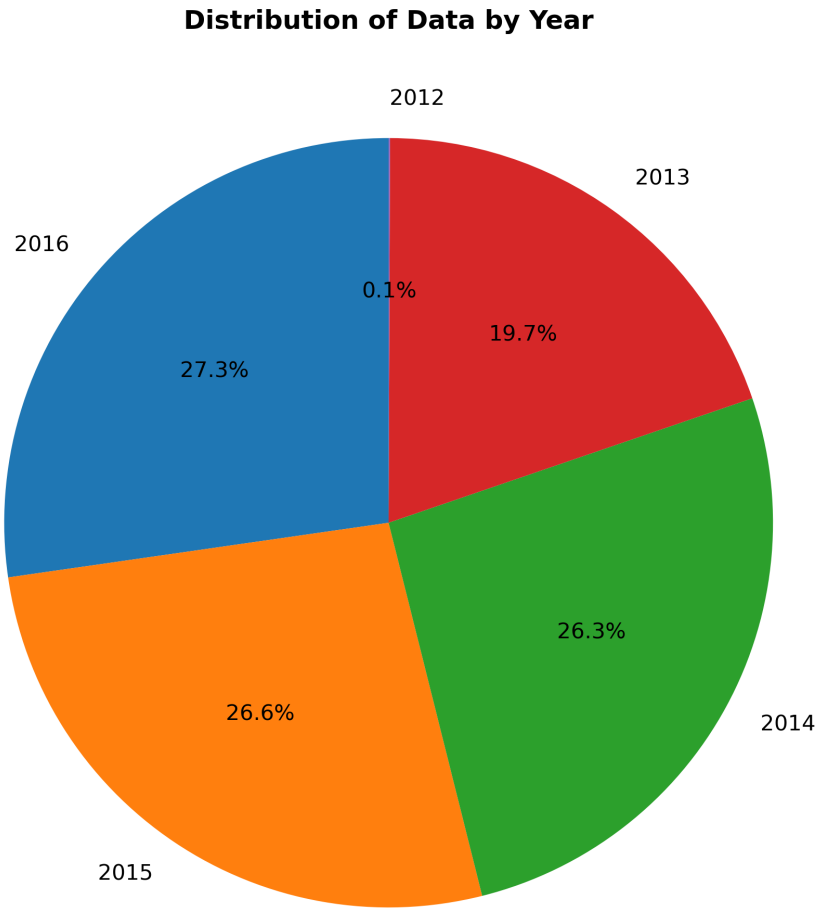
Predicting S&P500 Closing Stock Prices

Stefano Chiappo
Brown University
https://github.com/chistefano/Final_DATA1030.git

Introduction

Throughout this project, my aim was to predict the closing prices of S&P500 stocks, something that is crucial in making informed investment decisions. This is a regression problem whose target variable is the closing price of a stock. I created my dataset by merging the three datasets of Prices.csv, Fundamentals.csv, and Securities.csv, which provided data on stock prices, quarterly financial metrics, and the operating industry of the stock-issuing company, respectively. The merged dataset spanned the years 2012 to 2016 and contained 101,198 points with 81 features before splitting and preprocessing.

