Listening to Earthquakes: Seismic Signals as Predictors

Chi Gui (chigui2), Yuang Fan (yuangf2), Yiwen Mei (ymei10)
Siebel School Comp & Data Sci - MCS

Abstract

This project predicts earthquake "time to failure" using acoustic data from lab simulations. We divide data into 150,000-row segments, labeling each with its final time_to_failure value. After standardization and feature extraction, we'll implement tree-based models and Temporal Convolutional Networks as baselines, then explore transformers, MLPs, and LSTMs. All models use regression loss functions to predict remaining time until the next seismic event.
Keywords: Seismic signals, data mining, predictive modeling, machine learning.

1 Introduction and Motivation

People have been working on forecasting earthquakes due to their catastrophic impact. Scientific efforts in this domain typically aim to answer three questions: when an earthquake will occur, where it will occur, and how large it will be. This project focuses on the temporal aspect: predicting when an earthquake will happen by analyzing real-time acoustic emissions recorded in laboratory experiments that simulate seismic activity.[1] Using time series data from these experiments, we aim to predict the time remaining before an earthquake occurs. If successful, and if the physics underlying these models can be extended from the lab to real-world environments, this approach could significantly enhance earthquake early warning systems, potentially saving lives and reducing economic loss.

2 Related Work

Earthquake prediction with acoustic time series has been tackled using both traditional machine learning and deep learning models.

Traditional ML Approaches Gabriel Preda's work[2] extracted 154 statistical features (e.g., mean, std, skewness) from segmented windows and trained Random Forests [1] and Gradient Boosting Machines [2], highlighting the importance of feature engineering. A similar Random Forest approach[3] used a reduced set of features but also demonstrated strong performance on this regression task.

Deep Learning Approaches The LANL TCN trial2 work [4] sampled 150,000-point chunks, subdivided them into 1,000-point windows and extracted 4 statistical features (mean, std, min, max) per step across 3 different time scales, then fed them into a convolutional architecture for final regression. Mayer79's RNN-based [5] model[5] used the same approach to process the data, and fed these features into a GRU for time-to-failure prediction.

[1] https://www.kaggle.com/competitions/LANL-Earthquake-Prediction/data
[2] https://www.kaggle.com/code/gpreda/lanl-earthquake-eda-and-prediction
[3] https://www.kaggle.com/code/manyregression/lanl-earthquake-prediction-fastai-random-forest/notebook
[4] https://www.kaggle.com/code/wangwangsuibinbin/lanl-tcn-trial2
[5] https://www.kaggle.com/code/mayer79/rnn-starter-for-huge-time-series

3 Methodology

3.1 Statistical Feature Selection

In our data-preprocessing pipeline, we computed a comprehensive set of univariate statistical descriptors from each raw signal. These "statistical features" provide various summaries of the distribution and were the candidates for input into our downstream predictive models.

Candidate Features We extracted the following 24 features from each time-series segment: mean, std, min, max, range, median, q01, q05, q10, q25, q75, q90, q95, q99, iqr, skewness, kurtosis, mad, rms, energy, zcr, fractal_dimension, hurst_exponent, crest_factor, rise_time

Importance Metric: To quantify each feature's relevance to prediction accuracy, we employed **permutation importance** based on the mean absolute error (MAE). Concretely, for a fitted model M and test set $(X_{\text{test}}, y_{\text{test}})$, the importance of feature j is

$$Imp(j) = \text{MAE}\big(M; X_{\text{test}}^{\text{perm}(j)}, y_{\text{test}}\big) - \text{MAE}\big(M; X_{\text{test}}, y_{\text{test}}\big)\,,$$

where $X^{\text{perm}(j)}$ denotes X_{test} with column j randomly shuffled. A larger increase in MAE upon permutation indicates a more important feature.

Model-wise Feature Rankings: We evaluated feature importance separately for five different regressors: Random Forest, MLP, LSTM, TCN, and Transformer. For each model, we then selected the ten features whose permutation led to the largest deterioration in MAE. Table 1 summarizes these top-10 lists.

Table 1: Top 10 most important statistical features for each model, ranked by permutation-based MAE increase.

