# A Comparative Analysis of Linear Regression and DLinear for Time Series Forecasting: From Synthetic Benchmarks to Economic Indicators

Sofien Kaabar, CFA
sofien-kaabar@hotmail.com

February 5, 2026

**Abstract**

Time series forecasting remains a fundamental challenge across scientific and economic domains. While deep learning architectures have dominated recent literature, emerging evidence suggests that simpler linear models may achieve competitive or superior performance on many forecasting tasks. This paper presents a systematic comparison between classical Linear Regression and DLinear, a decomposition-based linear neural network architecture. We evaluate both approaches across three experimental settings: clean synthetic time series with known generating processes, noisy synthetic data with controlled signal-to-noise ratios, and real-world US core inflation data (CPILFESL). Performance is assessed using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Our findings reveal that while DLinear's trend-seasonal decomposition provides advantages in structured time series, Linear Regression demonstrates slightly better robustness in noisy environments and offers superior interpretability. These results contribute to the ongoing discussion regarding the appropriate complexity of forecasting models and provide practical guidance for practitioners selecting methods for economic time series prediction.

**Keywords:** Time series forecasting, Linear Regression, DLinear, Economic indicators, Model comparison, Inflation forecasting

# 1 Introduction

Time series forecasting constitutes one of the most consequential problems in statistical learning, with applications spanning financial markets, meteorology, epidemiology, and macroeconomic policy. The ability to accurately predict future values based on historical observations directly influences investment decisions, resource allocation, and strategic planning across public and private sectors.

The past decade witnessed a proliferation of deep learning approaches to time series forecasting. Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), Temporal Convolutional Networks (TCNs), and Transformer-based architectures have successively claimed state-of-the-art performance on various benchmarks [Hochreiter and Schmidhuber, 1997, Vaswani et al., 2017]. These models promise to capture complex nonlinear dependencies and long-range temporal patterns that classical methods allegedly cannot represent.

However, recent critical examinations have challenged this narrative. Zeng et al. [2023] demonstrated that simple linear models can outperform sophisticated Transformer architectures on multiple long-term forecasting benchmarks. This finding prompted a reevaluation of whether the complexity introduced by deep learning models is justified by commensurate improvements in predictive accuracy. The question is particularly pertinent for economic time series, where interpretability carries substantial value and overfitting to noise can produce misleading forecasts with costly consequences.

This paper contributes to this ongoing discourse by conducting a comparative analysis between two approaches occupying different positions on the complexity spectrum: classical Linear Regression and DLinear. Linear Regression represents the foundational approach to supervised learning, offering closed-form solutions, well-understood statistical properties, and direct interpretability of coefficients. DLinear, introduced by Zeng et al. [2023], occupies an intermediate position as it employs neural network machinery but restricts itself to linear transformations applied after decomposing the input into trend and seasonal components.

Our experimental design encompasses three distinct settings chosen to illuminate different aspects of model behavior:

1. **Clean Synthetic Time Series:** We generate data from known processes including autoregressive models, periodic functions, and combinations thereof. These experiments establish baseline performance where the true data-generating mechanism is accessible and noise is absent.

2. **Noisy Synthetic Time Series:** We introduce controlled Gaussian noise at varying signal-to-noise ratios to assess model robustness. This setting reveals how each approach degrades as the predictable signal becomes increasingly obscured.

3. **Economic Indicators:** We apply both methods to the US Core Inflation measure (CPILFESL).

We evaluate performance using a comprehensive suite of metrics: Mean Squared Error (MSE) quantifies average squared deviations; Root Mean Squared Error (RMSE) returns this to the original scale; and Mean Absolute Error (MAE) provides a robust alternative less sensitive to outliers.

The remainder of this paper is organized as follows. Section 2 provides a detailed exposition of Linear Regression for time series forecasting, including its mathematical

formulation, estimation procedures, and assumptions. Section 3 presents DLinear, explaining its decomposition strategy, architecture, and training methodology. Section 4 describes our experimental setup and data sources. Section 5 presents results across all experimental conditions.