Data Distribution by Year in Dataset

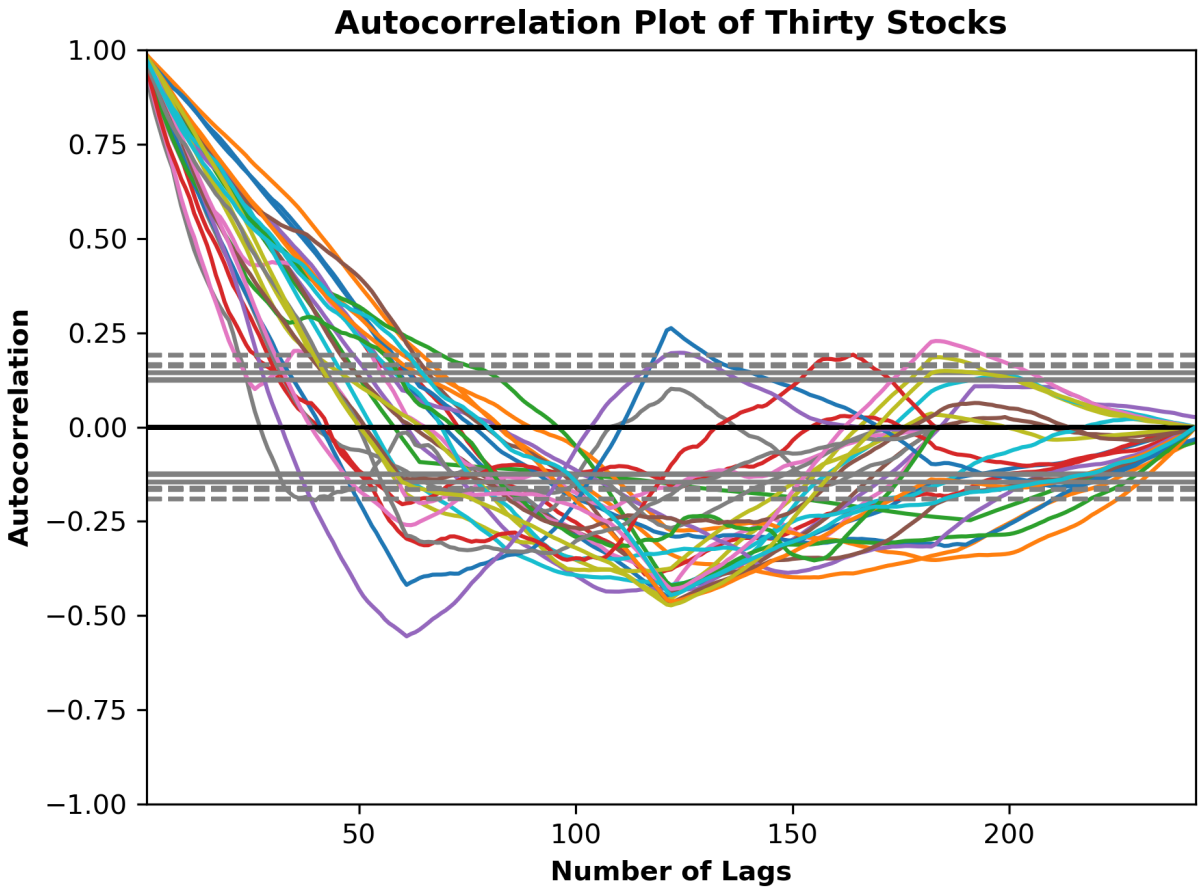


The price dataset used in my project has also been used in similar studies aiming to predict stock prices. For instance, Yassine Sfaihi, a Junior Data Scientist, developed a Long Short-Term Memory (LSTM) network to predict stock prices solely based on historical prices, achieving an RMSE of 0.00129 dollars. While this deep learning approach suggests a high level of accuracy, I anticipate that my model, although potentially less accurate, will be more interpretable due to its simpler machine learning framework. Additionally, the low RMSE in Sfaihi's model raises the possibility of data leakage, which I will try and prevent when carrying out data processing. Similarly, Data Science student Izam Mohammed also used the same dataset focused on lagged prices to also train an LSTM network, which predicted future stock prices with a RMSE of 2.65 dollars. The success of both studies in using lagged prices to forecast stock prices highlights the critical role of lagged prices in predicting my target variable, which is an insight that I used in my analysis.

EDA

Before training my models, I created various visualisations that would help me better understand my data.