Model	**Top 10 Statistical Features**
Random Forest	fractal_dimension, zcr, q95, mean, q90, crest_factor, mad, q99, range, std
MLP Regressor	range, max, min, energy, kurtosis, q01, q10, q99, rms, q25
LSTM	q01, q05, q10, q95, kurtosis, mad, q75, q90, rms, range
TCN	min, max, range, energy, q01, kurtosis, q05, q10, rms, q25
Transformer	q05, min, q10, max, energy, q01, q25, fractal_dimension, hurst_exponent, zcr

3.2 Data Processing

Due to the large, continuous nature of the seismic acoustic data, training full-length samples (150,000 data points) with models like TCN and LSTM is computationally infeasible. We therefore employ a sliding-window strategy, segmenting each seismic record into smaller, overlapping or non-overlapping windows. The continuous signals are divided into fixed segments of 150,000 data points. These segments are then processed with varying window sizes. These segments are then processed with varying window sizes (150,000, 15,000, and 1,500) and strides (150,000, 15,000, 7,500, 1,500, and 750), , producing five granularities that capture long- to short-term structure. Each segment's label is defined as the "time to failure" at its end. From every window we extract the 24 statistical descriptors listed in Section 3.1; for each downstream model we then keep its permutation-ranked *top10* features as input. The datasets have been published[6].

[6]https://www.kaggle.com/datasets/penguingui/listening-to-earthquakes/data

3.3 Baseline Methods

Random Forest predicts earthquake time-to-failure using 100 decision trees on seismic data (4194 samples $\times$ 199 timesteps $\times$ 10 features). By reshaping 3D features to 2D and training on random data subsets, it minimizes overfitting and improves generalization. Evaluated via Mean Absolute Error and R^2 score, it delivers stable, accurate predictions despite noisy geophysical data.

Temporal Convolutional Network (TCN) used causal padding and dilated convolutions to capture patterns in the seismic data. Input data was processed through stacked temporal blocks, each consisting of 1D dilated convolutions, ReLU activations, dropout layers, and residual connections to improve convergence stability. The model was trained using the MSE loss and optimized with Adam, selecting the best model using MAE on validation set.

3.4 Proposed Methods

MLP Model For the Multi-Layer Perceptron (MLP)[4] approach, each segment's extracted features were flattened into a single vector, serving as input to the neural network. The implemented MLP architecture consisted of configurable fully-connected layers, each followed by ReLU activation functions and dropout layers to mitigate overfitting. The number of hidden layers was selected as a key hyperparameter, and models with one, two, and three layers were systematically tested. Additional hyperparameters, such as dropout rate and hidden layer size, will be further explored in future experiments.

LSTM Model In the Long Short-Term Memory (LSTM)[3] based model, the sequential nature of the seismic data was explicitly leveraged by feeding the data as sequences of extracted feature vectors. The LSTM architecture consisted of configurable recurrent layers with dropout regularization between layers, followed by a fully-connected layer to output predictions. Similar to the MLP model, the number of recurrent layers was considered a key hyperparameter, and architectures with one, two, and three layers were evaluated. Future efforts will include tuning additional hyperparameters such as dropout rates and hidden layer sizes.

Transformer Model In the Transformer model[6], due to the sequence-to-single-output nature of this task, we used an encoder-only Transformer architecture. The input sequence, consisting of ten statistical features, first passes through an embedding layer that projects the features into a higher-dimensional space. Positional embeddings are then added to incorporate temporal information. The encoded representations are processed by the Transformer encoder, followed by an average pooling layer and a linear layer to produce the final output.

All models were trained using the mean absolute error (MAE) loss function and optimized with the Adam optimizer. Model performance was monitored and logged systematically, enabling effective tracking and tuning of hyperparameters.

4 Results

Table 2 compares Random Forest, TCN, MLP, and LSTM performance in earthquake "time to failure" prediction across five window/stride configurations. LSTM achieves lowest MAE (1.56-1.72 public, 2.49-2.91 private), performing best in smaller windows (W_1.5K_S_1.5K: 1.57 public, 2.49 private) due to superior temporal modeling. Random Forest delivers consistent results (MAE: 1.67-1.78 public, 2.64-2.68 private) but lacks LSTM's precision. TCN (1.80-2.14 public, 2.97-3.35

private) and MLP (1.76-2.16 public, 2.90-3.36 private) both underperform with smaller windows, struggling with dynamic patterns. **LSTM emerges as the most effective model overall.**
To obtain these MAE scores, predicted "time to failure" values were exported to a CSV file and submitted via the official Kaggle competition submission portal [7], which returns both public and private MAE metrics.

Table 2: Performance Comparison of Earthquake Time-to-Failure Prediction Models across Different Window/Stride Configurations.

