Improving Channel-Independent Transformer via Statistical Preprocessing for Time Series Forecasting

Keywords: Statistical Learning, Data Preprocessing, Time Series, Forecasting, Channel-Independence, Transformers, Machine Learning

Abstract: Effective data preprocessing is crucial in time series forecasting, as it enhances model performance by addressing data inconsistencies and scaling issues. Traditional forecasting models often rely on simple scaling techniques like StandardScaler to normalize data, but these methods are sensitive to outliers and implicitly assume a normal distribution, which can limit their effectiveness in real-world data. In this work, we propose an innovative approach that integrates multiple statistical preprocessing techniques, including logarithmic, Box-Cox, Yeo-Johnson, square root, and differencing, directly into the state-of-the-art model PatchTST (channel-independent patch time series Transformer) for time series forecasting. To our knowledge, this is the first time such comprehensive data preprocessing techniques have been added to a transformer model. By addressing issues like skewness, non-stationarity, and heteroscedasticity, our enhanced PatchTST achieves notable improvements in forecast accuracy. Experiments reveal substantial reductions in errors by 38% for the single target variable and 24% for all variables, underscoring the potential of statistical preprocessing in transformers.

1 INTRODUCTION

Time series forecasting is fundamental in many application domains, including finance, climate, epidemiology, traffic, and bioinformatics (Mudelsee, 2019; Andersen et al., 2005; Dastjerdi et al., 2022). The ability to accurately model temporal patterns has direct implications for operational efficiency and strategic decision-making in industries. Recent advances in deep learning have led to the adoption of transformer-based architectures, which leverage self-attention mechanisms to model long-range dependencies and capture complex temporal behaviors (Nie et al., 2022). Unlike traditional autoregressive models or recurrent neural networks, transformers excel at processing sequential data in parallel while maintaining sensitivity to temporal context. However, their performance is heavily influenced by the statistical properties of the input data, which are often non-ideal in real-world scenarios.

Real-world temporal data, such as sensor measurements, economic indicators, or disaster records, frequently exhibit the following.

1. **Non-Stationarity**: Time-varying statistical properties (e.g., mean, variance) obscure temporal patterns and degrade the generalizability of the model (Hyndman, 2018).
2. **Skewness**: Asymmetric distributions bias model training towards dominant trends, suppressing rare but critical events (Box and Cox, 1964).
3. **Heteroscedasticity**: The instability of the variation over time complicates the optimization as the models struggle to balance the error contributions between scales (Engle, 1982).

Due to these fundamental challenges in deep learning models, time series data often face a myriad of issues, like distribution shifts between training and inference phases, which cause models to learn spurious correlations rather than true causal relationships. Varying scales and variances create irregular loss surfaces, leading to unstable gradient updates and suboptimal convergence (Goodfellow, 2016). Moreover, models trained on normalized historical data often fail to generalize to future methods with differing statistical characteristics (Quiñonero-Candela et al., 2022).

To address these challenges, preprocessing is important for aligning the raw data with the assumptions of the model. Simple normalization techniques such as StandardScaler (zero mean, unit variance) are widely adopted in deep learning (Pedregosa et al., 2011) due to their computational efficiency and compatibility with gradient-based optimization. However, these methods fall short when it comes to handling the complex statistical challenges inherent in time series data. They are highly sensitive to outliers because they do not adjust for extreme values and assume that the data are normally distributed, an assumption that does not hold for skewed datasets. This limitation mo-

tivates the use of advanced transformations rooted in statistical theory.

Traditional statistical preprocessing techniques explicitly target non-stationarity and heteroscedasticity. Some examples of these methods that have been used widely in current state-of-the-art (SOTA) literature are the Box-Cox Transformation (Box and Cox, 1964); a powerful transformation that stabilizes variance and induces normality in positively skewed data, Yeo-Johnson Transformation (Yeo and Johnson, 2000); which extends Box-Cox to handle zero and negative values, critical for domains like energy demand forecasting (Bergmeir et al., 2018), Seasonal differencing to remove periodic trends to achieve stationarity. Statistical models such as Seasonal AutoRegressive Integrated Moving Average (SARIMA) leverage techniques like logarithmic transformations, differencing, and seasonal differencing to enhance forecast accuracy by stabilizing variance, removing trends, and accounting for seasonal patterns. As a result, these models tend to outperform ARIMA and ARMA approaches. These transformation techniques serve as foundational tools in traditional forecasting frameworks, including SARIMA, ARIMA (Box et al., 2015) and Exponential Smoothing (ETS) (Hyndman and Khandakar, 2008). Numerous studies have demonstrated the effectiveness of these methods in time series forecasting (Hyndman, 2018; Salles et al., 2019).

Despite their clear benefits, these preprocessing techniques are rarely integrated into modern deep learning frameworks. Instead, researchers typically rely on basic normalization methods (e.g., StandardScaler or MinMaxScaler) that do not fully address issues particular to statistical challenges in time series data. Recent work by (Bandara et al., 2020) shows that the integration of Box-Cox transformations into neural architectures significantly improves the accuracy of the forecast for non-stationary retail sales data.

We theorize that integrating domain-agnostic statistical preprocessing with modern transformer architectures could improve forecast performance while maintaining model generality. This work extends channel-independent transformer model through preprocessing by incorporating variance-stabilizing transformations (Log1p, Box-Cox, Yeo-Johnson), nonlinear scaling for heavy-tailed distributions (square root transformations) and temporal differencing for trend/seasonality removal. We are experimenting with PatchTST (Nie et al., 2022) because it is considered SOTA, and new models consistently treat PatchTST as their benchmark for comparison.

Our experimental framework contributions are in threefold:

1. **Enhanced Transformers:** We introduce a pioneering approach that integrates statistical transformations such as variance stabilization, skewness correction, and stationarity enhancement with the SOTA PatchTST transformer architecture. To our knowledge, this is the first study to combine these preprocessing techniques with a transformer model. Our experiments demonstrate significant improvements in forecasting accuracy, with error scores reduced by 38% for the single variable forecast and 24% when forecasting all variables.
2. **Optimized Data Preparation:** By applying techniques such as logarithmic, Box-Cox, and Yeo-Johnson transformations along with differencing methods, the framework tackles common challenges in time series data such as heteroscedasticity, skewed distributions, and non-stationarity. Despite their proven effectiveness in traditional statistical analysis, such methods have been largely overlooked in deep learning. This robust preprocessing improves the intrinsic statistical properties of the data.
3. **Simple yet powerful:** Our findings show that statistical preprocessing of the input data can serve as a simple yet powerful alternative to complex architectural modifications, such as those used in FEDformer (Zhou et al., 2021) and Autoformer (Wu et al., 2021) models.

