Лабораторная работа 1
Линейная регрессия

Цель работы: исследование методов применения линейной регрессии для анализа и обработки данных, и её оценки в задачах прогнозирования и выявления закономерностей.

Теоретическая Часть

В современных интеллектуальных системах и задачах машинного обучения нейронные сети широко применяются для решения задач регрессии и классификации. Регрессионный анализ представляет собой статистический метод, позволяющий исследовать взаимосвязь между зависимой переменной **Y** и одной или несколькими независимыми переменными **X**.

Линейная регрессия (Linear Regression) является фундаментальной моделью регрессионного анализа, в которой зависимая переменная представляется в виде линейной комбинации входных признаков. Данная модель используется для прогнозирования, выявления закономерностей и оценки влияния факторов в различных прикладных задачах.

Файлы, включенные в задание:

- **linear_regression.ipynb** – Jupyter Notebook, содержащий реализацию пошагового процесса построения модели простой линейной регрессии.

В рамках данного практического задания вы изучите теоретические основы линейной регрессии и научитесь реализовывать её с использованием библиотеки TensorFlow и языка программирования Python в среде Jupyter Notebook / Google Colab.

Простая линейная регрессия

Простая линейная регрессия представляет собой базовую модель прогнозирования, используемую для оценки количественного отклика **Y** на основе единственной независимой переменной **X**. Этот метод предполагает, что между переменными существует линейная зависимость, описываемая уравнением:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

где:

- β0— свободный член (intercept),
- β1 — коэффициент наклона (slope),
- ε— случайная ошибка (residual), учитывающая отклонения фактических данных от модели.

Модель строится на основе набора из **n** наблюдений (Xi,Yi), где каждое наблюдение содержит значение независимой переменной **X** и соответствующее значение отклика **Y**.

Цель обучения модели заключается в нахождении оптимальных оценок параметров **β0** и **β1** , которые минимизируют расхождение между предсказанными значениями и фактическими данными. Это достигается с использованием метода наименьших квадратов (OLS — Ordinary Least Squares), который минимизирует сумму квадратов отклонений предсказанных значений от реальных наблюдений.

Таким образом, итоговая регрессионная линия должна обеспечивать наилучшее соответствие данным, позволяя делать точные прогнозы на основе входных признаков.

Визуализация данных

Перед построением модели необходимо провести предварительный анализ данных, представив их в графическом виде. Визуализация помогает выявить основные закономерности, тренды и возможные аномалии в данных. Для набора данных, содержащего единственную независимую переменную **X** и зависимую переменную **Y**, наиболее подходящим способом визуализации является **точечная диаграмма (scatter plot)**. Данный метод позволяет отобразить распределение точек и проверить наличие линейной зависимости между переменными.

Построение точечной диаграммы позволяет:

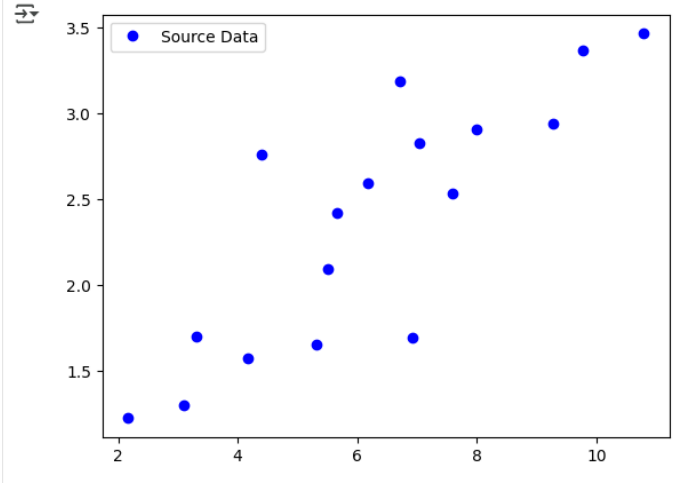
- Оценить характер взаимосвязи между X и Y (линейная, нелинейная или отсутствие связи).
- Обнаружить выбросы, которые могут повлиять на качество модели.
- Предварительно определить, насколько подходящим является использование линейной регрессии для моделирования данных.

