Ельцова, Бердовская, Божко  
МетаАлгоритм

1) Найти датасет на задачу классификации или регрессии с большим количеством точек (много наблюдений и много факторов)

2) Разделить данные на трейн, несколько уровней валидации (сильно больших, чем тест) и тест.

3) Обучить большое количество различных базовых алгоритмов различной сложности на трейне, далее найти оптимальные веса для блендинга алгоритмов с помощью метаэвристического алгоритма (выбрать любой, но описать принцип его действия). Проверить обученные веса на тесте.

Мы решили взять датасет, отражающий успеваемость учащихся в системе среднего образования двух португальских школ по португальскому языку. Данные включают оценки учащихся, демографические, социальные и связанные со школой характеристики и были собраны с использованием школьных отчетов и анкет. Набор смоделирован с использованием задач бинарной/пятиуровневой классификации и регрессии. Пропущенные данные и дубликаты отсутствуют.

school: школа учащегося (двоичный вариант: 'GP' - Gabriel Pereira или 'MS' - Mousinho da Silveira)

sex: пол учащегося (двоичный: 'F' - женский или 'M' - мужской)

age: возраст студента (числовой: от 15 до 22)

address: тип домашнего адреса студента (двоичный: 'U' - городской или 'R' - сельский)

famsize: размер семьи (двоичный: 'LE3' - меньше или равно 3 или 'GT3' - больше 3)

Pstatus: статус совместного проживания родителей (двоичный: 'T' - живут вместе или 'A' - раздельно)

Medu: образование матери (числовой показатель: 0 - нет, 1 - начальное образование (4 класса), 2 - с 5 по 9 класс, 3 - среднее образование или 4 - высшее образование)

Fedu: образование отца (числовой показатель: 0 - нет, 1 - начальное образование (4 класса), 2- с 5 по 9 класс, 3 - среднее образование или 4- высшее образование)

Mjob: работа матери (номинальная: 'учитель', 'связанная со здравоохранением', гражданская 'служба' (например, административная или полицейская), 'на\_домашнем\_устройстве' или 'другое')

Fjob: работа отца (номинальная: 'учитель', 'связанная со здравоохранением', гражданская 'служба' (например, административная или полицейская), 'на\_дома' или 'другое')

reason: причина выбора этой школы (номинально: близость к дому, репутация школы, предпочтение курса или «другое»)

guardian: опекун ученика (номинально: 'мать', 'отец' или 'другое')

traveltime: время в пути от дома до школы (числовой показатель: 1 - <15 минут, 2 - от 15 до 30 минут, 3 - от 30 минут до 1 часа или 4 - >1 часа)

studytime: еженедельное учебное время (числовые значения: 1 - <2 часов, 2 - от 2 до 5 часов, 3 - от 5 до 10 часов или 4 - >10 часов)

failures: количество неудач на занятиях в прошлом (числовое значение: n, если 1<=n<3, иначе 4)

schoolsup: дополнительная образовательная поддержка (двоичный вариант: да или нет)

famsup: семейная образовательная поддержка (двоично: да или нет)

paid: дополнительные платные занятия в рамках предмета (математика или португальский язык) (двоичное число: да или нет)

activities: внеклассная деятельность (двоичный: да или нет)

nursery: посещал детский сад (двоичное число: да или нет)

higher: хочет получить высшее образование (двоичное число: да или нет)

internet: Доступ к Интернету дома (двоичное число: да или нет)

romantic: состоит в романтических отношениях (двоичный: да или нет)

famrel: качество семейных отношений (числовой показатель: от 1 - очень плохо до 5 - отлично)

freetime: свободное время после школы (числовой показатель: от 1 - очень мало до 5 - очень много)

goout: прогулки с друзьями (числовой показатель: от 1 - очень низкий до 5 - очень высокий)

Dalc: употребление алкоголя в рабочие дни (числовой показатель: от 1 - очень низкий до 5 - очень высокий)

Walc: употребление алкоголя в выходные дни (числовой показатель: от 1 - очень низкий до 5 - очень высокий)

health: текущее состояние здоровья (числовой показатель: от 1 - очень плохое до 5 - очень хорошее)

absences: количество пропусков занятий в школе (числовое значение: от 0 до 93)

G1: оценка за первый период (числовое значение: от 0 до 20)

G2: оценка за второй период (числовое значение: от 0 до 20)

G3: итоговая оценка (числовое значение: от 0 до 20, целевая переменная)

