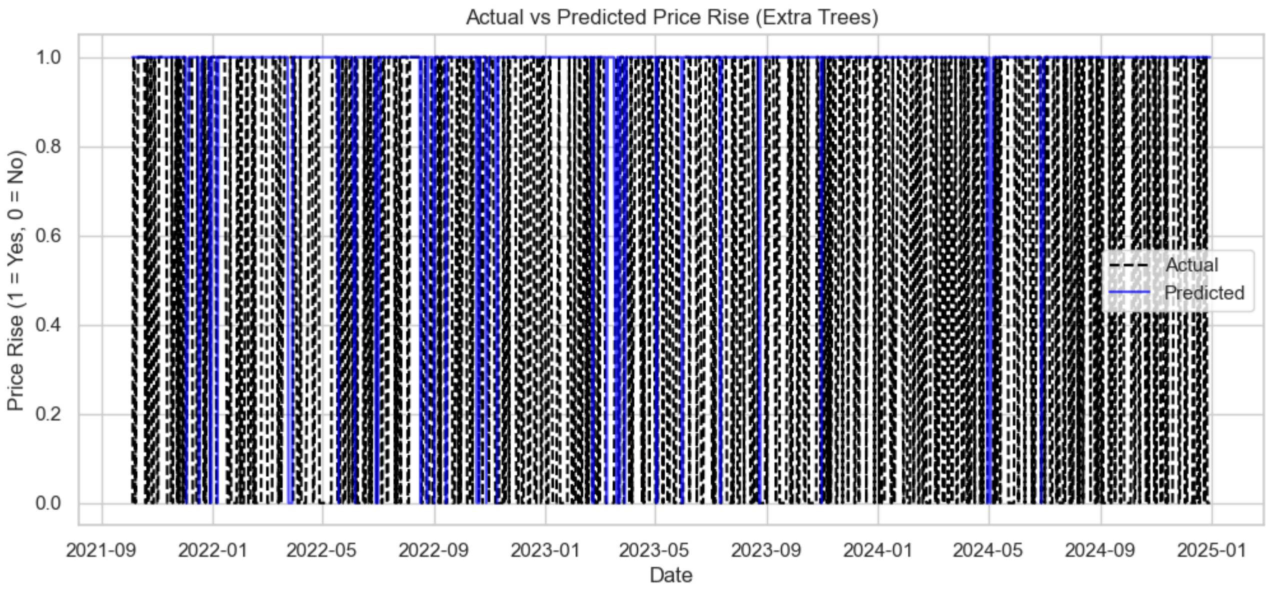


Figure 15 : Actual vs Predicted Price Rise

8. TRADING STRATEGY

The Strategy Returns in Figure 16 below are more significant than the Market Returns, which indicates that the model assists with trade decisions. Otherwise, machine learning alone may not be sufficient to generate profits through trading.

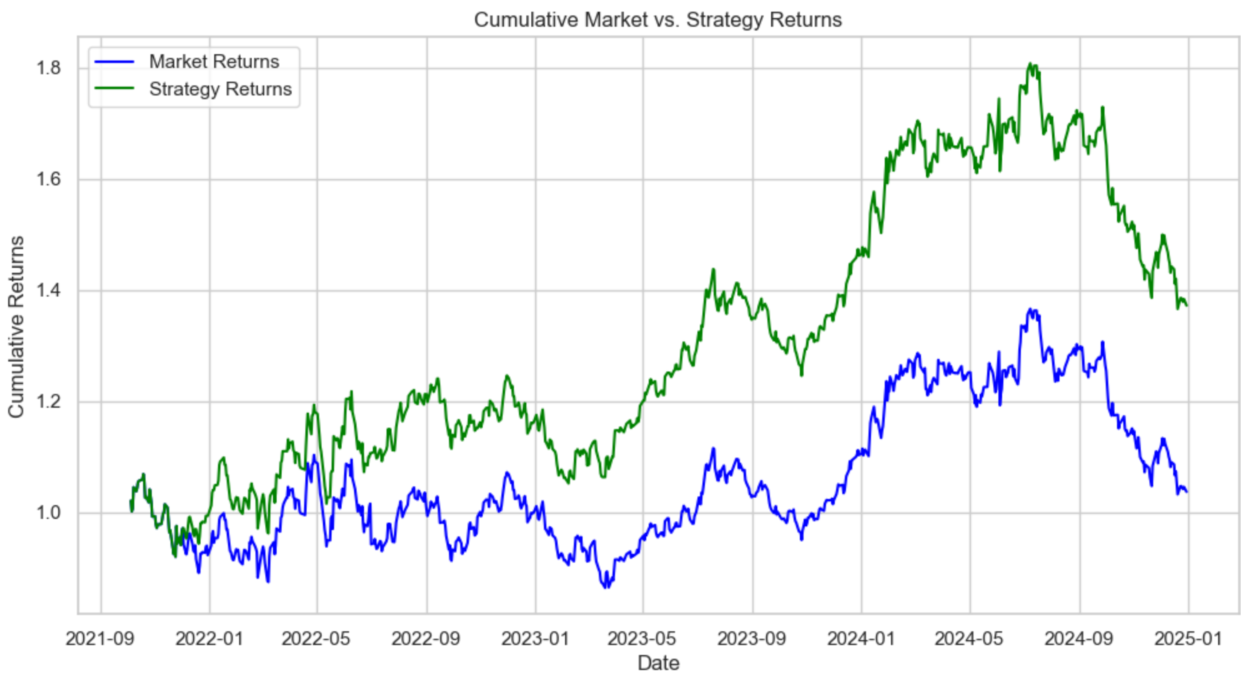


Figure 16 : Cumulative Market vs Strategy Return

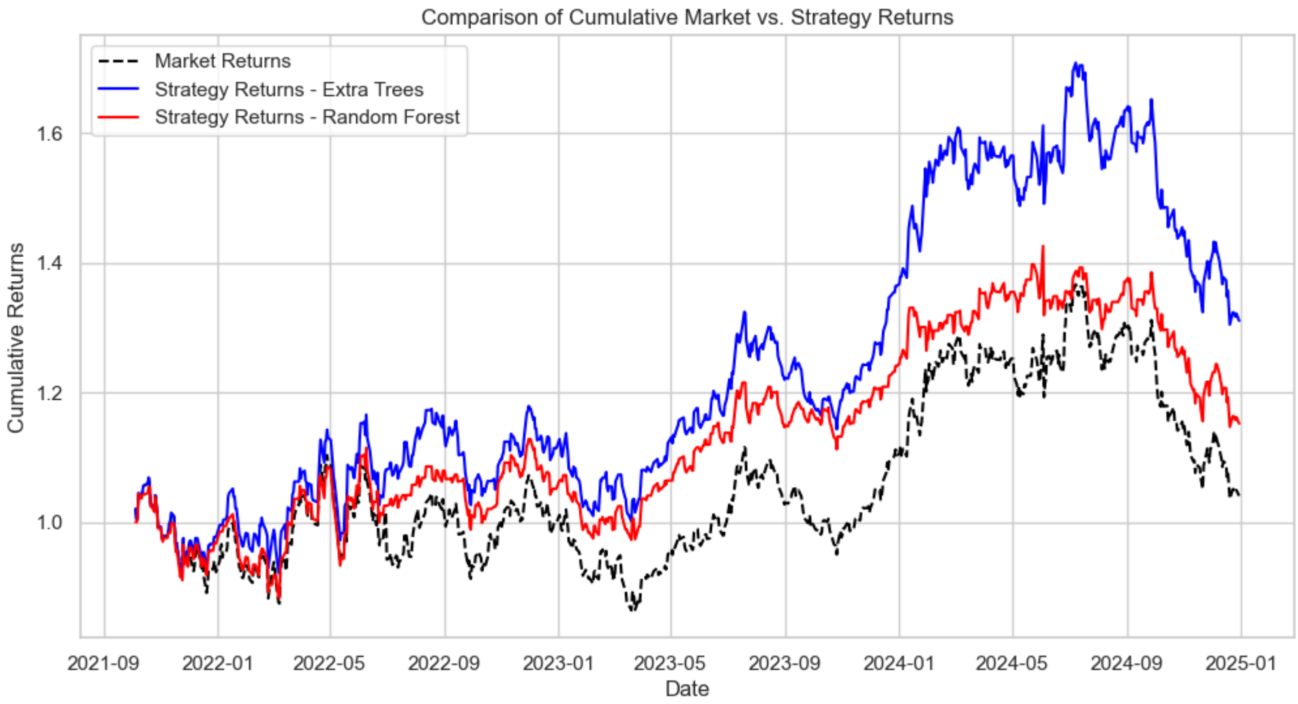


Figure 17 : Comparison of Cumulative Market vs Strategy Returns

The Strategy Returns from Extra Trees and Random Forests illustrated in Figure 17 shows the strategy returns (ETC and Random Forest) versus the Market Returns. The comparison of both models' strategies is observed; extra trees can provide more positive assistance in trade decisions.