Таким образом, визуальный анализ является важным этапом предварительной обработки данных

Задание :

1. Запустите **Jupyter Notebook** или **Google Colab** и откройте файл linear_regression.ipynb.
2. В первой ячейке загружается необходимые библиотеки.
3. Выполните инициализацию обучающих данных — переменных train_X и train_Y.
4. В следующей ячейке добавьте код для построения двумерного графика исходного набора данных, используя **scatter plot**.

```
plt.plot(train_X, train_Y, 'bo', label='Source Data')
plt.legend()
plt.show()
```



Метод градиентного спуска

Градиентный спуск — это один из наиболее эффективных методов оптимизации, применяемый для настройки параметров линейной регрессии θ (также обозначаемых как β). Основная идея метода заключается в движении по направлению наиболее быстрого убывания целевой функции, которое определяется отрицательным градиентом.

Градиент функции представляет собой вектор частных производных, определяющий направление наискорейшего возрастания:

$$f = f(x, y, z),$$

$$\nabla f = \left(\frac{\partial \mathrm{f}}{\partial x_1}, \frac{\partial \mathrm{f}}{\partial x_2}, \ldots, \frac{\partial \mathrm{f}}{\partial x_n}\right)$$

Стратегия оптимизации заключается в построении последовательности обновлений параметров :

$$x^{k+1} = x^k - t_k \nabla f(x^k), t_k > 0, k = 0, 1, 2, \ldots$$

Где t_k — шаг градиентного спуска (learning rate), определяющий скорость обновления параметров.

Алгоритм продолжается до тех пор, пока выполняется условие убывания целевой функции:

$$f(x^{k+1}) - f(x^k) < 0$$

Если данное условие нарушается, шаг t_k корректируется. Завершение итерационного процесса наступает, когда выполняется критерий остановки:

Метод наискорейшего градиентного спуска

Градиентный спуск — это метод поиска локального минимума функции, при котором параметры модели обновляются вдоль направления градиента. Для минимизации функции используется одномерная оптимизация, например, метод золотого сечения, для выбора оптимального шага.

Стратегия метода заключается в построении последовательности точек, где значение функции в каждой последующей точке должно быть меньше, чем в предыдущей. Формула для обновления точек последовательности выглядит следующим образом:

$$x_{k+1} = x_k - t_k \nabla f(x_{k)}$$

где величина шага t_k определяется для каждого значения $\boldsymbol{k}$ как минимизирующая следующую функцию:

$$\varphi(t) = f(x_k - t\nabla f(x_k)) \rightarrow \min \boldsymbol{t_k}$$

Метод минимизирует функцию по одному направлению, вычисляя шаг t_k таким образом, чтобы значение функции на каждой итерации уменьшалось.

Задача линейной регрессии

Задача линейной регрессии – минимизировать функцию стоимости (затрат)

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

где:

- m — количество обучающих примеров,
- $h\theta\left(x^{(i)}\right)$ — предсказанное значение для $x^{(i)}$ (модель линейной регрессии),
- $y^{(i)}$ — фактическое значение.

в котором уравнение гипотезы $h\theta$(x) представляет собой линейную модель. Напомним, что параметры модели, это значения θj. Именно их необходимо подобрать с целью минимизации стоимости J(θ). Один из способов это сделать - использовать алгоритм наискорейшего спуска, где каждая итерация обновляет

$$\theta_j = \theta_j - a\frac{1}{m} \sum_{i=1}^{m} (h_\theta x^i - y^i) x_j^i$$

(одновременно обновляет θj для всех j). С каждым шагом градиентного спуска параметры θj становятся ближе к оптимальным значениям, при которых достигается наименьшая величина стоимости J(θ).