**EDA**

Большее количество опрошенных учится в школе GP, количество девочек превышает мальчиков на 18%. Семьи опрошенных почти в 90% полные и чаще состоят из 4 человек и больше, а 76,5% отмечают хорошие взаимоотношения в семье. В выборе школы ученики больше опирались на качество курса или расстояние от дома. Так, для 90% человек их школа находится в пределах 30 минут от дома. Также 90% учеников настроены получить высшее образование, но 23% не имеют доступа к интернету. Распределение времени на прогулку с друзьями среди учащихся близко к нормальному с небольшой отрицательной асимметрией. Большинство не имеет серьезных жалоб на здоровье и не употребляют алкоголь в рабочие дни. Среди итоговых оценок G3 есть 0. Некоторые случаи, когда G2 так же равен 0 можно объяснить отчислением, однако случаи, когда ученик получал хорошие оценки за оба полугодия, а итоговую 0, можно списать на выбросы.

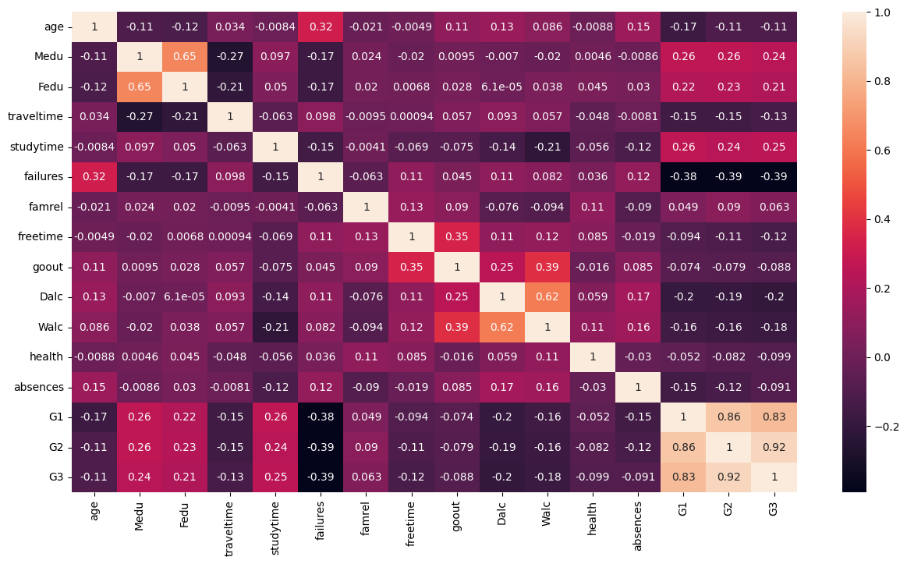
Для построения графиков были написаны функции ncols\_calculator, bar\_matrix и exam\_mean\_facetgrid.

1. ncols\_calculator(cols, nrows=3)   
Эта функция вычисляет количество столбцов, необходимых для сетчатого расположения графиков, на основе количества входных столбцов (cols) и заданного количества строк (nrows).

Логика функции:  
1) n = len(cols): Подсчитывает общее количество столбцов (или характеристик).  
2) ncols = n // nrows: Вычисляет начальное количество столбцов, выполняя целочисленное деление количества столбцов n на nrows.  
3) Если произведение ncols и nrows меньше n, оно увеличивает ncols на 1, чтобы разместить все столбцы.  
4) axdel = ncols \* nrows - n: Определяет, сколько осей (или подплощадок) будет оставлено пустыми (если таковые имеются).

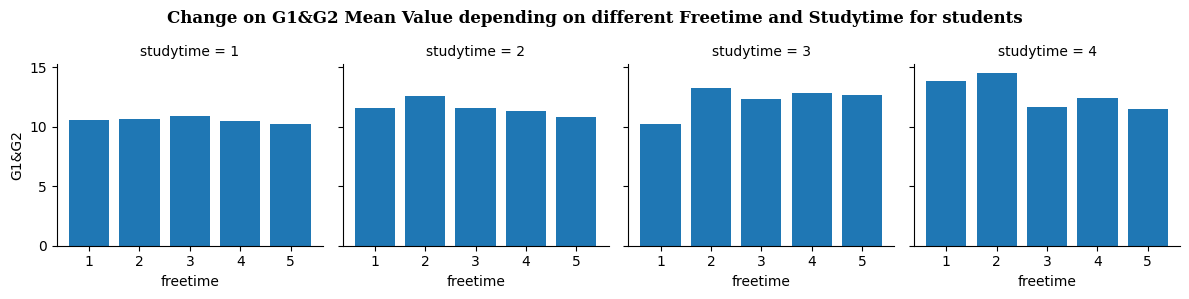
Возвращает: Функция возвращает количество столбцов (ncols) и все лишние оси, которые будут удалены (axdel).

