Predicting S&P 500 Return with Firm Characteristics and Macroeconomic Data

Yifan Lu
Data Science Institute, Brown University
Github

1. Introduction

In this project, we address the challenge of measuring empirical asset risk premia. Our aim is to understand and explain equity return variations using fundamental, macroeconomic, and technical factors from empirical data. However, capturing cross-sectional expected stock returns is complex due to market efficiency and unpredictable events. While traditional factor models use linear regressions on asset returns with observed factors for portfolio construction, they often overfit with many predictors and break under statistical assumptions. Recent advancements in machine learning offer promising solutions. Machine learning techniques, widely used in the finance sector's high noise-to-signal ratio environment, provide more flexibility for empirical asset pricing. These include dimension reduction for large feature sets, parameter penalization to prevent overfitting, and handling non-linear interactions between features and the target variable.

The project is inspired by many works on asset pricing, including Gu, Kelly and Xiu (2020). In their paper, they describe an asset's excess return as an additive prediction error model:

$$r_{i,t+1} = \mathrm{E}_t(r_{i,t+1}) + \epsilon_{i,t+1}$$

where

$$\mathrm{E}_t(r_{i,t+1}) = g^\star(z_{i,t})$$

The framework imposes some important restrictions on the g(*) function, which specifies that it depends neither on i nor t. In other words, they assume that the model maintain the same form over time and across different stocks. So our prediction does not use information from the history prior to t, or from individual stocks other than the ith.

1.1 S&P 500 Return Data

Given the large number of stocks listed on the US stock market, we narrowed the domain of our problem to the value-weighted return of 500 largest US companies, which is measured by S&P 500 return, a benchmark stock index commonly used by investors and portfolio managers. This choice also diminish the effect of group structure of stocks. We obtained the monthly stock return data from Center for Research in Securities Prices under Wharton Research Data Services. For sampling period, we chose the last 20 years or so, from Jan. 2003 to Dec. 2020.

1.2 Firm Characteristics

The fundamental factors are constructed from firm characteristics data, obtained from Prof. Dacheng Xiu's website. The dataset includes the following information:

DATE	The end day of each month (YYYYMMDD)
permno	CRSP Permanent Company Number
94 Lagged Firm Characteristics	
sic2	The first two digits of the Standard Industrial Classification code on DATE

Table 1. Firm Characteristics Data Information

To avoid the forward-looking bias, we assumed that monthly characteristics are delayed by at most 1 month. Therefore, in order to predict returns at month t + 1, we used most recent monthly characteristics at the end of month t. In this dataset, we've already adjusted the lags.

1.3 Macroeconomics Variables

The eight macroeconomic predictors follow the definitions by Welch and Goyal (2008, RFS). The data are available on Prof Goyal's website, including dividend-price ratio (dp), earnings-price ratio (ep), book-to-market ratio (bm), net equity expansion (ntis), Treasury-bill rate (tbl), term spread (tms), default spread (dfy), and stock variance (svar).

2. Exploratory Data Analysis

Having reduced and merged the above datasets, we conducted EDA to better understand the time series pattern of return, distribution and correlation of feature sets, and relationship between feature and target. We started with the time-series plot of S&P 500 return.

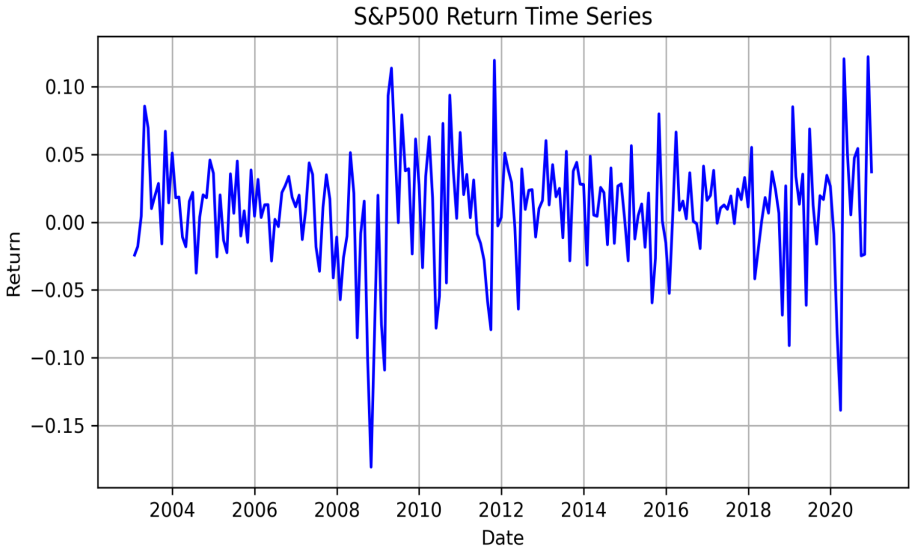

Fig 1. A time-series plot of S&P 500 return, calculated by the market-value weighted return of individual stocks.

