Predicting Changes in Stock Price with Machine Learning

Nuo Wen Lei
Brown University
GitHub [4]
nuo_wen_lei@brown.edu

1 Introduction

Predicting stock prices has been a popular topic among computer scientists and data scientists because the prospect of having a continuous passive income stream is simply too tempting to pass up. The motivation behind this project is similar, however seeing as many have tried and failed, a secondary, more realistic goal is to uncover the most predictive features of a stock's short-term success in hopes that future works can build upon these results to create more predictive models.

1.1 Dataset

The dataset [2][3] for this project is web-scraped from two different financial websites every market day over the course of a month, which results in ~1500 samples per day. The stocks that are collected each day are determined based on a screener such that all stocks collected have a minimum market cap of $1 billion dollars and a minimum average volume of 500,000. This achieves two goals. Firstly, larger companies tend to be more stable, so the risk is reduced by only investing in high market cap stocks. Secondly, a high average volume guarantees high liquidity in the stock, which means there is likely a stable availability of the stock in the trade market. Different stocks are collected each day due to these restrictions. The collected target variable is the change in stock price over 5 market days, which is one week including non-market days.

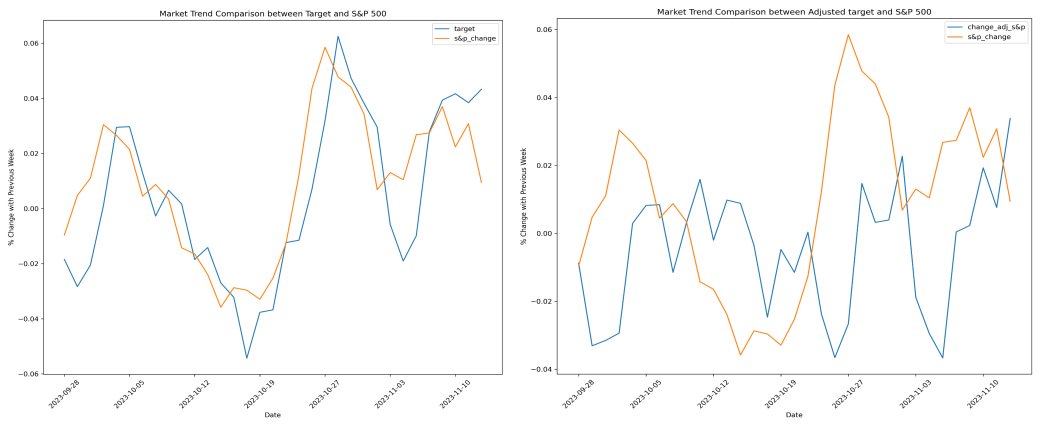

Fig 1. The **left** compares the average collected target with the change in S&P 500. The **right** compares the average adjusted target with the change in S&P 500. Due to the neutralization, the adjusted target no longer follows the S&P 500 as closely as before.

In terms of the features in this dataset, there are 335 continuous and two categorical features, totally 337. Since this dataset is collected repeatedly over a month, this is a time series dataset. And with some financial statistics being unavailable for certain stocks (e.g. debt-related statistics for debt-free companies), there are also missing values in this dataset.

2 Exploratory Data Analysis

In the Exploratory Data Analysis, we examine different parts of the dataset to find if there are any signs of predictive or concerning features.

2.1 Target Variable

The goal of this task is to outperform other low risk investment options, which is best represented by an index like S&P 500. Therefore, to formulate the problem as outperforming the S&P 500, we preprocess the 5-day % change in stocks by subtracting by the same 5-day % change in the S&P 500. The target variable compared to the % change in the S&P 500 is shown in Figure 1.

2.2 Categorical Features

The two categorical features in this dataset are company sector and area respectively. Company area represents a more specific subset of a general sector. In order to see the effect of these categories, we look at the distribution of beta across different sectors.