Указание: В формуле выше каждый пример хранится в строке матрицы X. С целью учёта дополнительного коэффициента θ_0, к X добавлен единичный столбец, что позволяет рассматривать θ_0 как дополнительный параметр. С учётом этого, формула, описывающая применение градиентного спуска, после вычисления частных производных функции стоимости $J(\theta_0, \theta_1)$ относительно θ_0 и θ , представленная в покомпонентной форме имеет вид:

$$\theta_0 = \theta_0 - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta x^i - y^i)x_j^i$$

$$\theta_1 = \theta_1 - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta x^i - y^i)x_j^i$$

С каждым шагом градиентного спуска параметры θ_j становятся ближе к оптимальным значениям, при которых достигается наименьшая величина функции стоимости $J(\theta)$.

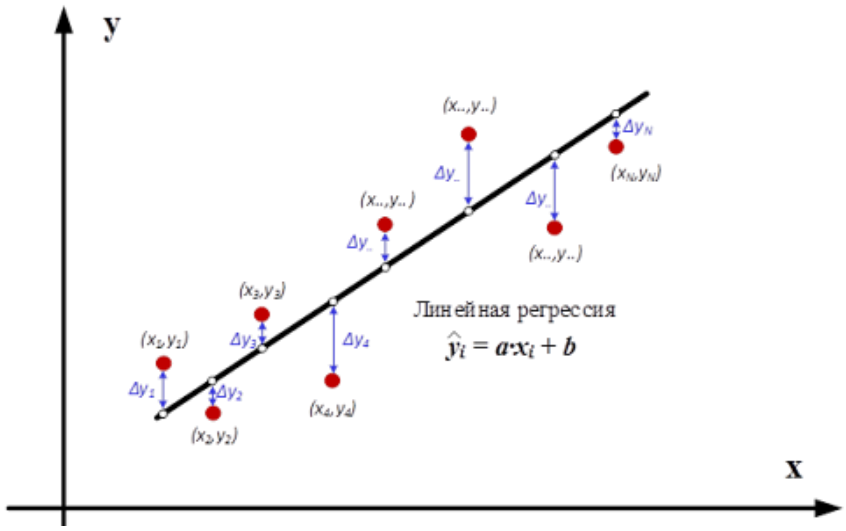


рис.1 Метод наименьших квадратов.

Выполнение

Сначала задаются начальные настройки: выбирается скорость обучения и количество шагов, которые сделает алгоритм (эпохи). Обычно скорость обучения устанавливают равной 0.01, а количество эпох — 1000. Эти значения можно изменить при необходимости.

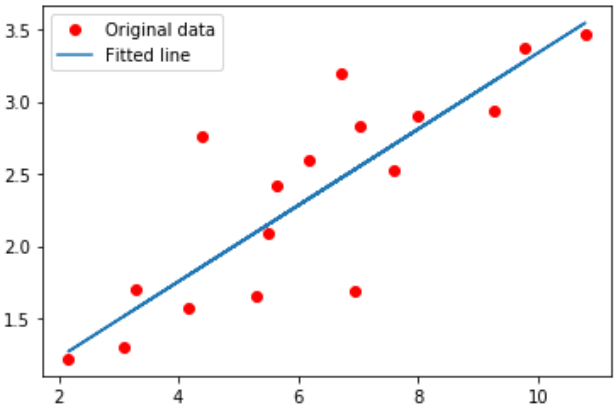
➔ Эпоха — это один полный проход по всем данным из обучающей выборки во время обучения модели.

В блокноте *linear_regression.ipynb* приведена готовая реализация метода линейной регрессии с использованием градиентного спуска. Все операции, использованные в данном методе, были рассмотрены ранее.

Необходимо детально изучить предложенный код, а также подобрать такие значения параметров обучения, при которых функция стоимости cost будет минимальной.

