Enhanced Random Forest Model for Trading Strategy Optimization

Jeff Tuche :)

February 14, 2025

1 Introduction

In our previous approach, we used a **Random Forest Regressor** to predict only the **test Sharpe ratio** based on various trading strategy parameters. However, this approach had several limitations:

- It failed to distinguish between **stable and unstable** parameter sets.
- It did not take into account the **train Sharpe ratio**, which is calculated over a larger dataset (70% of the window).
- It did not explicitly learn from **Sharpe degradation**, which measures overfitting.

To address these issues, we improved our methodology by training the model to predict **three key targets**:

1. **Train Sharpe Ratio** (S_{train}): Performance during the training period.
2. **Test Sharpe Ratio** (S_{test}): Performance on unseen data.
3. **Sharpe Degradation** ($D_{\text{sharpe}} = S_{\text{train}} - S_{\text{test}}$): Stability measure.

By incorporating these three factors, the model can better identify robust parameter sets that generalize well across different time periods.

2 Dataset Structure and Inputs

Each row in the dataset represents an **experiment**, where a particular set of strategy parameters is backtested in a specific walk-forward window.

2.1 Feature Matrix (X)

The input features consist of:

- **Trading strategy parameters** (e.g., moving average window size, envelope thresholds, position sizing).
- **Market conditions** derived from different walk-forward windows.

2.2 Target Variables (y)

Each experiment has three targets:

- $y_{\text{train}} = S_{\text{train}}$
- $y_{\text{test}} = S_{\text{test}}$
- $y_{\text{degradation}} = S_{\text{train}} - S_{\text{test}}$

Here S is the Sharp Ratio.

2.3 Example Data Table

Window ID	Train Sharpe	Test Sharpe	Sharpe Degradation	MA Window	Size	Envelo
1	2.1	1.8	0.3	14	0.05	0.0
1	1.5	1.2	0.3	20	0.07	0.0
1	0.5	0.3	0.2	8	0.10	0.0
2	1.7	1.6	0.1	12	0.04	0.0
2	2.2	2.0	0.2	15	0.06	0.0

Table 1: Example dataset with strategy parameters and performance metrics.

Each row corresponds to a different set of strategy parameters tested in a specific walk-forward window.

3 Random Forest Model Training

We train three separate **Random Forest models**, each predicting one of the three target variables.

3.1 Cross-Validation Process

We use **5-fold cross-validation**, which means:

1. The dataset is split into 5 subsets (folds).
2. The model is trained on 4 subsets and validated on the remaining subset.
3. This process repeats 5 times, with each subset serving as the validation set once.

3.2 Example of Data Splitting

For instance, if we have 5 different walk-forward windows (time periods), the cross-validation setup might look like this:

Training Windows	Validation Window
2018, 2020, 2021, 2022	2019
2019, 2020, 2021, 2022	2018
2018, 2019, 2021, 2022	2020
2018, 2019, 2020, 2022	2021
2018, 2019, 2020, 2021	2022

Table 2: Example of 5-fold cross-validation with different time windows.

This ensures that the model is trained on a variety of market conditions and tested on unseen data.

3.3 Random Forest Hyperparameters

The models are configured as follows:

- **n_estimators = 100**: Number of decision trees in the forest.
- **max_depth = None**: Allows deep trees to capture complex interactions.
- **min_samples_split = 2**: Ensures that nodes are split as long as there are at least two samples.
- **min_samples_leaf = 1**: Allows leaves to contain a single data point.
- **random_state = 42**: Ensures reproducibility.

4 Model Performance and Interpretation

Each fold produces:

- **Train R^2 Score**: Measures how well the model fits the training data.
- **Validation R^2 Score**: Indicates generalization ability.

4.1 Example Results

The low validation scores suggest that predicting test Sharpe is difficult, likely due to high variance.

Fold	Train R^2	Validation R^2
1	0.861	0.030
2	0.858	0.048
3	0.861	0.054
4	0.862	0.052
5	0.866	0.012

Table 3: Cross-validation results for test Sharpe ratio prediction.

5 Why We Train on Three Targets?

Previously, we trained on **only the test Sharpe ratio**, which led to poor generalization. However, the **train Sharpe distribution** showed that some parameter sets performed well in training but poorly in testing.

By also training on:

- **Train Sharpe**: The model can recognize consistently strong parameter sets.
- **Sharpe Degradation**: The model learns to avoid unstable configurations.

5.1 Visualization of Distributions

By combining these insights, we filter out parameter sets that appear promising in training but degrade significantly in testing.

6 Conclusion

By training **three Random Forest models**, we:

- Improve **robustness** by identifying stable parameters.
- Avoid **overfitting** by filtering unstable strategies.
- Prepare for **Gaussian Filtering and Bayesian Optimization** to refine our parameter selection further.

This refined approach enhances our ability to identify **strong and stable trading strategies** for live market conditions.

Figure 1: Distributions of train, test, and Sharpe degradation.