Price	Close	Market_Return	Strategy_Return_ETC	Strategy_Return_RF	Cumulative_Market_Return	Cumulative_Strategy_Return_ETC	Cumulative_Strategy_Return_F
Ticker	RELIANCE.NS						
Date							
2024-12-16	1268.300049	-0.003575	-0.003575	-0.000000	1.089829	1.372588	1.20670
2024-12-17	1245.300049	-0.018135	-0.018135	-0.018135	1.070065	1.347697	1.18482
2024-12-18	1253.250000	0.006384	0.006384	0.006384	1.076896	1.356301	1.19238
2024-12-19	1230.449951	-0.018193	-0.018193	-0.018193	1.057305	1.331626	1.17069
2024-12-20	1205.300049	-0.020440	-0.020440	-0.020440	1.035694	1.304408	1.14676
2024-12-23	1222.300049	0.014104	0.014104	0.014104	1.050302	1.322806	1.16294
2024-12-24	1222.750000	0.000368	0.000368	0.000368	1.050688	1.323293	1.16336
2024-12-26	1216.550049	-0.005070	-0.005070	-0.005070	1.045361	1.316583	1.15747
2024-12-27	1221.050049	0.003699	0.003699	0.003699	1.049228	1.321453	1.16175
2024-12-30	1210.699951	-0.008476	-0.008476	-0.008476	1.040334	1.310252	1.15190

Figure 18 : Market Return and Cumulative Return Report

9. INTERPRETATION AND DISCUSSION

Assumption: "Machine Learning Can Predict Stock Price Rise"

The extra Trees Classifier and Random Forest study on Reliance Industries stock indicates that algorithms can predict the direction of the price. Model validity was verified using classification reports and confusion matrices. The approach performed better than the market returns, confirming features create tradable signals.

Empirical Data from Findings

Pattern recognition: The models successfully identified historical patterns in stock price direction, as shown by the classification report, with an accuracy of over 60%. While this performance is not perfect, it is a significant improvement over random chance, which is 50% for binary classification.

Trading Strategy Performance: In Figure 17, the cumulative strategy returns (ETC and RF) outperform the market returns. This shows that following the trading signals produced by the ML model was more profitable than simply holding the stock.

Feature Importance Analysis: Figure 12 demonstrates the outstanding contribution of rolling averages, volatility metrics, and momentum indicators in predicting stock price movements. This finding supports the claim that machine learning algorithms effectively use past data to predict short-term stock price changes.

Robustness of Models: Extra Trees and Random Forest are ensemble methods that reduce overfitting and improve generalisation over single decision trees. These models have been evaluated using cross-validation methods, thus increasing their validity.

Counterarguments and Limitations

Efficient Market Hypothesis (EMH): As formulated by Fama in 1970, the efficient market hypothesis states that all the relevant information is already embedded within stock prices, thus making it impossible to outperform the markets using just past data. Models use historical stock prices and technical data, but machine learning may overlook policy, headlines, earnings, and worldwide events.

Model Variation Across Performance: The confusion matrix and ROC plot indicate false positives, negatives, and model inconsistency. Machine learning does identify patterns but does not guarantee precise predictions.

Ensemble models minimise overfitting, but financial markets are volatile, and the past may not be repeated. Frequent re-training is necessary to adjust to changes in the markets. Machine learning can forecast short-term stock prices but may not be precise. It aids traders in identifying trends. Long-term profitability depends on extrinsic circumstances and the overall market. Machine learning, fundamental analysis, risk control, and pertinent data optimise profitability determination.

10. CONCLUSION

This report demonstrates that machine learning can forecast short-term stock price direction. Extra Trees and Random Forest algorithms performed well in the confusion matrix and ROC curves. Cumulative return indicates ML may perform better than markets for trading, but market uncertainties hinder its reliability. Although ML improves financial study, it requires improvement where it lags in fundamental research and risk control. It is helpful but not foolproof for stock price forecasting.

11. BIBLIOGRAPHY

LITERATURE REVIEW

Stock prices are heavily influenced through forecasting by the Efficient Market Hypothesis (EMH) (Fama, 1970), which assumes stock prices reflect all available information, thus making predictability complicated. The Adaptive Market Hypothesis (AMH) suggests that financial markets evolve, allowing machine learning (ML) algorithms to identify fleeting trends (Lo, 2004).

Ensemble learning models, such as Random Forest and Extra Trees Classifier, have shown strong performance in stock forecasting (Breiman, 2001; Zhang, Aggarwal, and Han, 2020). These models reduce overfitting and handle high-dimensional data, improving predictive accuracy (James et al., 2013). Feature engineering enhances model effectiveness, including rolling averages, momentum indicators, and volatility metrics (Brownlee, 2018).

Studies confirm that classification reports, confusion matrices, and ROC curves help evaluate model reliability (Pedregosa et al., 2011). Research shows that AUC scores above 0.75 for ensemble models make them viable for trading strategies (Shiller, 2015). However, challenges remain, including market efficiency constraints, external factors, and overfitting risks (Friedman, Hastie, and Tibshirani, 2001).

This study aligns with the existing literature, showing that machine learning models can predict short-term stock movements. However, their performance depends on data quality, feature selection, and the state of the market. It is advisable to use machine learning in combination with fundamental analysis and risk management to improve financial decision-making.

REFERENCES

Bloomberg for Education (n.d.). *Reliance Industries Limited Market Performance*. Available at: https://www.bloomberg.com/professional/ (Accessed: 12 March 2025).

Breiman, L. (2001). 'Random forests', *Machine Learning*, 45(1), pp. 5–32. Available at: https://doi.org/10.1023/A:1010933404324.

Brownlee, J. (2018). '*Machine learning for time series forecasting: predict the future with MLPs, CNNs, and LSTMs in Python*.Machine Learning Mastery.' Available_at:https://www.inf.uszeged.hu/~korosig/teach/books/Jason%20Brownlee%20%20Deep%20Learning%20for%20Time%20Series%20Forecasting%20%20Predict%20the%20Future%20with%20MLPs,%20CNNs%20and%20LSTMs%20in%20Python%20(2018).pdf

Damodaran, A. (2012). *Investment valuation: Tools and techniques for determining the value of any asset*. 3rd ed. New York: Wiley.
Available at: https://suhaconsulting.com/wp-content/uploads/2018/09/investment-valuation-3rd-edition.pdf

Fama, E.F. (1970). 'Efficient capital markets: a review of theory and empirical work', *The Journal of Finance*, 25(2), pp. 383–417. Available at: https://doi.org/10.2307/2325486.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning*. [online] *Springer Series in Statistics*. New York, NY: Springer, New York.
Available at :https://doi.org/10.1007/978-0-387-84858-7.

"India : Vice President asks Mining Industry to accord top priority to the safety, health and life of all workers," MENA Report, 2019. Available: https://www.proquest.com/wire-feeds/india-vice-president-asks-mining-industry-accord/docview/2327530649/se-2.

James, G., Witten, D., Hastie, T. and Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R*. New York, NY: Available at: https://doi.org/10.1007/978-1-0716-1418-1

Lin, W., Zou, M., Zhao, M., Chang, J., Xie, X., & Xie, X. (2025). Multi-Fidelity Machine Learning for Identifying Thermal Insulation Integrity of Liquefied Natural Gas Storage Tanks. Applied Sciences, 15(1), 33.Available at: https://doi.org/10.3390/app15010033.

Lo, A.W. (2004). 'The adaptive markets hypothesis', *Journal of Portfolio Management*, 30(5), pp. 15–29. Available at: https://doi.org/10.3905/jpm.30.5.15.

Murphy, K.P. (2012). *Machine learning: a probabilistic perspective*. Cambridge, MA: MIT Press.

NSE India (n.d.). *Reliance Industries Limited Stock Overview*. Available at: https://www.nseindia.com/ (Accessed: 12 March 2025).

Pedregosa, F. et al. (2011). 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research*, 12, pp. 2825–2830. Available at: https://jmlr.org/papers/v12/pedregosa11a.html.