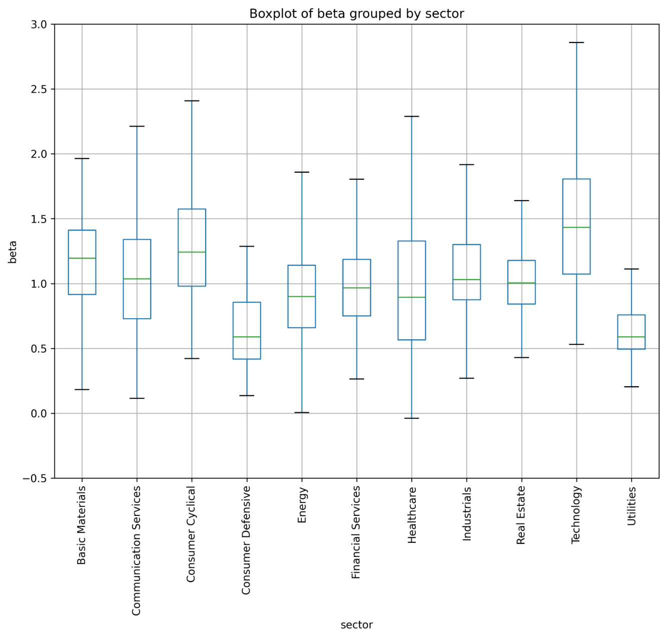

Fig 2. Boxplot of beta across all sectors. Beta measures the volatility of a stock with respect to the market.

One example is the distribution of beta, representing the volatility of a stock with respect to the market, across different sectors. As shown in Figure 2, Consumer Defensive and Utilities are sectors that have lower beta than the rest, which is expected. Consumer Defensive refers to daily essentials like food, beverages, and common household items while Utilities refers to electricity, water, and gas. Since these are essentials, this sector is not influenced by market conditions. This shows that the interaction of these categorical features with other features can create more complex attributes of stocks.

2.3 Feature Correlation and Autocorrelation

Since the dataset is collected from two similar financial websites and our prediction gap is within days, we expect some highly correlated and auto-correlated features in the dataset, so we examine a correlation matrix and autocorrelation plot of all 335 continuous features.

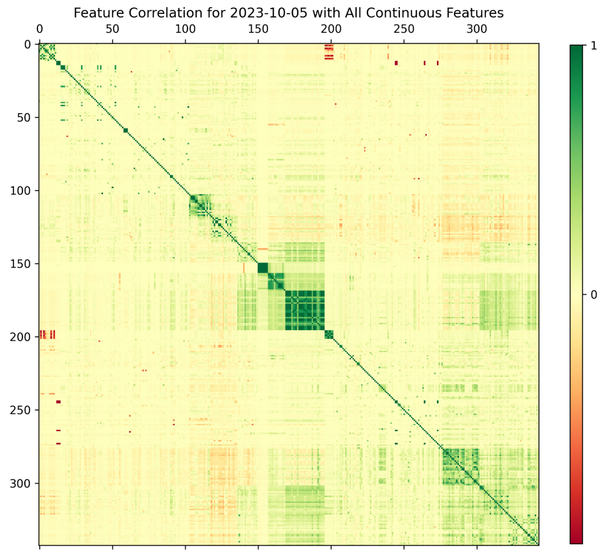

Fig 3. Correlation matrix for all 335 continuous features on a particular day. The feature names are omitted due to the number of features.

Figure 3 shows roughly 50 highly correlated features, likely due to coverage of similar common metrics by both data sources like stock price.

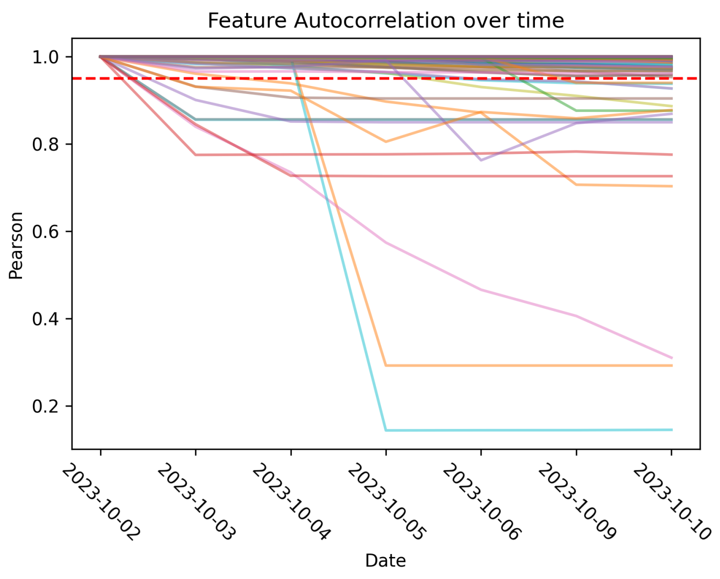

Fig 4. Autocorrelation plot of continuous features. The red dotted line shows the 95% correlation threshold. 317 features are above and 18 features are below the threshold after one week.

