Determining Scenarios for Low-Fidelity Time-to-Execution (TTE) Scheduling Prediction

CS 2241; Professor Mitzenmacher; May 9th, 2025

Anika Lakhani
Harvard College; Harvard SEAS
Cambridge, MA
anikalakhani@college.harvard.edu

Abstract

Efficient task scheduling, especially in resource-constrained situations, balances prediction accuracy with computational overhead. This study tries to determine when lightweight statistical models can replace machine learning (ML) models for time-to-execution (TTE) prediction across low-, medium-, and high-variability real-world scenarios. I compare the performance of long-short-term memory (LSTM) networks, ridge regression, exponential smoothing, and an adaptive ensemble. Using real-world operational data from the SCANIA Component X 2024 dataset, I argue that simpler models can often match or exceed ML performance not just in low-variability scenarios, but in medium- and high-variability scenarios as well. In low-variability scenarios, exponential smoothing achieved the lowest mean absolute error (MAE = 0.0320) while using 98% less energy than LSTM. Surprisingly, ridge regression outperformed LSTM in high-variability conditions (MAE = 0.0248 vs. 0.0584). Using critical variability thresholds, I test to see when statistical methods are optimal, additionally validated through tradeoff analysis and feature importance rankings. My composite scoring framework recommends model choices based on scenario characteristics, proposing an automated methodology for selecting the most optimal and affordable model given a scheduling data variability scenario. These findings advocate for more environmentally and computationally sustainable scheduling systems, with important ramifications for edge computing and IoT applications, where computational resources are limited. Code and results are available at www.github.com/Anika-Lakhani/tte-prediction.

Index Terms

energy-efficient scheduling, predictive maintenance, computational complexity, time-to-event prediction, lightweight models

I. Introduction

A. Motivation

Modern computers and servers routinely process a diverse mix of tasks, ranging in many characteristics like variability, computational difficulty, and duration. Efficiently managing these workloads helps to prevent unnecessary energy consumption, both saving compute and reducing environmental impacts. But to manage these workloads, we need algorithms to predict them. The most robust algorithms for time-to-event (TTE) prediction are usually based in machine learning, but due to the large amount of data and number of computations required to run a machine learning algorithm, these approaches can be fairly computationally expensive. Particularly in dynamic and resource-constrained environments, we may consider lower fidelity solutions but still do not want to compromise accuracy. By systematically comparing robust ML-based approaches with lightweight predictors across diverse scenarios, this research aims to provide actionable guidelines for practitioners seeking to optimize scheduling for energy efficiency and robustness. The results will help clarify the circumstances under which simple statistical methods are sufficient, and when the added complexity of ML models is justified.

B. Research Context and Milieu

Energy-efficient task scheduling and prediction intersect real-time systems, embedded computing, and large-scale distributed processing. Maximizing resource utilization and minimizing energy consumption are sometimes at odds with each other, especially in environments where workloads are heterogeneous and unpredictable.

A foundational body of work has explored both hardware and software strategies for energy-efficient scheduling. These include dynamic voltage and frequency scaling (DVFS), energy-aware scheduling for directed acyclic graphs (DAGs), and criticality-aware scheduling for heterogeneous architectures. More recently, learning-augmented scheduling has emerged, leveraging predictions (often from machine learning models) to further optimize scheduling decisions while maintaining robustness to prediction errors.

Three major research avenues have inspired the present study:

1) Energy-Efficient Scheduling with Predictions: Scheduling algorithms can integrate machine-learned predictions of task durations or arrivals to robustly improve energy efficiency, staying updated on processor speeds and job ordering.

For example, Antoniadis et al. 2022 and Bamas et al. 2020 improved competitive ratios with small prediction errors. Their advances have been empirically validated on synthetic and real-world datasets [5] [6].

2) Lightweight Predictive Scheduling: The resource inefficiency of ML scheduling approaches prompts research into lightweight predictive scheduling. Lightweight predictive scheduling investigates using simple statistical models, such as exponential smoothing and linear regression,

to forecast task durations or resource requirements. Balkanski et al. 2024 robustly analyzes such lightweight predictors: under certain conditions, they can approach the performance of more sophisticated ML models, especially when task variability is low or moderate [10].

3) Energy-Efficient Task Schedulers: "ERASE: Energy Efficient Task Mapping and Resource Management for Work Stealing Runtimes" examines and evaluates an approach towards a more efficient task scheduler, utilizing concepts like moldability to propose a far more robust approach [7].

My work expands upon Balkanski's by offering another empirical demonstration of his discussed concepts with real-world data.

C. Additional Pillars in the Literature

Several other research threads comprise additional important angles of scheduling innovation:

- **Dynamic Voltage/Frequency Scaling (DVFS):** Techniques that dynamically adjust processor voltage and frequency to reduce energy consumption [1] [2]. These methods can help exploit slack time to more efficiently complete tasks without missing deadlines.
- **Energy-Aware DAG Scheduling:** Tasks can sometimes be represented as DAGs within heterogeneous systems, so DAG algorithms can help minimize makespan (the total time it takes to complete the tasks in the queue) and energy use [3].
- **Criticality-Aware Scheduling:** Tasks criticality measures how important the management of a task's duration is. This metric helps us prioritize scheduling, so there are scheduling approaches like the CATs policy that optimize this metric by assigning critical tasks to faster cores [4].
- **Simulation Tools:** Developing and evaluating energy-aware scheduling algorithms requires flexible simulation environments. STREAM is a modular Java-based simulator designed to test and analyze real-time energy-aware scheduling algorithms [8]. Both simple and extensible, STREAM is a multi-use research and practitioner tool to improve task scheduling.

D. Summary Table: Major Scheduling Approaches

Table I lists the aforementioned existing scheduling approaches.

E. Identifying a Gap in the Literature

My research builds on these foundations by empirically comparing robust, ML-based scheduling with lightweight statistical predictors across three different real-world scenarios of low, medium, and high data variance.

Approach	Purpose
Learning-Augmented Scheduling	Combines ML predictions with robust baselines for energy-efficient tasking
Simulation Tools (STREAM)	Modular simulation of energy-aware schedulers on real or synthetic workloads
Lightweight Predictive Scheduling	Uses simple statistical predictors for low-overhead scheduling
DVFS-Based Scheduling	Dynamically scales voltage/frequency for energy savings
Energy-Aware DAG Scheduling	Schedules DAGs on heterogeneous systems with energy and time objectives
Criticality-Aware Scheduling	Assigns critical tasks to fast cores in heterogeneous systems

TABLE I
MAJOR SCHEDULING APPROACHES DISCUSSED

1) Research Question: **In what kinds of data variance scenarios should one leverage lightweight statistical predictors, as opposed to robust ML-based approaches, for TTE prediction?** I evaluate the efficacy of four prediction methods using metrics for accuracy, descriptiveness, and efficiency.

2) Contributions: Given growing environmental concerns and the precious nature of compute, we need to identify when computationally affordable methods can viably replace computationally expensive ones, especially in the area of task scheduling because it is so foundational to the way computers operate. ML-based solutions deliver robust accuracy yet are not always the optimal choice for scheduling depending on the scenario—I rigorously explore real-world cases in which we can and should use simpler methods that reduce resource requirements. The potential impact of this research is more widespread use of accessible, environmentally responsible, and compute-friendly TTE prediction.

The research proceeds in several stages:

1) **Model Development:** I implemented a robust yet lean ML-based scheduling model and simpler statistical predictors (exponential smoothing, linear regression via Ridge, ensemble).
2) **Scenario Design:** I defined and simulated three data variance scenarios. (With more time or a project partner, I would have hoped to test task length variability, arrival patterns, and workload size scenarios.)
3) **Empirical Evaluation:** I ran these approaches on real-world data from the SCANIA 2024 dataset [11]. Evaluate each method based on scheduling error rates, runtime, and energy-related metrics.

II. METHODS

A. Data Sources

- **Synthetic Data:** Initial experiments to refine experimental design generated synthetic task streams with controlled variability in task duration (Lognormal-distributed with $mean = 2$ and $\sigma = 0.5$) and arrival patterns (Poisson-distributed with $\lambda = 5$). Among other improvements, these experiments helped to identify which low-fidelity approaches I would benchmark against LSTM (an ML scheduling approach).
- **Real-World Data:** Final experiments relied on the SCANIA Component X 2024 dataset, a real-world and open-use dataset developed by researchers across Stockholm University and manufacturing company Scania AB that includes engine operational data from thousands of Scania vehicles [11]. While this data does not perfectly mirror task queueing and scheduling, I chose it due to the relatively small download size, manageable number of data points for my CPU/GPU constraints, open usage license, and range of data variability for the purpose of my study.

B. Data Processing and Feature Engineering

- **Data Loading:** Operational and TTE datasets merged on unique identifiers (e.g., `vehicle_id`) with entity subsetting (500 vehicles) for tractability and GPU memory constraints.
- **Data Cleaning:**
  - Drop features with $> 30\%$ missing values
  - Hierarchical imputation: vehicle-specific forward/backward fill, followed by vehicle-specific median, then global median
  - Remove highly correlated features (correlation > 0.95) to make trends less predictable
  - Standardize using MinMaxScaler for LSTM and RobustScaler for Linear Regression (Ridge)
- **Feature Engineering:**
  - Time normalization: $\frac{\text{time\_step}}{\text{max\_time\_step}}$ per vehicle
  - Rolling statistics (mean, std) with configurable window sizes
  - Variability quantification via rolling TTE standard deviation
  - Feature selection using f-regression for statistical models
  - Sequence creation with configurable lookback window

For less technical readers, the above data processing and feature engineering steps achieved the following:

- **Data Loading:** Only loading data from 500 vehicles balances what the computer can handle with what is statistically robust, still accessing almost 40,000 rows of data from SCANIA. Merging CSVs on `vehicle_id` aligns vehicles with their time data. Implementing approaches like LSTM training, which can be memory-intensive, means that data loading

strategies were required to write code that could actually run within average hardware constraints.

