Supervised Autoencoder MLP for Financial Time Series Forecasting

Bartosz Bieganowski, Robert Ślepaczuk

University of Warsaw, Faculty of Economic Sciences
Department of Quantitative Finance and Machine Learning, Quantitative Finance Research Group

March 4, 2024

Table of Contents

Aim

The aim of this study is to verify whether the following machine-learning related techniques can improve trading strategy performance:

- **Noise augmentation** - originally developed for Computer Vision problems, it has been noted that adding noise to the input data helps with generalization on image classification tasks.
- **Supervised Autoencoder** - originally developed for Natural Language Processing problems, we test if SAE-MLP architecture can be applied in algorithmic trading strategies.
- **Triple Barrier Labeling** - although already mentioned in the literature, we expand on this specific labeling method by developing an optimization metric that resembles the strategy return better.

Research Questions

- **RQ1: Does noise augmentation used with SAE-MLP architecture improve strategy performance as expressed by Information Ratio?**
- **RQ2: Does the triple barrier labeling with correct optimization metric improve strategy performance as expressed by Information Ratio?**
- **RQ3: Does the hyperparameter tuning improve strategy performance with SAE-MLP architectures as expressed by Information Ratio?**

Literature Review

- Efficient Market Hypothesis (EMH) suggests stock prices reflect all available information, making them unpredictable.
- Studies by Fama (1970) and Malkiel (2005) support EMH, while Barberis and Thaler (2002) suggest market inefficiencies.
- Machine learning (ML) techniques like LSTM outperform traditional methods in stock price prediction (Kryńska and Ślepaczuk, 2022).
- LSTM models show promise in forecasting, but challenges remain in handling non-stationary data and parameter sensitivity.
- Hybrid models combining LSTM and GRU demonstrate improved performance in forecasting financial assets (Baranovhnikov and Ślepaczuk, 2022)
- ML models, particularly deep learning, excel in predicting Bitcoin prices, indicating their relevance in cryptocurrency trading (Michanków et. al., 2022).

Data

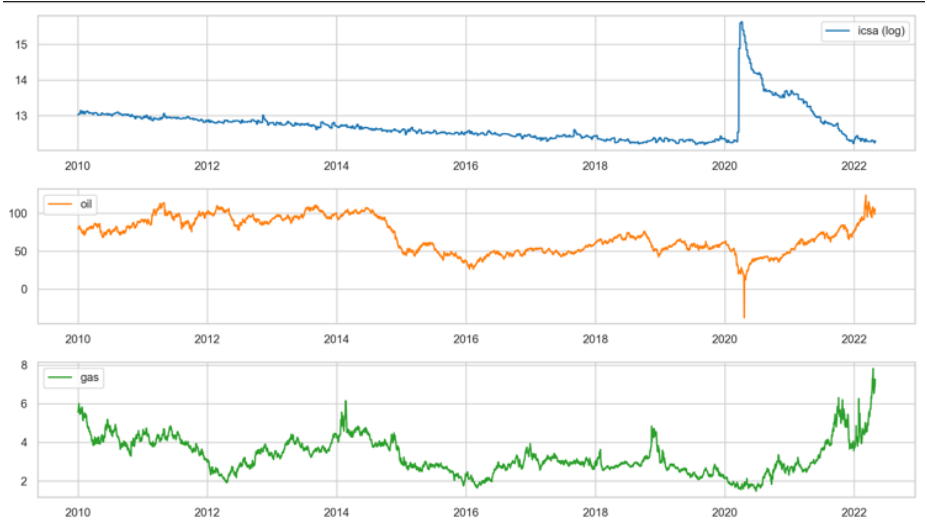
- S&P500 - Low volatility compared to individual stocks, correlated with economic growth indicators, right-skewed return distribution.
- EUR/USD - Moderate volatility, driven by monetary policy of UE and FED, and indicators from both regions, returns close to normal with leptokurtosis.
- BTC/USD - High volatility, driven by speculation, technological developments. Low correlation with traditional financial assets. Returns skewed and highly leptokurtic.

