NANYANG TECHNOLOGICAL UNIVERSITY

[Your Project ID]

The use of feature important XAI approaches in finding the influence of factors affecting the stock market

Submitted in Partial Fulfilment of the Requirements for the Degree of Bachelor of Computer Engineering of the Nanyang Technological University

By: Arjun Anand Pramod

School of Computer Science and Engineering

2024

Abstract

Acknowledgements

Table of Contents

2. Literature Review

2.1 Overview of Stock Market Prediction

The pursuit of stock market prediction harnesses the intricate dance of numerical data, investor sentiment, and global economic indicators. Traditional statistical models once dominated this space, offering linear approaches to understanding market movements. However, the advent of machine learning (ML) and, subsequently, deep learning (DL) technologies has revolutionized the field. Notably, Gite et al. (2021) underscore the utility of Long Short-Term Memory (LSTM) networks in capturing the sequential and temporal dependencies inherent in financial news and stock price data, presenting a novel approach that integrates sentiment analysis with quantitative data to forecast stock movements [1].

Moreover, the comprehensive survey by Vachhani et al. (2020) paints a broad picture of the ML landscape in stock prediction, exploring a variety of algorithms from Random Forest to Deep Neural Networks (DNNs). This work illuminates the complex decision-making involved in selecting appropriate models for specific prediction tasks, highlighting how different market conditions and data characteristics can influence model performance [2]. The exploration into the effects of global financial crises on predictive models reveals the adaptive nature of ML techniques, emphasizing the importance of dynamic model selection in response to evolving market scenarios.

2.2 Role of Explainable AI in Financial Modelling

As Machine Learning models become integral to financial analysis and decision-making, the black-box nature of many advanced algorithms poses a significant challenge. The drive for transparency and accountability in financial decisions has catalysed the development of Explainable Artificial Intelligence (XAI). Titan et al. (2015) delve into the Efficient Market Hypothesis, correlating it with the need for explainability in AI-driven models to ensure that predictions are not only accurate but also justifiable in real-time market contexts.

XAI's significance extends beyond theoretical appeal, touching on practical and regulatory concerns in financial operations. The analysis by Gite et al. (2021) on applying LIME for model explainability showcases how XAI techniques can demystify complex predictions, offering stakeholders insights into the factors driving stock market forecasts. This level of transparency is crucial in sensitive financial environments where decisions based on AI predictions can lead to significant economic impacts [1].

2.3 Previous Studies on LIME and SHAP

The emergence of LIME and SHAP as pillars of XAI represents a significant stride towards bridging the gap between advanced AI predictions and user interpretability. Gite et al. (2021) demonstrate the practical application of LIME in breaking down LSTM model predictions, offering a window into the model's reasoning by highlighting the influence of specific features [1]. This methodological approach is crucial for users requiring understandable insights from their predictive models, reinforcing the value of transparency in AI-driven financial analysis.

Similarly, SHAP's introduction into the financial modelling domain, as discussed in the broader survey of ML techniques by Vachhani et al. (2020), provides a robust framework for quantifying feature importance across various models. SHAP's game-theoretical foundation offers a consistent approach to attributing prediction influences, making it an invaluable tool for understanding and improving model performance in stock market predictions [2].

Conclusion

The literature underscores a dynamic evolution from traditional to ML-driven approaches in stock market prediction, emphasizing not just the pursuit of accuracy but also the necessity of transparency and interpretability in financial modelling. As the field continues to evolve, the integration of XAI techniques like LIME and SHAP with advanced ML models presents a promising frontier for making complex financial predictions more accessible and actionable for all market participants.

3. Methodology

This section outlines the comprehensive methodology adopted in this study to explore the various factors on stock market predictions through the lens of Explainable Artificial Intelligence (XAI). The methodology is split into three main stages: data collection, model development, and feature importance analysis with XAI techniques. Each stage plays a major role in the research process, from the initial gathering and preparation of data to the development and analysis of the predictive models.

1. **Data Collection:** This section includes the detailed procedures for obtaining and preparing the dataset used in this study, highlighting the collection of historical stock data from the S&P 500 and the calculation of key technical indicators. This foundation is critical for ensuring the quality and comprehensiveness of the data.

2. **Model Development:** This part describes the process of designing and training predictive models. It includes the selection of appropriate algorithms, data preprocessing steps, and the rationale behind the choice of models. The focus here is on developing models that are both accurate in their predictions and conducive to subsequent explanation and analysis.

3. **Feature Importance Analysis with XAI Techniques**: The final stage of the methodology emphasizes the application of XAI techniques, specifically LIME and SHAP, to interpret the predictive models. This analysis aims to uncover the significance of various features in the models' decisions, bridging the gap between prediction accuracy and model interpretability.

Overall, the methodology seeks to provide a structured approach to investigating how different factors influence stock market trends, leveraging advanced machine learning models and XAI techniques. The aim is to achieve a balance between predictive performance and the ability to understand and trust the model’s outputs.

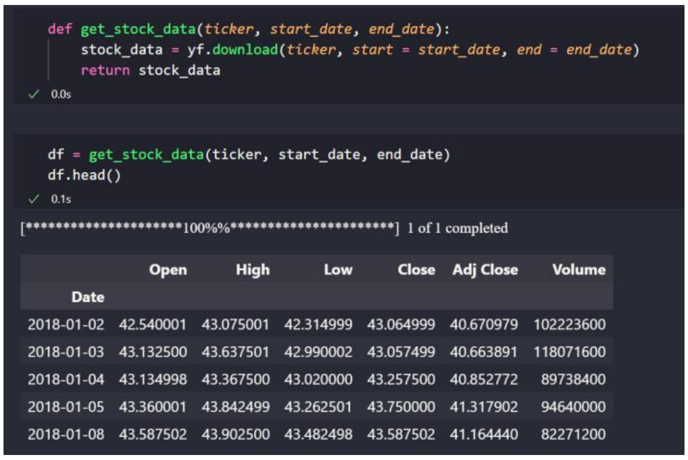
3.1 Methodology: Data Collection

This section describes the process and methodology used for collecting and preparing the datasets utilized in this study. Recognizing the diverse influences of industry-specific factors on stock performance, three distinct datasets were created for stocks within different industries: technology, healthcare, and finance. To illustrate the data collection method in a focused manner, we will primarily examine Apple Inc.'s stock, a representative from the technology sector. The primary data source for this analysis is the historical stock prices of companies listed in the S&P 500, accessed via the Yahoo Finance Python API. Additionally, technical indicators were calculated to enrich the dataset with insights into stock market trends and behaviours.

Data Collection Process

Historical Stock Data Retrieval:

- Utilized the Yahoo Finance Python API to collect historical daily stock prices for the S&P 500 listed companies, including a detailed focus on sectors such as technology, healthcare, and finance.
- The data collection spanned from January 2018 to November 2023, including the following key attributes for each stock: Date, Open, High, Low, Close, Adjusted Close, and Volume.



```
def get_stock_data(ticker, start_date, end_date):
    stock_data = yf.download(ticker, start = start_date, end = end_date)
    return stock_data
✓ 0.0s
```

```
df = get_stock_data(ticker, start_date, end_date)
df.head()
✓ 0.1s
```

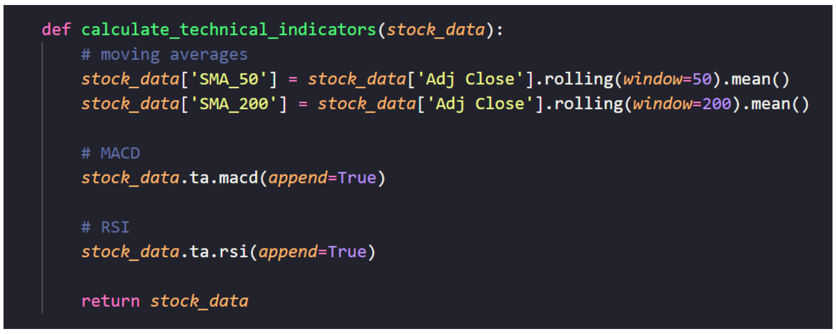
```
[*********************100%%**********************]  1 of 1 completed
```

	Open	High	Low	Close	Adj Close	Volume
Date						
2018-01-02	42.540001	43.075001	42.314999	43.064999	40.670979	102223600
2018-01-03	43.132500	43.637501	42.990002	43.057499	40.663891	118071600
2018-01-04	43.134998	43.367500	43.020000	43.257500	40.852772	89738400
2018-01-05	43.360001	43.842499	43.262501	43.750000	41.317902	94640000
2018-01-08	43.587502	43.902500	43.482498	43.587502	41.164440	82271200

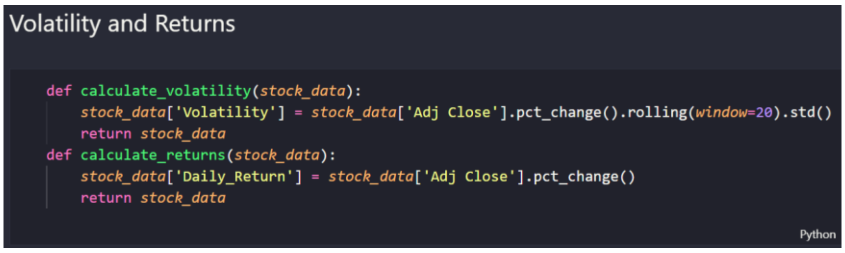
Figure

Calculation of Technical Indicators:

- Beyond basic stock information, several technical indicators were computed to provide deeper insights into market trends. These indicators include Simple Moving Averages (SMA) with periods of 50 and 200 days, Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI) over 14 days, volatility indicators, and Daily Returns.
- The calculations were performed using custom Python scripts, leveraging pandas and NumPy libraries for data manipulation and computation.

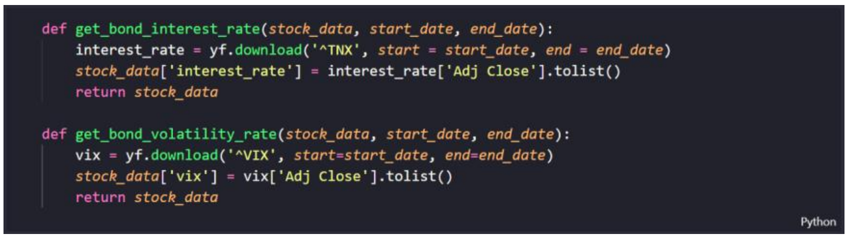


```python
def calculate_technical_indicators(stock_data):
    # moving averages
    stock_data['SMA_50'] = stock_data['Adj Close'].rolling(window=50).mean()
    stock_data['SMA_200'] = stock_data['Adj Close'].rolling(window=200).mean()

    # MACD
    stock_data.ta.macd(append=True)

    # RSI
    stock_data.ta.rsi(append=True)

    return stock_data
```



Volatility and Returns

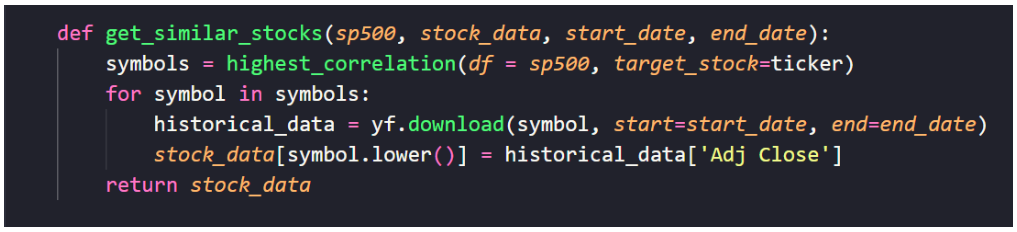
```python
def calculate_volatility(stock_data):
    stock_data['Volatility'] = stock_data['Adj Close'].pct_change().rolling(window=20).std()
    return stock_data
def calculate_returns(stock_data):
    stock_data['Daily_Return'] = stock_data['Adj Close'].pct_change()
    return stock_data
```



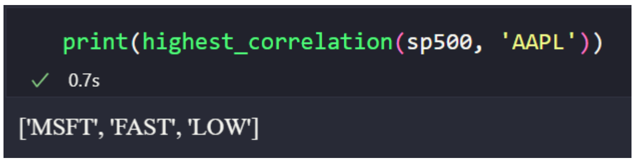
```python
def get_bond_interest_rate(stock_data, start_date, end_date):
    interest_rate = yf.download('^TNX', start = start_date, end = end_date)
    stock_data['interest_rate'] = interest_rate['Adj Close'].tolist()
    return stock_data

def get_bond_volatility_rate(stock_data, start_date, end_date):
    vix = yf.download('^VIX', start=start_date, end=end_date)
    stock_data['vix'] = vix['Adj Close'].tolist()
    return stock_data
```

Correlation Analysis for Stock Selection:

- To identify stocks with similar price movements within each industry, a correlation analysis was conducted using the historical price data.
- The **get_similar_stocks** function, based on the **highest_correlation** method, was applied to the SPY (S&P 500 ETF) data to select stocks that exhibit high correlation with the broader market index, with a special focus on Apple Inc. for demonstration purposes.

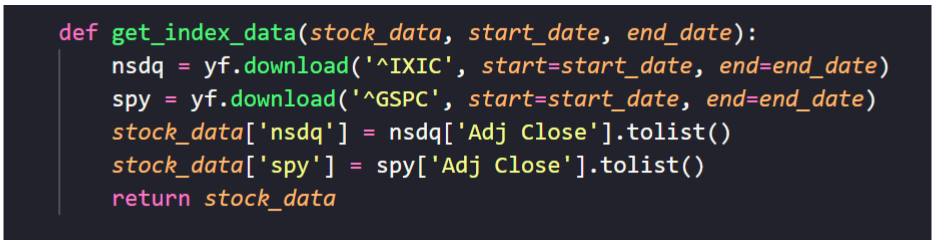


```
def get_similar_stocks(sp500, stock_data, start_date, end_date):
    symbols = highest_correlation(df = sp500, target_stock=ticker)
    for symbol in symbols:
        historical_data = yf.download(symbol, start=start_date, end=end_date)
        stock_data[symbol.lower()] = historical_data['Adj Close']
    return stock_data
```



```
print(highest_correlation(sp500, 'AAPL'))
✓ 0.7s
['MSFT', 'FAST', 'LOW']
```

- Additionally, the **get_index_data** function gets the Adjusted Close of the S&P 500 index and the NASDAQ over the same period.



```
def get_index_data(stock_data, start_date, end_date):
    nsdq = yf.download('^IXIC', start=start_date, end=end_date)
    spy = yf.download('^GSPC', start=start_date, end=end_date)
    stock_data['nsdq'] = nsdq['Adj Close'].tolist()
    stock_data['spy'] = spy['Adj Close'].tolist()
    return stock_data
```

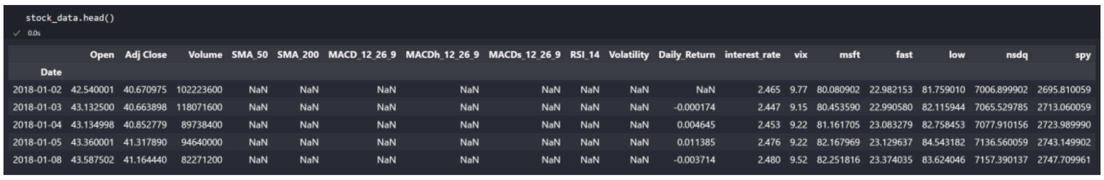
Examination of the Dataset

Following the collection and initial processing of the data, the dataset was downloaded as a CSV file for detailed analysis. The Pandas library, a powerful tool for data analysis in Python, was employed to import the dataset as a dataframe, facilitating a comprehensive examination of its structure and contents.