2. bar\_matrix  
Эта функция создает матрицу столбчатых диаграмм для указанных столбцов фрейма данных, визуализируя количество каждой категории в этих столбцах.



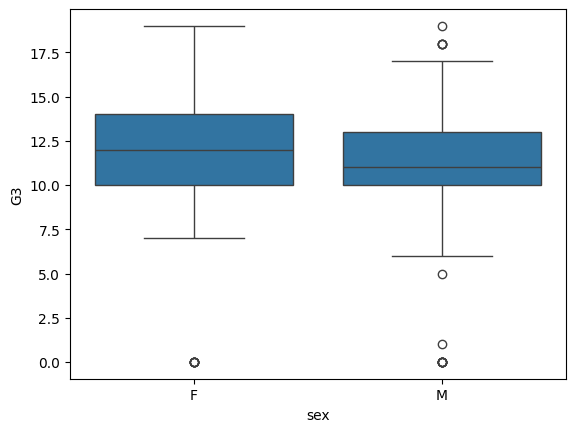
Целевая переменная G3 имеет сильную корреляцию с переменными G2 и G1. Это происходит, потому что G3 - это итоговая оценка за год, в то время как G1 и G2 соответствуют 1-му и 2-му полугодию. Хотя и может быть сложнее предсказать G3 без G2 и G1, но такой прогноз может быть гораздо полезнее. Однако вместо удаления признаков мы будем использовать методы регуляризации (Lasso и Ridge) и менее чувствительные к мультиколлинеарности модели.

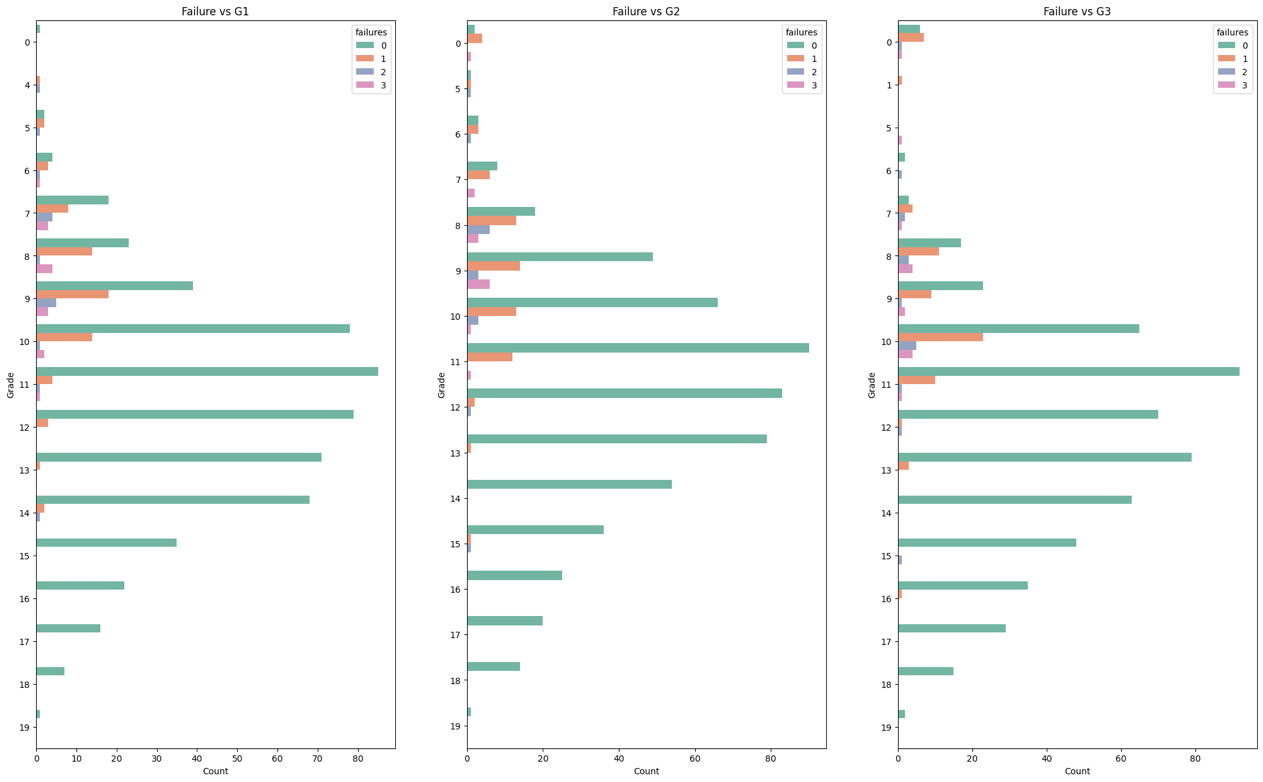
Логично предположить, что учебная занятость и свободное время могут сильно влиять на оценку. Для этого мы создали графики по категориям для разных значений учебного времени, где ось x соответствует значению свободного времени, а ось y равна среднему значению студента.  
Интересно отметить, что если ваше время обучения относится к 4 группе (что означает >10 часов) и у вас меньше свободного времени, вы лучше справляетесь с экзаменом, чем студент, который учится столько же, но имеет больше свободного времени. У учащихся с меньшим количеством свободного времени могут быть другие занятия, которые улучшают их успеваемость, а также стоит учитывать расплывчатость определения “>10ч”, и учащиеся с меньшим свободным временем могут на самом деле учиться больше.