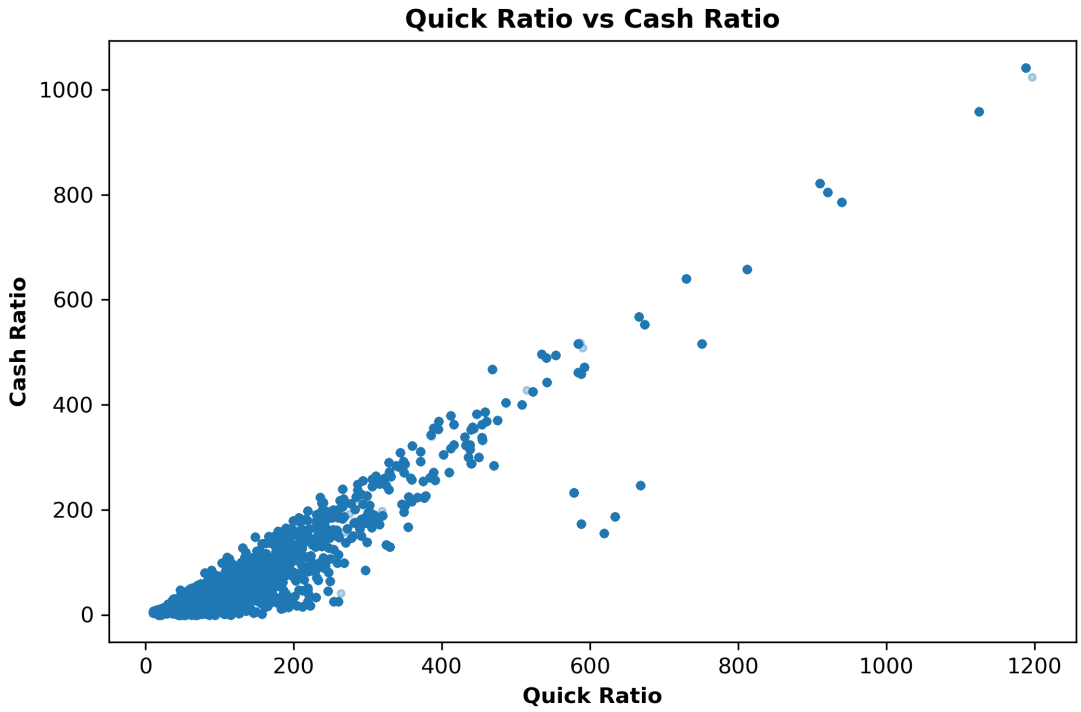
Target Variable Autocorrelation for Thirty Stocks



This graph shows the autocorrelation of the closing price for thirty stocks. Initially, we observe a consistent pattern of strong autocorrelation, indicating a strong relationship between consecutive price points. However, as we increase the lag period, this autocorrelation diminishes, tending to zero. This trend highlights the importance of recent stock price data in forecasting future closing prices. Based on this analysis, I have chosen to incorporate the closing prices from the past five days as lagged features in my model. By doing this, I was using the most relevant data to predict my stock price, boosting the model's ability to accurately predict my target variable.

Due to the large number of features, I decided to use a Pearson Correlation Matrix to determine some features that were highly correlated and I could drop.

Highly Correlated Features of Quick Ratio and Cash Ratio Plotted



In my analysis, 'Cash Ratio' and 'Current Ratio' were removed, each having around a 0.93 correlation with the 'Quick Ratio' feature. The removal of highly correlated variables will speed the convergence of our future machine learning models as well as increase their interpretability, allowing for a clearer understanding of the model without a significant decrease in its predictive power.

Methods

After dropping the highly correlated features, I had to drop the points that did not contain information regarding our target variable or the lagged target variable, as I could not use them for my model's predictions. After doing so, I was left with three features that had missing values: Quick Ratio, Earnings Per Share and Estimated Shares Outstanding. I then used IterativeImputer with LinearRegression estimator to handle these missing values as I knew these features were linearly correlated to other features in my dataset.

Due to the time series nature of my data, I had to ensure that when I split my dataset I did not allow for data leakage. Data leakage would occur if a model was trained on more recent stock price information of a company and then had to predict the older prices as a test set.

To prevent this from happening, I used the 'Date' and 'Company Name' features to divide the information for each individual company in ascending time order (least recent stock prices first). I then assigned the last 20% (the most recent 20%) of the points of each company's stock price to the test set. This ensured that my model would be trained on the older stock prices and would be tested on the newer ones.

The features in my dataset were all continuous, except for the company name and date, and the sector of the company which is a categorical variable. The continuous variables were preprocessed using StandardScaler, whereas 'GICS Sector', the only categorical variable, was processed using OneHotEncoder. The "Company Name" and "Date" features were used for splitting and lagging data but were dropped before they could be preprocessed as they were not used to predict the stock price.

I used a Predefined Split to divide the remaining data into five validation and training set combinations for cross-validation.This split, similarly to the previous one that created the test sets, ensured that the older values for closing stock prices were used in the validation set and the newer ones in the training set to prevent data leakage.

I then created a machine learning pipeline that would allow me to test my five different machine learning models. I also used this pipeline for cross validation, hyperparameter tuning and to calculate the RMSE for each model, in order to determine which one would best predict the target variable. RMSE was chosen as this is a regression problem and because RMSE is a metric that is easy to interpret due to it having the same unit as our target variable.

