



doi 10.26089/NumMet.v27r438

УДК 519.8

Применение методов машинного обучения для ускорения решения задачи выбора состава включенного генерирующего оборудования

С. А. Романов

Омский филиал Института математики имени С. Л. Соболева СО РАН, Омск, Российская Федерация
ORCID: 0009-0007-8439-6389, e-mail: romanovomsk2014@gmail.com

Аннотация: Задача выбора состава включенного генерирующего оборудования (unit commitment problem) занимает центральное место в краткосрочном планировании энергосистем и обычно формулируется в виде задачи смешанного целочисленного линейного программирования большой размерности. В работе исследуется подход к ускорению ее решения с помощью методов машинного обучения. Обученные классификаторы предсказывают состояние каждого генератора в каждый период планирования, после чего уверенно классифицированные генераторы фиксируются, а оставшаяся задача меньшей размерности решается точным методом. В настоящей работе, в дополнение к существующему подходу на основе логистической регрессии, сравниваются и применяются шесть алгоритмов классификации (логистическая регрессия, случайный лес, градиентный бустинг, многослойный персептрон, метод опорных векторов и метод ближайших соседей) к полной модели с ограничениями минимального времени работы и простоя генераторов. Эксперименты проведены на трех наборах данных библиотеки `rglib-us` с перекрестной проверкой по схеме “исключение одного”. Методы на основе деревьев решений обеспечивают ускорение точного решения в 2.2–3.4 раза при медианном отклонении целевой функции менее 0.05% на наиболее крупном наборе FERC (до 1028 генераторов), тогда как логистическая регрессия, метод опорных векторов и многослойный персептрон при таком масштабе теряют надежность.

Ключевые слова: задача выбора состава включенного генерирующего оборудования, машинное обучение, целочисленное линейное программирование, классификация, ускорение решения.

Благодарности: Работа выполнена в рамках государственного задания Института математики имени С. Л. Соболева СО РАН (проект № FWNF-2026-0019). Вычислительный сервер с AMD EPYC 7502 CPU предоставлен Омским филиалом Института математики имени С. Л. Соболева СО РАН.

Для цитирования: Романов С.А. Применение методов машинного обучения для ускорения решения задачи выбора состава включенного генерирующего оборудования // Вычислительные методы и программирование. 2026. 27, № 4. 623–638. doi 10.26089/NumMet.v27r438.

Application of machine learning methods to accelerate the solution of the unit commitment problem

Sergey A. Romanov

Omsk Department of Sobolev Institute of Mathematics SB RAS, Omsk, Russia

ORCID: 0009-0007-8439-6389, e-mail: romanovomsk2014@gmail.com

Abstract: The unit commitment problem is central to short-term power system planning and is usually formulated as a large-scale mixed-integer linear programming problem. The paper investigates an approach to accelerating its solution using machine learning methods. Trained classifiers predict the state of each generator in every planning period, after which confidently classified generators are fixed, and the remaining lower-dimensional problem is solved by an exact method. The existing approach based on logistic regression is extended: six classification algorithms (logistic regression, random forest, gradient boosting, multilayer perceptron, support vector machine, and k -nearest neighbors) are compared and applied to the full model including minimum up-time and minimum down-time constraints. Experiments are carried out on three datasets from the pglib-uc library using leave-one-out cross-validation. Tree-based methods accelerate the exact solution by a factor of 2.2–3.4 with a median objective deviation below 0.05% on the largest FERC dataset (up to 1028 generators), whereas logistic regression, support vector machine, and multilayer perceptron lose reliability at this scale.

Keywords: unit commitment, machine learning, integer linear programming, classification, solution acceleration.

Acknowledgements: The work was carried out within the framework of the state task of the Sobolev Institute of Mathematics SB RAS (project FWNF-2026-0019). The computing server with AMD EPYC 7502 CPU is provided by the Omsk Department of the Sobolev Institute of Mathematics SB RAS.

For citation: S. A. Romanov, “Application of machine learning methods to accelerate the solution of the unit commitment problem,” *Numerical Methods and Programming*. 27 (4), 623–638 (2026). doi 10.26089/NumMet.v27r438.

1. Введение. Задача выбора состава включенного генерирующего оборудования (Unit Commitment, UC) заключается в определении расписания включения и выключения генераторов электростанций на заданном горизонте планирования с целью минимизации суммарных затрат при соблюдении ограничений на баланс мощности, резервы, скорость изменения мощности генераторов (рамповые ограничения) и минимальные времена работы и простоя [1].

Обычно задача UC формулируется в виде задачи смешанного целочисленного линейного программирования (MILP), что позволяет использовать для ее решения коммерческие и свободно распространяемые программные решатели, такие как Gurobi, CPLEX и HiGHS. Размерность задачи растет линейно с числом генераторов и временных периодов. Однако, поскольку задача является NP-трудной, время точного решения в худшем случае растет экспоненциально. Ниже под экземпляром задачи понимается индивидуальная задача, в которой всем числовым параметрам присвоены конкретные значения: заданы горизонт планирования, профили спроса и резерва, а также характеристики генераторов. Такие экземпляры оператор энергосистемы решает многократно в течение суток. На практике в промышленных энергосистемах с сотнями генераторов решение одного экземпляра может занимать от нескольких минут до нескольких часов.

Одним из перспективных направлений ускорения стало применение машинного обучения для предварительного определения состояния генераторов. В работе [2] предложен следующий подход. Логистическая регрессия предсказывает состояние каждого генератора в каждый период. Для генераторов, уверенно классифицированных как постоянно включенные или постоянно выключенные, соответствующие бинарные переменные фиксируются, после чего решатель работает с задачей меньшей размерности.

Применение машинного обучения для ускорения решения задач комбинаторной оптимизации активно развивается (см., например, [3]). В задаче УС использование ранее решенных экземпляров для решения новых рассматривается в целом ряде работ, различающихся тем, что именно предсказывается и как это предсказание передается точному решателю. В [4] по накопленным решениям строятся предсказания трех видов: заведомо избыточные ограничения, которые можно исключить из модели; начальное допустимое решение; аффинное подпространство, которое с высокой вероятностью содержит оптимальное решение. На реалистичных системах большого масштаба применение этих предсказаний ускоряет решение в среднем в 4.3 раза с сохранением гарантии оптимальности и в 10.2 раза без нее. В [5] графовая нейронная сеть предсказывает состояния генераторов и активные сетевые ограничения, причем каждому предсказанию сопоставляется оценка уверенности, и лишь надежные предсказания после восстановления допустимости передаются решателю. Аналогичная схема порога уверенности применяется и в настоящей работе. В [6] состояния генераторов предсказываются методом опорных векторов, регуляризация которого дает гарантию качества вне обучающей выборки, а сами предсказания служат начальным приближением, а не жесткой фиксацией. В [7] машинное обучение применяется не к бинарным переменным напрямую, а к начальным двойственным оценкам для ускорения метода генерации столбцов. При этом в [8] показано, что на реалистичных данных логистическая регрессия нередко дает ускорение, сопоставимое со сложными моделями. Поэтому усложнение метода обучения оправдано лишь при статистически значимом выигрыше над таким базовым уровнем. Это соображение мотивирует включение в сравнение наивного классификатора. Систематическое изложение работ по применению машинного обучения к задаче УС приведено в обзорах [9, 10].

Цель настоящей работы — сравнить классификаторы разных семейств в рамках схемы с фиксацией уверенно предсказанных переменных на открытом наборе задач `pglib-us` с использованием свободно распространяемого решателя `HiGHS` и выяснить, какие из них сохраняют устойчивое ускорение без потери качества решения при росте размерности. По сравнению с [2], где применялась единственная логистическая регрессия, в данной работе подход развивается в нескольких направлениях. Вместо одной модели применяются и сравниваются шесть алгоритмов классификации, охватывающих основные семейства методов машинного обучения, а также наивный классификатор в роли нижней границы. Эксперименты проводятся на полной модели УС с ограничениями минимального времени работы и простоя генераторов, которые отсутствовали в [2]. Подход масштабируется до систем с 1028 генераторами (набор данных `FERC`), что втрое превышает масштаб исходного исследования (327 генераторов). Наконец, жесткие ограничения на обеспечение спроса заменены мягкими со штрафом, что сохраняет формальную допустимость задачи при любой фиксации переменных. В отличие от [4, 6, 7], где предсказание используется решателем в качестве начального приближения, в данной работе и в [2] уверенно классифицированные переменные фиксируются жестко, а качество методов оценивается перекрестной проверкой с исключением одного экземпляра. Альтернативным подходом может быть использование схемы “теплого старта” (*warm start*), в которой предсказание передается решателю в качестве начального приближения. Реализация этого подхода и сравнение с применяемой здесь жесткой фиксацией оставлены в качестве направления дальнейшей работы.