Для построения графика использовалась функция exam\_mean\_facetgrid

Функция группирует DataFrame по указанным x\_col и face\_column и вычисляет среднее значение. Результат сбрасывается в новый DataFrame (plot\_data), что облегчает доступ к нему для построения графиков. Используя sns.FacetGrid: функция создает сетку подплощадок. Это помогает визуализировать средние значения по нескольким категориям, что облегчает сравнение влияния одной категориальной переменной на другую.

Из следующего графика мы видим, что пол практически не влияет на оценки. У девочек они в среднем стабильно немного выше, и больше разброс, что может быть объяснено большим количеством девочек. В оценках мальчиков прослеживается больше выбросов.





Распределение учеников с отсутствием неудач на занятиях в прошлом близко к нормальному, однако данные имеют единичные выбросы. В целом заметна тенденция учеников с отсутствием “провалов” получать высокие оценки с некоторыми исключениями. От выбросов было принято решение не избавляться, так как некоторым может найтись объяснение.

**Создание дополнительных фич и препроцессинг**

Выделяем списки категориальных и числовых переменных:  
categorical\_features = ["school", "sex", "address", "famsize", "Pstatus", "Mjob", "Fjob", "reason", "guardian", "schoolsup", "famsup", "paid", "activities", "nursery", "higher", "internet", "romantic"]   
numeric\_features = ['age', 'Medu', 'Fedu', 'traveltime', 'studytime', 'failures', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences', 'G1', 'G2']

Создаем пайплайн для обработки данных:  
  
1) Масштабирование: числовые переменные имеют разные распределения, однако наиболее важные данные ('studytime', 'freetime', 'G1', 'G2') имеют близкое к нормальному распределение, поэтому было принято использовать стандартное масштабирование (Standard Scaler), так как Robust Scaler, который менее чувствительный к имеющимся выбросам дает большую ошибку. StandartScaler стандартализирует числовые признаки, приводя их к нулевому среднему и единичной дисперсии. Это помогает улучшить сходство числовых признаков и обеспечивает правильную работу алгоритмов, чувствительных к масштабу данных.

2) PolynomialFeatures: используется для генерации полиномиальных признаков из исходных чисел до нескольких сотен. degree=3: указывает уровень полинома, который мы создаем.

3) OneHotEncoder: преобразовывает категориальные признаки, для каждого уникального значения в категориальном признаке создается новое бинарное (0 и 1) значение.

Далее объединяем в ColumnTransformer, который позволяет применить трансформации (пайплайны) к разным столбцам набора данных, упрощая подготовку данных для моделей и создание новых признаков.

**Разбиение данных**

Сделаем несколько уровней валидации. Разделили исходный набор данных на 3 подмножества: трейн, валидацию (3 валидационных набора) и тест (15% от исходных данных).

Разделение на несколько наборов позволяет точнее оценить, насколько стабильна модель при разных разбиениях данных, и может уменьшить риск получения случайно удачной оценки, а также помочь выбрать модель с наилучшими характеристиками (гиперпараметрами).Для функции обучения большого количества алгоритмов для раздедения будем подставлять ct.

Размеры выборок:   
X\_train: (413, 858), y\_train: (413,)   
X\_val1: (138, 858), y\_val1: (138,)   
X\_val2: (275, 858), y\_val2: (275,)  
 X\_val3: (276, 858), y\_val3: (276,)   
X\_test: (98, 858), y\_test: (98,)

**Базовые алгоритмы**

Далее мы обучили 23 базовых алгоритма различной сложности на трейне и проверили на валидационных выборках и тесте. Некоторые параметры были подобраны через GridSearchCV.