2 RELATED WORK

This section reviews the literature on data preprocessing techniques, covering both statistical transformations and conventional data preprocessing approaches in machine learning models.

2.1 Statistical Transformations

For the model to learn meaningful patterns without being misled by variance instability of distributional shift, transforming time series data is crucial. The study by (Salles et al., 2019) talks about different transformation methods, e.g., Logarithmic Transform, Box-Cox Transform, Differencing, etc., applied to the ARIMA model. The results highlight the importance of selecting the appropriate transformation method based on the dataset's characteristics. Experiments show that although the transformation methods consistently improved prediction and stationarity, no single method was universally best.

Logarithmic transformations play an important role in the field of economic forecasting. The authors of the study (Lütkepohl and Xu, 2012), demonstrate that these transformations stabilize variance, improving ARIMA-based forecasts. Similarly, (Proietti and Luetkepohl, 2011) provides evidence that Box-Cox transformations enhance predictive performance in macroeconomic datasets. Nevertheless, these studies also warn against inappropriate usage, as transformations can introduce distortions when applied to already stable datasets. In addition, Douglas Curran-Everett (Curran-Everett, 2018) highlights the importance of validating transformations using the Box-Cox method.

2.2 Data Preprocessing in Deep Learning

Traditional time series models such as SARIMA and ETS have long utilized transformations like differencing to remove trends and achieve stationarity, which is also shown by(Perone, 2022). However, deep learning models often overlook such preprocessing steps, relying on simple normalization techniques such as StandardScaler or MinMaxScaler. The studies by (de Amorim et al., 2023), (Raju et al., 2020), and (Ahsan et al., 2021) highlight the uses of these normalization techniques and display improved results in their respective domains.

The study by (Ozsahin et al., 2022) has thoroughly investigated the effects of normalization, standardization, and no scaling in five machine learning models, namely K-Nearest Neighbors (K-NN), Bernoulli Naive Bayes (BNB), Decision Tree(DT), Logistic Regression(LR), and Support Vector Machine(SVM) for diabetes diagnosis. The findings reveal that the choice of scaling model has a substantial effect on model accuracy, which indeed emphasizes the need for carefully selecting the preprocessing techniques in medical diagnostics. Recent SOTA transformer models like (Nie et al., 2022) and (Zhou et al., 2021) also use StandardScaler for preprocessing.

A recent study by (Bandara et al., 2020) demonstrates that **incorporating Box-Cox transformations** into **Neural Networks** substantially enhances forecasting accuracy for non-stationary retail sales data. Our work proposes an approach that investigates multiple transformations to enhance the channel-independent PatchTST (Nie et al., 2022) across the Influenza dataset.

3 TRANSFORMATIONS

The preprocessing methods utilized in this paper are outlined below.

3.1 StandardScaler

StandardScaler is one of the most widely used scaling techniques in machine learning. It transforms each feature by subtracting its mean and dividing by its standard deviation, effectively centering the data around zero with a unit variance. In mathematical terms, for each value y, the transformation is given by:

$$y_{scaled} = \frac{y - \mu}{\sigma} \tag{1}$$

where y is the original value, μ is the mean, and σ is the standard deviation of the feature. The result, y_{scaled}, is the standardized value. The mean and standard deviation are obtained from only the training set instead of the entire dataset to avoid data leakage during training (Kuhn and Johnson, 2019).

However, this method assumes that the data follows a roughly normal (Gaussian) distribution. If the data deviates from normality or contains significant outliers, standardization might not produce reliable results (Pedregosa et al., 2011). Inverse scaling is obtained by adding the mean back and multiplying by standard deviation.

3.2 Log1p

The $log1p$ function adds one to each value and then takes the natural logarithm of the result. In other words, for a given number y,

$$log1p(y) = ln(1+y) \tag{2}$$

This approach is beneficial for values close to zero because adding one helps avoid numerical problems that can arise when directly applying the natural logarithm to very small or negative numbers. If you pass in real (i.e., non-complex) numbers to the function, it always tries to return a real result. However, if $(1+y)$ is not a real number or infinity, $log1p$ outputs NaN (not a number) (Harris et al., 2020).

For the inverse transformation of $log1p$, we use $expm1$, which calculates $e^y - 1$. This function reverses the effect of $log1p$, meaning that if you apply $log1p$ to a number and then use $expm1$ on the result, you get the original number.

3.3 Square Root

Another way to handle skewed data is the power transformation, Square Root, where you simply take

the square root of the variable y (Miller, 1976). Like the logarithmic transformation, this also helps shrink large values and makes the distribution more symmetric. To reverse the transformation, you take the square of the transformed values to get the original scale. Unlike log transformation, square root transformation can handle zeros but not negative values.

3.4 Box-Cox

The Box-Cox transformation, a combination of both logarithm and power transformation, was proposed by George Box and David Cox (Box and Cox, 1964). It is used to stabilize variance, reduce skewness in the data, and achieve normality when desired. The Box-Cox transformation is defined as:

$$y_{\text{trans}} = \begin{cases} \dfrac{y^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0, \\ \log y & \text{if } \lambda = 0. \end{cases} \tag{3}$$

where y is the input variable and λ is the transformation parameter. An optimal parameter λ is learned from the input data, which determines the specific power function applied to y; for example, λ = 1 gives the original value, λ = -1 gives the reciprocal transformation and λ = 2 corresponds to a square transformation. The log transformation is applied when λ is equal to zero. The optimal parameter λ is chosen via the maximum likelihood estimation.

The inverse of the Box-cox transformation is defined as follows (Pedregosa et al., 2011):

$$y = \begin{cases} \exp\left(y_{\text{trans}}\right) & \text{if } \lambda = 0, \\ \left(y_{\text{trans}}\lambda + 1\right)^{\frac{1}{\lambda}} & \text{if } \lambda \neq 0. \end{cases} \tag{4}$$

where y_{trans} is the transformed variable. To obtain the original data, the inverse transformation is applied using the fitted lambdas. It is important to note that the Box-Cox transformation can only be applied when all data values are positive. If your data includes negative values, you can either add a constant to shift the distribution into the positive range or opt for the Yeo-Johnson transformation, which accommodates negative values.

