Более подробно о каждой из функций можно узнать с помощью help(*Нужная функция*)

_get_hessian(func, args) - Возвращает матрицу Гессе входной функции по входным переменным.

_take_input(ask_restriction=False) - Создаёт интерактивный ввод для пользователя и возвращает полученные данные в виде словаря.

_make_real(points, args) - Функция проходит по всем входным точкам, и если у точки есть комплексное значение, оставляет только действительную часть.

_check_point(hesse, point) - Функция делает вывод о типе экстремума входной точки с помощью определённости матрицы Гессе.

_filter_points(args, points, bounds) - Отбирает те точки, которые лежат в пределах входных ограничений.

_plot(func, points, bounds=None, restriction=None) - Строит график входной и ограничивающей (если задана) функции отображает точки экстремумов на ней.

find_local_extremas() - Находит точки экстремумов функции, строит её график и отображает на нём найденные точки.

def lagrange() - Находит методом Лагранжа точки экстремумов функции, строит на графике исходную и ограничивающую функцию и отображает на ней найденные точки.

ask_input(ask_bounds=False, ask_initial_point=False) - Запрашивает у пользователя входные данные.

Параметры:

ask_bounds (bool, default=False):
Если True, запращивает у пользователя границы аргумента, точность метода и макс. кол-во итераций

ask_initial_point (bool, default=False):
Если True, запрашивает у пользователя начальную точку, первый и второй параметр для условия Вольфе, а также максимальное ограничение по аргументу

golden_ratio(func, bounds, accuracy=None, show_interim_results=False, max_iter=None, show_convergency=False, return_data=False) - Находит минимум функции методом золотого сечения на заданном интервале.

Параметры:
func (function): Исследуемая функция
bounds (tuple or list): Исследуемый интервал accuracy (float, default=None): Точность метода

max_iter (int, default=None): Максимальное количество итераций
show_interim_results (bool, default=False): Если True, выводит на экран датасет с промежуточными результатами

show_convergency (bool, default=False): Если True, выводит на экран график сходимости алгоритма return_data (bool, default=False): Если True, добавляет в возвращаемый словарь датасет с промежуточными результатами

parabola_method(func, bounds, accuracy=None, max_iter=None, show_interim_results=False, show_convergency=False, return_data=False) - Находит минимум функции парабол на заданном интервале.

Параметры:
func (function): Исследуемая функция
bounds (tuple or list): Исследуемый интервал accuracy (float, default=None): Точность метода max_iter (int, default=None): Максимальное количество итераций
show_interim_results (bool, default=False): Если True, выводит на экран датасет с промежуточными результатами
show_convergency (bool, default=False): Если True, выводит на экран график сходимости алгоритма return_data (bool, default=False): Если True, добавляет в возвращаемый словарь датасет с промежуточными результатами

combined_brent(func, bounds, accuracy=None, max_iter=None, show_interim_results=False, show_convergency=False, return_data=False) - Находит минимум функции комбинированным методом Брента на заданном интервале.

Параметры:
func (function): Исследуемая функция
bounds (tuple or list): Исследуемый интервал accuracy (float, default=None): Точность метода max_iter (int, default=None): Максимальное количество итераций

show_interim_results (bool, default=False): Если True, выводит на экран датасет с промежуточными результатами
show_convergency (bool, default=False): Если True, выводит на экран график сходимости алгоритма return_data (bool, default=False): Если True, добавляет в возвращаемый словарь датасет с промежуточными результатами

bfgs_method(func, x0, c1=None, c2=None, max_iter=None, max_arg=None, accuracy=None, show_interim_results=False, return_data=False) - Находит минимум функции методом BFGS.

Параметры:
func (function): Исследуемая функция
x0 (float): Начальная точка
c1 (float, default=None): Первый параметр условия Вольфе
c2 (float, default=None): Второй параметр условия Вольфе
max_iter (int, default=None): Максимальное количество итераций
max_arg (float, default=None): Ограничение на максимальное значение аргумента
accuracy (float, default=None): Точность метода show_interim_results (bool, default=False): Если True, выводит на экран датасет с промежуточными результатами
return_data (bool, default=False): Если True, добавляет в возвращаяемый словарь датасет с промежуточными результатами

solve(compare=False) - Создаёт интерактивный ввод и находит минимум вводимой функции выбранным методом

Параметры:
compare (bool, default=False):
Если True, выводит таблицу со сравнением решений задач всеми методами

_find_center(func, x1, x2, x3) - Находит центр параболы, построенной по трём точкам.

Параметры:
func (function): Исследуемая функция x1 (float): Первая точка
x2 (float): Вторая точка
x3 (float): Третья точка

_show_convergency(data) - Строит график сходимости алгоритма и выводит на экран размеры интервалов.

Параметры:
data(list): Список значений размеров интервалов

_gradient(expr, point) - Считает значение градиента функции в точке.