Since this dataset contains a lot of long-term features such as a 5-year average of dividend growth rate, it is expected that most features are highly auto-correlated after one week. The 18 features that are below the 95% correlation threshold all describe short-term metrics such as the volume traded on the last day.

2.4 Missing Values

This dataset contains a large portion of missing values. 23% of all cells are missing values and 95% of all features contain some missing values. Since missing values are distributed across features, we could not drop features with missing values.

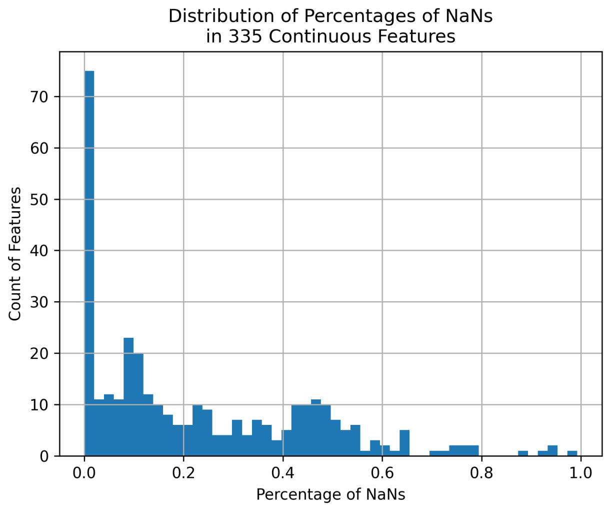

Fig 5. Histogram of missing value percentage in each feature. The highest counting bin around 0 shows that there are ~75 features that only have less than 2% of missing values.

3 Methods

In the Methods section, we specify what processes and pipelines we utilized for this project. Our selected models are XGBoost Regressor, Random Forest Regressor, K Neighbors Regressor, Linear Regression, and Elastic Net.

3.1 Data splitting

Since this is a time series dataset and there is a 5-day gap between training and prediction, we split the 27 days in the dataset into 15 days of training, 5 days of validation, and 7 days of testing to avoid data leakage between training and testing. Using a modified version of the TimeSeriesSplit method provided by Scikit-Learn [1], we create splits that each contain a moving window of 10 days of training data followed by a 5-day gap and ending with 1 evaluation day. Our data splitting totals to 5 train/validation splits and 7 train/test splits.

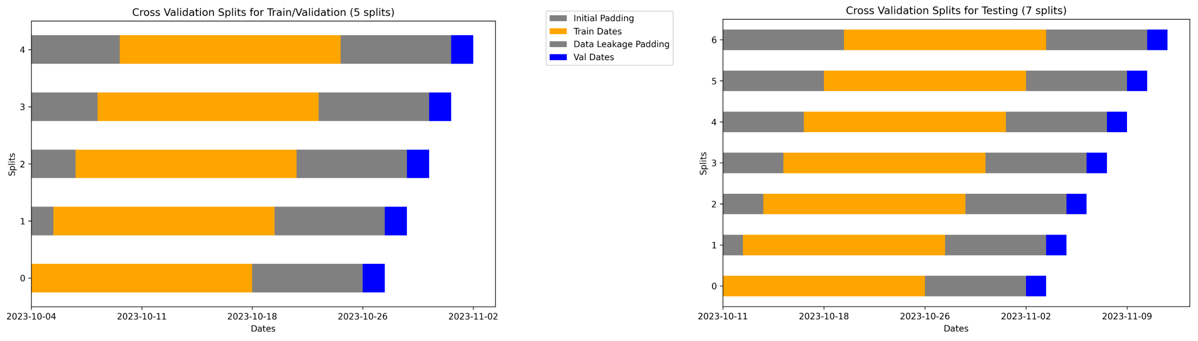

Fig 6. Cross validation splits for training, validation, and testing. Note that the dates for testing are later than those of train/validation.

3.2 Model Details

Uncertainty

Since this is a time series dataset, the splits are deterministic. Therefore, there is no uncertainty in deterministic models. However, for indeterministic models like XGBoost and Random Forest, we used 5 random states to train and test the model results.