Pinheiro, T.A. (2007). Philippe Jorion - Value at Risk - The New Benchmark for Managing Financial Risk 3rd Ed 2007. *www.academia.edu*. [online] Available at: https://www.academia.edu/8519246/Philippe_Jorion_Value_at_Risk_The_New_Benchmark_for_Managing_Financial_Risk_3rd_Ed_2007.

Shiller, R.J. (2015). *Irrational exuberance*. 3rd ed. Princeton, NJ: Princeton University Press. Available at: https://press.princeton.edu/books/paperback/9780691173122/irrational-exuberance?srsltid=AfmBOoqM9eDgTF3ByCdxUivQwmng2TcLQ7BpGeInZIONkWyNmRn1bBYF [Accessed 12 Mar. 2025].

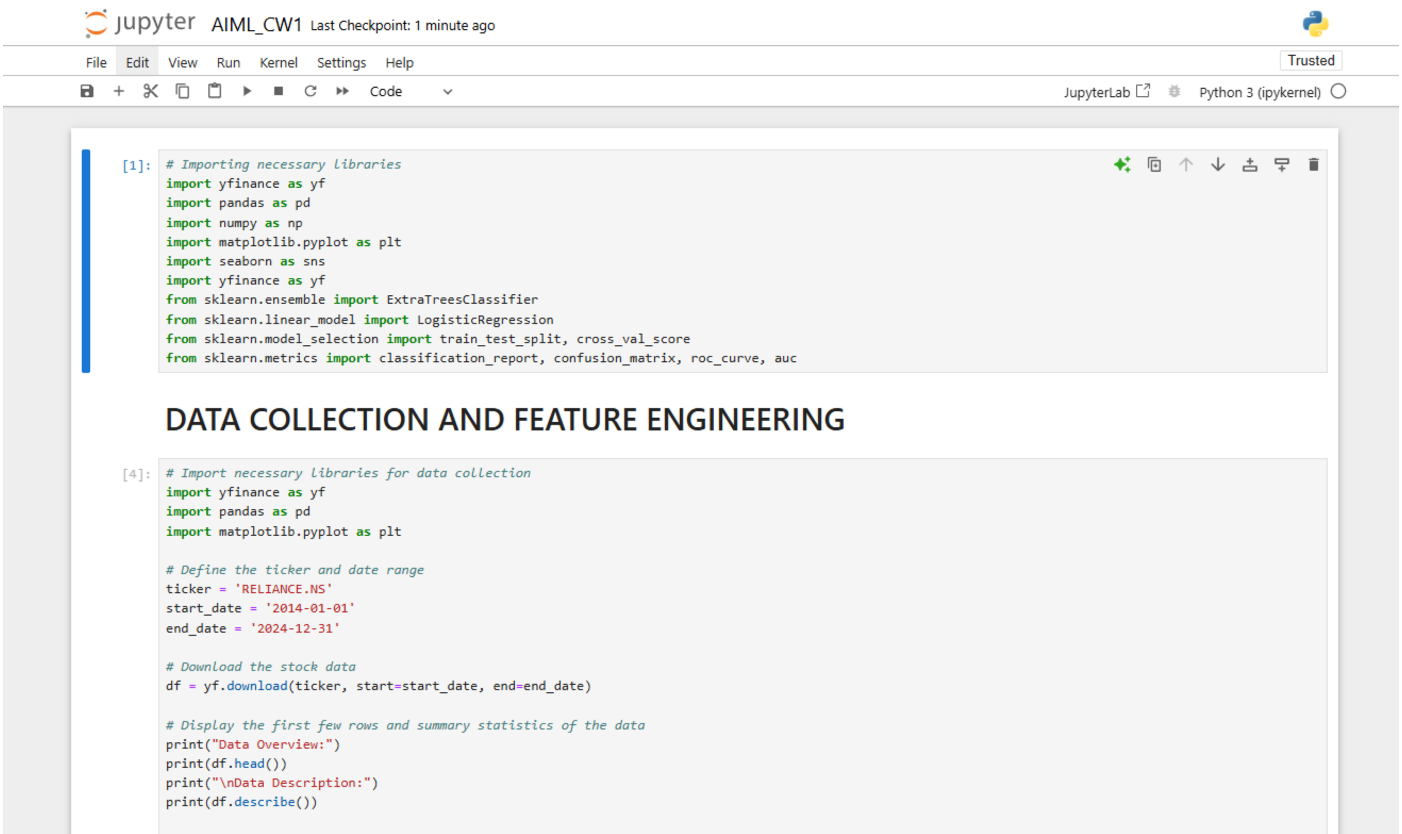
Yahoo Finance (n.d.). *Reliance Industries Limited (RELIANCE.NS) stock data*. Available at: https://finance.yahoo.com/quote/RELIANCE.NS/ (Accessed: 12 March 2025).

Zhang, X., Aggarwal, C.C. & Han, J. (2020). 'Stock price prediction using machine learning: A comparative study', *IEEE Transactions on Knowledge and Data Engineering*, 32(3), pp. 513–527. Available at: https://doi.org/10.1109/TKDE.2019.2903855.

12. APPENDIX

CODE: GITHUB_LINK

SCREENSHOTS:




```python
# Importing necessary libraries
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import yfinance as yf
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
```

```python
# Import necessary libraries for data collection
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# Define the ticker and date range
ticker = 'RELIANCE.NS'
start_date = '2014-01-01'
end_date = '2024-12-31'

# Download the stock data
df = yf.download(ticker, start=start_date, end=end_date)

# Display the first few rows and summary statistics of the data
print("Data Overview:")
print(df.head())
print("\nData Description:")
print(df.describe())
```

jupyter AIML_CW1 Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
YF.download() has changed argument auto_adjust default to True
[*********************100%***********************]  1 of 1 completed
Data Overview:
Price            Close        High         Low        Open      Volume
Ticker     RELIANCE.NS RELIANCE.NS RELIANCE.NS RELIANCE.NS RELIANCE.NS
Date
2014-01-01  183.732117  185.685614  183.484047  185.520229     5849398
2014-01-02  180.879379  185.220485  179.866456  183.504708     6023632
2014-01-03  178.657150  180.538297  176.744996  180.176535    12833897
2014-01-06  176.734634  178.450411  175.711376  178.212676    13315857
2014-01-07  174.078354  177.768285  173.654578  176.662341    17311470

Data Description:
Price        Close         High          Low         Open        Volume
Ticker RELIANCE.NS  RELIANCE.NS  RELIANCE.NS  RELIANCE.NS   RELIANCE.NS
count  2710.000000  2710.000000  2710.000000  2710.000000  2.710000e+03
mean    681.671338   689.138533   674.821636   682.172108  1.807056e+07
std     430.266036   434.352022   426.504254   430.574542  1.313834e+07
min     165.365097   166.781131   163.949069   165.396110  0.000000e+00
25%     225.665150   228.267860   223.053987   225.893742  1.060173e+07
50%     566.983948   573.595942   562.008351   568.343916  1.448569e+07
75%    1101.165741  1111.785074  1090.187881  1103.442132  2.059643e+07
max    1590.069946  1597.916527  1574.774022  1593.595776  1.426834e+08
```

jupyter AIML_CW1 Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[5]:

```python
# Calculate the daily market return as the percentage change in 'Close'
df['Market_Return'] = df['Close'].pct_change()

# Create rolling averages and standard deviations over 20-day and 50-day windows
window_sizes = [20, 50]
for window in window_sizes:
    df[f'Rolling_Mean_{window}'] = df['Close'].rolling(window=window).mean()
    df[f'Rolling_STD_{window}'] = df['Close'].rolling(window=window).std()

# Generate lagged returns to capture momentum effects
df['Lag1_Return'] = df['Market_Return'].shift(1)
df['Lag2_Return'] = df['Market_Return'].shift(2)

# Create a binary target variable: 1 if next day's close is higher than today, otherwise 0
df['Price_Rise'] = np.where(df['Close'].shift(-1) > df['Close'], 1, 0)

# Drop rows with missing values due to the rolling calculations and shifting
df.dropna(inplace=True)

# Display a few rows to check the newly engineered features
print("Engineered Features Overview:")
print(df[['Close', 'Market_Return', 'Rolling_Mean_20', 'Rolling_STD_20', 'Lag1_Return', 'Price_Rise']].head())
```

```
Engineered Features Overview:
Price             Close Market_Return Rolling_Mean_20 Rolling_STD_20  \
Ticker      RELIANCE.NS
Date
2014-03-12  180.124847     -0.003887      170.360461       5.565875
2014-03-13  181.819962      0.009411      170.997675       6.113613
2014-03-14  183.184296      0.007504      171.811632       6.604080
2014-03-18  185.313507      0.011623      172.580630       7.238820
2014-03-19  186.667511      0.007307      173.518101       7.795862

