ML Benchmark to ML based Model

Authors: Andrea Landini & Ali Yaghoubi
Supervisor: Phd. Merlin Bartel

AI AND MACHINE LEARNING
UNIVERSITÄT LIECHTENSTEIN

October 3, 2025

Table of Contents

Introduction: which target did we choose and why?

- We aim to target commodity prices, which should improve robustness of results.
- Spread focus (`target_4`): `LME_AH_Close` − `JPX_Gold_Standard_Futures_Close`
  - Aluminum is highly sensitive to global energy costs and industrial activity.
  - Gold serves as a safe-haven asset, reflecting financial risk sentiment.
- Methodology: first apply ML directly, then parametrize a model (still ML-based) to improve performance.

UNIVERSITÄT LIECHTENSTEIN

Framework: Which variable did we include?

We first benchmarked machine learning and econometric models on the **log return spread (Aluminum – Gold)**.

$$r_t = \hat{f}_{\text{Model}}(r_{t-1}, r_{t-2}, \ldots, r_{t-5})$$

Variables:

- Target: spread log return r_t
- Predictors: 5 lags $(r_{t-1}, \ldots, r_{t-5})$
- Naive benchmark: r_{t-1}

Models compared: OLS, Ridge, LASSO, Elastic Net, Random Forest, GBM, XGBoost, SVR, Neural Net, ARIMA.

UNIVERSITÄT LIECHTENSTEIN

Framework: Which variable did we include?

Table 1: Statistical accuracy metrics across ML models

Model	RMSE	MAE	MedAE	R^2	MASE	Corr	sMAPE
OLS	0.0141	0.0104	0.0077	0.0096	0.6591	-0.0979	1.7398
Ridge	0.0140	0.0103	0.0077	NA	0.6493	NA	1.7849
LASSO	0.0140	0.0103	0.0077	NA	0.6493	NA	1.7849
ElasticNet	0.0140	0.0103	0.0077	NA	0.6493	NA	1.7849
RandomForest	**0.0059**	**0.0044**	**0.0031**	**0.9787**	**0.2808**	**0.9893**	**0.6935**
GBM	0.0135	0.0100	0.0076	0.0859	0.6324	0.2932	1.5950
XGBoost	0.0137	0.0103	0.0079	0.0459	0.6545	0.2142	1.7535
SVR	0.0127	0.0091	0.0065	0.2160	0.5749	0.4647	1.3762
NeuralNet	0.0140	0.0103	0.0077	0.0080	0.6493	0.0895	1.7849
ARIMA	0.0140	0.0103	0.0081	0.0025	0.6536	-0.0502	2.0000

Framework: Which model did we choose?

Inspired by Gibson (1990), we extend to a two-factor model for the spread:

$$r_t = \underbrace{f_{\mathrm{ML}}(r_{t-1}, \ldots, r_{t-10}, \mathrm{RollMean}, \mathrm{RollVol})}_{\mu_t} + \underbrace{\sigma_t z_t}_{\text{Volatility factor}}$$

Variables:

- Target: spread log return r_t
- Predictors (Mean factor): 10 lags + rolling means (5,10,20) + rolling volatilities (5,10,20)
- Residuals: $\varepsilon_t = r_t - \mu_t$
- Variance (GARCH): $\sigma_t^2 = \omega + \alpha\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2$

Factors:

1. Machine Learning forecast of mean
2. GARCH forecast of volatility

Two-Factor Models: how did we optimize my model?

Table 2: Comparison of forecasting performance between XGBoost(mean)+GARCH(var) and RandomForest(mean)+GARCH(var).

Metric	XGBoost(mean)+GARCH(var)	RandomForest(mean)+GARCH(var)
RMSE	0.00493	0.00496
MAE	0.00367	0.00372
MedAE	0.00269	0.00295
R^2	0.8750	0.8736
MASE	0.2269	0.2300
Corr	0.9680	0.9683
sMAPE	60.92	61.98

Comparison of forecasting performance between XGBoost(mean)+GARCH(var) and RandomForest(mean)+GARCH(var). XGBoost achieves slightly lower error metrics (RMSE, MAE, MedAE, MASE), while RandomForest attains a marginally higher correlation. Overall, both models deliver nearly identical performance.

UNIVERSITÄT LIECHTENSTEIN

Two-Factor Models: how did we optimize my model?

- **Fixed ML + GARCH**

$$y_t = \hat{\mu}_t^{\text{ML (fixed params)}} + \varepsilon_t, \quad \varepsilon_t \sim \text{GARCH}(1,1)$$

- **Tuned ML + GARCH**

$$y_t = \hat{\mu}_t^{\text{ML (CV-optimized)}} + \varepsilon_t, \quad \varepsilon_t \sim \text{GARCH}(1,1)$$

- Where:
  - $\hat{\mu}_t$: conditional mean from ML (XGBoost/RandomForest)
  - GARCH(1,1): conditional variance of residuals

Two-Factor Models: how did we optimize my model?

