Стохастическая оптимизация в задачах обучения ML моделей

Романов Даниил Дмитриевич, 22.М03-мм

Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра статистического моделирования

Научный руководитель: к. ф.-м. н., доцент П. В. Шпилев
Рецензент: А. Г. Берлинков

Санкт-Петербург
2024г.

Введение

- Стохастическая оптимизация используется в различных классах задач, где традиционные детерминированные методы могут быть неэффективными или неприменимыми.
- Отправной точкой является работа Attention-based Random Forest and Contamination Model Utkin, L. V., Konstantinov, A. V. (2022)

Цели

- Рассмотрение представленных моделей;
- Оптимизация модели с помощью мини - пакетного стохастического градиентного спуска;
- Реализация моделей;
- Тестирование моделей;
- Сравнение результатов с существующими моделями машинного обучения;

Механизм внимания

Идея механизма внимания:

$$\sum_{i=1}^{n} \alpha(x, x_i) y_i, \tag{1}$$

где y_i — это бинарный вектор, состоящий из нулей и одной единицы, x - вектор соответствующий новым наблюдениям, x_i - вектор i-го экземпляра для набора данных $S = (x_1, y_1), ...(x_n, y_n)$.

Вес определяется следующим образом:

$$\alpha(x, x_i) = \frac{K(x, x_i)}{\sum_{j=1}^{n} K(x, x_j)}, \tag{2}$$

где $K(\frac{\rho(x,x_i)}{h})$ — ядро Надарйи-Ватсона.
Расширение:

$$\alpha(x, x_i) = \sigma\left(q^T K\right)_i = \frac{\exp\left(q^T k_i\right)}{\sum_{j=1}^{n} \exp\left(q^T k_j\right)}, \tag{3}$$

где σ это многопеременная логистическая функция, $q = W_q x$ и $k_i = W_k x_i$. W_q и W_k - матрицы параметров, $K = [k_1, \ldots, k_n]$.

Attention-based RF: общая модель

Введем вектор средних $A_k(x)$:

$$A_k(x) = \frac{1}{\mathcal{J}_i^{(k)}} \sum_{j \in \mathcal{J}_i^{(k)}} x_j, \tag{4}$$

где $\mathcal{J}_i^{(k)}$ — индекс экземпляра, попадающего в лист $t_i^{(k)}$, k — индекс дерева.

Распределение вероятностей класса $p(x)$ во всем случайном лесе:

$$p(x) = \sum_{k=1}^{T} \alpha\left(x, A_k(x), w\right) p_k(x), \tag{5}$$

где $\alpha\left(x, A_k(x), w\right)$ - вес внимания, w — вектор параметров тренировочного внимания, $p_k(x) = n_i^{(k)}(c)/n_i^{(k)}$, $c = 1, ..., C$.

Attention-based RF: общая модель

Задача оптимизации:

$$w_{opt} = \arg\min_{w \in \mathcal{W}} \sum_{s=1}^{n} L\left(p(x_s), h_s, w\right), \tag{6}$$

Функция потерь:

$$\sum_{s=1}^{n} L\left(p(x_s), h_s, w\right) = \sum_{s=1}^{n} \left\| h_s - \sum_{k=1}^{T} \alpha\left(x_s, A_k(x_s), w\right) p(x_s) \right\|^2, \tag{7}$$

где бинарный вектор $h_s = (h_s(1), ..., h_s(C))$, имеет единичный элемент с индексом, соответствующим классу s-го экземпляра.

Epsilon Attention Forest(EAF): модель классификации

Модель ϵ — загрязнения Губера:

$$(1-\epsilon)\cdot P+\epsilon\cdot Q, \tag{8}$$

Веса внимания:

$$\alpha\left(x_s, A_k(x_s), w\right) = (1-\epsilon)\cdot \operatorname{softmax}\left(d(x_s A_k(x_s))\right) + \epsilon\cdot w_k. \tag{9}$$

Задача оптимизации:

$$\min_{w\in\mathcal{W}} \sum_{s=1}^{n} \left\| h_s - \sum_{k=1}^{T} \left((1-\epsilon)D_k(x_s,\tau)+\epsilon w_k\right) p(x_s) \right\|^2 \tag{10}$$

Features Weighted Attention Forest(FWAF): модель классификации

Вес для k-го дерева:

$$\alpha\left(x, A_k(x), v, z\right) = \sigma\left(\frac{\|(x - A_k(x)) \circ z\|^2}{2} v_k\right), \; k = 1, ..., T, \tag{11}$$