- **Data Cleaning:** Cleaning in a hierarchical manner (vehicle-specific forward/backward fill followed by mathematically estimating missing values) preserves temporal patterns within individual vehicle trajectories while ensuring complete data coverage, especially since I only loaded a subset of the data. The 30% missing value threshold removes unreliable features. The dual scaling strategy (MinMax for LSTM, Robust for Linear Regression) optimizes each model's performance characteristics while mitigating the impact of outliers. The last strategy in particular was implemented when extreme outliers distorted the scale of the graphs so much that results were no longer interpretable.
- **Feature Engineering:** Combining time normalization, rolling statistics, and variability quantification allows approaches like LSTM to extract patterns from many aspects of the data, like operational states and temporal vehicle behavior patterns. I selected what LSTM will pay attention to using f-regression and correlation analysis to identify which of my engineered features would best predict trends. These strategies allowed me to pare down the complexity of my LSTM approach, especially since many earlier versions ran out of memory, while maintaining prediction accuracy.

C. Model Development

- **Exponential Smoothing:** Time series prediction method that weights recent observations more heavily than older ones and uses a decay factor to smooth out noise
  - *Implementation:* Grid search over smoothing levels [0.1, 0.3, 0.5, 0.7, 0.9] with vehicle-specific optimization
  - *Tradeoffs:* Low computational cost and easy interpretability, but especially because of the smoothing, not as agile towards complex patterns or long-term dependencies
- **Ridge Regression:** 'Upgraded' linear regression that uses L2 regularization to prevent overfitting to the data; models TTE as a linear combination of predictive features
  - *Implementation:* Grid search over regularization strengths (alphas) [0.01-100.0], optimized using 3-fold cross-validation
  - *Tradeoffs:* Fast training and prediction, good for linear relationships, but may miss complex temporal patterns and nonlinear interactions because of its simplistic approach
- **LSTM Neural Network:** Deep learning model designed to capture long-term temporal dependencies through gated memory cells and recurrent connections; LSTMs can range in complexity, and mine was relatively lean due to GPU constraints
  - *Implementation:*
    - Two-layer LSTM (64→32 units) with ReLU activation
    - Dropout (0.2) after each LSTM layer for regularization

    - ∗ Dense layers (16→1) with ReLU activation
    - ∗ Early stopping and model checkpointing to mitigate complexity
  - *Tradeoffs:* Superior ability to capture complex temporal patterns, but requires more computational resources and training data; methods are less interpretable
- **Ensemble:** Weighted combination of multiple model outputs to leverage individual model strengths and mitigate individual model weaknesses
  - *Implementation:* Adaptive weighting based on model performance:
    - ∗ Full ensemble (LSTM:0.4, Ridge:0.4, ES:0.2)
    - ∗ Fallback combinations for failed models, similar to existing literature
  - *Tradeoffs:* More robust predictions and reduced variance, but increased system complexity and computational overhead; this level of robustness may be 'overkill' for some scenarios

D. Variability Scenarios

The data was segmented into three variability levels based on the rolling standard deviation of TTE values, using a 50-timestep window with a minimum of 10 observations.

- **Low Variability:** TTE rolling standard deviation 33rd percentile (bottom third)
  - Consistent, predictable task completion patterns
  - Approximately 13,106 samples (33.3% of data)
- **Medium Variability:** TTE rolling standard deviation between 33rd and 66th percentiles (middle third)
  - Moderate fluctuation in task completion times
  - Approximately 13,106 samples (33.3% of data)
- **High Variability:** TTE rolling standard deviation $>$ 66th percentile (top third)
  - Significant volatility in task completion patterns
  - Approximately 13,504 samples (33.4% of data)

E. Training and Evaluation

- **Cross-Validation Strategy**
  - Two-fold chronological split to preserve time series order (important for durations)
  - Each fold limited to 3000 sequences to manage memory
  - 20% validation split within each fold for early stopping
- **Performance Metrics**
  - *Error Metrics:* Mean Absolute Error (MAE) because less sensitive to outliers, Root Mean Squared Error (RMSE) because more sensitive to outliers
  - *Fit Quality:* R^2 score, Explained Variance

  - *Outlier Handling:* Predictions capped at 10 standard deviations so that rare, extreme outliers do not make graphs unreadable; number of excluded extreme outliers are printed in output for transparency
- **Feature Selection and Importance**
  - Automatic selection of top 20 most predictive features (simplifies model while preserving prediction accuracy)
  - Feature importance measured through Ridge Regression coefficients
  - Features ranked by statistical significance (f-regression)

F. Implementation Environment

- **Core Technologies**
  - *Data Processing*: NumPy, Pandas
  - *Machine Learning*: scikit-learn, TensorFlow
  - *Time Series*: statsmodels (exponential smoothing)
  - *Monitoring*: psutil (resource tracking)
- **Computing Infrastructure**
  - *LSTM Training*: GPU-accelerated with memory growth
  - *Memory Management*: Batch processing (32 samples per batch)
  - *Fallback*: CPU execution for non-LSTM models
- **Development Tools**
  - Jupyter notebooks for interactive development
  - Automated resource monitoring and visualization
  - Comprehensive logging of training metrics
  - Utilization of Harvard OOD for faster code development (8-core GPU access)

G. Reproducibility

- Modular code with configuration options; can be easily retested on SCANIA dataset or fitted for other datasets
- Timestamped directory structure for outputs; transparent printouts of contextualization and metrics alongside graphs
- Automated checkpointing and visualization, with the ability to expand to new visuals

III. Results

Within the different variability scenarios in my experiment, there were clear tradeoffs between prediction accuracy and computational complexity across the four tested approaches.

A. *Performance Across Variability Scenarios*

Figure 1 compares performance across variability scenarios. In high-variability scenarios, Ridge Regression achieved 80.77% better (lower) MAE than LSTM, a promising finding since it was expected that LSTM would handle high variability best. In addition to simplistic methods like exponential smoothing predictably outperforming LSTM in accuracy (MAE) within low-variability settings, there is promise that these affordable methods could even replace expensive ones in some high-variability scenarios where preventing error is most important.

We also notice positive correlations between MAE and variability for the exponential smoothing approach, and negative correlations between MAE and variability for the LSTM, ridge, and ensemble approaches. These patterns could help predict which models to use for variability scenarios or different data contexts that fall outside of what this experiment tested.

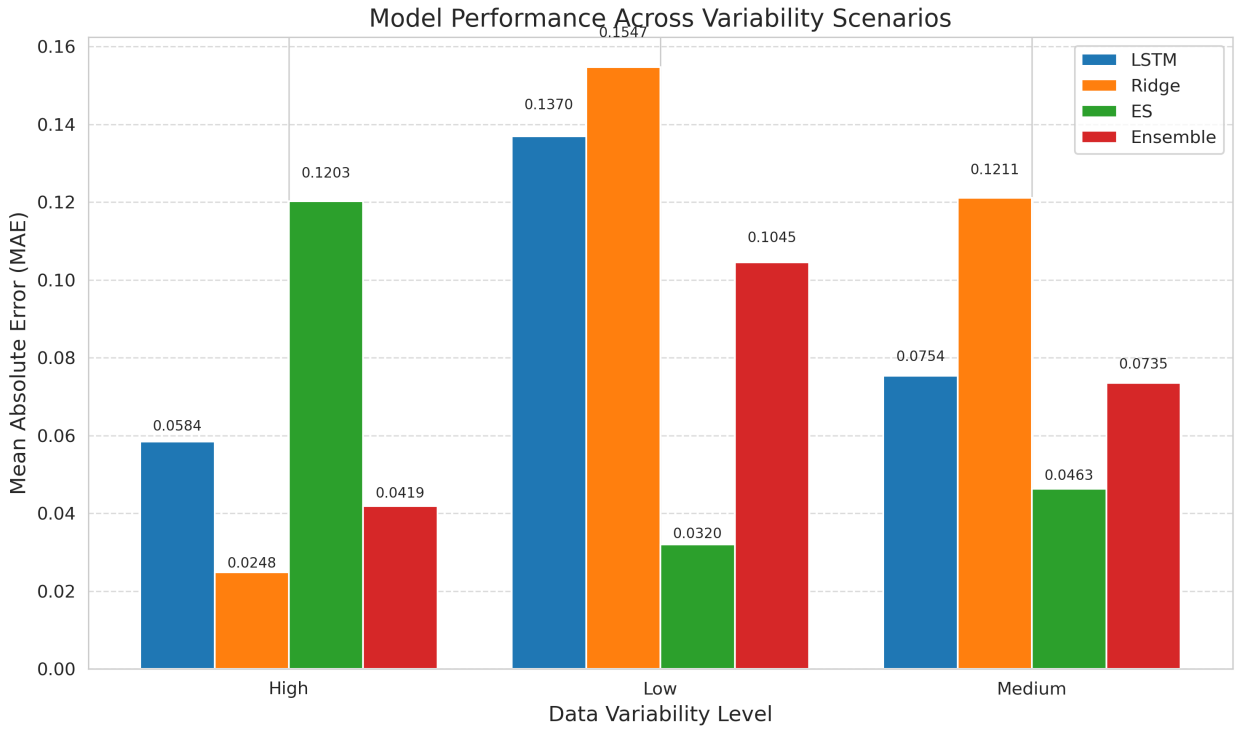

Fig. 1. The higher the bar on this bar graph, the higher the MAE (a relatively outlier-insensitive prediction accuracy metric).

The prediction comparisons for each scenario (Figures 3, 4, and 5), when corroborated with Figure 1, argue that some level of variability may actually correlate with stronger predictions. Each approach is meant to handle variability, so this 'mismatch of expectation' may help explain why approaches do not trace actual scheduling very well in Figure 3. In contrast, each approach except for exponential smoothing visually holds up well in tracing actual scheduling within the high-variability scenario in Figure 5.