После получения оптимальных параметров модели следует построить итоговую линию тренда, используя приведённый ниже код.

```
plt.plot(train_X, train_Y, 'ro', label='Original data')
plt.plot(train_X, sess.run(W) * train_X + sess.run(b), label='Fitted line')
plt.legend()
plt.show()
```



Задание

Задание к лабораторной работе

1) Ознакомиться с теории и выполните все упражнения из данного документа.

2) Измените начальные параметры скорости обучения и количеств эпох обучения и выведите результаты 3 полученных вариантов предсказаний на одном графике.

Логистическая регрессия: предсказание вероятности события по признакам.

Теоретические сведения

Нейронные сети активно используются для решения задач регрессии и классификации в различных областях. В контексте статистики регрессия — это метод анализа, который исследует влияние одной или нескольких независимых переменных $x_1, x_2, \ldots x_p$ на зависимую переменную Y.

Логистическая регрессия является разновидностью регрессионных моделей, которая, в отличие от линейной регрессии, не предсказывает саму зависимую переменную Y, а оценивает вероятность того, что объект принадлежит определённой категории. Это особенно полезно при решении задач классификации, где цель состоит в определении вероятности попадания объекта в одну из нескольких категорий.

В рамках данной лабораторной работы вы познакомитесь с методом логистической регрессии, который позволяет предсказать вероятность возникновения события на основе множества признаков. Вы научитесь реализовывать этот метод с использованием библиотеки TensorFlow, языка программирования Python и среды Jupyter.

Работа предполагает, что вы будете постепенно дополнять блокнот logistic_regression.ipynb, следуя инструкциям упражнений, и примените полученные знания на практике.

Классификация

Классификация — это задача машинного обучения, при которой модель обучается на данных с заранее известными метками (классами) для того, чтобы предсказать, к какому классу принадлежит новый объект, основываясь на его признаках. Задача классификации заключается в том, чтобы разделить объекты на несколько категорий или классов, исходя из их характеристик. Это может быть как бинарная классификация (например, «болен»/«не болен»), так и многоклассовая (например, «собака», «кошка», «птица»).

В отличие от регрессии, где модель предсказывает числовое значение, в классификации результатом работы модели является метка класса, которая может быть текстовой, числовой или даже категориальной. Классификация широко используется в задачах медицины, финансов, компьютерного зрения и многих других областях.

Задачи классификации встречаются в самых различных областях. Вот несколько примеров:

1. **Медицинская диагностика**: пациент поступает в отделение экстренной медицинской помощи с набором симптомов, которые могут быть связаны с одним из нескольких заболеваний. Какое заболевание у него?
2. **Финансовые мошенничества**: банковский сервис через Интернет должен определить, является ли онлайн-транзакция мошеннической, анализируя такие параметры, как IP-адрес, история предыдущих транзакций и другие данные.
3. **Генетические исследования**: биолог анализирует структуру ДНК пациентов, чтобы определить, какие мутации вызывают заболевание, а какие не связаны с развитием болезни.

Подобно тому, как это было в задаче линейной регрессии, у нас есть набор обучающих данных $(x_{1,}, y_{1,}), (x_2, y_{2,}), \dots, (x_n, y_{n,})$, который мы используем для построения классификатора. Важно, чтобы классификатор хорошо работал не только на обучающих данных, но и на новых, контрольных данных, которые не использовались при обучении модели.

Логистическая модель

Логистическая модель — это статистическая модель, которая используется для прогнозирования вероятности того, что объект принадлежит к одному из двух классов. В отличие от линейной регрессии, которая предсказывает числовые значения, логистическая регрессия моделирует вероятность принадлежности объекта к определённому классу (например, "да" или "нет", "болен" или "здоров", "успешно" или "неуспешно").

Цель логистической модели — найти зависимость между входными признаками (например, возраст, доход, симптомы и т. д.) и вероятностью того, что объект относится к положительному классу. Для этого используется специальная функция, которая преобразует линейную комбинацию признаков в значение вероятности, которое всегда находится в пределах от 0 до 1.