где $v = (v_1, \ldots, v_T)$ — вектор тренировочных параметров,
$z = (z_1, \ldots, z_m)$ — вектор весов признаков.
Задача оптимизации:

$$\sum_{s=1}^{n} L\left(p(x_s), h_s, v, z\right) = \sum_{s=1}^{n} \left\| h_s - \sum_{k=1}^{T} \sigma\left(\frac{\|(x - A_k(x)) \circ z\|^2}{2} v_k\right) p(x_s) \right\|^2. \tag{12}$$

Features Weighted Attention Forest: Модель регрессии

Целевая функция (12) в случае регрессии может быть записана следующим образом:

$$\sum_{s=1}^{n} L\left(\bar{y}, y_s, v, z\right) = \sum_{s=1}^{n} \left(y_s - \sum_{k=1}^{T} \sigma \left(\frac{\left\| \left(x - A_k(x)\right) \circ z \right\|^2}{2} v_k \right) B_k(x_s) \right)^2, \tag{13}$$

где $B_k(x) = \frac{1}{\mathcal{J}_i^{(k)}} \sum_{j \in \mathcal{J}_i^{(k)}} y_j$

Метод нахождения оптимальных весов

- является эффективным методом для нахождения локального оптимума в пространстве параметров,
- работает быстрее, чем методы оптимизации, требующие вычисления градиента по всему обучающему набору данных,
- хорошо работает с такими нелинейными функциями потерь,
- имеет свойство робастности к выбросам в данных,
- позволяет эффективно обучать параметры v и z, не зависимо от того, какие распределения они имеют,

Программная реализация

Программная реализация содержит следующие основные классы:

- Class ClfRegHot – абстрактный класс, определяющий интерфейс для последующих классов
- Class LeafData – класс для подготовки данных для помещения листьев в дерево
- Class AttentionForest – класс для общей реализации
- Class EpsAttentionForest – класс с реализацией модели ϵ - загрязнения Губера
- Class FeatureWeightetAttentionForest – класс с реализацией модели с дополнительными весами

  Изменены методы: get_dynamic_weightsy, optimize_weights, model, loss, predict.
  Добавлены методы: _prepare_leaf_data_cl, optimize_weights_sgd.

Информация о наборах данных

Информация о наборах данных:

Таблица: Информация о датасетах для классификации

Датасет	m	n	C
Seismic bumps	18	2584	2
Diabetic Retinopathy	20	1151	2
Eeg Eyes	14	14980	2
Tic-Tac-Toe Endgame	27	957	2

Таблица: Информация о датасетах для регрессии

Датасет	m	n
Wine red	12	1599
Boston housing	13	506
Concrete	8	1030
Yacht Hydrodynamics	6	308

FWAF vs SVM на наборах для классификации

Таблица: Сравнение результатов классификации

data set	ROC AUC score	Accuracy
	FWAF	
Seismic bumps	0.715	0.930
Eeg Eyes	0.976	0.933
Diabetic Retinopathy	0.734	0.667
Tic-Tac-Toe Endgame	0.999	0.990
	linear	
Seismic bumps	0.394	0.930
Eeg Eyes	0.677	0.639
Diabetic Retinopathy	0.8002	0.719
Tic-Tac-Toe Endgame	0.983	0.979
	poly	
Seismic bumps	0.526	0.919
Eeg Eyes	0.613	0.569
Diabetic Retinopathy	0.776	0.679
Tic-Tac-Toe Endgame	0.999	0.984
	rbf	
Seismic bumps	0.469	0.930
Eeg Eyes	0.612	0.549
Diabetic Retinopathy	0.785	0.697
Tic-Tac-Toe Endgame	0.998	0.992

FWAF vs SVR на наборах для регрессии

Таблица: Сравнение результатов регрессии

	FWAF	
Wine red	0.346	0.480
Boston housing	0.786	2.832
Concrete	0.817	5.223
Yacht Hydrodynamics	0.992	0.646
	linear	
Wine red	0.291	0.512
Boston housing	0.569	3.407
Concrete	0.529	8.732
Yacht Hydrodynamics	0.004	8.199
	poly	
Wine red	0.043	0.599
Boston housing	0.189	4.773
Concrete	0.456	10.201
Yacht Hydrodynamics	0.144	9.719
	rbf	
Wine red	0.132	0.566
Boston housing	0.195	4.788
Concrete	0.221	12.026
Yacht Hydrodynamics	0.169	9.905