_visualize(func, history) - Строит 3d график функции и scatter plot движения градиента

ask_input(ask_alpha=False, ask_alpha0=False, ask_delta=False, ask_gamma=False, ask_history=False, ask_visualizing=False) - Запрашивает у пользователя входные данные

constant_gradient_descent(func, x0, alpha=0.1, max_iter=500, epsilon=1e-5, show_history=False, visualize=False) - Находит точку минимума функции методом градиентного спуска с константным шагом

step_splitting_gd(func, x0, alpha0=0.1, delta=0.1, gamma=0.1, max_iter=500, epsilon=1e-5, show_history=False, visualize=False) - Находит точку минимума функции методом градиентного спуска с дроблением шага

fastest_gd(func, x0, max_iter=500, epsilon=1e-5, show_history=False, visualize=False) - Находит точку минимума функции методом наискорейшего градиентного спуска

conjugate_gradient_method(func, x0, max_iter=500, epsilon=1e-5, show_history=False, visualize=False) - Находит точку минимума функции методом Ньютона-сопряжённого градиента.

compare() - Запрашивает у пользователя входные данные через и выводит на экран таблицу с результатами работы всех алгоритмов

gradient_descent() - Предлагает пользователю выбрать алгоритм решения, запрашивает у него данные и находит точку минимума, выбранным алгоритмом

Lin_reg(x, y, test_size =0.2, graph = False, in_pairs = False)

Функция Lin_reg строит линейную регрессию для заданных данных. Параметры:

x - pandas Датафрейм экзогенных переменных

x - pandas Датафрейм эндогенных переменных

test_size - размер тестовой выборки. По умолчанию - 0.2

graph - Если True, выводит график регрессии. По умолчанию - False

in_pairs - Если True, строит парные регрессии для всех столбцов из x. По умолчанию - False

Функция возвраащает массив коэфициэнтов и свободный член регрессионной модели.

Poly_reg(x=x, y=y,test_size =0.2, degree = 2, graph = False)

Функция Poly_reg строит парные полиномиальные регрессии для каждого x и y из заданных данных.

Параметры:

x - pandas Датафрейм экзогенных переменных

x - pandas Датафрейм эндогенных переменных

test_size - размер тестовой выборки. По умолчанию - 0.2

graph - Если True, выводит график регрессии. По умолчанию - False

degree - Степень полинома. По умолчанию - 2

extract_coeffs(expr, symbols) - Извлекает коэффициенты из sympy выражения.

Параметры:

expr (sympy expression): уравнение или неравенство в виде sympy выражения

symbols (List[sympy.Symbol]): список из переменных, коэффициенты перед которыми извлекаются

Возвращаемое значение:
res (List[float]): список коэффициентов

solve() - Запрашивает у пользователя входную функцию, её аргументы, ограничения, начальную точку и находит точку экстремума при заданных ограничениях

Параметры: None
Возвращаемое значение:
opt.x (np.array): координаты точки экстремума

logistic_regression(X_train, y_train, X_test, y_test, regularization=None, visualize=False) - Классификатор, основанный на логистической регрессии.

Параметры:

X_train (np.ndarray):
массив признаков обучающей выборки

y_train (np.array):
Вектор меток целевого признака обучающей выборки

X_test (np.ndarray):
массив признаков тестовой выборки

y_test (np.array):
Вектор меток целевого признака тестовой выборки

regularization {"l1", "l2", "None"}, default='None':
Параметр регуляризации
visualize (bool), default=False:
Если True, строит график классификации
Возвращаемое значение: answer (dict) - Словарь, в котором хранятся предсказанные метки для тестовой выборки, а также массив весов признаков

svm(X_train, y_train, X_test, y_test, visualize=False) – Классификатор, основанный на векторе опорных векторов.

Параметры:

X_train (np.ndarray):
массив признаков обучающей выборки
y_train (np.array):
Вектор меток целевого признака обучающей выборки
X_test (np.ndarray):
массив признаков тестовой выборки
y_test (np.array):
Вектор меток целевого признака тестовой выборки
visualize (bool), default=False:
Если True, строит график классификации

Возвращаемое значение: answer (dict) - Словарь, в котором хранятся предсказанные метки для тестовой выборки, а также массив весов признаков

compare(X_train, y_train, X_test, y_test, regularization=None) - Сравнивает показатели времени и точности различных классификаторов на одном наборе данных.

Параметры:

X_train (np.ndarray):
массив признаков обучающей выборки
y_train (np.array):
вектор меток целевого признака обучающей выборки
X_test (np.ndarray):
массив признаков тестовой выборки
y_test (np.array):
Вектор меток целевого признака тестовой выборки

Возвращаемое значение: None

Функция-метод для ввода данных для решения задачи максимизации.

# def input_d()

'''Выходные данные: список введенных значений, list '''

Функция-метод для обработки введенных данных.