3.5 Yeo-Johnson

The Yeo-Johnson transformation, introduced by Yeo and Johnson (Yeo and Johnson, 2000), is an extension of the Box-Cox transformation. It is defined as:

$$y_{\text{trans}} = \begin{cases} \dfrac{(y+1)^{\lambda}-1}{\lambda} & \text{if } y \geq 0 \text{ and } \lambda \neq 0, \\ \log(y+1) & \text{if } y \geq 0 \text{ and } \lambda = 0, \\ -\dfrac{(-y+1)^{2-\lambda}-1}{2-\lambda} & \text{if } y < 0 \text{ and } \lambda \neq 2, \\ -\log(-y+1) & \text{if } y < 0, \lambda = 2. \end{cases} \tag{5}$$

Here, y is the input variable and λ is the transformation parameter. This transformation can be applied without any restrictions on y, which means that it is designed for variables that include zeros, negatives, and positive values. For positive values, the Yeo-Johnson transformation functions similarly to the Box-Cox transformation but is applied to $y+1$. This shift ensures that the transformation is well-defined (since logarithms require positive arguments) and helps in handling zeros. When the values are strictly negative, it is also equivalent to Box-Cox but this time it is applied to $-y+1$ with a power of $2-\lambda$. If the data has positive and negative values, it combines the two approaches by using different powers for the positive and negative parts of the data (Weisberg, 2001).

3.6 Differencing

Transformations like logarithms can help make the variance of a time series more stable. Differencing, on the other hand, helps stabilize the mean by removing shifts in the data over time. This can reduce or even eliminate trends and seasonal patterns, making the non-stationary time series stationary (Hyndman, 2018). In this paper, we explore two different types of differencing techniques: First, and Seasonal. Algorithm 1 shows the differencing function.

3.6.1 First Differencing

For First differencing, compute the difference between consecutive time steps by subtracting each previous time step from the current one (Hyndman, 2018). For reverse differencing, combine the initial seed value from the original input with the predicted differences and perform a cumulative sum to rebuild the original series, dropping the seed afterward.

3.6.2 Seasonal Differencing

For Seasonal differencing, subtract each data point from the data point that is a fixed number of time steps (defined by *seasonal* in the algorithm) behind it (Hyndman, 2018). For reverse differencing, combine the *seasonal* seed values with the predictions

Input: $data$: Tensor with dimensions $[batch\_size, sequence\_length, features]$
$diff order$: A string indicating the differencing order ("First", "Seasonal")
$seasonal$: (Optional) An integer for the seasonal lag (used if $diff order$ is "Seasonal")
Output: $differenced\_data$: Transformed data with differencing applied
Function
`DIFFERENCING`($data$, $diff order$, $seasonal$) **:**
  if $diff order$ *equals "First"* **then**
    $differenced\_data \leftarrow data[:, 1:\text{end}, :] - data[:, 0:\text{end}-1, :]$
  else if $diff order$ *equals "Seasonal"* **then**
    $differenced\_data \leftarrow data[:, seasonal:\text{end}, :] - data[:, 0:\text{end}-seasonal, :]$
  end
  return $differenced\_data$
Algorithm 1: Differencing Function

and add back the value from the corresponding previous *season* at each step, finally removing the *seasonal* seeds to reconstruct the original series.

4 EXPERIMENTS

4.1 Data and Code

Influenza-like illness (ILI)[1] dataset is collected from the Centers for Disease Control and Prevention (CDC) from 1^{st} October 2002 to 30^{th} June 2020, has the ILI patients data recorded every week for the United States. The data[2] consists of 966 samples and seven variables such as weighted and unweighted ILI cases, data by age group, number of providers, and total number of affected patients. The updated PatchTST code, which incorporates the above mentioned preprocessing techniques, is available on GitHub[3].

4.2 Evaluation

To evaluate the model, we use two main metrics: the symmetric mean absolute percentage error (sMAPE), and the mean absolute error (MAE). sMAPE measures the absolute difference between the actual and forecasted values normalized by absolute values of both (Miller, 1976). The sMAPE is a bounded metric where 0% indicates that the forecast is perfect, with no discrepancies between predicted and actual values, whereas a value of 200% implies that the predictions are completely off, with the forecasted values having opposite signs to the true values (Chicco et al., 2021). The sMAPE is calculated using equation 6

$$\text{sMAPE} = \frac{200}{n} \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|} \tag{6}$$

where y_t is the true value, and $\hat{y}_t$ is the forecasted value.

MAE measures the mean of the absolute differences between the predicted values and the actual values. The MAE is an unbounded metric, and it is calculated using equation 7

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^{n} |y_t - \hat{y}_t| \tag{7}$$

where y_t is the true value, and $\hat{y}_t$ is the forecasted value.

4.3 Implementation Details

4.3.1 PatchTST

We selected the SOTA channel-independent transformer model, PatchTST (Nie et al., 2022), and implemented it using the original code provided in the paper to replicate the reported results. The code is available on their GitHub[4], and we used the exact hyperparameters specified in the Illness script[5]. For the ILI dataset, we adopted prediction lengths {24, 36, 48, 60}, as in the original study, and additionally experimented with prediction lengths of 6 and 12. We obtained results that closely align with those reported in the paper (details in Section 5) and applied the StandardScaler $inverse\_transform$ method from the Scikit-Learn library to revert the transformations, ensuring an accurate comparison with the statistical transformation. Please note that the results reported in PatchTST (Nie et al., 2022) were generated by dropping the last test samples that did not meet the batch size (i.e., with $drop\_last$ set to True). In contrast, while reproducing these results under the same experimental conditions, we set $drop\_last$ to False and present our results.

[1] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

[2] https://github.com/scalation/data/blob/master/Influenza/national_illness.csv

[3] https://github.com/scalation/scalation_py/tree/dev_sr/Transformations

[4] https://github.com/yuqinie98/PatchTST/tree/main/PatchTST_supervised

[5] https://github.com/yuqinie98/PatchTST/blob/main/PatchTST_supervised/scripts/PatchTST/illness.sh

The NumPy library is used to implement transformations such as Log1p, Square Root, and Differencing, whereas the Box-Cox and Yeo-Johnson transformations are carried out using Scikit-Learn's PowerTransformer. This PowerTransformer leverages Maximum Likelihood Estimation (MLE) as its estimator and utilizes Brent's method for optimization. The seven lambda (λ) values for seven ILI features are [-0.28602651, -0.503773, 0.29751579, 0.14257209, 0.19154931, 1.18640383, 0.8973477] using Box-Cox and [-1.05905535, -1.35406619, 0.29698175, 0.14196202, 0.1913529, 1.18688668, 0.89734852] using Yeo-Johnson. The seasonal differencing function is adapted from (Nau, 2016; Hyndman, 2018). The ILI dataset exhibits a yearly pattern, we experimented with 13, 26, and 52 samples, and 26 as a seasonal parameter outperformed the others. In our study, we computed the sMAPE exclusively to evaluate the performance of the single target variable *ILITOTAL*, while the MAE was used to assess the overall performance of all seven variables as in the PatchTST paper.