# 2 Linear Regression for Time Series Forecasting

## 2.1 Mathematical Formulation

Linear Regression posits a linear relationship between a dependent variable and one or more independent variables. In the context of univariate time series forecasting, we seek to predict future values $y_{t+h}$ based on a window of $L$ past observations $(y_{t-L+1}, y_{t-L+2}, \ldots, y_t)$, where $h$ denotes the forecast horizon.

The model takes the form:

$$\hat{y}_{t+h} = \beta_0 + \sum_{i=1}^{L} \beta_i y_{t-L+i} + \epsilon_t \tag{1}$$

where $\beta_0$ is the intercept, $\beta_1, \ldots, \beta_L$ are the coefficients associated with each lagged observation, and $\epsilon_t$ represents the error term.

Expressing this in matrix notation, let $\mathbf{X} \in \mathbb{R}^{N \times (L+1)}$ denote the design matrix where each row contains a bias term and $L$ lagged values, and let $\mathbf{y} \in \mathbb{R}^N$ denote the vector of target values. The model becomes:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{2}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_L)^\top$ is the parameter vector.

## 2.2 Parameter Estimation via Ordinary Least Squares

The Ordinary Least Squares (OLS) estimator minimizes the sum of squared residuals:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \tag{3}$$

Taking the derivative with respect to $\boldsymbol{\beta}$ and setting it to zero yields the normal equations:

$$\mathbf{X}^\top \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}^\top \mathbf{y} \tag{4}$$

When $\mathbf{X}^\top \mathbf{X}$ is invertible, the closed-form solution is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{5}$$

This estimator possesses desirable properties under the Gauss-Markov assumptions. Among all linear unbiased estimators, OLS achieves minimum variance—the Best Linear Unbiased Estimator (BLUE) property.

## 2.3 Regularization

When the number of lagged features $L$ is large relative to the sample size $N$, or when lagged values exhibit high multicollinearity, regularization becomes essential to prevent overfitting.

**Ridge Regression** ($L_2$ regularization) adds a penalty proportional to the squared magnitude of coefficients:

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\} \tag{6}$$

with closed-form solution:

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} \tag{7}$$

**Lasso Regression** ($L_1$ regularization) induces sparsity by penalizing the absolute values of coefficients:

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\} \tag{8}$$

Lasso lacks a closed-form solution and requires iterative optimization, but it offers automatic feature selection by driving some coefficients exactly to zero.

**Elastic Net** combines both penalties:

$$\hat{\boldsymbol{\beta}}_{\text{elastic}} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2^2 \right\} \tag{9}$$

# 3 DLinear: Decomposition-Based Linear Forecasting

## 3.1 Motivation and Background

DLinear emerged from a provocative finding: on long-term forecasting benchmarks, simple linear models outperformed Transformers and other complex architectures [Zeng et al., 2023]. This result challenged the prevailing assumption that temporal pattern recognition requires sophisticated nonlinear modeling.

The key insight underlying DLinear is that many time series can be effectively represented as the sum of a slowly-varying trend component and a periodic seasonal component. Rather than learning to disentangle these patterns implicitly through deep networks, DLinear performs explicit decomposition and applies separate linear transformations to each component.

## 3.2 Architecture

DLinear operates on input sequences of length $L$ to produce forecasts of length $H$. The architecture consists of three stages: decomposition, linear projection, and aggregation.

### 3.2.1 Moving Average Decomposition

The input sequence $\mathbf{x} \in \mathbb{R}^L$ is decomposed using a moving average filter. Let $k$ denote the kernel size (typically chosen based on known seasonality). The trend component is extracted as:

$$\mathbf{x}_{\text{trend}}(t) = \frac{1}{k} \sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \mathbf{x}(t+i) \tag{10}$$

with appropriate padding at boundaries.

The seasonal (or remainder) component is obtained by subtraction:

$$\mathbf{x}_{\text{seasonal}} = \mathbf{x} - \mathbf{x}_{\text{trend}} \tag{11}$$

This decomposition is parameter-free and does not require learning, making it computationally efficient and stable.

