Machine Learning Project (Spring 2025)

Changrong Xue - 402929
Kaoru Kudou - 394696
Kanta Ono - 402355

June 15, 2025

1 Introduction and Problem Statement

The goal of this project is to support a hedge fund in building a machine learning model capable of predicting cross-sectional monthly stock returns. The model leverages two classes of datasets: return datasets such as CRSP and predictor datasets including JKP factors and Compustat firm characteristics. The task is framed as a regression problem, aiming to predict future returns from current factor values.

2 Data Preprocessing and Feature Engineering

2.1 JKP Factors

The JKP dataset contains over 150 financial factors, each representing a portfolio formed by sorting firms on specific characteristics (e.g., value, momentum, investment). Given the presence of redundant or low-quality signals, we apply several preprocessing steps:

- **Missing Value Filtering:** We remove factors with more than 30% missing data to ensure signal quality and model reliability.
- **Winsorization:** To mitigate the effect of extreme values often present in financial time series, we winsorize each factor at the 1% and 99% quantiles.
- **Variance Filtering:** Factors with negligible variation are excluded since they offer minimal discriminatory power for return prediction.
- **Correlation Pruning:** Highly correlated features (above 0.9) are pruned to prevent multicollinearity and overfitting.

After completing the preprocessing steps, we selected 18 representative factors from the JKP dataset. The selected factors are:

`beta_60m`, `bidaskhl_21d`, `dolvol_var_126d`, `ebitda_mev`, `f_score`, `inv_gr1`, `kz_index`, `ni_me`, `o_score`, `prc_highprc_252d`, `qmj`, `qmj_growth`, `resff3_6_1`, `ret_12_1`, `ret_3_1`, `ret_6_1`, `sale_me`, and `z_score`.

These factors were chosen based on their theoretical relevance in empirical asset pricing and their compatibility with the Compustat data.

2.2 Compustat Characteristics

We incorporated firm-level accounting information from Compustat to enhance the feature set. The preprocessing pipeline for the Compustat dataset was designed with both predictive accuracy and economic interpretability in mind.

First, we filtered the data to include only firms listed in the S&P 500 as of December 31, 2005. To ensure sufficient recent data coverage, we further restricted the sample to 321 firms for which Compustat data was available at least through October 31, 2024. This filtering ensures consistency and completeness in both historical and forward-looking evaluations.

Subsequent preprocessing was performed as follows:

- **Numerical Feature Selection:** Only columns with numeric data types were retained, excluding categorical or textual variables.
- **Missing Value Threshold:** Features with more than 50% missing values were removed to maintain data integrity.
- **Constant Feature Removal:** Variables with near-zero variance across the sample were excluded, as they provide no informational value.
- **Winsorization:** A 1% winsorization was applied to reduce the impact of extreme outliers while preserving underlying distributions.
- **Standardization:** All numerical features were normalized to have zero mean and unit variance.
- **Multicollinearity Filtering:** One variable from each pair of features with a Pearson correlation coefficient above 0.7 was removed to avoid redundancy.

Finally, to align the quarterly nature of Compustat data with the monthly return prediction task, we expanded each quarterly record over three consecutive months. That is, a data point dated at month t was replicated and assigned to months $t+1$, $t+2$, and $t+3$.

2.3 Merging JKP and Compustat Data

After preprocessing both the JKP and Compustat datasets, we merged them to create a unified feature set for monthly return prediction. The merge was conducted on the calendar month (`date`), aligning macroeconomic factors from JKP with firm-specific fundamentals from Compustat.

To mitigate multicollinearity resulting from merged features, we computed pairwise Pearson correlation coefficients. One variable from each pair with correlation above 0.95 was removed.

Next, we prepared for merging with CRSP by extracting and truncating CUSIP codes from Compustat to their first 8 digits, which are standard identifiers used in CRSP. This allowed for accurate firm-level matching.

The procedure yielded 321 unique 8-digit CUSIP codes, fully consistent with the firms selected from Compustat, thereby ensuring the validity of our data linkage.