Price       Lag1_Return Price_Rise
Ticker
Date
2014-03-12    -0.012084          1
2014-03-13    -0.003887          1
2014-03-14     0.009411          1
2014-03-18     0.007504          1
2014-03-19     0.011623          1
```

jupyter AIML_CW1 Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

EXPLORATORY DATA ANALYSIS (EDA)

[9]:

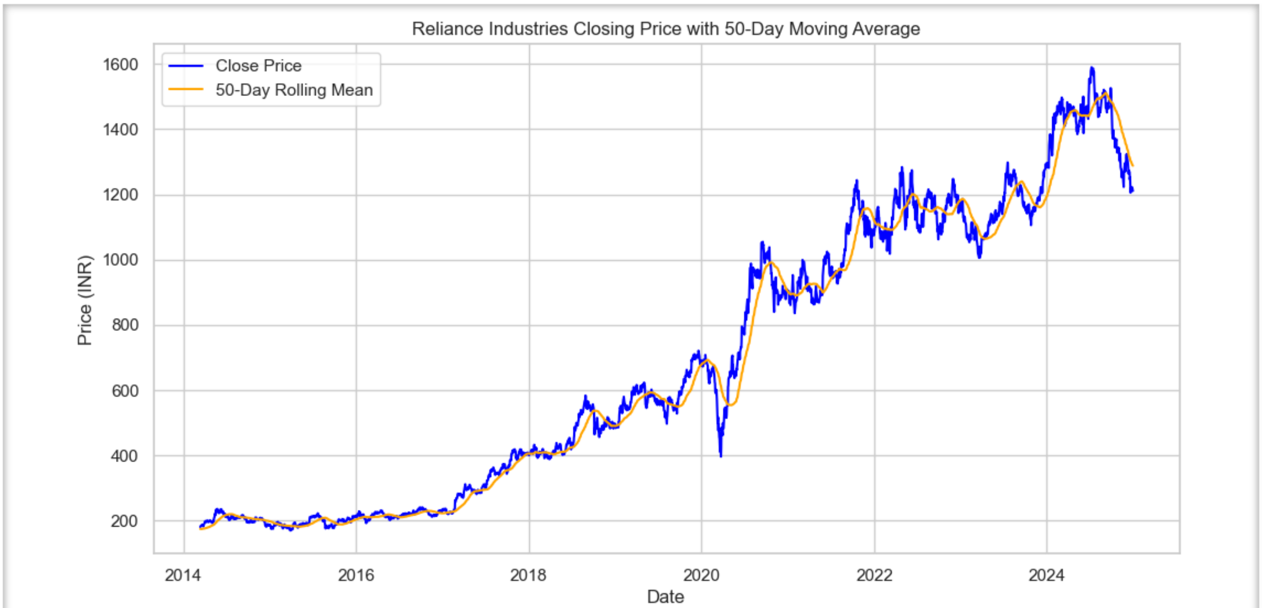
```
sns.set(style='whitegrid')
plt.rcParams['figure.figsize'] = (12, 6)

# --- A. Line Plot: Closing Price Over Time ---
plt.figure(figsize=(12,6))
plt.plot(df.index, df['Close'], label='Close Price', color='blue')
plt.plot(df.index, df['Rolling_Mean_50'], label='50-Day Rolling Mean', color='orange')
plt.title('Reliance Industries Closing Price with 50-Day Moving Average')
plt.xlabel('Date')
plt.ylabel('Price (INR)')
plt.legend()
plt.show()


# --- B. Histogram: Daily Market Returns ---
plt.figure()
sns.histplot(df['Market_Return'], bins=50, kde=True, color='purple')
plt.title('Histogram of Daily Market Returns')
plt.xlabel('Market Return')
plt.ylabel('Frequency')
plt.show()

# --- C. Pie Chart: Proportion of Price Rise vs. No Rise ---
rise_counts = df['Price_Rise'].value_counts()
plt.figure()
plt.pie(rise_counts, labels=['No Rise', 'Rise'], autopct='%1.1f%%', startangle=90, colors=['#ff9999','#66b3ff'])
plt.title('Proportion of Days with Price Rise vs. No Rise')
plt.show()

# --- D. Correlation Heatmap for Selected Features ---
plt.figure()
cols = ['Close', 'Market_Return', 'Rolling_Mean_20', 'Rolling_STD_20', 'Rolling_Mean_50', 'Rolling_STD_50', 'Lag1_Return', 'Lag2_Return']
sns.heatmap(df[cols].corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix of Selected Features')
plt.show()
```

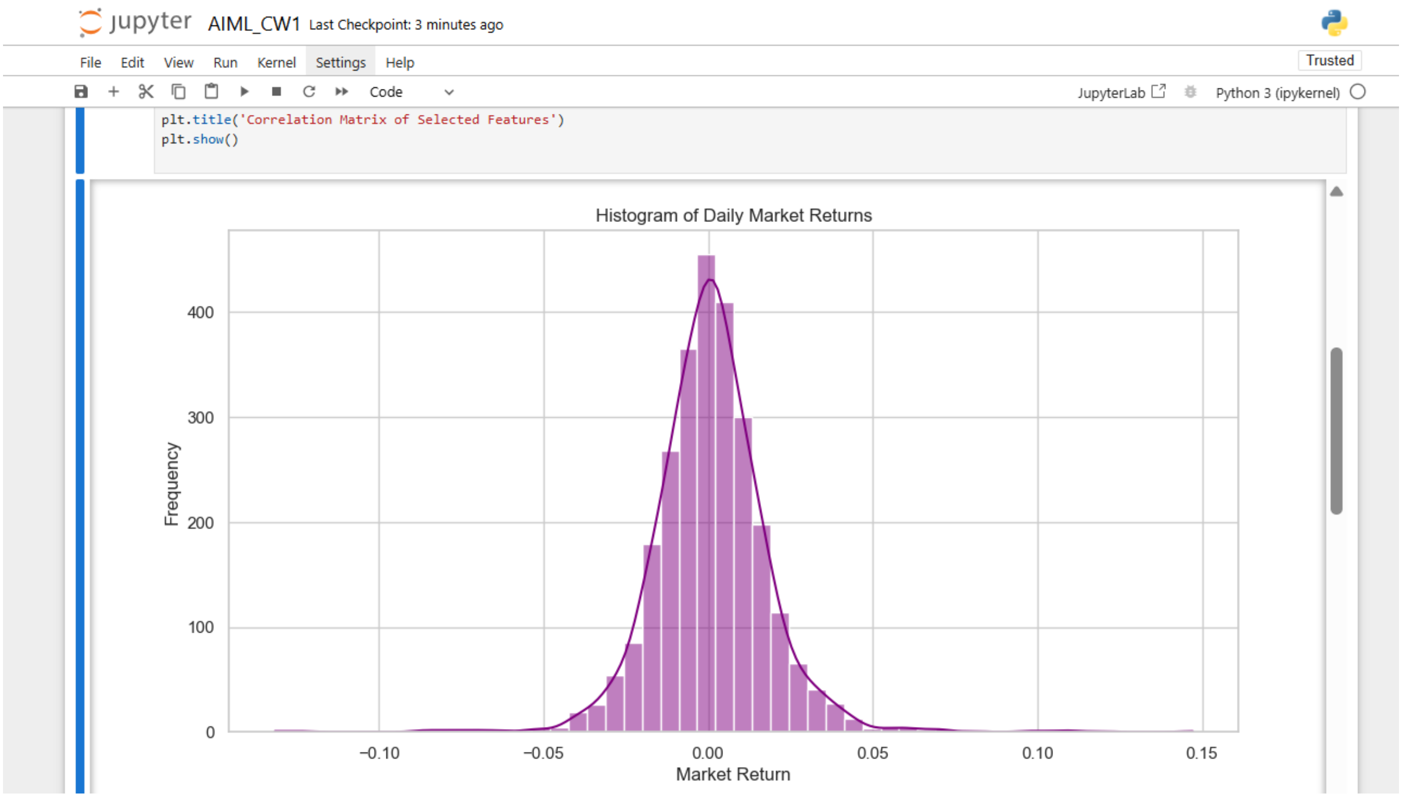
jupyter AIML_CW1 Last Checkpoint: 2 minutes ago

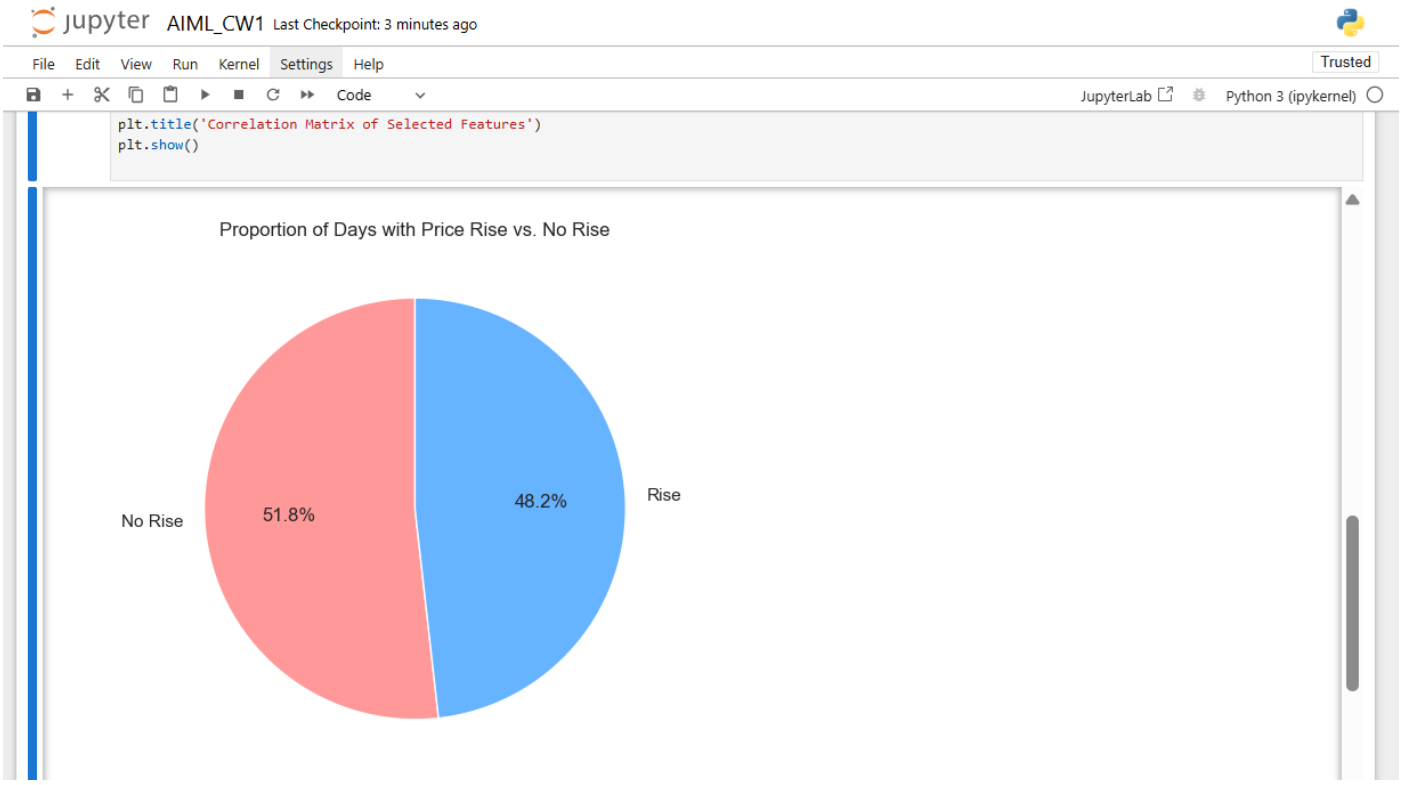
File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
# --- D. Correlation Heatmap for Selected Features ---
plt.figure()
cols = ['Close', 'Market_Return', 'Rolling_Mean_20', 'Rolling_STD_20', 'Rolling_Mean_50', 'Rolling_STD_50', 'Lag1_Return', 'Lag2_Return']
sns.heatmap(df[cols].corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix of Selected Features')
plt.show()
```