1. Linear Regressor (неустойчивая модель из-за выбросов и мультиколлинеарности)
2. Random Forest Regressor
3. GradientBoosting Regressor
4. Lasso
5. Ridge
6. XGBoost
7. DecisionTree Regressor
8. ElasticNet
9. CatBoostRegressor
10. AdaBoostRegressor

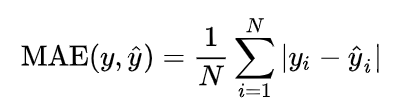
и другие

Методы трансформацции данных:

* Box-Cox: Этот метод требует, чтобы все входные данные были положительными. Наши данные после фичинга могут содержать нулевые или отрицательные значения, Box-Cox не может быть применен.
* Yeo-Johnson: Этот метод может обрабатывать как положительные, так и отрицательные значения. Это делает Yeo-Johnson более универсальным методом.

Для отображения качества моделей нам необходимы метрики, которые были бы устойчивы к выбросам и хорошо интерпретировались. Поэтому мы рассматривали:

1. MAPE:  
     
   Измеряет средний процент ошибки между предсказаниями и реальными значениями и позволяет понять ошибку в относительных терминах. Не подходит для задачи, так как есть фактические значения, равные нулю.
2. MAE:



Показывает, насколько в среднем прогнозы модели отклоняются от реальных значений по модулю. Данная метрика проста в интерпретации и устойчива к выбросам, однако не учитывает масштаб, насколько велико полученное отклонение, и направление ошибок.

1. MSE:

Показывает насколько в среднем прогнозы модели отклоняются от реальных значений в квадрате. Более чувствительна к выбросам, чем предыдущие, однако из-за относительно небольшого числа аномалий мы посчитали ее приемлемой и достаточно показательной для нашей задачи.

Для обучения большого числа моделей мы написали функцию train\_regression\_models, которая делит датасет на трейн, тест и 3 уровня валидации, основываясь на уже датасете с преобразованными фичами - ct. Функция проходит по всем моделям в списке models, считает метрику для каждой и складывает их в словарь results.

**Метаэвристические алгоритмы**

Метаэвристики, инспирированные природой, представляют собой математические преобразования, трансформирующие входной поток информации в выходной и основанные на правилах имитации биоинспирированных механизмов, на процедурах, содержащих элементы стохастической оптимизации со случайными переменными. С помощью метаэвристик, инспирированных природой, исследуется пространство поиска, синтезируются решения, являющиеся точками этого пространства, запрашивается оценка их качества или «приспособленности», которая затем используется для осуществления «естественного отбора». Тем самым, метаэвристики обучаются тому, какие области пространства поиска содержат наиболее приспособленных особей. Метаэвристики применяются к разнообразным задачам планирования, проектирования, логистики, для решения всех типов задач оптимизации.

Основная идея в том, чтобы создать алгоритм, который может обучаться на данных других алгоритмов и оптимизировать их производительность для достижения более эффективных результатов.

Рассмотрим несколько алгоритмов из уровня соответствия природной метафоре. Более чем 400 из существующих метаэвристик разделяются по критерию соответствия природной метафоре на шесть классов: (1) роевые алгоритмы (48 %); (2) алгоритмы, основанные на физических и химических процессах (18 %); (3) алгоритмы, основанные на когнитивных процессах и деятельности человека (12 %); (4) эволюционные алгоритмы (7 %); (5) алгоритмы, основанные на особенностях многоклеточных организмов, способных к фотосинтезу (5 %); (6) прочие алгоритмы.

**Алгоритм роя частиц**

Источник: <https://jenyay.net/Programming/ParticleSwarm?ysclid=m4roeiagsf305907701>

Алгоритм роя частиц представляет собой метод оптимизации нелинейных функций. Алгоритм имитирует поведение птиц во время полета в пространстве. Каждая птица в стае представлена как частица. Эти частицы, образующие рой, ищут пищу на основе своего собственного опыта и опыта других частиц в том же рое.

Принцип работы: в каждый момент времени (на каждой итерации) частицы имеют в пространстве некоторое положение и вектор скорости. Для каждого положения частицы вычисляется соответствующее значение целевой функции, и на этой основе по определённым правилам частица меняет своё положение и скорость в пространстве поиска. При определении следующего положения частицы учитывается информация о наилучшем положении из числа всех остальных соседних частиц.

**Генетические алгоритмы**

Источник: <https://jenyay.net/Programming/Genetic>

Генетические алгоритмы - это алгоритмы поиска, вдохновленные Теорией эволюции в природе Дарвина.

Моделируя процесс естественного отбора, размножения и мутации, генетические алгоритмы могут создавать высококачественные решения для различных задач, включая поиск и оптимизацию.