In order to determine the stationarity of the return series, we also drew the autocorrelation plot of return. The result shows that the return could be seen as a random walk process, thus no need to add lagged return as an additional feature.

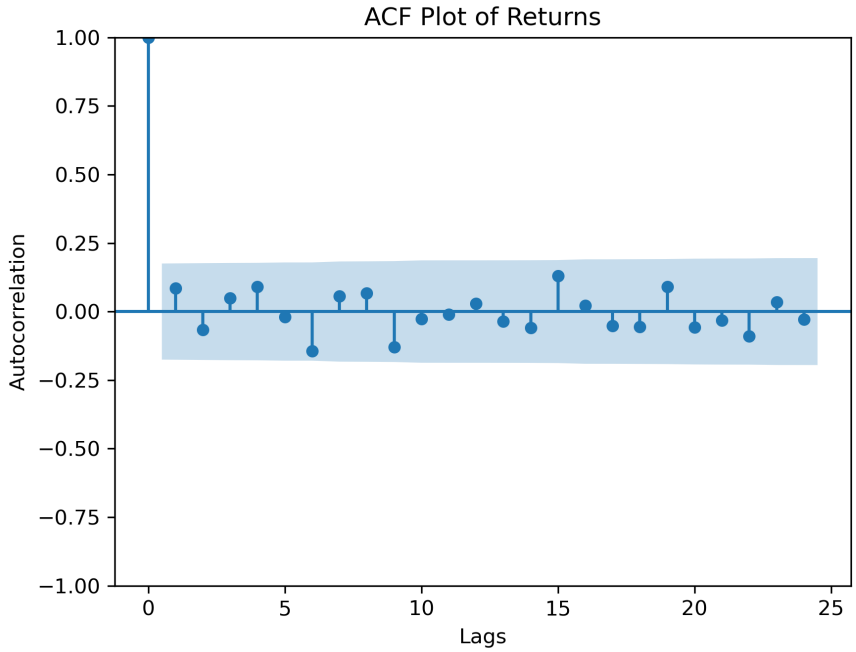

Fig 2. ACF plot of return, where the vertical lines represent the autocorrelation of different lags on the horizontal axis, and the shaded area is the confidence interval of stationarity test.

We also visualized the pairwise correlation between different features and selected most correlated ones.