2. Постановка задачи выбора состава включенного генерирующего оборудования. Рассматривается детерминированная формулировка УС на основе [1]. Полная спецификация модели приведена в [11]. Небольшая модификация, сделанная в настоящей работе, состоит в преобразовании жестких ограничений на обеспечение спроса в мягкие путем добавления переменных невязки и штрафных коэффициентов. Это позволяет сохранять формальную допустимость задачи при произвольной фиксации бинарных переменных и сравнивать качество решений даже на тех экземплярах, где выполнить эти ограничения не удалось.

Пусть $\mathcal{G}$ — множество тепловых генераторов, $\mathcal{T} = \{1, \dots, T\}$ — множество временных периодов. Основной переменной решения служит состояние генератора $u_{g,t} \in \{0, 1\}$: значение 1 соответствует включенному состоянию генератора g в период t , а значение 0 — выключенному. Именно эти бинарные переменные определяют комбинаторную сложность задачи и являются целью предсказания методами машинного обучения. Помимо них, модель включает вспомогательные бинарные переменные $v_{g,t}$ и $w_{g,t}$, фиксирующие моменты пуска и останова генератора, непрерывные переменные $p_{g,t} \geq 0$ (мощность выработки сверх минимальной) и $r_{g,t} \geq 0$ (резерв мощности), а также непрерывные переменные $s_t^D \geq 0$ и $s_t^R \geq 0$, отражающие непокрытую часть спроса и недостающий резерв в период t .

Целевая функция минимизирует суммарные затраты на производство электроэнергии, пуски гене-

раторов, работу на холостом ходу, а также стоимость нехватки генерации и резерва:

$$\min \left(\sum_{g \in \mathcal{G}} \sum_{t \in \mathcal{T}} [C_g^{\text{prod}}(p_{g,t}) + C_g^{\text{start}}(g, t) + C_g^{\text{nl}} \cdot u_{g,t}] + \sum_{t \in \mathcal{T}} [M^D \cdot s_t^D + M^R \cdot s_t^R] \right). \quad (1)$$

Здесь C_g^{prod} — кусочно-линейная функция затрат на выработку, C_g^{start} — стоимость пуска генератора, которая тем выше, чем дольше генератор находился в простое, C_g^{nl} — стоимость работы генератора на холостом ходу. Удельная стоимость нехватки задается штрафными коэффициентами $M^D = 10^6$ \$/МВт и $M^R = 10^5$ \$/МВт, многократно превышающими стоимость генерации.

Модель включает несколько групп ограничений. Ограничения баланса мощности и резерва связывают суммарную выработку и резерв включенных генераторов с прогнозом спроса D_t и требуемым уровнем резерва R_t . В отличие от [11], в эти ограничения добавлены переменные невязки s_t^D и s_t^R :

$$\sum_{g \in \mathcal{G}} (P_g^{\min} u_{g,t} + p_{g,t}) + s_t^D \geq D_t, \quad t \in \mathcal{T}, \quad (2)$$

$$\sum_{g \in \mathcal{G}} r_{g,t} + s_t^R \geq R_t, \quad t \in \mathcal{T}, \quad (3)$$

где $P_g^{\min}$ — минимальная мощность генератора g . При нехватке включенной мощности или резерва ограничения (2) и (3) остаются выполнимыми за счет положительных значений переменных s_t^D и s_t^R , штрафных в целевой функции (1). В этом состоит отличие используемой модели от исходной формулировки [11].

Логические ограничения связывают переменные состояния, пуска и останова: если генератор был выключен в предыдущий период и включен в текущий, фиксируется событие пуска, и наоборот. Ограничения минимального времени работы UT_g и простоя DT_g запрещают слишком частое переключение генераторов: после включения генератор должен проработать не менее UT_g периодов, а после выключения — простоять не менее DT_g периодов. Помимо этого, модель учитывает ограничения на скорость изменения мощности (рамповые ограничения) и кусочно-линейное представление функции затрат на выработку.

3. Метод МЛ-ускорения.

3.1. Обзор подхода. На практике задачу УС приходится решать многократно: расписание пересчитывается несколько раз в сутки при изменении профиля спроса, цен на топливо или состава доступного оборудования. Накопленный архив исторических расписаний содержит решения сотен похожих задач, и его естественно использовать как обучающую выборку для машинного обучения.

Предлагаемый метод основан на подходе работы [2], где обучалась единственная логистическая регрессия, а признаки строились по профилям спроса в каждом узле сети с учетом перетоков мощности. Формулировка `rglib-uc` не содержит сетевых ограничений, поэтому в качестве входа доступен только агрегированный спрос. Вектор признаков для экземпляра с T периодами определяется следующим образом:

$$\mathbf{x} = \left(\frac{D_1}{D_{\max}}, \dots, \frac{D_T}{D_{\max}}, \frac{R_1}{D_{\max}}, \dots, \frac{R_T}{D_{\max}} \right) \in \mathbb{R}^{2T}, \quad (4)$$

где $D_{\max} = \max_t D_t$ — пиковый спрос, используемый для нормализации.

Для каждой пары (генератор g , период t) обучается отдельный бинарный классификатор $M_{g,t}$, что позволяет учесть индивидуальные характеристики генераторов (мощность, стоимость, рамповые ограничения) и различие их поведения в разные периоды суток. Все классификаторы получают на вход один и тот же вектор признаков $\mathbf{x}$, а модель $M_{g,t}$ определяет, будет ли генератор g включен в период t при заданном профиле спроса. Общее число моделей равно $|\mathcal{G}| \cdot T$. Каждая из них решает простую задачу бинарной классификации с небольшим вектором признаков, поэтому обучение и предсказание выполняются быстро. Если в обучающей выборке некоторая пара (g, t) всегда находится в одном и том же состоянии, обучение модели не требуется, результат определяется константой.

Перед решением нового экземпляра УС все классификаторы $\{M_{g,t}\}$ применяются к его признакам и совместно дают предсказанное расписание $\hat{u}_{g,t}$ для каждого генератора во все периоды горизонта. Если для некоторого генератора g ансамбль моделей $\{M_{g,t}\}_{t \in \mathcal{T}}$ уверенно предсказывает один и тот же исход во всех периодах (либо $\hat{u}_{g,t} = 1$ для всех t , либо $\hat{u}_{g,t} = 0$ для всех t), соответствующие бинарные переменные фиксируются. Если же предсказания по периодам расходятся (например, $\hat{u}_{g,1} = 0$, а $\hat{u}_{g,t'} = 1$ при некотором $t' \in \mathcal{T}$) или хотя бы одна из моделей $M_{g,t}$ не достигает порога уверенности, генератор оставляется свободным. Решатель работает с задачей, в которой часть переменных уже определена, что существенно сокращает пространство поиска.

3.2. Аугментация данных. Библиотека `pglib-uc` содержит ограниченное число экземпляров на каждый набор данных. Для увеличения выборки применяется мультипликативная аугментация:

$$D'_t = D_t \cdot s \cdot (1 + \varepsilon_t), \quad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2), \quad (5)$$

где $s \sim \text{Uniform}(s_{\min}, s_{\max})$ — масштабирующий коэффициент, σ_ε — стандартное отклонение шума.

Параметры $s_{\min}$, $s_{\max}$, σ_ε подбираются с учетом разброса спроса в наборе данных: чем шире вариация исторических профилей, тем шире диапазоны аугментации. Каждая аугментированная копия решается точным решателем для получения обучающих меток.

3.3. Порог уверенности. Каждая обученная модель $M_{g,t}$ возвращает величину $\hat{p}_{g,t} \in [0, 1]$, которую можно интерпретировать как оцениваемую моделью условную вероятность того, что генератор g в период t окажется включен при заданном векторе признаков. Далее эта величина обозначается $\Pr(u_{g,t} = 1 \mid \mathbf{x})$.