Evaluation Metric

We use Root Mean Square Error (RMSE) as the evaluation metric to increase interpretability of model errors as the same unit.

3.3 Preprocessing

To separately preprocess continuous and categorical features, we use the ColumnTransformer provided by Scikit-Learn [1]. For categorical features, missing values are placed in a "missing" category and all categories are processed with Scikit-Learn's OneHotEncoder [1]. This adds 43 features to the dataset.

For continuous features, since there are outliers, we used Scikit-Learn's StandardScaler to standardize features around 0. To deal with missing values for all non-XGBoost models, we could not use the Reduced Features approach because there were too many unique missing value patterns. Therefore, we use KNNImputer from Scikit-Learn [1] to impute missing values in the dataset. We opted not to use IterativeImputer due to limitations on computational resources.

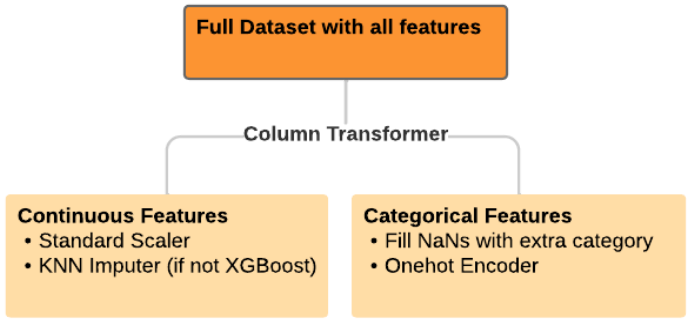
Full Dataset with all features
Column Transformer
Continuous Features
• Standard Scaler
• KNN Imputer (if not XGBoost)
Categorical Features
• Fill NaNs with extra category
• Onehot Encoder

Fig 7. Diagram for data preprocessing. The total amount of features after preprocessing changed from 337 to 380.

3.4 Model Hyperparameters

To find the sets of model hyperparameters that perform best on the validation splits, we search through every combination of the following hyperparameters for every model.

Models	Deterministic?	Hyperparameters
XGBoost	No	n_estimators: 100, 500, 1000 learning_rate: 1e-3, 1e-5, 1e-7 reg_lambda: 0.1, 1.0, 10.0 subsample: 0.5, 0.7, None max_depth: 3, 5, 7
Random Forest	No	n_estimators: 100, 250, 500 max_features: sqrt, log2 criterion: poisson, friedman_mse max_samples: 0.5, 0.7
K Nearest Neighbors	Yes	n_neighbors: 50, 100, 150, 200, 300, 500, 700, 1000 weights: uniform, distance

Linear Regression	Yes	None
Elastic Net	Yes	alpha: 1e-3, 1e-2, 1e-1, 1.0, 10.0 l1_ratio: 1e-2, 1e-1, 7e-1, 9e-1

Table 1. Hyperparameters of models.

3.5 Machine Learning Pipeline

Due to our unique windowing method of TimeSeriesSplit [1], we chose to manually do cross validation. For every train/validation split, we use a PredefinedSplit to do hyperparameter grid search with GridSearchCV from Scikit-Learn [1], then we record the best hyperparameters and validation scores for every split.

4 Results

In the Results section, we discuss the performance and feature importance of our best models after calculating metrics over test splits and random states for our tree-based models.

4.1 Model Performance vs Baseline

The baseline score for our models is calculated based on the training set mean for every corresponding test set. The total baseline on this dataset is ~0.057. As shown in Figure 8, the XGBoost Regressor performs the best out of all models.