Training timeframe: 2010-01-01 - 2019-12-31
Testing timeframe: 2020-01-01 - 2022-04-30

Toolset

- **Hardware**: GeForce RTX 2080 SUPER, Intel Core i7-9700K, Patriot 32GB RAM
- **Software**: Python 3.10, Tensorflow, Pandas, Matplotlib, Scikit-learn
- **Computation Time**: on average 4 minutes per 1 hyperparameter combination

Features - ICSA, Oil, Gas



Features - Corn, Gold, Copper

Features - Presumed Impact on Economy

Feature	Presumed Increase Impact	Presumed Decrease Impact
ICSA	Negative: Indicates rising unemployment, potential economic slowdown	Positive: Suggests decreasing unemployment, potential economic growth
Oil	Mixed: Benefits oil exporters, increases costs for importers and consumers	Mixed: Lowers costs for importers and consumers, but may harm oil-exporting economies
Gas	Negative: Increases energy costs, affects consumer spending and production costs	Positive: Decreases energy costs, boosts consumer spending and lowers production costs
Corn	Negative: Raises food and feed prices, impacts food industry and inflation	Positive: Lowers food and feed prices, beneficial for food industry and inflation control
Gold	Mixed: Often seen as a safe haven, increase may indicate economic uncertainty	Mixed: Decrease may reflect investor confidence, but could impact gold-producing economies
Copper	Positive: Suggests industrial growth and demand, often a positive economic indicator	Negative: May indicate reduced industrial activity and economic slowdown

Feature Engineering Question

What do we do with our features before we put them into the machine learning model?

- Should we differentiate the time series (d=1, losing the memory aspect)?
- Should we input it as-is (d=0, but data is not stationary)?

Fractionally differentiated features

We can apply ARFIMA (Granger, C. W. J.; Joyeux, Roselyne, 1980) assumptions to machine learning features. We consider the backshift operator B applied to a time series of a feature fX_tg such that $B^kX_t = X_{t-k}$.

It follows that the difference between current and last feature's value can be expressed as $(1 \quad B)^X_t$. For example, $(1 \quad B)^2 = 1 \quad 2B + B^2$, where $B^2X_t = X_{t-2}$ so that $(1 \quad B)^2X_t = X_t \quad 2X_{t-1} + X_{t-2}$.

For any positive integer n, it also holds that:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k \tag{1}$$

Fractionally differentiated features

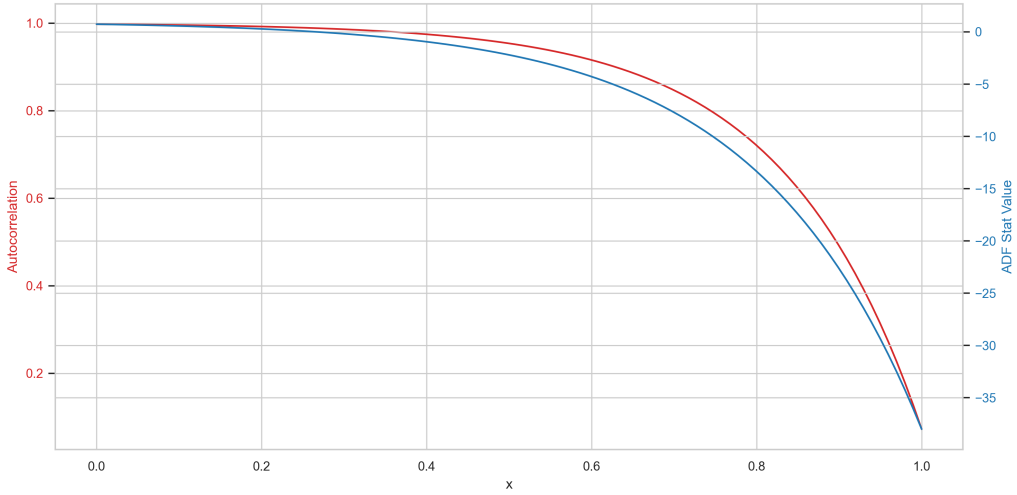
On the other hand for any real number d:

$$(1+x)^d = \sum_{k=0}^{1} \binom{d}{k} x^k \tag{2}$$

is the binomial series. In a model where d is allowed to be a real number, the binomial series can be expanded into a series of weights which can be applied to feature values:

$$\omega = \left\{1, \quad d, \frac{d(d \quad 1)}{2!}, \frac{d(d \quad 1)(d \quad 1)}{3!}, ..., (\quad 1)^k \prod_{i=0}^{k} \frac{1}{k!} d \quad i \right\} \tag{3}$$