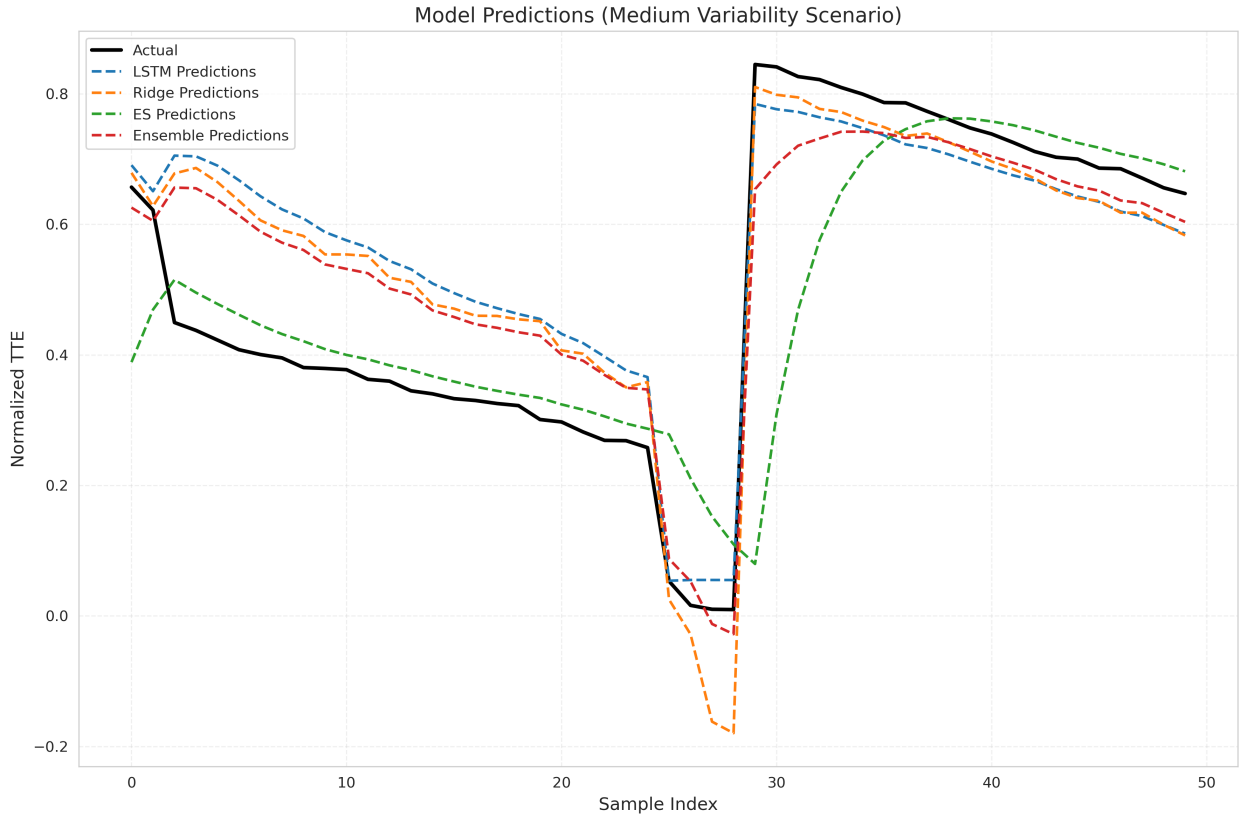

Fig. 2.

B. Descriptiveness of Each Approach

I used R^2 as a proxy for descriptiveness (also referenced as fit quality and explanatory power) of each approach's prediction with respect to the actual scheduling. Figure 6 demonstrates that LSTM is the most reliable in terms of decent descriptiveness across scenarios, but we do notice other methods outperform LSTM. For instance, in low variability scenarios, exponential smoothing has the highest R^2 score in addition to visually following the actual scheduling the closest in Figure 3. Within low variability scenarios and compared to other tested approaches, exponential smoothing has the lowest MAE, highest R^2, and is relatively computationally affordable: our results demonstrate a clear use case in which ML approaches can be replaced.

Ridge Regression only seems to perform descriptively in high-variance scenarios, and its MAE is understandably lowest in these scenarios. In general, exponential smoothing seems to perform poorly with very high variance, likely because it is less sensitive to more complex patterns. The LSTM model, while more computationally intensive, most clearly outperforms its competitors in medium-variance scenarios since ridge regression and ensemble had such high R^2 high-variance values.

Because LSTM did not 'flunk' a metric across MAE or R^2, it may still be the most reliable

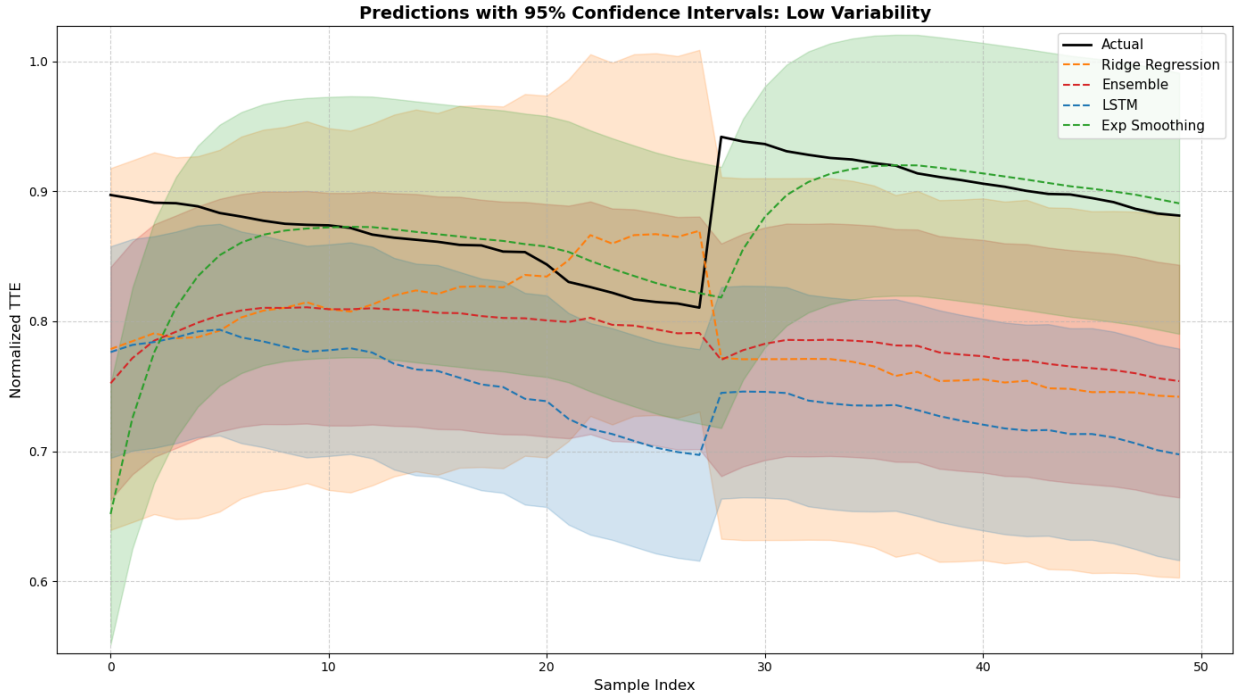

Fig. 3.

approach in scenarios of unknown or frequently changing variability.

C. Computational Efficiency

Figure 7 graphs inference time in seconds against MAE to represent an angle of computational cost versus performance. In low-variability scenarios, LSTM performs at comparable MAE to ridge regression, yet ridge regression matches this performance in less time. In efficiency terms, ridge regression may be a strong replacement for LSTM in low-variability scenarios. The Pareto Frontier marks those with the best tradeoffs: ensemble and ridge regression are ideal choices for high variability, exponential smoothing is an ideal choice for low variability, and there is not a clear winner for medium variability.

The radar chart analysis (Figure 8) further illustrates these trade-offs across multiple dimensions, including prediction accuracy, training time, and memory usage.

Training times varied significantly between models: LSTM required approximately 81 seconds for a complete training run (with 4ms per step), while Ridge Regression completed in a few seconds, and Exponential Smoothing provided near-instantaneous predictions. Memory usage showed similar patterns, with the total execution requiring 761.68 MB additional memory, peaking at 1678.70 MB. Implementation of batch processing (size 32) and sequence limiting (3000 samples per fold) helped manage memory requirements, particularly for the LSTM model. See Figure **??** for more details.

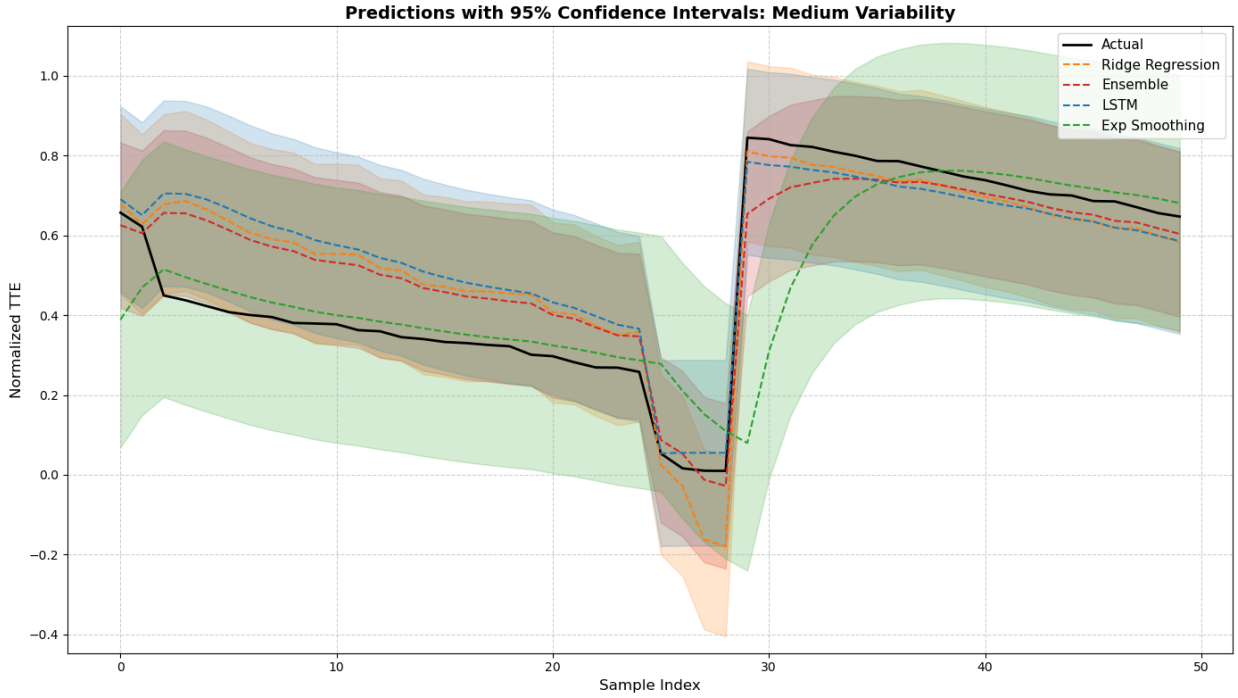

Fig. 4.