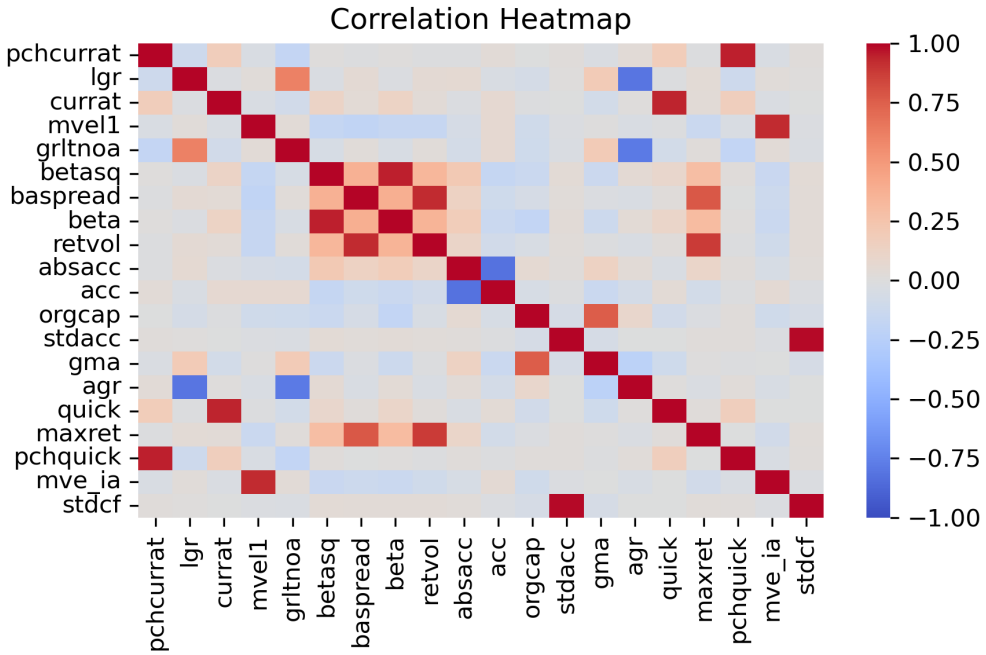

Fig 3. Correlation heatmap of feature sets, normalized to scale of [-1,1].
Only select top 20 features to visualize pairwise correlation.

To reduce model complexity, we dropped one of each highly correlated feature pairs whose absolute value of correlation exceeds the threshold of 0.7.

We have also drawn the scatter plots of individual feature and return to analyze the relationship between them. The distribution and patterns across the features show a degree of similarity, so we have chosen to present only one of them to illustrate our findings.

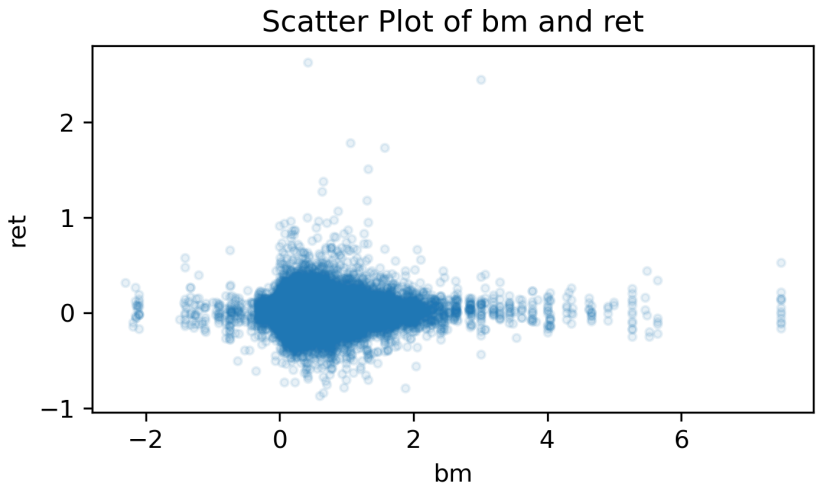

Fig 4. Scatter plot of individual stock's bm(book-to-market ratio) and return.

We observed that the relationships between features and return are non-linear and quite ambiguous. Besides, the distribution of features are mostly right-skewed, which prompts us to rank features cross-sectionally among stocks before standard scaling them (see the data preprocessing section below).

3. Methodology

The next step of model development involves train-validation-test splits, feature preprocessing, and hyperparameter tuning.

3.1 Splitting strategy

Due to the time-series nature of our dataset, we cannot use the traditional cross validation. Instead, we used the 5-fold recursive splitting scheme to preserve the temporal nature of our data but also to include uncertainties. The procedure is as follows: it starts with a subset of the whole sample period from the beginning, and recursively increase the training sample to retain the entire history, while maintaining fixed-sized rolling validation and test sets.

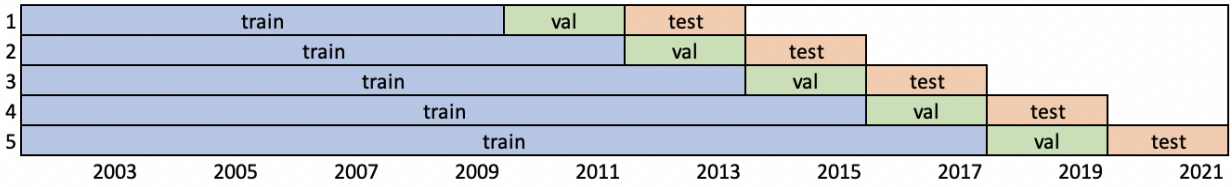

Fig 5. Five-fold recursive evaluation scheme of splitting data.