	W_150K_S_150K		**W_15K_S_15K**		**W_15K_S_7.5K**		**W_1.5K_S_1.5K**		**W_1.5K_S_0.75K**	
Model Type	**Private**	**Public**	**Private**	**Public**	**Private**	**Public**	**Private**	**Public**	**Private**	**Public**
Random Forest	2.65	1.68	2.68	1.69	2.69	1.67	2.69	1.72	2.67	1.74
TCN	2.97	1.80	3.28	2.03	3.14	1.95	3.35	2.08	3.35	2.14
MLP	2.75	1.97	2.94	1.86	3.14	1.74	3.63	2.05	4.22	2.87
LSTM	3.01	1.70	**2.50**	**1.61**	2.71	**1.60**	**2.67**	**1.56**	**2.59**	**1.50**
Transformer	**2.65**	**1.62**	2.72	1.69	**2.68**	1.75	2.69	1.68	2.72	1.67

The table reports private (approximately 87% of test data) and public (approximately 13% of test data) MAE score from Kaggle for Random Forest, TCN, MLP, and LSTM models. W_15K_S_15K, for example, indicates data segmented with a window size of 15,000 and a stride size of 15,000.
The **LSTM model** consistently delivers the lowest MAE across all window/stride configurations—most notably in the smallest windows, demonstrating its superior ability to capture fine-grained temporal dependencies in seismic signals. The **Transformer** closely follows, especially at larger window sizes (e.g., 1.62 public at W_150K_S_150K), suggesting that self-attention can effectively balance local and global context when sufficient data length is available. In contrast, non-recurrent architectures suffer as temporal resolution tightens: the TCN's fixed receptive field and the MLP's lack of sequential inductive bias lead to pronounced MAE increases in smaller windows, underscoring their limited capacity to model rapid signal fluctuations.
Despite not modeling sequences explicitly, Random Forest remains remarkably stable (public MAE $\approx$ 1.67–1.74), leveraging static statistical features for reasonable predictions. However, it never matches the precision of LSTM or Transformer in high-resolution settings. The consistent ranking of models on both public ($\sim$13% of data) and private ($\sim$87% of data) splits indicates robust generalization: while private MAEs are uniformly higher—reflecting more challenging or out-of-distribution segments—the relative performance gaps persist.
Overall, these results recommend LSTM (or Transformer) architectures for "time to failure" forecasting, with LSTM favored for its simplicity and robustness on tight windows and Transformer offering a scalable alternative for longer temporal contexts.

5 Ablation Study on Input–Feature Count

Goal. This ablation study focuses on *how the number of statistical features used affects the performance of LSTM.*
Keeping every other hyper-parameter fixed, we compare LSTM trained with the top–5 features against those using the top–10 features identified by permutation importance (Section 3.1).

[7] https://www.kaggle.com/competitions/LANL-Earthquake-Prediction/submissions

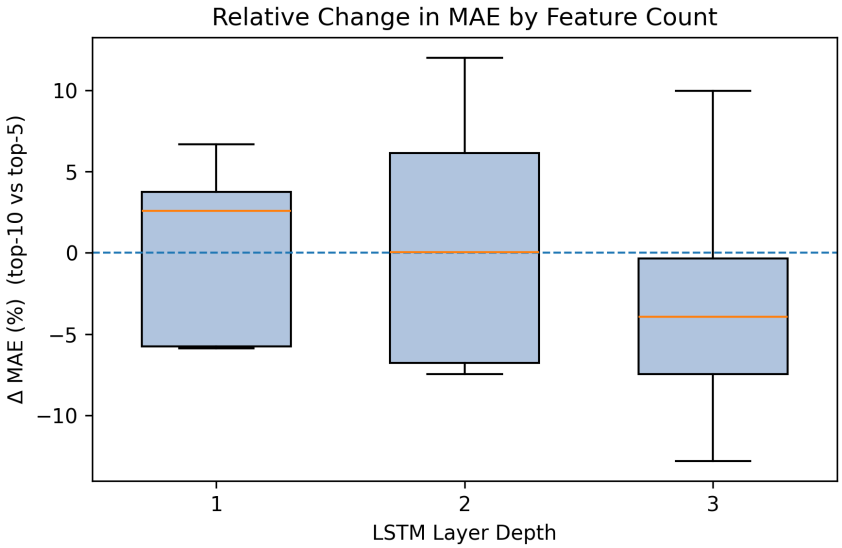


Figure 1: Distribution of the change in error $\left(\Delta\text{MAE} = \text{MAE}_{\text{top-10}} - \text{MAE}_{\text{top-5}}\right)$ for each LSTM depth across the five window/stride settings. Boxes indicate the inter-quartile range, centre lines mark the median, and whiskers span the full window-to-window variation; the dashed horizontal line corresponds to parity ($\Delta = 0$). Values above the line mean that using ten features increases error, whereas values below denote an improvement.