We conducted a brief experiment using the ETTh1 dataset. Since the dataset contains negative values, we were limited to applying only the Yeo-Johnson transformation. For further details, please refer to Section 5.

4.3.2 SARIMA

We adopted the SARIMA (Box et al., 2015) model as our baseline. The optimized hyperparameters $(p,d,q)X(P,D,Q)_m$ as $(3,1,3)X(3,0,1)_{13}$ are derived from (Rana et al., 2024). When applying other transformations, we set the differencing parameter (d) to zero. For first differencing, we assign $d = 1$. For seasonal differencing, we initialize $D = 1$. All other implementation details remain the same as in PatchTST.

4.4 Results Analysis

Table 1 presents the performance results for various data transformation techniques. The table reports the MAEs for all variables collectively, while the sMAPE is provided for the target variable *ILITOTAL*.

PatchTST leads SARIMA in most of the techniques for a single target variable. Among the tested methods, log1p, Box-Cox, Yeo-Johnson, and square root transformations outperform the baseline StandardScaler for the single target variable (*ILITOTAL*).

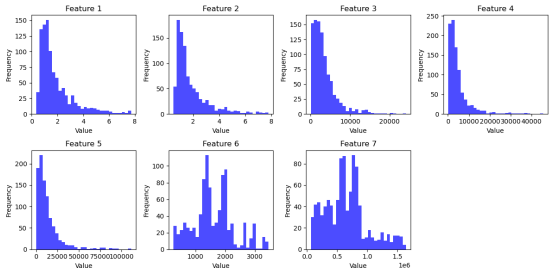
Logarithmic transformations are widely recognized for reducing right skewness in datasets (Galli, 2024; Feng et al., 2013; Yeo and Johnson, 2000). They mitigate the impact of outliers by compressing the scale of large values, thereby making the distribution more symmetric. As shown in the histograms in Figure 1, most features, including Feature 5 (*ILITOTAL*), are right-skewed. Applying the log1p, Box-Cox, Yeo-Johnson, and square root transformations adjusts these right-skewed distributions, bringing them closer to a normal distribution.

It is important to note that logarithmic transformations do not always reduce skewness; in cases where the data is already normally distributed, they may even introduce additional skewness (Feng et al., 2014). Note that for features 6 and 7, which were not right skewed originally, the transformation did not improve the distribution. However, for the other features, particularly for *ILITOTAL* (Feature 5), all four transformations clearly reduce skewness, as illustrated in the figures. Box-Cox and Yeo-Johnson transformations made the features appear more normally distributed by automatically selecting the optimal power (lambda) parameter.

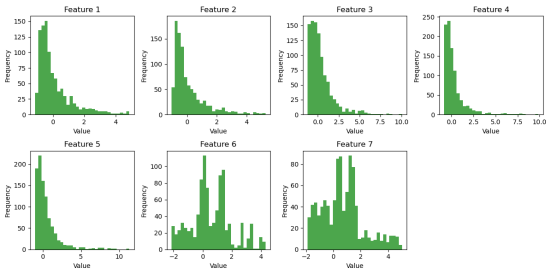
In contrast, StandardScaler assumes that the data follows a normal distribution and only scales the data, without altering its shape. As demonstrated in Figure 1b, although the data is rescaled, the original distribution remains unchanged. This behavior may explain why StandardScaler performs poorly when the underlying data deviates from normality and fails to mitigate the impact of outliers.

Figures 2a and 2b present the ground truth alongside 24-day-ahead forecasts for the target variable *ILITOTAL* using StandardScaler and log1p transformations, respectively. The log1p transformation effectively captures the peaks and troughs in the data, whereas StandardScaler does not represent these fluctuations as accurately.

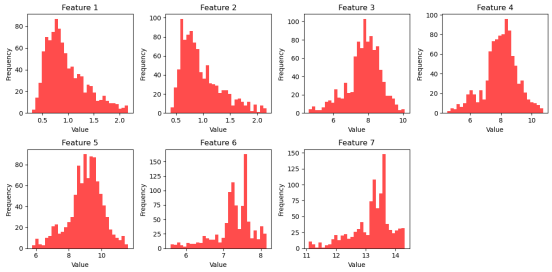
Finally, among the power and log transformations, the best results for *ILITOTAL* are achieved using the log1p and square root transformations. Among these, logarithmic transformations are more interpretable because changes in the logarithmic scale directly reflect relative (or percentage) changes in the original data (Hyndman, 2018). The primary objective of these transformations is to make the variations in the time series more homogeneous. Our dataset exhibits fluctuating magnitudes—small at certain points and larger at others. This means that the data shows structural breaks, which lead to differing mean levels across segments and considerable variance. These structural breaks can be identified by comparing the means of different segments (Salles et al., 2019). We aim to stabilize the data across its entire range by applying these transformations. Figures 3a and 3b illustrate that the high fluctuations are notably diminished in the logarithmically transformed series compared to the square root method. The log1p approach effec-