Moreover, we may not be able to use the TimeSeriesSplit function directly from sklearn since our data has multiple data points at one time. Thus, we wrote a *recursive_split* function to customize the splitting process.

3.2 Data preprocessing

There are mainly three steps involved in the data preparation process. The first step is dealing with missing data. We first drop the features with more than 40% missing values (['orgcap', 'rd_sale', 'rd_mve', 'realestate']). Then we chose to impute the missing data with its cross-sectional median for the whole dataset, as in the paper of Gu, Kelly and Xiu (2020). There is no data leakage because the imputed feature values only depend on other data during that specific month. The second step is to preprocess continuous variables, which includes 94 firm characteristics and 8 macroeconomic variables. From the previous EDA step, we noticed that most features have outliers, so we first map them into their cross-sectional rank among all stocks, then normalize them into the [0,1] interval. The third step is to use one-hot encoder on the industry variable ‘sic2’, transforming it into dummy variables as additional features.

3.3 ML pipeline

The ML pipeline we developed consists of multiple steps. We first split the data into train, validation and test set. For each fold of evaluation, we iterated over pairs of parameter combination to search for the best validation score and best model. However, we may not use a GridSearchCV due to the customized splitting method, so we wrote a manual grid search function. Since this is a regression problem, we chose the RMSE score for its scale sensitivity, making it intuitively interpretable as how much error the model typically makes in the units of the target variable. It's also popular among the works of others, making it easily comparable.

We trained and evaluated the performance of four models– Principal Components Regression (PCR), Linear Regression (L1), Random Forests and XGBoost – based on root mean squared error (RMSE). The PCR model can perform dimension reduction in out high-dimensional predictor setting, while the Lasso model can prevent overfitting by controlling the feature weights. To further measure the uncertainties of RMSE due to non-deterministic ML methods (random forest and XGBoost), we also tried different random_state when initializing the ML algorithms..The following table 2 summarizes the hyperparameter values we tuned, which is largely based on the works of Gu, Kelly and Xiu (2020).

PCR (PCA+Linear regression)	PCA(): 'n_components': [1,3,5,7] 'svd_solver': 'randomized'
LR with L1 Regularization (Lasso)	'alpha': np.logspace(-4,-1,10)
Random Forests Regression	'n_estimators': [200], 'max_depth': [1, 2, 3, 6], 'max_features': [3, 5, 10, 20] 'random_state': [1, 42, 58, 69, 72]
XGBoost	'n_estimators': [10000], 'early_stopping_rounds': [50], 'max_depth': [1, 2, 3], 'learning_rate': [.01, 0.1], 'random_state': [1, 42, 58, 69, 72] 'subsample':[0.66]

Table 2. Hyperparameters of four ML models

4. Results

For the results, all four models beat the baseline model of naively predicting zero returns, which is also the mean of a random walk process that our return series follows. Among the four ML models, XGBoost outperformed others in terms of mean of RMSE values, though with higher variability, as evidenced by the result table and plot below.

Model	Mean of RMSE	Standard deviation of RMSE	(baseline - Mean RMSE) / standard deviation
Baseline	0.0466	0.0114	-
PCR	0.0457	3.081e-5	29.211
Lasso	0.0452	2.096e-5	66.694
Random Forest	0.0450	3.293e-5	48.588
XGboost	0.0403	0.00052	12.115

