ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

**Программа подготовки бакалавров по направлению
Компьютерные науки и технологии**

Мухаметшин Руслан Рашидович

КУРСОВАЯ РАБОТА

Интерактивный конструктор моделей искусственного интеллекта с поддержкой мультимодальных задач.
Создание пайплайнов для реализации Classic ML

Научный руководитель
старший преподаватель НИУ
ВШЭ - НН

Саратовцев Артём Романович

Нижний Новгород, 2025г.

Структура работы

1. Введение

Современный этап развития машинного обучения характеризуется активным внедрением технологий в различные сферы человеческой деятельности: от медицины и финансов до промышленности и развлечений. Однако разработка и обучение ML моделей остаются сложными задачами, требующими глубоких знаний в программировании, математике и обработке данных. Это создаёт барьер для специалистов, не обладающих техническим бэкграундом, но заинтересованных в применении ML для решения своих задач.

В этом контексте особую актуальность приобретают No-code платформы и AutoML решения, которые автоматизируют ключевые этапы работы с ML: от предобработки данных до выбора оптимальной модели и её настройки. Такие системы не только ускоряют разработку, но и делают технологии ИИ доступными для широкого круга пользователей.

Целью данной работы является разработка интерактивного конструктора ML моделей, а также реализация и сравнение алгоритмов AutoML, предназначенных для подбора гиперпараметров моделей.

Задачи проекта:

1. Провести анализ существующих алгоритмов AutoML для подбора гиперпараметров.
2. Разработать и сравнить несколько алгоритмов AutoML
3. Реализовать удобный интерфейс для обучения классических ML моделей с использованием библиотеки scikit-learn.
4. Реализовать систему трекинга экспериментов для отслеживания результатов моделей.

2. Теоретические основы методов машинного обучения и AutoML

2.1. Основные концепции машинного обучения

Машинное обучение — это область исследований, посвящённая разработке алгоритмов, способных на основе анализа данных выявлять закономерности и принимать решения без явного программирования. В основе большинства подходов лежит идея поиска функции f, которая сопоставляет входным данным x соответствующие значения y. При наличии обучающей выборки

$$\{(x_i, y_i)\}_{i=1}^{N},$$

задача сводится к поиску такой модели α, что

$$\alpha(x_i) \approx y_i, \quad i = 1, \ldots, N.$$

В качестве примера можно рассмотреть линейную модель

$$\alpha(x) = w^\top x + b,$$

где w — вектор весов, а b — смещение.

Ключевой этап в обучении моделей связан с минимизацией функции потерь $\mathcal{L}(y, \hat{y})$, которая измеряет расхождения между истинными значениями y и предсказаниями модели $\hat{y}$. Для задачи регрессии часто используется среднеквадратичная ошибка:

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

При этом обучение моделей включает итеративную оптимизацию параметров, где такие алгоритмы, как градиентный спуск, позволяют находить минимумы функций потерь. Итеративное обновление параметров можно записать следующим образом:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\theta),$$

где θ представляет набор параметров модели, η — скорость обучения, а $\nabla_\theta \mathcal{L}$ обозначает градиент функции потерь.

2.2. Почему AutoML? Предпосылки и мотивация

Ранее, для настройки гиперпараметров моделей использовались методы, требующие значительного участия эксперта. Ручной подбор параметров, например, с

применением *Grid Search* или *Random Search*, заключался в последовательном тестировании заранее заданных вариантов. Такой подход нередко был сопряжён с большими вычислительными затратами и высокой зависимостью от интуиции специалиста. Кроме того, человеческий фактор мог приводить к субъективным решениям, что, в свою очередь, влияло на качество итоговых моделей.

С увеличением сложности моделей и расширением пространства поиска гиперпараметров стало понятно, что традиционные методы не всегда способны обеспечить оптимальные результаты в разумные сроки. Эти ограничения стимулировали разработку новых подходов, в числе которых особое место заняла автоматизация процессов подбора параметров. Именно здесь концепция AutoML демонстрирует свою актуальность: автоматизация позволяет не только упростить настройку моделей, но и значительно повысить их эффективность за счёт интеграции алгоритмов, способных анализировать пространство гиперпараметров и выбирать наиболее перспективные направления для поиска.

Таким образом, переход от ручных методов подбора к системам AutoML определяется стремлением к снижению временных и вычислительных затрат, уменьшению влияния субъективного выбора и повышению качества итоговых моделей. В дальнейшем будут рассмотрены конкретные алгоритмы автоматизированного поиска оптимальных гиперпараметров.

2.3. Автоматизированное машинное обучение (AutoML)

Автоматизированное машинное обучение (AutoML) представляет собой совокупность подходов и инструментов, направленных на минимизацию ручного вмешательства эксперта в процессе разработки модели. Основные задачи AutoML включают:

1. Автоматизацию этапов предобработки и отбора признаков.
2. Автоматический подбор архитектуры модели.
3. Оптимизацию гиперпараметров.

В дальнейшем, в этой работе будет рассматриваться 3 пункт - подбор гиперпараметров (или HPO - Hyper Paramater Optimization). Среди классических подходов к поиску оптимальных гиперпараметров выделяются:

- **Grid Search** – исчерпывающий перебор по заранее заданной сетке возможных значений гиперпараметров.

- **Random Search** – случайный выбор комбинаций параметров, что часто оказывается более эффективным при ограниченных вычислительных ресурсах.

- **Bayesian optimization** – метод, который использует вероятностную модель (например, гауссовский процесс) для оценки целевой функции и выбора следующих точек исследования. Формально, поиск оптимальных гиперпараметров можно записать как задачу:

$$x^* = \arg\max_{x \in \mathcal{X}} \alpha(x),$$

  где $\alpha(x)$ — функция приобретения, отражающая полезность проверки данной точки, а $\mathcal{X}$ представляет пространство гиперпараметров.

- **Evolutionary algorithms** – подходы, заимствованные из теории эволюции, где генерация новых кандидатов основана на операциях скрещивания, мутации и отбора, что позволяет эффективно исследовать большое пространство гиперпараметров.

Автоматизация позволяет не только уменьшить временные затраты на поиск оптимальных параметров, но и делает технологии машинного обучения доступными для специалистов с различным уровнем подготовки. В современных информационных системах AutoML становится неотъемлемой частью реализации сложных аналитических пайплайнов.

3. HPO алгоритмы

3.1. Формальная постановка задачи

Постановка задачи оптимизации гиперпараметров (HPO) заключается в поиске вектора $\boldsymbol{\lambda} \in \Lambda$, для которого функция качества модели $Q(\boldsymbol{\lambda})$ достигает оптимального значения. Формально это записывается как

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda} \in \Lambda} \mathcal{L}(\boldsymbol{\lambda}),$$

где $\mathcal{L}(\boldsymbol{\lambda})$ — функция потерь на пространстве гиперпараметров Λ [1].

Важные особенности задачи HPO:

- **Вычислительная сложность.** При полном переборе (Grid Search) или случайном поиске (Random Search) время выполнения растёт экспоненциально от числа гиперпараметров [2].

- **Баланс исследования и эксплуатации.** Современные методы, такие как байесовская оптимизация [3], эффективно сочетают исследование нового пространства гиперпараметров и использование накопленных данных. Так называемые : *exploration* - исследование тех областей, в которых у нас мало семплов на текущей итерации, что даёт нам возможность с меньшей вероятностью пропустить оптимальное значение и *exploitation* - выбирать больше семплов в областях, которые мы достаточно неплохо изучили и где, как мы считаем, с большой вероятностью находится оптимум. [4]

Подходы к решению данной задачи могут значительно различаться как по методологии, так и по используемым вычислительным ресурсам.

3.2. GridSearch и Random Search

Два базовых метода для подбора гиперпараметров, которые часто используются как отправная точка для более сложных алгоритмов, это *Grid Search* и *Random Search*.

Grid Search заключается в следующем: исходное пространство гиперпараметров дискретизируется с помощью заранее определённой сетки. Пусть для гиперпараметра λ_i задан набор значений $\{v_{i,1}, v_{i,2}, \ldots, v_{i,k_i}\}$. Тогда общая конфигурация параметров определяется декартовым произведением всех наборов. Таким образом, осуществляется полный перебор всех возможных комбинаций:

$$\Lambda_{\text{grid}} = \{(v_{1,j_1}, v_{2,j_2}, \ldots, v_{n,j_n}) \mid 1 \leq j_i \leq k_i\}.$$

Этот метод гарантирует нахождение оптимума при условии достаточной плотности сетки, однако его основным недостатком является резко возрастающее число конфигураций при увеличении размерности.

Random Search предлагает альтернативный подход, при котором выборка гиперпараметров осуществляется случайным образом из заданного пространства Λ. При таком подходе конфигурация параметров $\boldsymbol{\lambda}$ выбирается согласно заранее определённому распределению, что позволяет с большей вероятностью «случайно» попасть в область с лучшим значением целевой функции. На практике данный метод часто оказывается более эффективным при ограниченных

вычислительных ресурсах, поскольку позволяет исследовать пространство более равномерно, не тратя время на проверки «неинтересных» комбинаций.

[4] Есть ещё одно довольно интересное объяснение, почему Random Search работает хорошо. Рассмотрим случай, когда у нас конечная сетка гиперпараметров (каждому гиперпараметру сопоставлено конечное число значений).

Для того чтобы с вероятностью не менее 95% хотя бы один из случайно выбранных наборов гиперпараметров оказался среди лучших 5%, необходимо, чтобы количество таких случайных выборов n удовлетворяло следующему неравенству:

$$1 - (1 - 0.05)^n \geq 0.95$$

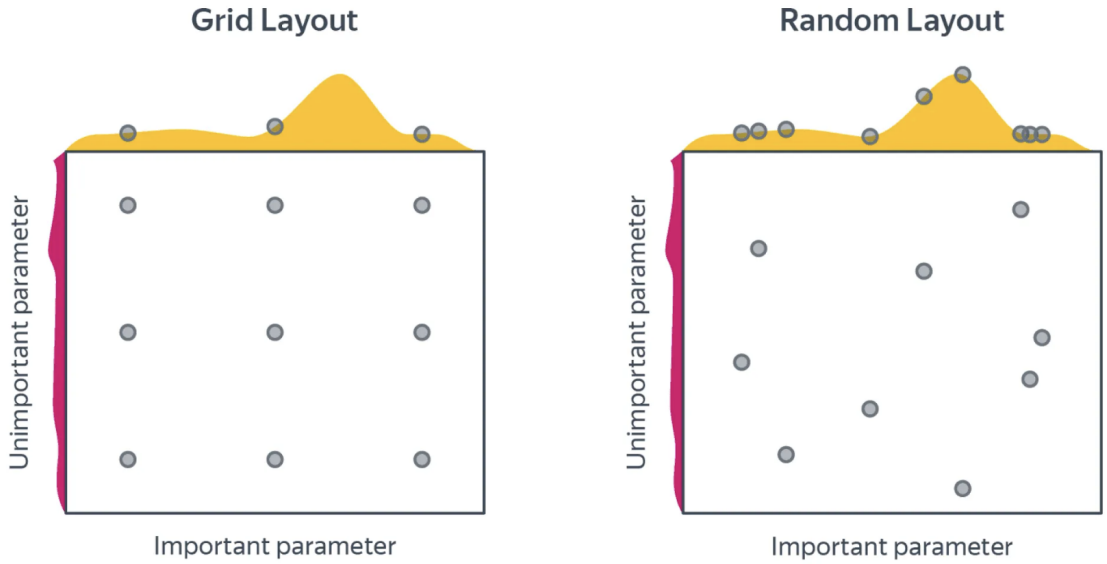

Рис. 1: Источник: Учебник яндекса по ML

Оба метода обладают своими достоинствами и недостатками. Выбор между ними зависит от конкретных условий задачи: размер пространства гиперпараметров, вычислительные возможности и требуемая точность поиска.

3.3. Bayesian Optimization

Bayesian Optimization (байесовская оптимизация) представляет собой последовательную стратегию глобальной оптимизации «чёрного ящика», которая не предполагает фиксированной формы целевой функции и оптимизирует её на основе прошлых наблюдений [5]. Обычно в качестве вероятностной модели целевой функции используется гауссовский процесс, который моделирует обобща-

ющую производительность алгоритма как выборку из GP и позволяет эффективно выбирать следующий набор гиперпараметров на основе приобретательной функции [6].

Основной алгоритм байесовской оптимизации можно свести к следующим шагам:

1. **Инициализация:** Запускается несколько экспериментов с различными значениями гиперпараметров для получения начальных значений $\{(\boldsymbol{\lambda}_i, \mathcal{L}(\boldsymbol{\lambda}_i))\}_{i=1}^{n_0}$.
2. **Построение модели:** На основе наблюдений формируется апостериорное распределение целевой функции $\mathcal{L}$.
3. **Функция приобретения:** Определяется функция приобретения $\alpha(\boldsymbol{\lambda})$, отражающая баланс между исследованием (exploration) и эксплуатацией (exploitation).
4. **Выбор следующей точки:**

$$\boldsymbol{\lambda}_{\text{new}} = \arg\max_{\boldsymbol{\lambda} \in \Lambda} \alpha(\boldsymbol{\lambda}).$$

5. **Обновление модели:** Эксперимент с новой конфигурацией проводится, данные добавляются к наблюдениям и модель пересчитывается.

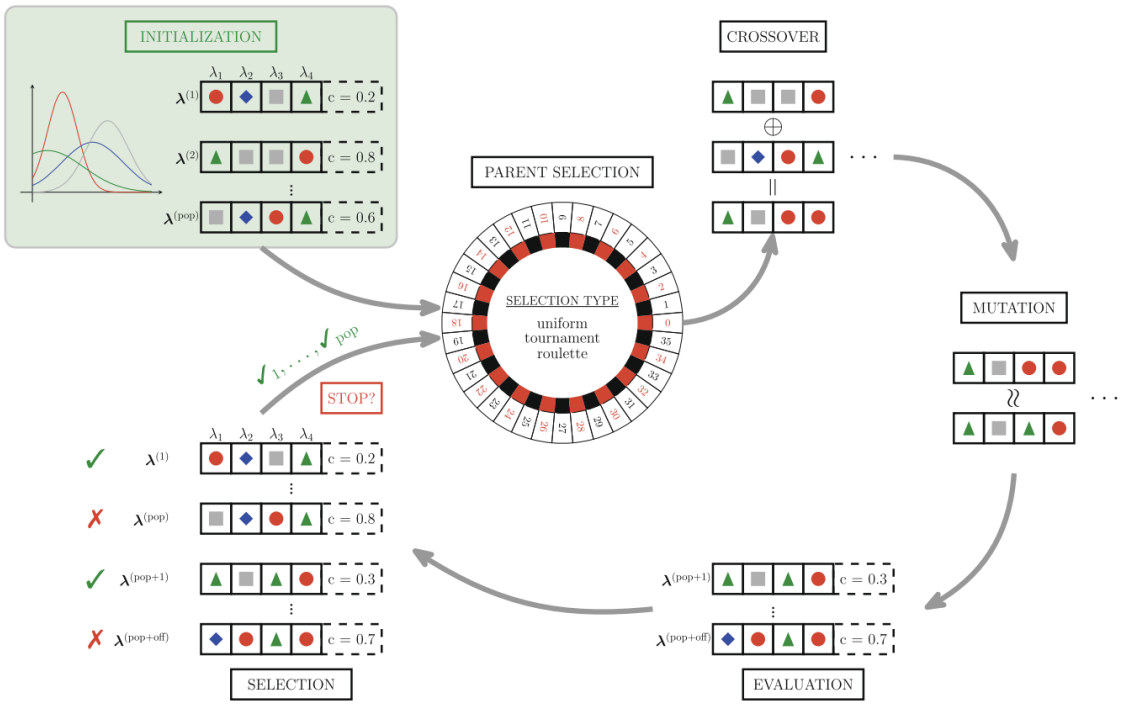

Рис. 2: Источник: Учебник Яндекса по ML

3.4. Evolutionary Algorithm

Эволюционные алгоритмы (ЭА) представляют собой класс методов, вдохновлённых естественным отбором и эволюцией, которые воспроизводят основные механизмы биологической эволюции — воспроизведение, мутацию, рекомбинацию и отбор — для решения сложных задач оптимизации «чёрного ящика» [7]. Каждый индивид в популяции соответствует вектору гиперпараметров, а его «пригодность» оценивается функцией потерь $\mathcal{L}(\boldsymbol{\lambda})$, что позволяет итеративно улучшать решения [8].

Ключевые этапы эволюционного алгоритма для HPO можно сформулировать следующим образом:

1. **Инициализация популяции:** Генерируется начальная популяция
$$P^{(0)} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_N\},$$
где N — размер популяции.
2. **Оценка пригодности:** Каждому индивиду сопоставляется значение функции потерь $\mathcal{L}(\boldsymbol{\lambda}_i)$, характеризующее качество решения.
3. **Селекция:** Выбор родителей (например, турнирный отбор или «колесо рулетки») на основе их пригодности.
4. **Скрещивание и мутация:**
   - **Скрещивание (crossover):** Объединение частей генотипов двух родителей.
   - **Мутация:** Случайные изменения в генотипе для поддержания разнообразия.
5. **Формирование нового поколения:** Интеграция потомков и отбор лучших особей в популяцию следующего поколения.
6. **Критерий остановки:** Повторять цикл до достижения заданного числа итераций или порогового значения $\mathcal{L}$.

Преимуществом ЭА является универсальность (не требует градиентной информации) и хорошая масштабируемость на многомерных пространствах гиперпараметров. Основным недостатком служит высокая вычислительная стоимость при большом числе итераций и оценок функции потерь.

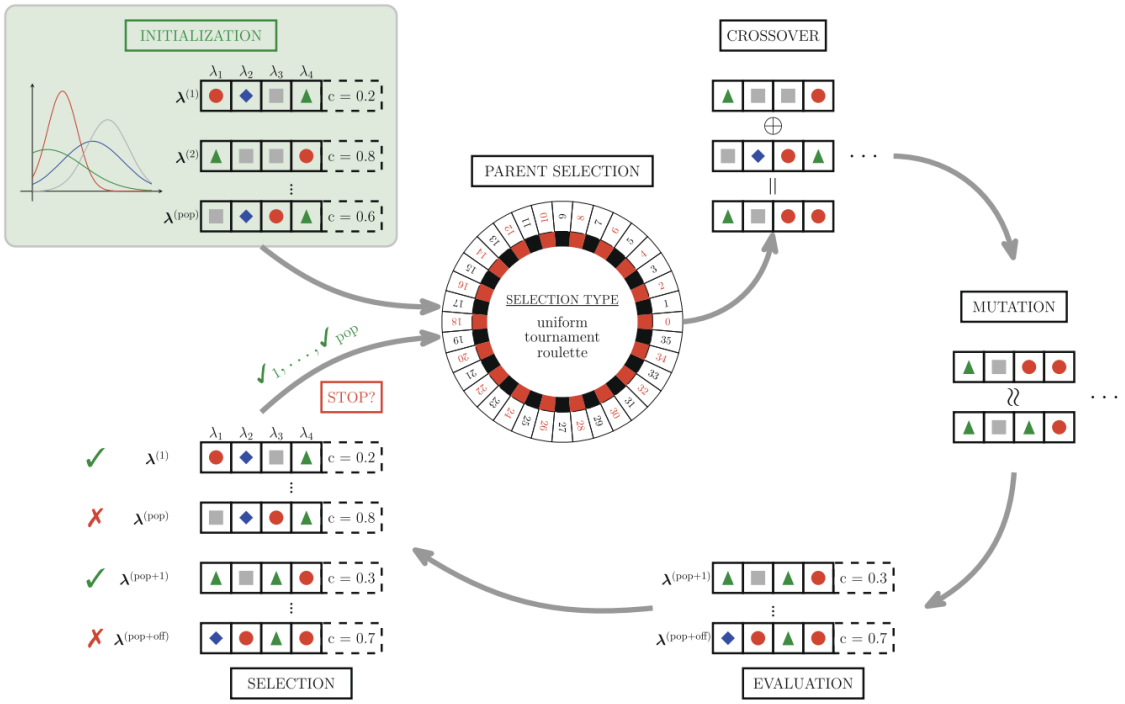

Рис. 3: Источник: Bischl et al. (2023), "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges WIREs Data Mining and Knowledge Discovery.

4. Сравнение и анализ алгоритмов HPO

В этом разделе представлен сравнительный анализ методов оптимизации гиперпараметров для двух классических задач машинного обучения — регрессии и классификации. В качестве моделей выбраны: метод ближайших соседей, случайный лес и градиентный бустинг. Для каждого алгоритма подбора гиперпараметров оцениваются ключевые показатели: точность решения на тестовой выборке и затраченное время работы (в секундах).

4.1. Используемые датасеты

Будут использоваться два публичных датасета, полностью подготовленных для прямого использования в scikit-learn:

- **California Housing** (регрессия): 442 наблюдения, 10 числовых признаков, целевая переменная — прогрессирование диабета через один год после исходного обследованияя.
- **Wine Quality (red)** (многоклассовая классификация): 1 599 наблюдений, 11 числовых признаков (физико-химические свойства вина), целевая пе-

ременная — оценка качества (от 3 до 8 баллов)

4.2. Экспериментальная настройка

- Модели и гиперпараметры (в скобках показана сетка для Grid Search):
  - **RandomForestClassifier & RandomForestRegressor:**
    - n_estimators от 50 до 200 (50, 100, 150, 200)
    - max_depth от 3 до 20 (3, 5, 10, 20)
    - min_samples_split от 2 до 10 (2, 5, 10)
    - min_samples_leaf от 1 до 4 (1, 2, 4)
    - min_impurity_decrease от 0.0 до 1.0 (0.0, 0.1, 0.5, 1.0)
  - **SVC & SVR:**
    - C от 0.1 до 10.0 (0.1, 1.0, 5.0, 10.0)
    - gamma от 0.01 до 1.0 (только для SVC) (0.01, 0.1, 0.5, 1.0)
    - epsilon от 0.01 до 0.2 (только для SVR) (0.01, 0.05, 0.1, 0.2)
  - **GradientBoostingClassifier & GradientBoostingRegressor:**
    - n_estimators от 50 до 200 (50, 100, 150, 200)
    - max_depth от 3 до 10 (3, 6, 10)
    - learning_rate от 0.01 до 0.3 (0.01, 0.1, 0.2, 0.3)
- Каждый алгоритм HPO выполняет 10, 25 и 50 запусков.

4.3. Результаты для задачи регрессии

Модель	**Алгоритм HPO**	**CV MSE**	**MSE**	**Время, с**
RandomForest	GridSearch	3207.9489		48.73
	RandomSearch	3247.1172		1.20
	BayesianOpt	3239.3289		1.68
	EvoStrategy	3234.2308		0.94
SVR	GridSearch	3038.1746		0.08
	RandomSearch	3048.3338		0.06
	BayesianOpt	3038.2845		0.55
	EvoStrategy	3048.3338		0.08
GradientBoosting	GridSearch	3273.8906		4.28
	RandomSearch	3587.1108		0.80
	BayesianOpt	3584.3281		1.89
	EvoStrategy	3410.7006		0.71

Таблица 1: Сравнение алгоритмов HPO на задаче регрессии (10 итераций)

Модель	**Алгоритм HPO**	**CV MSE**	**MSE**	**Время, с**
RandomForest	GridSearch	3207.9489		42.68
	RandomSearch	3231.6922		2.23
	BayesianOpt	3223.7244		4.59
	EvoStrategy	3225.5594		1.85
SVR	GridSearch	3038.1746		0.07
	RandomSearch	3044.2853		0.09
	BayesianOpt	3038.2486		1.41
	EvoStrategy	3038.3249		0.14
GradientBoosting	GridSearch	3273.8906		3.96
	RandomSearch	3381.8119		2.03
	BayesianOpt	3356.5876		4.62
	EvoStrategy	3368.6216		0.81

Таблица 2: Сравнение алгоритмов HPO на задаче регрессии (25 итераций)

Модель	Алгоритм HPO	CV MSE	MSE	Время, с
RandomForest	GridSearch	3207.9489		42.11
	RandomSearch	3231.6922		3.88
	BayesianOpt	3224.2415		9.28
	EvoStrategy	3222.1817		3.08
SVR	GridSearch	3038.1746		0.07
	RandomSearch	3041.0603		0.18
	BayesianOpt	3038.2486		3.53
	EvoStrategy	3038.2754		0.20
GradientBoosting	GridSearch	3273.8906		3.90
	RandomSearch	3365.1996		4.14
	BayesianOpt	3309.9584		8.97
	EvoStrategy	3276.3471		1.25

Таблица 3: Сравнение алгоритмов HPO на задаче регрессии (50 итераций)

4.4. Вывод по задаче регрессии

В задаче регрессии на датасете California Housing наиболее стабильные и при этом быстрые результаты показали эволюционная стратегия и байесовская оптимизация. При 10 итерациях эволюционная стратегия обеспечила CV MSE около 3234 (на порядка 1.4 % хуже GridSearch), но отработала почти в 50 раз быстрее (0.94 с против 48.7 с). При увеличении числа итераций до 50 эволюционная стратегия продолжила снижать CV MSE (3222), оставаясь в 13–15 раз быстрее классического GridSearch. Байесовская оптимизация при больших бюджетах итераций слегка превосходит эволюцию по точности (CV MSE 3309 при 50 итерациях у GBoost и 3224 у RF), но отрабатывает дольше. Классический GridSearch, хотя и демонстрирует наилучшие или близкие к ним результаты по точности, оказывается непрактичным по времени. SVR-модель в целом мало выигрывает от HPO с точки зрения MSE, все методы дают практически идентичные значения (около 3038), поэтому выбор стоит делать по критерию скорости — тут RandomSearch или эволюция выглядят предпочтительнее.

4.5. Результаты для задачи классификации

Модель	Алгоритм HPO	CV Accuracy	Accuracy	Время, с
RandomForest	GridSearch	0.6638		33.68
	RandomSearch	0.5426		0.58
	BayesianOpt	0.5403		1.28
	EvoStrategy	0.5426		0.53
SVC	GridSearch	0.5629		0.61
	RandomSearch	0.5528		0.51
	BayesianOpt	0.5574		1.03
	EvoStrategy	0.5528		0.50
GradientBoosting	GridSearch	0.6552		71.99
	RandomSearch	0.6560		12.95
	BayesianOpt	0.6615		27.55
	EvoStrategy	0.6583		17.76

Таблица 4: Сравнение алгоритмов HPO на задаче классификации (10 итераций)

Модель	Алгоритм HPO	CV Accuracy	Accuracy	Время, с
RandomForest	GridSearch	0.6638		30.78
	RandomSearch	0.5653		1.31
	BayesianOpt	0.5426		2.91
	EvoStrategy	0.5426		0.79
SVC	GridSearch	0.5629		0.63
	RandomSearch	0.5528		1.14
	BayesianOpt	0.5629		2.82
	EvoStrategy	0.5528		0.91
GradientBoosting	GridSearch	0.6552		70.54
	RandomSearch	0.6615		38.66
	BayesianOpt	0.6615		70.32
	EvoStrategy	0.6599		28.19

Таблица 5: Сравнение алгоритмов HPO на задаче классификации (25 итераций)

Модель	Алгоритм HPO	CV Accuracy	Accuracy	Время, с
RandomForest	GridSearch	0.6638		31.93
	RandomSearch	0.5653		2.25
	BayesianOpt	0.5895		6.17
	EvoStrategy	0.6482		1.56
SVC	GridSearch	0.5629		0.61
	RandomSearch	0.5621		1.93
	BayesianOpt	0.5629		5.58
	EvoStrategy	0.5543		1.17
GradientBoosting	GridSearch	0.6552		69.46
	RandomSearch	0.6615		72.83
	BayesianOpt	0.6615		147.08
	EvoStrategy	0.6630		60.23

Таблица 6: Сравнение алгоритмов HPO на задаче классификации (50 итераций)

4.6. Вывод по задаче классификации

В многоклассовой классификации на Wine Quality лучшие показатели по CV Accuracy и финальной Accuracy показывает градиентный бустинг, особенно при использовании байесовской оптимизации и эволюционной стратегии. Уже при 10 итерациях BayesianOpt повысил CV Accuracy до 0.6615 (против 0.6552 у GridSearch) и отработал примерно в 2.5 раза быстрее. При 50 итерациях эволюционная стратегия добилась CV Accuracy 0.6630 и финальной точности 0.664, при этом время работы было в 1.1 раз меньше, чем у GridSearch (60 с против 69 с). У RandomForest и SVC выгода от HPO по сравнению с GridSearch оказалась незначительной: CV Accuracy менялась мало, зато время существенно сокращалось у RandomSearch и эволюции. Таким образом, для классификации оптимальным балансом точности и скорости является применение эволюционной стратегии или байесовской оптимизации на GradientBoostingClassifier.

4.7. Общий вывод сравнения алгоритмов

Анализ показывает, что выбор метода оптимизации гиперпараметров зависит от компромисса между качеством модели и временными затратами. GridSearch остаётся надёжным способом достижения максимального качества, но его высокая вычислительная стоимость ограничивает применимость в ресурсозатрат-

ных сценариях. RandomSearch обеспечивает быструю настройку, однако нередко приносит заметное ухудшение метрик. Байесовская оптимизация часто сочетает хорошее качество и умеренное время работы, но её эффективность с увеличением числа итераций снижается из-за роста вычислительных затрат. Эволюционная стратегия демонстрирует наиболее гибкий баланс: она близка к лучшим метрикам качества (особенно для деревяистых моделей) и при этом требует значительно меньше времени по сравнению с GridSearch и BayesianOptimization.

5. Заключение

В ходе выполнения курсовой работы были разработаны и исследованы алгоритмы автоматическиго подбора гиперпараметров (HPO) для классических методов машинного обучения. Были решены следующие ключевые задачи:

1. Проведён анализ существующих подходов к HPO, включая Grid Search, Random Search, байесовскую оптимизацию и эволюционные алгоритмы.
2. Реализованы и сравнены алгоритмы подбора гиперпараметров на задачах регрессии (California Housing) и классификации (Wine Quality).
3. Систематизированы результаты экспериментов: представлены метрики качества (CV MSE, CV Accuracy), итоговые значения на тестовых данных и время выполнения.
4. Проведен глубокий анализ результатов проведённых измерений.
5. Создана платформа, обеспечивающая удобное обучение как классических ML-моделей с пользовательскими гиперпараметрами, так и автоматизированных HPO-алгоритмов.
6. Разработана система трекинга экспериментов для мониторинга и хранения их метрик.

Проведённые эксперименты показали, что эволюционные стратегии и байесовская оптимизация обеспечивают оптимальный баланс между качеством моделей и временем работы, значительно опережая по скорости классический Grid Search при сопоставимом уровне точности. Реализованный интерактивный конструктор моделей с поддержкой HPO позволяет пользователю без глубоких технических знаний получать качественные ML-решения при минимальных вычислительных затратах.

Таким образом, автоматизация настройки гиперпараметров является неотъемлемой составляющей современного ML-пайплайна, а предложенный подход

демонстрирует практическую эффективность и гибкость для широкого круга задач.

Список литературы

[1] B. Bischl и др. “Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges”. В: *WIREs Data Mining and Knowledge Discovery* 13.2 (2023), e1484. DOI: 10.1002/widm.1484. URL: https://doi.org/10.1002/widm.1484.

[2] James Bergstra и Yoshua Bengio. “Random Search for Hyper-Parameter Optimization”. В: *Journal of Machine Learning Research*. 2012.

[3] Jasper Snoek, Hugo Larochelle и Ryan P. Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. В: *Advances in Neural Information Processing Systems*. 2012.

[4] Yandex School of Data Analysis. *Machine Learning Handbook*. https://education.yandex.ru/handbook/ml. 2025.

[5] *Bayesian optimization*. https://en.wikipedia.org/wiki/Bayesian_optimization.

[6] Jasper Snoek, Hugo Larochelle и Ryan P. Adams. “Practical Bayesian Optimization of Machine Learning Algorithms”. В: *Advances in Neural Information Processing Systems*. 2012. URL: https://arxiv.org/abs/1206.2944.

[7] *Evolutionary algorithm*. https://en.wikipedia.org/wiki/Evolutionary_algorithm.

[8] A. E. Eiben и J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Berlin, Heidelberg: Springer, 2003. ISBN: 978-3-540-40184-3.