The machine learning models used and the hyperparameters we tuned are shown in the table below.

Models Used and Hyperparameters Tuned

Machine Learning Model	Tuned Hyperparameters	Hyperparameter Values Used
Lasso	Alpha	np.logspace(-10, 1, 10)
Ridge	Alpha	np.logspace(-10, 1, 10)
Random Forest Regression	n_estimators max_depth	[1, 3, 10, 30, 50, 100] [1, 3, 10, 20, 30]
K Neighbors Regression	n_neighbors Weights	[1, 10, 30, 50, 100] ['uniform', ‘distance']
xgBoost	n_estimators learning_rate max_depth	[100, 200, 300] [0.01, 0.1, 0.2] [3, 4, 5]

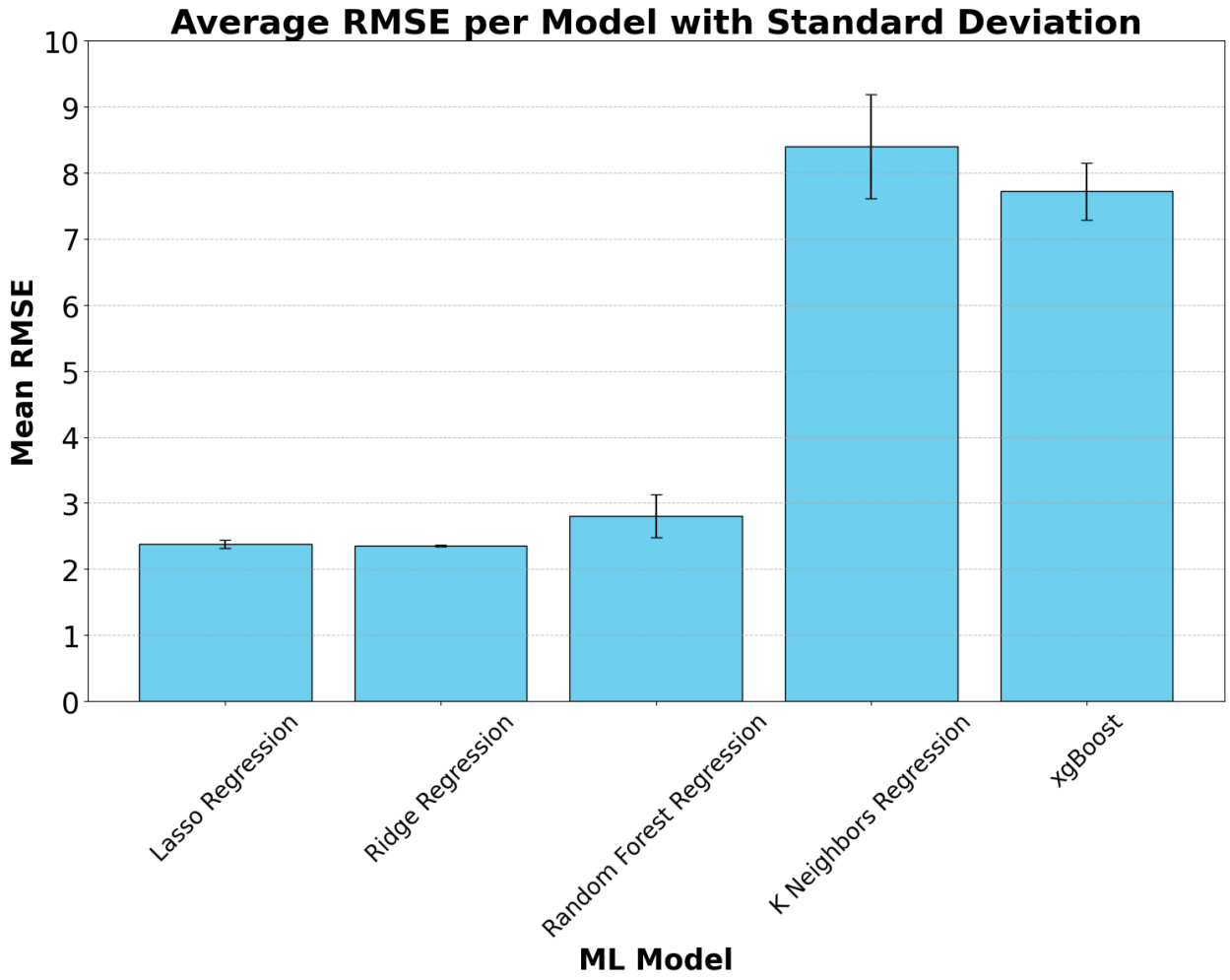
Results

In order to determine how well my models performed, I calculated a baseline score where I predicted the average price of a stock for each datapoint. I then calculated this model's RMSE and found it to be 89.45. I then compared this figure to the average RMSE of the other models, to see if my models had a stronger predicting power than the baseline model. The average RMSE and standard deviation calculated for each hyperparameter tuned model as well as the Baseline model is shown below.