Согласно теории эволюции Дарвина, эволюция поддерживает популяцию особей, которые отличаются друг от друга (вариативность). Те, кто лучше адаптирован к окружающей среде, имеют больше шансов выжить, размножиться и передать свои черты следующему поколению (выживает наиболее приспособленный).

Генетический алгоритм GA подходит для задач комбинаторной оптимизации, использует начальную популяцию в пространстве поиска решений, предполагает повторение следующих шагов до тех пор, пока не будет выполнен критерий остановки:

1) выбираются родители с использованием турнирного отбора;

2) генерируется потомство путем равномерного кроссинговера;

3) потомство мутирует, случайно инвертируя один бит в решении;

4) потомство оценивается и происходит замена поколений с учетом элитарного отбора.

GA использует дискретную структуру, память, а также три различные стратегии поиска рекомбинацию, мутацию и отбор. На компонентном уровне оператор мутации использует поиск в окрестности, оператор кроссинговера использует промежуточный поиск на основе популяции, популяционный отбор, программируемую память и мультизапуск. Смена поколений в популяции про исходит по схеме восхождения на вершину фитнес-ландшафта.

**Алгоритм имитации отжига**

Источник:<https://translated.turbopages.org/proxy_u/en-ru.ru.70302c27-675f1d6c-8b0de720-74722d776562/https/www.geeksforgeeks.org/implement-simulated-annealing-in-python/>

Имитация отжига (SA) - это вероятностный метод, используемый для нахождения приближенного решения задачи оптимизации. Он особенно полезен для больших пространств поиска, где найти точное решение непрактично. Алгоритм основан на процессе отжига в металлургии

1. Имитация отжига основана на процессе отжига в металлургии. Ключевая идея заключается в использовании параметра "температура", который постепенно уменьшается с течением времени, позволяя алгоритму более свободно исследовать пространство решений при высоких температурах и уточнять поиск по мере снижения температуры.

2. Целевая функция - это функция, которую мы хотим оптимизировать (минимизировать или максимизировать).

### 3. Создание соседней функции. Соседняя функция генерирует новое решение-кандидат, внося небольшое случайное изменение в текущее решение.

4. Реализация алгоритма имитированного отжига. Основными компонентами являются инициализация решения, обновление температуры, генерация новых кандидатов и принятие решения о том, принимать ли их.

### 5. Запуск алгоритма. Определить область задачи, задайте параметры и запустите алгоритм имитированного отжига.

**Сравнение алгоритмов**

Выбор метаэвристического алгоритма для задачи предсказания итоговой оценки (G3) зависит от нескольких факторов, включая размер данных, сложность модели, и желаемую точность.

**1. Алгоритм роя частиц (Particle Swarm Optimization - PSO)**

* **Принцип работы:** PSO имитирует поведение стаи птиц или рыб в поисках пищи. Каждая “частица” (решение) движется по пространству поиска, ориентируясь на лучшее положение, достигнутое ею самой и лучшим положением, найденным всей стаей.
* **Преимущества:**
  + **Простота реализации:** PSO относительно прост в реализации и понимании.
  + **Быстрая сходимость:** обычно сходится к хорошему решению достаточно быстро.
  + **Меньшее количество параметров:** по сравнению с генетическими алгоритмами, PSO имеет меньше настраиваемых параметров.
  + **Эффективность для непрерывных задач:** хорошо подходит для задач, где требуется оптимизировать непрерывные параметры (например, веса нейронной сети или коэффициенты в линейной регрессии).
* **Недостатки:**
  + **Возможна сходимость к локальному минимуму/максимуму:** может застрять в неоптимальных решениях, особенно если пространство поиска сложное.
  + **Менее эффективен для дискретных задач:** обычно требуется дополнительная настройка для применения к задачам с дискретными переменными.

**2. Алгоритм имитации отжига (Simulated Annealing - SA)**

* **Принцип работы:** SA имитирует процесс охлаждения металла. Начинается с высокой “температуры”, при которой разрешаются случайные “прыжки” в пространстве поиска, даже если они ухудшают решение. По мере “охлаждения” системы, вероятность принятия ухудшающих шагов уменьшается, что способствует поиску глобального оптимума.
* **Преимущества:**
  + **Хорош для сложных пространств поиска:** меньше подвержен застреванию в локальных оптимумах, чем градиентные методы.
  + **Простота реализации:** достаточно простой для понимания и реализации.
  + **Не зависит от градиента:** не требует вычисления градиента целевой функции.
* **Недостатки:**
  + **Медленная сходимость:** может требовать значительного времени для сходимости к хорошему решению.
  + **Зависимость от параметров:** Чувствителен к выбору начальной температуры и скорости охлаждения.