Одной из таких функций является логистическая функция (или сигмоида), которая имеет вид:

$$p(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Где

- $p(X)$— вероятность того, что объект принадлежит к положительному классу (например,$Y = 1$),
- X — вектор признаков объекта,
- β_0 — смещение (параметр модели),
- β_1 — коэффициенты, определяющие влияние признаков на вероятность.

Логистическая модель применяется в задачах классификации, где необходимо определить, к какому из двух классов относится объект. Примером может служить задача предсказания того, будет ли клиент банка согласен на кредит по заданным характеристикам, или классификация электронной почты как спам/не спам.

Подбор коэффициентов для регрессии

Что такое подбор коэффициентов в логистической регрессии?

Логистическая регрессия тоже ищет коэффициенты, но ее цель — не провести прямую через точки, а настроить модель так, чтобы она хорошо разделяла классы.

Модель логистической регрессии задается так:

$$p(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Как мы подбираем коэффициенты?

Мы хотим, чтобы вероятность $p(X)$:

- Была ближе к 1, когда объект реально из класса "1".
- Была ближе к 0, когда объект реально из класса "0"

Для этого мы используем **метод максимального правдоподобия**. Он подбирает такие β_0 , β_1 , чтобы вероятность правильно классифицировать все обучающие объекты была максимальной.

Идея метода максимального правдоподобия:

Мы рассматриваем, что для каждого объекта:

- Если он из класса "1", то хочется, чтобы $p(X)$ было близко к 1.
- Если он из класса "0", то хочется, чтобы $p(X)$ было близко к 0.

Мы записываем произведение вероятностей правильных ответов для всех объектов. Это и есть функция правдоподобия:

$$L(\beta_{0,}\beta_{1,}) = \prod_{i:y_i-1} p(x_i) \quad \prod_{i:y_i-0} 1 - (p(x_i))$$

Эта формула называется **логарифмическая функция правдоподобия**.

Зачем это всё?

Мы подбираем β_0, β_1 так, чтобы логарифм правдоподобия был максимальным. Это значит, что модель лучше всего объясняет наши обучающие данные.

После подбора коэффициентов:

- Подставляем новые объекты в формулу.
- Считаем $p(x)$ — вероятность принадлежности к классу "1".
- Если $p(x) > 0.5$, относим объект к классу "1", иначе — к классу "0

Предсказания.
После того как коэффициенты β_0, β_1, оценены, можно вычислить вероятность принадлежности объекта X к одному из двух классов. Для этого подставляем X в выражение для $p(x)$

Если $p(x)$ близко к 1, то объект скорее относится к классу «+».
Если $p(x)$ близко к 0, то объект, скорее всего, принадлежит к классу «-».
Часто используется порог 0.5: если $p(x) > 0.5$ $p(x) > 0.5$ $p(x) > 0.5$, объект относят к классу «+», иначе — к классу «-».

Множественная логистическая модель

Рассмотрим задачу предсказания бинарного отклика на основе нескольких признаков. Предположим, нам необходимо спрогнозировать, пройдет ли микрочип проверку качества, используя результаты нескольких тестов. В рамках данной задачи мы применим регуляризованную логистическую регрессию.

Ситуация следующая: каждый микрочип подвергается ряду испытаний для выявления дефектов. В нашем распоряжении имеются результаты двух таких тестов для нескольких микрочипов. Цель состоит в том, чтобы на основании этих данных построить модель, позволяющую классифицировать микрочипы на два класса:

- «прошел проверку» (работоспособный),
- «забракован» (дефектный).

Для решения этой задачи мы используем значения двух тестов в качестве признаков, и, опираясь на них, строим модель логистической регрессии, которая будет предсказывать вероятность того, что микрочип является работоспособным.
Графическое представление данных удобно выполнить на плоскости: значения одного теста откладываются по оси X, значения второго теста — по оси Y. Метки классов отображаются разными

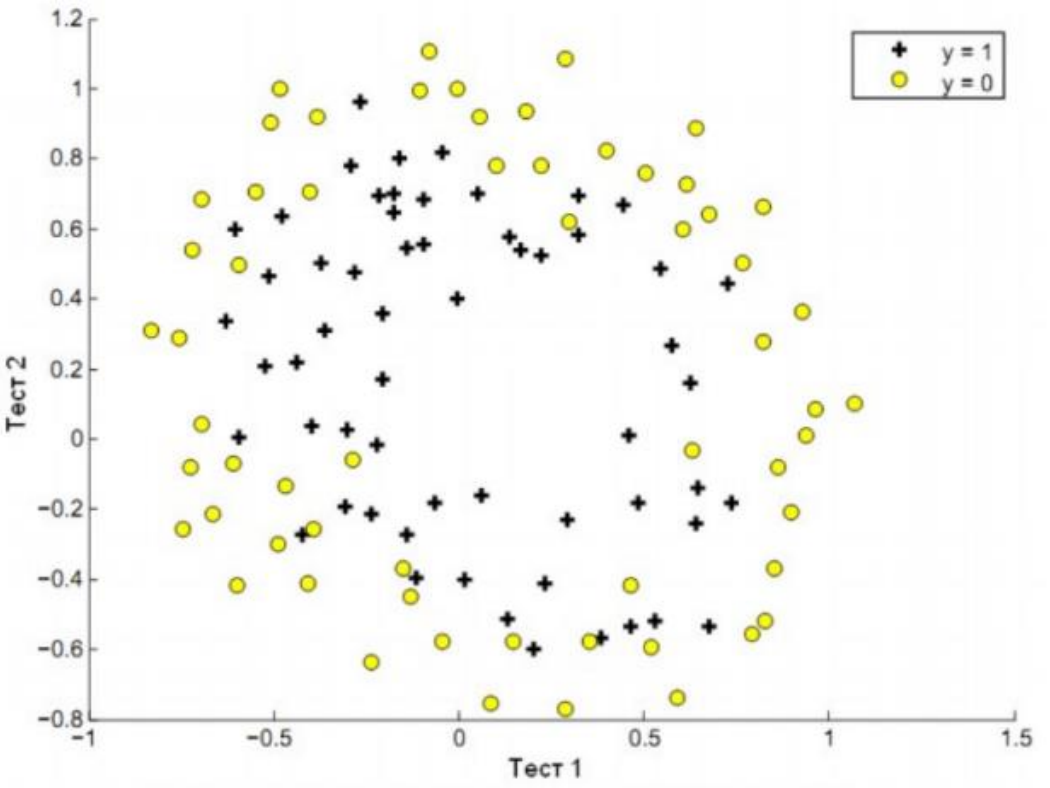


Рис 1 Исследуемые данные

Высшее редставлено, что исследуемые данные не являются линейно сепарабельными, то есть их невозможно разделить на положительный и отрицательный классы с помощью одной прямой линии. Следовательно, использование стандартной логистической регрессии в данном случае оказывается неприменимым, так как эта модель предполагает линейную границу раздела между классами.