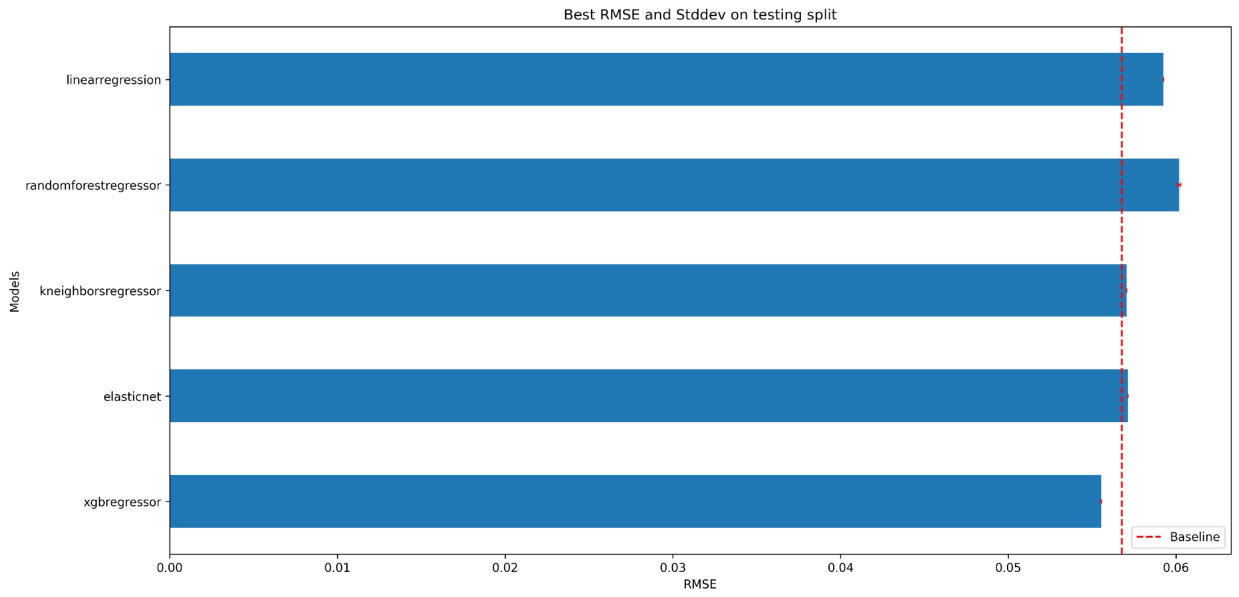

Fig 8. Model Performance with respect to Baseline (red dotted line). The red cap of each bar represents uncertainty, however uncertainty for XGBoost and Random Forest is minimal and hardly visible.

4.2 Global Feature Importance

To understand what features are predictive of % changes in stock price, we examine the global feature importance of our best model, XGBoost Regressor, with average gain, permutation importance, and SHAP summary plot.

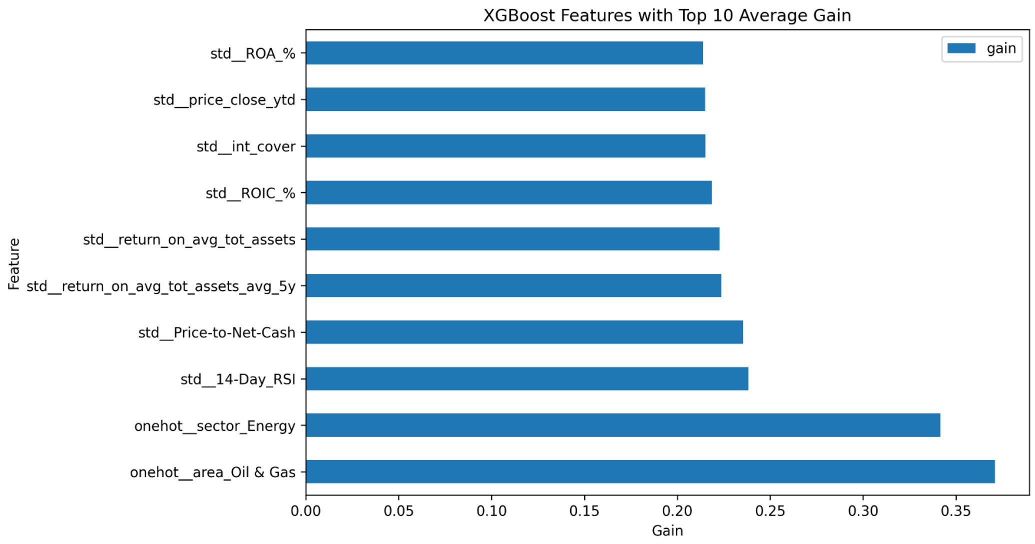

Fig 9. Top 10 feature importances of XGBoost based on average gain.

Figure 9 shows that the most important features based on average gain are '14-Day_RSI', 'sector_Energy', and 'area_Oil & Gas' for XGBoost.