Dataset Importation and Initial Inspection

The dataset was imported into a Python environment using Pandas, allowing for easy manipulation and analysis. To understand the basic structure and get an overview of the data, the following steps were taken:

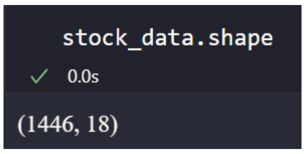
- **Head of Dataframe**: The **.head()** method was applied to the dataframe to display the first few rows of the dataset, providing a glimpse into the data's overall structure and the types of values it contains.



```
stock_data.head()
✓ 0.0s
```

	Open	Adj Close	Volume	SMA_50	SMA_200	MACD_12_26_9	MACDh_12_26_9	MACDs_12_26_9	RSI_14	Volatility	Daily_Return	interest_rate	vix	msft	fast	low	nsdq	spy
Date																		
2018-01-02	42.540001	40.670975	102223600	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.465	9.77	80.080902	22.982153	81.759010	7006.899902	2695.810059
2018-01-03	43.132500	40.663898	118071600	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.000174	2.447	9.15	80.453590	22.990580	82.115944	7065.529785	2713.060059
2018-01-04	43.134998	40.852779	89738400	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.004645	2.453	9.22	81.161705	23.083279	82.758453	7077.910156	2723.989990
2018-01-05	43.360001	41.317890	94640000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.011385	2.476	9.22	82.167969	23.129637	84.543182	7136.560059	2743.149902
2018-01-08	43.587502	41.164440	82271200	NaN	NaN	NaN	NaN	NaN	NaN	NaN	-0.003714	2.480	9.52	82.251816	23.374035	83.624046	7157.390137	2747.709961

- **Shape of Dataframe**: To ascertain the size of the dataset, the **.shape** attribute was examined, revealing the number of rows and columns. This information is crucial for understanding the dataset's scope and ensuring it is adequately sized for the analysis.

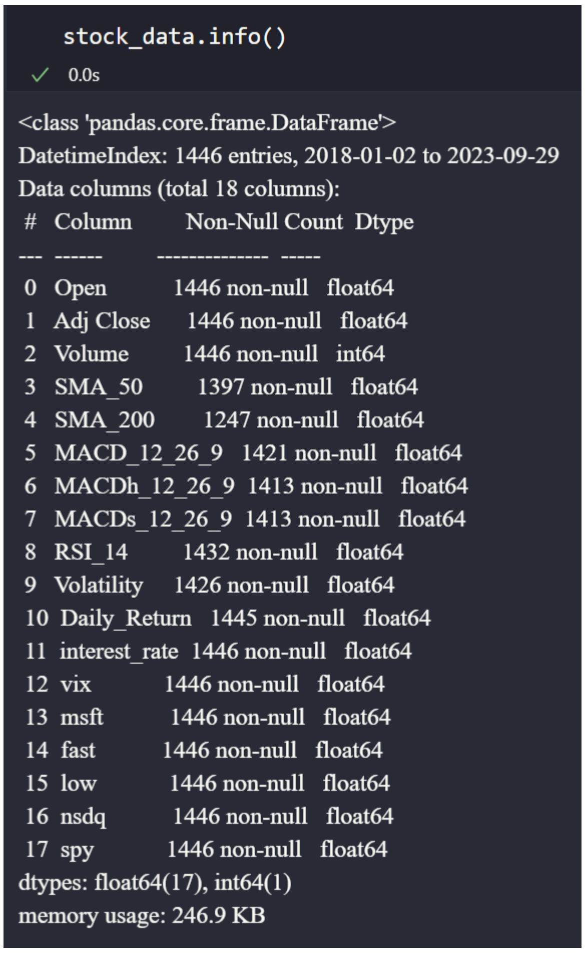


```
stock_data.shape
✓ 0.0s
(1446, 18)
```

Data Type Analysis

A critical aspect of preparing the dataset for analysis involves understanding the data types of each feature:

- **Feature Data Types**: Using the **.info()** attribute, the data types of each column were identified.



```
stock_data.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1446 entries, 2018-01-02 to 2023-09-29
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Open            1446 non-null   float64
 1   Adj Close       1446 non-null   float64
 2   Volume          1446 non-null   int64
 3   SMA_50          1397 non-null   float64
 4   SMA_200         1247 non-null   float64
 5   MACD_12_26_9    1421 non-null   float64
 6   MACDh_12_26_9   1413 non-null   float64
 7   MACDs_12_26_9   1413 non-null   float64
 8   RSI_14          1432 non-null   float64
 9   Volatility      1426 non-null   float64
 10  Daily_Return    1445 non-null   float64
 11  interest_rate   1446 non-null   float64
 12  vix             1446 non-null   float64
 13  msft            1446 non-null   float64
 14  fast            1446 non-null   float64
 15  low             1446 non-null   float64
 16  nsdq            1446 non-null   float64
 17  spy             1446 non-null   float64
dtypes: float64(17), int64(1)
memory usage: 246.9 KB
```

Feature Categorization

Upon closer examination, we can see that all of the features being used are numerical and of the float64 data type. Additionally, we observe that many of the columns have null values which have to be dealt with before using in our models.

This section of the methodology outlines the critical steps undertaken to inspect and analyse the dataset post-collection. By utilizing the Pandas library to import and examine the dataset, a foundational understanding of its structure, size, and feature types was established, setting the stage for more detailed analyses and model development.

Tools and Technologies Used

- **Programming Language**: Python 3.11.4
- **Libraries and APIs**: Yahoo Finance Python API, pandas, NumPy, requests, Beautiful Soup 4.
- **Data Storage**: The final cleaned and processed dataset was stored in a CSV format for easy access and manipulation in subsequent analysis stages.

3.2 Methodology: Model Development

This study employs a diverse set of models to predict the adjusted close price of stocks listed in the S&P 500. The models selected include Linear Regression, Support Vector Regression (SVR), Random Forest Regressor, LightGBM Regressor, XGBoost, and an Ensemble model. Each model brings unique strengths to multi-feature time series forecasting, leveraging the dataset spanning from January 2018 to September 2023. The exclusion of 'Date', 'Adj Close', and 'Open' from the features ensures the focus remains on indicators that provide predictive value without redundancy.

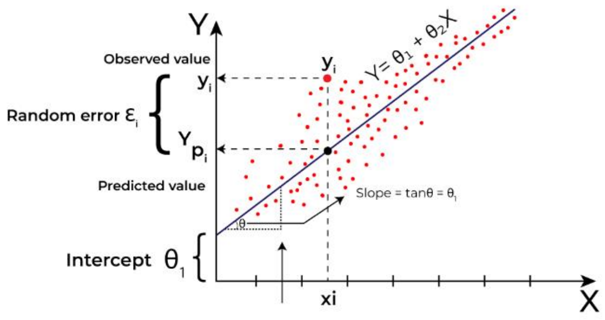
Model Selection and Rationale

1. **Linear Regression:**

   Core Concept: Linear regression predicts a response variable as a linear combination of predictor variables. It assumes a linear relationship between inputs and the target.

   Mathematical Representation: The model is defined as $y = X\beta + \epsilon$ where y is the vector of observed values, X is the matrix of input features, β is the vector of coefficients, and ϵ is the error term.

   Relevance to Time Series Forecasting: Despite its simplicity, linear regression can serve as a strong baseline in time series forecasting. Its interpretability is a significant advantage when evaluating the impact of different features on stock prices.



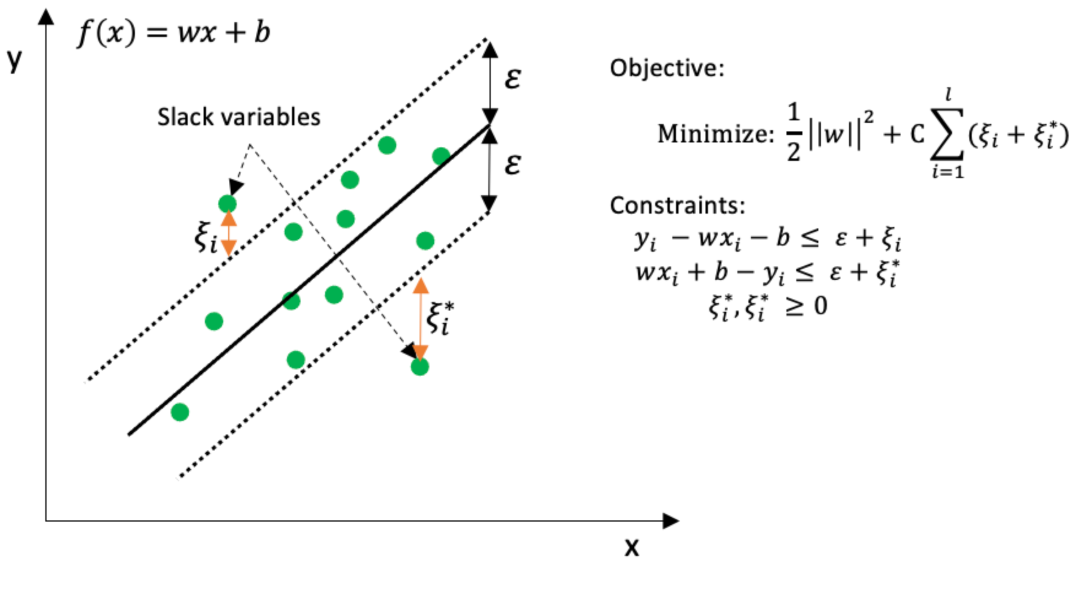
Y
Observed value
y_i
Random error ε_i
Y_{p_i}
Predicted value
y_i
$Y = \theta_1 + \theta_2 X$
Slope = tanθ = θ_i
θ
Intercept θ_1
xi
X

Figure

2. **Support Vector Regression (SVR):**

   Core Concept: SVR applies the principles of SVM to regression problems, creating a margin of tolerance (ϵ) and optimizing this margin to include as many instances as possible. The RBF kernel allows SVR to handle non-linear data by mapping input features into higher-dimensional spaces.

   Relevance to Time Series Forecasting: Its capacity to model complex non-linear relationships makes SVR with RBF kernel an excellent choice for capturing the intricate patterns in stock market data, where price movements are influenced by numerous factors.


$f(x) = wx + b$
y
x
Slack variables
ε
ε
ξ_i
ξ_i^*
Objective:
Minimize: $\frac{1}{2}||w||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$
Constraints:
$y_i - wx_i - b \leq \varepsilon + \xi_i$
$wx_i + b - y_i \leq \varepsilon + \xi_i^*$
$\xi_i^*, \xi_i^* \geq 0$

Univariate linear SVR (allowing for errors)

Figure

3. **Random Forest Regressor:**

Core Concept: A type of ensemble learning method that builds multiple decision trees during training and outputs the average prediction of the trees. It uses bagging and feature randomness when building each tree to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

Mathematical Representation: The prediction of the random forest regressor is given by $y = \frac{1}{N}\Sigma_{i=1}^{N} t_i(x)$, where $t_i(x)$ is the prediction of the i^{th} tree.

Relevance to Time Series Forecasting: Its ensemble nature allows it to effectively capture complex relationships within the data without the risk of overfitting, making it highly suitable for the multifaceted nature of stock market movements.

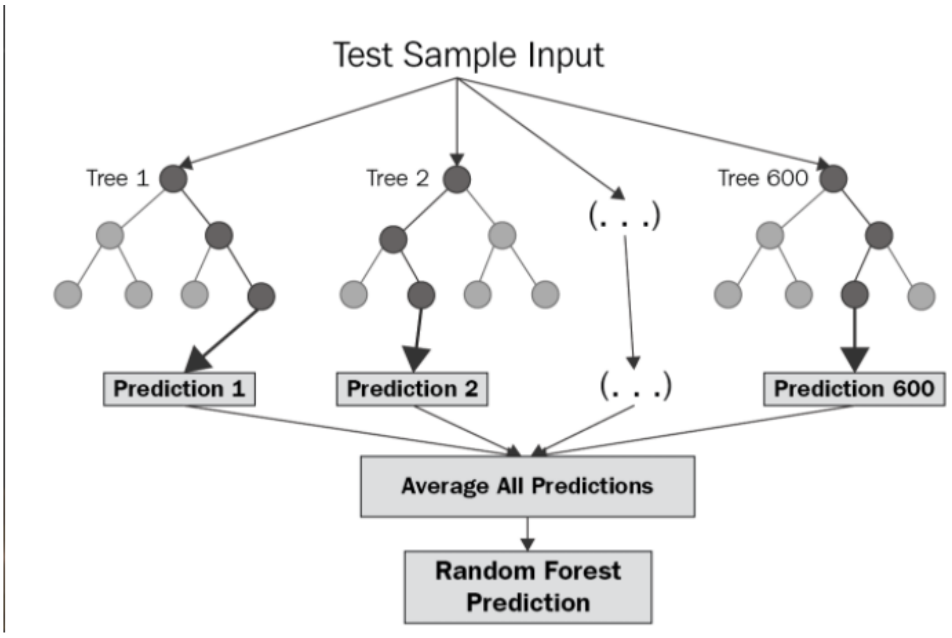


Figure:

4. **LightGBM Regressor:**

   Core Concept: An efficient gradient boosting framework that uses tree-based learning algorithms. It grows trees vertically (leaf-wise) rather than horizontally, meaning it splits the data on the leaf that minimizes the loss, leading to faster convergence.

   Mathematical Representation: Similar to XGBoost, LightGBM minimizes the following loss function: $\Sigma_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega\,(f_k)$, where l is a differentiable convex loss function, $\hat{y}_i$ is the predicted value, and Ω represents the regularization term.

   Relevance to Time Series Forecasting: LightGBM's speed and efficiency make it particularly useful for large datasets, like those found in stock market forecasting. Its ability to handle sparse data and its effectiveness in dealing with various types of features are critical for time series analysis

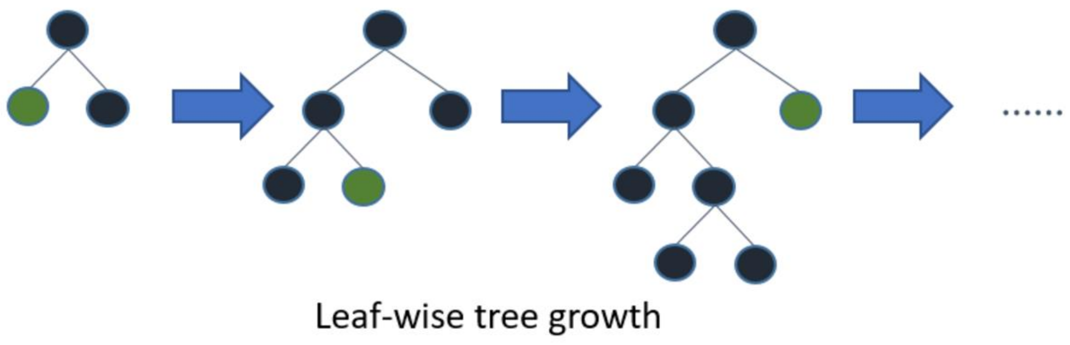


Figure:

5. **XGBoost:**

Core Concept: Stands for Extreme Gradient Boosting, XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It is renowned for its performance and speed.

Mathematical Representation: XGBoost aims to minimize the following regularized objective: $\Sigma_i l(y_i, \hat{y}_i) + \Sigma_k \Omega(f_k)$, focusing on computational speed and model performance.

Relevance to Time Series Forecasting: XGBoost's powerful predictive capabilities and flexibility in handling various types of predictive modelling make it ideal for the fluctuating and often non-linear nature of stock prices.

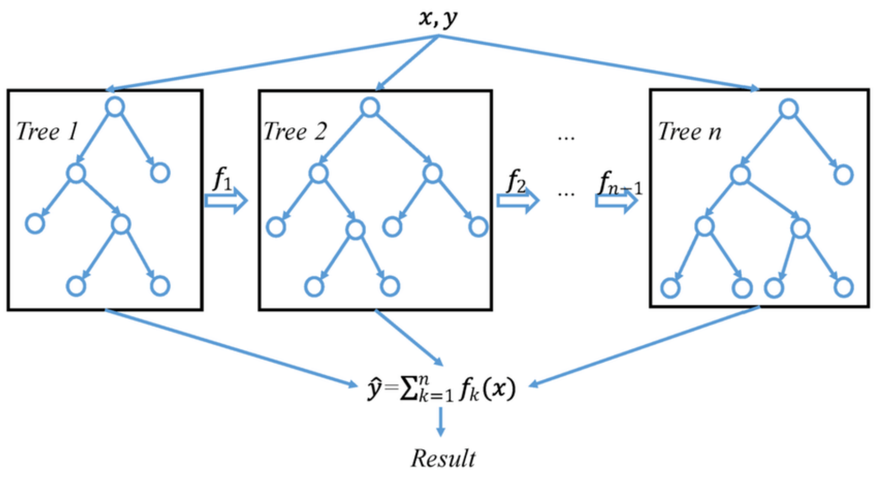


Figure:

6. **Ensemble:**

Core Concept: Combines predictions from several individual models to improve robustness over

3.3 Methodology: Feature Importance Analysis with XAI Techniques

This section discusses the application of XAI techniques, specifically LIME and SHAP, to analyse and interpret the feature importance derived from the predictive models developed in this study. These techniques are pivotal in demystifying the models' decisions, providing insights into how different features affect the adjusted close price predictions in stock market forecasting.