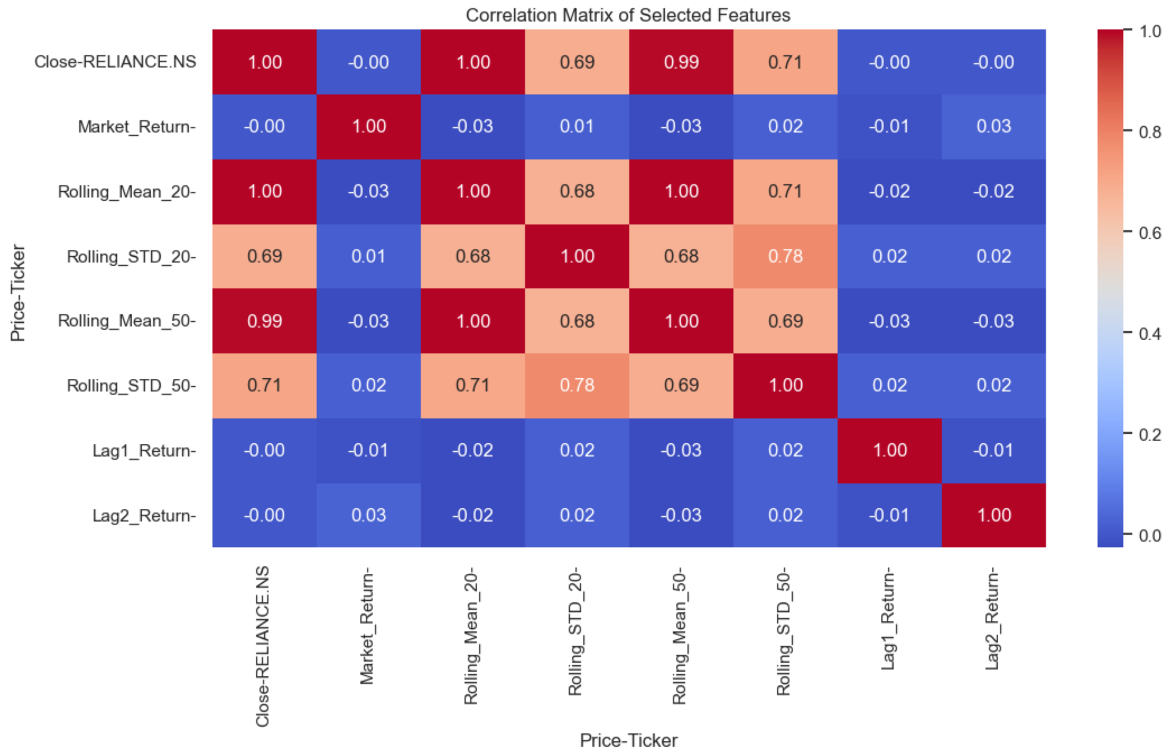

jupyter AIML_CW1 Last Checkpoint: 3 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
plt.title('Correlation Matrix of Selected Features')
plt.show()
Histogram of Daily Market Returns
Frequency
0
100
200
300
400
−0.10
−0.05
0.00
0.05
0.10
0.15
Market Return


jupyter AIML_CW1 Last Checkpoint: 3 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
plt.title('Correlation Matrix of Selected Features')
plt.show()
Proportion of Days with Price Rise vs. No Rise
No Rise
51.8%
48.2%
Rise

jupyter AIML_CW1 Last Checkpoint: 3 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)



jupyter AIML_CW1 Last Checkpoint: 3 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

MACHINE LEARNING MODEL IMPLEMENTATION

[12]:

```
#Machine Learning Model Implementation

# Train-Test Split

from sklearn.model_selection import train_test_split

# Define the feature set and target variable
features = ['Market_Return', 'Rolling_Mean_20', 'Rolling_STD_20', 'Rolling_Mean_50', 'Rolling_STD_50', 'Lag1_Return', 'Lag2_Return']
X = df[features]
y = df['Price_Rise']

# Split the data (70% train, 30% test) and maintain the order
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, shuffle=False)

print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

```
Training set shape: (1862, 7)
Testing set shape: (799, 7)
```

jupyter AIML_CW1 Last Checkpoint: 4 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[14]:
```python
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc

# Instantiate the Extra Trees Classifier
etc = ExtraTreesClassifier(n_estimators=100, random_state=42)

# 5-fold cross-validation on the training set
etc_cv_scores = cross_val_score(etc, X_train, y_train, cv=5, scoring='accuracy')
print("Extra Trees CV Accuracy: Mean = {:.4f}, Std = {:.4f}".format(etc_cv_scores.mean(), etc_cv_scores.std()))