2.4 Linking with CRSP and Final Dataset Construction

The final step in data preparation involved merging the cleaned JKP–Compustat dataset with CRSP monthly return data, which served as the prediction target.

To ensure reliable linkage, the merge was performed using two keys: calendar month (`date`) and the 8-digit CUSIP identifier. Only rows with exact matches between the `cusip8` field from Compustat and the `CUSIP` field in CRSP were retained.

This merge yielded a finalized dataset containing 313 unique firms, slightly fewer than the 321 Compustat entries, due to missing or unmatched return data in CRSP.

At this point, all features and targets were temporally and cross-sectionally aligned, making the dataset ready for modeling.

3 Predictive Models

3.1 Final Dataset Splitting and Setup

Non-predictive identifiers and metadata columns—such as firm codes (`gvkey`, `cik`), CUSIP fields, and date markers—were excluded to retain only relevant predictive inputs.

The dataset was then split into input features (X) and the target variable (y), where y corresponds to the monthly stock return (`MthRet`) provided by CRSP.

To simulate realistic prediction conditions, the data was partitioned based on time into three distinct periods:

- **Training Period:** February 2005 to December 2016
- **Validation Period:** January 2017 to December 2020
- **Test Period:** January 2021 to December 2024

3.2 Model Selection and Justification

For this study, we primarily focused on two advanced machine learning models: the Multi-Layer Perceptron (MLP) and Extreme Gradient Boosting (XGBoost). These models were selected for their ability to capture high-dimensional, non-linear relationships—common characteristics in financial datasets.

Multi-Layer Perceptron (MLP). An MLP is a type of feedforward artificial neural network composed of multiple layers of interconnected neurons. Each neuron applies a non-linear activation function to a weighted sum of its inputs, enabling the model to approximate complex functions. MLPs are well-suited for problems where the true mapping from features to outputs involves non-linear interactions. In our context, the feature set includes a variety of firm fundamentals and market-based factors, whose influence on returns may not be purely additive or linear.

Extreme Gradient Boosting (XGBoost). XGBoost is an efficient and scalable implementation of gradient-boosted decision trees. It builds an ensemble of trees in a stage-wise fashion and optimizes a regularized objective function. Its ability to model non-linear effects, handle multicollinearity, and prevent overfitting through built-in regularization makes it highly effective for structured tabular data like ours.

Model Rationale. MLP and XGBoost were chosen because they align well with the characteristics of our data: a mix of continuous financial variables, potential interactions among features, and a noisy target variable (monthly stock returns). MLP can capture latent patterns through deep representation learning, while XGBoost is robust to outliers and feature redundancy, making it a reliable alternative.

Benchmark Models. In addition to the two primary models, we also implemented Lasso regression and Decision Tree regression as benchmark models.

- **Lasso Regression:** As a linear model with L1 regularization, Lasso performs automatic variable selection and provides a strong baseline when linear assumptions are sufficient. It serves as a useful point of comparison to evaluate the necessity of complex non-linear modeling.
- **Decision Tree:** As a simple, interpretable model that partitions the feature space based on decision rules, Decision Trees offer an intuitive benchmark. While they often lack predictive accuracy compared to ensemble methods, they help illustrate the performance gap between basic and advanced algorithms.

3.3 Hyperparameter Tuning via RandomizedSearchCV

To enhance model performance, we applied `RandomizedSearchCV` from `scikit-learn` to tune hyperparameters for both the Multi-Layer Perceptron (MLP) and XGBoost models. Unlike grid search, which exhaustively evaluates all parameter combinations, randomized search samples a subset of the parameter space, offering a computationally efficient alternative.

For both models, we used 3-fold cross-validation and optimized for the R^2 score on the validation set.

MLP Parameters Tuned:

- `hidden_layer_sizes`: e.g., (32,), (64,), (32, 16), etc.
- `alpha`: L2 regularization term, sampled from $[0.0001, 0.2]$
- `max_iter`: Fixed at 1000
- `early_stopping`: Disabled

XGBoost Parameters Tuned:

- `n_estimators`, `learning_rate`, `max_depth`
- `subsample`, `colsample_bytree`
- `reg_alpha`, `reg_lambda`

This tuning procedure allowed us to identify robust model configurations that generalize well to unseen data.