Local Interpretable Model-agnostic Explanations (LIME):

Core Concept: LIME provides explanations for individual predictions by approximating the global model with a local, interpretable model around the prediction. It modifies the data sample to generate a new dataset of perturbed samples and observes how the predictions change with these perturbations. By fitting a simple model (like linear regression) to this new dataset, LIME highlights which features are most influential for a specific prediction.

Mathematical Representation: Given a complex model f and an instance x, LIME generates a new dataset of perturbed samples Z and their corresponding predictions $f(Z)$. It then weights these samples based on their proximity to x, often using an exponential kernel. The interpretable model g is trained on Z, with the objective function incorporating the proximity weights to ensure fidelity near x.

Relevance to Feature Importance: LIME's local approach allows us to understand which features significantly influence individual predictions. This is particularly useful in scenarios where global interpretability is challenging or less meaningful, providing actionable insights into specific data points.

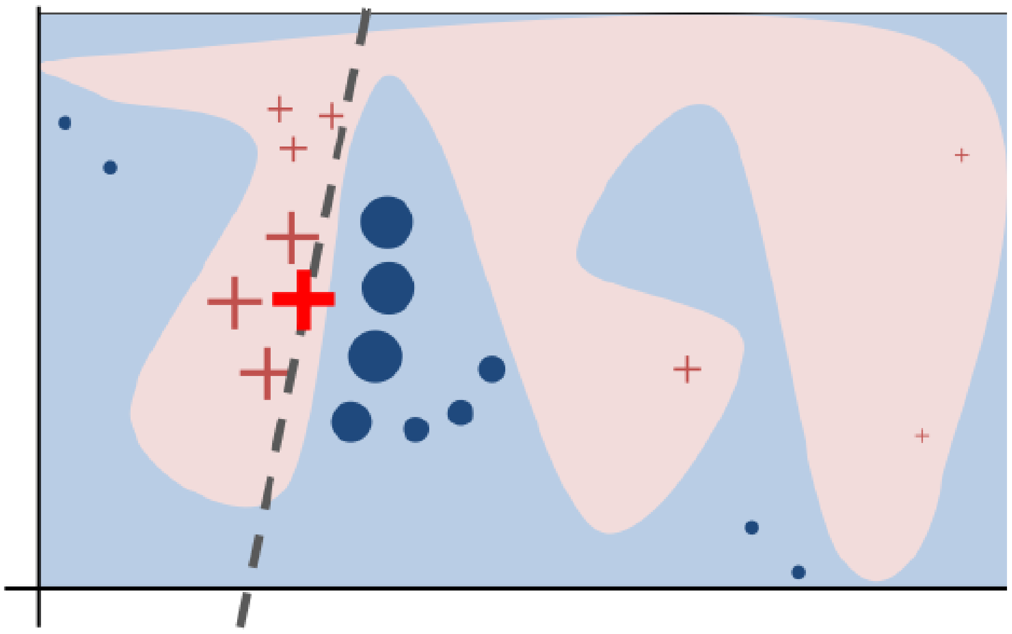


Figure:

SHapley Additive exPlanations (SHAP):

Core Concept: SHAP leverages the concept of Shapley values from cooperative game theory to assign each feature an importance value for a particular prediction. It computes the contribution of each feature by considering all possible combinations of features, essentially calculating the marginal contribution of a feature by adding it to a subset of features.

Mathematical Representation:

$$\phi_i(p) = \sum_{S \subseteq N/i} \frac{|S|!(n-|S|-1)!}{n!}(p(S \cup i) - p(S))$$

Where N is the set of all features, S is a subset of features without j, and $f(S)$ is the prediction model's output when only the features in S are used.

Relevance to Feature Importance: SHAP provides a unified measure of feature importance that is consistent and accurate, applicable across different models. It not only offers insights into the overall importance of features but also explains individual predictions with precise contributions from each feature, facilitating a deeper understanding of model behaviour.

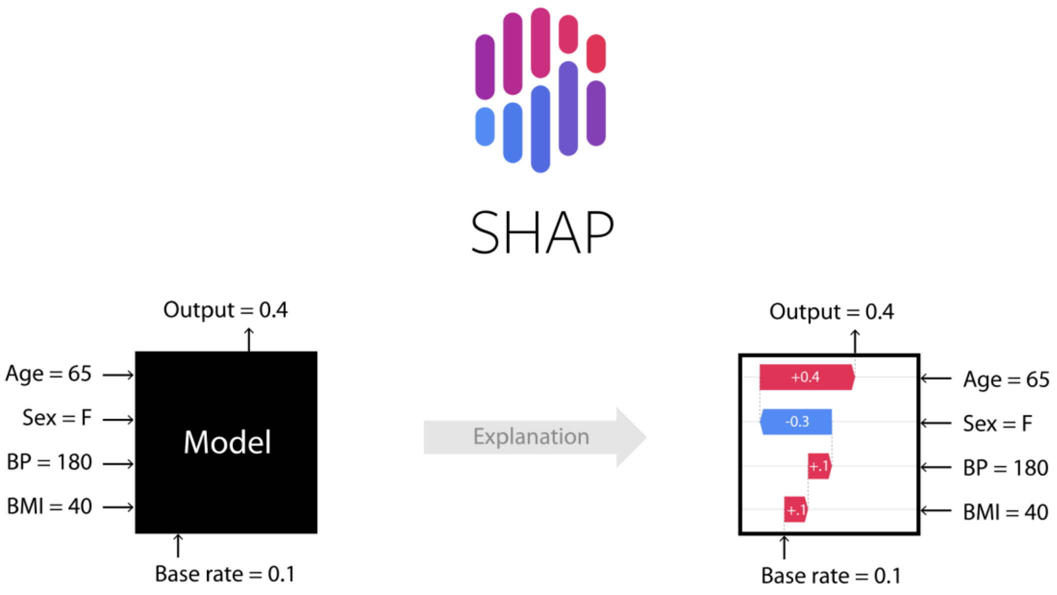


Figure:

4. Project Implementation

This section delves into the practical execution and application phases of the project, detailing the specific steps taken to preprocess data, develop and train predictive models, and apply XAI techniques for feature importance analysis. Following the foundational groundwork laid in the methodology, this section transitions into the tangible actions undertaken to bring the research objectives to fruition. It is structured into three critical stages: **Data Preprocessing, Model Training and Evaluation, and the Application of LIME and SHAP**. Each stage is instrumental in advancing from theoretical models and hypotheses to actionable insights and results.

Data Preprocessing: Here, we describe the comprehensive processes involved in cleaning, transforming, and preparing the dataset for analysis. This includes handling missing values, encoding categorical variables, normalizing numerical data, and creating training and testing splits. This stage ensures the data is in an optimal format for modelling, addressing any issues that could bias or invalidate the predictive models.

Model Training and Evaluation: This segment focuses on the practical aspects of building and assessing the performance of the predictive models outlined in the Model Development section. It covers the selection of hyperparameters, training the models on the pre-processed data, and rigorously evaluating their performance using designated metrics such as RMSE. The evaluation process is critical for identifying the most effective models and fine-tuning them for improved accuracy and reliability.

Application of LIME and SHAP: The final segment highlights the application of XAI techniques to interpret the models developed in the previous stage. By employing LIME and SHAP, this section aims to shed light on how the models make their predictions and which features significantly impact those predictions. This not only enhances the transparency and trustworthiness of the models but also provides deeper insights into the dynamics of stock market movements as captured by the models.

Overall, the Project Implementation section embodies the transition from theoretical methodologies to practical application, culminating in a set of predictive models that are both powerful and interpretable. Through meticulous data preprocessing, thoughtful model training and evaluation, and the strategic application of XAI techniques, this section reinforces the study's commitment to uncovering the nuanced factors that influence stock market trends. The ultimate goal is to forge a link between sophisticated machine learning techniques and actionable financial insights, ensuring the models developed are not only accurate but also comprehensible and useful for stakeholders.

4.1 Project Implementation: Data Preprocessing

The data preprocessing stage is crucial in preparing the dataset for the modelling phase, ensuring the data is clean, relevant, and devoid of any inconsistencies that might skew the results. This section specifically addresses the handling of missing values in the dataset, a common issue in time-series data, particularly when dealing with financial indicators like Simple Moving Averages (SMA) that require historical data points to calculate.

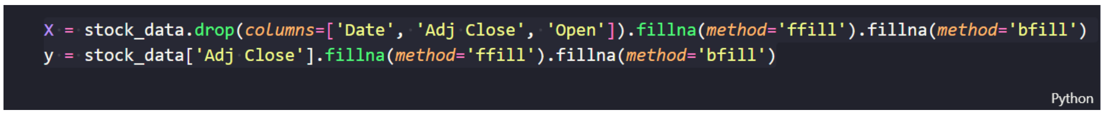
Handling Missing Values

The `stock_data` dataframe, exhibits missing values in several columns, notably in technical indicators such as SMA_50, SMA_200, MACD lines, RSI_14, and Volatility. These null entries primarily result from the nature of these indicators, which depend on a certain number of previous values for computation. For instance, SMA_50 cannot be calculated for the first 49 days in the dataset.

To address these missing values and maintain the integrity of the dataset, the following approach was adopted:

Forward Filling (ffill): This method propagates the last observed non-null value forward until another non-null value is encountered. It's particularly suited for time-series data where the next valid observation is likely to be similar to the previous one.

Backward Filling (bfill): After applying forward filling, backward filling is used as a complementary step to fill any remaining null values that occur at the beginning of the dataset. This works by filling in nulls with the next observed non-null value.



```python
X = stock_data.drop(columns=['Date', 'Adj Close', 'Open']).fillna(method='ffill').fillna(method='bfill')
y = stock_data['Adj Close'].fillna(method='ffill').fillna(method='bfill')
```

Figure:

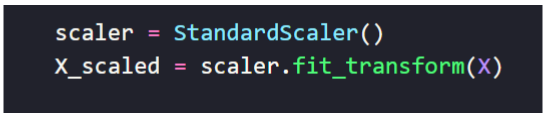
Application to Features and Target:

For the feature matrix `X`, columns 'Date', 'Adj Close', and 'Open' were dropped to focus on relevant predictors. The remaining columns were then subjected to null value handling using the `.fillna(method='ffill')` followed by `.fillna(method='bfill')` to ensure no missing values remain.

The target variable `y`, representing the 'Adj Close' prices, was also processed with the same forward and backward filling methods to ensure consistency in data points.

Data Transformation and Split

Following the missing value treatment, the dataset was ready for further transformations as needed by specific models, such as normalization or standardization, to put all variables on a similar scale.



```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Figure:

The pre-processed dataset was then split into training and testing sets based on the temporal order of observations. Typically, the most recent data points serve as the test set to evaluate the model's performance on unseen data, closely simulating real-world forecasting scenarios.

This preprocessing stage is foundational for the success of the subsequent modelling efforts. By handling missing values and ensuring data quality, the study sets a robust base for developing predictive models. The methods of 'forward fill' and 'backward fill' were chosen for their appropriateness in a time-series context, where the temporal sequence and continuity of data are paramount. This approach ensures that the models developed in the later stages are trained on a comprehensive and coherent dataset, enhancing their potential to yield accurate and reliable forecasts.

4.2 Model Training, Evaluation, and Explanation using XAI Techniques

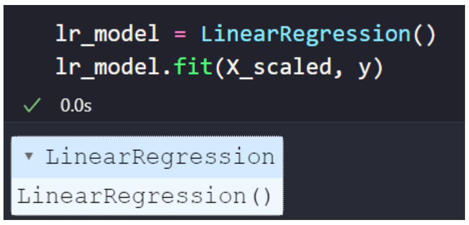
1. Linear Regression

The model begins with loading the stock market data, preprocessing it for analysis, and scaling the features to standardize the data distribution. A Linear Regression model is trained on the scaled features to predict the adjusted closing prices of stocks. The model aims to capture linear relationships between various stock indicators and the 'Adj Close' prices.

Data Preparation: The dataset is filtered up to a cutoff date, '2023-09-01', to separate the training and future prediction sets. Missing values are filled using forward and backward filling to ensure data integrity.

Feature Scaling: The **StandardScaler** from **sklearn.preprocessing** is applied to normalize the feature set, enhancing model performance by treating all variables equally in terms of scale.

Model Training: A **LinearRegression** model from **sklearn.linear_model** is fitted on the scaled features, encapsulating the essence of stock market behavior within its parameters.



```
lr_model = LinearRegression()
lr_model.fit(X_scaled, y)
```

```
✓ 0.0s

▾ LinearRegression
LinearRegression()
```

Figure:

Future Prediction Simulation: To forecast future stock prices, the model uses the last observed day's features as a base, introducing slight variations to simulate potential changes over the next 10 days. These simulated features are then used to predict future 'Adj Close' prices.

Visualization and RMSE Calculation: The actual and predicted prices for the immediate 10 days following the training data cutoff are visualized, comparing the model's forecasts against real-world outcomes. The Root Mean Square Error (RMSE) metric quantifies the model's prediction accuracy.

XAI Techniques for Model Explanation

To interpret the model's predictions and understand the influence of various features, two prominent XAI techniques are employed: LIME and SHAP.

LIME (Local Interpretable Model-agnostic Explanations): LIME is applied to an instance from the dataset to elucidate the model's decision-making process. By approximating the Linear Regression model locally around the instance, LIME identifies and ranks the features most contributing to the prediction, enhancing interpretability on an individual prediction basis.

SHAP (SHapley Additive exPlanations): SHAP values are computed for the entire dataset to offer a global perspective on feature importance. Utilizing **shap.LinearExplainer**, the technique assigns each feature a SHAP value for each prediction, indicating its contribution towards pushing the model output from the base value (the average model output over the dataset) to the actual prediction. A summary plot aggregates these contributions across all predictions, highlighting overall feature importance and distribution of effects.

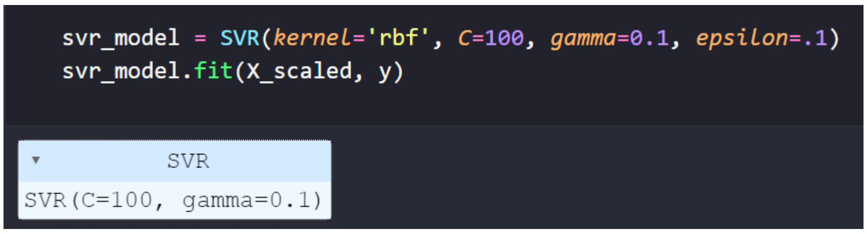
The combined analysis reveals key insights into the stock market dynamics captured by the Linear Regression model, validated by empirical evidence through visual comparison and RMSE metrics. The application of LIME and SHAP further dissects the model's rationale, offering transparency into the predictive process.

2. Support Vector Regression (SVR):

The SVR model, a type of Support Vector Machine used for regression tasks, is trained to predict stock prices, emphasizing its capacity to handle non-linear relationships through the use of kernel functions. This section outlines the preparation, training, and evaluation of the SVR model, focusing on its application to predict the adjusted closing prices of stocks.

Data Preparation: Similar to the Linear Regression approach, the dataset is prepared, with necessary preprocessing steps including feature scaling and missing value treatment. The SVR model benefits significantly from feature scaling, which ensures that all features contribute equally to the model's learning process.

Model Training: The SVR model is trained with a Radial Basis Function (RBF) kernel, chosen for its effectiveness in capturing complex, non-linear relationships between features. Key hyperparameters such as **C**, **gamma**, and **epsilon** are carefully selected to balance the model's bias-variance trade-off and to control the kernel's width and the margin of tolerance.