Score aggregation. Kaggle's test set is split into a public (13 %) and a private (87 %) partition. To obtain a single headline number for each run we compute a weighted mean-absolute-error

$$\text{MAE}_{\text{agg}} = 0.13\,\text{MAE}_{\text{public}} + 0.87\,\text{MAE}_{\text{private}},$$

mirroring the competition's final ranking. Table 4 reports these aggregated scores for every window/stride configuration (5 in total), feature set size (5 vs. 10), and LSTM depth (1–3 layers).

Findings. Figure 1 visualises, for each depth, the change $\Delta\text{MAE} = \text{MAE}_{10} - \text{MAE}_{5}$ across the five windows. Three consistent patterns emerge:

1. **1-layer LSTM** — extra features slightly *degrade* accuracy (median $\Delta > 0$), suggesting capacity is insufficient to exploit the richer input.
2. **2-layer LSTM** — the effect is neutral (median $\Delta \approx 0$) and highly window-dependent.
3. **3-layer LSTM** — adding five more descriptors now yields a mean 4 % MAE reduction (median $\Delta < 0$); the larger model benefits from the additional information.

The wide whiskers in the figure indicate that the magnitude of the gain (or loss) still varies with the temporal window. Overall, the study shows that *feature richness pays off only once the network has enough depth*, a useful guideline for future model design.

6 Parameter Study

Table 3 summarizes the performance comparison between MLP and LSTM models across various datasets. For each dataset, we trained three models with different layer numbers (1, 2, and 3) while

keeping the other parameters constant (hidden size of 100, dropout rate of 0.5, 100 epochs, learning rate of 0.001, and batch size of 32). We then selected and reported the best results for each file based on the private and public scores. This approach allowed us to determine the optimal model configuration for each dataset.

The Transformer model was tested on the following hyperparameters: embedding dimension: (32, 64), number of heads in multi-head self attention: (4, 8), number of stacks of the encoder: (2, 4), the forwarding dimension in the encoder: (64, 256), and the dropout rate: (0, 0.2). The model that performed the best is the one with hyperparameters 32, 8, 4, 64, 0.2 respectively on the W_150K_S_150K dataset.

7 Implications of Experimental Findings

Capacity must match feature richness. Adding five extra descriptors *hurts* a single-layer LSTM, is neutral for two layers, and helps once three layers are used. Deeper networks have enough capacity to filter noise and exploit subtle interactions; shallow ones do not.

Hand-crafted statistics still matter when data are scarce. Despite the availability of Transformers and TCNs, the Random-Forest baseline—fed only ranked statistics—remains competitive on the largest windows, confirming that lightweight feature engineering is a strong option when samples are few and signals are noisy.

8 Future Work

While our current framework demonstrates promising results, several limitations remain. Addressing these limitations will help enhance our work's practical utility and deepen understanding.

Training Data Scope Our current model is trained exclusively on seismic data collected within 16 seconds prior to a series of laboratory-simulated earthquake events. No data from periods far from an earthquake were included, which could lead to a high rate of false positives when applied to real-world settings. Expanding the training dataset to include background seismic signals from non-event periods will help the model make more precise decisions and reduce false alarms.

Magnitude Consideration The magnitude of an earthquake significantly influences the impact of an earthquake. However, our current model does not account for variations in magnitude during training or inference. Future work could explore predicting both the time-to-failure and the magnitude of the incoming earthquake, which would provide more comprehensive information to aid in coping with earthquake events.

Explainability and Interpretability Although our model, especially LSTM, demonstrates predictive capabilities, being a deep neural network it offers limited transparency regarding how the decision output is made. More explainable AI techniques can be used to increase model explainability and improving understanding in the relationship between seismic signal patterns and an occurrence of an earthquake.

9 Contributions

- **Chi Gui:** Implemented the sliding-window segmentation, trained and tuned the MLP and LSTM models, and conducted the ablation experiments.
- **Yuang Fan:** Implemented and trained the TCN and Transformer models.
- **Yiwen Mei:** Implemented Statistical Feature Selection and Random Forest model.

References

[1] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[2] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[4] Rudolf Kruse, Sanaz Mostaghim, Christian Borgelt, Christian Braune, and Matthias Steinbrecher. Multi-layer perceptrons. In *Computational intelligence: a methodological introduction*, pages 53–124. Springer, 2022.

[5] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A Model Parameters

A.1 Random Forest

- n_estimators = 100 (The number of trees in the forest is set to 100.)
- random_state = 42 (The seed for the random number generator is set to 42.)

A.2 TCN

Model Parameters:

- number of layers: 2, 3, 4
- number of channels: 32, 64
- kernel size: 3
- dilation: [1, 3], [1, 3, 9], [1, 3, 9, 27]
- number of stacks of residual block: 1, 2
- dropout rate: 0.2, 0.5

Training Parameters:

- epoch: 100
- learning rate: 0.001
- batch size: 32

A.3 MLP

Model Parameters:

- hiden size: 100
- dropout: 0.5
- layer number: 1, 2, 3

Training Parameters:

- epoch: 100
- learning rate: 0.001
- batch size: 32

A.4 LSTM

Model Parameters:

- hiden size: 100
- dropout: 0.5
- layer number: 1, 2, 3

Training Parameters:

- epoch: 100
- learning rate: 0.001
- batch size: 32

B Training Process

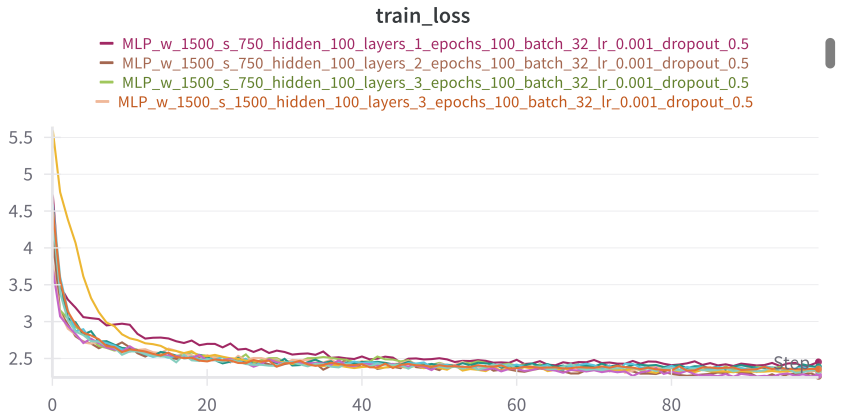
Figure 2 and 3 show the training process record for the MLP and LSTM models. We used wandb[8] to draw the plots.

C Details Results

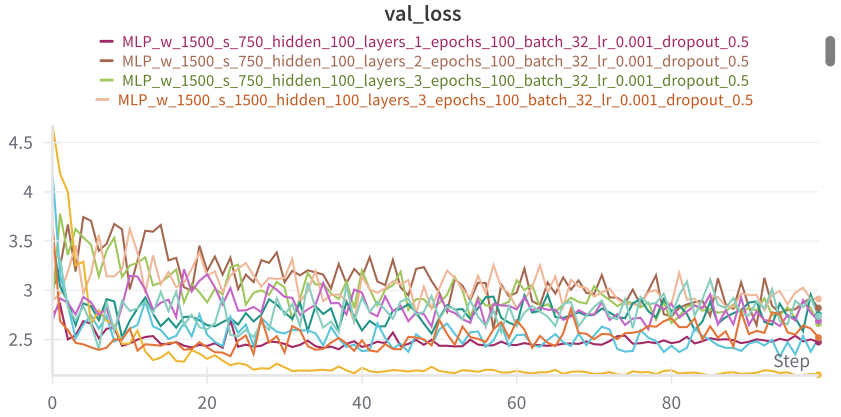
Table 3: Performance Comparison of MLP and LSTM with Different Parameter Configurations. For each dataset, we trained three models with different layer numbers (1, 2, and 3) while keeping the other parameters constant (hidden size of 100, dropout rate of 0.5, 100 epochs, learning rate of 0.001, and batch size of 32). MLP-1 means the layer number is 1 for this model.

	W_150K_S_150K		**W_15K_S_15K**		**W_15K_S_7.5K**		**W_1.5K_S_1.5K**		**W_1.5K_S_0.75K**	
Model Type	**Private**	**Public**	**Private**	**Public**	**Private**	**Public**	**Private**	**Public**	**Private**	**Public**
MLP-1	3.03	1.75	**2.94**	**1.86**	**3.14**	**1.74**	4.00	3.98	5.10	**2.15**
MLP-2	2.88	**1.68**	3.44	1.90	3.77	2.05	4.21	5.56	**4.22**	2.87
MLP-3	**2.75**	1.79	3.18	1.87	3.59	2.01	**3.63**	**2.05**	4.24	2.16
LSTM-1	3.16	1.72	**2.50**	**1.61**	2.79	**1.58**	2.75	1.61	2.74	1.61
LSTM-2	**3.01**	1.70	**2.50**	1.70	**2.71**	1.60	3.04	1.73	**2.59**	**1.50**
LSTM-3	3.08	**1.69**	2.59	**1.61**	2.84	1.61	**2.67**	**1.56**	2.65	1.51

[8]https://wandb.ai/site/



(a) MLP training loss

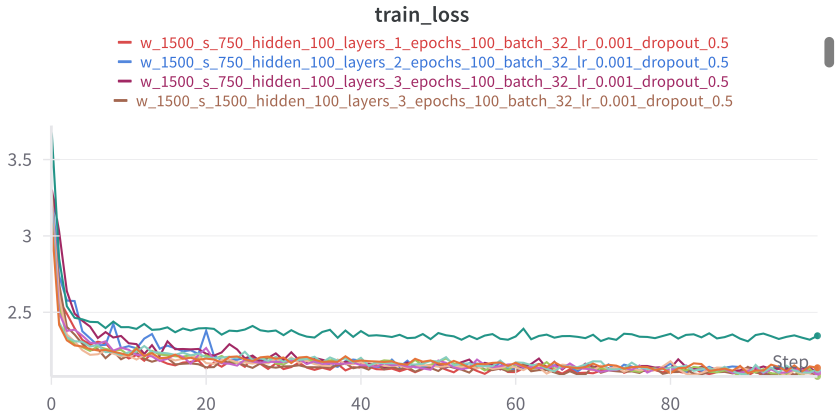


(b) MLP validation loss

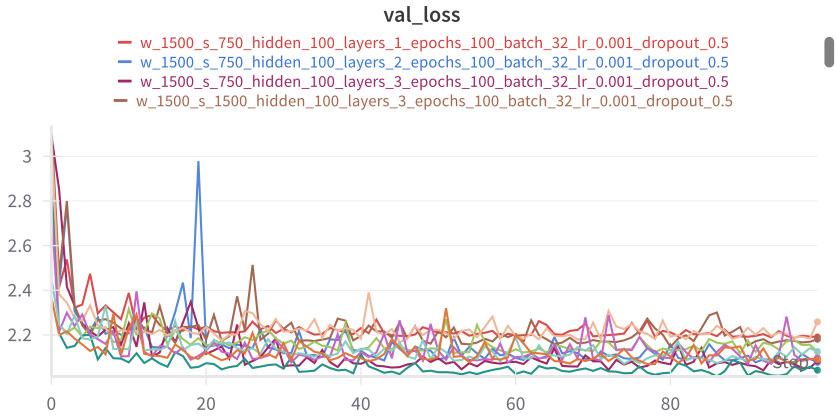
Figure 2: MLP training process

Table 4: Aggregated MAE (0.13 Public + 0.87 Private) for top-5 vs. top-10 feature subsets. Lower is better.

Window / Stride	**Top-5 features**			**Top-10 features**		
	1-layer	2-layer	3-layer	1-layer	2-layer	3-layer
W_150K_S_150K	2.79	2.68	2.64	2.97	2.84	2.90
W_15K_S_15K	2.53	2.59	2.82	2.38	2.40	2.46
W_15K_S_7.5K	2.54	2.56	2.69	2.63	2.57	2.68
W_1.5K_S_1.5K	2.76	2.56	2.73	2.60	2.87	2.53
W_1.5K_S_0.75K	2.53	2.63	2.60	2.59	2.45	2.50