#### 3.2.2 Linear Projection Layers

Each component is processed by a dedicated linear layer. For the trend component:

$$\hat{\mathbf{y}}_{\text{trend}} = \mathbf{W}_{\text{trend}}\mathbf{x}_{\text{trend}} + \mathbf{b}_{\text{trend}} \tag{12}$$

where $\mathbf{W}_{\text{trend}} \in \mathbb{R}^{H \times L}$ and $\mathbf{b}_{\text{trend}} \in \mathbb{R}^{H}$ are learnable parameters.

Similarly, for the seasonal component:

$$\hat{\mathbf{y}}_{\text{seasonal}} = \mathbf{W}_{\text{seasonal}}\mathbf{x}_{\text{seasonal}} + \mathbf{b}_{\text{seasonal}} \tag{13}$$

where $\mathbf{W}_{\text{seasonal}} \in \mathbb{R}^{H \times L}$ and $\mathbf{b}_{\text{seasonal}} \in \mathbb{R}^{H}$.

#### 3.2.3 Aggregation

The final prediction is the sum of the projected components:

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}_{\text{trend}} + \hat{\mathbf{y}}_{\text{seasonal}} \tag{14}$$

### 3.3 Mathematical Formulation

Combining the above, DLinear can be expressed as:

$$\hat{\mathbf{y}} = \mathbf{W}_{\text{trend}} \cdot \text{AvgPool}(\mathbf{x}) + \mathbf{b}_{\text{trend}} + \mathbf{W}_{\text{seasonal}} \cdot (\mathbf{x} - \text{AvgPool}(\mathbf{x})) + \mathbf{b}_{\text{seasonal}} \tag{15}$$

Rearranging:

$$\hat{\mathbf{y}} = (\mathbf{W}_{\text{trend}} - \mathbf{W}_{\text{seasonal}}) \cdot \text{AvgPool}(\mathbf{x}) + \mathbf{W}_{\text{seasonal}} \cdot \mathbf{x} + \mathbf{b}_{\text{trend}} + \mathbf{b}_{\text{seasonal}} \tag{16}$$

This reveals that DLinear is mathematically equivalent to a linear model, but with structure induced by the decomposition. The decomposition acts as an inductive bias, encouraging the model to treat trend and seasonal patterns differently.

### 3.4 Training Procedure

DLinear is trained via gradient descent to minimize a loss function, typically Mean Squared Error:

$$\mathcal{L} = \frac{1}{NH} \sum_{n=1}^{N} \sum_{h=1}^{H} (y_{n,h} - \hat{y}_{n,h})^2 \tag{17}$$

The training procedure follows standard neural network practice:

**Algorithm 1** DLinear Training

Initialize weights $\mathbf{W}_{\text{trend}}, \mathbf{W}_{\text{seasonal}}, \mathbf{b}_{\text{trend}}, \mathbf{b}_{\text{seasonal}}$
**for** epoch $= 1$ to $E$ **do**
  **for** each mini-batch $(\mathbf{X}, \mathbf{Y})$ **do**
    Compute trend: $\mathbf{X}_{\text{trend}} \leftarrow \text{AvgPool}(\mathbf{X})$
    Compute seasonal: $\mathbf{X}_{\text{seasonal}} \leftarrow \mathbf{X} - \mathbf{X}_{\text{trend}}$
    Compute predictions: $\hat{\mathbf{Y}} \leftarrow \mathbf{W}_{\text{trend}}\mathbf{X}_{\text{trend}} + \mathbf{W}_{\text{seasonal}}\mathbf{X}_{\text{seasonal}} + \mathbf{b}$
    Compute loss: $\mathcal{L} \leftarrow \text{MSE}(\mathbf{Y}, \hat{\mathbf{Y}})$
    Backpropagate and update parameters
  **end for**
**end for**

Common training choices include:

- **Optimizer:** Adam with learning rate in $[10^{-4}, 10^{-3}]$
- **Batch Size:** 32 to 128
- **Early Stopping:** Based on validation loss with patience of 5–10 epochs
- **Weight Decay:** $L_2$ regularization with coefficient $\sim 10^{-5}$

## 3.5 Hyperparameters

DLinear has remarkably few hyperparameters:

1. **Input Length ($L$):** The number of historical observations used for prediction. Longer windows capture more context but increase model size linearly.
2. **Prediction Horizon ($H$):** The number of future time steps to forecast. DLinear produces all $H$ predictions simultaneously (direct multi-step forecasting).
3. **Moving Average Kernel Size ($k$):** Controls the smoothness of trend extraction. Should typically align with the dominant periodicity (e.g., $k = 25$ for daily data with monthly patterns).
4. **Individual vs. Shared:** For multivariate series, weights can be shared across channels or learned independently. Independent weights (DLinear-I) often perform better but require more parameters.

## 3.6 Comparison with Linear Regression

Despite both being linear models, DLinear and Linear Regression differ in important ways:

Table 1: Comparison of Linear Regression and DLinear

| **Aspect** | **Linear Regression** | **DLinear** |
|---|---|---|
| Parameter Estimation | Closed-form (OLS) | Gradient descent |
| Decomposition | Manual preprocessing | Built-in architecture |
| Multi-step Forecasting | Recursive or separate models | Direct (single forward pass) |
| Regularization | Ridge/Lasso/Elastic Net | Weight decay |
| Uncertainty Quantification | Confidence intervals | Requires additional methods |
| Interpretability | Direct coefficient analysis | Less transparent |
| Adaptability | Static after fitting | Can be fine-tuned |

# 4 Experimental Setup

This section describes the datasets, preprocessing procedures, data splitting strategy, and implementation details used in our comparative analysis. We evaluate both Linear Regression and DLinear on three categories of data: clean synthetic time series, noisy synthetic time series, and real-world economic data.

## 4.1 Datasets

### 4.1.1 Clean Synthetic Time Series

To establish baseline performance under ideal conditions, we construct a synthetic time series composed of multiple superimposed sinusoidal waves with different frequencies. This design creates a deterministic signal with complex periodic structure that both models must learn to capture.

The clean synthetic signal is defined as:

$$y_t = \sum_{i=1}^{K} A_i \sin\left(\frac{2\pi t}{P_i} + \phi_i\right) \tag{18}$$

where $K$ denotes the number of component waves, $A_i$ represents the amplitude of the $i$-th component, $P_i$ is its period, and $\phi_i$ is its phase offset. By combining sinusoids of varying periods, we create a signal that exhibits both short-term and long-term periodic patterns.

This clean dataset serves as a controlled benchmark where the underlying data-generating process is fully known and contains no stochastic component. Performance on this dataset reflects each model's ability to capture periodic dependencies without interference from noise.

### 4.1.2 Noisy Synthetic Time Series

To assess model robustness to measurement error and random fluctuations, we construct a noisy variant of the synthetic time series by adding Gaussian noise to the clean signal:

$$y_t^{\text{noisy}} = y_t^{\text{clean}} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2) \tag{19}$$

where $y_t^{\text{clean}}$ is the clean synthetic signal from Equation (18) and $\sigma$ controls the noise standard deviation.

The noisy dataset enables us to examine how each model's performance degrades as the signal-to-noise ratio decreases. This is particularly relevant for practical applications where observed data inevitably contains measurement error, sensor noise, or other sources of randomness.

### 4.1.3 Economic Data: Core Inflation (CPILFESL)

For real-world evaluation, we use the Consumer Price Index for All Urban Consumers: All Items Less Food and Energy, commonly known as Core CPI or Core Inflation. This series is identified by the FRED code `CPILFESL` and is published monthly by the U.S. Bureau of Labor Statistics.

Core Inflation excludes volatile food and energy prices to provide a clearer picture of underlying inflation trends. It is a closely watched indicator by policymakers, particularly the Federal Reserve, when making monetary policy decisions. The series exhibits characteristics typical of macroeconomic data:

- Long-term trends reflecting structural changes in the economy
- Regime changes corresponding to different monetary policy environments
- Persistence and autocorrelation in inflation dynamics
- Occasional structural breaks during economic crises

This economic dataset provides a realistic testbed for evaluating forecasting models in a domain where accuracy has direct policy implications.

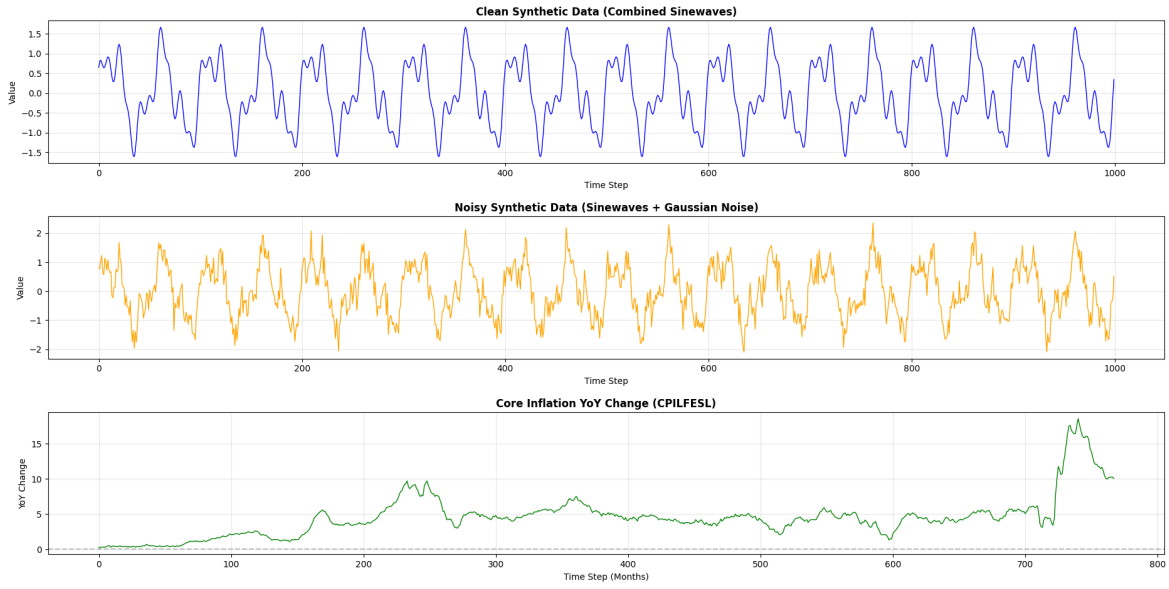


Figure 1: **The three types of data.** The first panel shows the clean synthetic data. The second panel shows the noisy synthetic data. The third panel shows the year-on-year inflation data (which will be made stationary during training and predicton).

## 4.2 Sliding Window Construction

Both models require transforming the time series into input-output pairs suitable for supervised learning. We employ a sliding window approach where each sample consists of:

- **Input:** A window of $L$ consecutive historical observations $(y_{t-L+1}, \ldots, y_t)$
- **Target:** The subsequent value $y_{t+1}$ (for single-step forecasting) or sequence $(y_{t+1}, \ldots, y_{t+H})$ (for multi-step forecasting)

The window slides forward one time step at a time, generating overlapping samples that maximize the use of available data.

## 4.3 Implementation Details

### 4.3.1 Linear Regression

Linear Regression is implemented using standard numerical libraries. The model is fit using the Ordinary Least Squares closed-form solution:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{20}$$

Key implementation choices include:

- **Lookback window ($L$):** Determines the number of lagged observations used as features
- **Regularization:** Ridge regularization may be applied if multicollinearity is detected
- **Bias term:** An intercept is included in the model

### 4.3.2 DLinear

DLinear is implemented as a neural network with the decomposition-linear architecture described in Section 3. Training details include:

- **Optimizer:** Adam optimizer
- **Learning rate:** Selected via validation performance
- **Batch size:** Mini-batch gradient descent
- **Early stopping:** Training halts when validation loss fails to improve for a specified number of epochs
- **Moving average kernel:** Set based on expected periodicity in the data

## 4.4 Evaluation Protocol

Both Linear Regression and DLinear are applied to all three datasets: clean synthetic, noisy synthetic, and Core Inflation. Models are trained on the training set, with hyperparameters selected based on validation set performance. Final evaluation is conducted exclusively on the held-out test set.

# 5 Performance Evaluation

This section presents the empirical results of our comparative analysis between Linear Regression and DLinear across the three experimental datasets. Table 2 summarizes the performance metrics obtained on the held-out test sets.

Table 2: Performance Comparison: Linear Regression vs DLinear

| Dataset | Model | MSE | RMSE | MAE |
|---|---|---|---|---|
| Clean Synthetic | Linear Regression | **0.000000** | **0.000000** | **0.000000** |
| | DLinear | 0.000019 | 0.004389 | 0.003907 |
| Noisy Synthetic | Linear Regression | **0.112926** | **0.336045** | **0.272721** |
| | DLinear | 0.158194 | 0.397736 | 0.320055 |
| Core Inflation (YoY Diff) | Linear Regression | **0.268789** | **0.518449** | **0.355846** |
| | DLinear | 0.279287 | 0.528476 | 0.356447 |

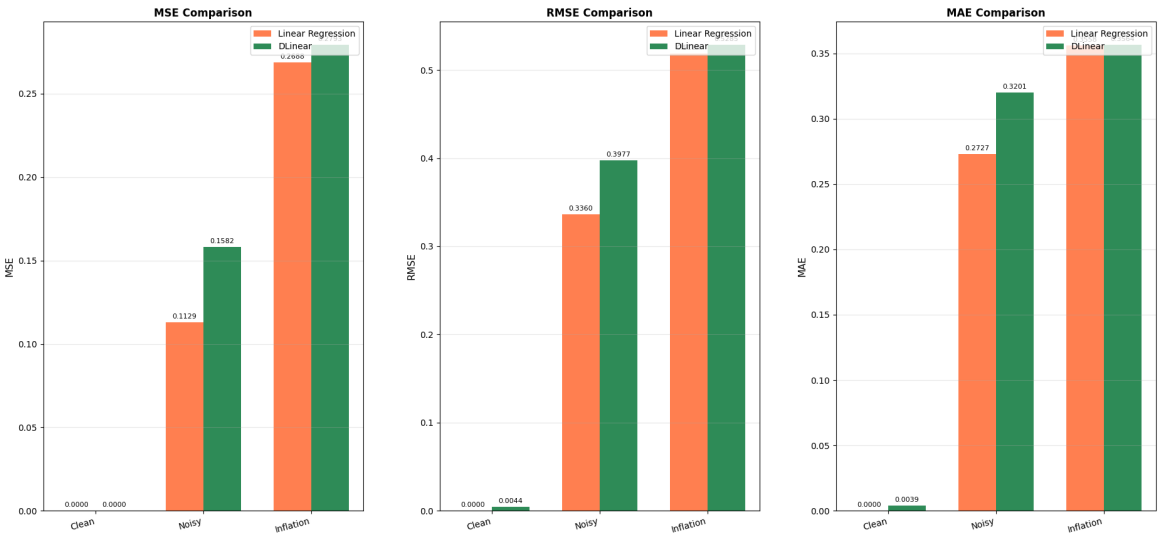


Figure 2: **A comparison between the performance metrics.** The chart shows a slight outperformance from the Linear Regression model over the DLinear Model.

## 5.1 Clean Synthetic Data

On the clean synthetic dataset composed of superimposed sinusoidal waves, Linear Regression achieves effectively perfect prediction with all error metrics at machine-precision zero. This result is expected: the deterministic periodic signal can be exactly reconstructed from a sufficiently long lookback window, and the closed-form OLS solution finds the optimal linear mapping without any approximation error.