Optimal Differentiation Order



Optimal Differentiation Order

Algorithm 1 Fractional Feature Differentiation in Walk-Forward Validation

Set a range of possible values for d (e.g., from 0 to 1)
Set significance level for ADF test (e.g., 1%)
Initiate a dictionary associating each feature with optimal d.
for each segment pair (train, test) **do**
    for each feature **do**
        Apply fractional differencing to train segment of feature at discrete intervals
        Calculate ADF test statistic and p-value for each d for a feature
        Choose lowest d such that p-value ¡ significance level.
        Save feature name and associated optimal d to dictionary
        Apply optimal d differencing to both train and test set of the feature
    end for
    Train the model on train segment, evaluate on test set
end for

Triple Barrier Labeling

Whenever we try to express trading problem as a machine learning problem, we have to think long and hard about what do we want our model to really predict (Y).

- Regression on price in x time? (unstationary, ignores path, uninformative error metrics).
- Regression on return over x time? (maybe stationary, ignores path, no directional sensitivity unless using custom loss metrics like MADL).
- Classification on movement direction? (better, still ignores path, high noise-to-signal).

Triple Barrier Labeling

Path-dependent classification, which is effectively ML-interpretation of concepts of *stop-loss*, *take-pro t*, and *timed-exit*.

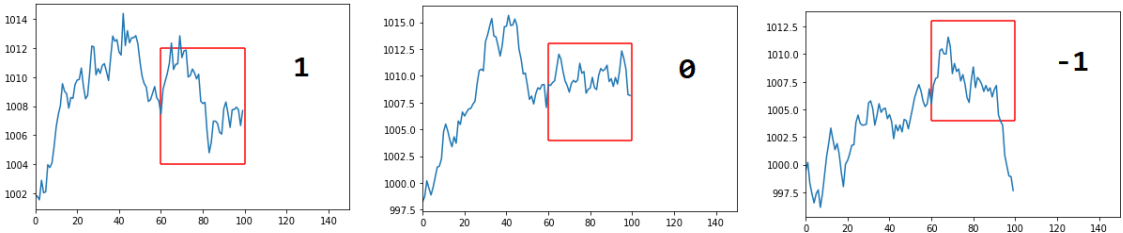


Figure: Exemplary labels in triple-barrier-labeling

Triple Barrier Labeling

$$P_t = \begin{cases} 1, & \text{if } \max(S_t, ..., S_{t+n}) \quad S_t \quad (1+\lambda) \\ 1, & \text{if } \min(S_t, ..., S_{t+n}) \quad S_t \quad (1 \quad \lambda) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

λ - window size in (%)

(Idea: λ was a constant for this study, but it might work well to base it on an estimate of future volatility)

Payoff Table

Table 2. Return on a trade given classification result.

Pred/True	1	0	-1
1	λ	(λ, λ)	λ
0	0	0	0
-1	λ	(λ, λ)	λ

Source: Own Elaboration

Derived Optimization Metric

We define *directly correct count* as the number of times the model entered correct position which resulted in return of λ. We can similarly define *directly incorrect count* as the number of times the model entered incorrect position:

$$DCC = jf(Y_{\text{pred}}, Y_{\text{true}}) \; 2 \; S \; j \; Y_{\text{pred}} \; 6= 0 \text{ and } Y_{\text{pred}} = Y_{\text{true}} gj \tag{5}$$

$$DIC = jf(Y_{\text{pred}}, Y_{\text{true}}) \; 2 \; S \; j \; Y_{\text{pred}} \; 6= 0 \text{ and } Y_{\text{pred}} \; 6= \; Y_{\text{true}} gj \tag{6}$$

Where jSj is the cardinality of set S.

Derived Optimization Metric

Basic optimization metric Φ:

$$\Phi = \prod_{1}^{DCC}(1+\lambda) \quad \prod_{1}^{DIC}(1 \quad \lambda) = (1+\lambda)^{DCC} \quad (1 \quad \lambda)^{DIC} \tag{7}$$

Optimization metric with δ dictating error preference strength:

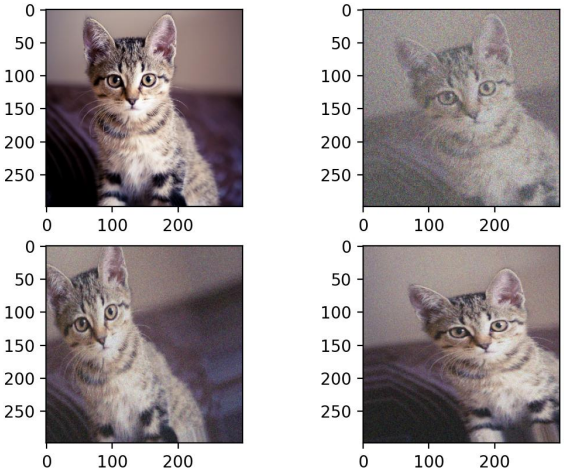
$$\Phi_{\delta} = (1+\lambda)^{DCC} \quad (1 \quad \lambda)^{DIC} \quad \left(1 \quad \frac{\lambda}{\delta}\right)^{TEC} \tag{8}$$

where $\delta > \lambda$. In our study, we set δ arbitrarily to 20, indicating that twenty timed exits are considered equally undesirable as one direct incorrect classification

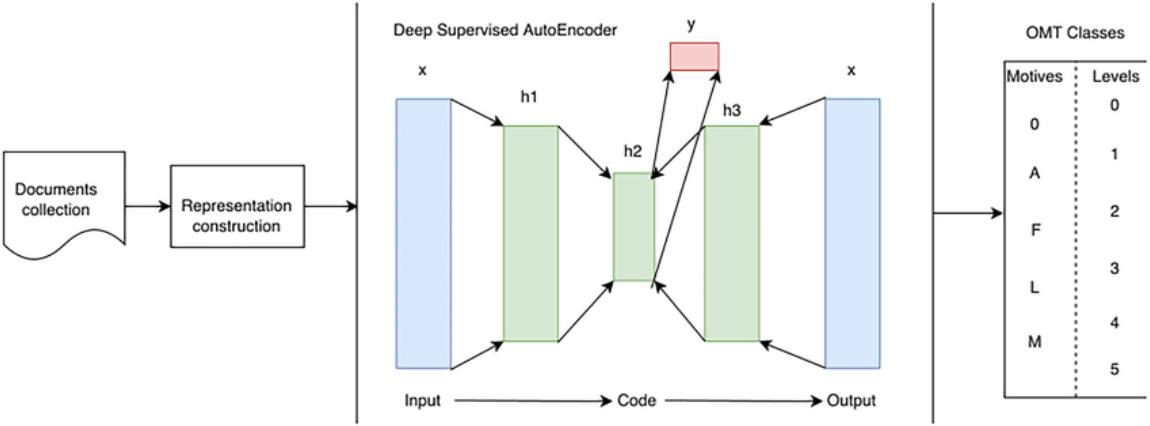
(Note: Accurate prediction of zeros could also be taken advantage of with an option butterfly)

Data Augmentation in CV

Can we apply the concept to financial time series?



Supervised Autoencoder



Supervised Autoencoder

- **Enhanced Feature Representation:** Supervised autoencoders can learn more relevant and discriminative features for the task at hand because they are trained to not only reconstruct the input data but also to optimize for an additional task-specific loss (like classification or regression).

- **Regularization E ect:** Incorporating the reconstruction objective alongside the task-specific objective (like classification accuracy) can act as a form of regularization. This helps in preventing overfitting to the training data by ensuring that the learned representations maintain information about the input data, leading to more generalized models.

- **E ciency in Data Use:** By leveraging unlabeled data for the reconstruction part and labeled data for the task-specific part, supervised autoencoders can make efficient use of datasets where obtaining labeled data is expensive or time-consuming. This can be particularly beneficial in semi-supervised learning scenarios, where the model can learn general features from a large pool of unlabeled data and fine-tune the representations for the task with a smaller set of labeled examples.

Approaches Comparison

Table 4: Comparison of Different Approaches

	Approach 1	Approach 2	Approach 3	Approach 4
Activation	tanh	tanh	swish	swish
Loss	mse	log-loss	log-loss	log-loss
Epochs	50	50	50	50
Learning Rate	0.01	0.01	0.01	0.01
Hidden Layers	1	1	1	1
Gaussian Noise Rate	0.00	0.00	0.05	0.05
Problem Type	regression	classification	classification	classification
Model Type	Base Model	+ Classification	+SAE/Denoising	+SAE/Denoising +TBL

Source: Own Elaboration

Drawdown-adjusted information ratio

We use the information ratio as our main metric, originally proposed by Kość et al. (2019) which is a modification of the Information Ratio measure. This measure also takes into account the sign of the portfolio's rate of return and the maximum drawdown:

$$IR = \frac{ARC^2 \ sign(ARC)}{ASD \ MDD} \tag{9}$$

ARC - Annualized Return Compounded
ASD - Annualized Standard Deviation
MDD - Maximum Drawdown

Results - Eq. Weight Portfolio of Strategies - IRR**

Figure 25: Comparison of information ratios across approaches and bar lengths

Bar Length	Approach 1	Approach 2	Approach 3	Approach 4
5min	-1.09	-0.11	0.54	0.26
15min	-0.17	0.53	1.71	2.06
30min	1.07	0.55	1.24	1.6

Source: Own Elaboration, table produced in Microsoft Excel.