**3. Генетический алгоритм (Genetic Algorithm - GA)**

* **Принцип работы:** GA имитирует процесс эволюции, где решения представляются в виде “хромосом”. В процессе “отбора”, “скрещивания” и “мутации” поколения решений эволюционируют, стремясь к более оптимальным значениям.
* **Преимущества:**
  + **Эффективен для широкого круга задач:** хорошо работает как с непрерывными, так и с дискретными переменными.
  + **Устойчивость к локальным оптимумам:** Способен исследовать пространство поиска более широко, уменьшая вероятность застревания в локальных оптимумах.
  + **Параллелизация:** легко распараллеливается для ускорения вычислений.
* **Недостатки:**
  + **Сложная настройка параметров:** требует настройки множества параметров (размер популяции, вероятность мутации, вероятность скрещивания и др.), что может быть сложным и ресурсозатратным.
  + **Медленная сходимость:** может сходиться медленнее, чем PSO, особенно на ранних этапах.

Таким образом, учитывая, что наша задача является дискретной, оптимальнее всего будет использование генетического алгоритма.

**Генетический алгоритм.**

С помощью этого алгоритма ищем оптимальные веса для комбинирования предсказаний нескольких моделей машинного обучения.

**Основные Понятия Генетического Алгоритма:**

* **Популяция:** набор решений (наборов весов). Состоит из набора потенциальных решений задачи.
* **Особь:** отдельное решение в популяции (набор весов)
* **Приспособленность (Fitness):** функция, оценивающая качество решения. В данном случае, отрицательное значение MSE.

Оценка каждого индивида с использованием функции приспособленности. Fi- — это значение функции приспособленности для i -го индивида.

* **Отбор:** процесс выбора лучших особей для создания следующего поколения.
* **Скрещивание (Crossover):** объединение частей двух особей для создания нового решения. Случайным образом выбираются пары родителей, и создаются новые потомки путем комбинирования генов родителей.
* **Мутация:** небольшое изменение случайного элемента решения для добавления разнообразия.

, delta~

* **Создание новой популяции:** Сформировав новую популяцию из потомков (и, возможно, родителей), идет цикл с началом с этапа проверки условия остановки, который может быть:   
  1.Достигнуто максимальное количество поколений.   
  2.Не улучшается приспособленность на протяжении определенного числа поколений. Фитнес лучших решений стабилизировался.  
  3.Найдено оптимальное решение, отвечающее заданным критериям.

Мы решили использовать 2 варианта генетического алгоритма: алгоритм с весами от 0 до 1 и алгоритм с весами от -1 до 1. Объединяем несколько моделей (переменная models) и стремимся найти оптимальные веса для каждой модели (weights), чтобы получить наилучшие прогнозы.

**1. Функция ensemble\_error(weights, models, X, y\_true):** вычисляет ошибку ансамбля моделей, используя заданные веса.

В качестве аргументов в функцию передаются:

* weights: Массив весов, соответствующих каждой модели в models.
* models: Список обученных моделей машинного обучения.
* X: Входные данные для прогнозирования.
* y\_true: Истинные значения целевой переменной.

**Работа функции:**

* Создается массив predictions, заполненный нулями и имеющий ту же форму, что и y\_true.
* Для каждой модели и соответствующего веса: выполняется предсказание на X с помощью model.predict(X) и результат умножается на вес и добавляется к массиву predictions.
* Вычисляется средняя абсолютная ошибка (MAE) между y\_true и predictions. Это наша функция потерь, которую генетический алгоритм будет минимизировать.

**2. Функция genetic\_algorithm(models, X, y, population\_size=50, generations=100, mutation\_rate=0.1):**

В качестве аргументов подаются:

* models: список обученных моделей машинного обучения.
* X: входные данные для обучения.
* y: истинные значения целевой переменной для обучения.
* population\_size: размер популяции (количество особей в каждом поколении), по умолчанию 50.
* generations: количество поколений, по умолчанию 100.
* mutation\_rate: вероятность мутации гена в хромосоме, по умолчанию 0.1.

**Работа функции:**

1) Инициализация популяции:создается начальная популяция случайных весов (хромосом) в диапазоне от -1 до 1. Размер популяции — population\_size, а длина каждой хромосомы (количество генов) равна количеству моделей.

2) Основной цикл по поколениям:

Для каждого поколения (до generations):

- Оценка популяции (Fitness Evaluation):

Вычисляется ошибка ансамбля (фитнес-функция) для каждого индивида в популяции, используя ensemble\_error.