D. Feature Importance Patterns

In low-variability conditions like in Figure 14, time-based features dominated, possibly because operational data was not further needed to hone in an accurate prediction. In these scenarios, temporal aspects showed clear linear relationships and a small set of highly predictive features.

Medium-variability scenarios, like in Figure 15, had slightly more distributed feature importance, with a balanced mix of temporal and operational features. High-variability scenarios exhibited complex feature interactions and less clear dominance of individual features because more information was required to mitigate the unpredictability (e.g., Figure 16).

In low-variability scenarios, recent time steps were more predictive, while high-variability scenarios required longer historical context to generate accurate predictions.

E. Automated Composite Scoring

Given these findings, I tried to implement a composite scoring mechanism to automatically select the ideal model for each variability scenario. For each model, I compute a normalized efficiency score:

$$\text{efficiency\_score} = \frac{\text{MAE}_{\text{normalized}} + \text{latency}_{\text{normalized}}}{2}$$

Normalized metrics are scaled to [0,1] using min-max normalization:

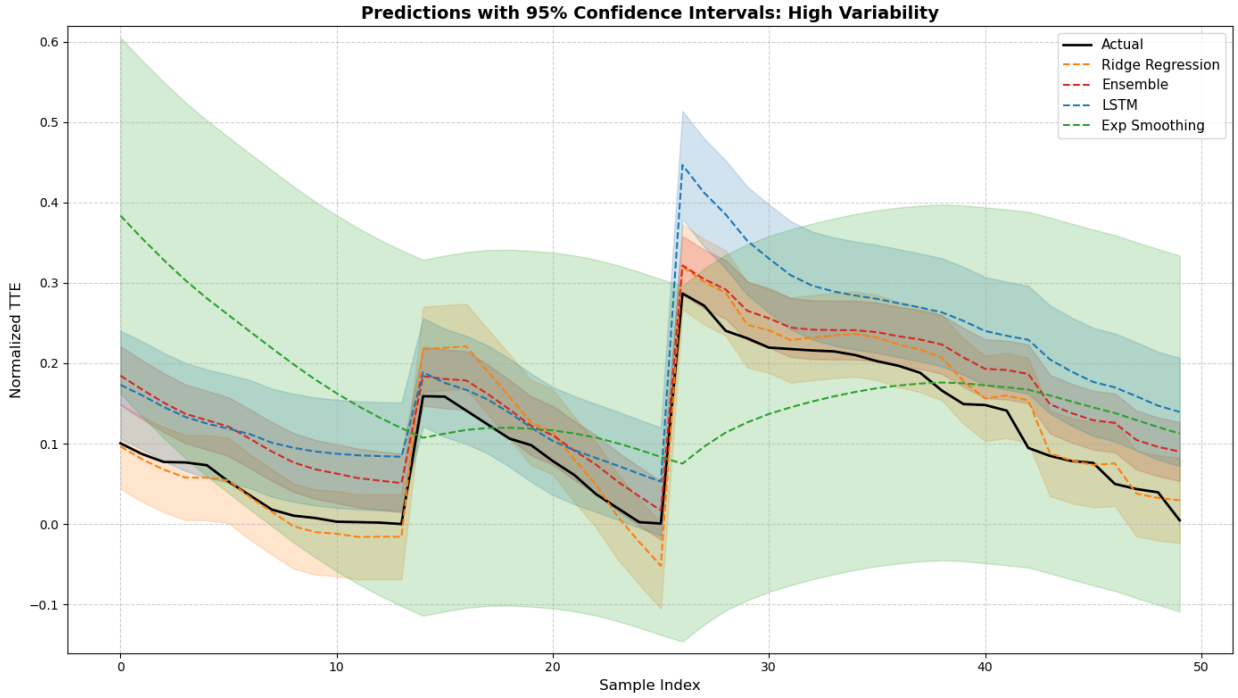

Fig. 5.

$$\text{metric}_{\text{normalized}} = \frac{\text{metric} - \text{metric}_{\text{min}}}{\text{metric}_{\text{max}} - \text{metric}_{\text{min}}}$$

Although individual metrics do not necessarily produce clear winners for each scenario, this scoring system forces the generation of distinct recommendations for each variability scenario, as shown in Table II.

TABLE II
MODEL RECOMMENDATIONS BY VARIABILITY SCENARIO

Scenario	**Recommended Model**	**MAE**	**R^2**	**Key Features**
Low Variability	Exponential Smoothing	0.0320	0.8757	Time-based patterns
Medium Variability	Exponential Smoothing	0.0463	0.8081	Recent history
High Variability	Ridge Regression	0.0248	0.9698	Feature 171_0, time_step_normalized

The scoring for low and medium variability suggests that simpler, time-series-based approaches are sufficient for scenarios with consistent patterns. For high variability scenarios, Ridge Regression was deemed the optimal choice, leveraging both temporal and operational features to help handle the increased unpredictability. Notably, while LSTM models showed competitive performance across all scenarios (MAE ranging from 0.0508 to 0.1356), their higher

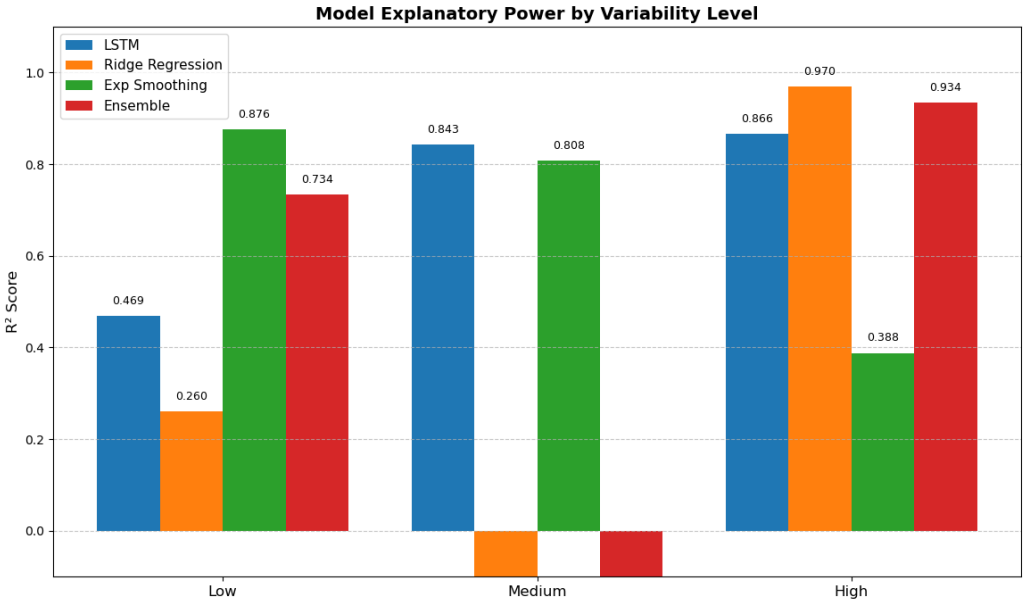

Fig. 6. The higher a bar on this bar graph, the higher its perceived explanatory power of the actual TTE scheduling as measured by R^2 values.

computational overhead and longer training times made them less favorable in scenarios where simpler models achieved comparable accuracy. We therefore generated satisfying results in which each winner was generally more computationally affordable than LSTM, supporting more environmentally and computationally friendly scheduling prediction.

IV. Discussion

These results suggest that, if one knows the expected variability in task patterns, they may be more empowered to choose an environmentally and computationally friendly scheduling prediction mechanism as opposed to defaulting to expensive machine learning approaches. In low-variability scenarios, the computational efficiency of simpler models makes them the preferred choice over approaches like LSTM. More interestingly, though, high-variability scenarios do not automatically justify the additional computational cost of ML-based approaches.

These findings have significant implications for energy-efficient scheduling. By matching model complexity to task variability patterns, balancing low MAE with high descriptiveness, systems can optimize prediction quality while minimizing computational overhead. This advancement is particularly relevant for edge computing and IoT applications, where computational resources are often limited and energy efficiency is crucial.