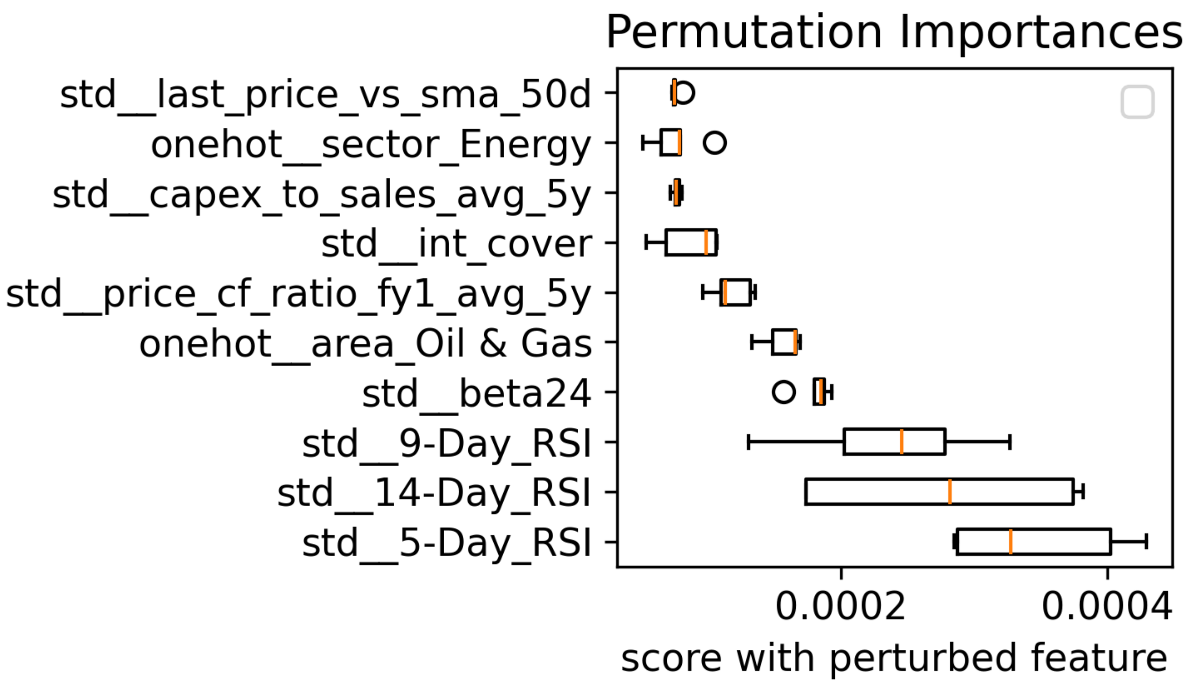

Fig 10. Top 10 feature importances of XGBoost based on permutation importance.

Figure 10 shows that the most important features based on permutation importance are '5-Day_RSI', '9-Day_RSI', and '14-Day_RSI'. Note that 'sector_Energy', and 'area_Oil & Gas' are also present.

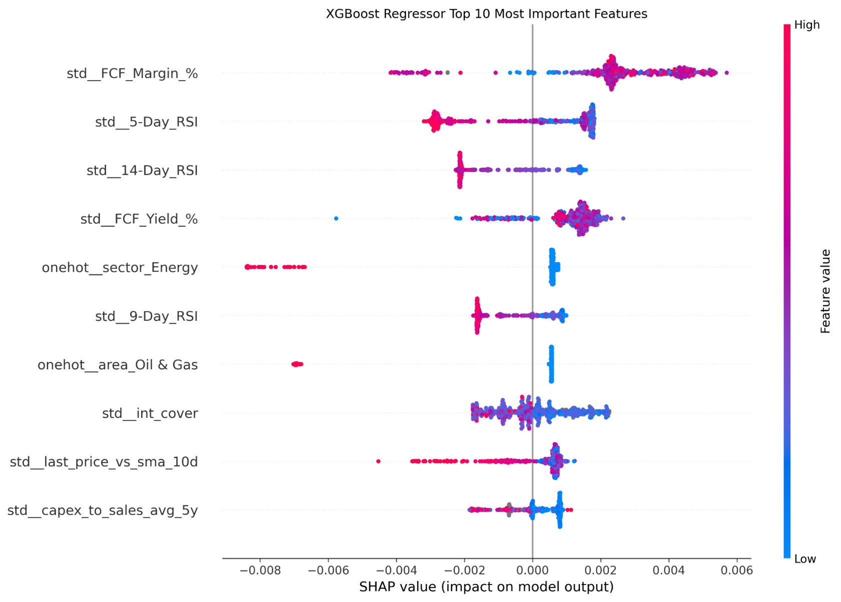

Fig 11. Top 10 feature importances of XGBoost based on SHAP summary plot.