- Инвертирование и нормализация фитнеса:

Фитнес-значения инвертируются (чем меньше ошибка, тем лучше фитнес) и нормализуются, чтобы их можно было использовать как вероятности выбора. Маленькое число (1e-6) добавляется для предотвращения деления на ноль.

3) Выбор родителей (Selection):

Родители выбираются случайным образом с учетом их фитнес-значений (турнирный отбор). Особи с лучшим фитнесом имеют более высокую вероятность быть выбранными.

4) Скрещивание (Crossover):для каждой пары выбранных родителей:

* выбирается случайная точка кроссовера;
* создаются два новых потомка путем обмена генами родителей до и после точки кроссовера.

5) Мутация (Mutation): для каждого гена в каждом потомке с вероятностью mutation\_rate ген заменяется на новое случайное число из диапазона -1 до 1.

6) Обновление популяции:популяция заменяется полученными потомками.

7) Выбор лучшего решения:

После окончания вычисляется ошибка для каждого индивида в финальной популяции, выбирается особь с наименьшей ошибкой и возвращаются лучшие найденные веса.

Отличие второго варианта генетического алгоритма в том, что:

* веса инициализируются случайными числами в диапазоне от 0 до 1, в то время как в первом коде они от -1 до 1
* при мутации ген заменяется на случайное число в диапазоне соответствующего варианта кода
* в первом коде веса на выходе не нормализуются (так как сложно представить себе вариант нормализации такого диапазона и мы ожидаем, что наличие отрицательных весов не даст сильно изменить масштаб данных); в случае модели с только положительными весами после нахождения лучших весов (best\_weights\_1), они нормализуются таким образом, что их сумма равна 1. Это гарантирует, что веса интерпретируются как доли вклада каждой модели в ансамбль, и в сумме дают 100%

Мы предполагаем, что в ансамбле есть модели, которые могут ухудшать качество предсказания, и хотим это компенсировать, установив веса от -1 до 1. Введение отрицательных весов позволяет моделям в ансамбле не только “усиливать” вклад предсказаний, но и “подавлять” его. Это может оказаться полезным, так как некоторые модели в ансамбле работают плохо в определенных областях пространства признаков.

Однако использовать алгоритм только с положительными весами от 0 до 1 полезно, когда мы хотим, чтобы веса представляли собой пропорциональный вклад от каждой модели в ансамбль.

Для обоих наборов весов (best\_weights и best\_weights\_1), полученных из соответствующих генетических алгоритмов, выполняется проверка, чтобы убедиться, что количество весов совпадает с количеством моделей.

Затем формируются предсказания:

1. Создается пустой массив blended\_pred\_4, который будет содержать предсказания ансамбля. Форма массива берется из предсказаний первой модели.
2. Происходит итерация по всем моделям и их соответствующим весам.
3. Для каждой модели, ее предсказания (которые, как предполагается, уже сохранены в словаре predictions по имени модели) умножаются на соответствующий вес из best\_weights и добавляются к blended\_pred\_4.

Для второго набора весов (best\_weights\_1):

Аналогично, создается пустой массив blended\_pred\_5.

Аналогично формируются предсказания ансамбля, используя веса best\_weights\_1.

Затем вычисляем метрики MAE и MSE.

Таким образом, код берет веса, которые были оптимизированы двумя разными генетическими алгоритмами, и использует их для создания предсказаний ансамбля. Затем он оценивает качество этих предсказаний с помощью MAE и MSE.

Вариант 1: веса [-1;1]  
Лучшие веса: [ 0.45385596 -0.99661683 -0.24618046 -0.37814895 -0.28830283 0.80481317 -0.32949931 -0.70254791 0.51714758 -0.7977946 0.25434342 -0.10893385 0.35568342 -0.53994158 0.55845338 -0.163023 0.6324947 0.0590943 0.87513894 0.28496567 0.01962793 0.90462023 -0.16478739]

Mean Absolute Error: 1.9783510410844236

Mean Squared Error: 7.828891395563887

Вариант 2: веса [0;1]

Лучшие веса: [0.03984263330944708, 0.09616175279350885, 0.010918234381961048, 0.06301461117717784, 0.014410880077761874, 0.10041980962709217, 0.15584853850957603, 0.04554483480970413, 0.013096795831419983, 0.03169934498732659, 0.00010457138446404938, 0.020134519706758777, 0.027656239340548616, 0.007442735019946557, 0.03916461419926733, 0.09989807596788763, 0.002305308200439086, 0.009196075204573172, 0.06845848581975043, 0.009130308890963348, 0.044141247084166856, 0.01861611568469884, 0.08279426799155976]

Mean Absolute Error: 0.7794570109904249   
Mean Squared Error: 1.739126568336929