```
svr_model = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
svr_model.fit(X_scaled, y)
```

```
SVR
SVR(C=100, gamma=0.1)
```

Figure:

Future Prediction Simulation: The model makes predictions for future stock prices by leveraging the last known feature set, introducing slight variations to account for possible market changes. This approach allows for a realistic simulation of the model's predictive capabilities over a short-term horizon.

Model Evaluation: The SVR model's performance is quantitatively assessed using the Root Mean Square Error (RMSE) metric, alongside a visual comparison of predicted versus actual stock prices. Such evaluation provides insights into the model's accuracy and reliability.

XAI Techniques for Model Explanation

To enhance the interpretability of the SVR model, XAI techniques, particularly LIME, and SHAP, are utilized.

Following the training and evaluation of the SVR model, SHAP (SHapley Additive exPlanations) is employed to dissect the model's predictive behavior further. The use of SHAP provides an in-depth look at how individual features influence the prediction of stock prices, offering a global perspective on feature importance.

SHAP Explainer Initialization and Value Computation

SHAP KernelExplainer: Due to the black-box nature of the SVR model, particularly with non-linear kernels like RBF, the **KernelExplainer** is a suitable choice for deriving SHAP values. This explainer approximates SHAP values for any model by using a weighted linear regression to estimate how much each feature contributes to the prediction in comparison to a baseline prediction.

Computational Considerations: Given the potentially high computational cost of **KernelExplainer**, especially with large datasets, a sample of 100 instances from the scaled feature set (**X_scaled**) is used. This sampling strikes a balance between computational efficiency and the representativeness of the explanations.

SHAP Value Computation: The SHAP values are computed for the same sample used to initialize the explainer. These values quantify the contribution of each feature to the deviation of a prediction from the mean prediction over the sampled dataset.

Summary Plot: The SHAP summary plot aggregates the effects of all features across all predictions, offering a visual representation of each feature's overall impact on the model's output. It ranks features by their importance and shows the distribution of the impacts each feature has on the model's predictions, differentiating between positive and negative contributions.

After evaluating the SVR model's performance, the project employs LIME to offer instance-specific explanations, complementing the global insights provided by SHAP. LIME's local interpretability approach is particularly useful for understanding how the model predicts stock price movements for individual observations, making it an invaluable tool for stakeholders seeking detailed explanations of specific predictions.

LIME Explainer Initialization: A **LimeTabularExplainer** is created, utilizing the scaled training data (**X_scaled**) and specifying that the model's task is regression. The explainer is configured to interpret the SVR model's predictions with respect to the dataset's features.

Generating Local Explanations: The explainer focuses on an individual instance from the dataset, providing a detailed breakdown of how each feature contributes to the SVR model's prediction for that instance. In this case, the 10th data point is chosen as an example.

Interpretation: The output from LIME presents the features that have the most significant impact on the prediction for the chosen instance, highlighting both positive and negative contributions. This local explanation helps demystify the model's behavior for individual predictions, offering insights into the rationale behind the SVR model's forecasts.

The combined use of SHAP and LIME for explaining the SVR model's predictions offers a holistic view of model interpretability:

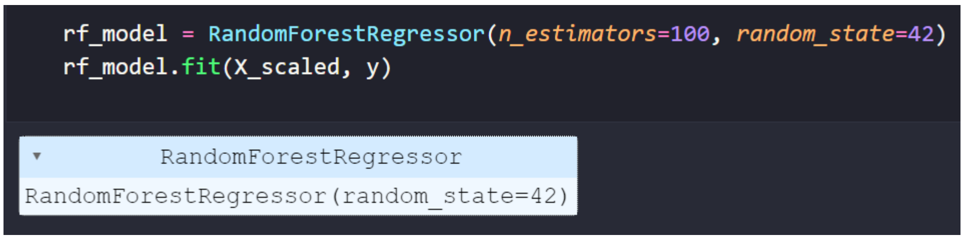
- **SHAP** provides a bird's-eye view of feature importance and their impact on the model's predictions across the dataset, revealing general trends and patterns in how features influence stock price forecasts.
- **LIME**, in contrast, zooms in on specific instances, detailing the contribution of each feature to a particular prediction. This granular approach is invaluable for understanding anomalies or validating the model's behaviour in specific scenarios.

3. Random Forest Regressor:

The Random Forest Regressor, known for its versatility and robustness, is trained on a dataset comprising various stock market indicators to predict the adjusted closing prices of stocks. This ensemble model combines multiple decision trees to reduce the risk of overfitting and improve prediction accuracy, making it particularly well-suited for complex regression tasks like stock price forecasting.

Data Preparation: The dataset is filtered to include data up to a specified cutoff date ('2023-01-01'), with subsequent dates reserved for future prediction testing. Missing values are handled via forward and backward filling to ensure data completeness. Features are then scaled using `StandardScaler` to normalize their distributions, a crucial step for models sensitive to feature magnitude.

Model Training: The Random Forest model is instantiated with 100 trees (`n_estimators=100`) and a set seed for reproducibility (`random_state=42`). It is then trained on the scaled feature set, aiming to capture both the linear and non-linear relationships between the predictors and the target variable, 'Adj Close'.



```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_scaled, y)
```

```
▾         RandomForestRegressor
RandomForestRegressor(random_state=42)
```

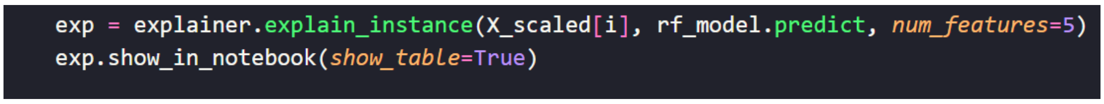
Figure:

Future Prediction and Evaluation: To evaluate the model's forecasting ability, a simulation of future market conditions is created by introducing slight variations to the last known set of features. The model's predictions for the next 5 days are visualized alongside actual prices for the same period, providing a direct comparison of its predictive accuracy. The Root Mean Square Error (RMSE) serves as the quantitative measure of the model's performance.

Explanation using XAI Techniques

To interpret the Random Forest model's predictions, both SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) techniques are employed, offering insights into how different features influence the model's output.

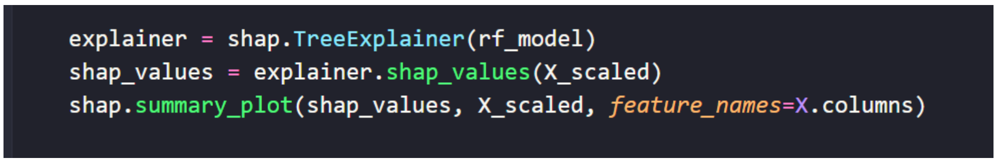
LIME for Local Interpretability: A `LimeTabularExplainer` is configured for the regression mode with the scaled training data. LIME is then applied to an individual instance (e.g., the 10th in the dataset), providing a detailed explanation of the model's prediction for that specific data point, highlighting the contribution of the top features.



```
exp = explainer.explain_instance(X_scaled[i], rf_model.predict, num_features=5)
exp.show_in_notebook(show_table=True)
```

Figure:

SHAP for Global and Local Insights: A `TreeExplainer`, tailored for tree-based models like Random Forest, is used to compute SHAP values for the entire dataset. These values offer a comprehensive view of feature importance, illustrating how each feature contributes to individual predictions across the dataset. The summary plot aggregates these contributions, showcasing the overall impact of each feature on the model's predictions.



```
explainer = shap.TreeExplainer(rf_model)
shap_values = explainer.shap_values(X_scaled)
shap.summary_plot(shap_values, X_scaled, feature_names=X.columns)
```

Figure:

Through the training, evaluation, and interpretability analysis of the Random Forest Regressor model, several key insights emerge:

- The visualization of predicted vs. actual prices and the RMSE metric collectively attest to the model's effectiveness in capturing the nuanced dynamics of stock market movements.

- The application of LIME provides valuable case-specific explanations, enhancing the transparency of individual predictions and fostering trust in the model's capabilities.

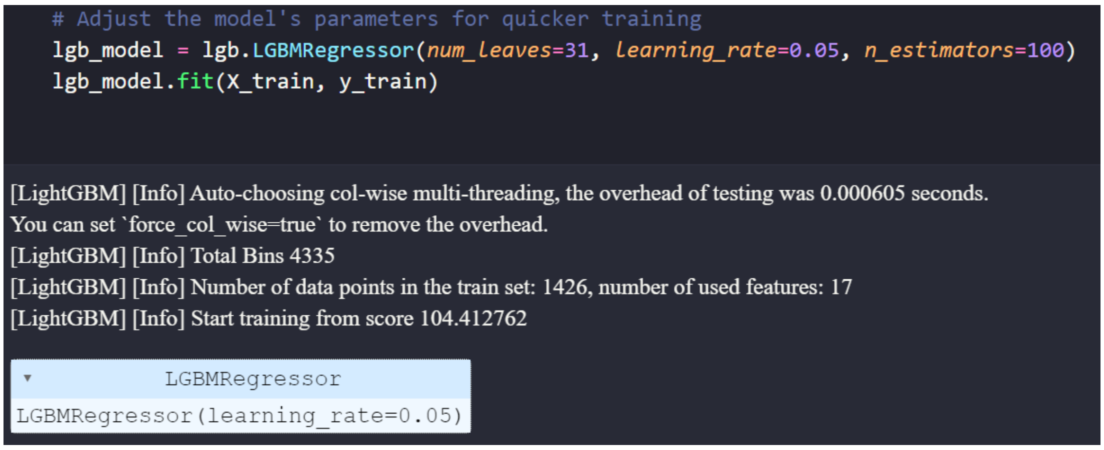
- SHAP's contribution analysis across the dataset reveals the overarching influence of certain features on stock price predictions, offering actionable intelligence that could guide investment strategies and further model refinements.

4. LightGBM Regressor:

The LightGBM Regressor, known for its speed and efficiency, particularly in handling large datasets with a high dimensionality, is employed to predict the adjusted closing prices of stocks. This gradient boosting framework is adept at capturing complex, non-linear relationships that typically characterize financial time series data.

Data Preparation: The dataset, filtered up to '2023-09-01', undergoes preprocessing to handle missing values through forward and backward filling, ensuring a consistent dataset for model input. The feature set excludes 'Date', 'Adj Close', and ‘Close’ to focus solely on predictors of stock price movements.

Model Training: The LightGBM model is configured with parameters aimed at optimizing performance for stock price prediction tasks, such as `num_leaves=31`, `learning_rate=0.05`, and `n_estimators=100`. The model is trained on all available data up to the cutoff point, excluding the last entry used for initial future prediction.



```
# Adjust the model's parameters for quicker training
lgb_model = lgb.LGBMRegressor(num_leaves=31, learning_rate=0.05, n_estimators=100)
lgb_model.fit(X_train, y_train)
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000605 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 4335
[LightGBM] [Info] Number of data points in the train set: 1426, number of used features: 17
[LightGBM] [Info] Start training from score 104.412762
```

```
▾          LGBMRegressor
LGBMRegressor(learning_rate=0.05)
```

Figure:

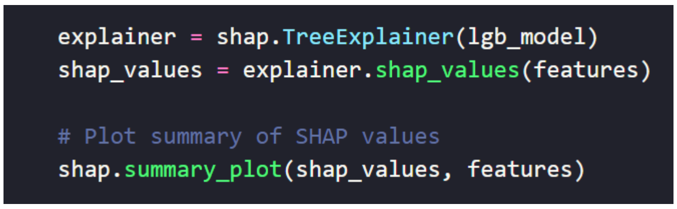
Future Prediction Simulation: Utilizing the trained model, predictions are made for the next 10 days based on the most recent data and features representing future market conditions. This iterative forecasting approach helps in understanding the model's adaptability to new information and its predictive capabilities over a short horizon.

Evaluation through Visualization: A comparison plot of actual vs. predicted prices for the forecasted period offers a visual assessment of the model's accuracy. This evaluation is supplemented by insights into the model's ability to track real-world price movements, adding a layer of qualitative evaluation to the quantitative analysis.

Explanation using XAI Techniques

To shed light on the decision-making processes underpinning the LightGBM model, both SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) are leveraged, providing a multifaceted view of model interpretability.

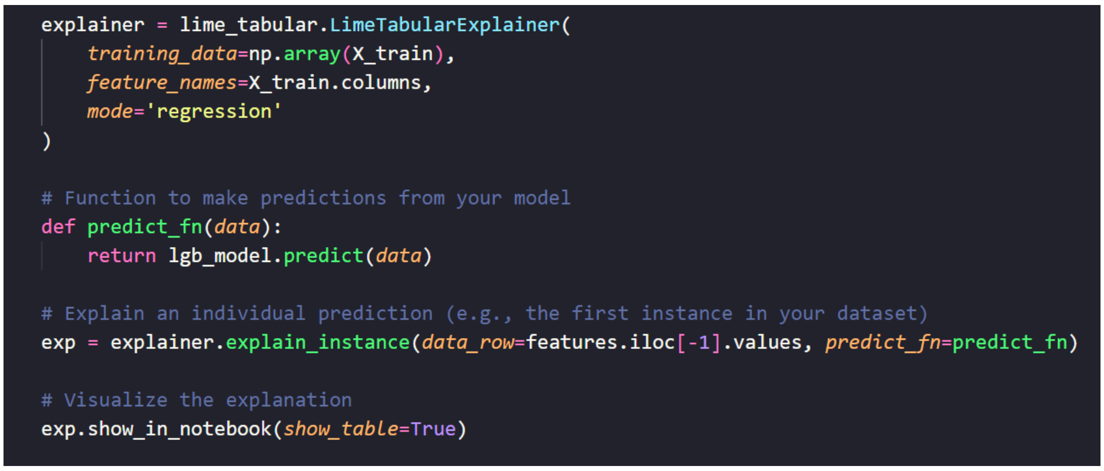
SHAP for Global Insights: Utilizing a `TreeExplainer`, SHAP values are computed for the features used in the LightGBM model. These values illustrate the contribution of each feature to the model's predictions, highlighting the direct and indirect relationships influencing stock prices. The SHAP summary plot aggregates these contributions, offering a global perspective on feature importance.



```
explainer = shap.TreeExplainer(lgb_model)
shap_values = explainer.shap_values(features)

# Plot summary of SHAP values
shap.summary_plot(shap_values, features)
```

Figure:

LIME for Local Interpretability: A `LimeTabularExplainer` is set up for regression analysis, focusing on providing interpretable explanations for individual predictions. By analyzing a specific instance (the last entry in the training set), LIME elucidates the model's rationale, pinpointing the features most influential to that particular prediction.



```python
explainer = lime_tabular.LimeTabularExplainer(
    training_data=np.array(X_train),
    feature_names=X_train.columns,
    mode='regression'
)

# Function to make predictions from your model
def predict_fn(data):
    return lgb_model.predict(data)

# Explain an individual prediction (e.g., the first instance in your dataset)
exp = explainer.explain_instance(data_row=features.iloc[-1].values, predict_fn=predict_fn)

# Visualize the explanation
exp.show_in_notebook(show_table=True)
```

Figure:

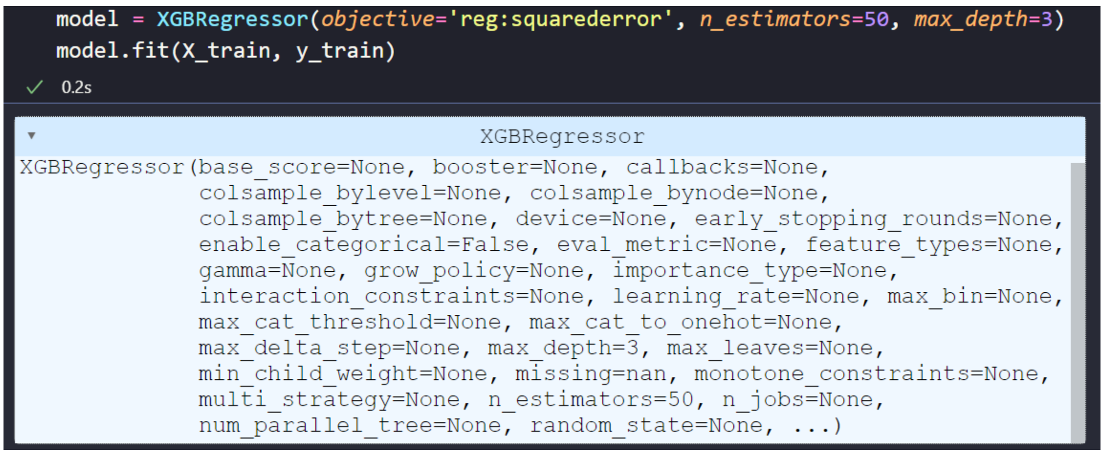
The implementation of the LightGBM model, complemented by the interpretability insights garnered through SHAP and LIME, offers a nuanced understanding of stock market forecasting. The model's performance, as visually evaluated, underscores its potential in accurately predicting stock price trends. Meanwhile, the application of XAI techniques enhances the transparency and trustworthiness of the predictions, allowing us to decipher the complex interplay of features that drive stock prices. This holistic approach not only validates the efficacy of the LightGBM model in financial forecasting but also underscores the value of interpretability in machine learning applications, fostering confidence in the model's utility and applicability in real-world scenarios.

5. XGBoost:

The XGBoost Regressor, renowned for its performance and efficiency in various machine learning competitions, is employed to forecast the adjusted closing prices of stocks. Leveraging gradient boosting, XGBoost is particularly adept at handling the complex and non-linear patterns typically observed in financial markets.

Data Preparation: Following the preprocessing steps to handle missing values and filtering the dataset based on the cutoff date ('2023-09-01'), features excluding 'Date', 'Adj Close', and ‘Close’ are used for training the model. This preparation ensures a clean and relevant dataset for the predictive task.

Model Training: The XGBoost model is configured with parameters tailored for the stock price prediction, including objective='reg:squarederror' to minimize squared errors, n_estimators=50 for the number of trees, and max_depth=3 to control the complexity of each tree. The model is trained on the dataset, excluding the last known entry to facilitate initial predictions.