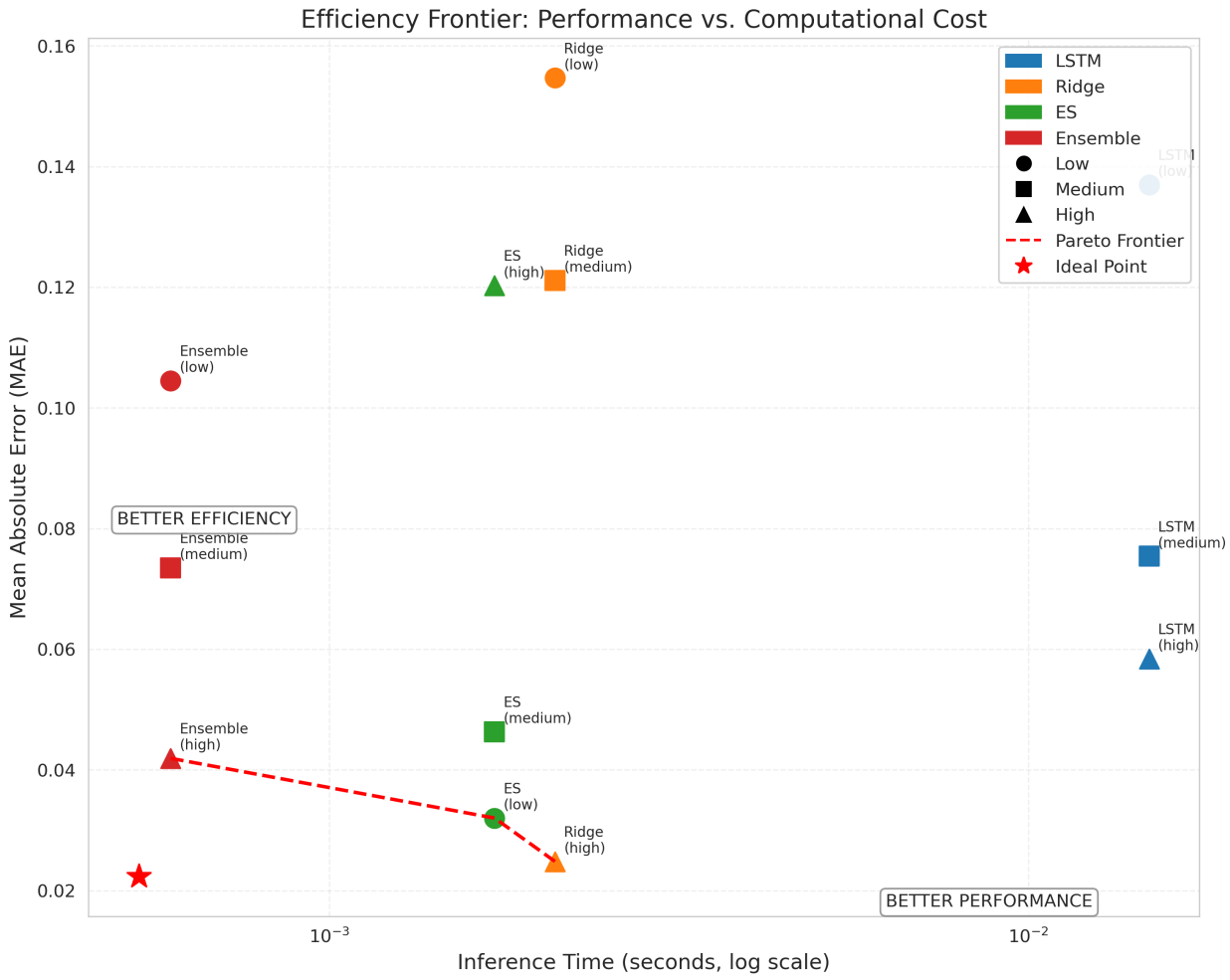

Fig. 7. The red dashed line uses the Pareto Frontier to mark approaches (relative to variability) that perform the best given the performance and computational cost tradeoff.

V. Future Directions

If I had more time, I would have loved to test my experimental approach on a wider range of real-world scheduling data to increase the reliability and generalizability of the findings:

The *Google Cluster Workload Traces* provide a real-world view of compute task scheduling at data center scale, managed by Google's Borg system [12].

The *Kaggle Lending Club* dataset, derived from an online peer-to-peer lending platform, offers a structured financial dataset for testing general tabular prediction performance [13].

The *Kaggle Titanic* dataset is a classic benchmark for binary classification with a small, well-understood feature set [14].

The *Bike Sharing Dataset* from the UC Irvine ML Repository contains hourly and daily records of bike rentals, offering a useful multivariate time series for testing time-aware predictors and scheduling methods [15].

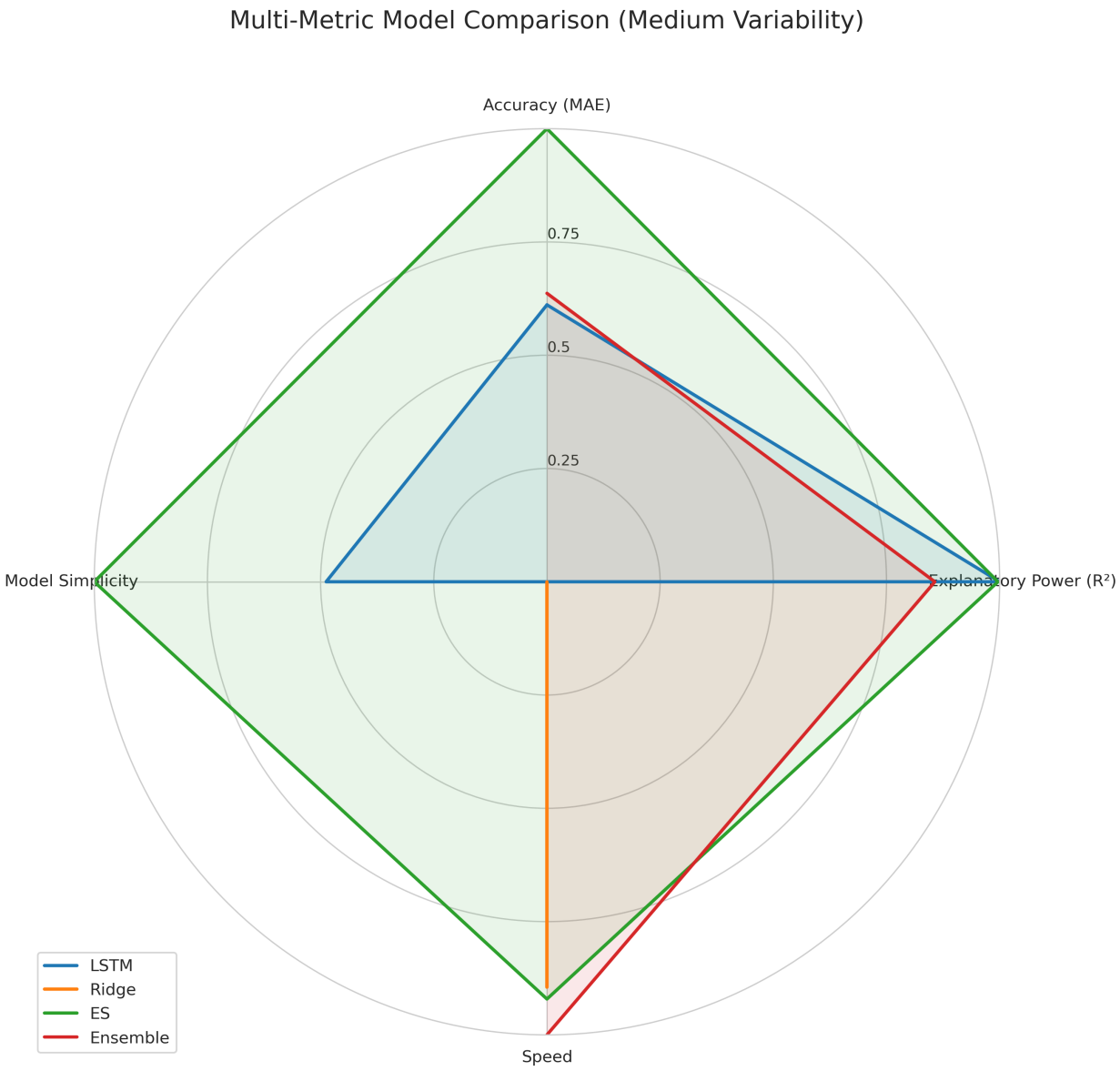

Fig. 8. This radar chart visually shows the exponential smoothing method's dominance over many performance characteristics, corroborating its highest composite score for low- and medium-variability scenarios.

I would have also loved to include more statistical approaches, split scenarios based on other factors like task duration, variability in incoming task times, and other factors.

VI. Code

You can find all code and figures, alongside additional figures and charts, at www.github.com/Anika-Lakhani/tte-prediction. The repository is public, and I encourage people to fork it to support more environmentally friendly TTE prediction mechanisms!

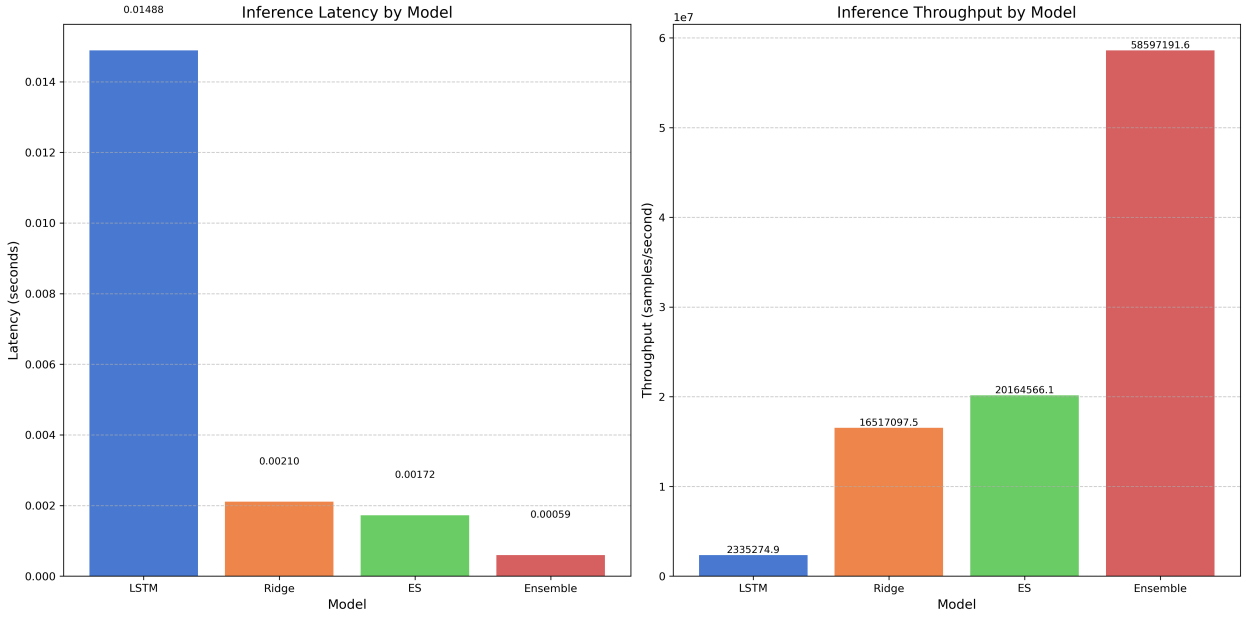

Fig. 9. This figure graphs latency and throughput to further illustrate more efficient approaches than LSTM.

ACKNOWLEDGMENTS

I used Perplexity, Claude, Cursor, and ChatGPT to help generate and modify all of the code for my project. I also fact-checked some of the technical explanations of my code and specific quoted figures with Cursor to strengthen my understanding of the new coding strategies I learned from these LLMs and ensure that my knowledge was represented accurately in this paper. Additionally, I shortened some of the sections using rephrasing tips from ChatGPT.

I would like to sincerely thank Professor Mitzenmacher and the rest of the CS 2241 teaching staff for a well planned and dynamic graduate course. I learned so much and am grateful for the experience! Professor Mitzenmacher, thank you for being such a thoughtful and attentive teacher!