Table 3. Summary table of model performance

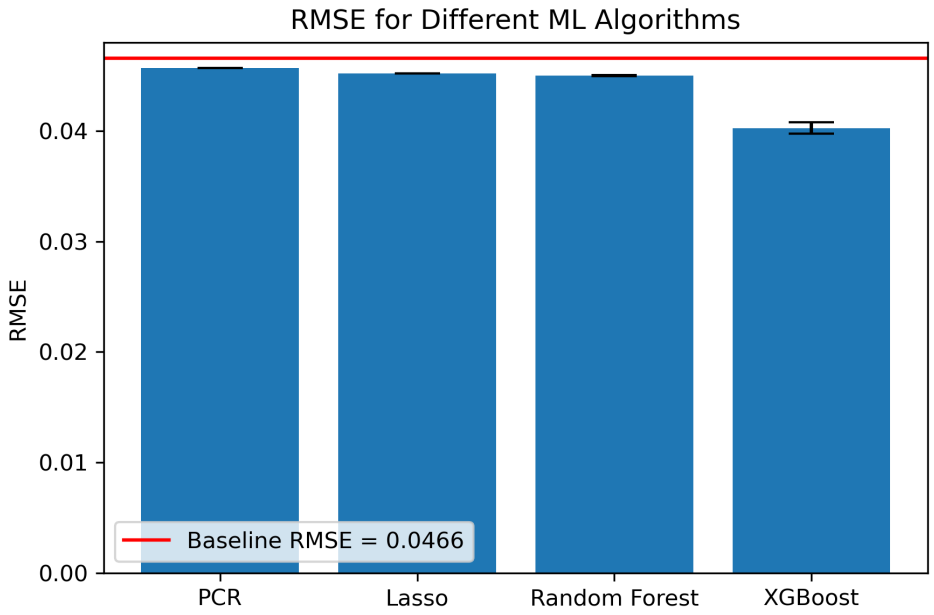

Fig 6. RMSE comparison plot of four ML algorithms. The vertical bars are the mean of RMSE values, and the error bars represent the standard deviation of RMSE values.

The plot below further displays the true and predicted S&P 500 return by the XGBoost model. We observed that the model captures the volatility clustering effect, though it may not predict some market shocks and seasonality effect since it cannot interpolate predictions outside the range of the training data.

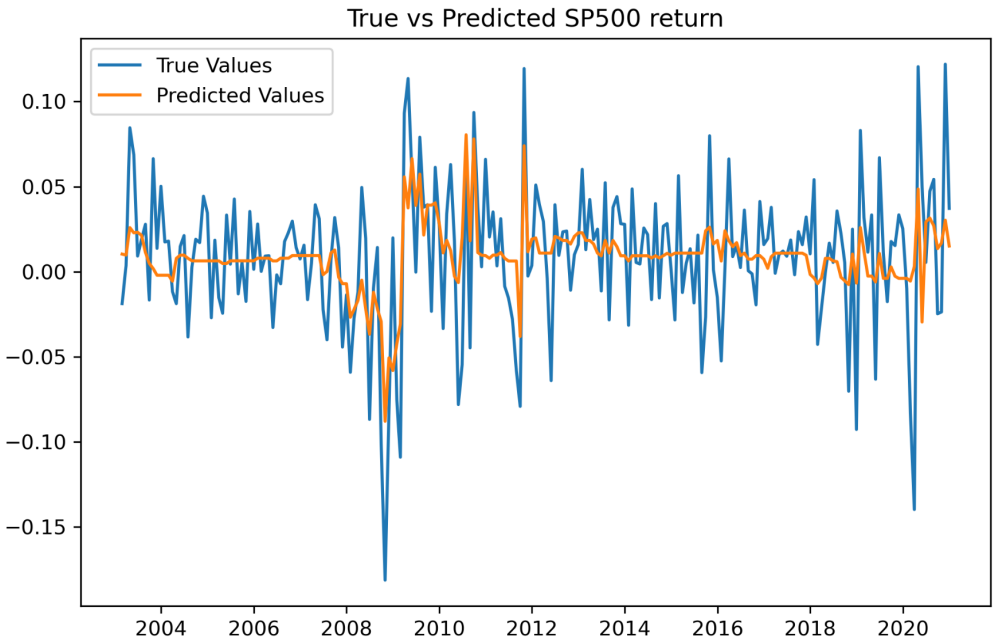

Fig 7. True vs predicted SP500 return plot of XGBoost

4.1 Global feature importance

To better interpret our model, we also calculated three global feature importances metrics, namely the permutation feature importance, total gain of XGBoost, and SHAP value.