Sensitivity Analysis - Triple Barrier Labelling

Y - window height λ X-window length (minutes)

	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
0.10%	-0.17	0.55	1.55	1.60	1.56	1.48	1.36	1.40	1.58	1.64	1.66	1.55	1.51	1.30	1.29	1.37	1.41	1.40
0.11%	-0.33	0.87	1.71	1.73	1.66	1.49	1.35	1.48	1.58	1.53	1.39	1.37	1.44	1.25	1.31	1.18	1.01	1.15
0.12%	-0.27	0.42	1.80	1.64	1.53	1.37	1.24	0.99	1.00	0.82	0.77	0.68	0.46	0.36	0.20	0.09	-0.19	-0.21
0.13%	-0.40	0.89	1.70	1.64	1.52	1.38	1.06	1.03	1.07	1.18	1.06	1.12	1.14	1.09	1.13	1.09	1.19	1.17
0.14%	-0.73	0.49	1.59	1.62	1.43	1.23	1.12	1.12	1.00	0.92	0.89	0.79	0.91	1.08	1.15	0.93	0.96	1.05
0.15%	-0.55	0.45	1.69	1.61	1.50	1.48	1.57	1.61	1.55	1.55	1.48	1.58	1.61	1.46	1.52	1.38	1.24	1.17
0.16%	0.00	0.86	1.63	1.58	1.60	1.45	1.37	1.16	1.05	1.07	0.99	1.00	1.06	0.84	0.62	0.49	0.57	0.58
0.17%	-0.02	0.73	1.59	1.50	1.41	1.39	1.34	1.49	1.25	1.28	1.17	1.13	1.09	0.98	0.92	1.04	0.79	0.76
0.18%	-0.15	0.97	1.92	2.02	2.09	1.97	1.91	1.74	1.61	1.38	1.33	1.44	1.28	1.12	0.88	0.81	0.69	0.72
0.19%	-0.34	1.02	2.05	1.98	2.03	2.02	2.04	2.03	1.98	1.99	1.90	1.75	1.60	1.64	1.67	1.75	1.53	1.57
0.20%	0.22	1.30	2.06	2.00	1.75	1.75	1.56	1.48	1.20	1.27	1.23	1.26	1.00	0.78	0.78	0.63	0.57	0.64
0.21%	-0.10	1.03	2.12	1.96	1.89	1.86	1.74	1.81	1.77	1.64	1.64	1.65	1.68	1.64	1.54	1.52	1.48	1.12
0.22%	0.33	0.98	2.00	2.07	1.99	1.96	1.90	1.87	1.68	1.56	1.59	1.54	1.53	1.58	1.55	1.56	1.49	1.48
0.23%	-0.04	1.01	1.90	2.06	1.90	1.74	1.88	1.87	1.84	1.86	1.80	1.81	1.85	1.91	1.85	1.82	1.71	1.69
0.24%	-0.04	0.80	1.80	1.85	1.80	1.88	2.03	2.11	2.00	1.96	1.94	1.83	1.81	1.84	1.87	1.64	1.57	1.45
0.25%	-0.46	0.72	1.70	1.63	1.72	1.81	1.76	1.76	1.74	1.59	1.45	1.54	1.72	1.59	1.38	1.25	1.09	0.93
0.26%	0.02	0.76	1.60	1.56	1.74	1.65	1.62	1.58	1.43	1.50	1.33	1.22	1.32	1.34	1.35	1.34	1.32	1.31
0.27%	-0.05	0.68	1.62	1.49	1.48	1.38	1.26	1.19	1.12	1.07	0.91	0.89	0.68	0.50	0.33	0.31	0.19	0.08
0.28%	-0.04	0.83	1.33	1.29	1.22	1.01	0.87	0.81	0.80	0.85	0.95	1.06	0.88	0.85	0.96	0.93	0.76	0.80
0.29%	-0.79	0.10	1.35	1.07	1.17	1.09	1.09	1.09	1.07	1.24	1.26	1.21	1.03	0.94	0.91	1.03	0.84	0.98
0.30%	-0.33	0.66	1.46	1.51	1.58	1.49	1.50	1.47	1.53	1.57	1.43	1.35	1.35	1.43	1.44	1.39	1.34	1.45