FWAF vs FWAF без sgd на наборах для классификации

Таблица: Результаты тестирования на датасетах для классификации

	FWAF без sgd	FWAF
Data set	ROC AUC score	ROC AUC score
Seismic bumps	0.507	0.715
Diabetic Retinopathy	0.646	0.734
Eeg Eyes	0.733	0.976
Tic-Tac-Toe Endgame	0.903	0.999

FWAF vs FWAF без sgd на наборах для регрессии

Таблица: Результаты тестирования на датасетах для регрессии

FWAF без sgd		
	R^2	MAE
Data set	test	test
Wine red	0.3411	0.5
Boston housing	0.7793	2.9767
Concrete	0.7864	7.5240
Yacht Hydrodynamics	0.9816	1.7001
FWAF		
	R^2	MAE
Data set	test	test
Wine red	0.3455	0.4804
Boston housing	0.7859	2.8324
Concrete	0.8172	5.2226
Yacht Hydrodynamics	0.9923	0.6465

FWAF vs EAF на наборах для классификации

Таблица: Результаты тестирования на датасетах для классификации

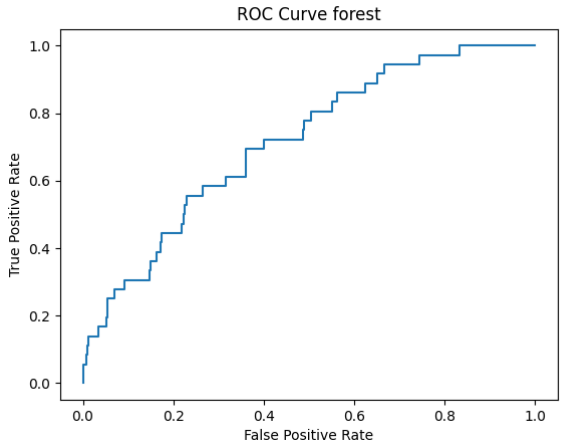
	EAF		FWAF	
Data set	train	test	train	test
Seismic bumps	0.633	0.483	0.979	0.715
Diabetic Retinopathy	0.704	0.628	0.911	0.734
Eeg Eyes	0.721	0.672	0.996	0.976
Tic-Tac-Toe Endgame	0.674	0.614	1.0	0.999

FWAF vs EAF на наборах для регрессии

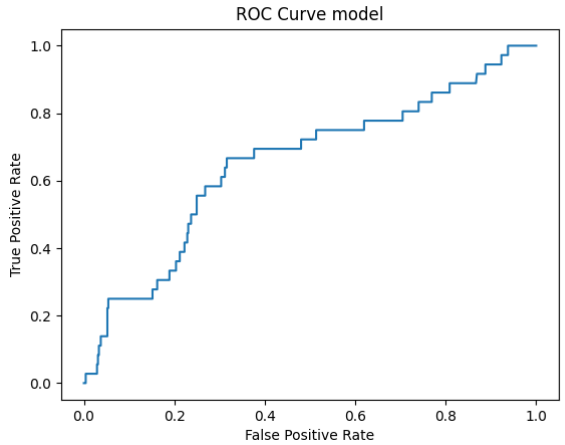
Таблица: Результаты тестирования на датасетах для регрессии

EAF				
	R^2		MAE	
Data set	train	test	train	test
Wine red	0.782	0.420	0.281	0.459
Boston housing	0.959	0.823	1.306	2.511
Concrete	0.958	0.853	2.412	4.694
Yacht Hydrodynamics	0.997	0.984	0.351	0.749
FWAF				
	R^2		MAE	
Data set	train	test	train	test
Wine red	0.741	0.346	0.305	0.480
Boston housing	0.949	0.786	1.391	2.832
Concrete	0.895	0.817	3.794	5.223
Yacht Hydrodynamics	0.997	0.992	0.376	0.646