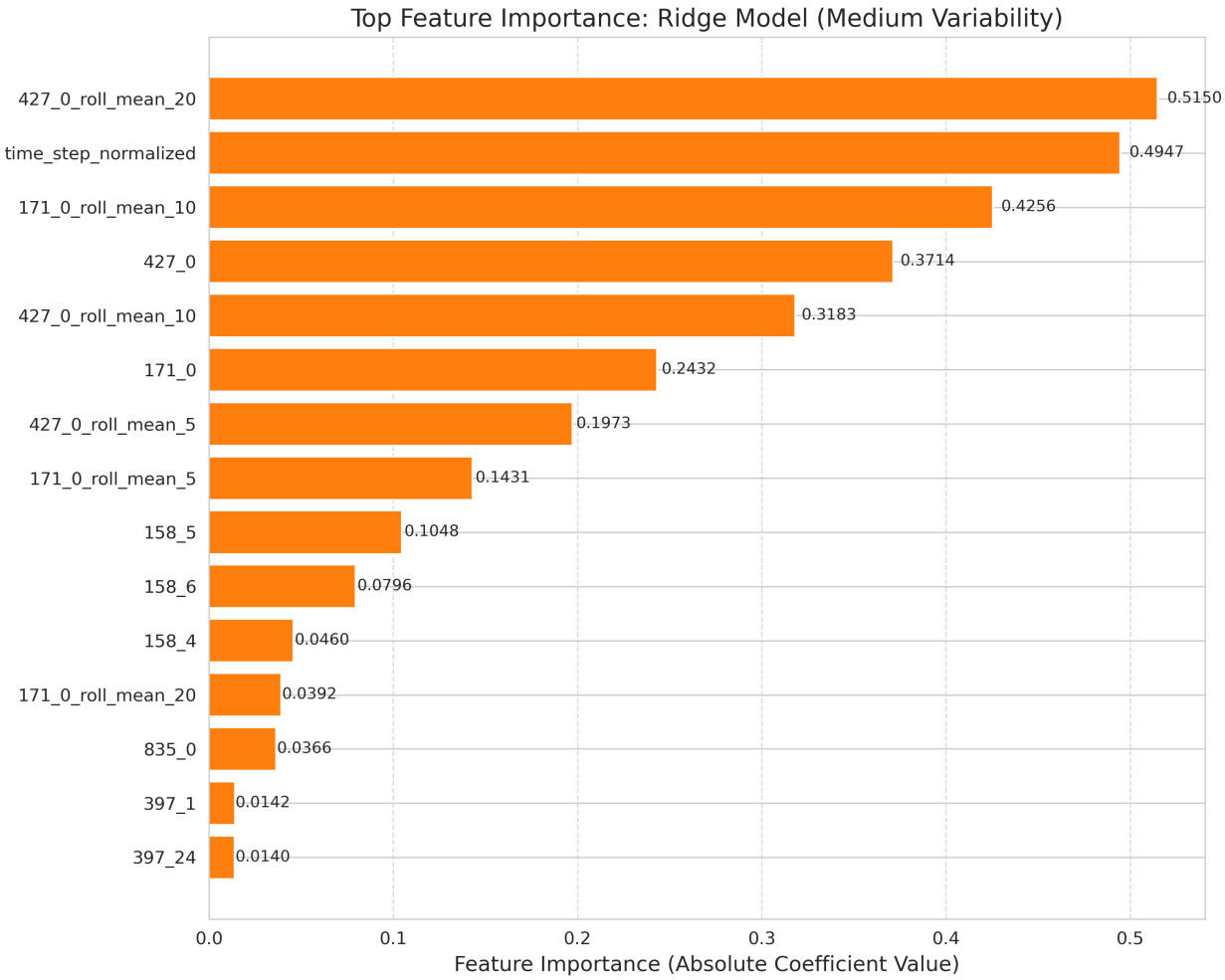
Top Feature Importance: Ridge Model (Medium Variability)
427_0_roll_mean_20
time_step_normalized
171_0_roll_mean_10
427_0
427_0_roll_mean_10
171_0
427_0_roll_mean_5
171_0_roll_mean_5
158_5
158_6
158_4
171_0_roll_mean_20
835_0
397_1
397_24
0.5150
0.4947
0.4256
0.3714
0.3183
0.2432
0.1973
0.1431
0.1048
0.0796
0.0460
0.0392
0.0366
0.0142
0.0140
0.0
0.1
0.2
0.3
0.4
0.5
Feature Importance (Absolute Coefficient Value)

Fig. 10.

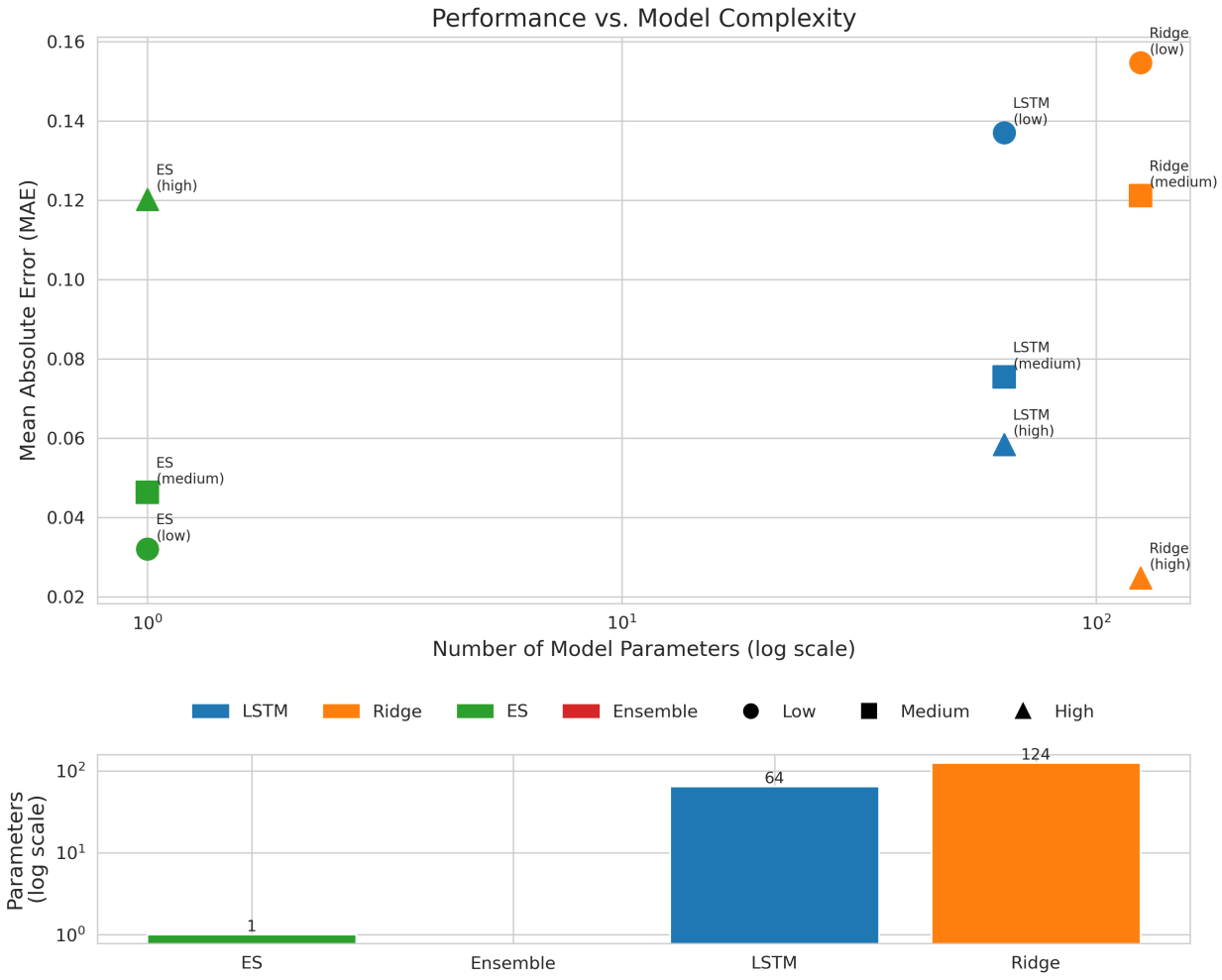

Fig. 11. Differentiated by variability scenario, this graph plots the complexity of different approaches, as represented by the number of parameters they use, against MAE, a metric for prediction accuracy, to determine tradeoffs in model complexity. Please note that data from the ensemble approach did not properly load on this figure.