Model	Hyperparameters Tuned	Average RMSE (in $)	Standard Deviation Average RMSE (in $)
Baseline	None	89.47	0.00
Lasso	Alpha	2.38	0.06
Ridge	Alpha	2.35	0.09
Random Forest Regression	n_estimators max_depth	3.16	0.45
K Neighbors Regression	n_neighbors Weights	8.40	0.79
xgBoost	n_estimators learning_rate max_depth	7.72	0.43

As we can see from the table above, all five machine learning model with tuned hyper parameters were better at predicting the closing price of a stock than the baseline model was.

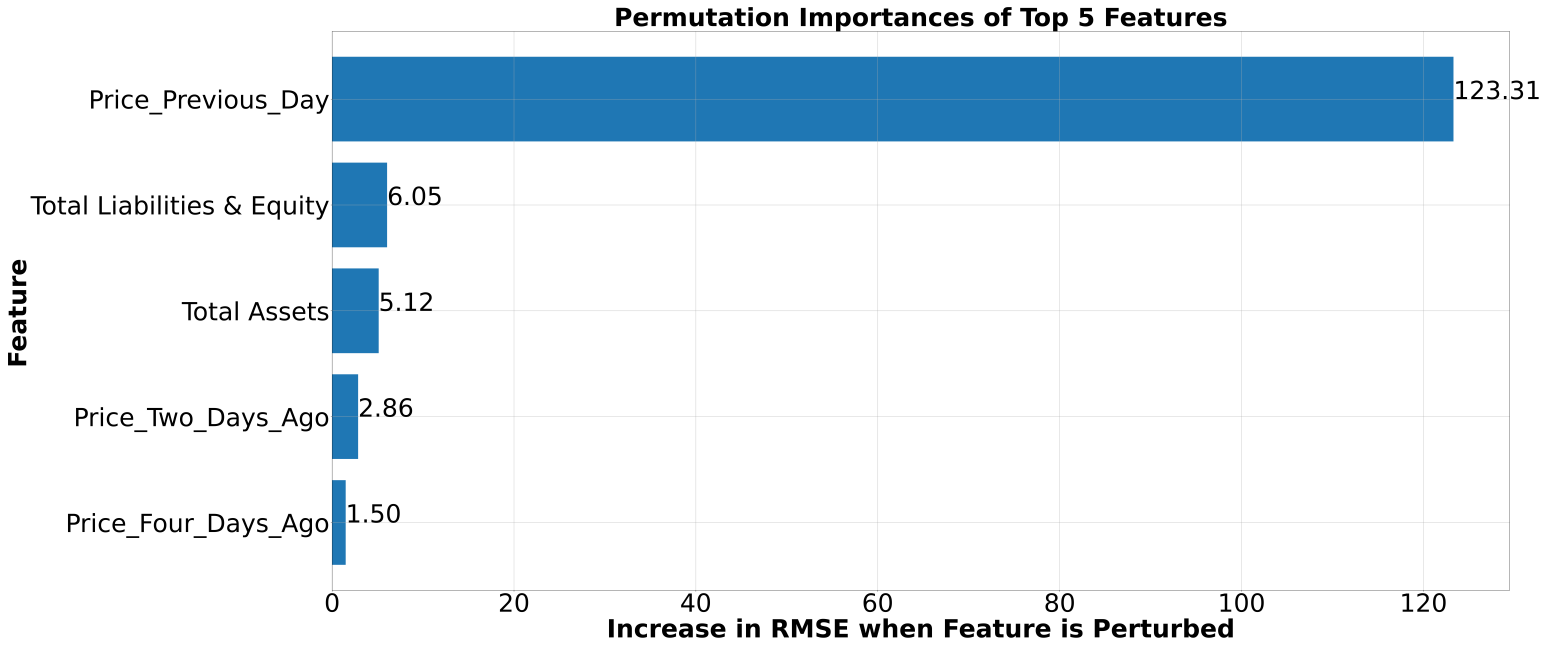
Machine Learning Model's mean RMSE and Standard Deviation



As seen above, amongst all the models I trained, the best at predicting the closing stock price was the Ridge Model with an average RMSE score of around 2.35 dollars. The Lasso Model came close second with an average RMSE of 2.38 dollars. Overall, the worst performing model that was not the baseline model was the K Neighbors Regression with an average RMSE of 8.40 dollars.