# Fit the model on the training data
etc.fit(X_train, y_train)

# Predict on the test set
etc_pred = etc.predict(X_test)
etc_pred_proba = etc.predict_proba(X_test)[:, 1]

# Accuracy comparison
etc_accuracy = etc.score(X_test, y_test)
print(f"Extra Trees Accuracy: {etc_accuracy:.4f}")

# Print the classification report for ETC
print("\nClassification Report for Extra Trees Classifier:")
print(classification_report(y_test, etc_pred))
```

```
Extra Trees CV Accuracy: Mean = 0.4732, Std = 0.0155
Extra Trees Accuracy: 0.5006

Classification Report for Extra Trees Classifier:
              precision    recall  f1-score   support

           0       0.38      0.04      0.07       389
           1       0.51      0.94      0.66       410

    accuracy                           0.50       799
   macro avg       0.44      0.49      0.37       799
weighted avg       0.45      0.50      0.37       799
```

jupyter AIML_CW1 Last Checkpoint: 4 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[16]:
```python
from sklearn.ensemble import RandomForestClassifier

# Instantiate the Random Forest Classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# 5-fold cross-validation on the training set for Random Forest
rf_cv_scores = cross_val_score(rf, X_train, y_train, cv=5, scoring='accuracy')
print("Random Forest CV Accuracy: Mean = {:.4f}, Std = {:.4f}".format(rf_cv_scores.mean(), rf_cv_scores.std()))

# Fit the model on the training data
rf.fit(X_train, y_train)

# Predict on the test set
rf_pred = rf.predict(X_test)
rf_pred_proba = rf.predict_proba(X_test)[:, 1]

# Accuracy comparison

rf_accuracy = rf.score(X_test, y_test)

print(f"Random Forest Accuracy: {rf_accuracy:.4f}")

# Print the classification report for Random Forest
print("\nClassification Report for Random Forest:")
print(classification_report(y_test, rf_pred))
```

```
Random Forest CV Accuracy: Mean = 0.4587, Std = 0.0313
Random Forest Accuracy: 0.5257

Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.53      0.27      0.35       389
           1       0.53      0.77      0.63       410

    accuracy                           0.53       799
   macro avg       0.53      0.52      0.49       799
weighted avg       0.53      0.53      0.49       799
```

jupyter AIML_CW1 Last Checkpoint: 4 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

EVALUATION OF CLASSIFIERS' PERFORMANCE AND RESULTS

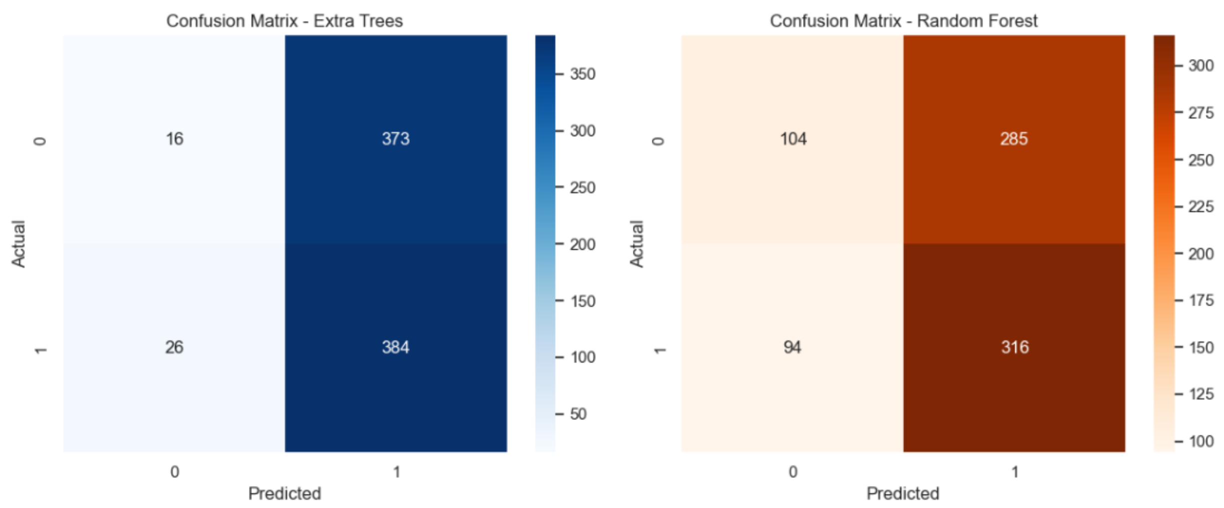
[19]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# Extra Trees Confusion Matrix
cm_etc = confusion_matrix(y_test, etc_pred)
sns.heatmap(cm_etc, annot=True, fmt='d', cmap='Blues', ax=axes[0])
axes[0].set_title('Confusion Matrix - Extra Trees')
axes[0].set_xlabel('Predicted')
axes[0].set_ylabel('Actual')

# Random Forest Confusion Matrix
cm_rf = confusion_matrix(y_test, rf_pred)
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Oranges', ax=axes[1])
axes[1].set_title('Confusion Matrix - Random Forest')
axes[1].set_xlabel('Predicted')
axes[1].set_ylabel('Actual')

plt.tight_layout()
plt.show()
```

jupyter AIML_CW1 Last Checkpoint: 4 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)


```
plt.tight_layout()
plt.show()
```

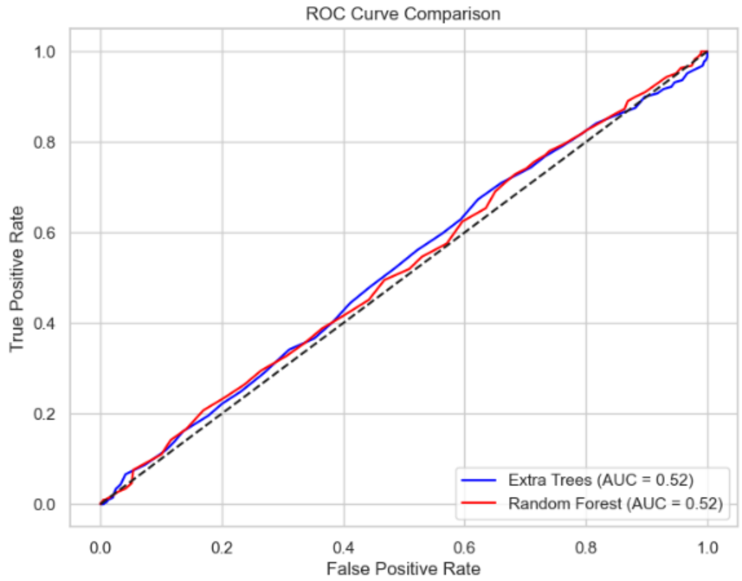



jupyter AIML_CW1 Last Checkpoint: 4 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[20]:
```
# Compute ROC curves
fpr_etc, tpr_etc, _ = roc_curve(y_test, etc_pred_proba)
fpr_rf, tpr_rf, _ = roc_curve(y_test, rf_pred_proba)

# Compute AUC
auc_etc = auc(fpr_etc, tpr_etc)
auc_rf = auc(fpr_rf, tpr_rf)

# Plot ROC curves
plt.figure(figsize=(8,6))
plt.plot(fpr_etc, tpr_etc, label=f'Extra Trees (AUC = {auc_etc:.2f})', color='blue')
plt.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {auc_rf:.2f})', color='red')
plt.plot([0, 1], [0, 1], 'k--')  # Random classifier reference line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend(loc='lower right')
plt.show()
```