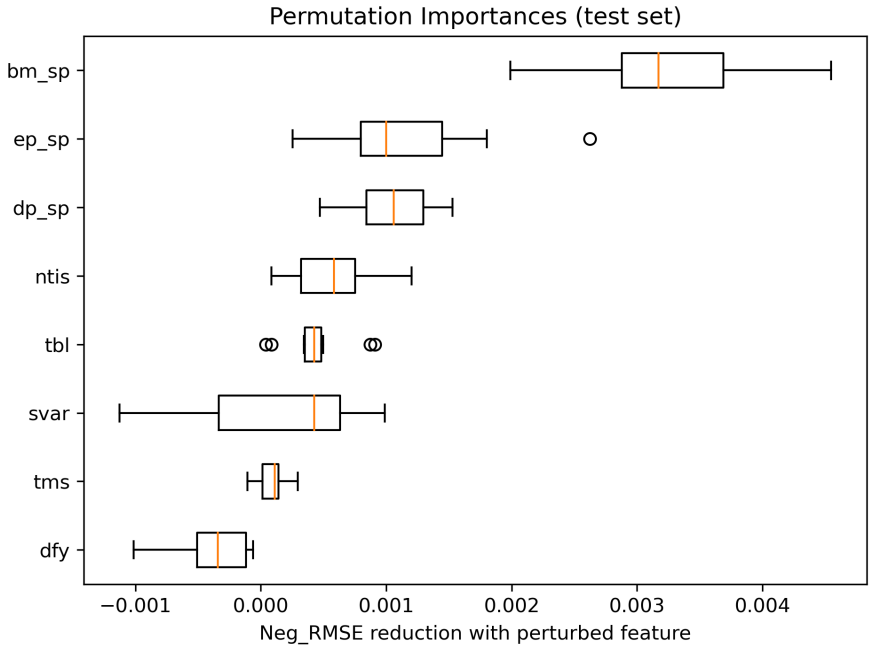

Fig 8. Most important features as per permutation feature importance

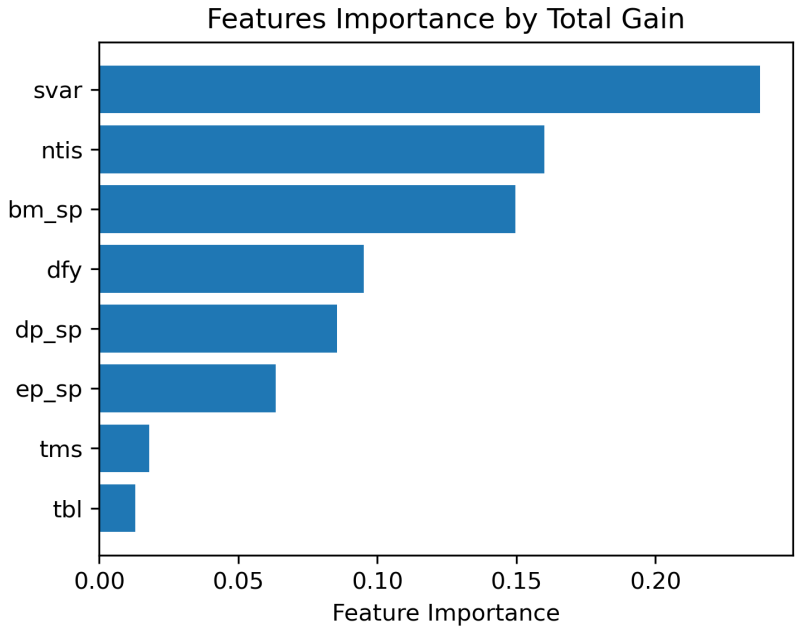

Fig 9. Most important features as per XGBoost relative importance measure - total gain

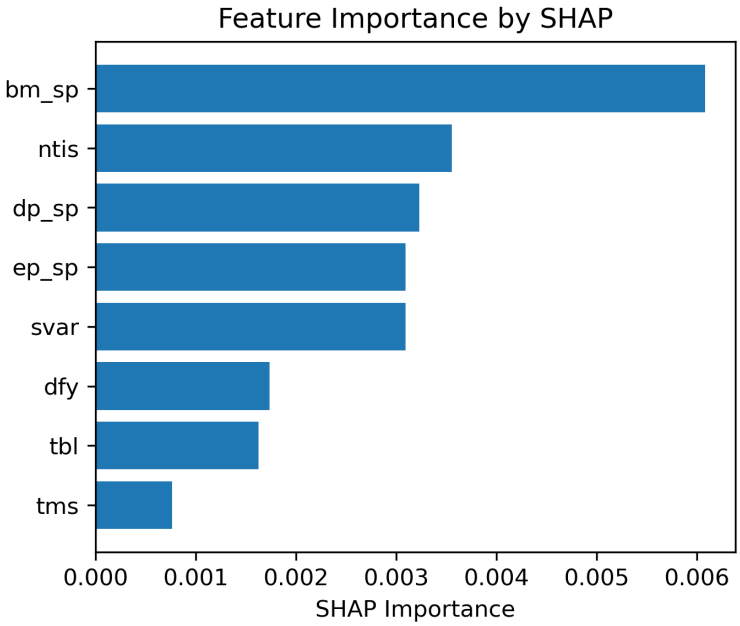

Fig 10. Most important features as per mean Shap value

Key findings:

- The most predictive features from three metrics are all macroeconomic variables, while the feature importances of firm characteristics are nearly zero. This is quite unexpected as a deviation from the result of Gu, Kelly and Xiu (2020), which we suppose may be due to our limited stock pool and feature set compared to theirs.