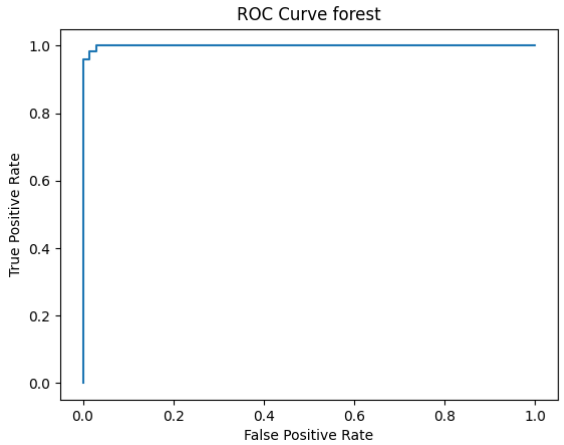
FWAF vs EAF: ROC AUC



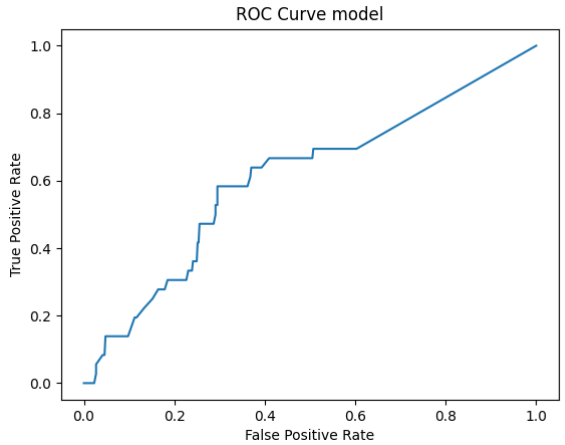
(a) ROC кривая FWAF AUC = 0.715



(b) ROC кривая EAF AUC = 0.483



(c) ROC кривая FWAF AUC = 0.999



(d) ROC кривая EAF AUC = 0.614

Таблица: Результаты тестирования на датасетах для классификации

	FWAF без sgd	FWAF	xgboost
Data set	ROC AUC score		
Seismic bumps	0.507	0.715	0.708
Diabetic Retinopathy	0.646	0.734	0.755
Eeg Eyes	0.733	0.976	0.979
Tic-Tac-Toe Endgame	0.903	0.999	0.996

Таблица: Результаты тестировани на датасетах для регрессии

	FWAF без sgd	FWAF	xgboost
Data set	R^2		
Wine red	0.341	0.346	0.432
Boston housing	0.779	0.786	0.885
Concrete	0.786	0.817	0.817
Yacht Hydrodynamics	0.982	0.992	0.996
Data set	MAE		
Wine red	0.500	0.480	0.403
Boston housing	2.977	2.832	2.136
Concrete	7.524	5.23	3.187
Yacht Hydrodynamics	1.700	0.647	0.390

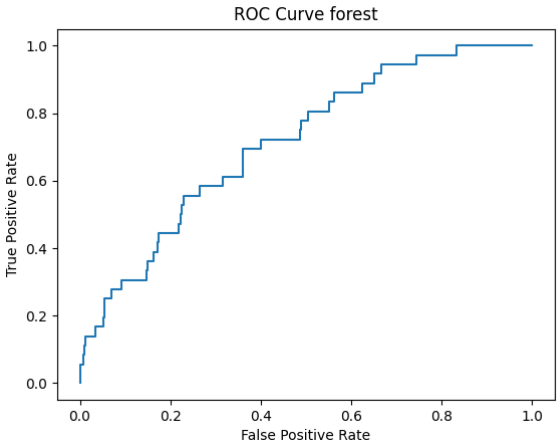
Таблица: Результаты тестирования на датасетах для классификации

	EAF	FWAF	catboost
Data set	ROC AUC score		
Seismic bumps	0.507	0.715	0.753
Diabetic Retinopathy	0.646	0.734	0.722
Eeg Eyes	0.733	0.976	0.927
Tic-Tac-Toe Endgame	0.903	0.999	0.999

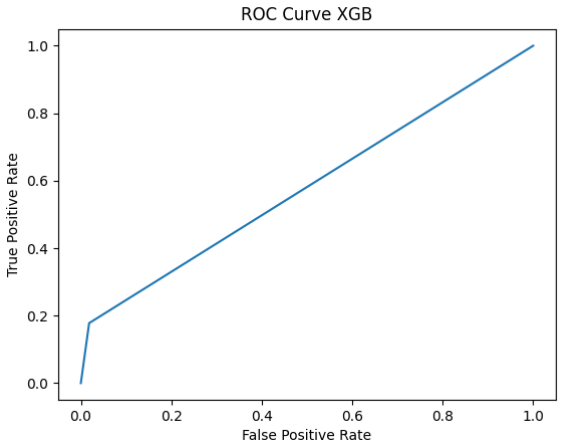
Таблица: Результаты тестировани на датасетах для регрессии