Порог уверенности P_{th} (аналог из [2]) контролирует агрессивность фиксации. Генератор g фиксируется как постоянно включенный, только если для всех периодов t выполняется неравенство

$$\Pr(u_{g,t} = 1 \mid \mathbf{x}) > P_{\text{th}}, \quad (6)$$

и как постоянно выключенный, только если справедливо условие

$$\Pr(u_{g,t} = 1 \mid \mathbf{x}) < 1 - P_{\text{th}}. \quad (7)$$

Фиксация происходит только при единогласном решении всех моделей $\{M_{g,t}\}_{t \in \mathcal{T}}$ для генератора g , то есть когда во всех периодах выполняется условие (6) или (7). Порог P_{th} настраивает чувствительность отдельной модели. При $P_{\text{th}} = 0.5$ для каждой модели $M_{g,t}$ достаточно, чтобы предсказанная ею вероятность выбранного класса превышала 0.5, пусть даже незначительно, тогда как более высокие значения порога требуют более уверенного предсказания. Чем выше P_{th} , тем меньше генераторов фиксируется и тем ниже риск ошибочной фиксации. Однако в задаче смешанного целочисленного линейного программирования остается больше свободных бинарных переменных, поэтому ускорение оказывается меньше. Значение порога подбирается таким образом, чтобы обеспечить компромисс между ускорением и качеством получаемого решения.

3.4. Учет начальных условий. При фиксации генераторов необходимо учитывать их состояние на момент начала планирования ($t = 0$). Например, если генератор был включен и еще не отработал минимально необходимое время UT_g^{remain} , его нельзя зафиксировать как выключенный в ближайшие периоды, поскольку это нарушило бы ограничения минимального времени работы. Аналогичное правило действует для генераторов, находящихся в простое.

Для корректной обработки таких ситуаций фиксация каждого генератора начинается не с первого периода, а с момента t_{start} , после которого начальные условия уже не накладывают ограничений на переключение.

3.5. Общий алгоритм. Разработанный здесь метод предполагает наличие двух фаз. Первая фаза состоит в обучении классификаторов $\{M_{g,t}\}$, которое выполняется однократно на исторических данных в соответствии с разделами 3.2–3.3. Вторая фаза заключается в применении уже обученных моделей к каждому новому экземпляру задачи. Она выполняется в два этапа: сначала генераторы классифицируются (алгоритм 1), затем решается задача с учетом фиксаций (алгоритм 2).

Множество $\mathcal{G}_{\text{on}}$ объединяет генераторы, состояние которых фиксируется как включенное во всех периодах, $\mathcal{G}_{\text{off}}$ — фиксируемые как выключенные, $\mathcal{G}_{\text{free}}$ — оставленные свободными для решателя. Величина $\hat{p}_{g,t}$ обозначает предсказанную моделью $M_{g,t}$ вероятность того, что $u_{g,t} = 1$.

4. Методы классификации. В работе сравниваются шесть алгоритмов бинарной классификации и один наивный классификатор. Каждый алгоритм обучается для отдельной пары (генератор, период) и предсказывает $u_{g,t} \in \{0, 1\}$ по вектору признаков $\mathbf{x} \in \mathbb{R}^{2T}$.

4.1. Сравнимые алгоритмы. Для каждого алгоритма ниже описаны принцип построения решающего правила и способ вычисления вероятностной оценки $\hat{p}_{g,t}$, к которой применяется порог уверенности из раздела 3.3.

Алгоритм 1. Классификация генераторов

Algorithm 1. Generator classification

Вход: признаки $\mathbf{x}$, обученные модели $\{M_{g,t}\}$, порог P_{th} , начальные условия**Выход:** множества $\mathcal{G}_{on}, \mathcal{G}_{off}, \mathcal{G}_{free}$

```

1:  $\mathcal{G}_{on}, \mathcal{G}_{off}, \mathcal{G}_{free} := \emptyset$ 
2: для каждого генератора  $g \in \mathcal{G}$  выполнить
3:   для всех  $t$  вычислить  $\hat{p}_{g,t} := M_{g,t}(\mathbf{x})$ 
4:   определить  $t_{start}$  из начальных условий генератора  $g$ 
5:   если  $\hat{p}_{g,t} > P_{th}$  для всех  $t \geq t_{start}$  тогда
6:      $\mathcal{G}_{on} := \mathcal{G}_{on} \cup \{g\}$ 
7:   иначе если  $\hat{p}_{g,t} < 1 - P_{th}$  для всех  $t \geq t_{start}$  тогда
8:      $\mathcal{G}_{off} := \mathcal{G}_{off} \cup \{g\}$ 
9:   иначе
10:     $\mathcal{G}_{free} := \mathcal{G}_{free} \cup \{g\}$ 
11:   конец если
12: конец для

```

Алгоритм 2. ML-ускоренное решение UC

Algorithm 2. ML-accelerated UC solution

Вход: данные экземпляра $\mathcal{D}$, обученные модели $\{M_{g,t}\}$, порог P_{th} **Выход:** решение UC

```

1: извлечь признаки  $\mathbf{x}$  из  $\mathcal{D}$  по формуле (4)
2:  $\mathcal{G}_{on}, \mathcal{G}_{off}, \mathcal{G}_{free} :=$  Алгоритм 1( $\mathbf{x}, \{M_{g,t}\}, P_{th}$ )
3: зафиксировать  $u_{g,t} = 1$  для  $g \in \mathcal{G}_{on}$ ,  $u_{g,t} = 0$  для  $g \in \mathcal{G}_{off}$ 
4: решить MILP со свободными переменными для  $g \in \mathcal{G}_{free}$ 
5: вернуть решение

```

§ 4.1.1. Логистическая регрессия (LR).

Логистическая регрессия [12] — линейная модель бинарной классификации, в которой вероятность принадлежности к положительному классу моделируется логистической (сигмоидной) функцией:

$$\Pr(u_{g,t} = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + b)}}, \quad (8)$$

где $\mathbf{w} \in \mathbb{R}^{2T}$ — вектор весов, b — смещение, $\sigma(\cdot)$ — сигмоидная функция. Параметры $\mathbf{w}$ и b в (8) оптимизируются максимизацией правдоподобия с L_2 -регуляризацией. Обратная сила регуляризации задается параметром C . Решающей границей LR служит гиперплоскость в пространстве признаков. В работе [2] LR использовалась как единственный классификатор.

§ 4.1.2. Случайный лес (RF).

Случайный лес [13] объединяет множество деревьев решений в единый ансамбль. Алгоритм строит B деревьев, каждое из которых обучается на случайной подвыборке обучающих данных. При этом в каждом узле дерева рассматривается случайное подмножество из m признаков (по умолчанию $m = \lfloor \sqrt{2T} \rfloor$). Случайность подвыборок и подмножеств признаков снижает корреляцию между деревьями. Каждое дерево возвращает оценку вероятности класса “включен”, равную доле обучающих примеров этого класса в листе, в который попадает $\mathbf{x}$. Итоговая оценка ансамбля получается усреднением по всем деревьям:

$$\Pr(u_{g,t} = 1 \mid \mathbf{x}) = \frac{1}{B} \sum_{i=1}^B h_i(\mathbf{x}), \quad (9)$$

где $h_i(\mathbf{x}) \in [0, 1]$ — оценка вероятности $u_{g,t} = 1$, выдаваемая i -м деревом, B — общее число деревьев в ансамбле.

Каждое из деревьев h_i в (9) рекурсивно разбивает пространство признаков по пороговым значениям отдельных признаков, выбирая разбиение, максимизирующее информационный выигрыш (критерий Джини). Максимальная глубина дерева ограничена сверху и задается гиперпараметром.



§ 4.1.3. Градиентный бустинг (GB, CB, HGB).

Градиентный бустинг [14] строит ансамбль деревьев решений последовательно, так что каждое новое дерево корректирует ошибки предыдущих:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \eta \cdot h_m(\mathbf{x}), \quad (10)$$

где F_m — ансамбль после m -й итерации, h_m — дерево, обученное на антиградиенте функции потерь, η — скорость обучения. В отличие от случайного леса, где деревья строятся независимо, в бустинге каждое дерево целенаправленно уменьшает остаточную ошибку предыдущих. Для бинарной классификации оценка вероятности получается применением сигмоидной функции к выходу ансамбля (10): $\Pr(u_{g,t} = 1 \mid \mathbf{x}) = \sigma(F_m(\mathbf{x}))$. В данной работе сравниваются три реализации этого алгоритма:

1. Классический градиентный бустинг (GB) соответствует оригинальному алгоритму Фридмана [14]. В каждом узле дерева перебираются все уникальные значения каждого признака как кандидаты на разбиение.