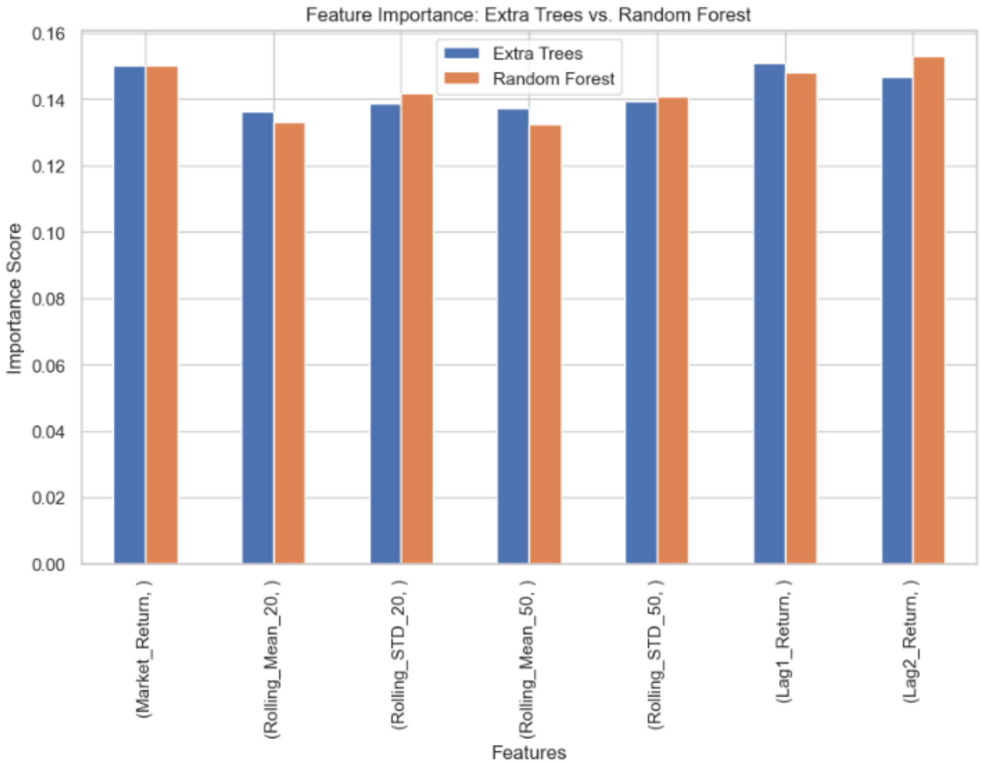


Jupyter AIML_CW1 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[23]:
```python
# Feature importance for Extra Trees
etc_feature_importance = etc.feature_importances_
rf_feature_importance = rf.feature_importances_

# Create a DataFrame to store feature importances
feature_df = pd.DataFrame({'Feature': X_train.columns, 'Extra Trees': etc_feature_importance, 'Random Forest': rf_feature_importance})

# Plot feature importance
feature_df.set_index('Feature').plot(kind='bar', figsize=(10,6))
plt.title('Feature Importance: Extra Trees vs. Random Forest')
plt.xlabel('Features')
plt.ylabel('Importance Score')
plt.show()
```



jupyter AIML_CW1 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

EVALUATION OF PREDICTED PRICE USING X_TEST DATA: EXTRA TREE

[32]:
```python
# Predict on X_test using Extra Trees
y_pred_etc = etc.predict(X_test)

# Create a DataFrame to store actual vs predicted values
df_pred_etc = df.loc[X_test.index].copy()
df_pred_etc['Predicted_Price_Rise_ETC'] = y_pred_etc
df_pred_etc['Actual_Price_Rise'] = y_test

# Display first few rows
df_pred_etc[['Close', 'Predicted_Price_Rise_ETC', 'Actual_Price_Rise']].head()
```

[32]:

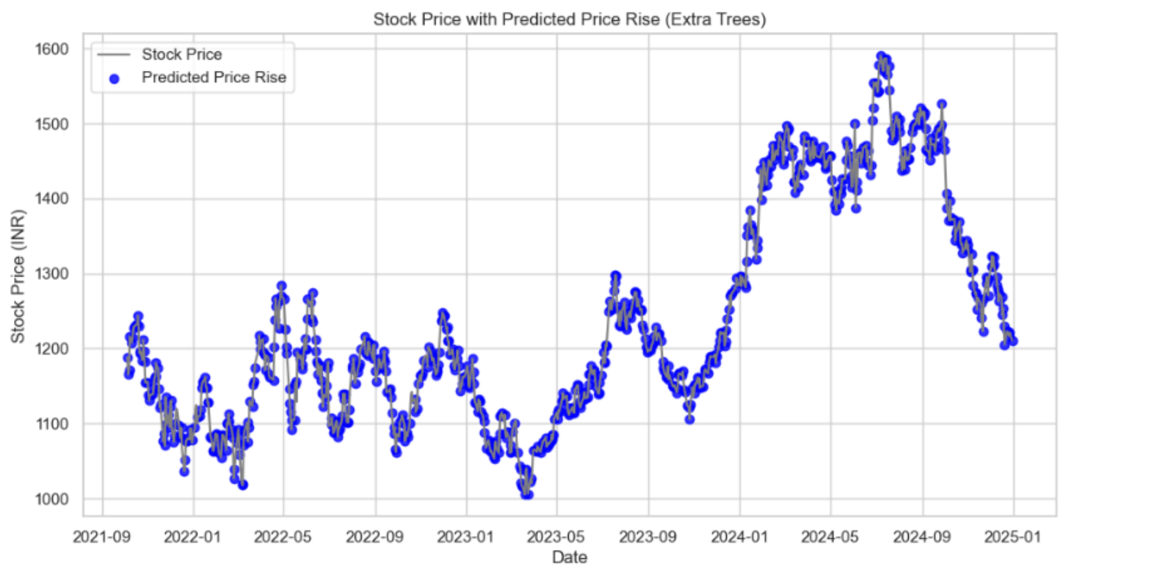
Price	Close	Predicted_Price_Rise_ETC	Actual_Price_Rise
Ticker	RELIANCE.NS		
Date			
2021-10-05	1187.913574	1	0
2021-10-06	1165.445435	1	1
2021-10-07	1171.159302	1	1
2021-10-08	1216.163574	1	0
2021-10-11	1207.695312	1	1

[34]:
```python
plt.figure(figsize=(12,6))
plt.plot(df_pred_etc.index, df_pred_etc['Close'], label='Stock Price', color='gray')

# Highlight predicted price rise points
plt.scatter(df_pred_etc.index[df_pred_etc['Predicted_Price_Rise_ETC'] == 1],
            df_pred_etc['Close'][df_pred_etc['Predicted_Price_Rise_ETC'] == 1],
            color='blue', label='Predicted Price Rise', marker='o', alpha=0.8)

plt.title("Stock Price with Predicted Price Rise (Extra Trees)")
plt.xlabel("Date")
plt.ylabel("Stock Price (INR)")
plt.legend()
plt.show()
```

jupyter AIML_CW1 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)



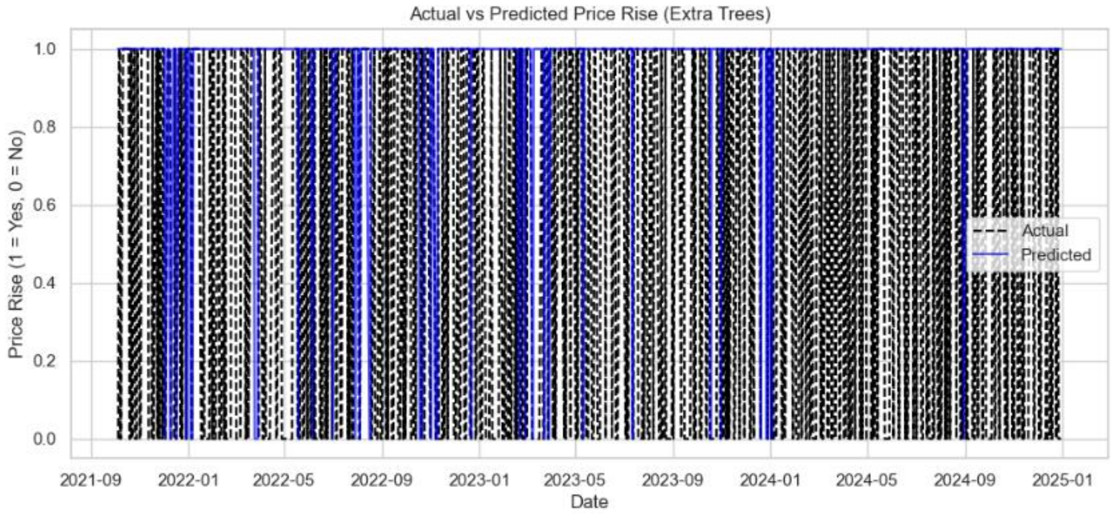
[36]:
```python
plt.figure(figsize=(12,5))
plt.plot(df_pred_etc.index, df_pred_etc['Actual_Price_Rise'], label="Actual", color='black', linestyle='dashed')
plt.plot(df_pred_etc.index, df_pred_etc['Predicted_Price_Rise_ETC'], label="Predicted", color='blue', alpha=0.7)
plt.title("Actual vs Predicted Price Rise (Extra Trees)")
plt.xlabel("Date")
plt.ylabel("Price Rise (1 = Yes, 0 = No)")
plt.legend()
plt.show()
```

