Машинное обучение: часть 1

План

Сегодня:

Термины машинного обучения

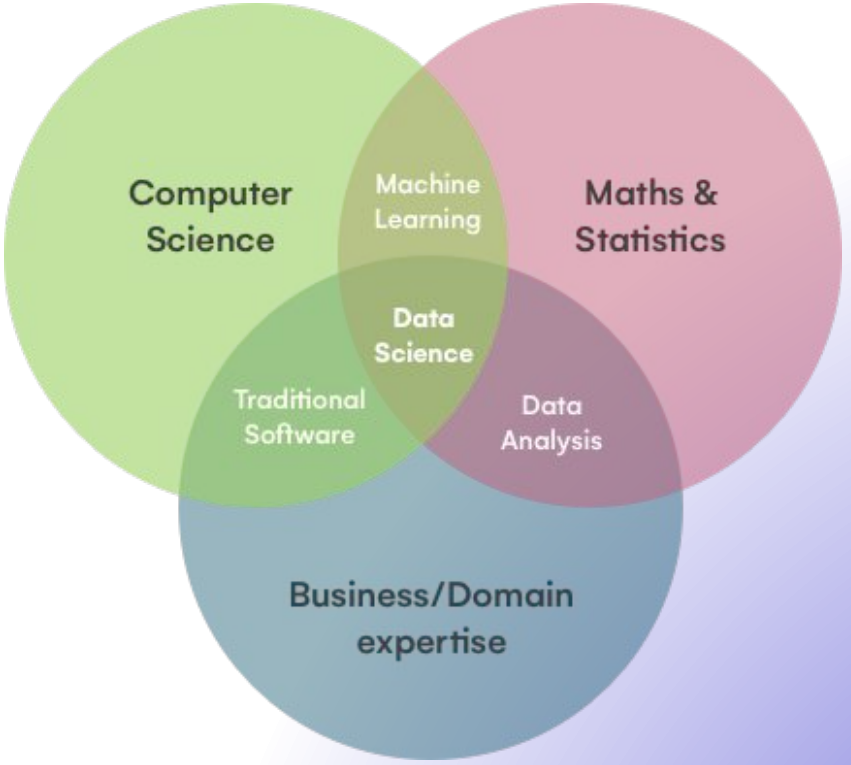
Классификатор

Завтра:

Линейная регрессия

Четверг:

Алгоритм ближайших соседей



Дано

- Пусть **X** - наш массив с данными (пример со вчера - данные пассажиров Титаника)
- элементы X обычно называют ***объектами.***
- Информация, содержащаяся в X обычно называется ***признаками***.
- **Y** - вектор правильных ответов для каждого объекта из X, обычно называется ***целевая переменная***.

Y может быть:

- бинарной (принимать значения 0 и 1)
- мультиклассовой (значения от 0 до K)
- непрерывной (принимать вещественные значения)

