

Scalability of Predictive Models on Multi-Core CPUs and GPUs: An Empirical Analysis

Atif Mahmood¹, Wan Joe Dean², P. Ganesh Kumar³, Adnan N. Qureshi⁴

Faculty of Data Science and Information Technology, INTI International University, Malaysia^{1,2}

Department of Computer Science and Engineering-College of Engineering Guindy, Anna University, Chennai-25, Tamil Nadu³

Faculty of Arts, Society and Professional Studies, Newman University, Birmingham, UK⁴

Abstract—The scalability of predictive models has become a critical factor in modern machine learning, as data volumes grow and computational resources diversify. This study presents an empirical benchmark of three widely used regression paradigms: Elastic Net, XGBoost, and Multi-Layer Perceptrons (MLPs). The *Obesity Estimation* dataset is used to evaluate both predictive performance and computational scalability across multi-core CPUs and GPUs. Unlike prior studies that primarily emphasize accuracy, we explicitly examine the trade-offs between accuracy, training time, and hardware efficiency. Models are evaluated under staged training loads (10–100% of data) with grid-searched hyperparameters (for Elastic Net and XGBoost) and regularized deep architectures (for MLP). Results demonstrate that while XGBoost achieves the highest predictive accuracy ($R^2 = 0.91$), it incurs significant computational overhead on CPUs, whereas GPU acceleration substantially improves its scalability. MLPs provide competitive accuracy ($R^2 = 0.87$) with an order-of-magnitude lower training time on GPUs, making them attractive for rapid or repeated retraining. Elastic Net offers interpretability and linear scalability on CPUs, but lags in predictive power. These findings provide practitioners with a decision framework: *XGBoost for maximum accuracy, MLPs for efficient retraining, and Elastic Net for interpretability and small-scale tasks*. More broadly, this work highlights that hardware selection is as important as algorithm choice, with GPUs serving as enablers of state-of-the-art performance on structured data.

Keywords—Scalability; XGBoost; Neural Networks; Elastic Net; resource efficiency

I. INTRODUCTION

The rapid expansion of data in the 21st century has firmly established machine learning as a cornerstone of innovation in science, industry, and society. The ability to extract predictive insights from large-scale datasets has enabled transformative applications ranging from medical diagnostics to financial modeling. However, this data-centric revolution also introduces a significant challenge: the scalability of computation. As datasets grow from megabytes to gigabytes and beyond, the computational time and resources required to develop increasingly complex predictive models become a critical research and development bottleneck [1], [2].

Selecting an appropriate machine learning model is, therefore, a multidimensional decision problem. While traditional approaches have primarily emphasized predictive accuracy as the benchmark of success, modern data science practice demands a broader evaluation framework. Factors such as training time, computational cost, interpretability, and ease of deployment are now equally important [3]. Consequently, researchers and practitioners are scrutinizing how different

model architectures perform under diverse computational and hardware constraints.

The computational landscape itself has undergone dramatic change. Historically, machine learning tasks were executed primarily on multi-core Central Processing Units (CPUs), which are optimized for sequential task execution. The emergence of General-Purpose Graphics Processing Units (GPGPUs), however, has redefined the paradigm. With thousands of parallel cores, GPUs are capable of delivering orders of magnitude improvements in speed, particularly for algorithms dominated by large-scale matrix and vector operations [4], [5]. This shift has sparked growing interest in systematically examining the interplay between algorithmic complexity, hardware efficiency, and scalability.

In this context, the study presents an empirical investigation of these trade-offs, focusing on three widely adopted model families that represent distinct paradigms of predictive modeling:

- Regularized Linear Models (Elastic Net): A strong and interpretable baseline that combines L1 and L2 regularization to address multicollinearity and feature selection. Its convex optimization formulation makes it computationally efficient on CPUs [6].
- Gradient Boosted Trees (XGBoost): A state-of-the-art ensemble method highly effective for structured and tabular data [7]. Despite its sequential tree-building nature, modern implementations exploit parallelization on both CPUs and GPUs for improved efficiency.
- Neural Networks (Multi-Layer Perceptrons): Foundational deep learning architectures capable of learning complex non-linear relationships [4]. Their reliance on matrix multiplications makes them particularly well-suited for GPU acceleration during training.

The purpose of this study is to provide a transparent and practical benchmark for data scientists and engineers using the publicly available Obesity Dataset [8]. We evaluate these three model families under identical preprocessing pipelines, comparing their predictive accuracy, computational cost, and scalability across CPU and GPU environments. Unlike prior studies that focus predominantly on accuracy or single-hardware evaluations, this work provides empirical evidence of the performance–scalability trade-offs and highlights the decisive role of hardware in model selection. In doing so, the study delivers actionable insights into how practitioners can balance accuracy, efficiency, and interpretability when

selecting the appropriate model–hardware configuration for real-world applications.

II. RELATED WORK

Comparative analysis of machine learning models has long been a cornerstone of the discipline. However, contemporary research is increasingly extending beyond accuracy-focused evaluations toward computational trade-offs, scalability, and interpretability. This section reviews key literature about the model families and hardware configurations that underpin the present study.

A. Foundations of Gradient Boosting

Gradient Boosting, first proposed by Friedman [9], revolutionized predictive modeling by demonstrating that a strong learner could be constructed through the sequential training of weak learners—typically decision trees—to correct predecessor errors. The introduction of XGBoost by Chen and Guestrin [7] marked a pivotal engineering advance, offering novel solutions to scalability challenges through efficient sparse-aware algorithms and block-based cache optimization for out-of-core computing. Their implementation further leveraged multi-core CPU parallelization to achieve significant performance gains.