# def processing(s)

'''Входные данные: s - список введенных значений, list

Выходные данные: кортеж обработанных данных, tuple'''

Функция-метод для создания матрицы симплекс-метода. # def to_tableau(c, A, b)

'''Входные данные:

c - вектор коэффициентов функции максимизации, list A - матрица коэффициентов ограничений, list
b - вектор ограничений, list
Выходные данные: матрица симплекс-метода, list '''

Функция-метод для проверки матрицы симплекс-метода. # def can_be_improved(tableau)
''' Входные данные:

tableau - матрица симплекс метода, list

Выходные данные: содержит ли последняя строка с ограничениями положительные значения, bool '''

Функция-метод для нахождения базового значения в симлекс матрице. # def get_pivot_position(tableau)
''' Входные данные:
tableau - матрица симплекс метода, list

Выходные данные: строки и столбец координаты базового значения, tuple '''

Функция-метод для шага симплекс метода.

# def pivot_step(tableau, pivot_position)

'''Входные данные:
tableau - матрица симплекс метода, list
pivot_position - координаты базового значения в матрице, tuple Выходные данные: новая матрица симплекс метода, list'''

Функция-метод для проверки столбца.

# def is_basic(column)

'''Входные данные:

column - столбец матрицы симплекс метода, list
Выходные данные: является ли столбец базовым или нет, bool '''

Функция-метод для нахождения решения по матрице симплекс метода.

# def get_solution(tableau)

'''Входные данные:

tableau - матрица симплекс метода, list
Выходные данные: решение симплекс метода, list '''

Функция-метод симплекс метода.

# def simplex(c, A, b)

'''Входные данные:

c - вектор коэффициентов функции максимизации, list
A - матрица коэффициентов ограничений, list
b - вектор ограничений, list
Выходные данные: решение симплекс метода и матрица, tuple '''

Функция-метод для нахождения целочисленных решений методом ветвей и границ.

# def branches_and_bound(c, A, b, last_sol)

'''Входные данные:

c - вектор коэффициентов функции максимизации, list
A - матрица коэффициентов ограничений, list
b - вектор ограничений, list
last_sol - решение на предыдущем шаге, list
Выходные данные: список списков решений по всем ветвям, list '''

Функция-метод для распаковки списка и списков решений.

# def unwrap_list(mylist, result)

''' Входные данные:
mylist - список полученных решений, list
result - список распакованных решений, list
Выходные данные: список решений по всем ветвям, list '''

Функция-метод симплекс метода.

# def gomori(c, A, b, f, list_f)

'''Входные данные:

c - вектор коэффициентов функции максимизации, list
A - матрица коэффициентов ограничений, list
b - вектор ограничений, list
f - функция максимизации в аналитическом виде, sympy.expression list_f - список переменных, list

Выходные данные: целочисленное решение задачи методом Гомори. '''

Функция-метод для нахождения целочисленных решений методом ветвей и границ.

# def find_good(c,A,b,f,list_f,last_sol=[0])

'''Входные данные:

c - вектор коэффициентов функции максимизации, list

A - матрица коэффициентов ограничений, list

b - вектор ограничений, list
f - функция максимизации в аналитическом виде, sympy.expression list_f - список переменных, list

```
last_sol - решение на предыдущем шаге, list
Выходные данные: решение методом ветвей и границ. '''

Функция-метод для объединения вызова функций для задачи метода Гомори

# def all_f_gomori()

'''Выходные данные: целочисленное решение задачи методом Гомори. '''

Функция-метод для объединения вызова функций для задачи метода ветвей и границ

# def all_f_branches_and_bound()

''' Выходные данные: целочисленное решение задачи методом ветвей и границ. '''

def getBatch(X, Y, batch_size):
    '''
    Возвращает батчи входных данных.
        Параметры:
            X (np.ndarray):
                массив признаков обучающей выборки
            Y (np.array):
                Вектор меток целевого признака обучающей выборки
            batch_size (int):
                размер батча

        Возвращаемое значение:
            X_batch (np.ndarray):
                батч X
            Y_batch (np.array):
                батч Y
    '''
def predict(x, H_params):
    '''
    Возвращает предсказание для нового экземпляра.
        Параметры:
            x (np.array):
                массив признаков нового экземпляра
            H_params (tuple):
                кортеж с гиперпараметрами модели
```

```
        Возвращаемое значение:
            предсказание для нового экземпляра (float)
    '''

def plotHyperPlane(ax, X, Y, H_params):
    '''
    Строит точки и разделяющие опорные вектора.
        Параметры:
            ax :
                ось
            X (np.ndarray):
                массив признаков обучающей выборки
            Y (np.array):
                Вектор меток целевого признака обучающей выборки
            H_params (tuple):
                кортеж с гиперпараметрами модели

        Возвращаемое значение:
            None
    '''

def SVM_SGD(X, Y, X_new, C=0.1, plot=False):
    '''
    Классификация на 2 класса методом опорных векторов с использованием градиентного спуска.
        Параметры:
            X (np.ndarray):
                массив признаков обучающей выборки
            Y (np.array):
                Вектор меток целевого признака обучающей выборки
            X_new (np.ndarray):
                массив признаков тестовой выборки
            C (float default=0.1):
                параметр регуляризации
            plot (bool default=False):
                Если True, визиализирует классификацию

        Возвращаемое значение:
            словарь с гипепараметрами и прогнозами для тестовых данных
    '''
```