(a) LSTM training loss



(b) LSTM validation loss

Figure 3: LSTM training process

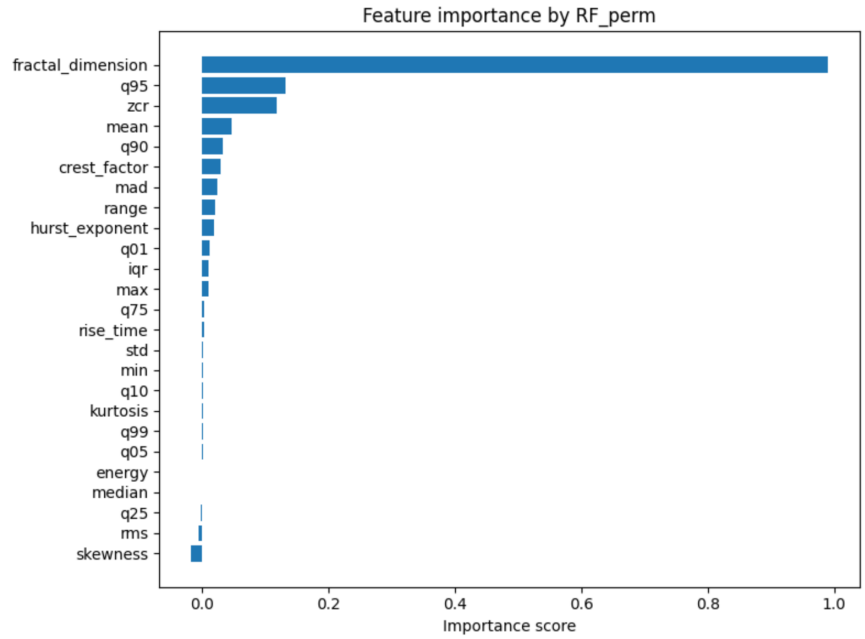


Figure 4: Feature importance by RF_perm

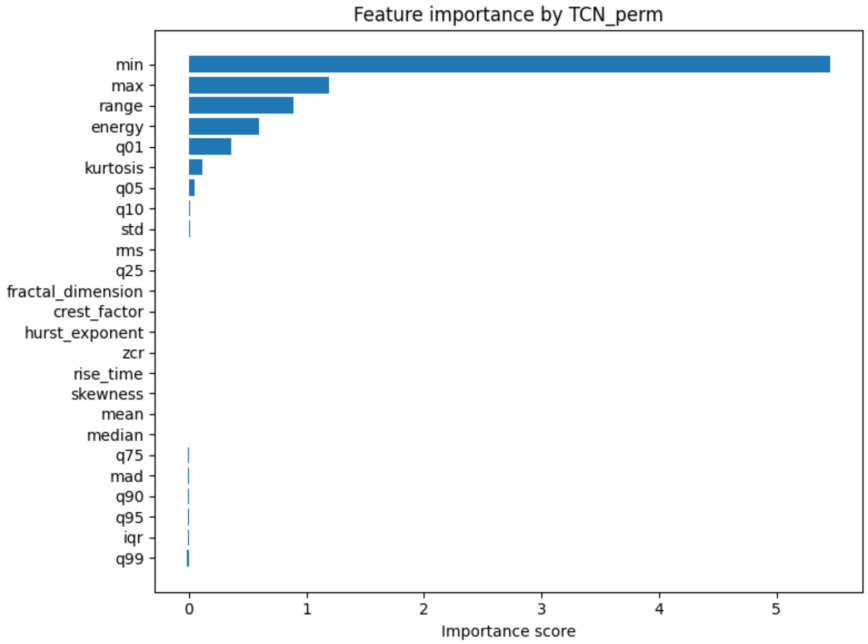


Figure 5: Feature importance by TCN_perm