Figure 11 shows that the most important features based on SHAP values are 'FCF_Margin_%', '5-Day_RSI', and '14-Day_RSI'. Note again that 'sector_Energy', 'area_Oil & Gas', and '9-Day_RSI' are present.

The recurring RSI-related features are expected because Relative Strength Index (RSI) measures whether a stock is overbought (high RSI) or underbought (low RSI) in the short term of 5, 9, or 14 days. Therefore, it makes sense that a short term feature is used to predict short term targets.

However, the appearance of the Energy sector and Oil & Gas area is unexpected because these features are not short term. This likely means that the entire Energy sector and especially the Oil & Gas area have been decreasing in stock value recently.

4.3 Local Feature Importance

In order to see specific examples of how features contribute to predictions, we examine SHAP force plots.

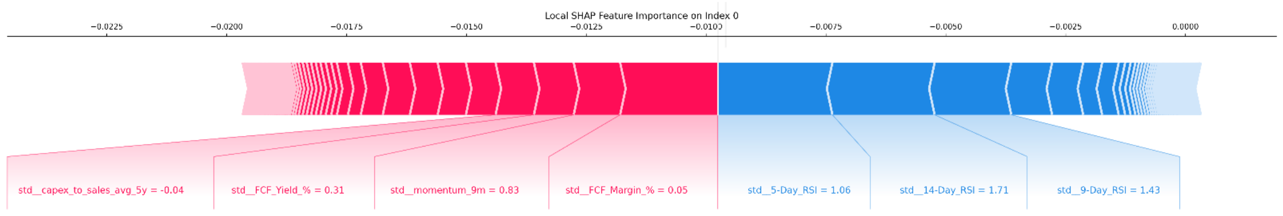

Fig 12. Force plot for index 0 of the test set. Red means positive contribution, and Blue means negative contribution.

Figure 12 shows that the largest positive contributor is 'FCF_Margin_%' and the largest negative contributors are the RSI features. This corroborates what we have found through global feature importance.

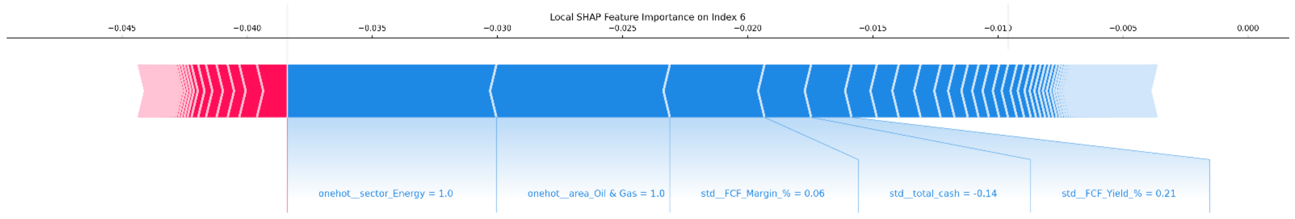

Fig 13. Force plot for index 6 of the test set. Red means positive contribution, and Blue means negative contribution.

Figure 13 is an overwhelmingly negative prediction. The largest negative contributors are 'sector_Energy', 'area_Oil & Gas', and 'FCF_Margin_%'. This example shows how negatively a prediction is impacted by being in the Energy sector and Oil & Gas area.

5 Outlook

There are many further areas of improvement to increase predictive power.

- **Expand the dataset** - stock prices are influenced by more than company financials through other real-world factors that could diversify the dataset.
- **Autoregressive features** - we did not use autoregressive features due to the daily inconsistency of stocks outlined in **1.1 Dataset**, however autoregressive features are worth exploring with models that can handle missing values.
- **Hyperparameter tuning** - the hyperparameters especially for tree-based models were by no means comprehensive.
- **Training window and prediction gap** - the 10-day window of training and 5-day prediction gap discussed in **3.1 Data splitting** was due to limited amounts of data. These numbers remain another overarching hyperparameter to tune.

6 References

[1] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[2] SeekingAlpha for daily stock financial data
[3] GuruFocus for daily stock financial data
[4] GitHub Project Repository - https://github.com/QuantBears/data-analysis