2. CatBoost (CB) [15] отличается от классического алгоритма двумя ключевыми механизмами. Во-первых, используется упорядоченный бустинг: статистики разбиения для каждого примера вычисляются только по предшествующим ему в случайной перестановке примерам. Во-вторых, применяются симметричные деревья, у которых все узлы одного уровня используют одно и то же условие разбиения. Кроме того, для компенсации дисбаланса классов примерам назначаются веса, обратно пропорциональные частотам их классов в обучающей выборке.

3. Гистограммный градиентный бустинг (HGB) ускоряет обучение за счет предварительного разбиения значений каждого признака на фиксированное число интервалов (гистограмму): кандидатами на разбиение служат не все уникальные значения признака, как в классическом GB, а лишь границы интервалов. Это существенно сокращает время построения деревьев при незначительной потере точности. В работе используется реализация `HistGradientBoostingClassifier` из `scikit-learn`. Тот же принцип лежит в основе широко применяемой библиотеки `LightGBM`. Как и в CB, для компенсации дисбаланса классов примеры взвешиваются обратно пропорционально частотам их классов.

§ 4.1.4. Многослойный перцептрон (MLP).

Многослойный перцептрон [16] относится к нейронным сетям прямого распространения с полносвязными слоями. В данной работе используется архитектура с двумя скрытыми слоями и функцией активации ReLU:

$$\mathbf{h}_1 = \text{ReLU}(W_1 \mathbf{x} + \mathbf{b}_1), \quad \mathbf{h}_2 = \text{ReLU}(W_2 \mathbf{h}_1 + \mathbf{b}_2), \quad \hat{p} = \sigma(W_3 \mathbf{h}_2 + \mathbf{b}_3), \quad (11)$$

где $\hat{p}$ — оценка $\Pr(u_{g,t} = 1 \mid \mathbf{x})$, $\text{ReLU}(z) = \max(0, z)$, σ — сигмоидная функция, $W_i, \mathbf{b}_i$ — весовые матрицы и векторы смещений соответствующих слоев. Параметры $\{W_i, \mathbf{b}_i\}$ сети (11) оптимизируются методом Adam [17] с минимизацией бинарной перекрестной энтропии. Число итераций обучения ограничено сверху и задается гиперпараметром.

§ 4.1.5. Метод опорных векторов (SVM).

Метод опорных векторов [18] ищет в пространстве признаков разделяющую границу с максимальным отступом между классами. Для нелинейно разделимых данных применяется ядерное преобразование, неявно отображающее признаки в пространство более высокой размерности. В данной работе используется ядро на основе радиальной базисной функции (RBF):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \quad (12)$$

где $\gamma = 1/(2T \cdot \text{Var}[\mathbf{x}])$.

Сам SVM выдает не вероятность, а вещественное значение, знак которого определяет предсказанный класс, а абсолютная величина показывает удаленность точки от разделяющей границы. Для получения вероятностных оценок, необходимых для порога уверенности, применяется калибровка Платта [19]: поверх выхода SVM обучается логистическая регрессия.

§ 4.1.6. Метод k ближайших соседей (KNN).

Метод k ближайших соседей [20] не строит явной модели. При предсказании алгоритм находит k обучающих примеров, ближайших к тестовому в пространстве признаков, и определяет класс их голо-

сованием:

$$\Pr(u_{g,t} = 1 \mid \mathbf{x}) = \frac{\sum_{i \in N_k(\mathbf{x})} w_i \cdot \mathbf{1}[y_i = 1]}{\sum_{i \in N_k(\mathbf{x})} w_i}, \quad (13)$$

где $N_k(\mathbf{x})$ — множество k ближайших соседей тестового примера $\mathbf{x}$ среди обучающих, $y_i \in \{0, 1\}$ — метка класса i -го обучающего примера, $\mathbf{1}[\cdot]$ — индикаторная функция, $d(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{x} - \mathbf{x}_i\|_2$ — евклидово расстояние, $w_i = 1/d(\mathbf{x}, \mathbf{x}_i)$ — вес соседа. Согласно (13), чем ближе обучающий пример к классифицируемому, тем сильнее его метка влияет на итоговую оценку вероятности.

§ 4.1.7. Наивный классификатор (Naive).

В качестве нижней границы эффективности введен наивный классификатор, не использующий входные признаки. Для каждой пары (генератор, период) классификатор анализирует обучающую выборку: если генератор был включен во всех обучающих экземплярах, предсказанная вероятность равна 1.0; если выключен во всех экземплярах, то равна 0.0; в остальных случаях вероятность полагается равной 0.5.

4.2. Обоснование выбора методов. Шесть методов покрывают основные семейства алгоритмов классификации: линейные модели (LR — базовый метод из [2]), ансамбли деревьев решений (RF и три реализации градиентного бустинга GB, CB и HGB), нейронные сети (MLP), ядерные методы (SVM) и непараметрические методы без фазы обучения (KNN).

Наивный классификатор (Naive) задает нижнюю границу, не использующую признаки. Такой набор позволяет определить, какой класс алгоритмов лучше предсказывает состояние генераторов при ограниченном объеме данных.

5. Вычислительный эксперимент. Ниже описаны использованные наборы данных, процедура перекрестной проверки и программно-аппаратная среда. После чего приводятся результаты сравнения методов классификации по временным затратам на решение, по способности сохранять покрытие спроса и устойчивости при росте размерности задачи, а также анализ влияния настройки гиперпараметров и выбора решателя.

5.1. Методика эксперимента. Все методы сравниваются в одинаковых условиях: на общих наборах данных, по единому протоколу перекрестной проверки и в одной программно-аппаратной среде.

§ 5.1.1. Наборы данных.

Эксперименты проведены на трех наборах данных из библиотеки pglib-uc [11], характеристики которых приведены в табл. 1.

Таблица 1. Характеристики наборов данных pglib-uc

Table 1. Characteristics of pglib-uc datasets

Набор Dataset	Число примеров Number of instances	Число генераторов Number of generators	Число бинарных переменных Number of binary variables	Пиковый спрос, МВт Peak demand, MW
RTS-GMLC	12	73	10 512	4 328–8 018
CA	20	610	87 840	26 623–44 215
FERC	24	934–1028	134 496–140 832	84 123–138 995

Набор FERC содержит 24 экземпляра, но реальных профилей спроса 12, так как они разбиваются на пары, которые отличаются только параметрами возобновляемых источников. Поскольку в используемой модели возобновляемые источники отключены, такие пары неотличимы. Отношение максимального пикового спроса к минимальному составляет 1.65, что затрудняет обобщение.

§ 5.1.2. Протокол оценки.

Для оценки используется перекрестная проверка с исключением одного экземпляра: для каждого из N экземпляров модели обучаются на остальных $N - 1$ (с учетом аугментаций) и тестируются на оставшемся экземпляре.

Качество оценивается по трем метрикам:

- Ускорение: отношение времени стандартного решения к времени ML-ускоренного решения;



- Отклонение целевой функции: $\frac{|z_{\text{ML}} - z_{\text{std}}|}{z_{\text{std}}} \cdot 100\%$;
- Доля зафиксированных переменных: процент зафиксированных бинарных переменных от общего числа.

Время решения (стандартного и ML-ускоренного) измеряется как среднее по 10 независимым запускам решателя. Относительное стандартное отклонение времени решения не превышает 5%, что свидетельствует о воспроизводимости измерений.

§ 5.1.3. Программная и аппаратная среда.

Реализация выполнена на Python 3.13 с использованием следующих библиотек: Pyomo 6.9 [21] для формулировки MILP, scikit-learn 1.7 [22] для классификаторов LR, RF, GB, MLP, SVM, KNN и CatBoost 1.2 для СВ. В качестве решателя MILP используется HiGHS 1.11 [23] (через интерфейс `appsi_highs`) с допустимым отклонением (MIP gap) 1%. Чтобы исключить конкуренцию за ресурсы с решателем, все классификаторы обучались и применялись в однопоточном режиме. Эксперименты проведены на сервере с двумя процессорами AMD EPYC 7502 (128 потоков) и 472 ГБ оперативной памяти под управлением Ubuntu 20.04.