```
model = XGBRegressor(objective='reg:squarederror', n_estimators=50, max_depth=3)
model.fit(X_train, y_train)
```

```
0.2s
```

```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=3, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=50, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

Figure:

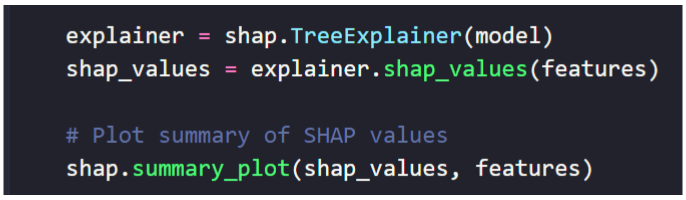
Future Prediction Simulation: Utilizing the most recent data, the model predicts the next day's stock price, and this process is iteratively applied to forecast prices for the subsequent days. This approach simulates real-world forecasting, where predictions are based on the latest available information.

Evaluation: A visual comparison between the model's predicted prices and the actual prices for the days following the training period provides an intuitive assessment of its accuracy. Additionally, the model's effectiveness is quantified through RMSE.

Explanation using XAI Techniques

The interpretability of the XGBoost model's predictions is enhanced through the application of SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations), providing insights into the model's decision-making process.

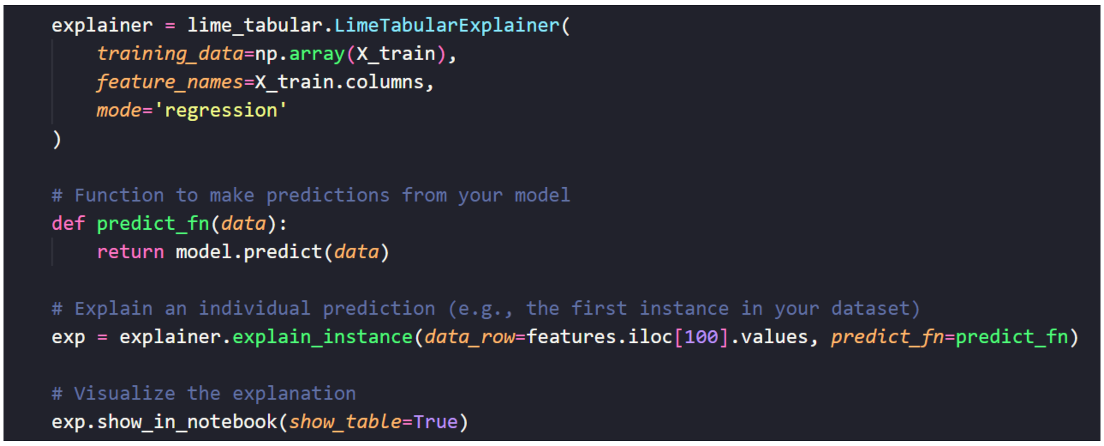
SHAP Explanations: By employing a **TreeExplainer**, SHAP values are computed for the model's features, illustrating the impact of each feature on the model's predictions. The summary plot aggregates these impacts, offering a global view of feature importance and how individual features influence the prediction outcomes.



```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(features)

# Plot summary of SHAP values
shap.summary_plot(shap_values, features)
```

Figure:

LIME Explanations: A LimeTabularExplainer configured for regression is used to explain individual predictions, shedding light on how the model arrives at its forecasts for specific instances. This local interpretability complements the global insights provided by SHAP, offering a detailed explanation for a selected prediction (e.g., the 100th instance in the dataset).



```
explainer = lime_tabular.LimeTabularExplainer(
    training_data=np.array(X_train),
    feature_names=X_train.columns,
    mode='regression'
)

# Function to make predictions from your model
def predict_fn(data):
    return model.predict(data)

# Explain an individual prediction (e.g., the first instance in your dataset)
exp = explainer.explain_instance(data_row=features.iloc[100].values, predict_fn=predict_fn)

# Visualize the explanation
exp.show_in_notebook(show_table=True)
```

Figure:

The deployment of the XGBoost model, accompanied by the interpretability analysis through SHAP and LIME, underscores the model's capacity to accurately predict stock market trends. SHAP's global explanations reveal the overall influence of features on the model's predictions, while LIME offers granular insights into individual predictions, enhancing our understanding of the model's behaviour.

5. Results

Before delving into the performance and interpretability insights for each model, it's crucial to highlight that the analysis encompasses data from three distinct sectors: tech, healthcare, and finance. This diversified approach aims to uncover not only the general effectiveness of each predictive model but also how their performance varies across sectors with different market dynamics and volatility levels. This section will present a comprehensive evaluation of each model's predictive accuracy, interpretability, and the implications of these findings, emphasizing the nuanced behaviours observed across sectors.

Tech stock: Apple Inc (AAPL)

Healthcare stock: Eli Lilly & Co. (LLY)

Finance stock: Blackrock, Inc. (BLK)

5.1 Model Performance and Explanations

Tech Sector

Model Comparison

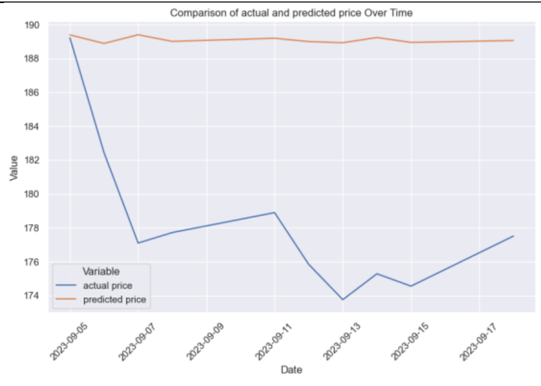


Figure: Linear Regression Model

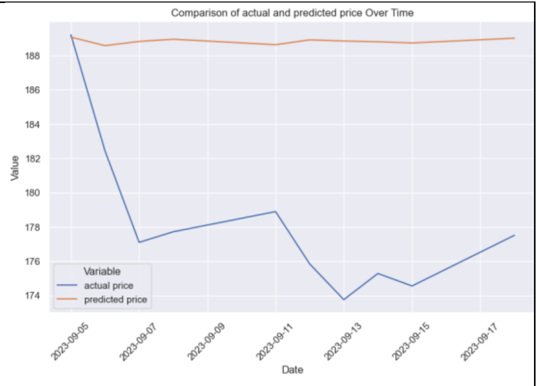


Figure: SVR Model

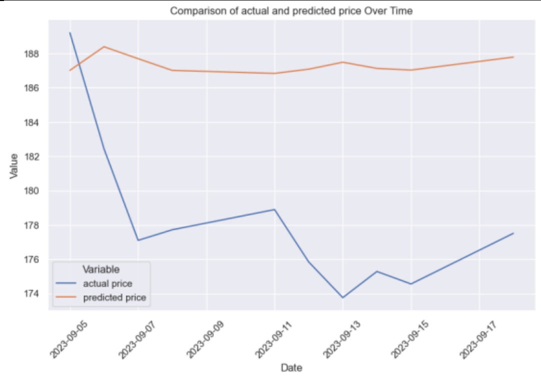


Figure: Random Forest Model

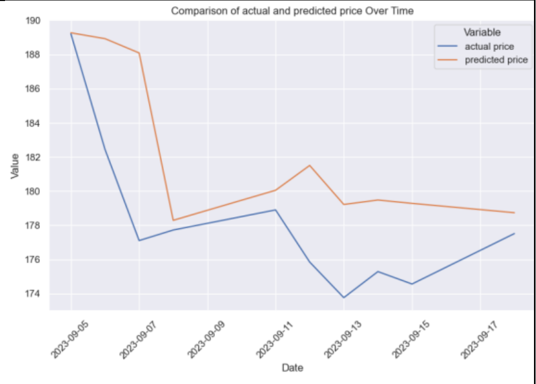


Figure: LightGBM Model

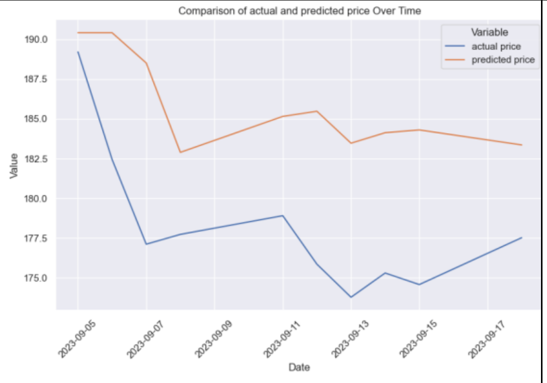


Figure: XGBoost Model

Model	RMSE Score
Linear Regression	11.676362551538794
Support Vector Regression	11.446465812329457
Random Forest	10.093235951934764
LightGBM	5.172562522839702
XGBoost	8.094390244796866

As we can observe from the models shown above, all of the models do a good job at predicting the first value. However, the simpler models like Linear Regression and SVR are unable to capture the variations and day to day fluctuations especially as time goes on. From these graphs, we can conclude that the two best models for predicting the prices of Tech stocks are LightGBM and XGBoost. Additionally, when looking at RMSE, we can see that they have the lowest error. Therefore, we will be explaining these models to obtain the important features.

SHAP Explanations:

The analysis conducted with SHAP elucidates the feature importance within our predictive models, particularly focusing on the LightGBM and XGBoost frameworks. The SHAP value distributions highlight SMA_50, SMA_200, and the MSFT stock as predominant features influencing the models' predictions. This pattern suggests a notable emphasis on momentum-based indicators alongside the performance metrics of analogous stocks in determining stock market trends in the near term. Such insights corroborate our initial theory, underscoring the critical role that momentum indicators and comparable stock performances play in shaping short-term market behaviour as interpreted by both LightGBM and XGBoost models. This observation not only affirms the hypothesized significance of these features but also sheds light on the nuanced understanding these models possess regarding the dynamics of stock market movements.

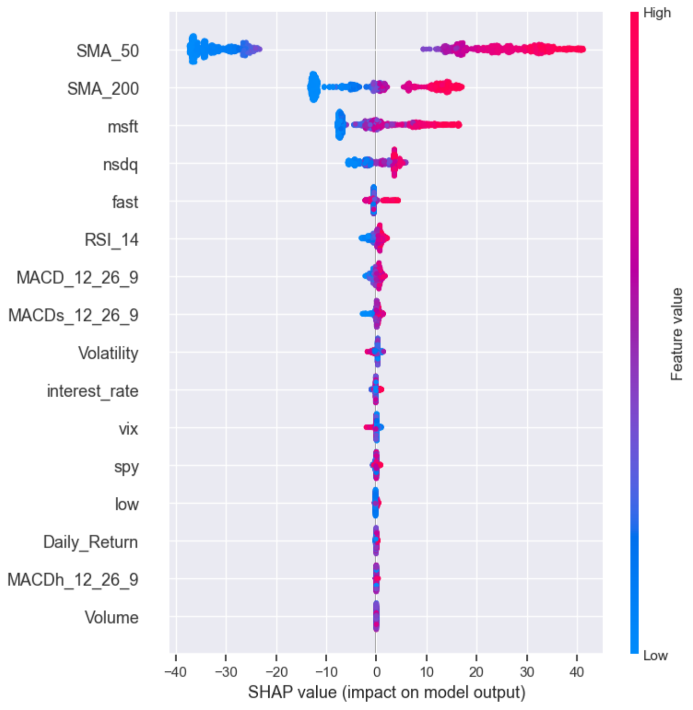


Figure:

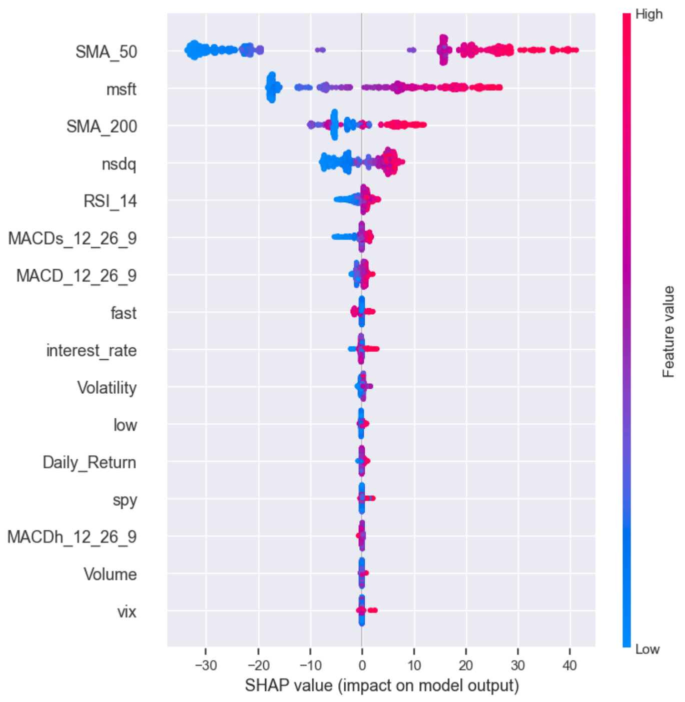
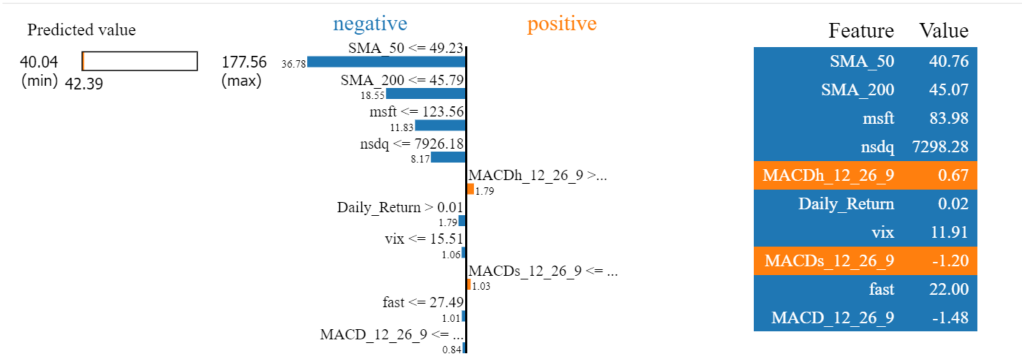


Figure:

LIME Explanations:

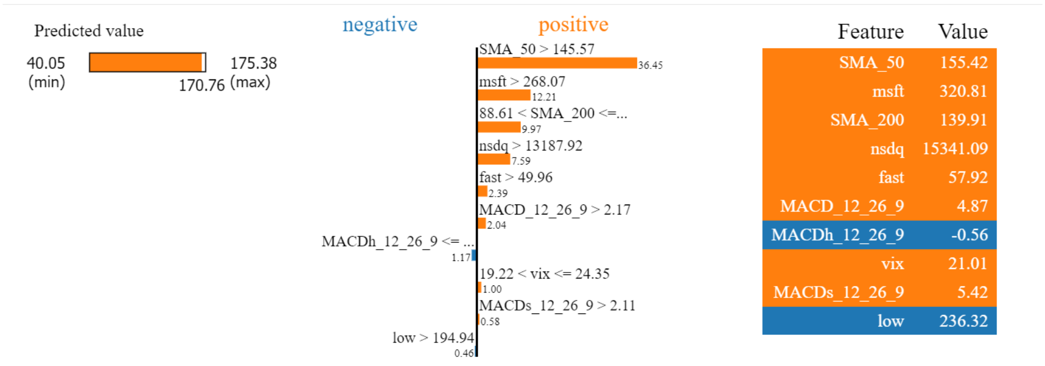
For the LIME explanations, we take 3 individual observations for each model and look at the most important features to draw insights.