After determining that the best model was the Ridge Model, I decided to investigate what were the most important global features for this model. I then decided to find the top five global features by permutation feature importance. The results are shown below.

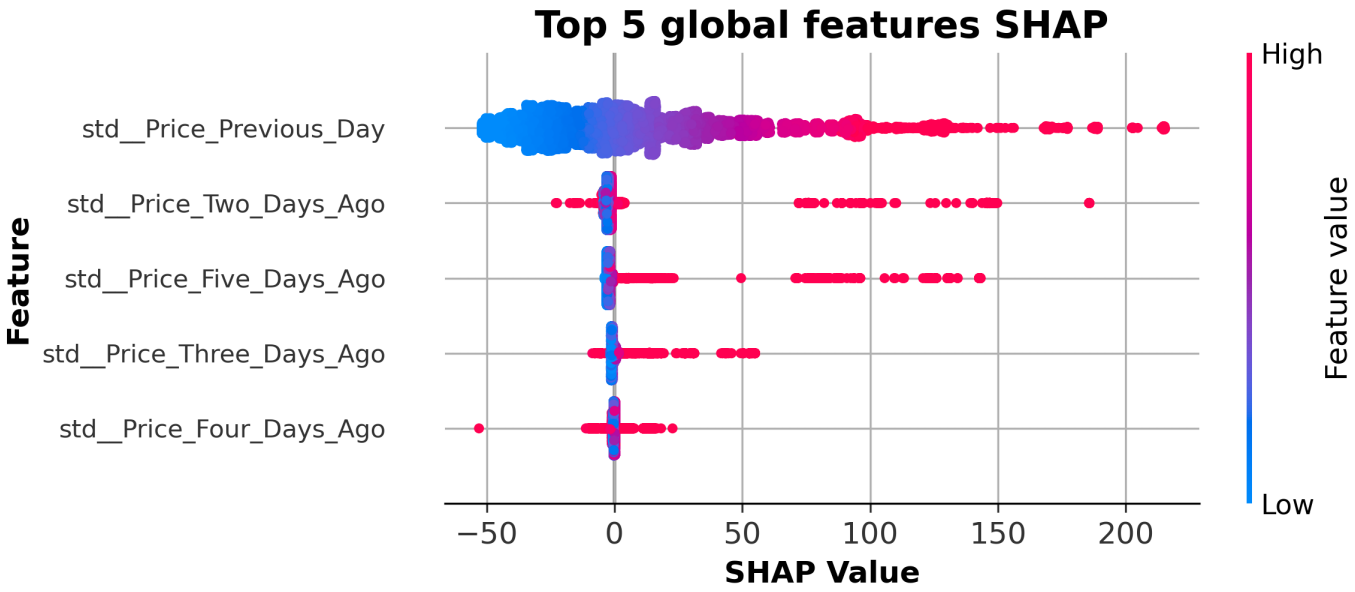
Feature Permutation Importance Ridge Model



The graph clearly shows that the most significant global feature, when permutating our features, is the stock's closing price from the previous day. This is consistent with the autocorrelation graph that highlights a close relationship between a stock's current and previous prices. Other key features identified were Total Liabilities and Equity, and Total Assets. However, this analysis, which involved permuting each feature individually, may underrepresent the importance of the price two days prior because of its strong correlation with the previous day's price. This issue affects all time-lagged features, as their influence is diminished due to the offsetting effect of other correlated prices during the permutation process.