Subsequent adaptations extended gradient boosting to GPU architectures, with Mitchell and Frank [10] demonstrating that the computational bottleneck in feature histogram construction could be massively parallelized via GPU processing. The result was the `gpu_hist` algorithm, which dramatically accelerated training times. Later, Chen et al. [11] conducted comprehensive comparisons between GPU-accelerated XGBoost and competing frameworks such as LightGBM [12] and CatBoost, concluding that GPU acceleration yields substantial benefits for large-scale datasets—particularly when the CPU–GPU communication overhead becomes negligible relative to computation.

B. Neural Networks for Tabular Data

Although deep learning, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [13], dominates unstructured data domains such as image and text analysis, its superiority over tree-based methods for structured tabular data remains a subject of debate. For years, ensemble tree methods like XGBoost were considered the de facto standard for such data types. However, recent work by Kadra et al. [14] challenges this assumption, demonstrating that even relatively simple Multi-Layer Perceptrons (MLPs) can rival or outperform gradient-boosted trees when properly tuned and regularized using techniques such as Batch Normalization and Dropout.

The scalability of neural networks is inherently linked to hardware acceleration. The matrix operations central to forward and backward propagation are highly parallelizable, making GPUs particularly well-suited for these workloads [15]. In their foundational work, Abadi et al. [16] detailed how frameworks such as TensorFlow were explicitly designed for heterogeneous, distributed GPU environments, allowing neural network training to scale across thousands of cores. Consequently, the performance disparity between CPU and GPU

implementations in neural network training remains among the largest in machine learning.

C. Linear Models and the Timelessness of Linear Modeling

Despite advances in complex architectures, regularized linear models continue to serve as a fundamental reference point in predictive modeling. Zou and Hastie [6] introduced the Elastic Net, which blends Lasso (L1) and Ridge (L2) penalties to manage correlated predictors while maintaining sparsity through feature selection. These models offer consistent interpretability and computational efficiency due to their convex optimization landscape, which ensures a unique global minimum and efficient convergence [17]. Implementations in widely used libraries such as `scikit-learn` have solidified their role as benchmark baselines. Nonetheless, as noted by Lu et al. [5], such models are not inherently optimized for GPU parallelization, limiting their scalability compared to deep learning counterparts.

D. Hardware Benchmarking Investigations

A growing body of research explicitly investigates the hardware-level benchmarking of machine learning algorithms. The RAPIDS cuML library, for example, provides GPU-accelerated analogs to many `scikit-learn` functions, demonstrating speedups ranging from $10\times$ to $100\times$ on large datasets [18]. Li et al. [19] and Reuther et al. [20] have compared various deep learning systems, reinforcing the dominant role of GPUs in accelerating model training and inference. Similarly, Reuther et al. [20] presented a comprehensive mapping of hardware acceleration strategies, highlighting that the optimal hardware configuration is model-dependent and sensitive to dataset size, model complexity, and memory overhead.

Building on these studies, our work contributes a focused, task-specific benchmarking analysis across three widely used and computationally distinct model paradigms—linear models, tree-based ensembles, and neural networks—under both CPU and GPU configurations. This approach provides a practical framework for understanding the trade-offs between computational efficiency, scalability, and predictive performance in tabular data applications.

E. Literature Summary

The literature demonstrates strong empirical evidence for XGBoost’s scalability across different architectures. Mitchell et al. [21] implemented a multi-GPU gradient boosting algorithm that processed 115 million training instances in under three minutes using end-to-end GPU parallelism with data compression techniques to optimize memory usage. Chen and Guestrin [7] established XGBoost as a scalable tree boosting system that scales beyond billions of examples using fewer resources than existing systems through novel sparsity-aware algorithms and cache optimization.

Somoye et al. [22] reported that distributed XGBoost achieved approximately a $3.5\times$ speedup, demonstrating practical scalability in distributed contexts. Edwards and Vishkin [23] showed potential for a $3.3\times$ speedup over NVIDIA’s most powerful GPU using hybrid memory architectures.

Research on neural network scalability reveals more complex performance patterns. Ma et al. [24] conducted comprehensive experiments showing GPUs outperformed parallel CPUs for synchronous SGD by only 2–5× for simple models and below 7× for fully connected deep networks. Notably, for asynchronous SGD, CPUs were the optimal solution, outperforming GPUs even when GPUs had 10× or more speedup potential. Barrachina et al. [25] analyzed CNN training scalability on GPU clusters, identifying crucial bottlenecks at both GPU and cluster levels. Farias et al. [26] achieved training speedups from one to four orders of magnitude for parallel MLP training by exploiting locality principles and parallelization capabilities. Yin et al. [27] addressed CPU scalability limitations through ParaX, which improved throughput by 1.73× to 2.93× on many-core CPUs by assigning instances to individual cores rather than entire CPUs.

Zhou et al. [28] provided significant evidence for Elastic Net scalability through algorithmic reduction to Support Vector Machines, creating a parallel solver that naturally utilizes GPUs and multi-core CPUs. Their implementation achieved identical results to the optimized `glmnet` but was up to two orders of magnitude faster across twelve real-world datasets.

Overall, the literature reveals that scalability performance is highly dependent on specific use cases, data characteristics, and computational approaches. While GPUs generally provide superior raw performance, CPUs remain competitive for certain scenarios, particularly asynchronous processing and when considering power efficiency. The choice between architectures should consider task-specific requirements, data sparsity, and synchronization needs rather than assuming GPU superiority across all contexts.

III. MAINTAINING THE INTEGRITY OF THE SPECIFICATIONS

A. Dataset Description

This study utilizes the *Estimation of Obesity Levels Based on Eating Habits and Physical Condition* dataset [8], publicly available through the UCI Machine Learning Repository. The dataset contains 2,111 individual records and 17 attributes, collected to characterize obesity levels among citizens of Mexico, Peru, and Colombia.

For this regression-based investigation, the objective variable is **Weight**, modeled as a continuous outcome predicted from the remaining 16 features. The original categorical class label, *NObeyesdad*, was excluded to avoid data leakage, since it is directly correlated with both *Weight* and *Height*.

The predictor variables consist of a combination of numerical and categorical features:

1) *Numerical (6)*: Age, Height, FCVC (frequency of vegetable consumption), NCP (number of main meals per day), CH2O (daily water intake), and FAF (frequency of physical activity).

2) *Categorical (10)*: Gender, family history with overweight, FAVC (frequent consumption of high-caloric food), CAEC (consumption between meals), SMOKE, SCC (calorie monitoring), CALC (alcohol consumption), MTRANS (transport mode), and two other minor lifestyle indicators.

B. Exploratory Data Analysis (EDA)

An exploratory assessment was performed to examine the structure, distribution, and integrity of the dataset prior to modeling.

1) *Numerical feature summary*: The numeric variables displayed realistic and consistent ranges. The mean participant age was 24.3 years (SD = 6.3). Heights ranged between 1.45 m and 1.98 m, while the target variable (*Weight*) spanned 39 kg to 173 kg. No missing or anomalous entries were detected, simplifying preprocessing. A correlation matrix revealed a strong positive correlation between *Height* and *Weight*, as expected, confirming *Height* as a dominant predictor.

2) *Categorical feature distribution*: The categorical variables provide valuable lifestyle context. For instance, 81.9% of respondents reported a family history of overweight, indicating a significant genetic or environmental predisposition. Likewise, 74.9% reported using public transportation, suggesting an urban or suburban sample profile. Recognizing such imbalances is critical, as they may influence how models learn latent patterns.

Overall, the EDA confirmed that the dataset was suitable for regression analysis and rich in both continuous and discrete predictors, warranting specialized preprocessing techniques.

3) *Preprocessing pipeline*: To ensure data consistency, prevent leakage, and tailor transformations to model requirements, a comprehensive preprocessing pipeline was implemented using the `Scikit-learn` `Pipeline` and `ColumnTransformer` objects [17]. This approach encapsulated all transformations while ensuring that training-set statistics (e.g., means, standard deviations) were applied consistently to the test data.

The pipeline workflow is summarized as follows:

- **Train-Test Split**: The dataset was divided into 80% training ($n = 1,688$) and 20% testing ($n = 423$) subsets using a fixed random state (`random_state = 42`) for reproducibility. Stratification was applied to maintain proportional class distributions, and the held-out test set was used exclusively for final evaluation.
- **Feature Transformation**: The `ColumnTransformer` applied data-type-specific transformations:
 - **Numerical Features**: Standardization was performed using `StandardScaler`, which centers features to zero mean and unit variance according to:

$$z = \frac{x - \mu}{\sigma}$$

For Elastic Net regression, scaling ensures that the regularization penalty is applied uniformly across coefficients, regardless of magnitude. For MLP neural networks, scaling improves gradient-descent efficiency and stabilizes convergence.

- **Categorical Features**: `OneHotEncoder` was used to encode the ten categorical features into

binary indicator variables. This prevents the model from assuming any false ordinal relationship among categories (e.g., “Automobile” < “Motorbike” < “Walking”). The parameter `handle_unknown = "ignore"` safeguarded against unseen categories during inference.

- **Model-Specific Augmentation:** For Elastic Net regression, a `PolynomialFeatures` transformer (degree = 2) was appended to generate interaction (e.g., $Age \times Height$) and quadratic (e.g., Age^2) terms. This manual feature engineering step enriches the model’s expressiveness by allowing it to capture non-linear relationships within an otherwise linear framework.

This structured pipeline ensures that every model receives clean, standardized, and consistently formatted input, thereby maintaining a fair comparison across algorithmic families while upholding data integrity and reproducibility.

IV. METHODOLOGY

To evaluate performance and scalability, three distinct models were compared under identical preprocessing conditions. The experiments were conducted in a Google Colab environment, utilizing both its standard multi-core CPU and NVIDIA T4 GPU configurations to evaluate performance differentials.

The methodology encompasses comparative evaluation across:

- **Regularized Linear Model:** Elastic Net Regression [6], providing an interpretable baseline with controlled feature shrinkage.
- **Gradient Boosted Trees:** XGBoost [7], representing state-of-the-art ensemble learning for tabular datasets.
- **Neural Network:** Multi-Layer Perceptron (MLP) [14], trained using backpropagation and stochastic gradient descent, designed to assess non-linear representation learning efficiency.

All models were evaluated for predictive performance, computational cost, and scalability across CPU and GPU hardware environments, establishing a rigorous foundation for subsequent analysis.

A. Models

We compared three different paradigms of machine learning models, presenting three distinct approaches.

The first model is a regularized linear regression model, known as the **Elastic Net**. It is an extension of standard Ordinary Least Squares (OLS) regression with the addition of a penalty term to the cost function that limits the size of the learned coefficients. The purpose of this regularization is to prevent overfitting and improve the model’s performance on unseen data. The Elastic Net combines the two most common types of regularization—Lasso (L1) and Ridge (L2)—to balance sparsity and stability. The Elastic Net model aims to minimize the following cost function:

$$\hat{\beta} = \arg \min_{\beta} \left(\frac{1}{2n} \sum_{i=1}^n (y_i - X_i \beta)^2 + \lambda \left[\alpha \|\beta\|_1 + \frac{1-\alpha}{2} \|\beta\|_2^2 \right] \right)$$

where, λ controls the overall strength of regularization, and α controls the trade-off between L1 and L2 penalties.

The second model is **XGBoost**, a highly efficient and scalable decision tree-based gradient-boosted implementation. Gradient boosting is an ensemble technique that builds models sequentially—each new tree is trained to correct the residual errors made by the previous ensemble. The objective function at boosting iteration t is defined as:

$$Obj^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

where, l represents the loss function that measures the difference between predicted and true values, f_t is the new tree added at iteration t , and $\Omega(f_t)$ is the regularization term that penalizes model complexity to prevent overfitting.

The last model used in the analysis was a **Multi-Layer Perceptron (MLP)**, which is an archetypal feedforward neural network. Each neuron in a layer receives a weighted input from the previous layer and adds a bias term. The resulting value is passed through a non-linear activation function to introduce learning flexibility. The forward propagation step for a neuron in layer (l) is expressed as:

$$a^{(l)} = f \left(W^{(l)} a^{(l-1)} + b^{(l)} \right)$$

where, $W^{(l)}$ represents the weight matrix, $b^{(l)}$ is the bias vector, $a^{(l-1)}$ is the input from the previous layer, and $f(\cdot)$ denotes the activation function (e.g., ReLU or sigmoid).

In this study, we implemented a regression-oriented MLP architecture with one input layer, two hidden layers, and a single output neuron. The two hidden layers consisted of 128 and 64 neurons, respectively, while the output neuron was configured to perform continuous-value prediction.

B. Experimental Protocol

Our experimental protocol was designed to evaluate both the computational scalability and predictive performance of each model in a rigorous and reproducible manner. To obtain scalability measurements, each model was trained on progressively larger random subsets of the training data—specifically at fractions of 10%, 25%, 50%, 75%, and 100%. This staged approach enabled the construction of an empirical scaling curve that clearly demonstrates how the computational time requirements evolve for each algorithm as the data volume increases.

For each training run, the *wall-clock time* was recorded to quantify efficiency. All experiments used fixed random seeds to ensure that the results for deterministic models (Elastic Net and XGBoost) were fully reproducible.

After scalability testing, the final optimized versions of all models were trained on 100% of the training data to assess

their maximum predictive potential. These fully trained models were then evaluated on the held-out test data, which were never used during the training or model selection process. The final evaluation employed three standard regression metrics that together provide a comprehensive performance assessment:

- Root Mean Squared Error (RMSE): Sensitive to large errors, RMSE quantifies the standard deviation of the residuals, providing insight into prediction deviation magnitude.
- Mean Absolute Error (MAE): A robust measure of average prediction error magnitude, less sensitive to outliers and easy to interpret.
- Coefficient of Determination (R^2): Measures the proportion of variance in the target variable explained by the model, providing an indicator of goodness-of-fit.

C. Hyperparameter Configuration

A systematic hyperparameter optimization was conducted using `GridSearchCV` combined with 5-fold cross-validation to determine the optimal configuration for each model. This ensured a fair and exhaustive evaluation across different subsets of the data, balancing bias and variance in performance.

1) *Elastic Net*: The Elastic Net regressor was tuned over the following parameter grid:

- alpha: [0.01, 0.05, 0.1, 0.5] (Controls overall strength of regularization)
- l1_ratio: [0.7, 0.9, 0.95, 1.0] (Controls the trade-off between L1 and L2 penalties)

2) *XGBoost*: The XGBoost model underwent hyperparameter tuning with the following configuration space:

- n_estimators: [500, 700] (Number of boosting trees)
- learning_rate: [0.03, 0.05] (Step size shrinkage to prevent overfitting)
- max_depth: [7, 9] (Maximum depth of individual trees)
- subsample: [0.7, 0.8] (Fraction of samples used to grow each tree)
- colsample_bytree: [0.7, 0.8] (Fraction of features used per tree)

To enable GPU acceleration, the `tree_method` parameter was set to `gpu_hist`.

3) *Multi-Layer Perceptron (MLP)*: The MLP model was not tuned through exhaustive grid search but was configured following widely accepted deep learning best practices, emphasizing regularization and convergence stability.

Architecture:

- Input Layer
- Dense (256, activation='relu') + BatchNormalization + Dropout(0.4)
- Dense (128, activation='relu') + BatchNormalization + Dropout(0.4)

- Dense (64, activation='relu') + BatchNormalization
- Dense (1, activation='linear')

Training Configuration:

- Optimizer: Adam (initial learning rate = 0.001)
- Epochs: 300 (with `EarlyStopping` patience = 20)
- Callbacks: `EarlyStopping`, `ReduceLROnPlateau`

This configuration was selected to ensure stable convergence, reduce overfitting, and maintain balanced generalization performance across different computational environments.

V. RESULTS AND EVALUATION

This section presents the quantitative analysis of the conducted experiments, summarizing the results, interpretability insights, and detailed scalability observations derived from the comparative evaluation of the models.

A. Evaluation Metrics

The performance of each model was evaluated on the held-out test dataset using three standard regression metrics that collectively provide a comprehensive view of predictive accuracy and reliability:

- Coefficient of Determination (R^2): Measures the proportion of variance in the dependent variable explained by the independent variables. Values closer to 1.0 indicate a better model fit.
- Mean Absolute Error (MAE): Represents the average absolute difference between predicted and actual values, providing an interpretable measure of average prediction error in the same units as the target variable (kg).
- Root Mean Squared Error (RMSE): The square root of the mean of squared errors, which penalizes large deviations more heavily than MAE, emphasizing the model's sensitivity to large errors.

B. Compared Performance

Following hyperparameter optimization, the final tuned versions of all models were evaluated on the held-out test dataset. As summarized in Table I, each algorithm demonstrates distinct strengths when balancing predictive accuracy, computational efficiency, and scalability. The performance metrics and total training times, measured on the complete training data after model tuning, highlight these trade-offs—showing that while some models achieve superior accuracy, others offer faster convergence and better scalability under GPU or CPU configurations.

The findings indicate a clear performance hierarchy among the evaluated models. As illustrated in Fig. 1, **XGBoost** achieved the highest predictive accuracy with an impressive R^2 value of 0.9097, demonstrating its strong capability to capture complex non-linear relationships within the data. The **MLP** model followed closely, attaining an R^2 of 0.8672,

TABLE I. COMPARATIVE MODEL PERFORMANCE ON THE TEST SET

Model	RMSE	MAE	R^2	Time (s)
XGBoost (GPU)	7.9811	5.0612	0.9097	361.43
MLP (GPU)	9.6749	6.9852	0.8672	24.11
Elastic Net (CPU)	12.2405	9.4815	0.7875	36.29

which validates its effectiveness as a non-linear function approximator. In contrast, the **Elastic Net** model, despite the inclusion of polynomial features, produced the lowest R^2 value of 0.7875. This outcome suggests that the relationships within the dataset were too intricate to be accurately represented using a regularized linear approach.

In terms of computational efficiency, Fig. 2 provides a comparative visualization of the models across all performance metrics— R^2 , RMSE, and MAE. The **MLP** demonstrated the fastest training time, completing in only 24.1 seconds. This superior speed can be attributed to its GPU-accelerated implementation and the use of the `EarlyStopping` callback, which effectively reduced unnecessary training epochs. Conversely, **XGBoost**, while achieving the highest accuracy, required a longer training duration due to its extensive hyperparameter search space and more complex ensemble architecture, resulting in a higher computational cost.

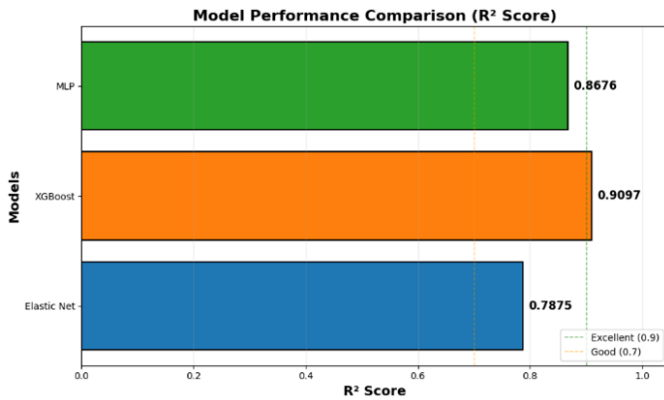


Fig. 1. Model performance comparison R^2 .

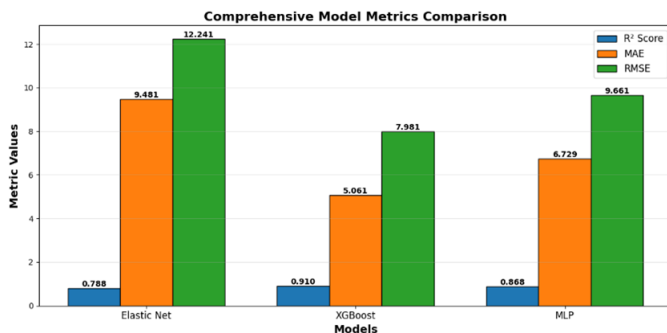


Fig. 2. Comprehensive model metrics comparison.

C. Analysis of Error and Interpretability

To gain a deeper understanding beyond aggregate metrics, an additional diagnostic analysis was conducted to examine model residuals and interpretability.

1) *Residual analysis*: Residual plots serve as essential diagnostic tools in regression analysis, visualizing the discrepancy between actual and predicted values. The residuals, defined as $(y_i - \hat{y}_i)$, are plotted against the predicted values to reveal systematic model behavior. In a well-calibrated model, residuals are expected to be randomly distributed around the horizontal line at zero—forming a cloud-like pattern with no discernible structure. Such a distribution indicates that the model has effectively captured the underlying data signal and that remaining deviations are primarily due to random noise rather than systematic bias.

The random scatter of points around the $y = 0$ line indicates a well-fitted model with no systematic error patterns. When examining the residual plot of our **XGBoost** model with the best performance Fig. 3, an ideal behavior is observed. The residuals are well distributed both above and below the $y = 0$ line across the entire range of predicted weights. No visible curvature is present, suggesting the absence of non-linear relationships that the model failed to capture, and no funnel-like pattern (heteroscedasticity) appears, implying that the model’s error variance remains consistent across different magnitudes of prediction. The random scatter provides strong evidence that the model is unbiased, the residuals are homoscedastic, and thus the predictive validity of the model can be considered reliable.

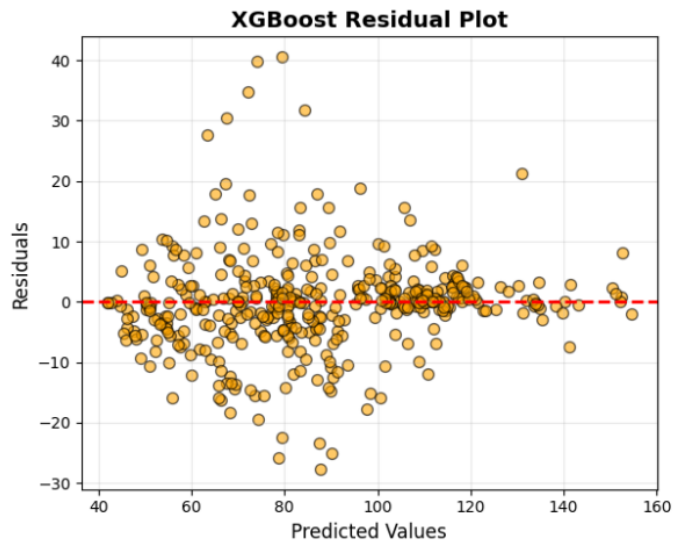


Fig. 3. Residual plot for the XGBoost model on the test set.

2) *Feature importance*: To further interpret the strong performance of **XGBoost**, its internal feature importance scores were examined. These scores quantify the contribution of each feature to the boosted trees. The most influential features are presented in Fig. 4, where the top fifteen attributes exhibit the greatest impact on model performance.

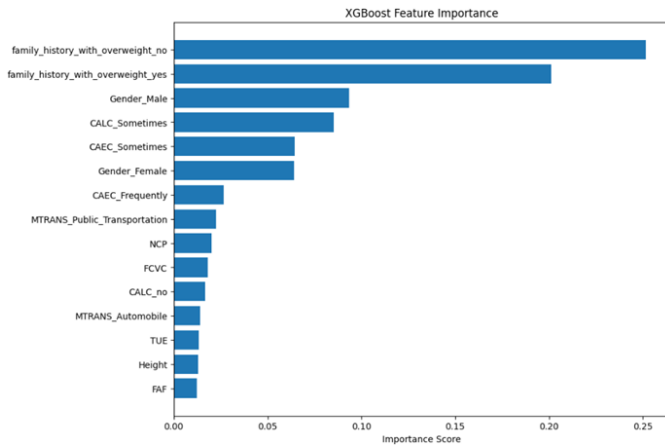


Fig. 4. Scores of feature importance of the most tuned XGBoost model.

The scores are used to show the relative contribution of each feature to the model. Height is, as expected, the most predictive of all features of being overweight. This finding is consistent with established biological principles. There is also a high rank of gender and family history with overweight, which endorses the fact that the demographic and genetic factors are the driving forces. Such lifestyle factors as FAVC (consumption of high-caloric food many times) and FCVC (consumption of vegetables many times) are also reported to be significant predictors. This provides assurance that the model has learned meaningful relationships within the data.

D. Scalability Analysis

The core objective of this study is the analysis of scalability, which quantifies the dependency of the time spent in training models on the size of a data set to be considered with each model and hardware setup. The number of training samples exhibits a near-linear increase in training time. Fig. 5 given below shows that the Elastic Net model exhibits linear and predictable scalability on the CPU. The training time grows smoothly with the amount of data added to it, so it should be a choice when dealing with smaller to medium datasets on typical hardware.

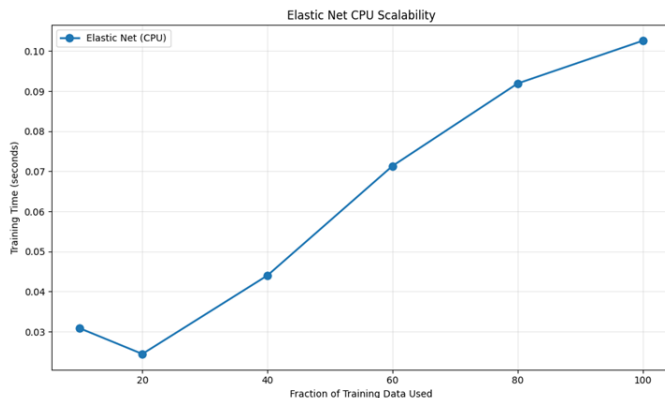


Fig. 5. Elastic Net model is scalable on a multi-core processor.

The logarithmic scale illustrates the widening performance gap between CPU and GPU computations as data volume

increases. While GPU training times remain nearly constant, CPU times exhibit exponential growth. The strongest evidence for the influence of hardware acceleration is depicted in Fig. 6 and Fig. 7.

1) *CPU performance:* For both XGBoost and the MLP, training time exhibits super-linear—nearly exponential—growth when executed on the CPU. This demonstrates that as data volume increases, these sophisticated algorithms quickly become computationally infeasible on a purely CPU-based system.

2) *GPU performance:* In contrast, both models exhibit remarkable scalability when trained on a GPU. The parallel architecture of GPUs allows massive data batches (as in MLP) and feature histogram construction (as in XGBoost) to be processed with impressive efficiency. The scaling curve of the MLP, in particular, remains nearly flat, as the entire dataset can be processed within seconds. For XGBoost, GPU benefits increase further with larger datasets, since the initial overhead of data transfer becomes negligible compared to the computational gain.

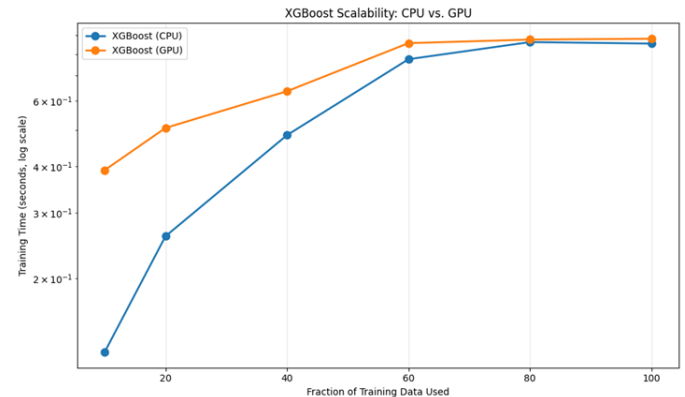


Fig. 6. XGBoost model on CPU vs. GPU. First y-axis is a logarithmic scale.

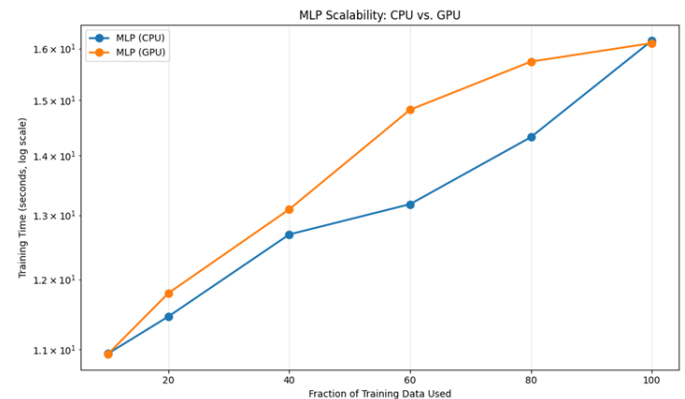


Fig. 7. Scalability of the MLP model between CPU and GPU.

VI. DISCUSSION

The experimental results present a multifaceted view of the trade-offs among accuracy, computation, and interpretability in

contemporary machine learning. This section synthesizes the key findings to derive practical insights and recommendations.

A. Accuracy–Compute Trade-off

The results reveal a classical trade-off between accuracy and computational cost. XGBoost achieved the highest accuracy with an R^2 of 0.91 but was also the most computationally expensive to train. Its exhaustive hyperparameter search—despite parallelization—demanded substantial time resources. The MLP achieved slightly lower accuracy ($R^2 = 0.87$) but was an order of magnitude faster in training.

This presents a strategic dilemma for practitioners: whether a 4–5% improvement in R^2 justifies a fourteenfold increase in training time. In static, one-time prediction tasks where model accuracy is paramount, XGBoost’s performance merits its computational cost. However, in dynamic environments with frequent retraining (e.g., streaming or production systems), the rapid training time of the MLP makes it a more practical and cost-efficient choice. The Elastic Net, though least accurate, serves as a valuable baseline due to its simplicity, speed, and interpretability, making it ideal for initial experimentation before committing to complex models.

B. The Hardware Acceleration Imperative

Fig. 6 and Fig. 7 (Scalability plots) represent one of the most significant contributions of this study. They empirically confirm that GPU acceleration is not merely advantageous but essential for scalable machine learning involving modern algorithms. The performance difference between CPU and GPU training is not incremental—it is transformative.

Without GPU acceleration, training times for MLP and XGBoost increase almost exponentially, severely restricting feasible dataset sizes and hyperparameter search depth. In contrast, GPUs enable researchers to train on larger datasets, execute more exhaustive tuning, and achieve optimal performance within seconds rather than minutes. Consequently, hardware acceleration emerges as a cornerstone for state-of-the-art machine learning research and practice.

C. Interpretability vs. Performance

The findings also reaffirm the long-standing trade-off between interpretability and predictive power. The Elastic Net functions as a white-box model—its coefficients are directly interpretable, allowing explicit inference on how each feature contributes to the prediction outcome. This makes it ideal for use cases requiring transparency and causal interpretation.

XGBoost occupies a middle ground. Although individual trees are complex, the model offers robust feature importance scores (see Fig. 4), which provide insight into the relative influence of predictors—a level of interpretability sufficient for most applied contexts.

The MLP, however, is a classical black-box model. Its deep, nested non-linear transformations make it extremely difficult to map inputs to outputs directly. While explainability methods such as SHAP (SHapley Additive exPlanations) can approximate local feature contributions, the model itself lacks intrinsic interpretability. In this study, moving from the most interpretable model (Elastic Net) to the most accurate model

(XGBoost) resulted in a 12% improvement in R^2 . This suggests that, in this context, the gain in performance outweighs the loss in interpretability—a trade-off often encountered in applied machine learning.

D. Limitations and Ethical Considerations

This study acknowledges several limitations. First, it relies on a single dataset; thus, generalization to datasets with different structures, dimensionalities, or feature distributions remains to be validated. Second, the hyperparameter optimization relied on grid search; more advanced techniques such as Bayesian optimization could yield superior configurations.

From an ethical standpoint, predictive models involving personal attributes (such as weight or family history) may inadvertently learn and propagate social biases. In applications like healthcare or insurance, such biases could lead to unfair outcomes for individuals based on immutable characteristics. Therefore, any deployment of such models should be accompanied by a rigorous bias and fairness audit to ensure equitable performance across demographic groups.

VII. CONCLUSION AND FUTURE WORK

This empirical study explored the relationship between predictive accuracy, computational scalability, and hardware selection across three distinct machine learning paradigms—Elastic Net, XGBoost, and Multi-Layer Perceptron—using the Obesity dataset. Our results clearly establish that for structured, tabular data, XGBoost remains the state-of-the-art in predictive strength due to its ability to capture complex feature interactions. However, a well-optimized MLP, combined with GPU acceleration, can achieve competitive accuracy while offering dramatically faster training throughput—making it ideal for iterative or time-sensitive applications.