Расширение свойств

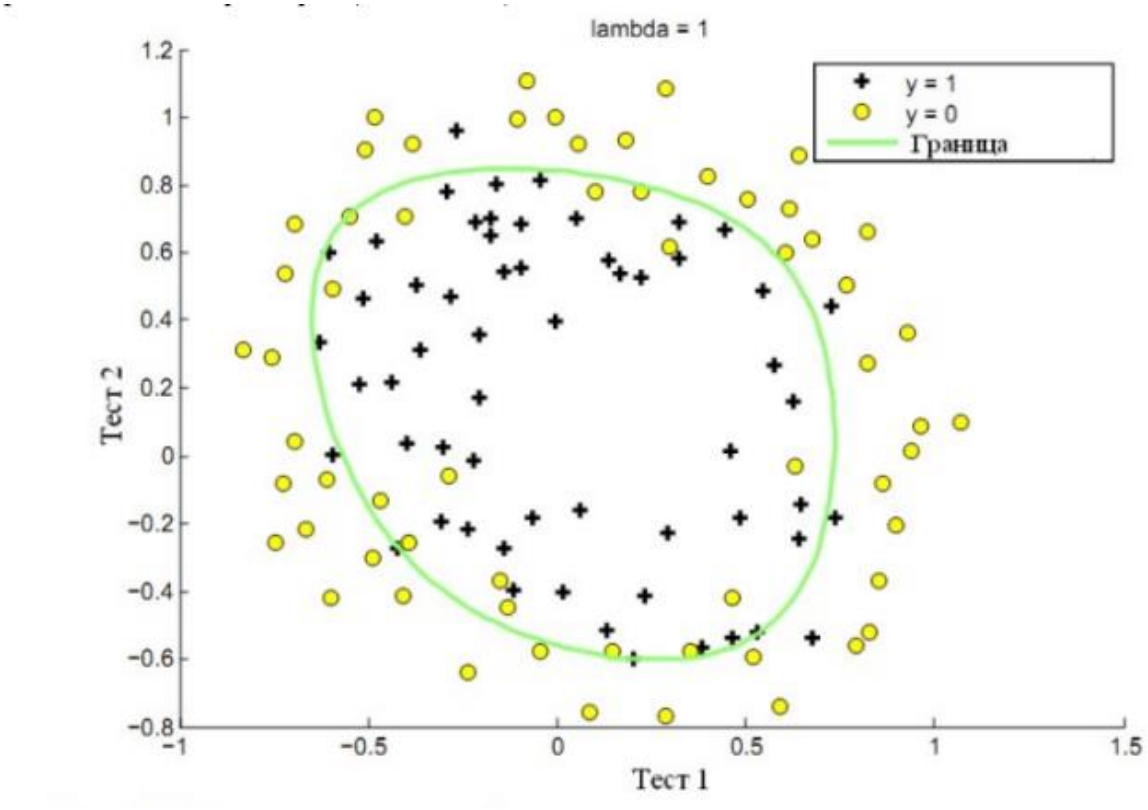
Один из способов лучше оценить данные это создать больше свойств в каждой точке. Расширить исследуемые свойства можно увеличив степень полинома (x1, x2) до 6 степени:

$$\text{mapFeature}(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ \vdots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

В результате повышения степени полиномиальных признаков вектор, изначально состоявший из двух исходных признаков (двух оценок), был преобразован в 28-мерный вектор признаков. На основе этого расширенного пространства признаков классификатор логистической регрессии осуществил подбор коэффициентов, что позволило выделить более сложную, нелинейную границу раздела между классами. Этот эффект можно наглядно наблюдать, построив соответствующий двумерный график.

Добавление полиномиальных признаков увеличивает гибкость модели, что способствует построению более чувствительного классификатора. Однако такой подход приводит к риску переобучения (overfitting), когда модель чрезмерно подстраивается под обучающие данные, формируя излишне сложную границу раздела.

После получения оптимальных оценок параметров β_0,β_1…, строится нелинейная граница разделения областей, отделяющая положительные и отрицательные примеры (см. рисунок ниже).



Упражнение

В предоставленном блокноте *logistic_regression.ipynb* реализован метод логистической регрессии с использованием градиентного спуска. Все операции, применённые в реализации данного метода, были рассмотрены в рамках предыдущих практических занятий.

Целью задания является детальное изучение предложенной реализации метода логистической регрессии, а также настройка гиперпараметров процесса обучения с целью минимизации значения функции стоимости (*cost*) и максимизации точности классификации на тестовой выборке (*Accuracy*).

Задание:

1. Выполнить все практические упражнения, приведённые в данном методическом пособии.
2. Представить отчёт в формате отредактированного файла *logistic_regression.ipynb,* содержащего результаты выполненных заданий и соответствующие выводы.