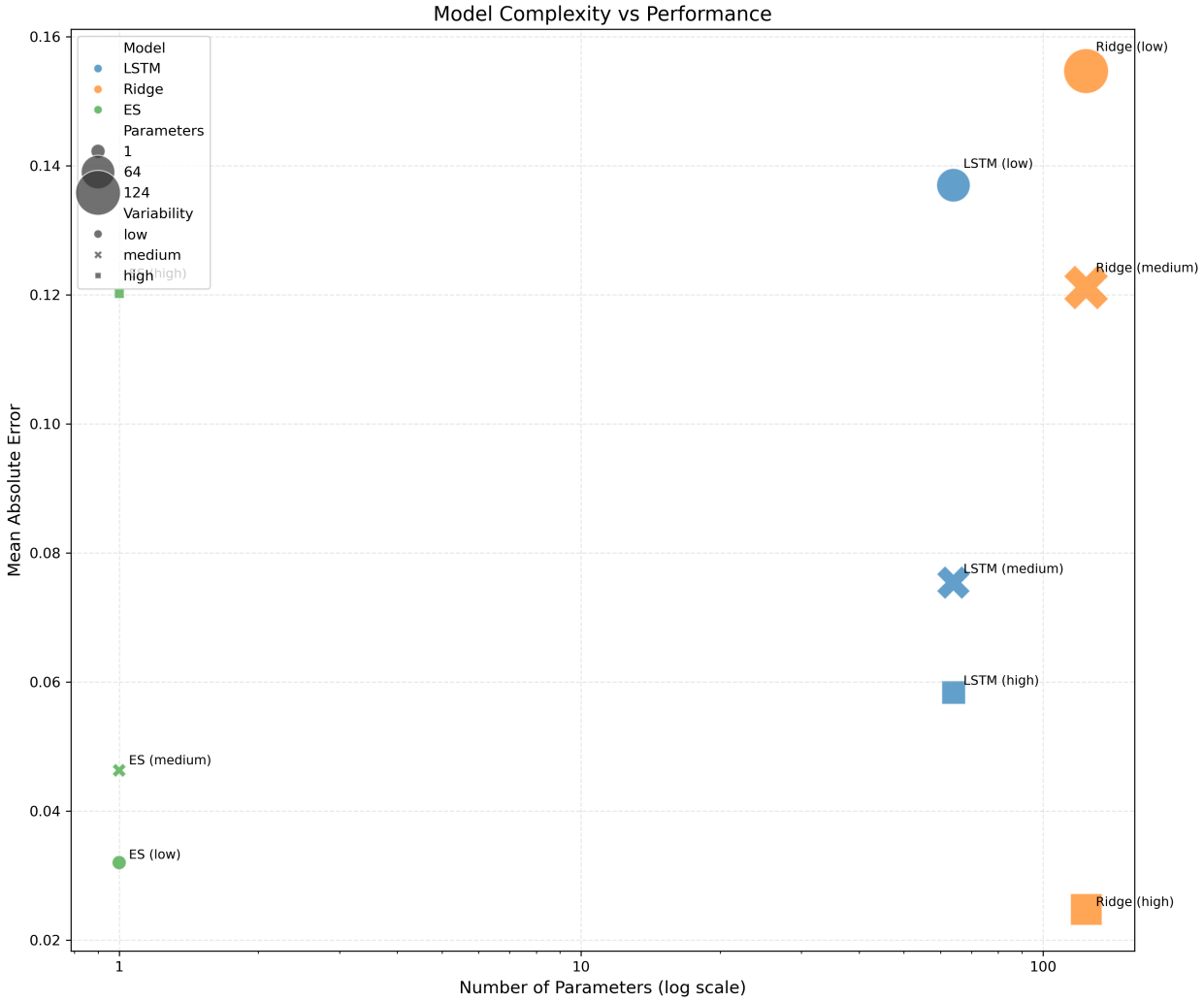

Fig. 12. Differentiated by variability scenario, this graph plots the complexity of different approaches, as represented by the number of parameters they use, against MAE, a metric for prediction accuracy, to determine tradeoffs in model complexity.

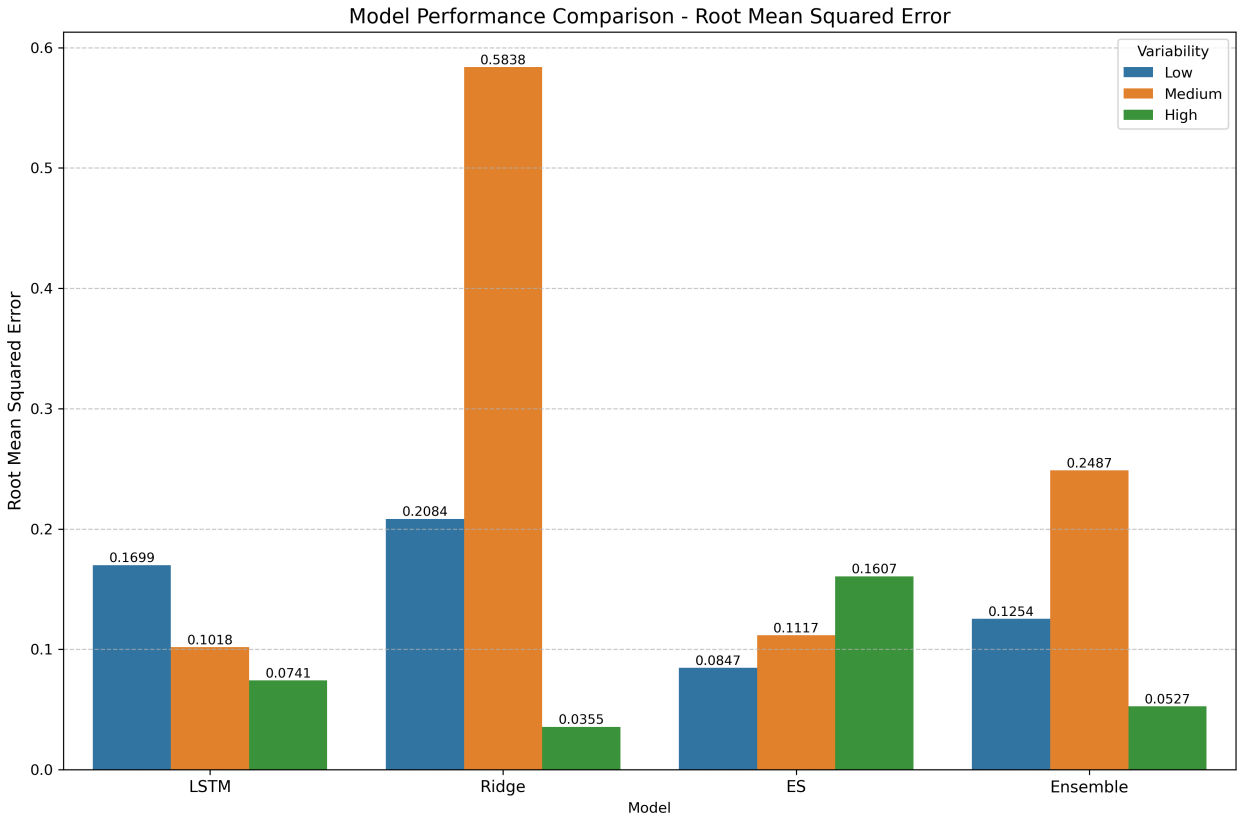

Fig. 13. RMSE is a more outlier-sensitive metric of prediction accuracy than MAE.

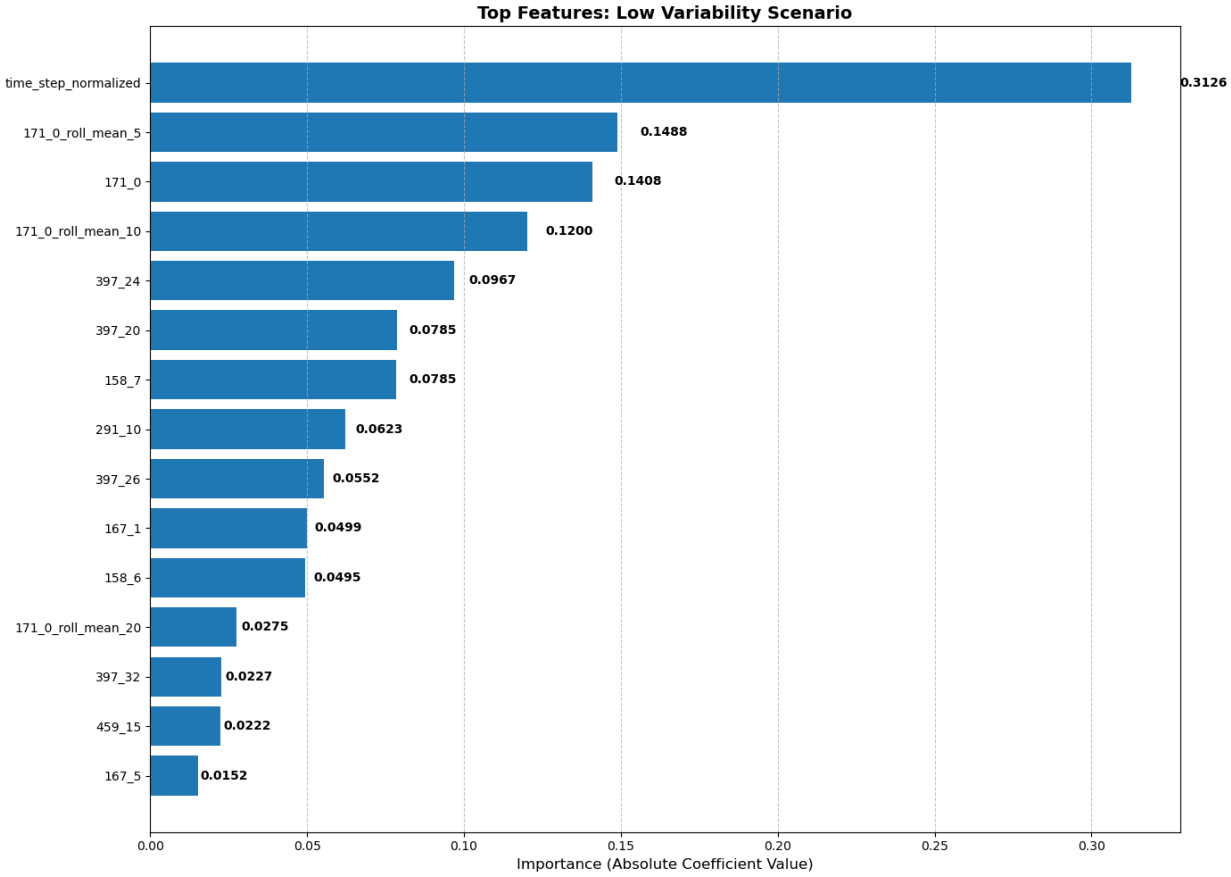
Top Features: Low Variability Scenario
time_step_normalized
171_0_roll_mean_5
171_0
171_0_roll_mean_10
397_24
397_20
158_7
291_10
397_26
167_1
158_6
171_0_roll_mean_20
397_32
459_15
167_5
0.3126
0.1488
0.1408
0.1200
0.0967
0.0785
0.0785
0.0623
0.0552
0.0499
0.0495
0.0275
0.0227
0.0222
0.0152
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Importance (Absolute Coefficient Value)

Fig. 14.