Most importantly, the findings highlight the transformative impact of hardware acceleration. As data volumes and model complexities continue to rise, GPU-enabled computation transitions from an optional enhancement to a fundamental requirement. The scalability analysis shows that, while CPU-based training becomes prohibitively slow with large data, GPU-based models maintain efficiency even under full dataset loads. Ultimately, model selection emerges as a multi-objective optimization problem—balancing accuracy, interpretability, training time, and computational resources. Future research should expand this benchmark to include other ensemble frameworks such as LightGBM and CatBoost, and extend interpretability analysis using model-agnostic explainers like SHAP for both global and local feature insights. Moreover, testing on higher-dimensional datasets will further clarify the boundaries of scalability and the evolving role of hardware in machine learning.

AUTHORS’ CONTRIBUTION

Wan Joe Dean served as the principal contributor to this study under the supervision of Atif Mahmood for the Machine Learning Applications course. The co-authors, Adnan Qureshi, Saaidal Razalli, and Qilong Yu, contributed by reviewing the manuscript, assisting in writing, and verifying experimental results.

REFERENCES

- [1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and challenges," *Science*, vol. 349, no. 6245, pp. 255–260, 2015. [Online]. Available: <https://doi.org/10.1126/science.aaa8415>
- [2] D. A. Dewi, H. K. R. Singh, J. Periasamy, T. B. Kurniawan, H. Henderi, and M. S. Hasibuan, "Scalable machine learning approaches for real-time anomaly and outlier detection in streaming environments," *Journal of Applied Data Sciences*, vol. 5, no. 4, pp. 1949–1962, 2024.
- [3] J. Dean, "The deep learning revolution and its implications for computer architecture and chip design," in *Proc. IEEE Intl. Solid-State Circuits Conf. (ISSCC)*, 2020. [Online]. Available: <https://doi.org/10.1109/ISSCC19947.2020.9063094>
- [4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015. [Online]. Available: <https://doi.org/10.1038/nature14539>
- [5] Z. Lu *et al.*, "Gpu-accelerated machine learning: a survey," *IEEE Access*, vol. 8, pp. 179 536–179 553, 2020. [Online]. Available: <https://doi.org/10.1109/ACCESS.2020.3028308>
- [6] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B*, vol. 67, no. 2, pp. 301–320, 2005. [Online]. Available: <https://doi.org/10.1111/j.1467-9868.2005.00503.x>
- [7] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2016, pp. 785–794. [Online]. Available: <https://doi.org/10.1145/2939672.2939785>
- [8] F. Palechor and A. de la Hoz Manotas, "Dataset for estimation of obesity levels based on eating habits and physical condition," *Data in Brief*, vol. 25, 2019.
- [9] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Online]. Available: <https://doi.org/10.1214/aos/1013203451>
- [10] R. Mitchell and E. Frank, "Accelerating xgboost with gpus," in *Proc. GPU Technology Conference*, 2017.
- [11] T. Chen, H. He, M. Ben-Shimon *et al.*, "A practical guide to gpu-accelerated gradient boosting," *ACM SIGKDD Explorations Newsletter*, vol. 21, no. 2, pp. 1–12, 2020. [Online]. Available: <https://doi.org/10.1145/3373464.3373466>
- [12] G. Ke, Q. Meng, T. Finley *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2017. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html>
- [13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: <https://doi.org/10.1162/neco.1997.9.8.1735>
- [14] A. Kadra, M. Lindauer, F. Hutter *et al.*, "Well-tuned simple nets excel on tabular data," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2021. [Online]. Available: <https://proceedings.neurips.cc/paper/2021/hash/24b424052fa619583786195462572111-Abstract.html>
- [15] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th ACM/IEEE Intl. Symp. on Computer Architecture (ISCA)*, 2017. [Online]. Available: <https://doi.org/10.1145/3079856.3079868>
- [16] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016. [Online]. Available: <https://arxiv.org/abs/1603.04467>
- [17] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: <http://www.jmlr.org/papers/v12/pedregosa11a.html>
- [18] M. Dixon *et al.*, "Rapids: Open gpu data science," in *Proc. 19th Python in Science Conference*, 2020.
- [19] S. Li *et al.*, "A survey of benchmarking deep learning frameworks," *Journal of Computer Science and Technology*, vol. 35, pp. 117–133, 2020.
- [20] A. Reuther *et al.*, "A survey on hardware accelerators for deep neural networks," *Proceedings of the IEEE*, vol. 108, no. 6, pp. 811–839, 2020.
- [21] R. Mitchell, A. Adinets, T. Rao, and E. Frank, "Xgboost: Scalable gpu accelerated learning," *arXiv preprint arXiv:1806.11248*, 2018.
- [22] O. I. Somoye *et al.*, "Scalable machine learning algorithms for processing high-dimensional data: A case study approach," *Asian Journal of Advanced Research and Reports*, 2025.
- [23] J. A. Edwards and U. Vishkin, "Linking parallel algorithmic thinking to many-core memory systems and speedups for boosted decision trees," in *Proc. International Symposium on Memory Systems*, 2018.
- [24] Y. Ma *et al.*, "Stochastic gradient descent on modern hardware: Multi-core cpu or gpu? synchronous or asynchronous?" in *Proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2019, pp. 1063–1072.
- [25] S. Barrachina *et al.*, "Using machine learning to model the training scalability of convolutional neural networks on clusters of gpus," *Computing*, vol. 105, no. 5, pp. 915–934, 2023.
- [26] F. C. Farias *et al.*, "Embarrassingly parallel independent training of multi-layer perceptrons with heterogeneous architectures," *arXiv preprint arXiv:2206.08369*, 2022.
- [27] L. Yin *et al.*, "Parax: Boosting deep learning for big data analytics on many-core cpus," in *Proc. VLDB Endowment*, vol. 14, 2021, pp. 864–877.
- [28] Q. Zhou *et al.*, "A reduction of the elastic net to support vector machines with an application to gpu computing," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.