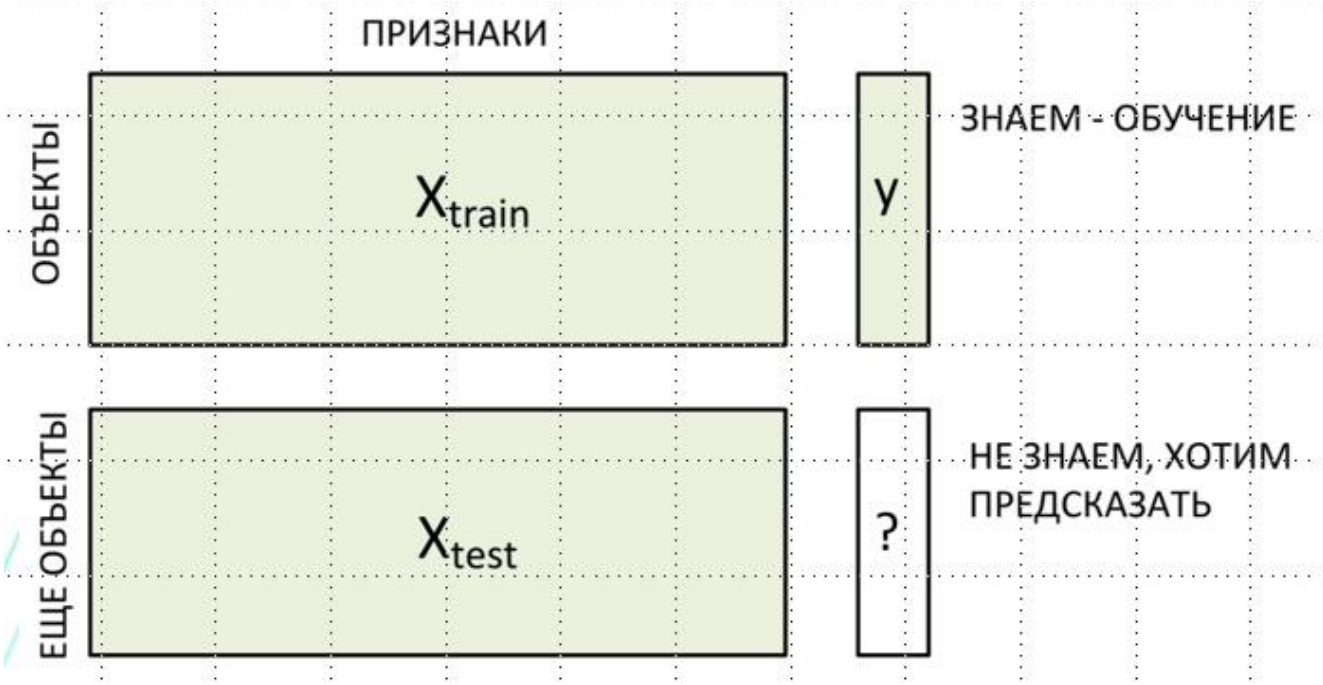
Основные типы задач

Есть два основных типа задач, которые можно решать с помощью машинного обучения:

1. *Обучение с учителем* – у нас есть обучающие данные и правильные ответы для них (знаем и **X**, **Y**)
2. *Обучение без учителя* – у нас есть только обучающие данные, правильных ответов (знаем только **X**)

В этом курсе сосредоточимся на первом типе

Как выглядит решение задачи



Как выглядит решение задачи

Разберем пример. Пусть предсказывается стоимость квартиры по ее признакам.

- $x = (x_1, x_2, ..., x_d)$ - признаки одной квартиры
- $a(x)$ - предсказание модели, то есть стоимость квартиры x по мнению модели ML
- y - правильный ответ (истинная стоимость квартиры).

Цель – подобрать и настроить такую модель *a(X)*, которая будет точнее всего предсказывать правильные ответы, то есть модель с наименьшим отклонением предсказания *a*(*x*) от правильного ответа *y*. Причем отклонение будем смотреть по всем элементам обучающей выборки

После обучения такой модели уже можем использовать ее для новых предсказаний

1. Классификация

Cat

Dog

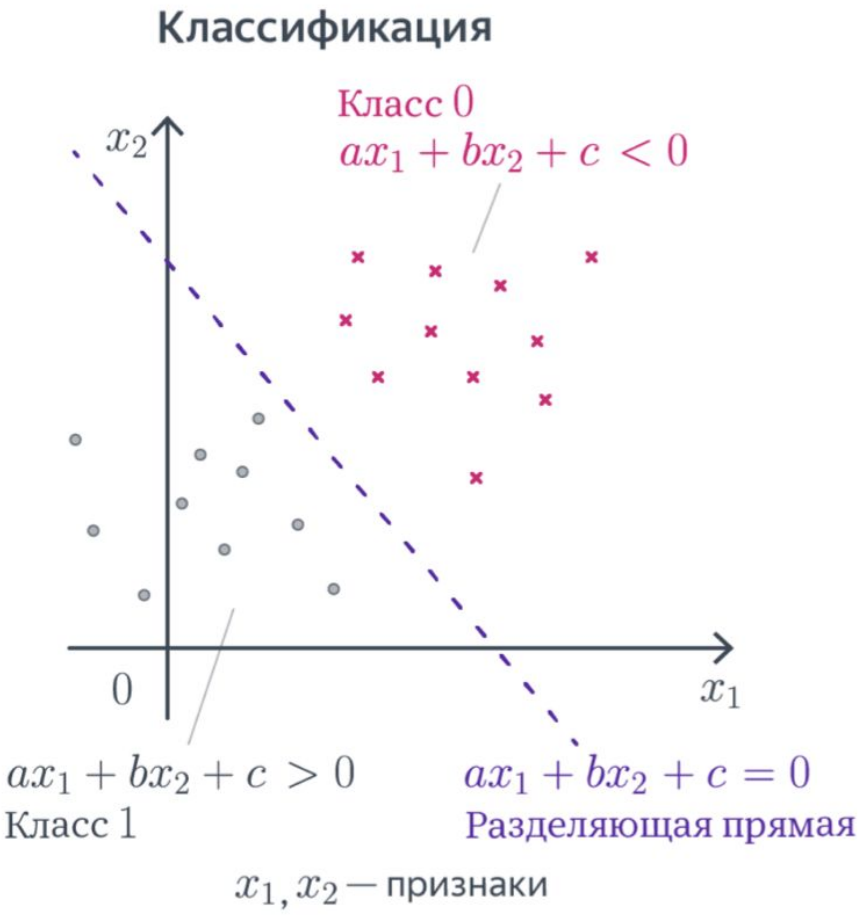
$$\mathbf{X} = \begin{pmatrix} x_{11} & \dots & x_{1d} \\ \dots & \dots & \dots \\ x_{n1} & \dots & x_{nd} \end{pmatrix}, \mathbf{Y} = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}$$