3.4 Validation Performance Comparison Across Models

Using the hyperparameters obtained via `RandomizedSearchCV`, we trained four models on the training dataset (2005–2016) and evaluated their predictive performance on the validation dataset (2017–2020).

The evaluation was based on two metrics:

- **Root Mean Squared Error (RMSE)**: Measures the average magnitude of the prediction error. Lower RMSE indicates better fit.

table1 Validation Performance Comparison Across Models

Model	Validation RMSE	Validation R^2
MLP	0.12730	0.13454
XGBoost	0.12865	0.11605
Lasso	0.12652	0.14517
Decision Tree	0.12842	0.11919

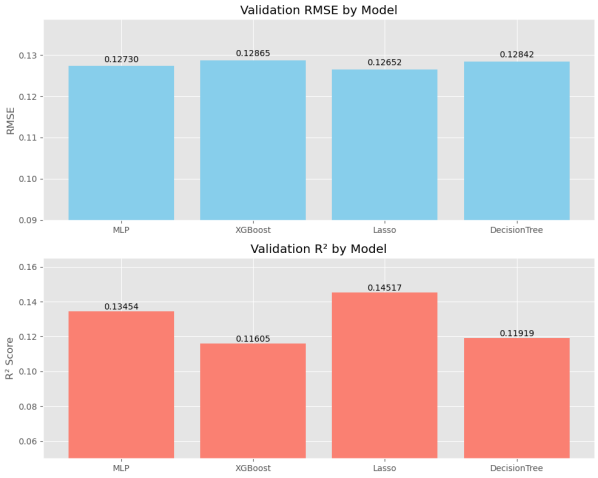

Figure 1: Model Validation Performance (Bar Chart)

- **R-squared (R^2)**: Represents the proportion of variance in the dependent variable that is predictable from the features. Higher R^2 indicates better predictive power.

The results on the validation set are summarized in Table 1.

From this table, we observe that Lasso Regression slightly outperforms the other models in both RMSE and R^2. MLP and XGBoost show nearly identical performance, both superior to Decision Tree.

table2 Test Performance Comparison: MLP vs. Lasso

Model	Test RMSE	Test R^2
Lasso (Train)	0.094490	0.083149
Lasso (Train+Val)	0.094544	0.082105
MLP (Train)	0.094637	0.080301
MLP (Train+Val)	**0.093830**	**0.095919**

3.5 Test Performance Comparison: MLP vs. Lasso

From table2, although Lasso slightly outperformed MLP on the validation set, test performance highlights the superior generalization ability of MLP, especially when trained on the combined training and validation data. This is likely due to MLP's capacity to capture complex nonlinear relationships that extend beyond the validation horizon.

Furthermore, the results confirm that including validation data in training improves model performance across both MLP and Lasso, due to increased data coverage and reduced variance. Consequently, we adopt the MLP (Train+Val) model as our final model for predicting monthly returns.

3.6 Feature Selection via Permutation Importance

To enhance model interpretability and potentially improve generalization, we applied *Permutation Importance* to evaluate and select the most relevant features for our final MLP model. Permuta-

tion Importance is a model-agnostic method that measures the impact of each feature on model performance by randomly shuffling its values and observing the resulting degradation in accuracy. The greater the drop in performance, the more important the feature is deemed to be. Figure

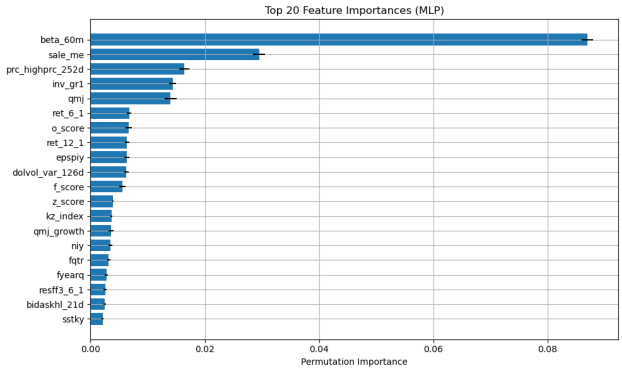

Figure 2: Top 20 Feature Importances

2 shows the top 20 features ranked by their permutation importance values, computed using the MLP model trained on the combined training and validation sets.

Based on the distribution of importance scores, we set a threshold of 0.001 to remove features with negligible contribution. This reduced the dimensionality from 62 features to 24, effectively halving the input space without degrading predictive performance.

This selection not only reduces the risk of overfitting but also facilitates economic interpretation of the most influential variables in return prediction.

4 Final Model Retraining and Test Performance

After selecting the top 24 features via permutation importance, we retrained the MLP model using a new round of hyperparameter tuning with `RandomizedSearchCV`. This ensured that the model architecture remained optimally tailored to the reduced feature space.

Using the newly identified best parameters, the MLP model was trained on the combined training and validation datasets and evaluated on the held-out test set.

The final test performance was as follows:

- Final MLP Test RMSE (optimized, selected features): **0.09309**
- Final MLP Test R^2 (optimized, selected features): **0.11009**

Compared to the previously best-performing MLP configuration using the full 62 features:

- Test RMSE: 0.093830
- Test R^2: 0.095919

this model demonstrated improved predictive accuracy, despite using fewer features.

This result confirms that feature selection based on permutation importance not only enhances model interpretability but also leads to better generalization performance. This result indicates that the final MLP model captures a meaningful portion of the signal in monthly stock returns, despite the noisy nature of financial data.

5 Conclusion

In this project, we developed a predictive model for monthly stock returns by integrating firm-level fundamentals from Compustat with asset pricing factors from the JKP dataset. After extensive preprocessing, feature engineering, and data alignment with CRSP returns, we experimented with multiple models including MLP, XGBoost, Lasso, and Decision Tree.

Model selection based on validation performance suggested that Lasso performed best on the validation set. However, further evaluation on the unseen test set revealed that the MLP model trained on both training and validation data ultimately outperformed all others, highlighting its stronger generalization ability.

We then refined the MLP model by applying permutation importance for feature selection, reducing the feature set from 62 to 24 variables. A final round of hyperparameter tuning and model retraining yielded the best test performance of the project:

- Test RMSE: **0.09309**
- Test R^2: **0.11009**

This result demonstrates that careful preprocessing, rigorous model selection, and feature pruning can significantly enhance the performance of machine learning models in financial return prediction tasks. While the R^2 remains modest—as is common in financial modeling—the incremental improvements underscore the potential of combining structured financial data with machine learning techniques.

Future work may consider incorporating macroeconomic indicators, alternative data sources, or deep learning architectures to further improve predictive performance.

article [utf8]inputenc url hyperref

References

- Scikit-learn. *Scikit-learn: MLPRegressor*. Retrieved June 15, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.neural$_n etwork.MLPRegressor.html$
- Sejal Jaiswal. (2025, April 5). *Multilayer Perceptrons in Machine Learning: A Comprehensive Guide*. DataCamp. Retrieved from https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning
- XGBoost Developers. *XGBoost Python Package*. Retrieved June 15, 2025, from https://xgboost.readthedocs.
- Aayush Tyagi. (2025, April 23). *What is XGBoost Algorithm?* Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
- Jensen, T. I., Kelly, B., & Pedersen, L. H. *Global Factor Data Documentation*. Retrieved June 15, 2025, from https://jkpfactors.s3.amazonaws.com/documents/Documentation.pdf
- Scikit-learn. *Scikit-learn: RandomizedSearchCV*. Retrieved June 15, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.model$_s election.RandomizedSearchCV.html$
- Arindam. (2022, November 2). *Hyperparameter Tuning Using Randomized Search*. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2022/11/hyperparameter-tuning-using-randomized-search/

- Scikit-learn. *5.2. Permutation Feature Importance.* Retrieved June 15, 2025, from https://scikit-learn.org/stable/modules/permutation$_i$*mportance.html*