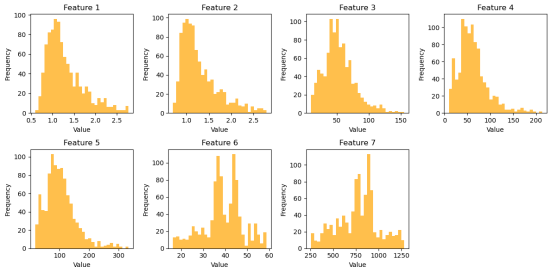
(a) Without any transformation



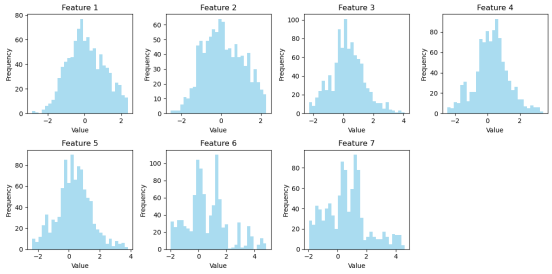
(b) StandardScaler



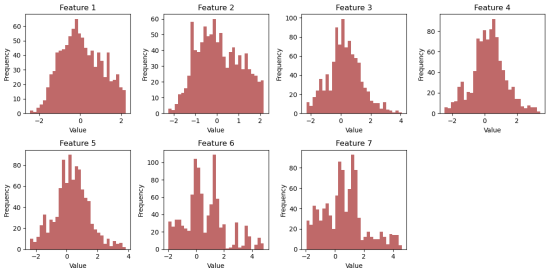
(c) Log1p



(d) Square Root



(e) Box-Cox

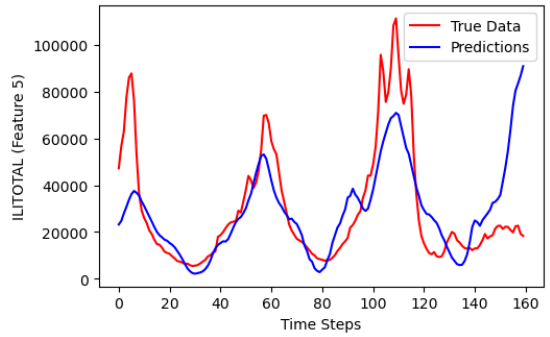


(f) Yeo-Johnson

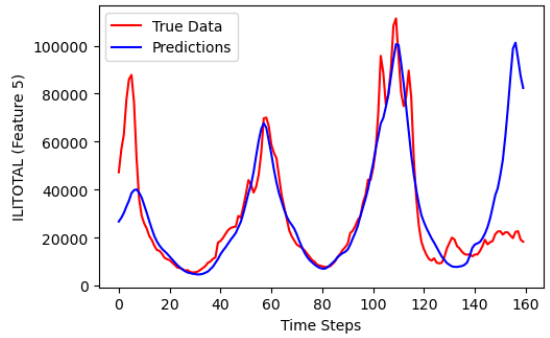
Figure 1: Histograms illustrating the effects of different data transformation techniques on the ILI dataset of 7 features. These plots reveal how each transformation alters the distribution, affecting skewness and dispersion.

Table 1: The sMAPE and MAE results for the transformations using the ILI weekly dataset. Among SARIMA and PatchTST, the best results are highlighted in red. For the comparison of transformations within PatchTST, the best results are shown in bold, while the second-best results are underlined. Here, Log1p + First Diff. refers to the combination of Log1p transformation with first differencing, whereas Log1p + S.Diff. represents the combination of Log1p transformation with seasonal differencing.

	StandardScaler		**Log1p**		**Square Root**		**Box-Cox**		**Yeo-Johnson**		**First Diff.**		**Seasonal Diff.**		**Log1p + First Diff.**		**Log1p+S.Diff**	
Week	SARIMA	PatchTST	SARIMA	PatchTST	SARIMA	PatchTST	SARIMA	PatchTST	SARIMA	PatchTST	SARIMA	PatchTST	SARIMA	PatchTST	SARIMA	PatchTST	SARIMA	PatchTST
	sMAPEs - *ILITOTAL*																	
6	31.92	35.40	22.63	23.84	23.54	27.06	24.27	23.56	23.97	23.77	24.63	34.37	26.56	51.69	19.00	17.25	19.89	26.17
12	48.81	49.40	37.52	31.28	39.96	31.87	39.50	35.06	38.58	35.28	38.14	46.20	38.50	57.95	27.69	25.94	30.29	30.53
24	65.14	46.71	54.94	31.00	59.39	29.72	57.60	36.08	55.64	35.22	49.45	58.26	46.36	54.02	34.37	34.63	39.01	31.34
36	61.92	47.97	61.11	31.06	66.32	30.14	63.00	32.51	60.94	31.57	51.81	67.72	48.63	52.78	36.38	33.64	41.15	32.23
48	59.54	58.11	59.83	24.32	67.83	28.20	63.01	29.40	61.38	30.43	52.71	69.51	49.86	56.94	37.53	32.40	42.16	27.97
60	58.67	38.75	58.47	27.70	73.13	31.96	64.29	34.32	63.34	33.52	53.44	63.97	50.86	52.37	39.28	32.29	43.50	29.65
Avg	54.33	46.05	49.08	**28.20**	55.02	29.82	51.94	31.82	50.64	31.63	45.03	56.67	43.46	54.29	32.37	29.35	36.00	29.64
	MAEs - All variables																	
6	16844.74	16839.83	14056.24	19881.51	14454.47	15818.69	15924.42	15104.86	15876.90	15267.16	14215.94	12693.62	13801.11	16969.63	13868.02	13736.18	13682.41	19233.16
12	24082.89	22629.97	19918.28	23602.77	20729.71	20537.00	22701.11	19485.43	22622.50	20261.13	19673.40	17378.38	17470.73	21997.68	19245.43	18402.62	16861.64	22675.73
24	31469.72	29072.70	26134.05	29559.48	27453.10	25617.46	29617.99	25305.86	29498.33	26222.96	25025.35	21269.48	20724.31	22697.83	24722.42	22994.84	19925.57	28495.22
36	34538.67	29322.07	27712.96	32954.31	29260.57	27244.28	32580.43	30362.54	32436.79	30212.62	26430.90	24432.68	21964.31	20395.40	26060.21	26615.64	21164.80	25096.66
48	36247.89	33369.20	27051.30	29512.84	30020.48	30600.16	34230.99	28597.62	34090.00	30909.66	25810.38	25000.00	22623.02	20836.90	25601.89	25344.86	21788.76	24309.00
60	38046.28	31820.30	26644.67	34074.13	31415.01	26688.68	35998.75	32260.86	35871.03	32150.22	25369.91	22462.34	23278.89	19759.23	25379.16	24491.31	22315.92	23335.12
Avg	30205.03	27175.67	23586.25	28264.17	25555.55	24417.71	28508.94	25186.19	28399.25	25837.29	22754.31	20539.41	19977.06	**20442.77**	22479.52	21930.90	19289.85	23857.48



(a) StandardScaler



(b) Log1p

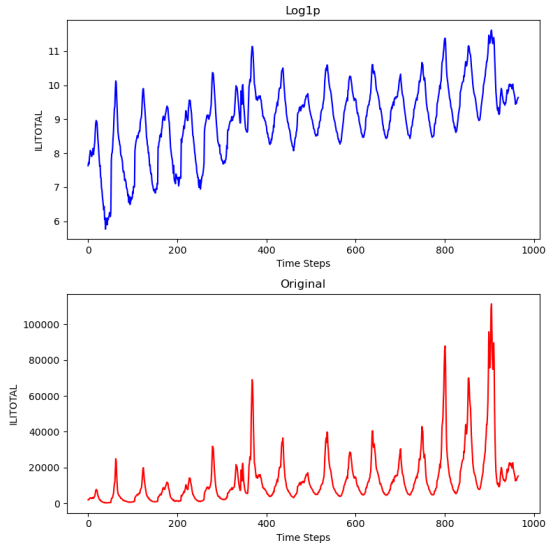
Figure 2: The above plots compare the ground truth ILITOTAL values with 24-weeks ahead forecasts obtained through transformations, capturing the data's peaks and troughs.

tively lowers the variance, while the mean becomes nearly constant at the later stage, though not entirely stabilized. Although the square root transformation moderates the fluctuations somewhat, significant variability remains. In contrast, the logarithmic transformation yields a series of more consistent fluctuations throughout which simplifies the modeling process.