Наше предсказание считается как:

$$\hat{y} = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_d \cdot x_d$$

Хотим найти коэффициенты (***веса***), так чтобы разница между нашим предсказанием и правильным ответом была *в среднем* как можно меньше. Предлагается такой **алгоритм**:

1. Выбрать какие-то значения весов
   1.1. Для всех объектов подсчитать предсказания
   1.2. Посмотреть на разницу между предсказаниями и правильными ответами
   1.3. *Некоторым образом* понять, в какую сторону нам нужно сдвинуть каждый из весов, чтобы уменьшить среднюю ошибку
   1.4. Сдвинуть эти веса
2. Повторять пока не добьемся нужного качества или пока не закончатся итерации цикла



Пусть у нас признаки представлены в виде векторов размерности два. В таком случае мы можем изобразить наши объекты на координатной плоскости.

Тогда задача классификатора - построить такую разделяющую прямую, так, что по одну сторону от нее лежат одни классы, а по другую - другие

Вопрос - всегда ли возможно построить такую прямую?

Какие проблемы могут возникнуть?

1. Дисбаланс классов - одних классов больше чем других
2. Как вообще оценивать качество нашей модели?

Метрики

Можно оценивать то, как хорошо обучается модель

Обычно это называют функцией потерь

Можно оценивать, то как она в итоге хорошо работает

Это и называем метриками, их существует очень много для разных задач

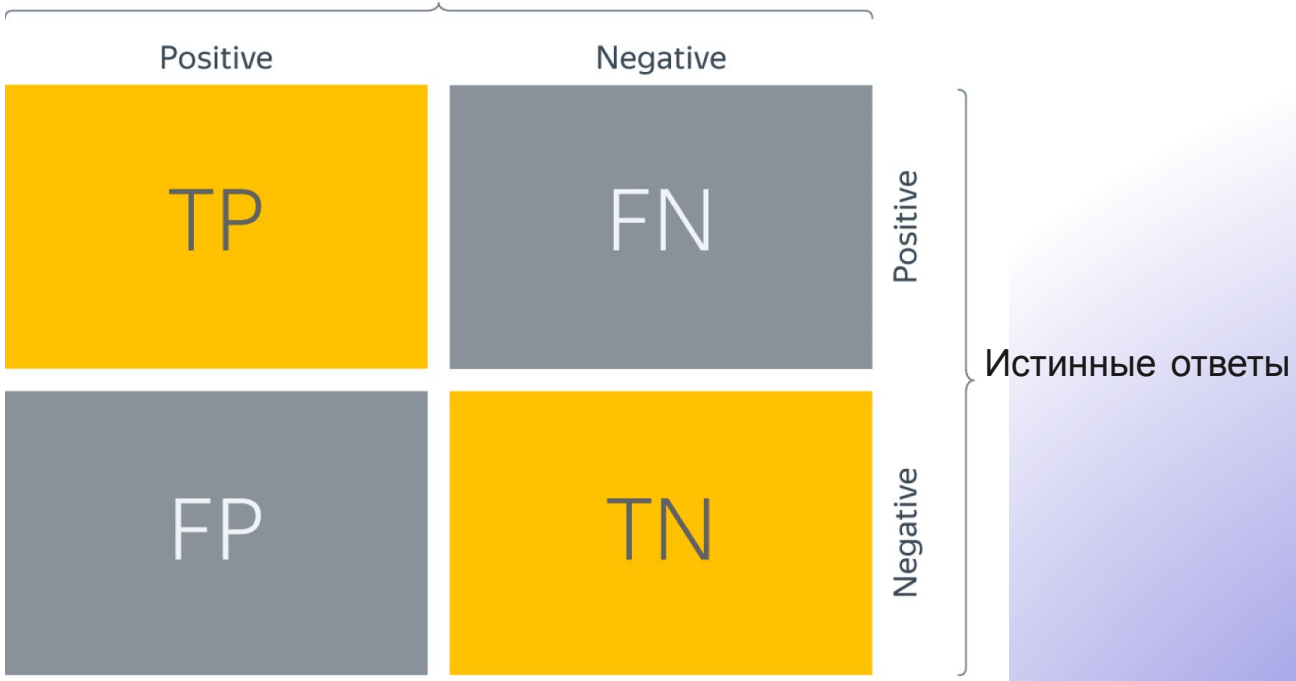
Метрики классификации

Есть идеи, как мы можем оценивать наш классификатор?

Матрица ошибок

Напоминание:
Мы предсказываем бинарный ответ (ДА / НЕТ)

Какие значения мы хотим максимизировать, а какие минимизировать?



Четыре основные метрики

Просто будем оценивать качество нашего классификатора - долю истинных ответов (то есть там где наш классификатор угадал) от доли всех ответов

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

У этой метрики есть одна проблема - она присваивает одинаковый вес и FP и FN

Пример, где важнее FN

Пусть наш датасет - Breast cancer wisconsin (diagnostic) dataset.

Объекты выборки — фотографии биопсии грудных опухолей. С их помощью было сформировано признаковое описание, которое заключается в характеристиках ядер клеток (таких как радиус ядра, его текстура, симметричность). Положительным классом в такой постановке будут злокачественные опухоли, а отрицательным — доброкачественные.

Что такое **FN** и **FP** в данной задаче?

Пример, где важнее FN

- **FP** — доля доброкачественных опухолей, которым ошибочно присваивается метка злокачественной;
- **FN** — доля злокачественных опухолей, которые классификатор пропускает.

В такой постановке становится понятно, что при сравнении выиграет модель с меньшим FN, ведь каждая не обнаруженная опухоль может стоить человеческой жизни.

Пример, где важнее FP

Другая задача - по данным о погоде предсказать, будет ли успешным запуск спутника. **FN** в такой постановке — это ошибочное предсказание неуспеха, то есть не более, чем упущенный шанс.

С **FP** всё серьёзней: если вы предскажите удачный запуск спутника, а на деле он потерпит крушение из-за погодных условий, то ваши потери будут в разы существеннее.

Пример, где **Accuracy** в принципе не подходит

Рассмотрим ситуацию, когда положительный класс это событие редкое. Возьмем в качестве примера поисковую систему - в нашем хранилище хранятся миллиарды документов, а релевантных к конкретному поисковому запросу на несколько порядков меньше.

Пусть мы хотим решить бинарную задачу «документ *d* релевантен запросу *q*». Благодаря большому дисбалансу, Accuracy у классификатора, объявляющего все документы нерелевантными, будет близка к единице.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Высокое значение метрики будет обеспечено членом **TN**, в то время для пользователей более важен высокий **TP**.

Увеличение веса TP

В случае асимметрии классов, можно использовать метрики, которые не учитывают TN и ориентируются на TP.

Точность (precision) – то, как точно мы предсказываем наши классы, в том смысле что если классификатор дал ответ 1 для некоторого объекта, то это действительно так

$$\text{Precision} = \frac{TP}{TP + FP}$$

Увеличение веса TP

В случае асимметрии классов, можно использовать метрики, которые не учитывают TN и ориентируются на TP.