	EAF	FWAF	catboost
Data set		R^2	
Wine red	0.341	0.346	0.442
Boston housing	0.779	0.786	0.855
Concrete	0.786	0.817	0.940
Yacht Hydrodynamics	0.982	0.992	0.982
Data set		MAE	
Wine red	0.500	0.480	0.426
Boston housing	2.977	2.832	2.222
Concrete	7.524	5.23	2.741
Yacht Hydrodynamics	1.700	0.647	0.766

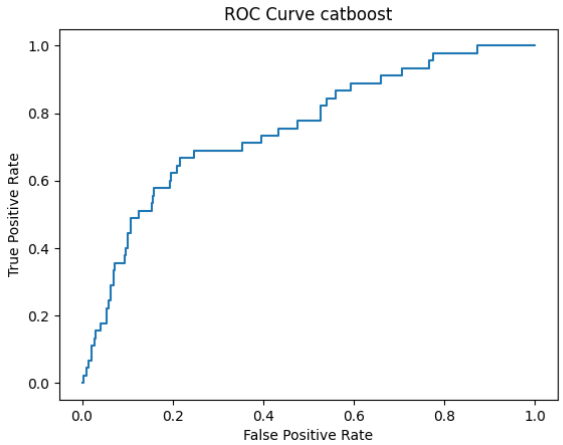
FWAT и ансамблевые методы: ROC AUC



(a) ROC кривая FWAF AUC = 0.715

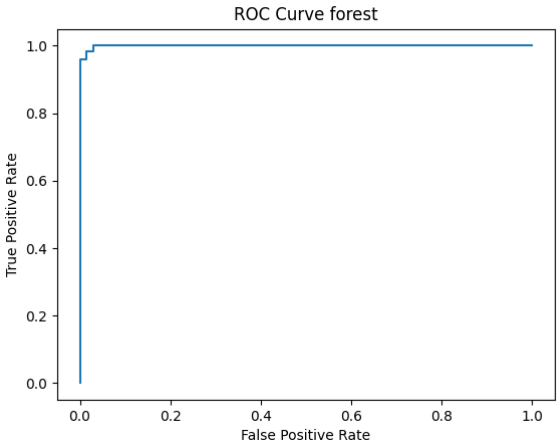


(b) ROC кривая XGB AUC = 0.708

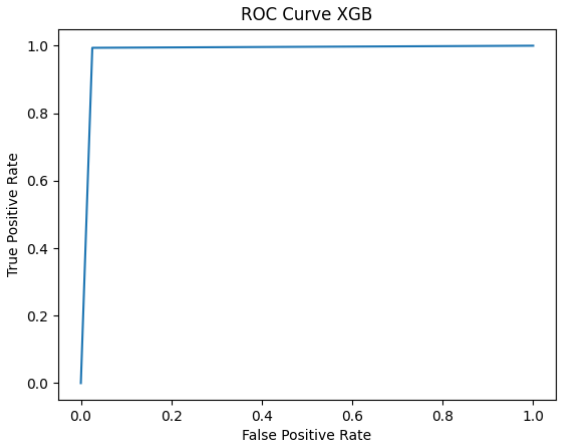


(c) ROC кривая catboost AUC = 0.753

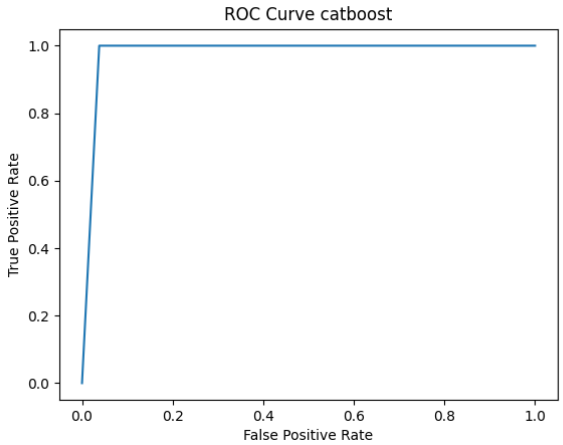
FWAT и ансамблевые методы: ROC AUC



(a) ROC кривая FWAF AUC = 0.999



(b) ROC кривая XGB AUC = 0.996



(c) ROC кривая catboost AUC = 0.999

Сравнение с моделью с механизмом внимания

Таблица: FWAF и CNN-CBAM-SVM

Результаты классификации изображений				
		FWAF		
Data set	Accuracy	Precision	Recall	F1 Score
ISH	0.9836	0.9813	0.9814	0.9817
Annadatha	0.9737	0.9744	0.9737	0.9739
		CNN-CBAM-SVM		
ISH	0.9961	0.9921	1.0	0.9964
Annadatha	0.9981	1.0	0.9968	0.9984