LightGBM:



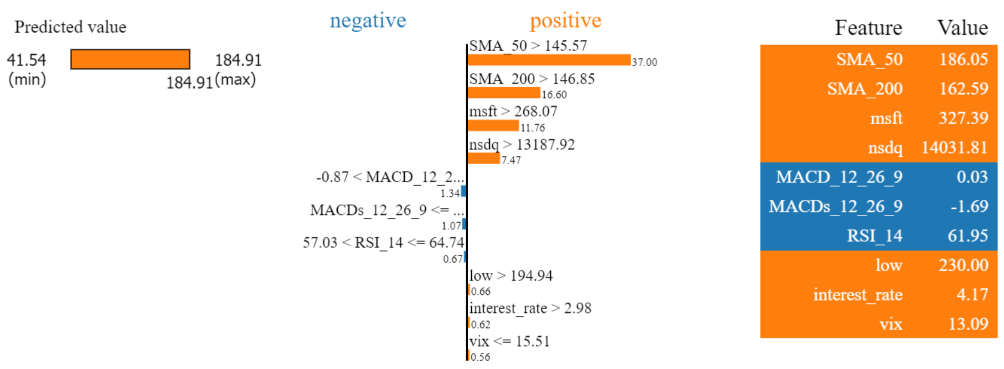
Feature	Value
SMA_50	40.76
SMA_200	45.07
msft	83.98
nsdq	7298.28
MACDh_12_26_9	0.67
Daily_Return	0.02
vix	11.91
MACDs_12_26_9	-1.20
fast	22.00
MACD_12_26_9	-1.48

Figure: observation 10



Feature	Value
SMA_50	155.42
msft	320.81
SMA_200	139.91
nsdq	15341.09
fast	57.92
MACD_12_26_9	4.87
MACDh_12_26_9	-0.56
vix	21.01
MACDs_12_26_9	5.42
low	236.32

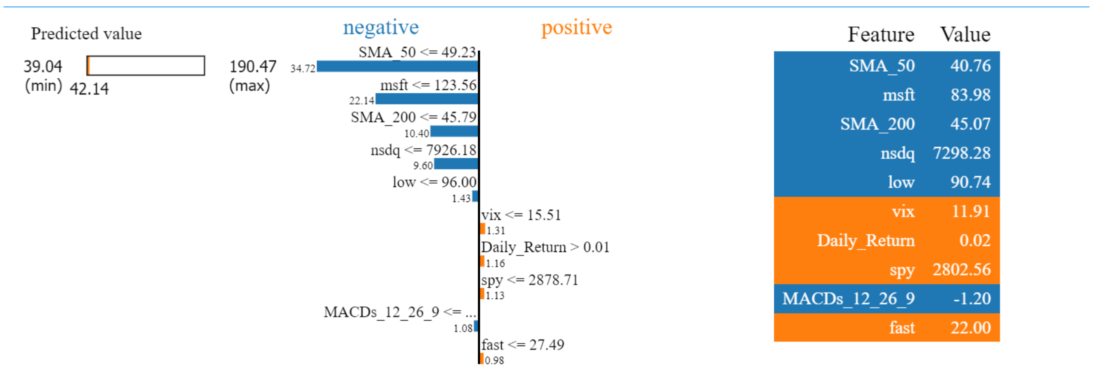
Figure: observation 1000



Feature	Value
SMA_50	186.05
SMA_200	162.59
msft	327.39
nsdq	14031.81
MACD_12_26_9	0.03
MACDs_12_26_9	-1.69
RSI_14	61.95
low	230.00
interest_rate	4.17
vix	13.09

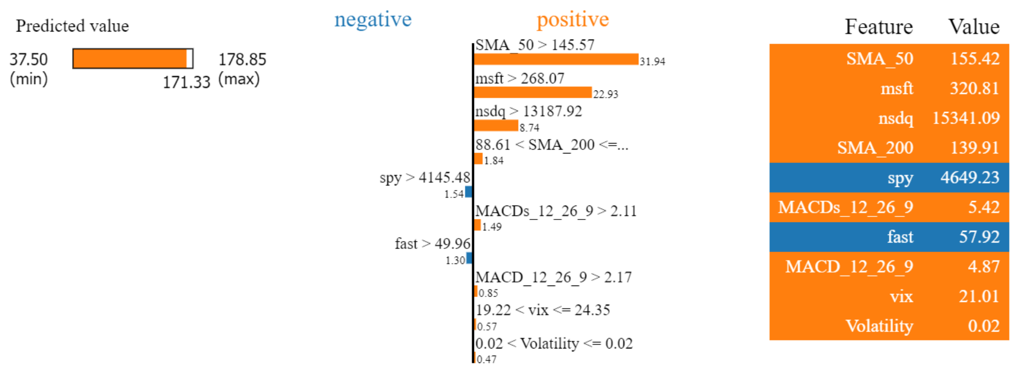
Figure: last observation

XGBoost:



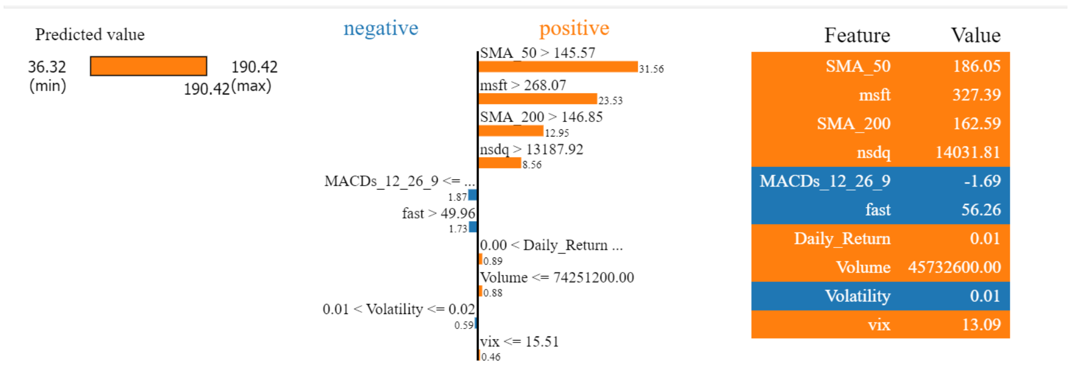
Feature	Value
SMA_50	40.76
msft	83.98
SMA_200	45.07
nsdq	7298.28
low	90.74
vix	11.91
Daily_Return	0.02
spy	2802.56
MACDs_12_26_9	-1.20
fast	22.00

Figure: observation 10



Feature	Value
SMA_50	155.42
msft	320.81
nsdq	15341.09
SMA_200	139.91
spy	4649.23
MACDs_12_26_9	5.42
fast	57.92
MACD_12_26_9	4.87
vix	21.01
Volatility	0.02

Figure: observation 1000



Feature	Value
SMA_50	186.05
msft	327.39
SMA_200	162.59
nsdq	14031.81
MACDs_12_26_9	-1.69
fast	56.26
Daily_Return	0.01
Volume	45732600.00
Volatility	0.01
vix	13.09

Figure: last observation

As we can observe in the LIME explanations, the features favouring momentum and analogous stocks remain the most predominant features.

Healthcare Sector

Model Comparison

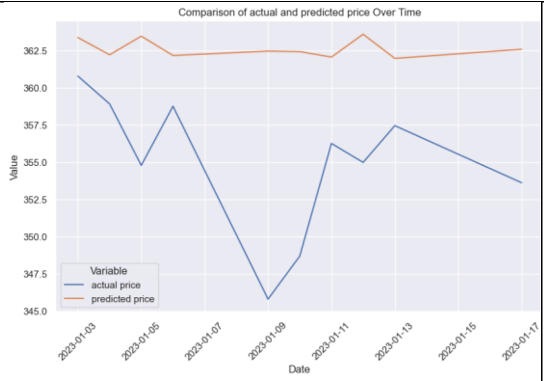


Figure: Linear Regression Model

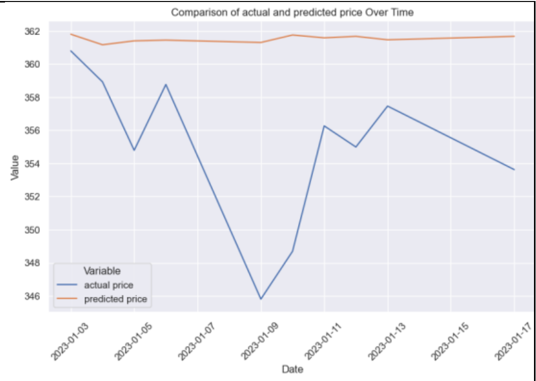


Figure: SVR Model

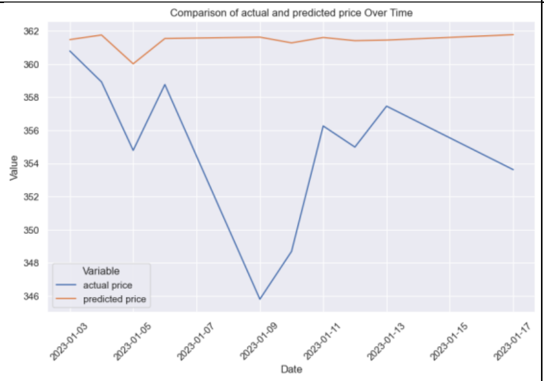


Figure: Random Forest Model

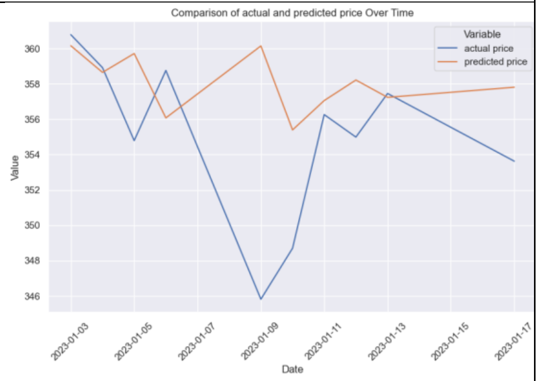


Figure: LightGBM Model

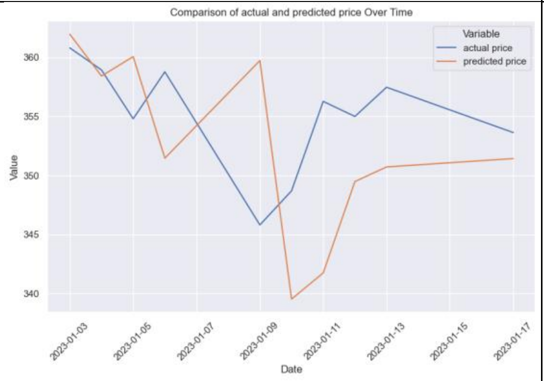


Figure: XGBoost Model

Model	RMSE Score
Linear Regression	8.83063232970614
Support Vector Regression	7.882031265830925
Random Forest	7.774597816260695
LightGBM	5.581581870084197
XGBoost	8.072791286747211

The models have a lower RMSE on average when compared to the tech stock dataset. Linear Regression, SVR, and Random Forest are still unable to capture the variations and daily movements. Once again LightGBM has the lowest RMSE but when looking at the graphs we can see that it is not as accurate as XGBoost when capturing the swings. There also appears to be some lag. However, both of these models are still the most accurate so they will be used to obtain the most important features for this sector.

SHAP Explanations:

SMA_200, SMA_50, and the Interest rate as predominant features influencing the models' predictions. While it is apparent that momentum-based indicators are still favoured, the presence of the interest rate feature (calculate from the 10-year interest rate on Treasury Bonds), is something completely different to the features obtained when training on the tech stock data. This pattern suggests a notable emphasis on momentum-based indicators alongside the interest rate and RSI in determining the stock market trends in the near term.

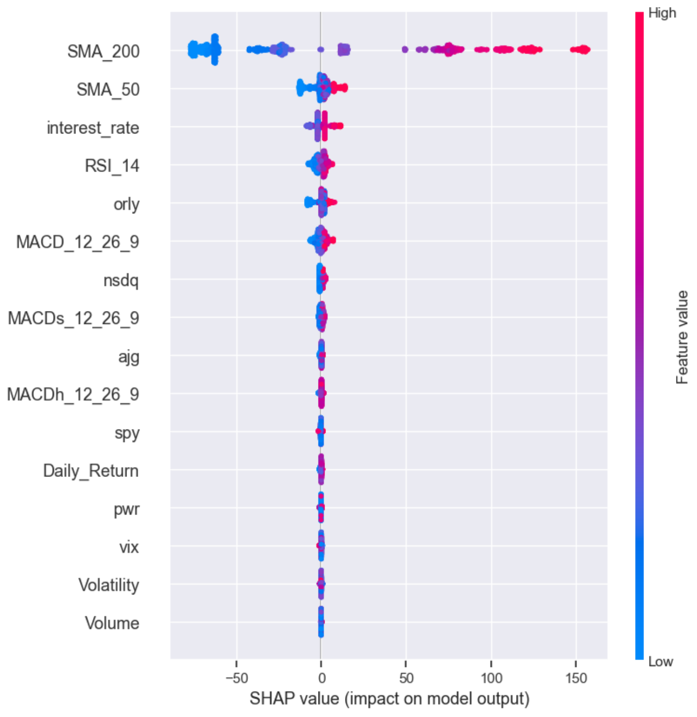


Figure:

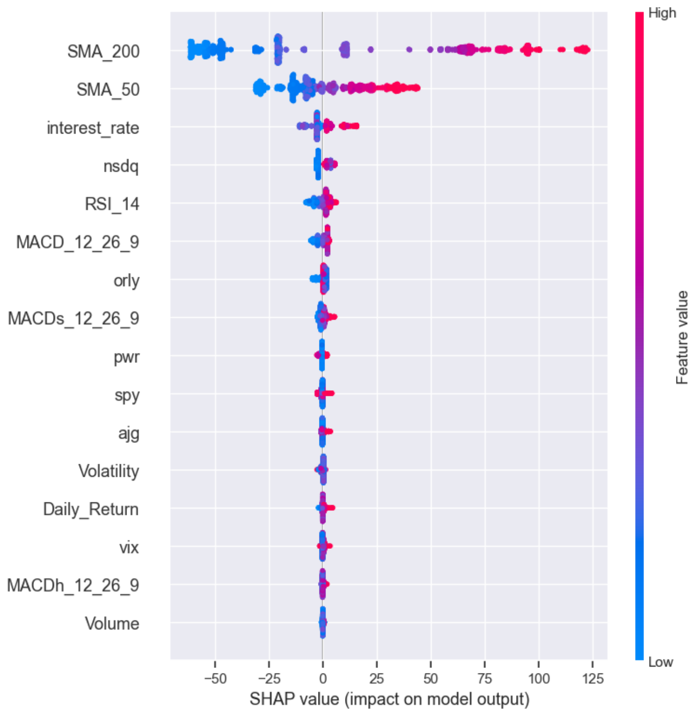
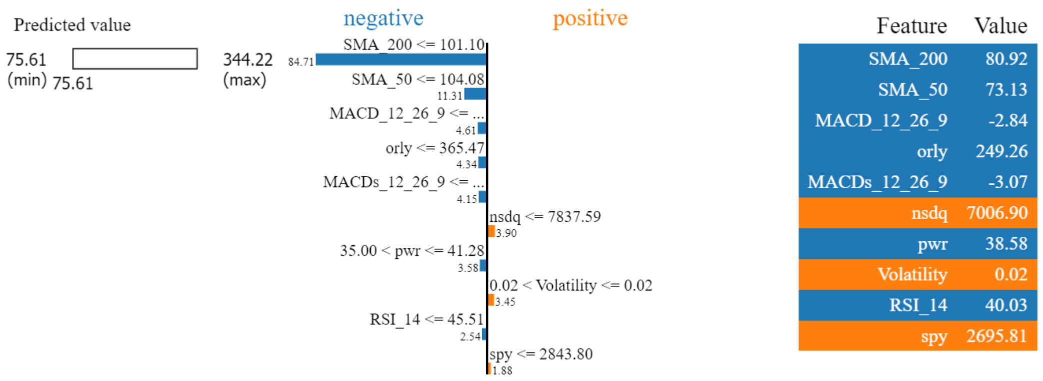


Figure:

LIME Explanations:

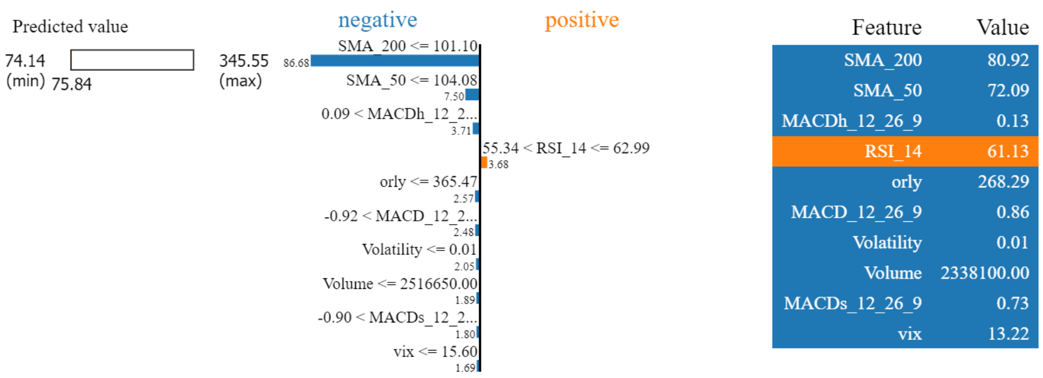
For the LIME explanations, we take 3 individual observations for each model and look at the most important features to draw insights.

