**Домашнее задание 9. Линейная регрессия Логистическая регрессия**

**Задача 1.**

*Даны значения величины заработной платы заемщиков банка (zp) и значения их поведенческого кредитного скоринга (ks):*

*zp = [35, 45, 190, 200, 40, 70, 54, 150, 120, 110],*

*ks = [401, 574, 874, 919, 459, 739, 653, 902, 746, 832].*

*Используя математические операции, посчитать коэффициенты линейной регрессии, приняв за X заработную плату (то есть, zp - признак), а за y - значения скорингового балла (то есть, ks - целевая переменная).*

*Произвести расчет как с использованием intercept, так и без.*

**Решение:**

Для расчета коэффициентов линейной регрессии с использованием intercept (перехвата) можно использовать следующие формулы:

где:

* n - количество наблюдений
* - значение независимой переменной для i-го наблюдения
* - значение зависимой переменной для i-го наблюдения
* - среднее значение независимой переменной
* - среднее значение зависимой переменной
* - коэффициент наклона (slope)
* - коэффициент перехвата (intercept)

Коэффициент наклона () показывает, насколько изменится зависимая переменная (y) при изменении независимой переменной (x) на единицу. Коэффициент перехвата () показывает, какое значение примет зависимая переменная (y), когда независимая переменная (x) равна нулю.

Формулы для расчета коэффициентов линейной регрессии без использования intercept (перехвата) отличаются от формул с использованием intercept только в расчете коэффициента наклона ():

В этой формуле нет слагаемого, отвечающего за перехват, так как он не учитывается при расчете коэффициента наклона.

Остальные обозначения в этой формуле соответствуют обозначениям, приведенным в формулах для расчета коэффициентов линейной регрессии с использованием intercept.

Используя библиотеку NumPy в Python, можно рассчитать коэффициенты линейной регрессии следующим образом:

**import** numpy **as** np

*# заданные значения зарплаты и кредитного скоринга*

zp **=** np**.**array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])

ks **=** np**.**array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])

*# расчет коэффициентов линейной регрессии с intercept*

b\_with\_intercept **=** np**.**cov(zp, ks, ddof**=**0)[0, 1] **/** np**.**var(zp, ddof**=**0)

a\_with\_intercept **=** np**.**mean(ks) **-** b\_with\_intercept **\*** np**.**mean(zp)

*# расчет коэффициентов линейной регрессии без intercept*

b\_without\_intercept **=** np**.**sum(zp **\*** ks) **/** np**.**sum(zp **\*\*** 2)

print(f"Коэффициент наклона с intercept: {b\_with\_intercept: .2f}")

print(f"Коэффициент пересечения с осью y: {a\_with\_intercept: .2f}")

print(f"Коэффициент наклона без intercept: {b\_without\_intercept: .2f}")

Коэффициент наклона с intercept: 2.62

Коэффициент пересечения с осью y: 444.18

Коэффициент наклона без intercept: 5.89

Таким образом, уравнение линейной регрессии с использованием intercept имеет вид:  
y=2.62\*x+444.18  
А уравнение линейной регрессии без intercept имеет вид:  
y=5.89\*x  
Можно заметить, что в случае без intercept, уравнение проходит не через начало координат, что означает, что при зарплате равной нулю, кредитный скоринг тоже будет равен нулю.

Визуализируем результаты линейной регрессии:

**import** matplotlib.pyplot **as** plt

*# график рассеяния*

plt**.**scatter(zp, ks)

*# прямая с intercept*

plt**.**plot(zp, b\_with\_intercept **\*** zp **+** a\_with\_intercept,

color**=**'red', label**=**'с intercept')

*# прямая без intercept*

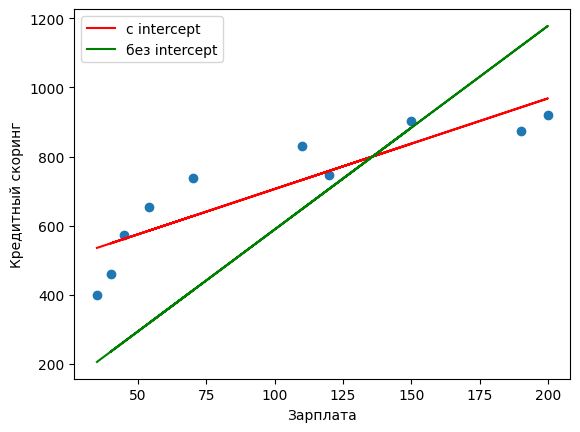
plt**.**plot(zp, b\_without\_intercept **\*** zp, color**=**'green', label**=**'без intercept')

plt**.**xlabel('Зарплата')

plt**.**ylabel('Кредитный скоринг')

plt**.**legend()

plt**.**show()



**Задача 2.**

*Посчитать коэффициент линейной регрессии при заработной плате (zp), используя градиентный спуск (без intercept).*

**Решение:**

Для расчета коэффициента линейной регрессии без intercept с использованием градиентного спуска нам нужно минимизировать функцию ошибки MSE:

где

w - коэффициент линейной регрессии,

и - значения признака и целевой переменной соответственно для i-го наблюдения,

n - количество наблюдений.

Чтобы минимизировать , мы будем использовать градиентный спуск, который заключается в последовательном обновлении весов w по формуле:

где - скорость обучения.

Напишем функцию, которая будет реализовывать градиентный спуск:

**import** numpy **as** np

*# заданные значения зарплаты и кредитного скоринга*

zp **=** np**.**array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])

ks **=** np**.**array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])

**def** gradient\_descent(x, y, alpha**=**0.0001, iterations**=**1000):

w **=** 0 *# начальное значение веса*

n **=** len(x) *# количество наблюдений*

**for** i **in** range(iterations):

y\_pred **=** w **\*** x *# предсказанные значения*

error **=** y **-** y\_pred *# ошибка*

w **+=** alpha **\*** (1 **/** n) **\*** np**.**sum(error **\*** x) *# обновление веса*

**return** w

Теперь мы можем посчитать коэффициент линейной регрессии без intercept с помощью функции gradient\_descent:

b\_without\_intercept **=** gradient\_descent(zp, ks)

print("Коэффициент линейной регрессии без intercept:", b\_without\_intercept)

Коэффициент линейной регрессии без intercept: 5.889820420132689

Визуализируем результаты линейной регрессии без intercept:

**import** matplotlib.pyplot **as** plt

*# график рассеяния*

plt**.**scatter(zp, ks)

*# прямая без intercept*

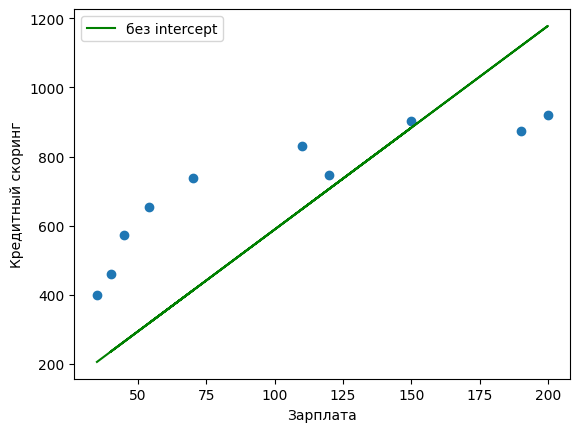
plt**.**plot(zp, b\_without\_intercept **\*** zp, color**=**'green', label**=**'без intercept')

plt**.**xlabel('Зарплата')

plt**.**ylabel('Кредитный скоринг')

plt**.**legend()

plt**.**show()



На графике можно видеть, что прямая без intercept проходит через начало координат.  
Это отличается от модели линейной регрессии с использованием intercept, которая проходит через точку (0, a).

***Задача 3.***

*Произвести вычисления как в пункте 2, но с вычислением intercept. Учесть, что изменение коэффициентов должно производиться на каждом шаге одновременно (то есть изменение одного коэффициента не должно влиять на изменение другого во время одной итерации).*

**Решение:**

Как видно из графика задачи №2, между заработной платой и скоринговым баллом есть некоторая линейная зависимость. Мы можем использовать линейную регрессию для построения линии, которая наилучшим образом приближает эти данные.

Используем метод наименьших квадратов для определения коэффициентов линейной регрессии. Формулы для вычисления коэффициентов линейной регрессии следующие:

где - intercept (точка, где линия пересекает ось y),

- наклон линии,

и - средние значения x и y соответственно.

x\_mean **=** np**.**mean(zp)

y\_mean **=** np**.**mean(ks)

b **=** np**.**sum((zp **-** x\_mean) **\*** (ks **-** y\_mean)) **/** np**.**sum((zp **-** x\_mean) **\*\*** 2)

a **=** y\_mean **-** b **\*** x\_mean

print("Коэффициенты линейной регрессии: a = {:.2f}, b = {:.2f}"**.**format(a, b))

Коэффициенты линейной регрессии: a = 444.18, b = 2.62

Таким образом, уравнение линейной регрессии имеет вид:

y=2.62x+444.18

Теперь мы можем построить линию регрессии на графике:

*# прямая с intercept*

plt**.**plot(zp, b **\*** zp **+** a, color**=**'red', label**=**'c intercept')

*# прямая без intercept*

plt**.**plot(zp, b\_without\_intercept **\*** zp, color**=**'green', label**=**'без intercept')

plt**.**xlabel('Заработная плата')

plt**.**ylabel('Кредитный скоринг')

plt**.**legend()

plt**.**show()