- The book-to-market ratio and dividend-to-price ratio are two most important features, both of which are frequently cited factors in the asset pricing literature.

4.2 Local feature importance

In addition, we calculated the SHAP local feature importance of two data points and drew their force plots as below.

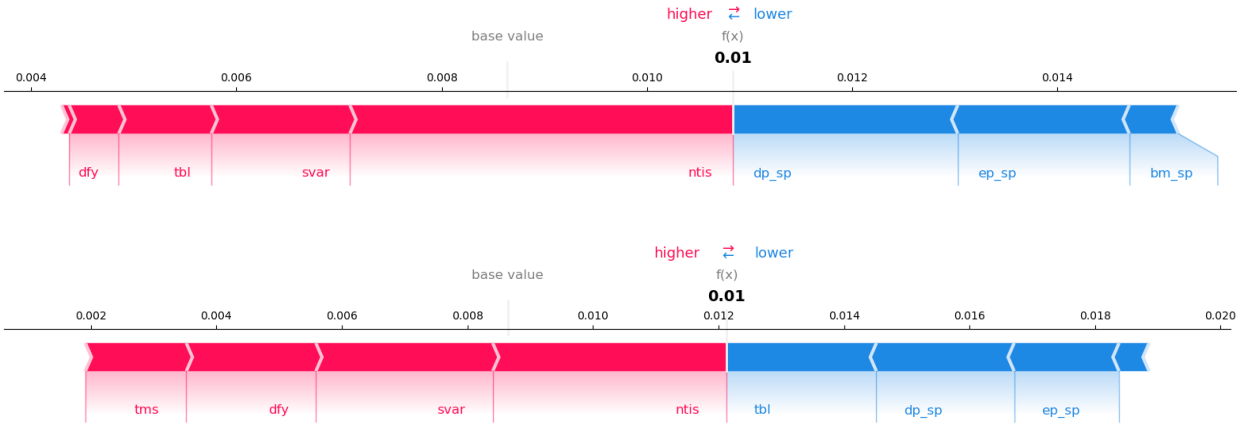

Fig 11. SHAP force plots of the 10th and 30th data points.

We observed that the dp_sp (dividend-to-price ratio) drives the return lower, which is to our surprise because higher dividends are typically associated with higher return. We supposed it may be due to some feature interaction for these two data points. On the other hand, ntis (net equity expansion) drives the return higher, which is consistent with financial theory because it may suggest capital raising activities.

5. Outlook

Given more time, our model performance could be potentially enhanced by the following methods. First, we could try more on feature engineering, such as using the product of firm characteristics and macroeconomics data as additional predictor to account for feature interaction. Moreover, we could train linear regression models with L2 regularization or elastic net to compare with XGBoost. Lastly, we could experiment with larger dataset that contains more stocks or longer sample period, to test the robustness of our model performance, though with higher computational cost as well.

6. References

[1] Gu, Shihao and Kelly, Bryan T. and Xiu, Dacheng, Empirical Asset Pricing via Machine Learning (2020). Chicago Booth Research Paper No. 18-04; 31st Australasian Finance and Banking Conference 2018; Yale ICF Working Paper No. 2018-09. Available at SSRN

[2] Welch, Ivo, and Amit Goyal, A Comprehensive Look at The Empirical Performance of Equity Premium Prediction (2008). Review of Financial Studies 21, 1455–1508.
[3] Github Repository: https://github.com/hrbzkm98/ml-research/tree/master

Data:

https://dachxiu.chicagobooth.edu/
https://sites.google.com/view/agoyal145
https://wrds-www.wharton.upenn.edu/pages/get-data/center-research-security-prices-crsp/annual-update/stock-security-files/monthly-stock-file/