LightGBM:



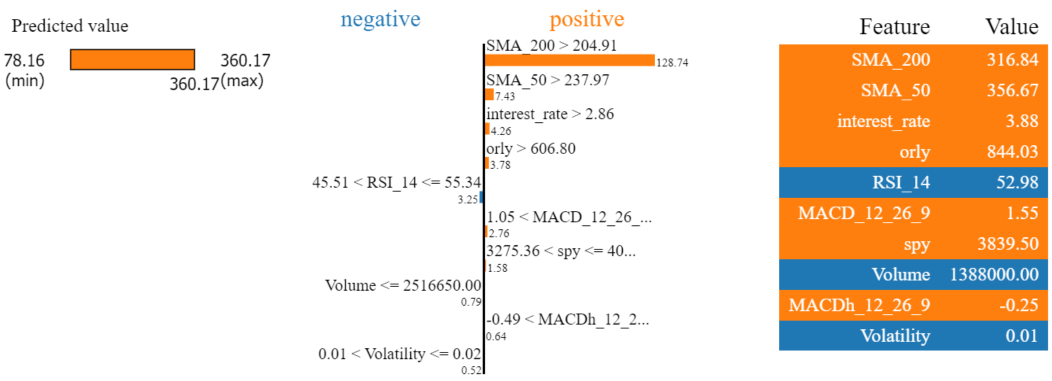
Feature	Value
SMA_200	80.92
SMA_50	73.13
MACD_12_26_9	-2.84
orly	249.26
MACDs_12_26_9	-3.07
nsdq	7006.90
pwr	38.58
Volatility	0.02
RSI_14	40.03
spy	2695.81

Figure: first observation



Feature	Value
SMA_200	80.92
SMA_50	72.09
MACDh_12_26_9	0.13
RSI_14	61.13
orly	268.29
MACD_12_26_9	0.86
Volatility	0.01
Volume	2338100.00
MACDs_12_26_9	0.73
vix	13.22

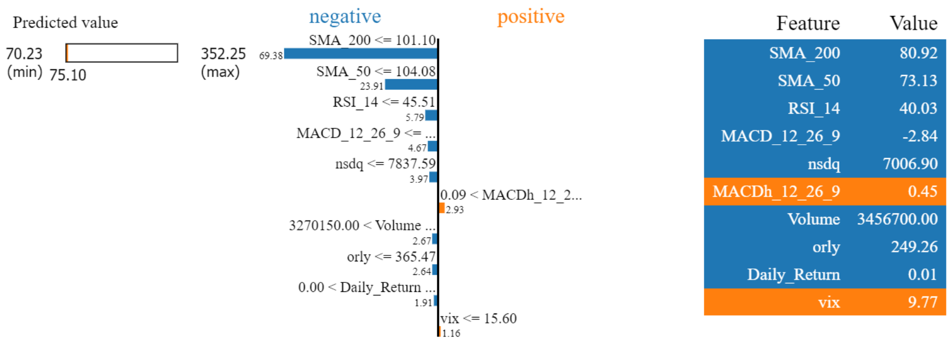
Figure: observation 100



Feature	Value
SMA_200	316.84
SMA_50	356.67
interest_rate	3.88
orly	844.03
RSI_14	52.98
MACD_12_26_9	1.55
spy	3839.50
Volume	1388000.00
MACDh_12_26_9	-0.25
Volatility	0.01

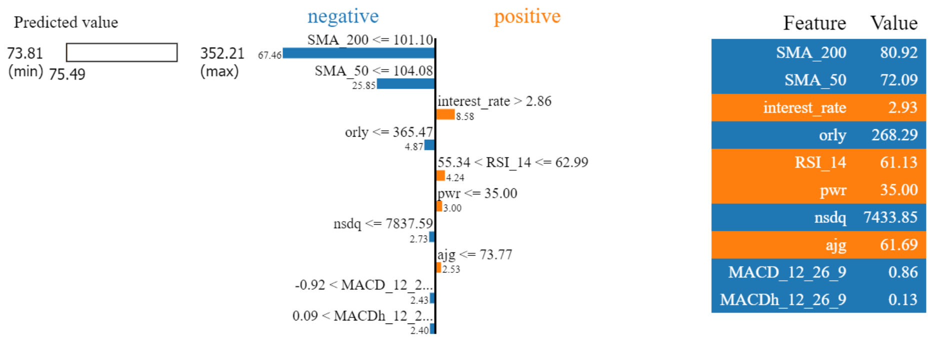
Figure: last observation

XGBoost:



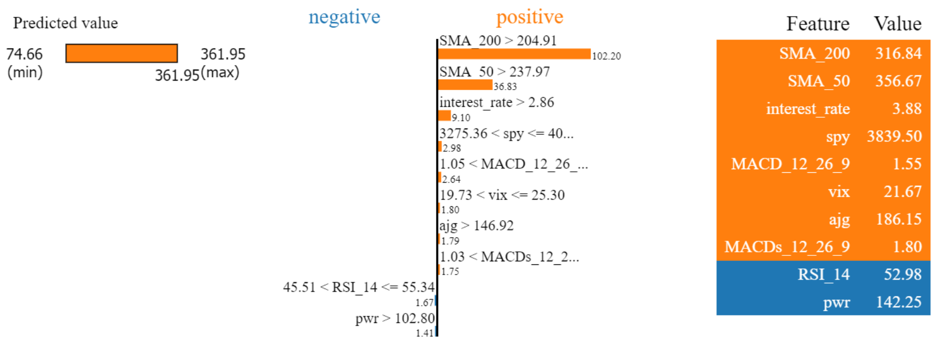
Feature	Value
SMA_200	80.92
SMA_50	73.13
RSI_14	40.03
MACD_12_26_9	-2.84
nsdq	7006.90
MACDh_12_26_9	0.45
Volume	3456700.00
orly	249.26
Daily_Return	0.01
vix	9.77

Figure: first observation



Feature	Value
SMA_200	80.92
SMA_50	72.09
interest_rate	2.93
orly	268.29
RSI_14	61.13
pwr	35.00
nsdq	7433.85
ajg	61.69
MACD_12_26_9	0.86
MACDh_12_26_9	0.13

Figure: observation 100



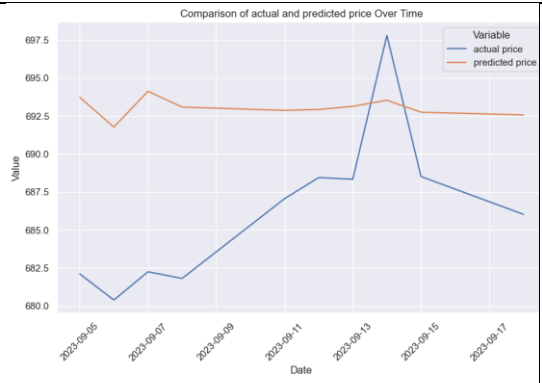
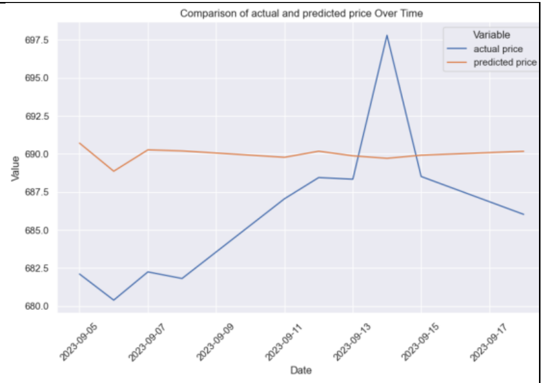
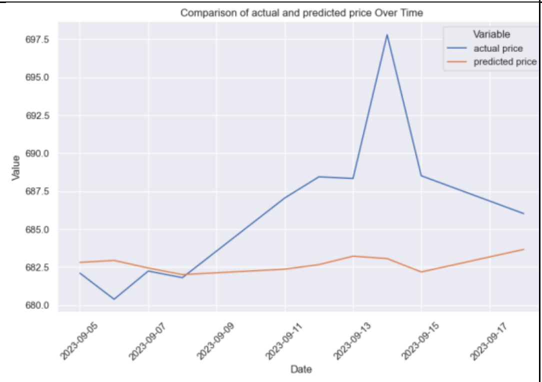
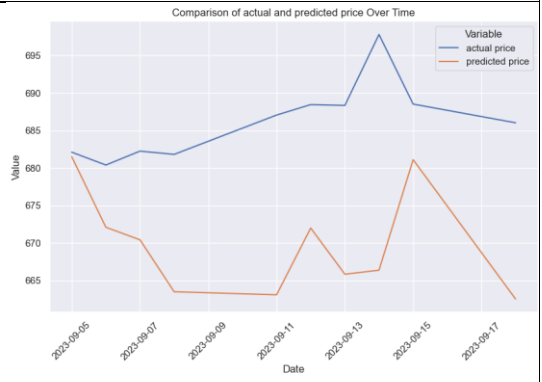
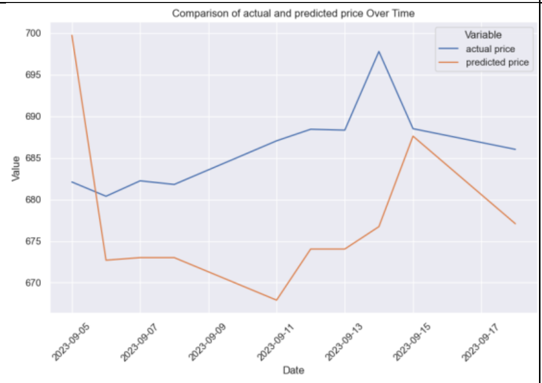
Feature	Value
SMA_200	316.84
SMA_50	356.67
interest_rate	3.88
spy	3839.50
MACD_12_26_9	1.55
vix	21.67
ajg	186.15
MACDs_12_26_9	1.80
RSI_14	52.98
pwr	142.25

Figure: last observation

An interesting thing to note when looking at the individual explanations in this case is that we see more variance among the top features with some having a strong emphasis on MACD (Moving Average Convergence Divergence), others having a strong emphasis on interest rate, and so on. However, the general trend of momentum-based indicators being the most important can still be seen.

Financial Sector

Model Comparison

 Figure: Linear Regression Model	 Figure: SVR Model
 Figure: Random Forest Model	 Figure: LightGBM Model
 Figure: XGBoost Model	

Model	RMSE Score
Linear Regression	8.297408575002134
Support Vector Regression	6.146274161040639
Random Forest	5.926072863305243
LightGBM	18.683602627515533
XGBoost	13.548833216179139

This is a good example of qualitative over quantitative results. While Linear Regression, SVR, and Random Forest all have significantly lower RMSE scores, they all fail to capture the swings and variance of the price. Their predictive performance in this sector appears to be better compared to the other sectors but still not to a satisfactory amount. Even though LightGBM and XGBoost have higher RMSE values, they act as much better predictors as can be observed in the graphs where they are much more accurately able to predict the day-to-day variation and general direction.

SHAP Explanations:

The prominent features identified include SMA_200, nxpi, avy, and spy. The identification of SMA_200 as a key feature reiterates the significance of momentum-based indicators in forecasting stock price trends. However, the presence of specific stocks like nxpi and avy, and indexes like the S&P 500 and Nasdaq shows that there is a substantial influence exerted by correlated stocks and major market indexes on the models' predictions. Such insights validate the hypothesis that in the context of financial sector predictions, a blend of momentum indicators, correlated stock performances, and the influence of major indexes like the S&P 500 and Nasdaq plays a pivotal role in short-term market forecasts. This approach aligns with the expectation that in financial sector analysis, broader market indicators and the performance of closely related stocks significantly impact prediction accuracy, offering a comprehensive view of market dynamics.

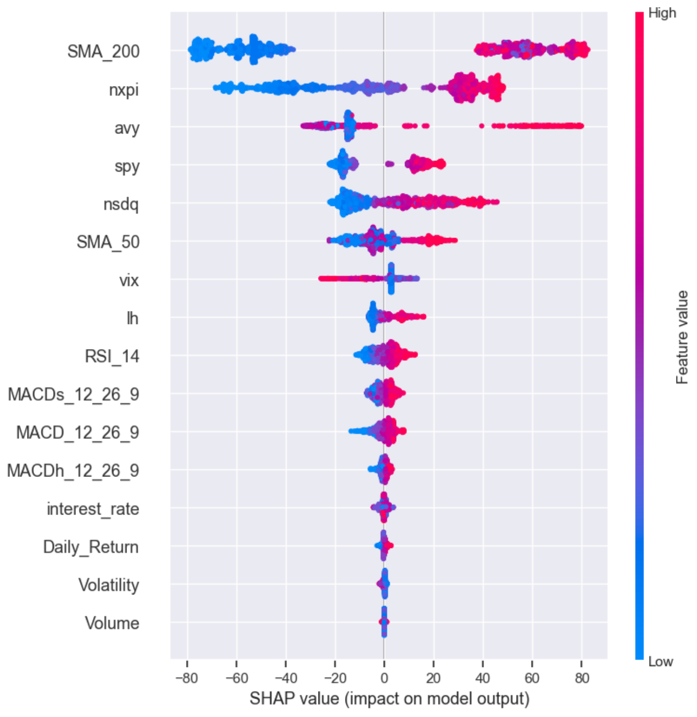


Figure:

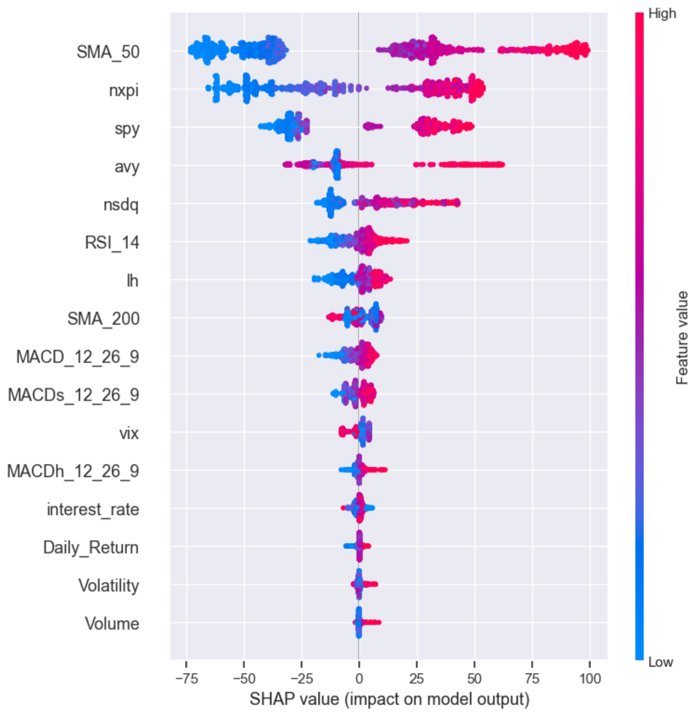
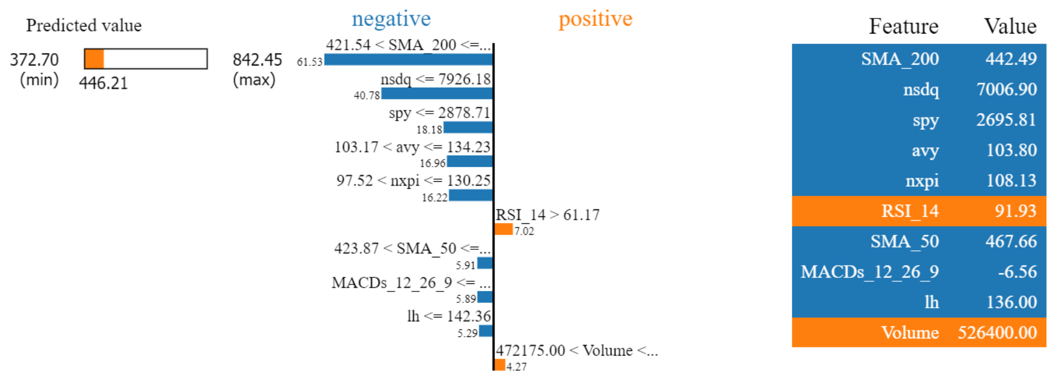