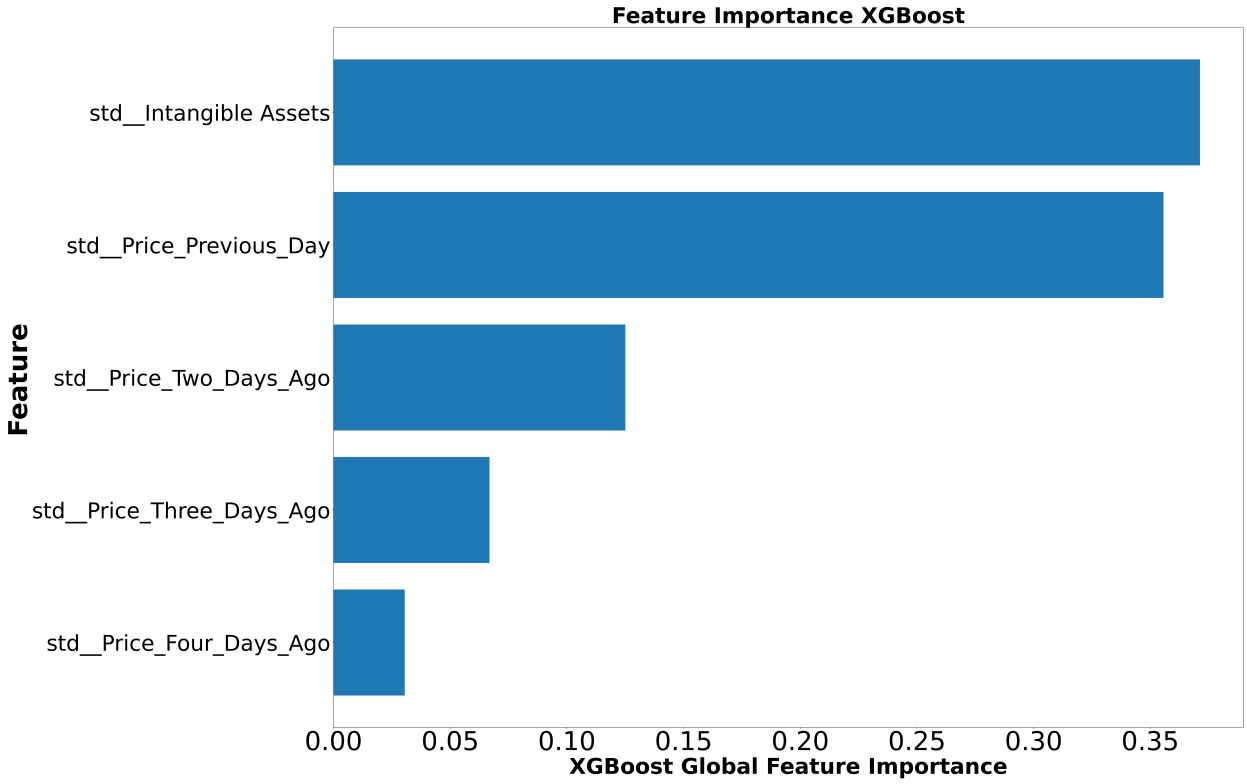
Considering the strong correlation between a stock's current and past prices, I opted to employ SHAP analysis for a more accurate assessment of each feature's global importance, as it effectively accounts for correlations.

Top 5 Global Features SHAP



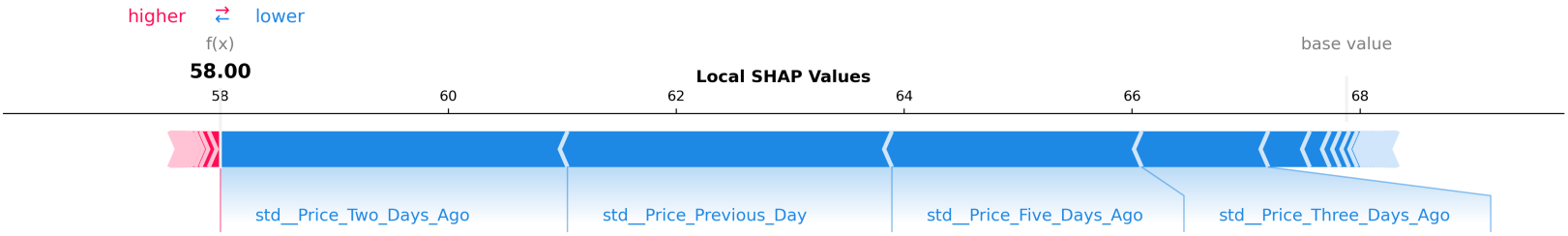
As we can see, this graph also points at the price the previous day as the most important feature in future closing stock price predictions. The figure also shows that the top five features by global importance are the lagged closing prices. As the SHAP value is not affected by correlation between features, this stresses the importance of the previous closing prices in our target variable prediction.

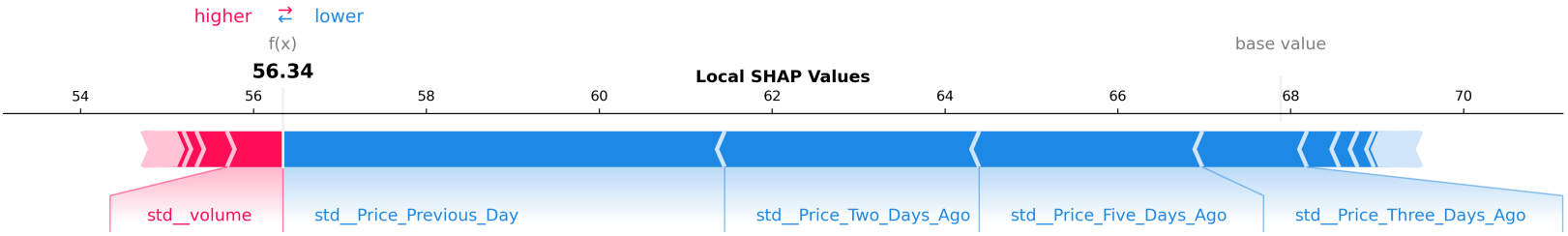
Top 5 Global Features xgBoost 'weight' Metric



As shown above, the global importance of features in the xgBoost model was also determined using the 'weight' metric, which evaluates the frequency of a feature's use in splits across trees. Interestingly, 'Intangible Assets' emerged as the most significant feature, followed by the 'price previous day'. All three graphs collectively suggest that the 'price previous day' is a key determinant of the target variable.

Finally, I also decided to see how different features impacted the prediction of a specific point in my dataset. To do so I used SHAP explainer to tell me what features affected the prediction of the stock's closing price negatively and which ones positively. The plots showing the features affecting two randomly chosen points' closing price can be seen below:





The graphs above further show the importance that the lagged prices, especially the price the previous day, have on determining the predicted value of a point. However, they also show how other features not mentioned in the above graphs, such as volume of stock traded, can also have an impact on the prediction of our target variable.

Ultimately, from these finding it is possible to determine that the closing price of the stock the previous day is the most important feature in determining our target variable, with the other lagged features for closing prices also being important in the prediction. This makes sense as the current price is very highly correlated to previous prices as a stock's closing price will usually not change much from the closing price in previous days.

Outlook

Although my model is able to predict the closing price of a stock reasonably accurately, I believe I could take more steps to ensure the model is more interpretable. For example, dropping more highly correlated features could make my model's results easier to interpret.

Moreover, this model only relies on quarterly financial data, previous prices and the sector the company operates in. However, it does not have other information that affects the S&P500 stock prices such as market sentiment, monetary policy or foreign markets. Integrating these features could likely further improve my model's predictive power.

Furthermore, I could also increase the model's predictive power by gathering information from more recent years. At the moment, the model is trained on data from 2012 to 2016 meaning that as time passed, the model is less generalisable to the current stock market.

Finally, if I had more time I would also look more into implementing the deep learning models used in other projects, such as LSTM, as they are capable of predicting the current stock price more accurately and using less features. However, using a deep learning tool could decrease the interpretability of the model.

References

Gawlik, Dominic. "New York Stock Exchange | Kaggle." *Kaggle: Your Machine Learning and Data Science Community*, https://www.kaggle.com/datasets/dgawlik/nyse. Accessed 9 Dec. 2023.

Izam, Mohammed. "All ML Algos ." *Kaggle: Your Machine Learning and Data Science Community*, Kaggle, 15 Sept. 2023, https://www.kaggle.com/code/izammohammed/all-ml-algos.

Sfaihi, Yassine. "[LSTM] S&P500 Stocks Time Series Forecasting | Kaggle." *Kaggle: Your Machine Learning and Data Science Community*, Kaggle, 17 Feb. 2023, https://www.kaggle.com/code/yassinesfaihi/lstm-s-p500-stocks-time-series-forecasting.