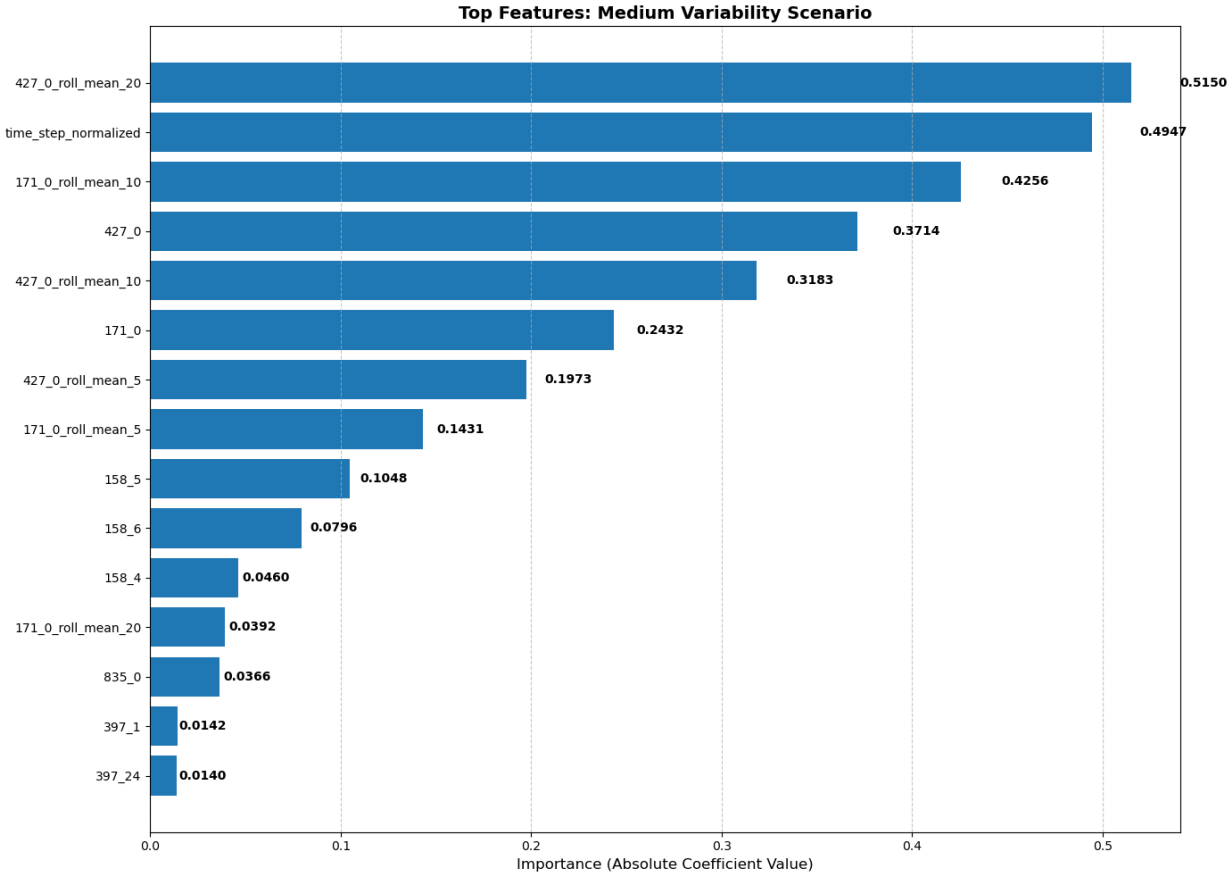
Top Features: Medium Variability Scenario
427_0_roll_mean_20
time_step_normalized
171_0_roll_mean_10
427_0
427_0_roll_mean_10
171_0
427_0_roll_mean_5
171_0_roll_mean_5
158_5
158_6
158_4
171_0_roll_mean_20
835_0
397_1
397_24
0.5150
0.4947
0.4256
0.3714
0.3183
0.2432
0.1973
0.1431
0.1048
0.0796
0.0460
0.0392
0.0366
0.0142
0.0140
0.0
0.1
0.2
0.3
0.4
0.5
Importance (Absolute Coefficient Value)

Fig. 15.

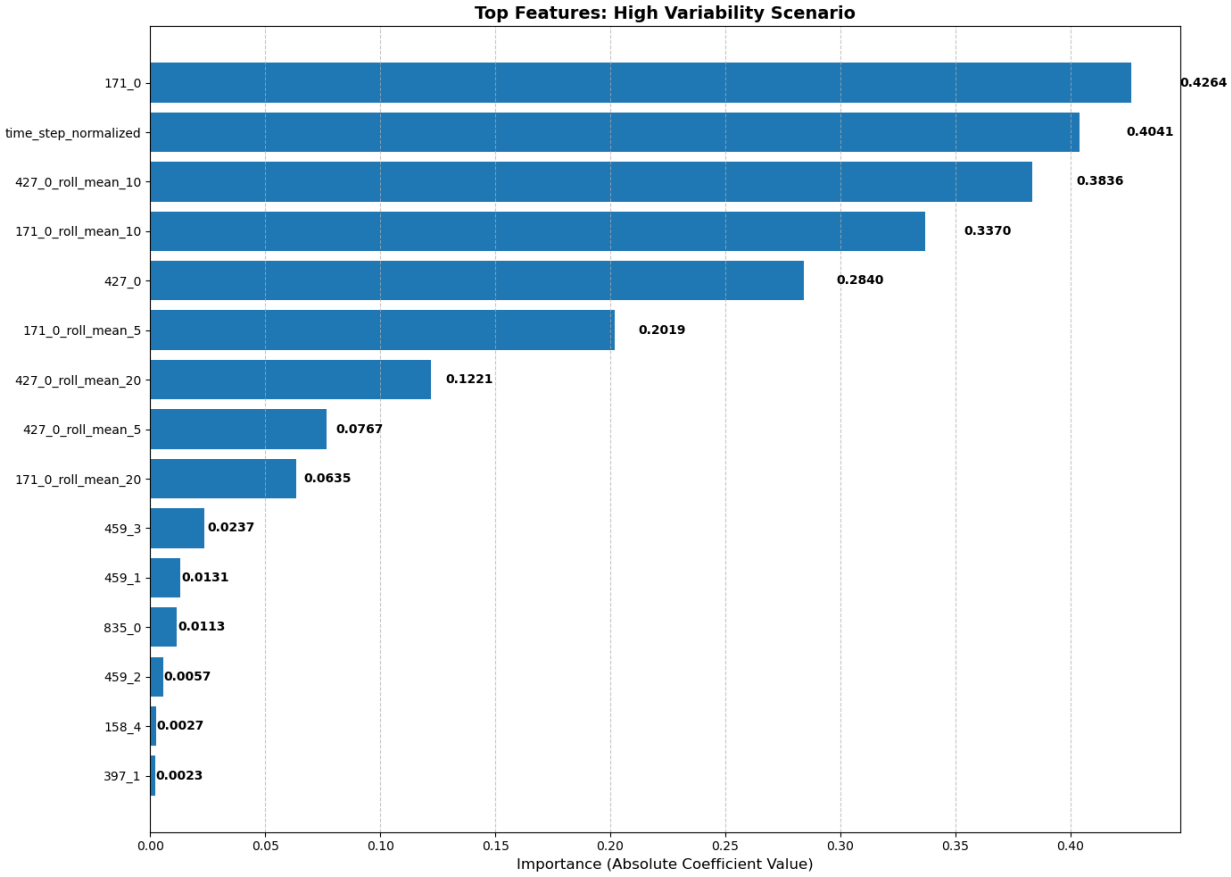
Top Features: High Variability Scenario
171_0
time_step_normalized
427_0_roll_mean_10
171_0_roll_mean_10
427_0
171_0_roll_mean_5
427_0_roll_mean_20
427_0_roll_mean_5
171_0_roll_mean_20
459_3
459_1
835_0
459_2
158_4
397_1
0.4264
0.4041
0.3836
0.3370
0.2840
0.2019
0.1221
0.0767
0.0635
0.0237
0.0131
0.0113
0.0057
0.0027
0.0023
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Importance (Absolute Coefficient Value)

Fig. 16.

REFERENCES

[1] Pillai, P., & Shin, K. G. (2001). Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. ACM SIGOPS Operating Systems Review.

[2] Herbert, S., et al. (2008). Analysis of Dynamic Voltage/Frequency Scaling in Chip-Multiprocessors. ACM SIGARCH Computer Architecture News.

[3] Baskiyar, S. & Abdel-Khader, R. (2010). Energy Aware DAG Scheduling on Heterogeneous Systems. Journal of Cluster Computing.

[4] Chronaki, K., et al. (2015). Criticality-Aware Dynamic Task Scheduling for Heterogeneous Architectures. ACM SIGPLAN Notices.

[5] Antoniadis, A., Ganje, P. J., Shahkarami, G. (2022). A novel prediction setup for online speed-scaling. In Czumaj, A. Xin, Q. (Eds.), *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2022, June 27-29, 2022, Tórshavn, Faroe Islands* (Vol. 227 of LIPIcs, pp. 9:1–9:20). https://doi.org/10.4230/LIPIcs.SWAT.2022.9

[6] Bamas, É., Maggiori, A., Rohwedder, L., Svensson, O. (2020). Learning augmented energy minimization via speed scaling. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., Lin, H.-T. (Eds.), *Advances in Neural Information Processing Systems 33 (NeurIPS 2020), December 6–12, 2020, virtual*. https://proceedings.neurips.cc/paper/2020/hash/af94ed0d6f5acc95f97170e3685f16c0-Abstract.html

[7] Jing, C., Madhavan, M., Mustafa, A., Pericàs, M. (2022). ERASE: Energy Efficient Task Mapping and Resource Management for Work Stealing Runtimes. *ACM Transactions on Architecture and Code Optimization*, 19(2).

[8] Digalwar, M., Gahukar, P., Mohan, S., Raveendran, B. K. (2015). STREAM: A Simulation Tool for Real-Time Energy-Aware Scheduling and Analysis for Multi-Core Processors. *Proceedings of the Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS)*.

[9] Google. (2019). Google Cluster Workload Traces 2019 [Dataset]. GitHub repository. https://github.com/google/cluster-data

[10] Balkanski, E., Daniel, L., Jayanti, A., Kakade, S. M., Lee, K. (2024). Energy-Efficient Scheduling with Lightweight Predictors: A Robustness Analysis. *arXiv preprint*, arXiv:2402.17143.

[11] Lindgren, T., et al. (2024). SCANIA Component X Dataset: A Real-World Multivariate Time Series Dataset for Predictive Maintenance. *Scania CV AB*. https://doi.org/10.5878/jvb5-d390

[12] Google. (2019). Google Cluster Workload Traces 2019 [Dataset]. GitHub repository. https://github.com/google/cluster-data

[13] Kaggle. (2022). Lending Club Loan Data [Dataset]. https://www.kaggle.com/datasets/wordsforthewise/lending-club

[14] Kaggle. (2022). Titanic - Machine Learning from Disaster [Dataset]. https://www.kaggle.com/c/titanic

[15] Fanaee-T, H., Gama, J. (2014). Bike Sharing Dataset. *UCI Machine Learning Repository*. https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset