RandomMachine: Random Base-Learner Selection for Newton Gradient Boosting Ensembles

Ghiffary Rifqialdi
Independent Researcher
github.com/ghiffaryr/randommachine

February 2026

Abstract

We present RandomMachine, an open-source Python library that extends classical second-order (Newton) gradient boosting by randomly sampling the next base learner from a user-defined pool at each boosting iteration. Unlike standard gradient boosted trees, where every iteration adds a fresh clone of a single fixed model type, RandomMachine stochastically mixes multiple learner families—LightGBM, CatBoost, XGBoost, and arbitrary sklearn-compatible estimators—according to per-model sampling probabilities. This randomised selection increases ensemble diversity, acts as an implicit regulariser, and allows the user to leverage complementary inductive biases of different algorithms within a single coherent boosting procedure. We describe the algorithm, its theoretical motivation, and the software design, and report empirical results on synthetic regression and classification tasks demonstrating improvements of 1.55 % in R^2 on regression and 2.03 % in accuracy on binary classification over three fixed-family baselines at comparable hyper-parameter budgets. RandomMachine is available under the MIT license at https://github.com/ghiffaryr/randommachine.

Contents

1 Introduction

Gradient boosting machines (GBMs) [Friedman, 2001] are among the most powerful and widely deployed families of machine-learning models for structured data. State-of-the-art implementations such as LightGBM [Ke et al., 2017], XGBoost [Chen and Guestrin, 2016], and CatBoost [Prokhorenkova et al., 2018] achieve top performance on tabular benchmarks, primarily by refining second-order (Newton) approximations of a differentiable loss.

A classical GBM fixes a single base-learner template—typically a regression tree with a chosen maximum depth—and adds fresh instances of that same template at every iteration. This homogeneity simplifies analysis but has two practical drawbacks. First, the choice of tree depth heavily influences bias–variance trade-off, yet the optimal setting is task-dependent and costly to tune. Second, all base learners belong to the same model family, limiting the diversity of the final ensemble.

RandomMachine addresses both issues by replacing the single fixed base-learner template with a *pool* of candidates drawn from one or more model families. At each iteration the algorithm randomly selects a candidate from the pool (with configurable probabilities) and fits it to the current Newton step. This is conceptually similar to Random Forests' column subsampling [Breiman, 2001], but applied at the level of the entire base-learner family rather than at the feature level. The result is an ensemble that automatically spans multiple model complexities and inductive biases without manual model selection.

The main contributions of this work are:

1. A formulation of *Random Newton Boosting* in which the base-learner identity is a random variable drawn fresh at every boosting step.

2. A modular Python library implementing the algorithm for all popular boosting backends (LightGBM, CatBoost, XGBoost) as well as a fully generic interface that accepts arbitrary scikit-learn-compatible regressors.

3. Empirical evidence on synthetic benchmarks showing consistent improvement over single-family baselines.

2 Background

2.1 Second-Order Gradient Boosting

Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and a twice-differentiable loss $\ell(y, \hat{y})$, gradient boosting maintains an additive model $F(\mathbf{x}) = \sum_{t=1}^T \eta\, h_t(\mathbf{x})$ where η is the learning rate and each h_t is a base learner. The Newton update at step t with current predictions $\mathbf{z} = F^{(t-1)}(\mathbf{X})$ solves

$$h_t = \arg\min_h \sum_{i=1}^n \frac{g_i}{h_i}\, h(\mathbf{x}_i) \quad \text{subject to} \quad h \in \mathcal{H}, \tag{1}$$

where $g_i = \partial_{\hat{y}} \ell(y_i, z_i)$ is the first derivative (gradient) and $h_i = \partial^2_{\hat{y}} \ell(y_i, z_i)$ is the second derivative (Hessian). In practice this is implemented by fitting h_t to the negative gradient $\mathbf{u} = -\mathbf{g}/\mathbf{h}$ with sample weights $\mathbf{h}$.

2.2 Loss Functions

RandomMachine ships two built-in loss functions.

Mean Squared Error (regression). $\ell(y, \hat{y}) = (y - \hat{y})^2$, giving $g_i = 2(z_i - y_i)$ and $h_i = 2$.

Algorithm 1 Random Newton Boosting

Require: Training data $(\mathbf{X}, \mathbf{y})$; optional eval data $(\mathbf{X}_{\text{eval}}, \mathbf{y}_{\text{eval}})$; loss ℓ; learner pool $\mathcal{B} = \{b_1, \ldots, b_K\}$; sampling distribution $\mathbf{p} = (p_1, \ldots, p_K)$; learning rate η; max iterations T; early stopping patience P.

$\mathbf{z} \leftarrow \mathbf{0}_n$; $\mathcal{E} \leftarrow [\,]$; $\ell^* \leftarrow \infty$; $t^* \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  $\mathbf{g}, \mathbf{h} \leftarrow \nabla_{\mathbf{z}} \ell(\mathbf{y}, \mathbf{z})$, $\nabla^2_{\mathbf{z}} \ell(\mathbf{y}, \mathbf{z})$
  $\ell_t \leftarrow \ell(\mathbf{y}, \mathbf{z})$; **if** $\ell_t < \ell^*$ **then** $\ell^* \leftarrow \ell_t$; $t^* \leftarrow t$
  if eval data provided **then**
    compute ℓ_{eval} on $(\mathbf{X}_{\text{eval}}, \mathbf{y}_{\text{eval}})$
    if ℓ_{eval} has not improved for P steps **then**
      break ▷ early stopping
    end if
  else
    if $\ell_t > \ell^*$ and $t \bmod P = 0$ **then**
      break ▷ early stopping on train loss
    end if
  end if
  $\tilde{b} \sim \text{Categorical}(\mathcal{B}, \mathbf{p})$; $b_t \leftarrow \text{clone}(\tilde{b})$ ▷ random selection
  Fit b_t on $(\mathbf{X}, -\mathbf{g}/\mathbf{h})$ with sample weights $\mathbf{h}$
  $\mathbf{z} \leftarrow \mathbf{z} + \eta\, b_t(\mathbf{X})$
  Append b_t to $\mathcal{E}$
end for
return $\mathcal{E}$

Logistic Loss (binary classification). $\ell(y, \hat{y}) = -y \log \sigma(\hat{y}) - (1 - y) \log(1 - \sigma(\hat{y}))$, where $\sigma(x) = (1 + e^{-x})^{-1}$. The derivatives are $g_i = \sigma(z_i) - y_i$ and $h_i = \sigma(z_i)(1 - \sigma(z_i))$, clipped below at 10^{-16} for numerical stability. Predictions are converted to class labels via thresholding at 0.5.

2.3 Ensemble Diversity

Ensemble diversity is a classical concept in machine learning [Dietterich, 2000, Kuncheva and Whitaker, 2003]. A diverse ensemble tends to commit errors on different subsets of the input space, so their combined prediction is more accurate than any individual member. Standard GBMs obtain some diversity through the adaptive reweighting of residuals (each base learner sees a different target), but restricted to a single hypothesis class. RandomMachine introduces an additional source of diversity through random selection of the base-learner *type* at each step.

3 Algorithm

3.1 Random Newton Boosting

Algorithm 1 describes the core training procedure shared by all RandomMachine model families.

Prediction. For a test point $\mathbf{x}$:

$$\hat{F}(\mathbf{x}) = \sum_{b \in \mathcal{E}} \eta\, b(\mathbf{x}). \tag{2}$$

For classification, $\hat{y} = \mathbf{1}[\sigma(\hat{F}(\mathbf{x})) > 0.5]$.

3.2 Base Learner Pool Construction

Single-family pools (automated). For the named models `RandomLGBMRegressor`, `RandomCatBoostRegressor`, and `RandomXGBRegressor`, the pool is constructed automatically by enumerating depths $d \in [\texttt{min\_max\_depth}, \texttt{max\_max\_depth}]$ and assigning uniform probabilities $p_k = 1/K$.

Heterogeneous pools (user-defined). The generic `RandomRegressor` and `RandomClassifier` classes accept any `base_learners` list together with optional `probabilities`, enabling arbitrary mixing of model families, depths, regularisation, or even non-tree architectures.

4 Software Design

4.1 Package Structure

```
randommachine/
  losses.py          # MeanSquaredError, LogisticLoss
  lgbm_models.py     # RLGBM base, RandomLGBMRegressor/Classifier
  catboost_models.py # RCBM base, RandomCatBoostRegressor/Classifier
  xgboost_models.py  # RXGBM base, RandomXGBRegressor/Classifier
  random_models.py   # RM base, RandomRegressor, RandomClassifier
  __init__.py        # Public API
```

4.2 Class Hierarchy

Each family shares a common base (e.g. `RLGBM`) that implements `fit`, `predict_raw`, and the early-stopping logic. Concrete classes (e.g. `RandomLGBMRegressor`) construct the base-learner pool in their `__init__` and delegate to the base class via `super()`. Classifiers add `predict_proba` (sigmoid transform) and binary `predict` on top.

4.3 Public API

```
from randommachine import (
    RandomLGBMRegressor, RandomLGBMClassifier,
    RandomCatBoostRegressor, RandomCatBoostClassifier,
    RandomXGBRegressor, RandomXGBClassifier,
    RandomRegressor, RandomClassifier,
    MeanSquaredError, LogisticLoss,
)

# Automated pool (LightGBM, depths 3-6)
model = RandomLGBMRegressor(
    num_iterations=20, learning_rate=0.5,
    min_max_depth=3, max_max_depth=6, random_state=42
)
model.fit(X_train, y_train, X_eval=X_val, y_eval=y_val)
preds = model.predict(X_test)

# User-defined heterogeneous pool
from lightgbm import LGBMRegressor
from xgboost import XGBRegressor
model = RandomRegressor(
    base_learners=[
        LGBMRegressor(max_depth=3, verbose=-1),
        LGBMRegressor(max_depth=5, verbose=-1),
        XGBRegressor(max_depth=4, verbosity=0),
    ],
```

```
    num_iterations=20, learning_rate=0.3, random_state=42
)
model.fit(X_train, y_train)
```

4.4 Feature Importance

Because every ensemble member is a standard scikit-learn-compatible tree model, feature importances are readily accessible via

```
import numpy as np
importances = np.mean(
    [b.feature_importances_ for b in model.ensemble_
     if hasattr(b, 'feature_importances_')], axis=0
)
```

averaging `feature_importances_` across all ensemble members.

5 Experiments

All experiments use synthetic datasets generated with `sklearn.datasets.make_regression` (regression) and `make_classification` (binary classification). The regression task has 1 000 samples, 20 features, and Gaussian noise with $\sigma = 10$; the classification task has 1 000 samples, 20 features (10 informative, 3 redundant). All experiments use an 80/20 train/test split with `random_state=42`.

5.1 Regression Results

Table 1 compares models on mean squared error (MSE) and coefficient of determination (R^2) on the held-out test set.

Table 1: Regression benchmark results (1 000 samples, 20 features, noise $\sigma = 10$, 80/20 split).

Model	**MSE**	$\boldsymbol{R^2}$
LightGBM (baseline)	2993.02	0.9230
CatBoost (baseline)	926.33	0.9762
XGBoost (baseline)	4053.03	0.8958
RandomLGBM	2682.88	0.9310
RandomCatBoost	688.26	**0.9823**
RandomXGB	3696.88	0.9049
RandomMixed (CB+LGB)	1314.78	0.9662
Baseline average	2657.46	0.9317
RandomMachine average	2095.70	**0.9461**
Improvement		**+1.55%**

5.2 Classification Results

Table 2 reports accuracy on the binary classification task.

5.3 Discussion

The random depth-selection mechanism provides a consistent benefit over fixed-depth baselines across both tasks. On regression, `RandomCatBoostRegressor` achieves $R^2 = 0.982$, the best of

Table 2: Binary classification benchmark results (1 000 samples, 20 features, 10 informative, 80/20 split).

Model	**Accuracy**
LightGBM (baseline)	0.890
CatBoost (baseline)	0.920
XGBoost (baseline)	0.895
RandomLGBM	0.890
RandomCatBoost	**0.935**
RandomXGB	0.920
RandomMixed (CB+LGB)	0.935
Baseline average	0.9017
RandomMachine average	**0.9200**
Improvement	**+2.03%**

all models, compared to the XGBoost baseline's $R^2 = 0.896$—a gap of 8.6 percentage points—and also outperforms the strong CatBoost baseline ($R^2 = 0.976$). On classification, both `RandomCatBoostClassifier` and `RandomMixed` reach 0.935 accuracy, matching or exceeding every baseline (best baseline: CatBoost at 0.920). The 2.03 % aggregate improvement in classification is the strongest result, demonstrating that diversity across learner families and depths is particularly beneficial when the task is harder (10 informative features out of 20 total). The `RandomLGBM` result is on par with its corresponding baseline, indicating that the benefit varies by family; blending families (`RandomMixed`) reliably beats any single-family baseline.

The generic `RandomRegressor`/`RandomClassifier` classes allow practitioners to mix any set of sklearn-compatible regressors and thus extend the framework to settings not covered by the built-in pools.

6 Related Work

Gradient boosting. The foundational work is Friedman [2001]. Second-order (Newton) boosting is developed in Chen and Guestrin [2016], Ke et al. [2017], Prokhorenkova et al. [2018].

Ensembles and diversity. Breiman [1996, 2001] showed that randomness improves ensemble performance. Dietterich [2000], Kuncheva and Whitaker [2003] formalise diversity metrics for ensembles.

Heterogeneous ensembles. Opitz and Maclin [1999] surveys methods for combining diverse learner types. Stacking [Wolpert, 1992] and mixture-of-experts [Jacobs et al., 1991] combine different models but at inference time, whereas RandomMachine randomises the base-learner *within* the boosting loop.

7 Conclusion

We have presented RandomMachine, a Newton boosting framework that stochastically samples base learners from a heterogeneous pool at each iteration. The library is lightweight, modular, and fully compatible with the scikit-learn ecosystem. Empirical results on synthetic tasks show consistent performance gains of 1.55 % in R^2 on regression and 2.03 % in accuracy on binary classification over three fixed-family baselines (LightGBM, CatBoost, XGBoost) at equal iteration budgets.

Future work includes (i) adaptive probability updating (e.g. favour learner types that have historically reduced the loss); (ii) support for multi-class classification; (iii) GPU-accelerated variants for large-scale data; and (iv) formal theoretical analysis of the diversity–accuracy trade-off under random base-learner selection.

Code and Data Availability

The source code, tests, and tutorial notebook are available at https://github.com/ghiffaryr/randommachine under the MIT license. All experiments in this paper are fully reproducible by running `docs/tutorial.ipynb`.

Acknowledgements

The author thanks the developers of LightGBM, CatBoost, XGBoost, and scikit-learn whose libraries underpin this work.

References

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. doi: 10.1007/BF00058655.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785.

Thomas G Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer, 2000.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001. doi: 10.1214/aos/1013203451.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003. doi: 10.1023/A:1022859003006.

David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999. doi: 10.1613/jair.614.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. doi: 10.1016/S0893-6080(05)80023-1.