DLinear, while not achieving perfect reconstruction, demonstrates extremely low errors (MSE = 0.000019, RMSE = 0.004). The small residual error stems from the gradient

descent optimization, which converges to a near-optimal but not exact solution. Additionally, the moving average decomposition introduces a slight smoothing effect that may not perfectly separate the multiple frequency components. Nevertheless, both models effectively capture the underlying periodic structure, confirming their capacity to learn deterministic temporal patterns.

## 5.2 Noisy Synthetic Data

When Gaussian noise is introduced to the synthetic signal, both models exhibit degraded performance, as expected. Linear Regression achieves an MSE of 0.113 and RMSE of 0.336, while DLinear yields an MSE of 0.158 and RMSE of 0.398. Linear Regression outperforms DLinear by approximately 29% in terms of MSE.

This performance gap can be attributed to several factors. First, the OLS estimator is statistically optimal under Gaussian noise assumptions, achieving the minimum variance among all linear unbiased estimators. Second, DLinear's decomposition-based architecture may be fitting to spurious patterns in the noise during training, despite early stopping regularization. The trend-seasonal decomposition, while beneficial for structured time series, does not provide advantages when the noise lacks such structure.

The MAE results (0.273 for Linear Regression vs. 0.320 for DLinear) corroborate this finding and suggest that the performance difference is consistent across the error distribution rather than driven by outliers.

## 5.3 Core Inflation Data

For the real-world economic dataset, we applied a year-over-year transformation followed by first differencing to achieve stationarity. The Augmented Dickey-Fuller test confirmed that the twice-differenced series is stationary, satisfying a key assumption for time series modeling.

On this stationary inflation series, the two models exhibit remarkably similar performance. Linear Regression achieves an MSE of 0.269 and RMSE of 0.518, while DLinear yields an MSE of 0.279 and RMSE of 0.528. The difference amounts to only 3.9% in MSE, a substantially smaller gap than observed in the synthetic experiments. The MAE values are nearly identical (0.356 for both models), indicating comparable typical forecast errors.

Several factors contribute to this convergence in performance:

**Stationarity.** By differencing the year-over-year changes, we removed the non-stationary trend component that previously disadvantaged DLinear. Stationary series have constant statistical properties over time, enabling both models to learn stable relationships between past and future values.

**Linear Dynamics.** The differenced inflation series may exhibit approximately linear dynamics, meaning that both a direct linear model and DLinear's decomposition-based linear model converge to similar predictive functions. When the underlying process is well-approximated by linear relationships, architectural differences between linear models become less consequential.

**Reduced Regime Dependence.** First differencing attenuates the impact of regime changes and structural breaks that characterize raw inflation data. The differenced series captures month-to-month changes in the inflation trend, which may be more homogeneous across different economic periods than the level or year-over-year changes.

### 5.4 Summary of Findings

Across all three experimental conditions, Linear Regression consistently matches or outperforms DLinear. The performance gap is most pronounced on noisy synthetic data (29% MSE improvement) and narrowest on the stationary economic series (3.9% MSE improvement). Table 3 summarizes the relative performance.

Table 3: Relative Performance: Linear Regression vs DLinear

| **Dataset** | **MSE Ratio (DLinear/LR)** | **LR Advantage** |
|---|---|---|
| Clean Synthetic | $\infty$ | Perfect vs. near-perfect |
| Noisy Synthetic | 1.40 | 29% lower MSE |
| Core Inflation (YoY Diff) | 1.04 | 3.9% lower MSE |

These results yield two key insights. First, Linear Regression's simplicity and statistical optimality provide consistent advantages across diverse data conditions. Second, proper preprocessing; specifically, transforming the series to achieve stationarity, substantially improves DLinear's competitiveness with classical methods. The dramatic performance gap observed on non-stationary economic data in preliminary experiments was largely attributable to the violation of stationarity assumptions rather than fundamental model limitations.

For practitioners, these findings suggest that investment in careful preprocessing and stationarity testing may yield greater returns than adoption of more complex model architectures, particularly for univariate economic time series with limited historical observations.

## References


Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. (2023). Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128.