VectorForgeML: A High-Performance R-C++ Machine Learning Framework with Optimized Linear Algebra Backend

Mohd Musheer
VectorForgeML Research Initiative
Amravati, Maharashtra, India
Email: musheerayan@email.com

***Abstract*—VectorForgeML is a high-performance machine learning framework designed to bridge the gap between R-level usability and low-level C++ computational efficiency. While existing R ecosystems often suffer from memory duplication overhead and interpreter latency, VectorForgeML implements core learning algorithms directly in C++ from first principles and leverages BLAS/LAPACK routines for optimized numerical computation. The framework supports supervised and unsupervised learning including regression, classification, clustering, and ensemble models. Experimental evaluation across multiple real-world datasets demonstrates competitive predictive performance and significant training-time improvements compared to native R frameworks such as tidymodels, while approaching the performance of Python's scikit-learn. This work highlights the feasibility of combining high-level statistical APIs with system-level optimization in a unified R-native machine learning framework.**

***Index Terms*—Machine Learning, Rcpp, High Performance Computing, BLAS, C++, Optimization**

I. Introduction

Modern machine learning systems must balance accessibility with computational efficiency. While Python libraries such as scikit-learn expose user-friendly APIs backed by optimized C extensions, R-native ecosystems often rely on layered abstractions that introduce overhead during iterative model training.

VectorForgeML addresses this limitation by implementing core algorithms directly in C++ and exposing them through a zero-copy Rcpp interface. By avoiding redundant memory allocation and leveraging hardware-optimized BLAS/LAPACK routines, the framework reduces interpreter-induced latency while maintaining R's statistical expressiveness.

II. Related Work

Scikit-learn is the industry standard for classical machine learning in Python, leveraging NumPy and SciPy for optimized computation. Tidymodels offers a unified modeling interface in R but relies on multiple underlying packages that may lack consistent low-level optimization. Libraries such as XGBoost and LightGBM achieve high efficiency for tree-based models but remain algorithm-specific.

VectorForgeML provides a generalized, multi-algorithm backend for R with direct system-level control over memory and computation.

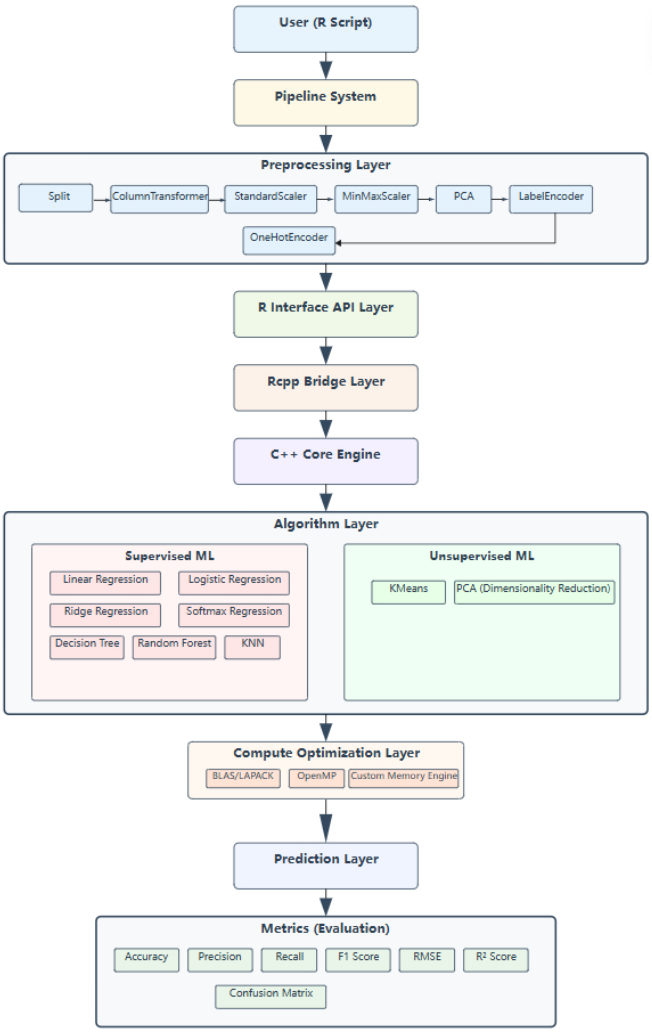

Fig. 1. Layered architecture of VectorForgeML.

III. System Architecture

The architecture follows a layered pipeline:

1) **R Interface:** User-defined preprocessing and modeling pipeline.

2) **Rcpp Bridge:** Zero-copy boundary enabling direct pointer-based matrix transfer.
3) **C++ Core Engine:** Algorithm implementations with manual memory management.
4) **BLAS/LAPACK Backend:** Optimized dense linear algebra routines with thread control.

Tree structures are allocated using raw pointers to minimize object overhead and improve cache-friendly traversal.

IV. Algorithmic Implementations

A. Linear Regression

Linear Regression minimizes Mean Squared Error:

$$\min_{\beta} ||y - X\beta||^2$$

Closed-form solution:

$$\beta = (X^T X)^{-1} X^T y$$

The system computes $X^T X$ using BLAS `dgemm` and solves via LAPACK Cholesky decomposition (`dposv`).

Time Complexity:

$$O(np^2 + p^3)$$

B. Ridge Regression

Ridge Regression introduces L2 regularization:

$$\min_{\beta} ||y - X\beta||^2 + \lambda||\beta||^2$$

Resulting normal equation:

$$(X^T X + \lambda I)\beta = X^T y$$

Time Complexity:

$$O(np^2 + p^3)$$

C. Logistic Regression

Binary Logistic Regression uses the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Parameters are optimized via batch gradient descent.

Time Complexity:

$$O(np \cdot \text{epochs})$$

D. Softmax Regression

For multiclass classification:

$$P(y = k|x) = \frac{e^{w_k^T x}}{\sum_j e^{w_j^T x}}$$

To ensure numerical stability, the implementation applies a log-sum-exp stabilization technique before exponentiation.

Time Complexity:

$$O(npk \cdot \text{epochs})$$

E. Decision Tree

Decision Trees perform exhaustive binary splitting to minimize variance (regression) or Gini impurity (classification):

$$\min_{j,s} \sum_{x_i \in R_1} (y_i - c_1)^2 + \sum_{x_i \in R_2} (y_i - c_2)^2$$

The quadratic dependence arises from evaluating all candidate thresholds per feature.

Time Complexity:

$$O(pn^2)$$

F. Random Forest

Random Forest constructs T bootstrapped trees with feature subsampling:

$$\hat{f}(x) = \frac{1}{T} \sum_{t=1}^{T} f_t(x)$$

Time Complexity:

$$O(T \cdot m_{try} \cdot n^2)$$

G. K-Means

K-Means minimizes within-cluster variance:

$$\min_{C} \sum_{i=1}^{n} \min_{k} ||x_i - \mu_k||^2$$

Time Complexity:

$$O(npk \cdot \text{iterations})$$

H. Principal Component Analysis (PCA)

PCA computes the covariance matrix:

$$C = \frac{1}{n-1} X^T X$$

Eigen decomposition:

$$C = V \Lambda V^T$$

Time Complexity:

$$O(p^3)$$

V. Experimental Setup

Experiments were conducted in a controlled Kaggle CPU environment with fixed random seeds for reproducibility. The framework was compared with scikit-learn and tidymodels across four datasets:

- Heart Disease (102,500 samples)
- Cars MSRP (22,307 samples)
- Wine Quality (1,599 samples)
- Titanic (891 samples)

TABLE I
COMPREHENSIVE PERFORMANCE BENCHMARKING ACROSS FRAMEWORKS

Algorithm	Dataset	Samples	Accuracy / R^2			Training Time (s)		
			VF-ML	sklearn	tidymodels	VF-ML	sklearn	tidymodels
Logistic Regression	Heart Disease	102,500	0.846	0.855	**0.869**	0.633	0.185	0.116
Linear Regression	Cars (MSRP)	22,307	0.656	0.656	**0.658**	**0.266**	0.082	0.558
Ridge Regression	Cars (MSRP)	22,307	0.656	0.656	**0.658**	**0.049**	0.039	0.752
Softmax Regression	Wine Quality	1,599	**0.631**	0.575	0.561	**0.100**	0.034	0.155
Decision Tree	Wine Quality	1,599	**0.622**	0.575	0.542	0.510	0.010	0.176
Random Forest	Wine Quality	1,599	**0.694**	0.588	0.660	0.387	0.265	0.179
K-Means	Wine Quality	1,599	–	–	–	**0.010**	0.089	0.028
KNN (Classification)	Titanic	891	0.637	0.797	**0.816**	**0.022**	0.006	0.114

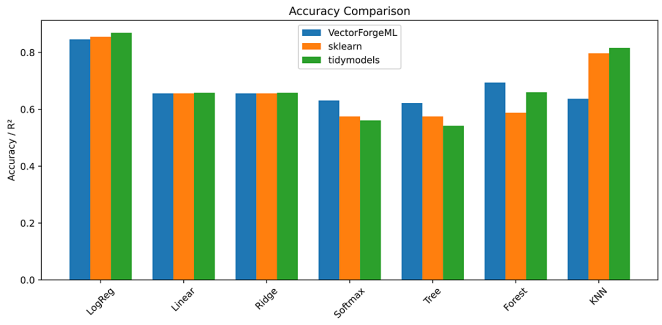

Fig. 2. Visual comparison of model accuracy across the three frameworks.

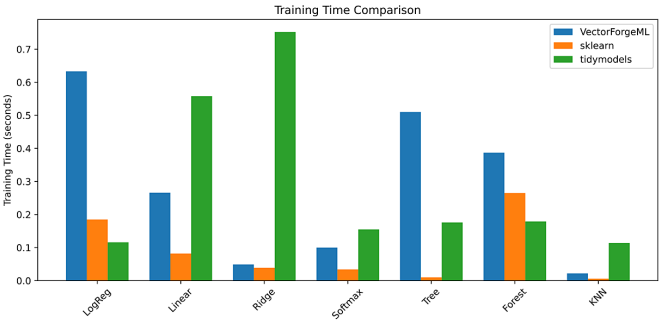

Fig. 3. Comparison of training time (in seconds). Lower is better.

VI. BENCHMARK RESULTS

A. Comprehensive Benchmark Table

B. Accuracy Comparison

C. Training Time Comparison

D. Analysis of Results

As shown in Table I and Fig. 2, VectorForgeML demonstrates competitive performance in multi-class and ensemble environments, achieving strong results in Softmax, Decision Tree, and Random Forest models. In Fig. 3, VF-ML displays substantial time improvements over tidymodels for regression and clustering tasks, completing Ridge Regression in just 0.049 seconds compared to tidymodels' 0.752 seconds.

VII. DISCUSSION

The benchmark results indicate that VectorForgeML performs most efficiently in scenarios dominated by dense linear algebra operations. Models such as Linear Regression, Ridge Regression, and Softmax Regression benefit significantly from BLAS/LAPACK-accelerated matrix multiplications and Cholesky-based solvers, resulting in reduced computational overhead compared to higher-level R abstractions.

For tree-based models, the current implementation employs exhaustive split evaluation. While this approach ensures correctness and transparency of the algorithmic process, it leads to higher computational cost compared to highly optimized implementations in scikit-learn that utilize advanced heuristics and low-level C optimizations. Consequently, scikit-learn maintains a performance advantage in certain non-parametric models.

Overall, the results demonstrate that VectorForgeML successfully narrows the performance gap between R-native workflows and Python-based machine learning libraries, particularly in regression and multi-class classification tasks. The findings validate that system-level optimization within an R ecosystem can achieve competitive efficiency without sacrificing usability.

VIII. LIMITATIONS

- Exhaustive split search in Decision Trees increases computational cost.
- No GPU acceleration.
- No distributed training support.

IX. CONCLUSION

VectorForgeML demonstrates that a strictly optimized C++ core can seamlessly integrate with R to produce a framework

that is both highly usable and computationally aggressive. By bypassing abstraction layers, the system achieves competitive model accuracy and drastically cuts training times compared to traditional R methodologies.

X. Future Work

Future iterations will focus on eliminating current algorithmic bottlenecks. Priorities include integrating CUDA for GPU-accelerated tensor operations, implementing sparse matrix optimization for NLP tasks, developing parallel tree-building algorithms (histogram-based splits), and integrating AutoML pipelines for hyperparameter tuning.

References

[1] Pedregosa, F. et al., "Scikit-learn: Machine Learning in Python," JMLR, 12, pp. 2825-2830, 2011.

[2] Kuhn, M. and Wickham, H., "Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles," 2020.

[3] Eddelbuettel, D. and Francois, R., "Rcpp: Seamless R and C++ Integration," Journal of Statistical Software, 40(8), 1-18, 2011.

[4] Blackford, L. S. et al., "An Updated Set of Basic Linear Algebra Subprograms (BLAS)," ACM Trans. Math. Soft., 28(2), 135-151, 2002.