Параметры алгоритма установлены следующим образом: порог уверенности $P_{\text{th}} = 0.95$, число аугментаций на каждый оригинальный экземпляр $N_{\text{aug}} = 5$. Параметры аугментации (5) зависят от разброса спроса в наборе данных: для FERC использованы широкий диапазон $s \in [0.5, 1.5]$ и $\sigma_\varepsilon = 0.05$; для СА — узкий диапазон $s \in [0.9, 1.1]$ и $\sigma_\varepsilon = 0.03$. Размерность вектора признаков при числе периодов $T = 48$ (горизонт планирования 48 ч с шагом 1 ч) составляет 96. Гиперпараметры классификаторов приведены в табл. 2. Все классификаторы используются с гиперпараметрами по умолчанию соответствующих библиотек: приводимое сравнение отражает поведение методов при типичных настройках, а не результат их индивидуальной оптимизации.

Таблица 2. Методы классификации и их гиперпараметры

Table 2. Classification methods and their hyperparameters

Обозначение Notation	Метод Method	Библиотека и класс Library and class	Гиперпараметры Hyperparameters
LR	Логистическая регрессия Logistic regression	scikit-learn LogisticRegression	<code>solver=liblinear</code> , $C = 1.0$
RF	Случайный лес Random forest	scikit-learn RandomForestClassifier	100 деревьев, <code>max_depth=5</code> 100 trees, <code>max_depth=5</code>
GB	Градиентный бустинг Gradient boosting	scikit-learn GradientBoostingClassifier	100 деревьев, <code>depth=4</code> , $\eta = 0.1$ 100 trees, <code>depth=4</code> , $\eta = 0.1$
CB	CatBoost	catboost CatBoostClassifier	200 итераций, <code>depth=4</code> , <code>balanced</code> 200 iterations, <code>depth=4</code> , <code>balanced</code>
HGB	Гистограммный градиентный бустинг Histogram-based gradient boosting	scikit-learn HistGradientBoosting- Classifier	200 итераций, <code>max_depth=4</code> , <code>balanced</code> 200 iterations, <code>max_depth=4</code> , <code>balanced</code>
MLP	Многослойный перцептрон Multilayer perceptron	scikit-learn MLPClassifier	слои (64, 32), ReLU layers (64, 32), ReLU
SVM	Метод опорных векторов Support vector machine	scikit-learn SVC	ядро RBF, $C = 1.0$ RBF kernel, $C = 1.0$
KNN	k ближайших соседей k -nearest neighbors	scikit-learn KNeighborsClassifier	$k = 3$, взвешивание по расстоянию $k = 3$, distance weighting
Naive	Наивный классификатор Naive classifier	собственная реализация own implementation	без гиперпараметров no hyperparameters

Среднее время обучения одной модели на наборе FERC (138 обучающих примеров, 96 признаков) приведено в табл. 3. На наборе FERC для каждой итерации перекрестной проверки обучается 43 640 моделей (из $1028 \times 48 = 49\,344$ пар (генератор, период) 5 706 оказываются константными по обучающей выборке и не требуют обучения), поэтому суммарное время обучения одной итерации составляет от

нескольких минут до нескольких часов. KNN не имеет фазы обучения: все вычисления выполняются на этапе предсказания.

Таблица 3. Время обучения на наборе FERC: среднее на одну модель и суммарное по всем моделям одной итерации перекрестной проверки (43 640 моделей)

Table 3. Training time on the FERC dataset: average per model and total over all models of one cross-validation iteration (43 640 models)

Метод Method	Среднее время обучения на модель, с Mean training time per model, s	Общее время обучения, мин Total training time, min
GB	1.08	786
MLP	0.84	611
CB	0.52	378
HGB	0.45	327
SVM	0.24	175
RF	0.16	116
LR	0.036	26
Naive	<0.004	<3
KNN	—	—

5.2. Результаты и их обсуждение. Основные результаты получены на наиболее крупном наборе FERC; наборы RTS-GMLC и CA используются при анализе влияния размера задачи и настройки гиперпараметров.

§ 5.2.1. Анализ временных затрат.

В табл. 4 приведены агрегированные результаты перекрестной проверки на наборе FERC (24 экземпляра, до 1028 генераторов). Величина $n_{\text{нар}}$ обозначает число экземпляров, на которых ML-решение нарушает баланс спроса, что выражается в активации переменной невязки s_t^D в ограничении (2). Средняя доля зафиксированных бинарных переменных по всем экземплярам обозначена $\bar{F}$.

Таблица 4. Результаты на наборе FERC (24 экземпляра, до 1028 генераторов)

Table 4. Results on the FERC dataset (24 instances, up to 1028 generators)

Метод Method	$\bar{S}$	$\widetilde{\Delta z}$, %	Δz_{max} , %	$n_{\text{нар}}$ n_{break}	$\bar{F}$, %
CB	3.32	0.04	1.40	0	63.6
HGB	3.40	0.02	0.76	0	63.8
GB	2.17	0.02	0.23	0	62.9
RF	2.60	0.02	0.56	0	63.6
KNN	2.73	0.03	0.72	0	63.5
LR	3.99	1.55	4305	9	62.4
MLP	2.89	8.00	2978	15	64.1
SVM	2.75	8.45	7442	13	61.6
Naive	1.33	0.04	0.39	0	1.5

Наивный классификатор фиксирует лишь 1.5% переменных и обеспечивает скромное ускорение $1.3\times$, ни разу не активировав штраф. Максимальное отклонение 0.39% сопоставимо с лучшими методами, что подтверждает безопасность подхода с фиксацией только стабильных генераторов. Все восемь обученных классификаторов фиксируют существенно больше переменных (61–64%), что и обеспечивает им выигрыш в ускорении.

При сопоставимом среднем ускорении (рис. 1a) медианное отклонение (рис. 1b) у CB, RF и KNN

на два порядка ниже, чем у LR, MLP и SVM. Ускорение усредняется арифметически (как относительно стабильная по экземплярам величина), тогда как для отклонения целевой функции используется медиана (на части экземпляров методы LR, MLP и SVM дают выбросы в тысячи процентов из-за ошибочной фиксации критичных генераторов). Медиана устойчива к таким выбросам и отражает типичную точность.

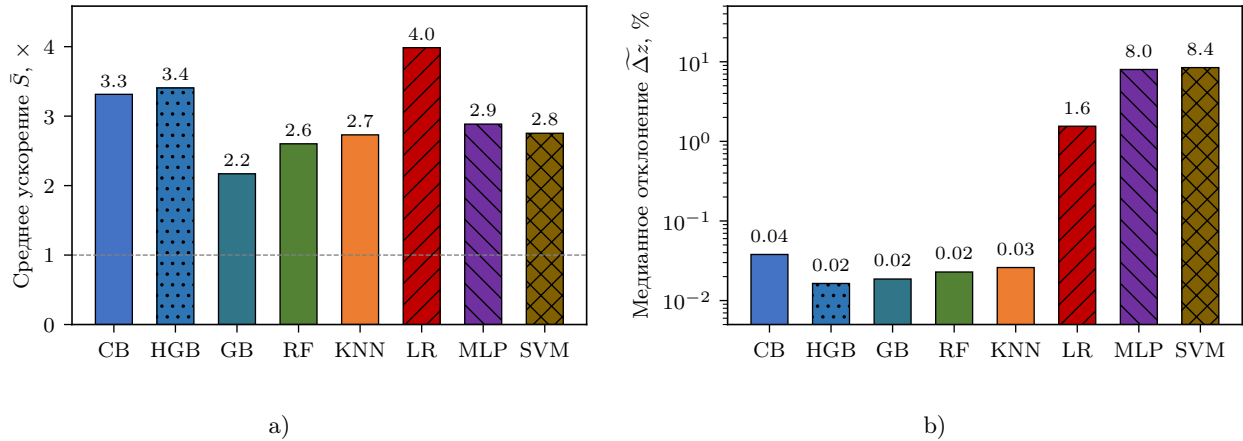


Рис. 1. Характеристики восьми классификаторов на наборе FERC: а) среднее ускорение $\bar{S}$; б) медианное отклонение целевой функции $\tilde{\Delta z}$

Fig. 1. Characteristics of eight classifiers on the FERC dataset: a) average speedup $\bar{S}$; b) median objective deviation $\tilde{\Delta z}$

Пять методов (CB, HGB, GB, RF и KNN) обеспечивают устойчивое ускорение при минимальном отклонении целевой функции. В проведенных расчетах они ни разу не активировали штраф за непокрытие спроса. Наибольшее среднее ускорение показывают гистограммные реализации градиентного бустинга: HGB ($3.4\times$) и CatBoost ($3.3\times$) при медианном отклонении не выше 0.04%. Классическая реализация GB при том же качестве решения (медианное отклонение 0.02%) ускоряет в среднем в 2.2 раза, а случайный лес дает $2.6\times$ при отклонении 0.02%. Все три реализации бустинга фиксируют близкую долю переменных (62–64%). По сравнению с наивным классификатором ($1.3\times$) все пять методов ускоряют решение существенно эффективнее, фиксируя на порядок больше переменных благодаря учету профиля спроса.