Figure:

LIME Explanations:

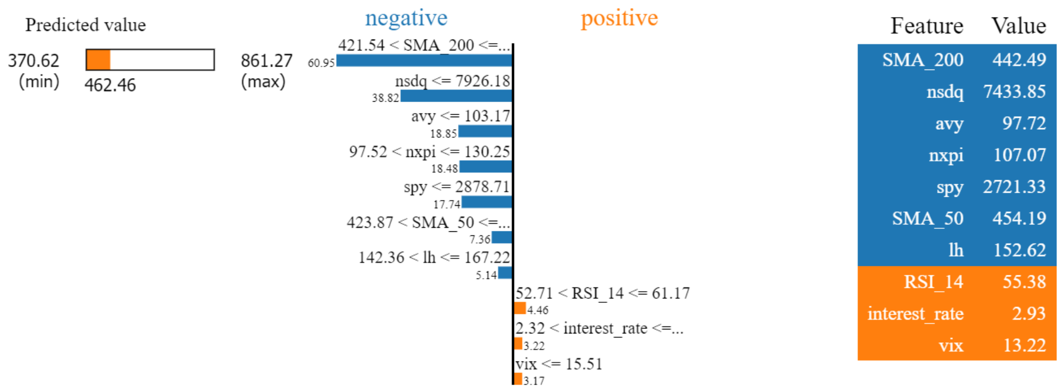
For the LIME explanations, we take 3 individual observations for each model and look at the most important features to draw insights.

LightGBM:



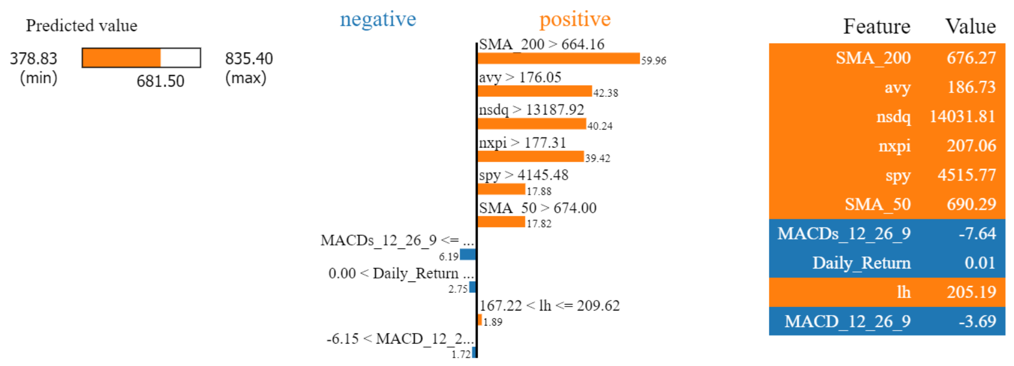
Feature	Value
SMA_200	442.49
nsdq	7006.90
spy	2695.81
avy	103.80
nxpi	108.13
RSI_14	91.93
SMA_50	467.66
MACDs_12_26_9	-6.56
lh	136.00
Volume	526400.00

Figure: first observation



Feature	Value
SMA_200	442.49
nsdq	7433.85
avy	97.72
nxpi	107.07
spy	2721.33
SMA_50	454.19
lh	152.62
RSI_14	55.38
interest_rate	2.93
vix	13.22

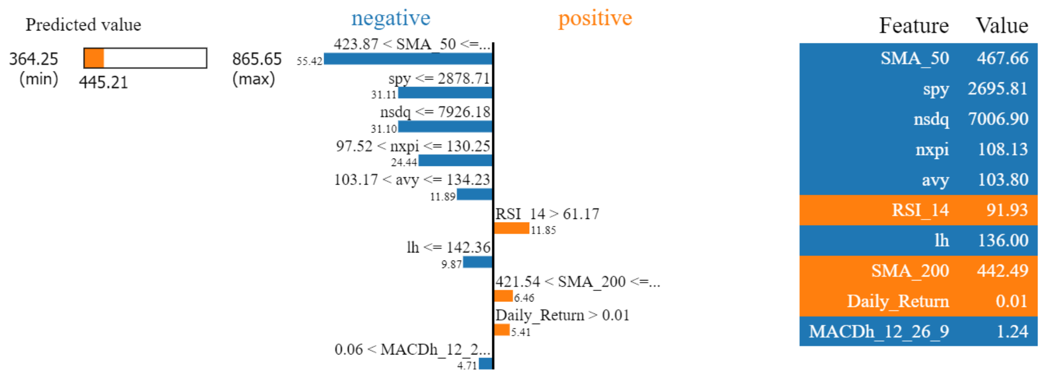
Figure: observation 100



Feature	Value
SMA_200	676.27
avy	186.73
nsdq	14031.81
nxpi	207.06
spy	4515.77
SMA_50	690.29
MACDs_12_26_9	-7.64
Daily_Return	0.01
lh	205.19
MACD_12_26_9	-3.69

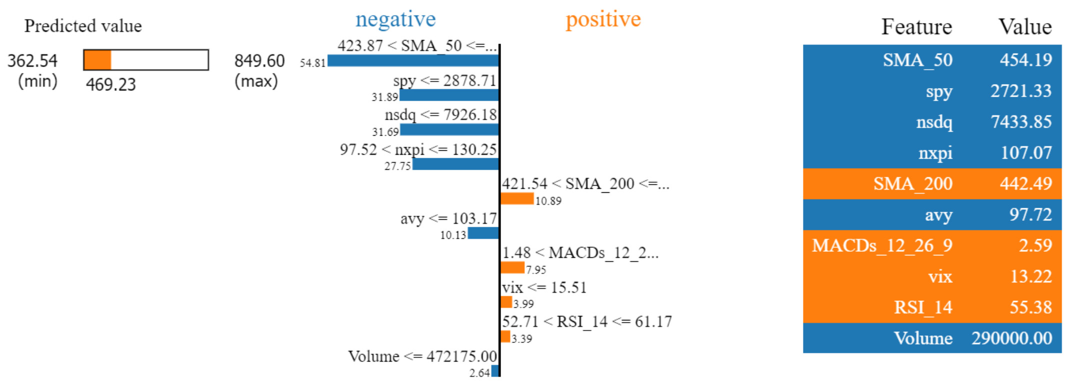
Figure: last observation

XGBoost:



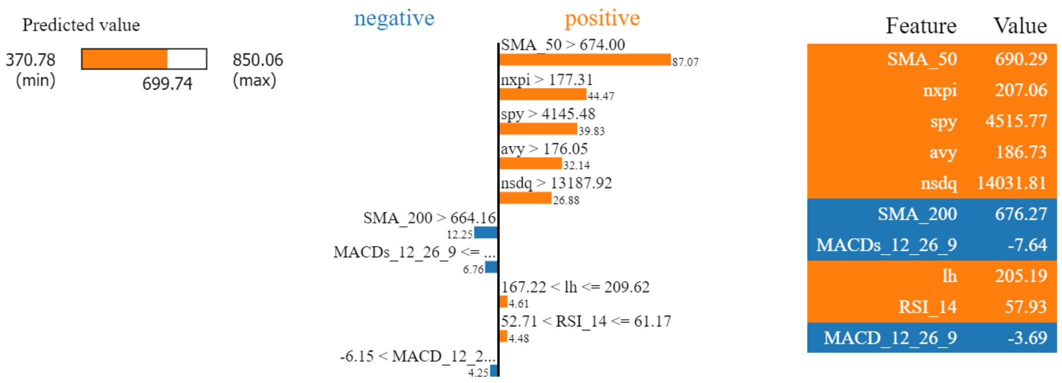
Feature	Value
SMA_50	467.66
spy	2695.81
nsdq	7006.90
nxpi	108.13
avy	103.80
RSI_14	91.93
lh	136.00
SMA_200	442.49
Daily_Return	0.01
MACDh_12_26_9	1.24

Figure: first observation



Feature	Value
SMA_50	454.19
spy	2721.33
nsdq	7433.85
nxpi	107.07
SMA_200	442.49
avy	97.72
MACDs_12_26_9	2.59
vix	13.22
RSI_14	55.38
Volume	290000.00

Figure: observation 100



Feature	Value
SMA_50	690.29
nxpi	207.06
spy	4515.77
avy	186.73
nsdq	14031.81
SMA_200	676.27
MACDs_12_26_9	-7.64
lh	205.19
RSI_14	57.93
MACD_12_26_9	-3.69

Figure: last observation

As we can observe in the LIME explanations, the features favouring momentum. analogous stocks, and indexes remain the most predominant features.

6. Conclusions and Recommendations

6.1 Conclusions

The comprehensive analysis across three distinct sectors—technology, healthcare, and financial—using various predictive models sheds light on the nuanced dynamics of stock market behaviour and the efficacy of machine learning models in capturing these intricacies. The investigation culminates in several key findings, highlighting the differential performance of models like LightGBM and XGBoost in comparison to simpler models such as Linear Regression and Support Vector Regression, and the significant role of SHAP and LIME in elucidating model predictions.

Model Performance Across Sectors

In the **technology sector**, the superior performance of LightGBM and XGBoost models underscores their ability to navigate the complex and volatile nature of tech stocks. Their lower RMSE scores and adeptness at capturing day-to-day fluctuations demonstrate their robustness, especially when contrasted with the simpler Linear Regression and SVR models, which struggle with the sector's variability.

The **healthcare sector** presents a scenario where, despite lower overall RMSE scores, the intricate movements and swings of stock prices are best captured by LightGBM and, to a lesser extent, XGBoost. This sector's analysis reveals a preference for models that balance between capturing swift market changes and maintaining predictive accuracy, with a slight edge towards XGBoost for its nuanced capture of price swings despite LightGBM's lower RMSE.

In the **financial sector**, the qualitative assessment becomes crucial. Despite higher RMSE values, LightGBM and XGBoost emerge as more capable of accurately predicting the sector's pronounced day-to-day variance and directional trends, outperforming Linear Regression, SVR, and Random Forest models that, while showing lower RMSE scores, fail to grasp the market's dynamism effectively.

SHAP and LIME Explanations

SHAP analyses reveal a consistent emphasis on momentum-based indicators across sectors, with the addition of sector-specific stocks and market indexes playing a pivotal role in model predictions. This pattern confirms the hypothesis that not only do momentum indicators such as SMA_200 hold significant predictive power, but the performance of correlated stocks and broader market trends are also crucial, especially in the financial sector where broader market indicators and the performance of closely related stocks significantly influence predictions.

LIME explanations further enrich our understanding by offering observation-specific insights. Across models and sectors, the prominence of momentum indicators, analogous stocks, and, notably in the financial sector, market indexes, highlights the multifaceted nature of stock market predictions. These explanations validate the selected models' capacity to discern the critical drivers of stock price movements, offering a granular perspective that aligns with broader market dynamics.

Recommendations

While the insights derived from this comprehensive study are not to be construed as direct financial advice, the findings offer valuable perspectives for individuals and traders navigating the stock market. Based on the experiment's outcomes and the analysis of various machine learning models across different sectors, several recommendations emerge, underscoring the predictive power of certain features in stock market behaviour.

Emphasis on Momentum-Based Indicators

The consistent prominence of momentum-based indicators, such as SMA_200 and SMA_50, across all models and sectors, underscores their significant role in forecasting stock prices. These indicators, capturing the ongoing trends and velocities of stock movements, provide a reliable foundation for understanding market dynamics. Investors and traders might benefit from incorporating these indicators into their analysis toolkit, utilizing them as key signals for making informed decisions about market entries and exits.

Importance of Analogous Stocks

The influence of analogous stocks' performances—evident from the models' prioritization of specific stocks like MSFT, nxpi, and avy—highlights the interconnected nature of the stock market. This observation suggests that traders should not only monitor the stocks of interest but also keep a close eye on comparable stocks within the same sector or those exhibiting similar market behaviours. Such a strategy can offer additional insights into potential market movements, enabling a more nuanced understanding of sector-specific trends and external factors impacting stock prices.

Sector-Specific Considerations

While momentum indicators and analogous stocks form the cornerstone of predictive analysis, slight variations based on the industry are crucial. For instance, the financial sector's analysis reveals the significant impact of broader market indexes on stock predictions. This insight suggests that traders operating within the financial sector should also weigh the influence of major indexes like the S&P 500 and Nasdaq in their decision-making processes. Similarly, in sectors like healthcare, attention to sector-specific economic indicators or regulatory changes can provide additional layers of predictive accuracy.

Final Thoughts

The endeavour to harness the capabilities of machine learning for stock market prediction is a testament to the ongoing evolution of financial analytics. As individuals and traders seek to navigate the complexities of the market, the insights garnered from this study emphasize the importance of a balanced approach that incorporates both technical indicators and a keen understanding of market interrelations. While the recommendations provided herein draw upon the empirical findings of our analysis, it is imperative for each investor to conduct their own due diligence, considering the broader economic landscape, market conditions, and individual risk tolerance before making investment decisions.

6.2 Recommendations for future work

The exploration of machine learning models in predicting stock market trends has yielded insightful findings and highlighted the robustness of certain models over others in various market sectors. To build upon this foundational work and further enhance the predictive accuracy and applicability of these models, several avenues for future research are recommended:

Building an Ensemble Model

Future studies could benefit from developing an ensemble model that combines the strengths of individual models explored in this study, such as LightGBM, XGBoost, and others. An ensemble approach, leveraging the predictive capabilities of multiple models, may offer a more nuanced understanding of stock price movements by capturing a broader spectrum of patterns and anomalies within the data. This method could potentially lead to a more robust model that performs well across different sectors and market conditions.

Utilizing Data from Different Time Horizons

Expanding the dataset to include features derived from various time horizons—such as quarterly and monthly data—could provide a more comprehensive view of the market's dynamics. Incorporating these varied timeframes might uncover longer-term trends and cyclical patterns not evident in daily data alone, thereby enhancing the model's ability to predict long-term price movements and offering investors insights into both short-term fluctuations and long-term investment strategies.

Exploring Long-Term Price Movement

While the current study focuses primarily on short-term market behaviour, future work should explore models specifically tailored to predicting long-term price movements. This could involve adjusting the feature set to include more macroeconomic indicators, longer-term momentum indicators, or sector-wide trends that influence stock prices over extended periods. Understanding the factors driving long-term movements could provide valuable insights for long-term investors and portfolio managers.

Repeating the Models Under Various Market Conditions

Conducting further experiments to test the models under different market conditions—such as bull markets, bear markets, and periods of recession—could reveal valuable information about the models' adaptability and resilience. This approach would help identify which models are not only accurate in a specific market environment but also versatile enough to adjust to changing market dynamics. Such insights could be crucial for developing trading strategies that remain effective regardless of the market's state.

Adding Sentiment Data

Incorporating sentiment data derived from news articles, social media, and financial reports could add a layer of qualitative analysis to the quantitative models. Sentiment analysis has the potential to capture the market's emotional response to events, announcements, and economic indicators, providing a predictive edge that complements traditional data sources. Exploring the integration of sentiment data could uncover new correlations between public sentiment and stock market movements, offering a more holistic view of the factors influencing stock prices.

By pursuing these recommendations for future work, researchers can continue to advance the field of stock market prediction.