- **Initial approach:** Machine learning model with fixed parameters for the mean, combined with GARCH for variance.
- **Improved approach:**
  - Hyperparameters chosen automatically through cross-validation (instead of fixed ad-hoc values).
  - Model adapts to data structure, improving accuracy and robustness.
  - Residuals still modeled with GARCH to capture volatility clustering.
- **Key change:** Machine learning is no longer just applied, but also *optimized by itself.*

UNIVERSITÄT LIECHTENSTEIN

Two-Factor Models: how did we optimize my model?

Table 3: Forecasting performance comparison: Fixed vs. Tuned ML (with GARCH).

	XGBoost		RandomForest	
Metric	Fixed	Tuned	Fixed	Tuned
RMSE	0.00493	0.00379	0.00496	0.00466
MAE	0.00367	0.00295	0.00372	0.00354
MedAE	0.00269	0.00230	0.00295	0.00281
R^2	0.8750	0.9261	0.8736	0.8885
MASE	0.2269	0.1822	0.2300	0.2189
Corr	0.9680	0.9736	0.9683	0.9685
sMAPE	60.92	52.37	61.98	59.35

Observation: Tuning consistently improves XGBoost across all metrics, and yields modest but clear improvements for RandomForest.

Two-Factor Models: what did I do to prevent overfitting?

Table 4: Train vs Test performance across ML models (rolling CV). Shaded rows indicate overfitting: RandomForest shows severe overfitting (very low train error but weak generalization), while GBM and XGBoost show moderate overfitting. Linear models, ARIMA, and the small NeuralNet display no overfitting but underfit (train and test errors similar and relatively high).

Model	Train RMSE	Test RMSE	Train MAE	Test MAE	Train R^2	Test R^2
ARIMA	0.0142	0.0140	0.0106	0.0111	0.0280	0.0817
ElasticNet	0.0142	0.0141	0.0106	0.0111	0.0154	0.0455
GBM	0.0120	0.0145	0.0092	0.0115	0.3640	0.0474
LASSO	0.0142	0.0141	0.0106	0.0111	0.0154	0.0451
NeuralNet	0.0142	0.0141	0.0106	0.0111	0.0114	0.0464
OLS	0.0141	0.0142	0.0105	0.0112	0.0130	0.0452
RandomForest	**0.0069**	**0.0144**	**0.0051**	**0.0114**	**0.950**	**0.0410**
Ridge	0.0141	0.0141	0.0105	0.0111	0.0152	0.0322
SVR	0.0127	0.0145	0.0092	0.0115	0.2410	0.0559
XGBoost	0.0110	0.0146	0.0085	0.0116	0.4740	0.0543

UNIVERSITÄT LIECHTENSTEIN

Two-Factor Models: what did I do to prevent overfitting?

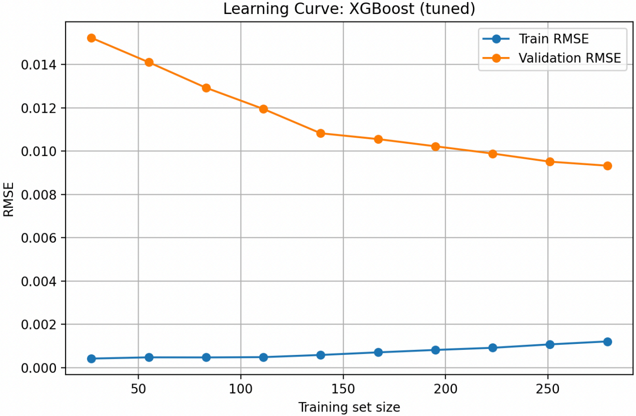
Table 5: Comparison of training and test performance for tuned ML models. Small gaps between train and test indicate no overfitting.

Model	Dataset	RMSE	MAE	MedAE	R^2	MASE	Corr	sMAPE
XGBoost	Train	0.00363	0.00287	0.00239	0.941	0.177	0.980	49.70
XGBoost	Test	0.00379	0.00295	0.00230	0.926	0.182	0.974	52.37
RandomForest	Train	0.00488	0.00343	0.00261	0.893	0.212	0.977	51.35
RandomForest	Test	0.00466	0.00354	0.00281	0.888	0.219	0.969	59.35

Both models show nearly identical train and test errors. XGBoost performs best overall, with no evidence of overfitting.

UNIVERSITÄT LIECHTENSTEIN

Learning Curve: XGBoost (tuned)



- The gap between curves is narrowing, indicating no strong overfitting.
- Model would likely benefit from even more data for further convergence.
- Train RMSE remains very low and stable across sample sizes.
- Validation RMSE decreases steadily as training size increases.

UNIVERSITÄT LIECHTENSTEIN

Project Repository

You can find the project repository at:

`https://github.com/andrealandini/ai-and-machine-learning`

UNIVERSITÄT LIECHTENSTEIN

Contact

Thanks for your attention.

Andrea Landini	andrea.landini@uni.li
	`https://andrealandini.info`
Ali Yaghoubi	ali.yaghoubi@uni.li

UNIVERSITÄT LIECHTENSTEIN