Логистическая регрессия, MLP и SVM показывают сопоставимое среднее ускорение ($2.75\text{--}3.99\times$), но на значительной доле экземпляров порождают решения со штрафом за нехватку мощности. Нарушением баланса спроса завершились 9 из 23 рассматриваемых экземпляров у LR, 13 из 23 у SVM и 15 из 21 у MLP.

§ 5.2.2. Анализ прогнозируемого покрытия спроса.

Распределение отклонений по всем экземплярам FERC (рис. 2) показывает, что методы делятся на две группы по способности сохранять покрытие спроса. У CB, HGB, GB, RF и KNN все отклонения остаются ниже 1%, тогда как у LR, MLP и SVM медиана и верхний квартиль превышают 5%.

Большие отклонения у LR, SVM и MLP связаны со структурой целевой функции (1). Если классификатор ошибочно отключает генератор, без которого спрос не покрывается, нехватка компенсируется через переменную s_t^D ценой существенного роста целевой функции: один период с нехваткой 100 МВт добавляет 10^8 при типичных затратах порядка $10^7\text{--}10^8$. Так, на экземпляре 2015-01-01_hw метод SVM получил значение целевой функции $6.8 \cdot 10^9$ при базовом значении $9.0 \cdot 10^7$, то есть отклонение составило 7442%.

Причины ошибочных фиксаций у каждого метода свои. LR строит линейную решающую границу в 96-мерном пространстве признаков, однако для 1028 генераторов с разнородными характеристиками этого оказывается недостаточно: при нетипичном профиле спроса предсказания систематически смещены. SVM с ядром RBF (12) вычисляет расстояния в тех же 96 измерениях по 130 обучающим примерам, поэтому евклидова метрика перестает различать экземпляры, и вероятностные оценки на ее основе оказываются уверенными, но неверными. MLP с 8353 параметрами переобучается на выборке из 130 примеров; ограничение числа итераций (`max_iter=150`) сдерживает эту тенденцию лишь отчасти.

У CB, HGB, GB, RF и KNN ошибочных фиксаций критичных генераторов на FERC не возникало. Деревья решений в RF, GB, CB и HGB моделируют пороговые зависимости вида “генератор включается при спросе выше X МВт”, а усреднение по ансамблю сглаживает разброс предсказаний отдельных дере-

вьев. Классификатор KNN при $k = 3$ опирается напрямую на решения ближайших по профилю спроса обучающих экземпляров, что при небольшом объеме обучающей выборки дает устойчивые оценки. Однако с ростом этой выборки вычислительные затраты KNN на этапе предсказания увеличиваются, поскольку метод лишен фазы обучения и все вычисления выполняет при предсказании, что может ограничить его применимость при большом объеме исторических данных.

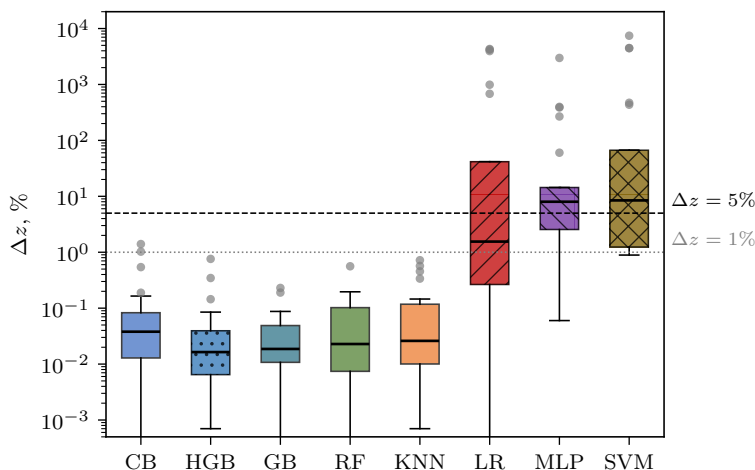


Рис. 2. Распределение отклонений целевой функции на наборе FERC в формате “ящик с усами” (логарифмическая шкала). Ящик — межквартильный размах (Q_1 – Q_3), линия внутри — медиана, усы — $1.5 \cdot IQR$ от ящика, точки — выбросы. Пунктирные линии — пороги 1% и 5%

Fig. 2. Box-and-whisker distribution of objective function deviations on the FERC dataset (logarithmic scale). Box — interquartile range (Q_1 – Q_3), inner line — median, whiskers — $1.5 \cdot IQR$ from the box, points — outliers. Dashed lines indicate thresholds of 1% and 5%

профиля спроса там используются сетевые характеристики (топология передающей сети, загрузка линий, узловые ограничения), которые несут информацию о том, какие именно генераторы критичны для покрытия нагрузки. В постановке `pglib-us` такие сетевые признаки отсутствуют, поэтому логистической регрессии доступен лишь агрегированный спрос.

§ 5.2.3. Влияние размера задачи.

Для оценки влияния масштаба задачи эксперимент проведен с логистической регрессией (базовый метод из [2]) и CatBoost на всех трех наборах данных. Соответствующие результаты продемонстрированы на рис. 3.

На малых системах оба метода показывают сопоставимые результаты. С ростом числа генераторов CatBoost сохраняет стабильное ускорение и низкое отклонение целевой функции, тогда как логистическая регрессия на крупнейшем наборе дает медианное отклонение в 40 раз выше при формально большем ускорении за счет ошибочной фиксации критичных генераторов. Этот вывод относится к логистической регрессии с гиперпараметрами по умолчанию и ограниченным набором признаков в виде агрегированного спроса. В [4] линейные модели остаются работоспособными и на системах большого масштаба, однако, предположительно, это достигается за счет учета существенно более широкого набора признаков: помимо

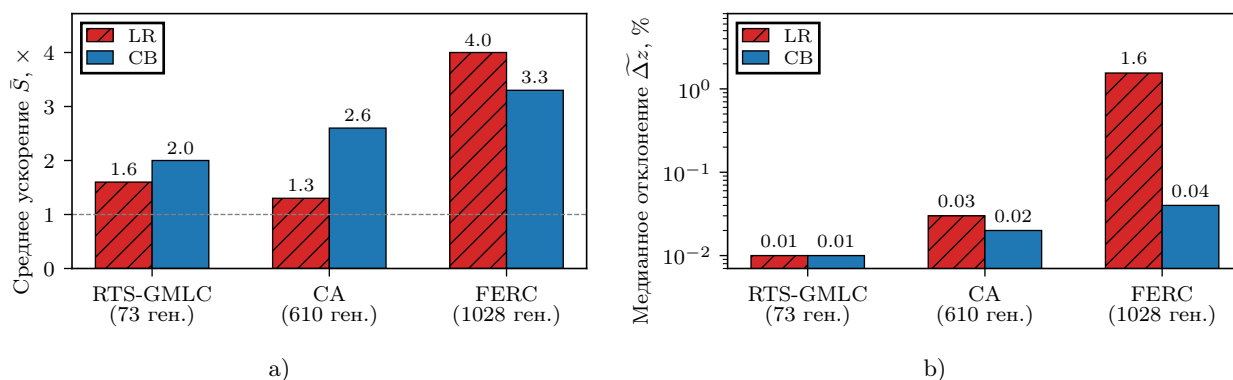


Рис. 3. Сравнение логистической регрессии и CatBoost на трех наборах данных в зависимости от числа генераторов: а) среднее ускорение $\bar{S}$; б) медианное отклонение целевой функции $\tilde{\Delta z}$

Fig. 3. Comparison of logistic regression and CatBoost on three datasets depending on the number of generators: а) average speedup $\bar{S}$; б) median objective deviation $\tilde{\Delta z}$

§ 5.2.4. Влияние настройки гиперпараметров.

Чтобы проверить, не обусловлены ли различия между методами библиотечными настройками по умолчанию, на наименьшем наборе RTS-GMLC проведен перебор гиперпараметров с полной перекрестной проверкой. Для логистической регрессии варьировалась обратная сила регуляризации $C \in \{0.01, 0.1, 1, 10, 100\}$ (а также применялась L_1 -регуляризация вместо L_2), для случайного леса — максимальная глубина и число деревьев, а для двух реализаций градиентного бустинга (CatBoost и HGB) — глубина, число итераций и скорость обучения. Разброс результатов во всех случаях оказался незначительным: у LR среднее ускорение изменялось в пределах $1.67\text{--}1.98\times$ (при настройке по умолчанию $1.81\times$), у RF оно составило $1.93\text{--}2.06\times$ ($1.97\times$), у HGB — $1.81\text{--}1.88\times$ ($1.88\times$), у CatBoost — $1.85\text{--}2.27\times$ ($2.27\times$), а медианное отклонение целевой функции во всех конфигурациях оставалось ниже 0.03%. У лучших методов (CatBoost и HGB) наибольшее ускорение достигалось уже при настройках по умолчанию, и перебор гиперпараметров его не улучшал. Таким образом, настройка не изменила ни ранжирования методов, ни качественных выводов, и сравнение при настройках по умолчанию можно считать репрезентативным.

§ 5.2.5. Влияние выбора решателя.

Наблюдаемое ускорение отчасти могло бы объясняться слабостью предобработки (presolve) открытого решателя HiGHS: коммерческие решатели с более агрессивной предобработкой и генерацией отсечений способны самостоятельно устранять часть переменных, уменьшая выигрыш от фиксации. Чтобы это проверить, на наборе FERC стандартная и ML-урезанная задачи (с одинаковой фиксацией методом HGB) решались тремя решателями: открытым HiGHS, коммерческими CPLEX и Gurobi.

По абсолютному времени коммерческие решатели заметно быстрее: HiGHS решает задачу в среднем за 3240 с, тогда как Gurobi — за 2726 с, а CPLEX — за 2855 с. Фиксация ускоряет решение каждого из них в 2.9–3.4 раза: среднее время падает до 981 с у HiGHS, до 892 с у Gurobi и до 930 с у CPLEX. Медианное отклонение целевой функции во всех случаях остается ниже 1%. Сопоставимый относительный выигрыш у всех трех решателей показывает, что он обеспечивается сокращением пространства поиска, а не слабостью конкретного решателя.

Хотя открытый HiGHS уступает коммерческим решателям в абсолютном времени, он свободно распространяется и не накладывает лицензионных ограничений. Это делает его практичным выбором для эксплуатации в условиях больших размерностей задачи и ограниченного доступа к коммерческим лицензиям. Предложенный метод ускорения при этом остается полезным независимо от выбранного решателя.

6. Заключение. Предложенный метод ML-ускорения задачи выбора состава включенного генерирующего оборудования протестирован на наборах данных pglib-uc с числом генераторов от 73 до 1028. Шесть алгоритмов классификации и наивный классификатор предсказывали состояние генераторов, после чего уверенно классифицированные генераторы фиксировались, а решатель работал с задачей уменьшенной размерности.

Лучшие результаты в сочетании ускорения и качества решения показали методы на основе деревьев (гистограммный градиентный бустинг HGB, CatBoost, случайный лес) и метод ближайших соседей. Они обеспечивают наибольшее ускорение при минимальном отклонении целевой функции и не приводят к фиксации, нарушающим покрытие спроса. Сравнение трех реализаций градиентного бустинга (классической, CatBoost и гистограммной HGB) показало, что при сопоставимом качестве решения наибольшее ускорение дают гистограммная реализация ($3.4\times$) и CatBoost ($3.3\times$). Таким образом, специализированные механизмы CatBoost не обеспечивают заметного преимущества перед стандартной гистограммной реализацией в случае рассматриваемой задачи. Логистическая регрессия (базовый метод из [2]), а также MLP и SVM устойчиво работают на малых системах, но при масштабировании до 1028 генераторов теряют надежность. Они приводят к ошибочной фиксации критичных генераторов и резкому росту целевой функции.

Проведена дополнительная проверка на коммерческих решателях CPLEX и Gurobi, которая показала, что ускорение от фиксации сохраняется.

Дальнейшая работа предполагает адаптацию порога уверенности P_{th} к конкретному экземпляру, задачу ML-решения в решатель через начальное приближение (схема “теплого старта”) вместо жесткой фиксации переменных и расширение обучающих данных за счет аугментации с учетом сезонных закономерностей спроса.

Список литературы

1. *Knueven B., Ostrowski J., Watson J.-P.* On mixed-integer programming formulations for the unit commitment problem // *INFORMS Journal on Computing*. 2020. **32**, N 4. 857–876. doi [10.1287/ijoc.2019.0944](https://doi.org/10.1287/ijoc.2019.0944).
2. *Ramesh A.V., Li X.* Machine learning assisted model reduction for security-constrained unit commitment // 2022 North American Power Symposium (NAPS), Salt Lake City, UT, USA, October 9–11, 2022. New York: IEEE Press, 2022. pp. 1–6. doi [10.1109/NAPS56150.2022.10012236](https://doi.org/10.1109/NAPS56150.2022.10012236).
3. *Bengio Y., Lodi A., Prouvost A.* Machine learning for combinatorial optimization: a methodological tour d’horizon // *European Journal of Operational Research*. 2021. **290**, N 2. 405–421. doi [10.1016/j.ejor.2020.07.063](https://doi.org/10.1016/j.ejor.2020.07.063).
4. *Xavier A.S., Qiu F., Ahmed S.* Learning to solve large-scale security-constrained unit commitment problems // *INFORMS Journal on Computing*. 2020. **33**, N 2. 739–756. doi [10.1287/ijoc.2020.0976](https://doi.org/10.1287/ijoc.2020.0976).
5. *Park S., Chen W., Han D., et al.* Confidence-aware graph neural networks for learning reliability assessment commitments // *IEEE Transactions on Power Systems*. 2024. **39**, N 2. 3839–3850. doi [10.1109/TPWRS.2023.3298735](https://doi.org/10.1109/TPWRS.2023.3298735).
6. *Pourahmadi F., Kazempour J.* Unit commitment predictor with a performance guarantee: a support vector machine classifier // *IEEE Transactions on Power Systems*. 2025. **40**, N 1. 715–727. doi [10.1109/TPWRS.2024.3400405](https://doi.org/10.1109/TPWRS.2024.3400405).
7. *Sugishita N., Grothey A., McKinnon K.* Use of machine learning models to warmstart column generation for unit commitment // *INFORMS Journal on Computing*. 2024. **36**, N 4. 1129–1146. doi [10.1287/ijoc.2022.0140](https://doi.org/10.1287/ijoc.2022.0140).
8. *Pineda S., Morales J.M.* Is learning for the unit commitment problem a low-hanging fruit? // *Electric Power Systems Research*. 2022. **207**. Article Number 107851. doi [10.1016/j.epsr.2022.107851](https://doi.org/10.1016/j.epsr.2022.107851).
9. *Yang Y., Wu L.* Machine learning approaches to the unit commitment problem: current trends, emerging challenges, and new strategies // *The Electricity Journal*. 2021. **34**, N 1. Article Number 106889. doi [10.1016/j.tej.2020.106889](https://doi.org/10.1016/j.tej.2020.106889).
10. *Milla J., Pérez-Díaz J.I.* The use of artificial intelligence for the unit commitment problem: state of the art // *ELECTRIMACS 2024. Lecture Notes in Electrical Engineering*, Vol. 1275. Cham: Springer, 2025. pp. 713–725. doi [10.1007/978-3-031-73921-7_54](https://doi.org/10.1007/978-3-031-73921-7_54).
11. Power Grid Lib — Unit Commitment. <https://github.com/power-grid-lib/pglib-uc>. Cited October 8, 2026.
12. *Hosmer D.W., Lemeshow S., Sturdivant R.X.* Applied logistic regression. 3rd ed. Hoboken: Wiley, 2013. doi [10.1002/9781118548387](https://doi.org/10.1002/9781118548387).
13. *Breiman L.* Random forests // *Machine Learning*. 2001. **45**, N 1. 5–32. doi [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324).
14. *Friedman J.H.* Greedy function approximation: a gradient boosting machine // *The Annals of Statistics*. 2001. **29**, N 5. 1189–1232. doi [10.1214/aos/1013203451](https://doi.org/10.1214/aos/1013203451).
15. *Prokhorenkova L., Gusev G., Vorobev A., et al.* CatBoost: unbiased boosting with categorical features // *Advances in Neural Information Processing Systems*. 2018. **31**. 6638–6648. <https://arxiv.org/abs/1706.09516>. Cited October 8, 2026.
16. *Rumelhart D.E., Hinton G.E., Williams R.J.* Learning representations by back-propagating errors // *Nature*. 1986. **323**. 533–536. doi [10.1038/323533a0](https://doi.org/10.1038/323533a0).
17. *Kingma D.P., Ba J.* Adam: a method for stochastic optimization // arXiv:1412.6980. 2014. <https://arxiv.org/abs/1412.6980>. Cited October 8, 2026.
18. *Cortes C., Vapnik V.* Support-vector networks // *Machine Learning*. 1995. **20**, N 3. 273–297. doi [10.1007/BF00994018](https://doi.org/10.1007/BF00994018).
19. *Platt J.C.* Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods // *Advances in Large Margin Classifiers*. 1999. 61–74.
20. *Cover T., Hart P.* Nearest neighbor pattern classification // *IEEE Transactions on Information Theory*. 1967. **13**, N 1. 21–27. doi [10.1109/TIT.1967.1053964](https://doi.org/10.1109/TIT.1967.1053964).
21. *Hart W.E., Watson J.-P., Woodruff D.L.* Pyomo: modeling and solving mathematical programs in Python // *Mathematical Programming Computation*. 2011. **3**, N 3. 219–260. doi [10.1007/s12532-011-0026-8](https://doi.org/10.1007/s12532-011-0026-8).
22. *Pedregosa F., Varoquaux G., Gramfort A., et al.* Scikit-learn: machine learning in Python // *Journal of Machine Learning Research*. 2011. **12**. 2825–2830. <https://jmlr.org/papers/v12/pedregosa11a.html>. Cited October 9, 2026.
23. *Huangfu Q., Hall J.A.J.* Parallelizing the dual revised simplex method // *Mathematical Programming Computation*. 2018. **10**, N 1. 119–142. doi [10.1007/s12532-017-0130-5](https://doi.org/10.1007/s12532-017-0130-5).



Получена
 21 июня 2026 г.

Принята
 31 августа 2026 г.

Опубликована
 21 октября 2026 г.

Информация об авторе

Сергей Александрович Романов — аспирант; Омский филиал Института математики имени С. Л. Соболева СО РАН, ул. Певцова, 13, 644043, Омск, Российская Федерация.

References

1. B. Knueven, J. Ostrowski, and J.-P. Watson, “On Mixed-Integer Programming Formulations for the Unit Commitment Problem,” *INFORMS J. Comput.* **32** (4), 857–876 (2020). doi [10.1287/ijoc.2019.0944](https://doi.org/10.1287/ijoc.2019.0944).
2. A. V. Ramesh and X. Li, “Machine Learning Assisted Model Reduction for Security-Constrained Unit Commitment,” in *2022 North American Power Symposium (NAPS), Salt Lake City, UT, USA, October 9–11, 2022* (IEEE Press, New York, 2022), pp. 1–6. doi [10.1109/NAPS56150.2022.10012236](https://doi.org/10.1109/NAPS56150.2022.10012236).
3. Y. Bengio, A. Lodi, and A. Prouvost, “Machine Learning for Combinatorial Optimization: A Methodological Tour d’Horizon,” *Eur. J. Oper. Res.* **290** (2), 405–421 (2021). doi [10.1016/j.ejor.2020.07.063](https://doi.org/10.1016/j.ejor.2020.07.063).
4. A. S. Xavier, F. Qiu, and S. Ahmed, “Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems,” *INFORMS J. Comput.* **33** (2), 739–756 (2020). doi [10.1287/ijoc.2020.0976](https://doi.org/10.1287/ijoc.2020.0976).
5. S. Park, W. Chen, D. Han, et al., “Confidence-Aware Graph Neural Networks for Learning Reliability Assessment Commitments,” *IEEE Trans. Power Syst.* **39** (2), 3839–3850 (2024). doi [10.1109/TPWRS.2023.3298735](https://doi.org/10.1109/TPWRS.2023.3298735).
6. F. Pourahmadi and J. Kazempour, “Unit Commitment Predictor with a Performance Guarantee: A Support Vector Machine Classifier,” *IEEE Trans. Power Syst.* **40** (1), 715–727 (2025). doi [10.1109/TPWRS.2024.3400405](https://doi.org/10.1109/TPWRS.2024.3400405).
7. N. Sugishita, A. Grothey, and K. McKinnon, “Use of Machine Learning Models to Warmstart Column Generation for Unit Commitment,” *INFORMS J. Comput.* **36** (4), 1129–1146 (2024). doi [10.1287/ijoc.2022.0140](https://doi.org/10.1287/ijoc.2022.0140).
8. S. Pineda and J. M. Morales, “Is learning for the unit commitment problem a low-hanging fruit?,” *Electric Power Syst. Res.* **207**, Article Number 107851 (2022). doi [10.1016/j.epsr.2022.107851](https://doi.org/10.1016/j.epsr.2022.107851).
9. Y. Yang and L. Wu, “Machine Learning Approaches to the Unit Commitment Problem: Current Trends, Emerging Challenges, and New Strategies,” *Electr. J.* **34** (1), Article Number 106889 (2021). doi [10.1016/j.tej.2020.106889](https://doi.org/10.1016/j.tej.2020.106889).
10. J. Milla and J. I. Pérez-Díaz, “The Use of Artificial Intelligence for the Unit Commitment Problem: State of the Art,” in *ELECTRIMACS 2024, Lecture Notes in Electrical Engineering* (Springer, Cham, 2025), Vol. 1275, pp. 713–725. doi [10.1007/978-3-031-73921-7_54](https://doi.org/10.1007/978-3-031-73921-7_54).
11. Power Grid Lib — Unit Commitment. <https://github.com/power-grid-lib/pglib-uc>. Cited October 8, 2026.
12. D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. (Wiley, Hoboken, 2013). doi [10.1002/9781118548387](https://doi.org/10.1002/9781118548387).
13. L. Breiman, “Random Forests,” *Machine Learning* **45** (1), 5–32 (2001). doi [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324).
14. J. H. Friedman, “Greedy Function Approximation: A Gradient Boosting Machine,” *Ann. Statist.* **29** (5), 1189–1232 (2001). doi [10.1214/aos/1013203451](https://doi.org/10.1214/aos/1013203451).
15. L. Prokhorenkova, G. Gusev, A. Vorobev, et al., “CatBoost: Unbiased Boosting with Categorical Features,” *Advances in Neural Information Processing Systems* **31**, 6638–6648 (2018). <https://arxiv.org/abs/1706.09516>. Cited October 8, 2026.
16. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature* **323**, 533–536 (1986). doi [10.1038/323533a0](https://doi.org/10.1038/323533a0).
17. D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” arXiv:1412.6980 (2014). <https://arxiv.org/abs/1412.6980>. Cited October 8, 2026.
18. C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning* **20** (3), 273–297 (1995). doi [10.1007/BF00994018](https://doi.org/10.1007/BF00994018).
19. J. C. Platt, “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods,” *Advances in Large Margin Classifiers* 61–74 (1999).
20. T. Cover and P. Hart, “Nearest Neighbor Pattern Classification,” *IEEE Trans. Inf. Theory* **13** (1), 21–27 (1967). doi [10.1109/TIT.1967.1053964](https://doi.org/10.1109/TIT.1967.1053964).
21. W. E. Hart, J.-P. Watson, and D. L. Woodruff, “Pyomo: modeling and solving mathematical programs in Python,” *Math. Program. Comput.* **3** (3), 219–260 (2011). doi [10.1007/s12532-011-0026-8](https://doi.org/10.1007/s12532-011-0026-8).

22. F. Pedregosa, G. Varoquaux, A. Gramfort, et al., “Scikit-learn: Machine Learning in Python,” J. Mach. Learn. Res. **12**, 2825–2830 (2011). <https://jmlr.org/papers/v12/pedregosa11a.html>. Cited October 9, 2026.
23. Q. Huangfu and J. A. J. Hall, “Parallelizing the Dual Revised Simplex Method,” Math. Program. Comput. **10** (1), 119–142 (2018). doi [10.1007/s12532-017-0130-5](https://doi.org/10.1007/s12532-017-0130-5).

Received
June 21, 2026

Accepted
August 31, 2026

Published
October 21, 2026

Information about the author

Sergey A. Romanov — PhD; Omsk Department of Sobolev Institute of Mathematics SB RAS, Pevtsova ulitsa, 13, 644043, Omsk, Russia.