Точность (precision) – то, как точно мы предсказываем наши классы, в том смысле, что если классификатор дал ответ 1 для некоторого объекта, то это действительно так

Чем меньше ложноположительных срабатываний (FP) будет допускать модель, тем больше будет её Precision.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Подсчет полноты

Рассмотрим долю правильно найденных положительных объектов среди всех объектов положительного класса, то мы получим метрику, которая называется **полнотой (recall)** - по сути, как много действительно объектов с меткой 1 мы нашли

$$\text{Recall} = \frac{TP}{TP + FN}$$

Подсчет полноты

Рассмотрим долю правильно найденных положительных объектов среди всех объектов положительного класса, то мы получим метрику, которая называется **полнотой (recall)** - по сути, как много действительно объектов с меткой 1 мы нашли

Интуитивно метрика показывает долю найденных документов из всех релевантных. Чем меньше ложно отрицательных срабатываний, тем выше **recall** модели.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Как учесть и то, и то?

В задаче предсказания злокачественности опухоли **точность** показывает, сколько из определённых нами как злокачественные опухолей действительно злокачественные, а **полнота** — какую долю злокачественных опухолей нам удалось выявить.

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} =$$

$$= 2\frac{Recall \cdot Precision}{Recall + Precision} = \frac{TP}{TP + \frac{FP+FN}{2}}$$

Выводы

- **Accuracy** – процент верно угаданных ответов полезно, когда классы для нас имеют одинаковое значение
- В ситуациях, где важно не пропустить положительный класс (например, в диагностике болезней), предпочтение отдаётся **Recall**.
- В задачах, где критично сократить количество ложных срабатываний (например, в фильтрации спама), важнее **Precision**.
- **F1-мера** – возможность получить одно число и учесть и полноту и точность

Что еще важно знать?

Наша цель - построить модель, которая будет "соответствовать" нашим представлениям о задаче.

Нам не нужно, чтобы модель научилась идеально подгоняться под обучающие данные, мы хотим, чтобы она адекватно работала и на тех данных, которые она еще не видела.

Однако, у моделей как раз таки есть свойство подгоняться под данные, если начать бесконечно крутить цикл алгоритма

Переобучения

Такая ситуация называется переобучением – для нее характерно почти идеальное качество на обучающей выборке и довольно плохое качество на тестовой выборке.

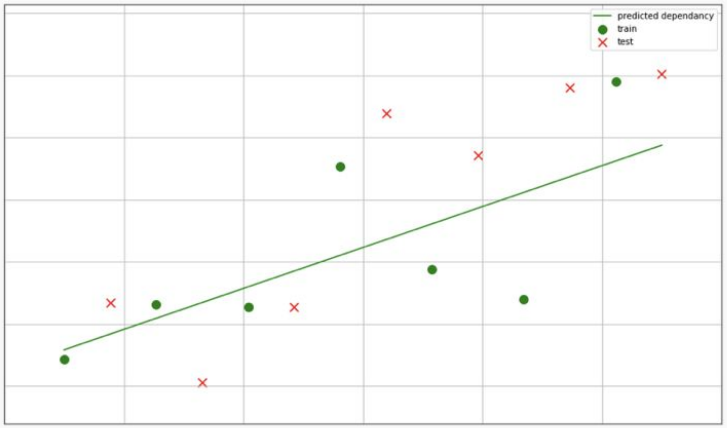


Рис. Приближение многочленом степени 1.

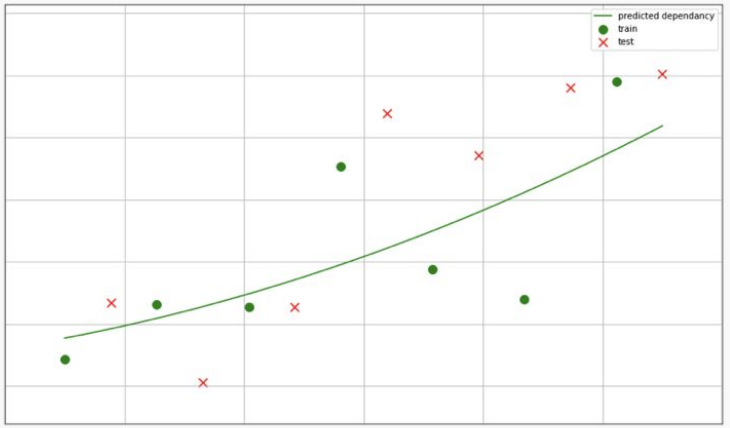


Рис. Приближение многочленом степини 2.

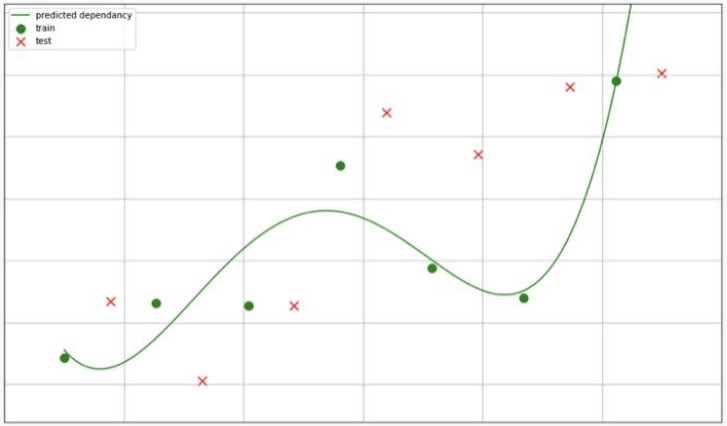


Рис. Приближение многочленом степини 3.

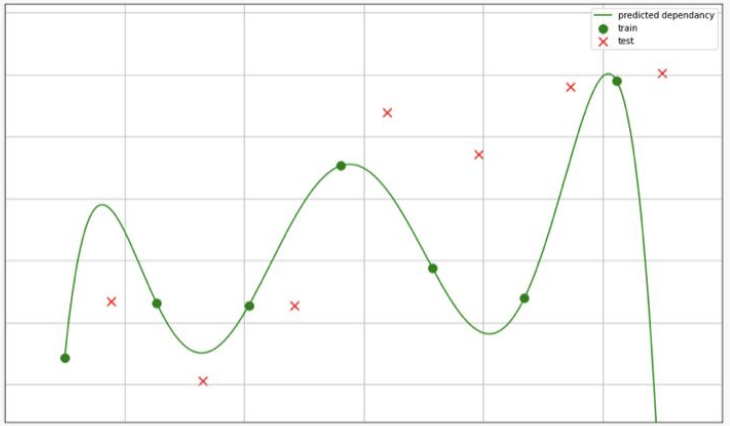
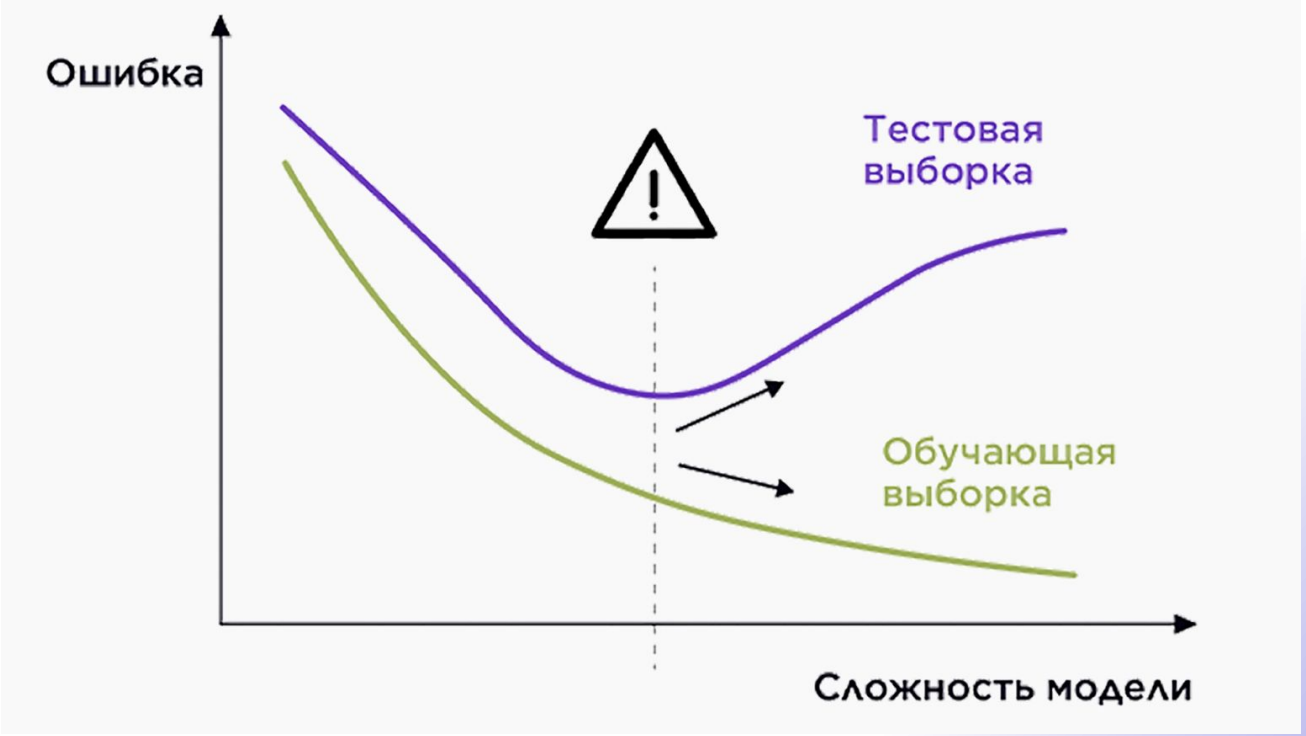
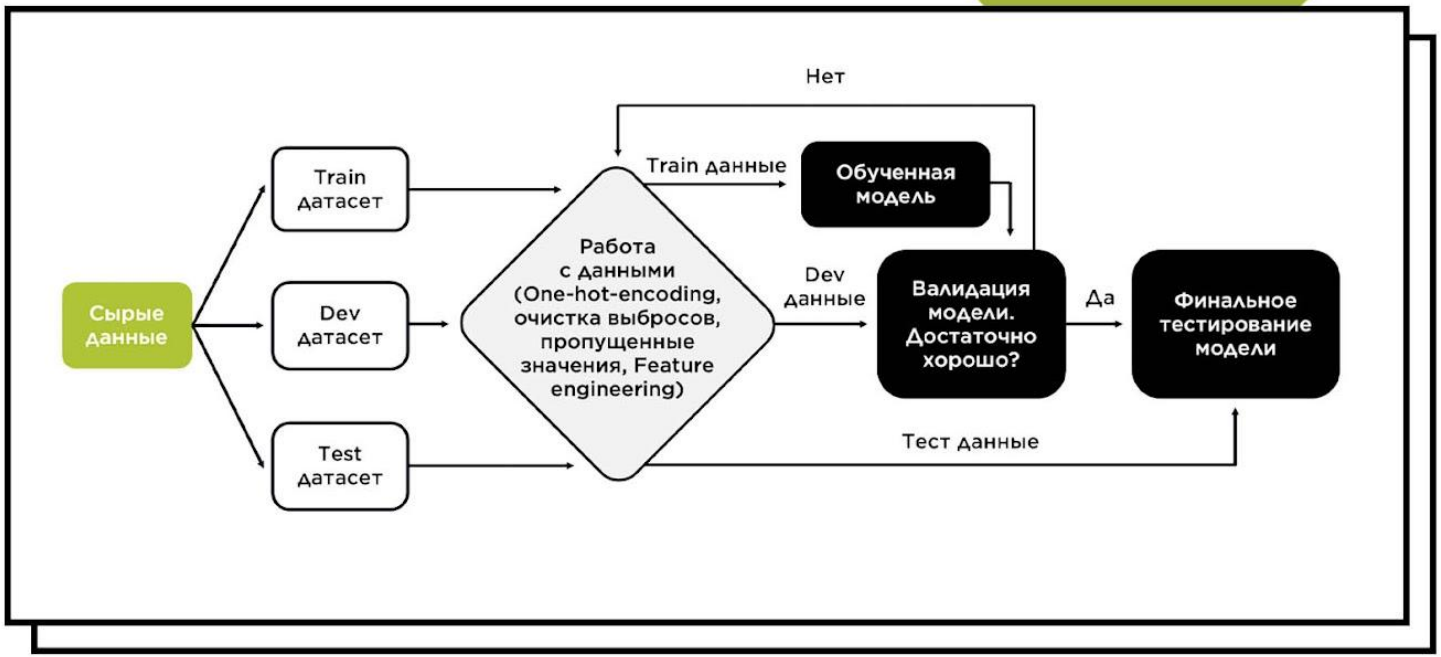


Рис. Приближение многочленом степини 5.

График функции потерь при этом выглядит вот так:




Нет
Train данные
Обученная модель
Train датасет
Работа с данными (One-hot-encoding, очистка выбросов, пропущенные значения, Feature engineering)
Dev данные
Валидация модели. Достаточно хорошо?
Да
Финальное тестирование модели
Сырые данные
Dev датасет
Test датасет
Тест данные