Sensitivity Analysis - Supervised Autoencoder

Y - Gaussian nosie rate X - Bottleneck size (% of feature count)

	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
0%	1.46	1.46	1.46	1.47	1.46	1.46	1.45	1.44	1.41	1.41
2%	1.62	1.68	1.67	1.66	1.65	1.65	1.64	1.64	1.60	1.59
3%	1.77	1.89	1.87	1.86	1.85	1.84	1.84	1.82	1.79	1.79
4%	1.75	1.94	1.92	1.92	1.91	1.91	1.91	1.90	1.89	1.87
5%	2.07	2.06	2.04	2.03	2.02	2.02	2.03	2.02	2.00	1.98
6%	2.09	2.07	2.06	2.03	2.03	2.02	2.00	2.00	1.99	2.00
7%	1.78	1.93	1.93	1.92	1.91	1.90	1.90	1.89	1.87	1.88
8%	1.96	1.95	1.94	1.93	1.92	1.92	1.91	1.90	1.88	1.88
9%	1.80	1.95	1.95	1.93	1.93	1.90	1.89	1.87	1.88	1.87
10%	1.70	1.88	1.87	1.87	1.87	1.87	1.86	1.84	1.83	1.84
11%	1.45	1.65	1.64	1.62	1.63	1.63	1.62	1.61	1.61	1.59
12%	1.56	1.52	1.52	1.50	1.48	1.48	1.47	1.47	1.46	1.45
13%	1.17	1.29	1.28	1.27	1.27	1.29	1.28	1.27	1.24	1.24
14%	1.13	1.32	1.29	1.30	1.28	1.28	1.25	1.23	1.21	1.18
15%	1.10	1.31	1.29	1.30	1.28	1.26	1.24	1.23	1.23	1.22
16%	1.04	1.09	1.10	1.09	1.09	1.08	1.07	1.06	1.05	1.04
17%	0.82	0.94	0.92	0.91	0.90	0.90	0.88	0.87	0.85	0.83
18%	0.76	1.02	1.01	1.00	0.99	0.99	0.97	0.99	0.99	0.99
19%	0.91	0.96	0.94	0.95	0.94	0.92	0.90	0.89	0.89	0.87
20%	1.06	1.03	1.03	1.03	1.02	1.02	1.02	1.01	1.01	0.99

Sensitivity Analysis - Supervised Autoencoder

Y - Encoder hidden layer count - Decoder hidden layer count

	1	2	3	4
1	2.061	2.010	1.983	1.814
2	1.647	2.100	1.792	1.491
3	1.962	1.689	1.980	1.492
4	1.905	1.802	1.860	2.156

Research Question Findings

- **RQ1: Impact of Data Augmentation and Denoising**
  - Data augmentation (Gaussian noise) and denoising (autoencoders) significantly improve strategy performance.
  - Approach 3 excels over Approaches 1 & 2 in Information Ratio for all bar lengths.
  - Optimal noise level and autoencoder size are critical; relationship is non-linear, requiring careful calibration.
- **RQ2: E cacy of Triple Barrier Labelling**
  - Triple barrier labelling surpasses simple direction classification, enhancing market noise handling and optimization metrics.
  - Approach 4 outperforms others in 15 and 30-minute bars but falls short in high-frequency (5-minute bars) trading scenarios.
- **RQ3: Role of Hyperparameter Tuning**
  - Crucial for superior investment strategy performance; optimal results with specific noise levels and autoencoder sizes.

Further elaboration ideas

- **Dynamic lambda** - setting lambda (TBL window size) to be a dynamic estimate of future volatility.
- **Dynamic window length** - setting dynamic length size based on estimate of market activity.
- **Zero-classi cations** - Accurate predictions of the price staying the same can be taking advantage of with options (theta decay).
- **Other architectures** - More elaborate models than MLP can be stacked on top of SAE (Random Forest, ADABoost, CatBoost).
- **Feature engineering** - more elaborate feature engineering to see how SAE reacts to greater number of features.

Conclusion

Thank you!

Q&A