***Алгоритм: Логистическая регрессия***

**Пример1.**

В основе логистической регрессии лежит “сигмоида”. Это монотонная возрастающая нелинейная функция, имеющая форму буквы “S” (формула 1).

|  |  |
| --- | --- |
|  | (1) |

Логистическая функция определена на бесконечности и изменяется в диапазоне от 0 до 1.

В логистической регрессии используется преобразование вида (формула 2):

|  |  |
| --- | --- |
|  | (2) |

Оно называется логит-преобразованием и обладает такими полезными свойствами, как линейность, непрерывность и определенность на бесконечности.

Задача обучения логистической регрессии заключается в подборе коэффициентов (весов) для максимизации правдоподобия.

Критерий правдоподобия описывается формулой 3:

|  |  |
| --- | --- |
|  | (3) |

Для поиска минимума данного выражения применяется метод градиентного спуска.

Для этого на каждой итерации подсчитывается значение градиента (формула 4), который вычитается из значений весов.

|  |  |
| --- | --- |
|  | (4) |

Алгоритм можно реализовать на языке программирования Python

В программном коде реализован класс LogisticRegression, имеющий следующие методы:

* sigmoid(z) – метод, реализующий функцию сигмоиды
* fit(x, y, iterations, lr) – метод, позволяющий выполнить “обучение” модели на заданных значениях
* predict\_probability(X, threshold) – метод, позволяющий определить вероятность по заданному значению
* loss(Y, Y\_pred) – метод определения ошибки

Для демонстрации корректности работы реализованного алгоритма был выполнен пошаговый расчёт.

Пусть были сгенерированы 50 точек со значением y = 0 (таблица 1.1).

*Таблица 1.1 – Сгенерированные точки со значением y = 0*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (6.225953; 0)  (7.090864; 0)  (0.938448; 0)  (4.883051; 0)  (4.585243; 0)  (4.767659; 0)  (4.263978; 0)  (7.028656; 0)  (8.734312; 0)  (0.455714; 0) | (6.795691; 0)  (3.377665; 0)  (6.057999; 0)  (5.284733; 0)  (3.516788; 0)  (4.869974; 0)  (9.433221; 0)  (9.159726; 0)  (1.890128; 0)  (1.940041; 0) | (5.729737; 0)  (3.523604; 0)  (4.273961; 0)  (9.553667; 0)  (5.684523; 0)  (0.413468; 0)  (3.510537; 0)  (0.252018; 0)  (6.800133; 0)  (7.226425; 0) | (6.168154; 0)  (4.755167; 0)  (6.120781; 0)  (9.241946; 0)  (2.755540; 0)  (4.834614; 0)  (7.755265; 0)  (8.361042; 0)  (5.347481; 0)  (2.938204; 0) | (9.646393; 0)  (2.097223; 0)  (7.023243; 0)  (4.152719; 0)  (2.182873; 0)  (5.029063; 0)  (1.875534; 0)  (2.075754; 0)  (6.338961; 0)  (5.299832; 0) |

А также сгенерированы 50 точек со значением y = 1 (таблица 1.2).

*Таблица 4.2 – Сгенерированные точки со значением y = 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (24.68904; 1)  (21.00722; 1)  (19.38777; 1)  (23.63270; 1)  (19.64973; 1)  (19.25936; 1)  (18.69575; 1)  (17.41184; 1)  (16.85810; 1)  (16.39972; 1) | (21.96098; 1)  (24.18971; 1)  (23.48216; 1)  (17.78697; 1)  (23.71791; 1)  (23.08646; 1)  (20.26889; 1)  (19.41455; 1)  (20.84174; 1)  (20.93166; 1) | (23.16128; 1)  (20.60788; 1)  (21.31941; 1)  (24.57976; 1)  (16.38619; 1)  (21.54752; 1)  (22.16996; 1)  (17.77842; 1)  (18.71960; 1)  (23.44474; 1) | (19.74865; 1)  (15.44030; 1)  (20.62114; 1)  (18.00286; 1)  (22.63674; 1)  (16.24574; 1)  (22.37409; 1)  (15.17103; 1)  (15.54813; 1)  (18.59273; 1) | (15.05515; 1)  (18.54310; 1)  (16.51185; 1)  (23.72693; 1)  (23.11799; 1)  (22.87555; 1)  (18.79101; 1)  (16.61081; 1)  (21.42116; 1)  (24.99244; 1) |

Начальные значения весов также генерируются случайно. Пусть это значения [0.53207354, -1.42440414].

Рассмотрим расчёт шага на примере одной случайной точки: (15.17103; 1).

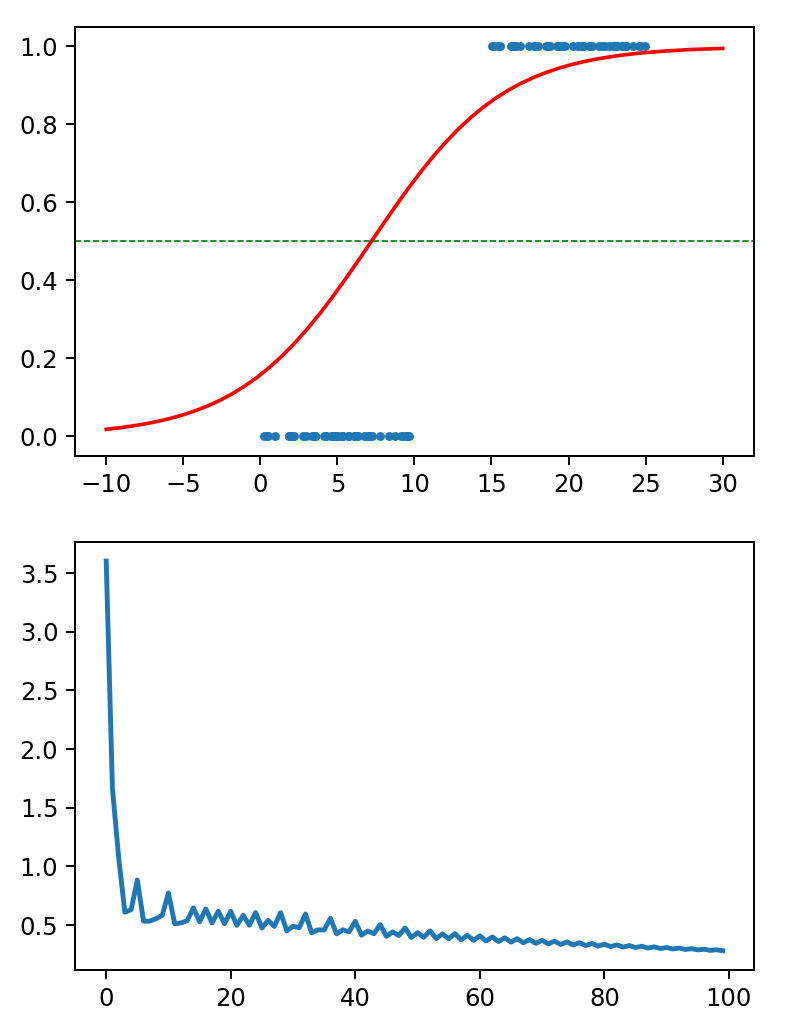
Воспользовавшись формулой

Полученное значение соответствует вероятности принадлежности к классу 1.

Воспользуемся методом градиентного спуска:

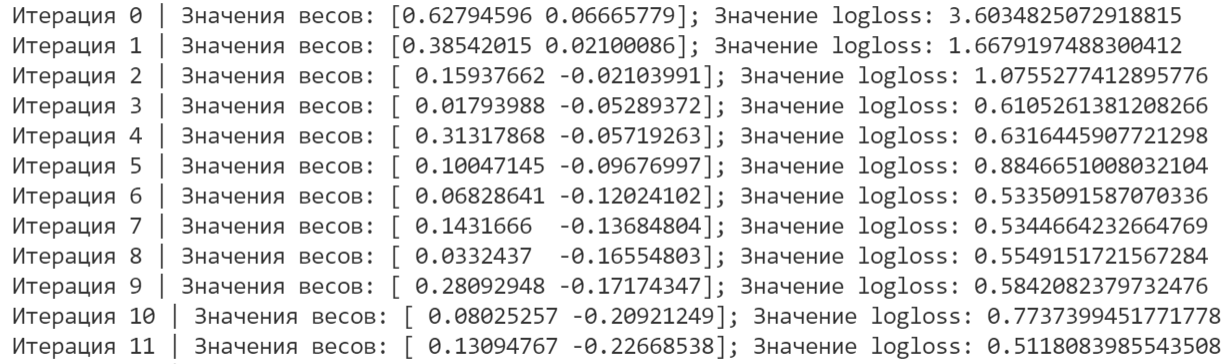
Новые значения весов:

Результат работы программы представлен в виде отмеченных на координатной плоскости точек и нарисованной кривой логистической регрессии.



**Рисунок 1 – Результат работы программы**

Также результатом работы программы является вывод в консоль информации о текущих значениях весов и logloss’а по итерациям (рисунок 2).



**Рисунок 2 – Результат работы программы**

**Пример 2.**

Создадим файл, содержащий информацию о результатах отбора 100 студентов в ВУЗ (Таблица 1).

*Таблица 4 – исходные данные*

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Баллы за 1-ый экзамен** | **Баллы за 2-ой экзамен** | **y** |
| **0** | 34 | 78 | 0 |
| **1** | 30 | 43 | 0 |
| **2** | 35 | 72 | 0 |
| **3** | 60 | 86 | 1 |
| **4** | 79 | 75 | 1 |
| **...** | ... | ... | ... |
| **95** | 83 | 48 | 1 |
| **96** | 42 | 87 | 1 |
| **97** | 99 | 68 | 1 |
| **98** | 55 | 64 | 1 |
| **99** | 74 | 89 | 1 |

Модель логистической регрессии выглядит следующим образом:

(1)

Обучим модель, максимизируем правдоподобие по всей выборке:

(2)

где

* – предсказание модели (0 <= <= 1)
* – входное значение целевой переменной («пройден отбор или нет»)
* – матрица входных данных (результаты экзаменации)
* – веса (параметры) модели

Оптимизируем параметры модели *методом градиентного спуска*:

(3)

***Сделаем 1 итерацию алгоритма вручную***.

Начальные веса модели инициализируем нулями. Тогда при умножении на X и применения сигмоиды получим значение 0,5 из свойств функции в 0. Посчитаем значение логарифмической ошибки для таких результатов модели (таблица 2).

*Таблица 2 – ответы модели и логарифмическая ошибка*

| **№** | **Баллы за 1-ый экзамен** | **Баллы за 2-ой экзамен** | **y'** | **y** |  |
| --- | --- | --- | --- | --- | --- |
| **0** | 34 | 78 | 0.091 | 0 | -0.0954 |
| **1** | 30 | 43 | 0.0 | 0 | -4.225e-05 |
| **2** | 35 | 72 | 0.0439 | 0 | -0.0449 |
| **3** | 60 | 86 | 0.9904 | 1 | -0.0096 |
| **4** | 79 | 75 | 0.9982 | 1 | -0.0018 |
| **...** | ... | ... | ... | ... | ... |
| **95** | 83 | 48 | 0.8586 | 1 | -0.1524 |
| **96** | 42 | 87 | 0.7509 | 1 | -0.2865 |
| **97** | 99 | 68 | 0.9999 | 1 | -0.0001 |
| **98** | 55 | 64 | 0.3393 | 1 | -1.0814 |
| **99** | 74 | 89 | 0.9998 | 1 | -0.0002 |

Значение ошибки L(W1) = 0.6931.

Градиент функции логарифмической ошибки по параметрам равен:

Тогда изменим значение весовых коэффициентов по формуле (3):

Значение ошибки при новом значении коэффициентов регрессии равно:

L(W2) = 0.6318

**После остановки алгоритма.**

После завершения прогона итераций, значение ошибки становится равным 0.2035, а весовые коэффициенты при каждой переменной составляют выражение:

**Ручной расчёт для 2 абитуриентов**

1. Возьмём первого абитуриента из датасета (не прошедшего отбор) со следующими показателями.

*Таблица 3 – данные абитуриента №1 из датасета*

|  |  |
| --- | --- |
| Баллы за 1-ый экзамен | Баллы за 2-ой экзамен |
| 34 | 78 |

Вероятность, что он поступит (y=1) равна:

= σ() = σ(-2.4335) =

Значит, модель тоже уверена в том, что этот человек не пройдет отбор.

2. Возьмём другого случайного абитуриента из датасета, который прошел отбор:

*Таблица 4 – данные абитуриента №96 из датасета*

|  |  |
| --- | --- |
| Баллы за 1-ый экзамен | Баллы за 2-ой экзамен |
| 83 | 48 |

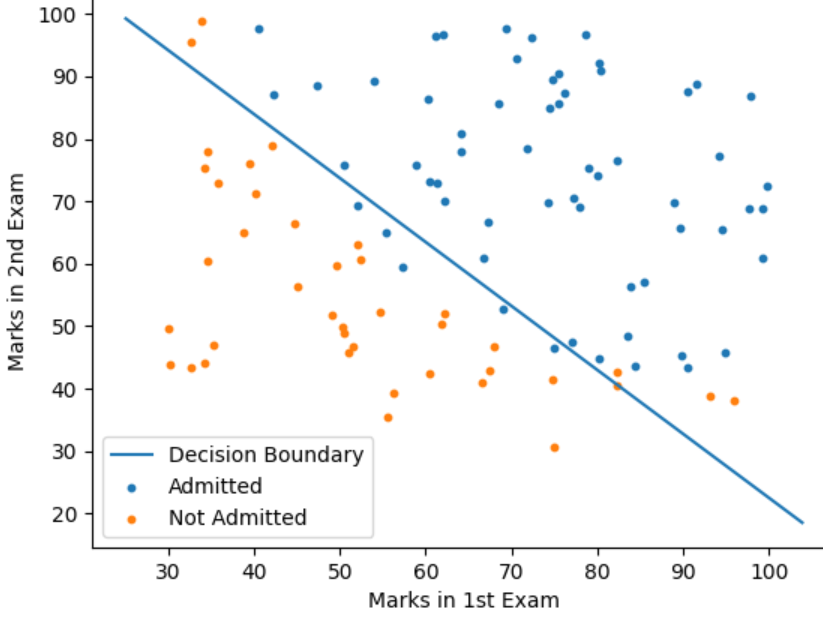
Вероятность, что он поступит (y=1) равна:

= σ() = σ(1.6253) = 0.8356

Значит, модель тоже уверена в том, что этот человек поступит.

Примененим алгоритм логистической регрессии для классификации абитуриентов на «прошедших» и «непрошедших» отбор (синие точки – «прошел отбор», оранжевые - «не прошел отбор»).

Также определим границу решения, которая присвоит будущей точке значение целевой переменной (точка, выше прямой – «прошел отбор», ниже – «не прошел отбор», см. рис. 1)



**Рисунок 1 – Визуализация применения логистической модели к данным и определение границы решения**

Результат работы алгоритма показан в таблице 5.

*Таблица 5 – результаты работы алгоритма*

| **№** | **Баллы за 1-ый экзамен** | **Баллы за 2-ой экзамен** | y' | y |
| --- | --- | --- | --- | --- |
| **0** | 34 | 78 | 0.091 | 0 |
| **1** | 30 | 43 | 0.0 | 0 |
| **2** | 35 | 72 | 0.0439 | 0 |
| **3** | 60 | 86 | 0.9904 | 1 |
| **4** | 79 | 75 | 0.9982 | 1 |
| **...** | ... | ... | ... | ... |
| **95** | 83 | 48 | 0.8586 | 1 |
| **96** | 42 | 87 | 0.7509 | 1 |
| **97** | 99 | 68 | 0.9999 | 1 |
| **98** | 55 | 64 | 0.3393 | 1 |
| **99** | 74 | 89 | 0.9998 | 1 |

где y – верный ответ, а y’ – предсказание алгоритма.