Вывод

- Рассмотрен подход ABRF (the attention-based random forest) и модели использующие данный подход,
- предложена оптимизация модели с помощью метода Mini-batch стохастического градиентного спуска,
- рассмотренная и оптимизированная модели были реализованы на языке Python для тестирования и сравнения результатов,
- сделан вывод о том, что оптимизированная модель не сильно уступает, либо превосходит рассмотренные модели в качестве предсказаний для классификации или регрессии.

Обзор литературы

Alan Q. Wang, Mert R. Sabuncu. A Flexible Nadaraya-Watson Head Can Offer Explainable and Calibrated Classification

Hong Bo Li, Wei Wang, Hong Wei Ding, Jin Dong. Trees Weighting Random Forest Method for Classifying High-Dimensional Noisy Data

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017, 2017.

Utkin, L. V., Konstantinov, A. V. (2022). Attention-based Random Forest and Contamination Model.

В данной работе предлагается новый подход ABRF (the attention-based random forest), и его модификации для применения механизма внимания к случайному лесу (RF) для регрессии и классификации.

Технические Слайды

Задача классификации

Формально, пусть у нас будет обучающая выборка:
$S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$,

где $x_i = (x_{i1}, ..., x_{im}) \in \mathbb{R}^m$ представляет собой вектор признаков,
$y_i \in \mathbb{R}$ - метки класса объектов из набора $\{1, ..., C\}$.

Задача классификации заключается в нахождении алгоритма
$f : \mathcal{X} \to \mathcal{Y}$, который максимизирует вероятность правильной классификации новых объектов.

Формула, используемая для классификации объекта x:

$$\hat{y} = f(x),$$

где $\hat{y}$ - предсказанная метка класса для объекта x.
Задача:

$$L(\hat{y}, y),$$

где L - функция потерь, которая выбирается в зависимости от конкретной задачи классификации.

Задача регрессии

Формально, пусть имеем обучающую выборку
$S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$,

где $x_i = (x_{i1}, ..., x_{im}) \in \mathbb{R}^m$ представляет собой вектор признаков i-го объекта, а $y_i \in \mathbb{R}$ - наблюдаемый выход, такой, что

$$y_i = f(x_i) + \varepsilon_i, \qquad i = 1, \ldots, n.$$

Цель: Найти $\hat{y} = f$, которая наилучшим образом описывает зависимость между X и y, и которая минимизирует ожидаемый риск $L(\hat{y}, y)$.

Случайный лес

Предположим, требуется выполнить задачу классификации или регрессии. Допустим, имеется обучающая выборка $S = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, где x_i - входные данные, а y_i - соответствующие им метки классов в случае классификации или значения целевой переменной в случае регрессии.
Алгоритм обучения случайного леса:

- Выборка с замещением (Bootstraping): Из обучающей выборки S случайным образом выбираются n примеров с замещением.
- Построение деревьев решений: На каждом шаге построения дерева решений:
  - Выбирается случайное подмножество признаков из всего набора признаков.
  - По этому подмножеству признаков строится дерево решений.
  - Дерево строится до достижения максимальной глубины или до тех пор, пока в узле не останется минимальное количество объектов.
- Объединение деревьев: После построения всех деревьев каждое из них голосует за классификацию или предсказывает значение для регрессии.

Mini-batch SGD

- Инициализация параметров v и z некоторыми начальными значениями.
- Для каждого эпохи обучения:
  - Перемешивание обучающего набора данных для создания случайного порядка.
  - Разделение обучающего набора данных на мини-батчи фиксированного размера.
  - Для каждого мини-батча:
    - Выбор текущего мини-батча данных x_s и соответствующих целевых меток h_s.
    - Вычисление градиента функции потерь L по отношению к параметрам v и z:

$$\nabla_v L = \frac{\partial L}{\partial v}, \quad \nabla_z L = \frac{\partial L}{\partial z}$$

    - Обновление параметров v и z в направлении, противоположном градиенту, используя скорость обучения η:

$$v_{t+1} = v_t - \eta \nabla_v L, \quad z_{t+1} = z_t - \eta \nabla_z L$$

- Повторение этого процесса до тех пор, пока не будет достигнуто определенное условие останова (например, количество эпох или сходимость функции потерь).