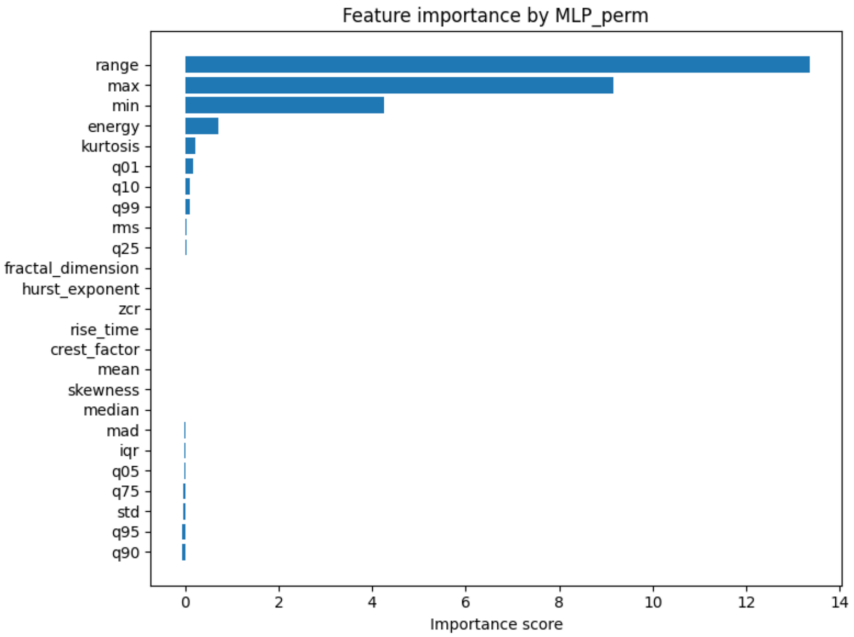


Figure 6: Feature importance by MLP_perm

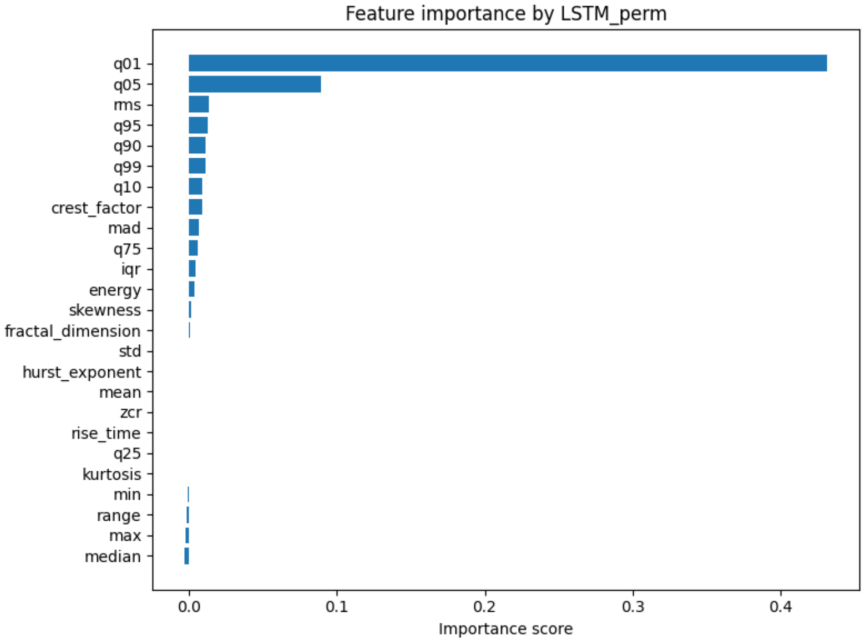


Figure 7: Feature importance by LSTM_perm

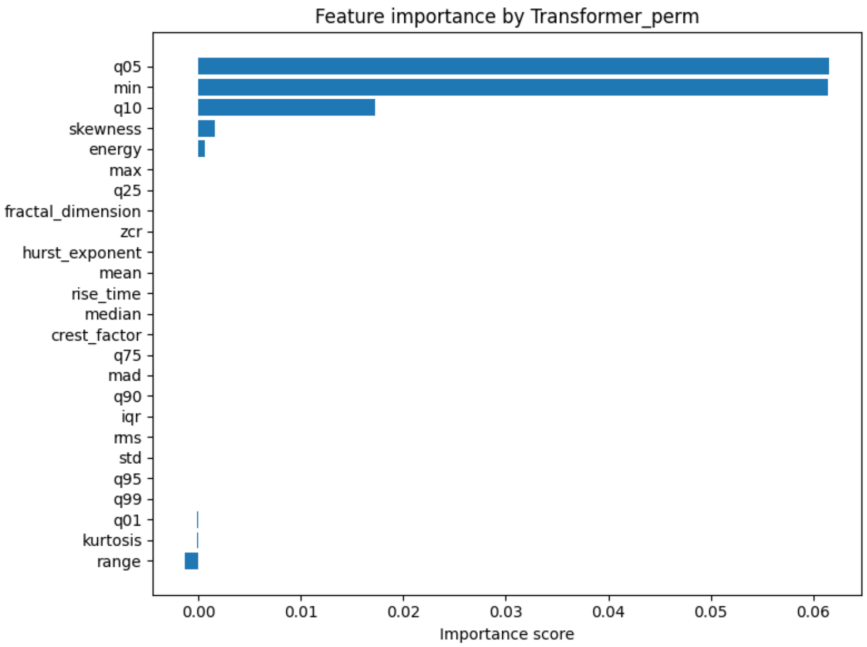


Figure 8: Feature importance by Transformer_perm