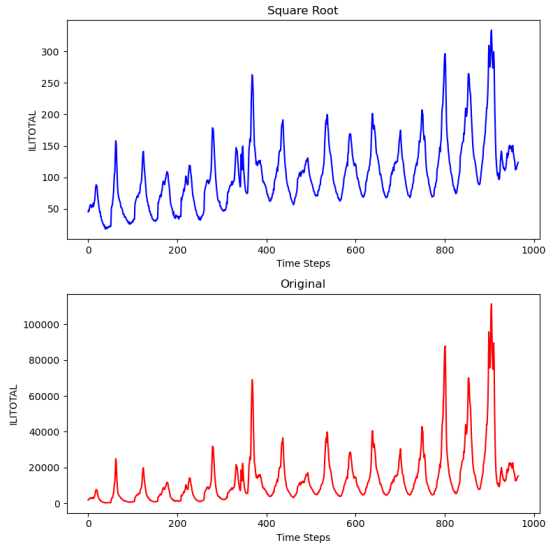
Figure 4 compares the plots of the first-differenced series with and without the log1p transformation. Although both series oscillate around zero, their variance behaviors differ significantly. In particular, the log1p-differenced series performs better than the series that has been only differenced—a finding that is also supported by (Lütkepohl and Xu, 2012; Hossain et al., 2019). The log1p-differenced *ILITOTAL* series exhibits considerable variance in the early years, which then stabilizes during the middle and later periods, as shown in the plots and confirmed using $np.var()$. In contrast, the series that underwent only differencing shows a trend of increasing variance over time. We conducted Levene's test (Virtanen et al., 2020) on both series, and according to the p-values reported in Table 2, the null hypothesis of equal variances across segments is rejected for both transformations. However, the log1p-differenced series displays reduced heteroscedasticity in the later segments ($p \approx 0.000014$), while the only differenced series exhibits even greater variance inconsistency ($p \approx 1.40 \times 10^{-15}$). Although neither method achieves perfect homoscedasticity, log1p-differencing offers superior variance stabilization, making it a more effective transformation for achieving uniform variance over time. That said, additional preprocessing may still be required to fully normalize the variance, particularly in the early data.

In our analysis of MAEs across all variables, PatchTST outperforms SARIMA on five techniques and exceeds StandardScaler on almost all techniques. The stationarity tests from Statsmodels (Seabold and Perktold, 2010) - the Augmented Dickey-Fuller (ADF) and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) - indicate that ILI features 1 and 2 are stationary, while features 6 and 7 are non-stationary and features 3, 4, and 5 are difference stationary (according to case 4[6]). The respective p-values and null hypothesis are provided in Table 2. The difference stationary or non-stationarity in a series indicates the presence of

[6]https://www.statsmodels.org/dev/examples/notebooks/generated/stationarity_detrending_adf_kpss.html



(a) Log1p vs Original



(b) Square Root vs Original

Figure 3: The above plots display the transformed ILITOTAL series alongside the original data, allowing you to compare the variations before and after the transformations.

a unit root in a series, which can be removed by differencing (Salles et al., 2019). Almost all series show a clear seasonal pattern. By applying first differencing to eliminate the unit root and seasonal differencing to remove the seasonal effects, we successfully transform the series into stationary ones (Hyndman, 2018) as shown in Table 2. As a result, the forecast performance improves significantly, resulting in the best results when these differencing methods are employed. Additionally, logarithmic and power transformations also produce better results than StandardScaler due to their ability to stabilize the series.

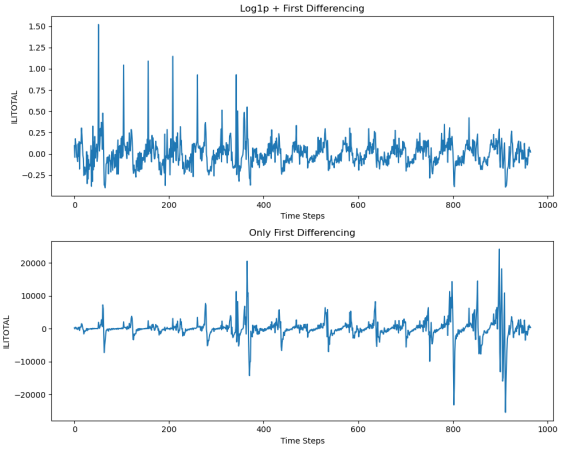


Figure 4: In the figure above, the top plot shows the ILITOTAL series after applying both a log1p transformation and first differencing, while the bottom plot shows the series after applying only first differencing.

Table 2: The table presents the results of the ADF, KPSS, and Levene's tests. For the ADF test, the null hypothesis is that the series contains a unit root, implying non-stationarity. In contrast, the KPSS test's null hypothesis is that the process is stationary. Similarly, Levene's test operates under the null hypothesis that all input samples come from populations with equal variances.

	KPSS		**ADF**	
	Original Data			
	p-value	Null Hypothesis	p-value	Null Hypothesis
Feature 1 (% WEIGHTED ILI)	0.1	stationary	5.7510×10^{-12}	Rejected/stationary
Feature 2 (% UNWEIGHTED ILI)	0.1	stationary	1.0285×10^{-11}	Rejected/stationary
Feature 3 (AGE 0-4)	0.01	Rejected/non-stationary	1.1239×10^{-8}	Rejected/stationary
Feature 4 (AGE 5-24)	0.01	Rejected/non-stationary	2.2039×10^{-8}	Rejected/stationary
Feature 5 (ILITOTAL)	0.01	Rejected/non-stationary	7.1639×10^{-8}	Rejected/stationary
Feature 6 (NUM. OF PROVIDERS)	0.01	Rejected/non-stationary	0.4243	non-stationary
Feature 7 (OT)	0.01	Rejected/non-stationary	0.7598	non-stationary
	First Differencing			
	p-value	Null Hypothesis	p-value	Null Hypothesis
Feature 1 (% WEIGHTED ILI)	0.1	stationary	1.2414×10^{-20}	Rejected/stationary
Feature 2 (% UNWEIGHTED ILI)	0.1	stationary	3.0196×10^{-21}	Rejected/stationary
Feature 3 (AGE 0-4)	0.1	stationary	1.2410×10^{-14}	Rejected/stationary
Feature 4 (AGE 5-24)	0.1	stationary	5.7794×10^{-14}	Rejected/stationary
Feature 5 (ILITOTAL)	0.1	stationary	8.6532×10^{-14}	Rejected/stationary
Feature 6 (NUM. OF PROVIDERS)	0.1	stationary	6.3667×10^{-15}	Rejected/stationary
Feature 7 (OT)	0.1	stationary	5.3756×10^{-15}	Rejected/stationary
	Seasonal Differencing			
	p-value	Null Hypothesis	p-value	Null Hypothesis
Feature 1 (% WEIGHTED ILI)	0.1	stationary	1.7049×10^{-24}	Rejected/stationary
Feature 2 (% UNWEIGHTED ILI)	0.1	stationary	5.3655×10^{-25}	Rejected/stationary
Feature 3 (AGE 0-4)	0.1	stationary	2.0115×10^{-19}	Rejected/stationary
Feature 4 (AGE 5-24)	0.1	stationary	5.6644×10^{-25}	Rejected/stationary
Feature 5 (ILITOTAL)	0.1	stationary	1.1578×10^{-24}	Rejected/stationary
Feature 6 (NUM. OF PROVIDERS)	0.1	stationary	3.7319×10^{-28}	Rejected/stationary
Feature 7 (OT)	0.1	stationary	1.1596×10^{-24}	Rejected/stationary
	Levene's Test			
	Log1p + Differencing		**Only Differencing**	
	p-value	Null Hypothesis	p-value	Null Hypothesis
Feature 5 (ILITOTAL)	0.000014	Rejected/unequal variance	1.40×10^{-15}	Rejected/unequal variance

4.4.1 Optimized Hyperparameters

In our work, the SARIMA hyperparameters used were optimized, whereas the original PatchTST paper did not optimize its hyperparameters. This difference contributes to SARIMA's superior performance in some cases. (Rana et al., 2024) conducted a comprehensive hyperparameter optimization and found that a shorter look-back window can be advantageous for short-term forecasting since predicting the near future requires less historical data. Consequently, PatchTST tends to perform worse than SARIMA for short forecasting horizons; however, its performance improves for longer horizons, likely because it lever-

Table 3: The table below displays the sMAPE and MAE metrics for both PatchTST and its optimized version, oPatchTST, derived from fine-tuned hyperparameters.

	Log1p		**Log1p + S.Diff.**		**Square Root**	
Week	PatchTST	oPatchTST	PatchTST	oPatchTST	PatchTST	oPatchTST
	sMAPEs - *ILITOTAL*					
6	23.84	19.54	26.17	23.36	27.06	20.78
12	31.28	26.27	30.53	28.85	31.87	25.30
24	31.00	28.25	31.34	28.34	29.72	28.21
36	31.06	25.57	32.23	28.30	30.14	29.81
48	24.32	23.11	27.97	27.97	28.20	26.18
60	27.70	24.28	29.65	27.47	31.96	27.17
Avg	28.20	24.50	29.64	27.38	29.82	26.24
	MAEs - All variables					
6	19881.51	15985.25	19233.16	17410.22	15818.69	13579.23
12	23602.77	21621.27	22675.73	21733.49	20537.00	18814.69
24	29559.48	26018.47	28495.22	26927.14	25617.46	23077.38
36	32954.31	29162.06	25096.66	24974.86	27244.28	23301.66
48	29512.84	32294.95	24309.00	24309.00	30600.16	24952.17
60	34074.13	30842.34	23335.12	24146.60	26688.68	26210.98
Avg	28264.17	25987.39	23857.48	23250.21	24417.71	21656.01

Table 4: This table presents the optimized hyperparameter settings for PatchTST based on the techniques applied. Abbreviations used are as follows: Learning Rate (LR), Sequence Length (SL), Encoder Layers (eL), and Model Dimensions (Mdim).

		6	12	24	36	48	60
Log1p	LR	0.0001	0.0001	0.0001	0.0001	0.0025	0.0025
	SL	60	60	104	104	104	104
	eL	2	2	4	4	4	4
	Mdim	16	16	16	16	16	16
Log1p + Seasonal Diff.	LR	0.0025	0.0001	0.0025	0.0025	0.0025	0.0025
	SL	70	104	60	60	104	60
	eL	5	4	2	2	3	3
	Mdim	64	64	16	16	16	16
Square Root	LR	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025
	SL	80	60	70	104	90	80
	eL	3	2	6	2	2	3
	Mdim	32	32	16	16	32	16

ages up to 104 past values, which is more appropriate for longer horizons. We optimized the hyperparameters for some of the best-performing techniques to illustrate how performance generally improves when these parameters are fine-tuned. The results are presented in Table 3 and the optimized hyperparameters are listed in Table 4. All other hyperparameters remain unchanged, though further optimization of other hyperparameters could potentially improve the results.

5 CONCLUSIONS AND FUTURE WORK

In this work, we have presented an enhanced channel-independent transformer, PatchTST, for time series forecasting by incorporating statistical preprocessing techniques. Our approach integrates variance-stabilizing transformations (such as Log1p, Box-Cox, and Yeo-Johnson), non-linear scaling methods (using the square root transformation), and temporal differencing to tackle non-stationarity, skewness, and heteroscedasticity in real-world data.

Experimental evaluations on the benchmark dataset demonstrate that our preprocessing strategies, including logarithmic transformations and differencing significantly improve forecasting accuracy compared to traditional normalization method. In particular, we observe a 38% improvement in sMAPE for the single target variable and a 24% in MAE for all variables. Moreover, the enhanced PatchTST model robustly captures complex temporal patterns while reducing the impact of outliers and structural breaks.

In conclusion, our innovative approach successfully reduces skewness, lowers variance, and enhances the stationarity of the dataset, resulting in more reliable forecasts. For future work, we will explore additional techniques to achieve full homoscedasticity. In addition, we plan to experiment on more diverse datasets and investigate models that incorporate channel mixing, aiming to understand their effects on capturing temporal dependencies.

REFERENCES

Ahsan, M. M., Mahmud, M. A. P., Saha, P. K., Gupta, K. D., and Siddique, Z. (2021). Effect of data scaling methods on machine learning algorithms and model performance. *Technologies*, 9(3).

Andersen, T. G., Bollerslev, T., Christoffersen, P., and Diebold, F. X. (2005). Volatility forecasting.

Bandara, K., Hewamalage, H., Liu, Y.-H., Kang, Y., and Bergmeir, C. (2020). Improving the accuracy of global forecasting models using time series data augmentation.

Bergmeir, C., Hyndman, R. J., and Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83.

Box, G. E. and Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 26(2):211–243.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Chicco, D., Warrens, M. J., and Jurman, G. (2021). The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, 7:e623.

Curran-Everett, D. (2018). Explorations in statistics: the log transformation. *Adv Physiol Educ*, 42(2):343–347.

Dastjerdi, F. R., Robinson, D. A., and Cai, L. (2022). α-hmm and optimal decoding higher-order structures on sequential data. *Journal of Computational Mathematics and Data Science*, 5:100065.

de Amorim, L. B., Cavalcanti, G. D., and Cruz, R. M. (2023). The choice of scaling technique matters for classification performance. *Applied Soft Computing*, 133:109924.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4).

Feng, C., Wang, H., Lu, N., Chen, T., He, H., Lu, Y., and Tu, X. M. (2014). Log-transformation and its implications for data analysis. *Shanghai archives of psychiatry*, 26(2):105–109.

Feng, C., Wang, H., Lu, N., and Tu, X. M. (2013). Log transformation: application and interpretation in biomedical research. *Statistics in medicine*, 32(2):230–239.

Galli, S. (2024). Python feature engineering cookbook: A complete guide to crafting powerful features for your machine learning models.

Goodfellow, I. (2016). Deep learning.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hossain, Z., Rahman, A., Hossain, M., and Karami, J. H. (2019). Over-differencing and forecasting with non-stationary time series data. *Dhaka University Journal of Science*, 67(1):21–26.

Hyndman, R. (2018). *Forecasting: principles and practice*. OTexts.

Hyndman, R. J. and Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–22.

Kuhn, M. and Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. Chapman and Hall/CRC.

Lütkepohl, H. and Xu, F. (2012). The role of the log transformation in forecasting economic variables. *Empirical Economics*, 42(3):619–638.

Miller, J. A. (1976). *Introduction to Computational Data Science using ScalaTion*.

Mudelsee, M. (2019). Trend analysis of climate time series: A review of methods. *Earth-science reviews*, 190:310–322.

Nau, R. (2016). Statistical forecasting: notes on regression and time series analysis. *Stepwise and All Possible Regressions. Available online: https://people. duke. edu/˜ rnau/regstep. htm (accessed on 2 May 2019)*.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.

Ozsahin, D. U., Taiwo Mustapha, M., Mubarak, A. S., Said Ameen, Z., and Uzun, B. (2022). Impact of feature scaling on machine learning models for the diagnosis of diabetes. In *2022 International Conference on Artificial Intelligence in Everything (AIE)*, pages 87–94.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Perone, G. (2022). Comparison of arima, ets, nnar, tbats and hybrid models to forecast the second wave of covid-19 hospitalizations in italy. *The European Journal of Health Economics*, 23(6):917–940.

Proietti, T. and Luetkepohl, H. (2011). Does the Box-Cox transformation help in forecasting macroeconomic time series?

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2022). *Dataset shift in machine learning*. Mit Press.

Raju, V. N. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., and Padma, V. (2020). Study the influence of normalization/transformation process on the accuracy of supervised classification. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 729–735.

Rana, S., Miller, J. A., Nesbit, J., Barna, N. H., Aldosari, M., and Arpinar, I. B. (2024). How effective are time series models for pandemic forecasting? In *International Conference on Big Data*, pages 3–17. Springer.

Salles, R., Belloze, K., Porto, F., Gonzalez, P. H., and Ogasawara, E. (2019). Nonstationary time series transformation methods: An experimental review. *Knowledge-Based Systems*, 164:274–291.

Seabold, S. and Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Weisberg, S. (2001). Yeo-johnson power transformations. *Department of Applied Statistics, University of Minnesota. Retrieved June*, 1:2003.

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430.

Yeo, I.-K. and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press.

APPENDIX

Reproducing Results: We executed the PatchTST code using the Illness, Weather, and ETTh1 scripts[7] and successfully replicated the exact results for the Weather and ETTh1 datasets, while obtaining closely matching results for the Illness dataset. Table 5 presents the results reported in the original paper (Nie et al., 2022) alongside our findings obtained with *drop_last* set to both True and False. Note the difference in the number of samples for both conditions. In this paper, we present the StandardScaler inverse-transformed results obtained with *drop_last* set to False.

Table 5: The table displays the sample counts for *drop_last* set to True (DL: T) and False (DL: F), along with the MAE results reported in the original paper (Nie et al., 2022) and our reproduced outcomes for both conditions using StandardScaler.

Weeks	Samples - DL: T	PatchTST	Ours	Samples - DL: F	Ours
24	160	0.814	0.734	170	0.913
36	144	0.834	0.898	158	0.890
48	144	0.854	0.879	146	0.916
60	128	0.862	0.790	134	0.875
Average	-	0.841	0.825	-	0.898

ETTh1 Dataset: In this study, we experimented with the ETTh1 dataset—one of the ETT datasets (Zhou et al., 2021)—which comprises 17,420 samples across 7 features and is available on the ETT GitHub repository[8]. Given the presence of negative values in the dataset, we applied the Yeo-Johnson transformation, which can effectively handle such data. Table 6 summarizes the MAE results using both Yeo-Johnson and StandardScaler with inverse transformation after replicating the PatchTST (Nie et al., 2022) normalized results. We adopted prediction lengths of $96, 192, 336, 720$, as in the PatchTST paper. Notably, the results using the Yeo-Johnson transformation outperform those using the StandardScaler for all four horizons.

Table 6: The table presents the MAE results on the ETTh1 dataset after applying inverse transformation using the StandardScaler and Yeo-Johnson methods. The best-performing outcomes are highlighted in bold.

Weeks	Samples	StandardScaler	Yeo-Johnson
96	2785	1.488	1.450
192	2689	1.578	1.542
336	2545	1.669	1.605
720	2161	1.779	1.674
Average	-	1.628	**1.567**

[7] https://github.com/yuqinie98/PatchTST/tree/main/PatchTST_supervised/scripts/PatchTST

[8] https://github.com/zhouhaoyi/ETDataset/tree/main