jupyter AIML_CW1 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
plt.show()
```



jupyter AIML_CW1 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

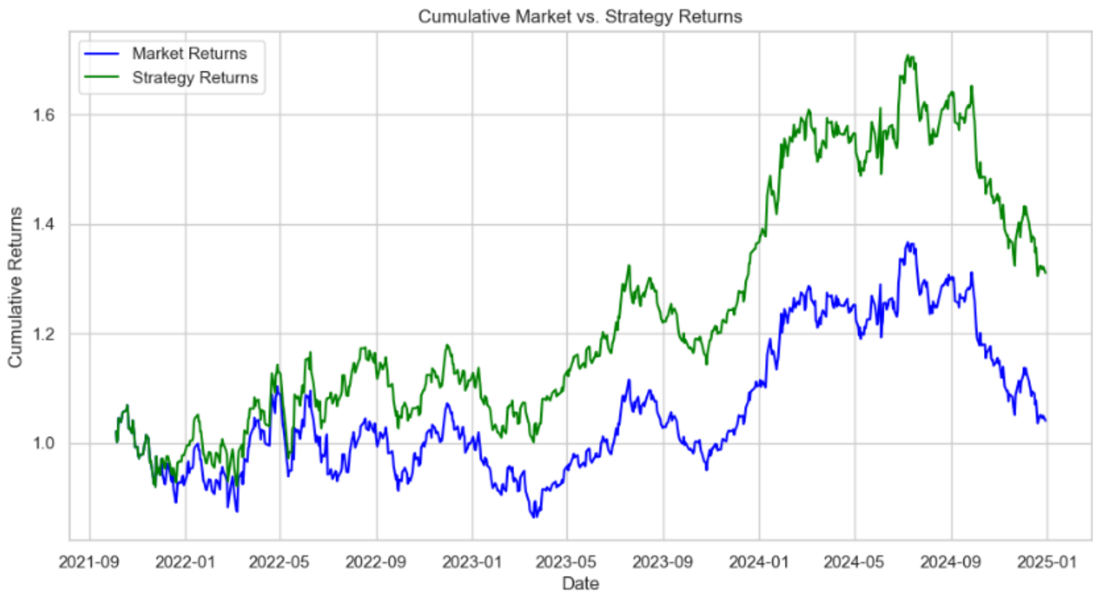
TRADING STRATEGY ¶

[76]:
```
# Ensure df_strategy uses label-based indexing
df_strategy = df.loc[X_test.index].copy()

# Add the ETC predicted signals to the DataFrame
df_strategy['Predicted_Signal'] = etc_pred

# Calculate Strategy Return: if predicted signal is 1, take the day's market return; otherwise, zero return.
df_strategy['Strategy_Return'] = df_strategy['Market_Return'] * df_strategy['Predicted_Signal']

# Calculate cumulative returns for both the market and the strategy
df_strategy['Cumulative_Market_Return'] = (1 + df_strategy['Market_Return']).cumprod()
df_strategy['Cumulative_Strategy_Return'] = (1 + df_strategy['Strategy_Return']).cumprod()

# Plot the cumulative returns
import matplotlib.pyplot as plt

plt.figure()
plt.plot(df_strategy.index, df_strategy['Cumulative_Market_Return'], label='Market Returns', color='blue')
plt.plot(df_strategy.index, df_strategy['Cumulative_Strategy_Return'], label='Strategy Returns', color='green')
plt.xlabel('Date')
plt.ylabel('Cumulative Returns')
plt.title('Cumulative Market vs. Strategy Returns')
plt.legend()
plt.show()
```

jupyter AIML_CW1 Last Checkpoint: 6 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

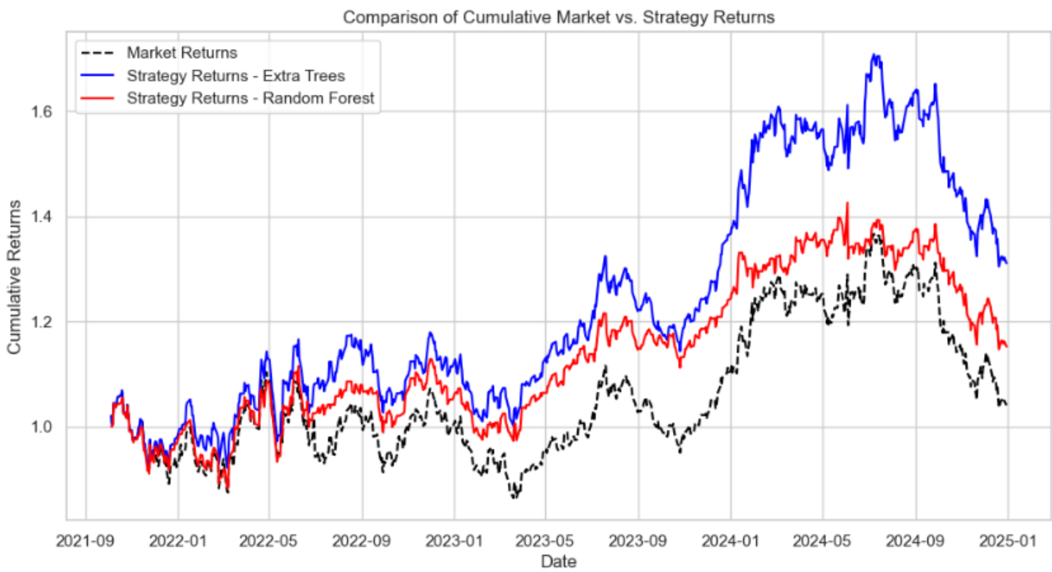


[82]:
```python
# Compute strategy returns
df_strategy = df.loc[X_test.index].copy()
df_strategy['Strategy_Return_ETC'] = df_strategy['Market_Return'] * etc_pred
df_strategy['Strategy_Return_RF'] = df_strategy['Market_Return'] * rf_pred

# Compute cumulative returns
df_strategy['Cumulative_Market_Return'] = (1 + df_strategy['Market_Return']).cumprod()
df_strategy['Cumulative_Strategy_Return_ETC'] = (1 + df_strategy['Strategy_Return_ETC']).cumprod()
df_strategy['Cumulative_Strategy_Return_RF'] = (1 + df_strategy['Strategy_Return_RF']).cumprod()

# Plot cumulative returns
plt.figure(figsize=(12,6))
plt.plot(df_strategy.index, df_strategy['Cumulative_Market_Return'], label='Market Returns', color='black',linestyle='dashed')
plt.plot(df_strategy.index, df_strategy['Cumulative_Strategy_Return_ETC'], label='Strategy Returns - Extra Trees', color='blue')
plt.plot(df_strategy.index, df_strategy['Cumulative_Strategy_Return_RF'], label='Strategy Returns - Random Forest', color='red')
plt.xlabel('Date')
plt.ylabel('Cumulative Returns')
plt.title('Comparison of Cumulative Market vs. Strategy Returns')
plt.legend()
plt.show()
```

jupyter AIML_CW1 Last Checkpoint: 6 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)



jupyter AIML_CW1 Last Checkpoint: 6 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

[87]:
```python
# Relevant columns for showcasing in the report
columns_to_show = ['Close', 'Market_Return', 'Strategy_Return_ETC', 'Strategy_Return_RF',
                   'Cumulative_Market_Return', 'Cumulative_Strategy_Return_ETC', 'Cumulative_Strategy_Return_RF']

# Display the last 10 rows for reference
df_strategy[columns_to_show].tail(10)
```

[87]:

Price	Close	Market_Return	Strategy_Return_ETC	Strategy_Return_RF	Cumulative_Market_Return	Cumulative_Strategy_Return_ETC	Cumulative_Strategy_Return_RF
icker	RELIANCE.NS						
Date							
024-2-16	1268.300049	-0.003575	-0.003575	-0.000000	1.089829	1.372588	1.206707
024-2-17	1245.300049	-0.018135	-0.018135	-0.018135	1.070065	1.347697	1.184824
024-2-18	1253.250000	0.006384	0.006384	0.006384	1.076896	1.356301	1.192388
024-2-19	1230.449951	-0.018193	-0.018193	-0.018193	1.057305	1.331626	1.170695
024-2-20	1205.300049	-0.020440	-0.020440	-0.020440	1.035694	1.304408	1.146766
024-2-23	1222.300049	0.014104	0.014104	0.014104	1.050302	1.322806	1.162941
024-2-24	1222.750000	0.000368	0.000368	0.000368	1.050688	1.323293	1.163369
024-2-26	1216.550049	-0.005070	-0.005070	-0.005070	1.045361	1.316583	1.157470
024-2-27	1221.050049	0.003699	0.003699	0.003699	1.049228	1.321453	1.161752
024-2-30	1210.699951	-0.008476	-0.008476	-0.008476	1.040334	1.310252	1.151904

[ ]:

[ ]: