3.1. Постановка задач регрессии и примеры таких задач

Обычная задача машинного обучения — построить модель зависимости «выхода» от входных данных. Мы будем рассматривать **задачу регрессии.** В такой задаче на выходе модели получается непрерывная величина в заданном диапазоне.

Для начала приведем формальную постановку нашей задачи:

- Задана выборка значений признаков: x_n

$$\{x, \ldots, x_n \mid x \in R^d\}.$$

Здесь ***d*** — размерность признакового пространства и ***n*** — количество элементов в нашей выборке входных данных.

- Задана выборка соответствующих значений целевой переменной:

$$\{y, \ldots, y_n \mid y \in R\}.$$

Получаем множество исходных данных:

$$D = \{(x, y)_i\}_{i=1}^{n}$$

- Задано параметрическое семейство функций $f(\omega, x)$, зависящее от параметров $\omega \in R$ и от входных признаков *x*.

Здесь $f(\omega, x)$ — функция регрессионной зависимости. Формально это параметрическое семейство функций вида $f: W \times X \to Y$, где W — пространство параметров (весов модели) ω, X — пространство значений признаков *x* и Y — пространство целевых переменных y.

- Нужно построить модель, предсказывающую по x_i значение $\hat{y}_i$, наиболее близкое к y_i : $\hat{y}_i = f(\omega, x_i)$
- Для оценки близости предсказания к истинному значению используются функции потерь, например **MSE** — **M**ean **s**quared **e**rror, или среднеквадратичная ошибка:

$$\mathcal{L}(x, y, \omega) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\omega, x_i))^2$$

Чем меньше значение MSE, тем ближе к истинным значениям будет выход модели.

Таким образом, задача регрессии состоит в том, чтобы решить оптимизационную задачу $\mathcal{L}(x, y, \omega) \to \min_{\omega}$ и найти подходящий набор параметров ω.

Примеры задач регрессии

В целом задача регрессии — это прогнозирование вещественного числа на основе выборки объектов с различными признаками. Так, можно смоделировать задачу нахождения стоимости недвижимости или количества кликов пользователей на баннеры рекламы.

Давайте рассмотрим задачу прогнозирования трафика автомагистрали:

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	date_time	traffic_volume
0	None	288.28	0.0	0.0	40.0	Clouds	scattered clouds	2012-10-02 09:00:00	5545.0
1	None	289.36	0.0	0.0	75.0	Clouds	broken clouds	2012-10-02 10:00:00	4516.0
2	None	289.58	0.0	0.0	90.0	Clouds	overcast clouds	2012-10-02 11:00:00	4767.0

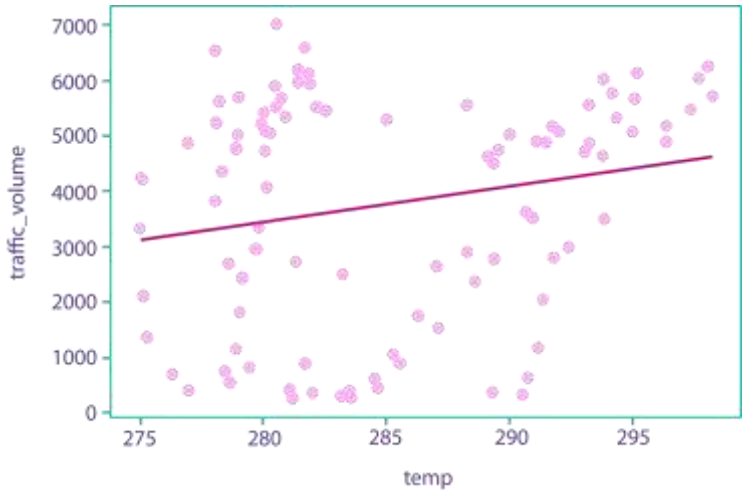
Здесь, первый столбец — это индексы строк датасета, а последний — целевая переменная ***y***, значение трафика, зависящее от вектора признаков $\boldsymbol{x} = (x^{holiday}, x^{temp}, \ldots, x^{date\_time})$. Чтобы спрогнозировать значение трафика, мы можем взять линейную модель, например *линейную регрессию* (подробнее методы регрессии рассмотрим далее в курсе, здесь же приведем только функцию регрессионной зависимости):

$$\hat{y}_i = f(\omega, x_i) = \sum_{j \in F} \omega_j \cdot x_i^j = \omega^T x_i$$

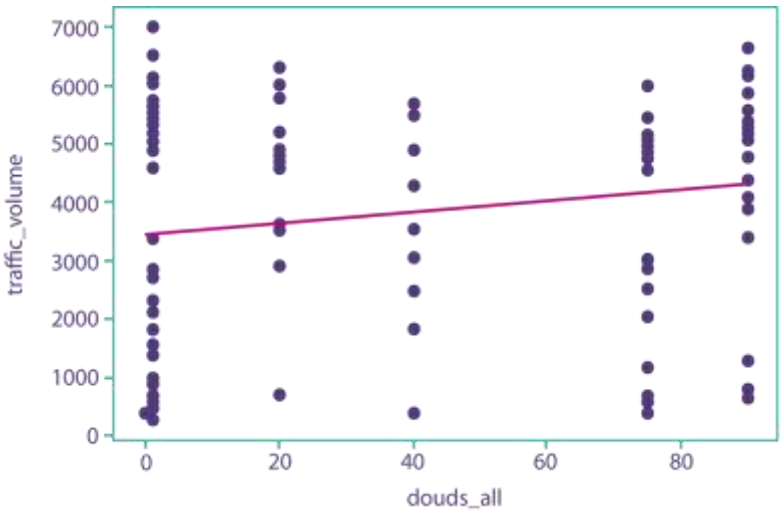
Линейная регрессия является скалярным произведением вектора весов ω и вектора признаков x_i, то есть их поэлементным произведением по всем признакам j из пространства признаков ***F***.

Затем с помощью минимизации среднеквадратичной ошибки **MSE** найдем подходящий набор параметров $\hat{\omega}$.

Для примера, если взять из признаков только температуру (столбец „temp”) и построить зависимость трафика от температуры, то получим следующий график:



Как видим, данные сильно разбросаны, так что только по показаниям температуры вряд ли получится сделать предсказание, близкое к истинному. Если же учитывать только погодный показатель (столбец „clouds_all”), то получим еще менее надежные результаты:



Но если объединить все признаки в линейную комбинацию (что и представляет собой линейная регрессия), то предсказания модели будут близки к истинным.

3.2. Метрики, критерии качества для задач регрессии

Метрики, критерии качества для задач регрессии. Зачем минимизировать ошибку

Выбор метрики качества

Метрика качества — это оценка качества нашей модели. Причем для конкретной задачи может возникать целая иерархия метрик.

Рассмотрим на примере. Пусть перед нами стоит задача создать предсказательную модель для портала поиска недвижимости для ее последующей аренды или покупки. В таком случае иерархия метрик может иметь следующий вид:

- Самый верхний уровень: будущий доход сервиса — практически невозможно измерить в моменте, сложным образом зависит от совокупности наших усилий как в выборе и обучении модели, так и в продакшене
- Доля удовлетворенных качеством подбора жилья экспертов, на которых мы протестируем модель, прежде чем представлять ее пользователям
- Функция потерь, на которой мы будем обучать нашу модель

Как видно, можно выделить следующую закономерность: во-первых, бизнес-метрики, интересующие нас в первую очередь, практически неимплементируемы в процесс обучения модели; во-вторых, функция потерь ничего не говорит нам о том, насколько хорошо мы решили поставленную задачу.

Поэтому в общем случае метрика качества не совпадает с функцией потерь, и нужно уметь различать эти понятия:

Метрика — внешний критерий качества, зависящий от предсказанных значений, но не от параметров модели.

Функция потерь — критерий «обучаемости» нашей модели. Возникает, когда мы переходим от построения модели к задаче оптимизации, то есть поиску оптимальных параметров модели.

Теперь вернемся к задаче регрессии. Мы уже рассматривали такую функцию потерь, как среднеквадратичная ошибка MSE. Такая функция может быть одновременно и метрикой в силу постановки задачи. Мы стараемся не точно предсказать значение, а найти наиболее близкое в интервале по всей совокупности объектов в выборке.

В таком случае появляются другие вопросы о характеристиках модели. Важно понимать, подходит ли выбранная метрика под решаемую задачу. Например, MSE, с которой мы познакомились в прошлом лонгриде, накладывает большой штраф на сильные отклонения в выборке за счет возведения в квадрат, но в то же время ничего не говорит об «абсолютном» качестве модели. Если мы получим значение MSE в 10 000, то отклонение составит $\sqrt{10\,000} = 100.$ Тогда при предсказанном значении в интервале [0; 200] смысла от такой модели будет мало. Если же предсказываются значения из интервала [0; 20 000], то отклонение в 100 получается не таким уж и большим.

Далее мы рассмотрим различные метрики регрессии и приведем характеристики каждой из них.

Метрики качества для задачи регрессии

Итак, давайте повторим обозначения:

$\{x, ..., x_n \mid x \in R^d\}$ — выборка значений признаков;

$\{y, ..., y_n \mid y \in R\}$ — выборка соответствующих значений целевой переменной;

$f(\omega, x) = \omega^T x$ — функция линейной регрессии.

В прошлый раз мы уже познакомились с одной из самых распространенных метрик задачи регрессии — **MSE** (**M**ean **s**quare **e**rror):

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2$$

Как мы уже рассмотрели выше, MSE-функционал сильно чувствителен к выбросам за счет возведения в квадрат; однако о том, насколько хорошо наша модель решает поставленную задачу, используя MSE, вывод мы сделать не можем.

Также есть и другие метрики:

- **MAE** (**M**ean **a**bsolute **e**rror):

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - w^T x_i|$$

- **MAPE** (**M**ean **a**bsolute **p**ercentage **e**rror):

$$\frac{1}{n}\sum_{i=1}^{n}|\frac{y_i - w^T x_i}{y_i}|$$

Используется, чтобы учитывать масштаб целевой переменной. То есть мы минимизируем *процент*,на который наша модель ошибается. MAPE — метрика с очень простой интерпретацией, измеряемая в долях или процентах. Если у вас получилось, например, что MAPE = **n**%, то это говорит о том, что ошибка составила **n**% от реальных значений. Однако такая метрика нестабильна: любое сильное отклонение может значительно повлиять на размер ошибки.
- **MASE** (**M**ean **a**bsolute **s**caled **e**rror):

$$\frac{\sum_{i=1}^{n}|y_i - w^T x_i|}{\frac{n}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|}$$

Функционал MASE является отличным показателем точности модели: не зависит от масштаба данных, является симметричным и дает несмещенную оценку. Хотя на практике данную метрику почти невозможно интерпретировать.

- **R2** (коэффициент детерминации):

$$1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y_i})^2}$$

Здесь $\overline{y}$ — среднее истинного значения *y*.

Заметим, что дробь в формуле можно понимать как отношение дисперсии, *объясняемой моделью*, к дисперсии выборки (дисперсии константной модели, которая всегда предсказывает выборочное среднее). Тогда формулу можно переписать следующим образом:

$$R^2 = 1 - \frac{MSE}{\sigma_y^2}$$

В отличие от других метрик, эту мы стремимся максимизировать при обучении модели, т. к. она максимальна при MSE = 0, например. Значение R^2 меньше 0 означает, что рассматриваемая модель *хуже* константой.

Для других задач можно использовать иные метрики качества, в своей задачи вы должны отталкиваться от того, что модель должна делать хорошо на выходе.

Итак, давайте подведем итоги по полученным метрикам в виде таблицы:

Метрика	Достоинства	Недостатки
MSE	Позволяет накладывать большие штрафы на сильные отклонения, практически обнуляя соответствующие веса признаков	Нестабильна при сильных отклонениях в выборке
MAE	Легко интерпретировать	В отличие от MSE не штрафует за сильные отклонения
MAPE	Легко интерпретировать	Нельзя использовать, если есть нулевые или близкие к нулю значения, также нет верхнего предела процентной ошибки для слишком больших предсказаний — из-за отклонений в выборке можно получить ошибку в сотни процентов
MASE	Лишена недостатков MAE и инвариантна к масштабу данных, поэтому ее можно использовать для сравнения прогнозов по разным наборам данных	Не интерпретируется
R^2	Легко интерпретировать — определяет долю изменений, обусловленных влиянием признаков на целевую переменную	Не уменьшается при добавлении новых признаков, независимо от их связи с целевой переменной; поэтому модели с разным количеством признаков с помощью R^2 сравнить не получится

Как видим, нет универсальной метрики, отвечающей всем требованиям, поэтому стоит не зацикливаться на одной оценке качества, а проводить комплексный анализ модели.

Связь квадратичной ошибки и метода максимума правдоподобия

Давайте разберемся в этом вопросе на примере **линейной регрессии.** Для начала введем несколько определений:

- Функция правдоподобия

Пусть дано параметрическое семейство распределений вероятности $\{P_\omega\}_{\omega \in W}$ и выборка $X_1, X_2, ..., X_d \sim P_\omega$.

Тогда для фиксированной реализации выборки x (при работе с табличными данными *x* — строчка датасета, а X_i — i-й столбец датасета) совместное распределение этой выборки $f_X(x \mid \omega),\ x \in R^d$ называется *функцией правдоподобия*.

- Метод максимизации правдоподобия

Это метод оценивания неизвестного параметра путем максимизации функции правдоподобия.

Неформально правдоподобие позволяет оценивать неизвестные параметры (параметры модели ω), основанные на известных результатах (множество исходных данных). Функция правдоподобия — это функция вероятности, поэтому и работать изначально мы будем с вероятностью, а не функцией потерь.

С математической точки зрения регрессия — зависимость математического ожидания случайной величины от других случайных величин:

$$E(y|x) = f(x)$$

Мы можем представить регрессию как сумму неслучайной (***f***) и случайной частей:

$$y = f(w, x) + \epsilon$$

Здесь, $\omega \in W$ — пространство параметров. $x \in X$ — пространство признаков и Y — пространство зависимых (целевых переменных). Также стоит отметить, что ϵ — аддитивная случайная величина, имеющая нормальное (гауссово) распределение с нулевым средним и фиксированной дисперсией. Такое предположение позволит нам в дальнейшем использовать, например, метод наименьших квадратов для линейной регрессии.

$f(\omega, x)$ - функция регрессионной зависимости. Формально это параметрическое семейство функций вида $f: W \times X \to Y$.

Теперь приведем формулировку **задачи регрессии** с прошлого раза, но в несколько другой форме:

- Задана выборка значений признаков: x_n

$$\left\{x, \ldots, x_n \mid x \in R^d\right\}.$$

Здесь ***d*** — размерность признакового пространства и ***n*** — количество элементов в нашей выборке входных данных.

- Задана выборка соответствующих значений целевой переменной:

$$\left\{y, \ldots, y_n \mid y \in R\right\}.$$

Получаем множество исходных данных:

$$D = \{(x,\ y)_i\}_{i=1}^{n}$$

- Задано параметрическое семейство функций $f(\omega,\ x)$, зависящее от параметров $\omega\ \in\ R$ и от входных признаков *x*.
- Нужно найти наиболее вероятные параметры (или **максимум правдоподобия**)

$$\hat{w} = \underset{w \in R^W}{\operatorname{argmax}}\, p(D|w, f)$$

Однако на деле же работают не с вероятностью, а с функцией ошибки $\mathcal{L}(x, y, \omega)$, например MSE: $\mathcal{L}(x, y, w) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2$

Давайте разберемся на примере линейной регрессии, почему такой переход возможен.

Итак, модель **линейной регрессии**:

$$y = f(w, x) + \epsilon,\ \ f(w, x) = w^T x,\ \ \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Запишем модель в новом виде:

$$p(D|w, f) = p(y_i|x_i, w) =$$

$$\sum_{j=1}^{n} w_j x_{ij} + \mathcal{N}(0, \sigma^2) = \mathcal{N}(\sum_{j=1}^{n} w_j x_{ij}, \sigma^2)$$

Максимизация правдоподобия, то есть нахождение максимума ***p***, — это то же самое, что и максимизация его логарифма:

$$\log p(y_i|x_i, w) = \log \prod_{i=1}^{N} \mathcal{N}(\sum_{j=1}^{n} w_j x_{ij}, \sigma^2) =$$

$$\sum_{i=1}^{N} \log \mathcal{N}(\sum_{j=1}^{n} w_j x_{ij}, \sigma^2) = -\frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - w^T x_i)^2 \Leftrightarrow$$

$$\hat{w} = \underset{w \in R^W}{\operatorname{argmax}}\, p(D|w, f) = \underset{w \in R^W}{\operatorname{argmax}} -\mathcal{L}(x, y, w) = \underset{w \in R^W}{\operatorname{argmin}}\, \mathcal{L}(x, y, w)$$

Таким образом, мы свели **метод максимального правдоподобия** к минимизации среднеквадратической ошибки. Здесь мы пользовались предположением, что случайная величина распределена нормально.

3.3. Обучающая и тестовая выборка. Скользящий контроль

Мы уже вводили понятия признакового пространства и целевой переменной:

- Задана выборка значений признаков: x_n

$$\left\{x, \ldots, x_n \mid x \in R^d\right\}.$$

Здесь ***d*** — размерность признакового пространства и ***n*** — количество элементов в нашей выборке входных данных.

- Задана выборка соответствующих значений целевой переменной:

$$\left\{y, \ldots, y_n \mid y \in R\right\}.$$

Получаем множество исходных данных:

$$\mathrm{D} = \{(x, y)_i\}_{i=1}^{n}.$$

Таким образом, ***D*** — это вся выборка наших данных. Почему бы нам просто не обучить модель на всей этой выборке? В прошлом модуле мы уже говорили о важности данных для решения бизнес-задач с помощью машинного обучения. Чем больше данных, тем выше может быть получена точность модели. Здесь мы имеем в виду, что проверка модели происходит на реальных данных, отличных от исходной выборки. В прошлый раз мы уже говорили о том, что для оценки работоспособности модели в реальной бизнес-задаче используется не одна метрика, являющаяся функцией потерь, а целая иерархия метрик, в которой наиболее значимые метрики основываются на внешних статистиках (например, на отзывах асессоров), а не на доступной выборке.
Однако в реальности ***D*** — это все доступные нам данные, и процесс оценки качества модели — ее *тестирование (или валидация)* — происходит только на выборке ***D***, как и процесс обучения модели. Поэтому перед нами встает задача разделения всей выборки на *обучающую и тестовую* части.

Но не все так просто. Ошибка на обучающей выборке является смещенной. Не вдаваясь в математику, модель отражает зависимости только на тех данных, на которых она обучалась. А вот на тестовых данных предположение о близости выхода модели к истинному значению уже не будет выполняться. На обучающей выборке мы получаем оптимистически заниженную функцию потерь и, соответственно, высокую точность предсказаний. В то же время на данных вне обучающей выборки (на тестовой) точность низкая. Такая проблема возникает при переобучении модели, о которой мы уже говорили в видеоролике.

Поэтому появляется необходимость разбиения всей выборки на обучающую и тестовую (валидационную) части определенным образом.

Первым соображением мы можем разбить нашу выборку на части случайно. Но в таком случае качество метрики не улучшится, и проблема переобучения не решится. Можно брать среднюю ошибку по такому разбиению. Метрика при этом также усредняется, но оценка останется смещенной. Чтобы получить *несмещенную среднюю ошибку*, будем использовать **скользящий контроль**:

1. Разбиваем множество исходных данных на две непересекающиеся подвыборки ***J*** различными способами:

$$D = \{D_i^k \cup D_i^m\}_{i=1}^J$$

Здесь k — длина обучающей подвыборки и $m = N - k$ — длина тестовой подвыборки.

2. Для каждого разбиения из ***J*** обучаем модель и получаем метрику. Затем считаем среднее по всем полученным ***J*** значениям метрики.

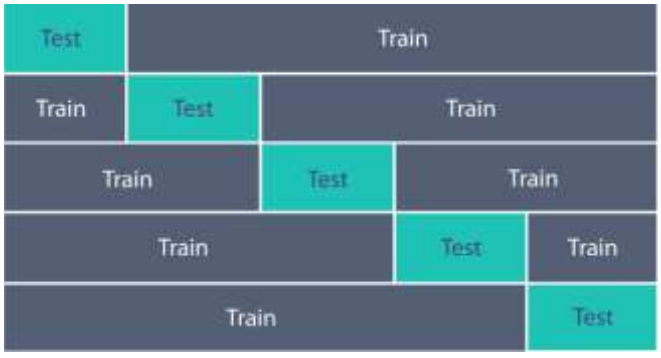
Мы рассмотрим подробно один из вариантов скользящего контроля, а именно **кросс-валидацию**:

1. Делим выборку на **k блоков (folds)**, непересекающихся и равных по объему
2. Производим **k** итераций:
   a. обучаем заранее заданную модель на **k – 1** фолде
   b. тестируем модель на **1** фолде, не участвующем в обучении, и считаем метрику q_i на ***i***-йитерации
   c. смещаем инициализацию фолдов для обучения на **1**
3. Считаем среднее по полученным метрикам:

$$q = \frac{1}{k}\sum_{i=1}^{k} q_i$$

На рисунке ниже показан пример инициализации фолдов для кросс-валидации.

Первый фолд **F[0]** будет тестовым, остальные k – 1 фолдов пойдут в обучающую выборку **F[1 : k – 1]**. После итерации обучения и тестирования назначаем следующий фолд в качестве тестового — **F[1]**, а для обучающей выборки — фолды **F[2 : k – 1] + F[0]**. Затем снова итерация обучения + тестирования и так далее.



3.4. Методы решения задачи регрессии: линейная регрессия, метод k-ближайших соседей, решающие деревья, ансамбли моделей

В данном материале рассмотрим алгоритмы решения задачи регрессии (предсказание численной величины по имеющимся признакам), а именно такие классические подходы как:

- Линейная регрессия
- Метод k-ближайших соседей
- Решающие деревья
- Ансамбли моделей

Линейная регрессия

Линейная регрессия — один из базовых и самых простых методов решения задачи регрессии. Модель линейной регрессии хорошо находит линейные зависимости данных, поскольку является линейной комбинацией вектора признаков и вектора весов. Модель линейной регрессии выглядит следующим образом:

$$a_{linreg}(\boldsymbol{x}) = \hat{y} = \sum_{i=1}^{d} w_i x_i + w_0,$$

где $\boldsymbol{x} = (x_1, \ldots, x_d)$ — объекты выборки, $\boldsymbol{w} = (w_1, \ldots, w_d)$ — веса модели $a_{linreg}(\boldsymbol{x}), w_0$ — смещение, $\hat{y}$ — предсказание модели. Если $w_0 = 0$, то $a_{linreg}(\boldsymbol{x}) = \hat{y} = \sum_{i=1}^{d} w_i x_i = \langle \boldsymbol{w}, \boldsymbol{x} \rangle$.

Рассмотрим более простой вариант линейной регрессии, когда мы пытаемся восстановить зависимость целевой переменной от одного признака. Пусть целевая переменная зависит от входного признака следующим образом:

$\boldsymbol{y} = w_0 + w_1 \boldsymbol{x} + \boldsymbol{\varepsilon}$, где $\boldsymbol{y} = (y_1, \ldots, y_n)$ — целевая переменная, $\mathbf{x} = (x_1, \ldots, x_n)$ — входные данные, w_0, w_1 — параметры, которые мы будем оценивать, и $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)$ — шум, нормальное распределение с мат. ожиданием $\mathbb{E}(\epsilon_i | x_i) = 0$ и дисперсией $\mathbb{V}(\epsilon_i | x_i) = \sigma^2$, т. е. $(\epsilon_i | x_i) \sim \mathcal{N}(0, \sigma^2)$. Тогда $y_i | x_i \sim \mathcal{N}(\mu_i, \sigma^2)$, где $\mu_i = \omega_0 + \omega_1 x_i$. Оценим коэффициенты w_0 и w_1 и с помощью **метода максимального правдоподобия**. В нашем случае правдоподобие будет иметь вид:

$$\mathcal{L} = \prod_{i=1}^{n} f(x_i, y_i) = \prod_{i=1}^{n} f_x(x_i) f_{y|x}(y_i|x_i) = \prod_{i=1}^{n} f_{y|x}(y_i|x_i) = \mathcal{L}_1 \cdot \mathcal{L}_2$$

где $\mathcal{L}_1 = \prod_{i=1}^{n} f_x(x_i)$ и $\mathcal{L}_2 = \prod_{i=1}^{n} f_{y|x}(y_i|x_i)$.

Функция $\mathcal{L}_1$ не содержит параметры w_0 и w_1, поэтому рассмотрим подробнее $\mathcal{L}_2$ — условную функцию правдоподобия:

$$\mathcal{L}_2 \equiv \mathcal{L}(\omega_0, \omega_1, \sigma) = \prod_{i=1} f_{y|x}(y_i|x_i) \propto \sigma^{-n} \exp\{-\frac{1}{2\sigma^2} \sum_i (y_i - \mu_i)^2\}$$

а логарифм от $\mathcal{L}_2$:

$$l(\omega_0, \omega_1, \sigma) = -n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - (\omega_0 + \omega_1 x_i))^2.$$

Далее путем максимизации $l(\omega_0, \omega_1, \sigma)$ по σ получаем оценку: $\hat{\sigma}^2 = \frac{1}{n} \sum_i \hat{\epsilon}_i^2$.

Мы можем получить оценки параметров w_0 и w_1:

$$\widehat{\omega_1} = \frac{\sum_{i=1}^{n} (x_i - \overline{x_n})\,(y_i - \overline{y_n})}{\sum_{i=1}^{n} (x_i - \overline{x_n})^2}$$

$$\widehat{\omega_0} = \bar{y}_n - \widehat{\omega_1} \bar{x}_n$$

где $\bar{x}_n$ и $\bar{y}_n$ — средние значения ***x*** и ***y*** соответственно. Полученные оценки идентичны тем, которые могут быть получены и путем использования среднеквадратичной ошибки. Абсолютные ошибки обычно не используются, поскольку они не дифференцируемы и по ним нельзя понять, насколько модель близка к правильному прогнозу.

Перечислим преимущества и недостатки линейной регрессии.

К преимуществам данного метода относят легкую имплементацию и интерпретацию: мы можем сказать, почему модель принимает решение, посмотрев на веса w (чем больше вес w_i, тем более важен $i^{\text{ный}}$ признак).

Недостатки:

- Плохо работает в случаях, когда в данных существенно нелинейные зависимости. Это происходит очень часто, особенно при наличии категориальных признаков

- Плохо работает, если в данных есть линейно зависимые или похожие друг на друга признаки

- Лучше работает, если входные данные нормализованы

Теперь рассмотрим, как можно бороться с недостатками метода.

Генерация признаков

Линейные модели все-таки могут восстанавливать нелинейные зависимости, но только после преобразования признакового пространства:

$$\boldsymbol{X} = (x_1, \ldots, x_d) \rightarrow \varphi(\boldsymbol{x}) = (\varphi_1(\boldsymbol{x}), \ldots, \varphi_m(\boldsymbol{x})).$$

Таким образом можно, например, перейти к квадратичным признакам:

$$\varphi(\boldsymbol{x}) = (x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, x_1x_2, \ldots, x_{d-1}x_d).$$

После перехода к квадратичным признакам линейная модель сможет приближать любые квадратичные закономерности. Также можно работать и с *полиномиальными признаками* более высоких порядков, генерировать новые признаки с помощью радиально-базисной функции и т. д. Существуют и некоторые рекомендации по выбору преобразований:

- Если у значений признака *j* много знаков после запятой, и они все важны, то $\log(x_j)$
- Если имеется периодическая зависимость: $\sin\left(\frac{x_j}{T}\right)$
- Если важна близость к какой-то точке: $exp\frac{||x_j - \mu||^2}{\sigma}$

Нормализация

Признаки бывают разными в плане масштаба. Например, если рассматривать данные о квартирах, то площадь, скорее всего, будет порядка 30–200 кв. м., в то время как количество близлежащих продуктовых магазинов вряд ли будет сильно больше 10, близость до метро — от 2 до 60 мин. Чтобы привести все признаки к единому масштабу, используют *нормализацию*. Наиболее популярными являются следующие типы:

- Z-нормализация:

$x' = \frac{x - \underline{x}}{\sigma}$, где $\underline{x}$ — среднее значение выборки, σ — среднеквадратичное отклонение. В этом случае большинство значений попадет в интервал $(-3\sigma;\ 3\sigma)$.

- Минимакс-нормализация:

$x' = \frac{x - \min[X]}{\max[X] - \min[X]}$, где $\min[X]$ и $\max[X]$ — минимальное и максимальное значения выборки соответственно. В этом случае все значения будут лежать в интервале (0;1), дискретные бинарные значения определяются как 0 и 1.

Если качество модели на обучающей выборке близко к идеальному, а на тестовой выборке гораздо хуже, то это может говорить о том, что модель переобучилась (ее поведение похоже на запоминание ответов из обучающей выборки). Одним из индикаторов переобучения является большая разница в весах признаков. Например, если один признак имеет вес порядка 10^{-3}, а другой 10^{3}. Для предотвращения данного эффекта используют специальный механизм штрафов за разные порядки весов, называемый *регуляризацией*. Наиболее распространенными являются два типа регуляризации:

1) L_1- регуляризация (Лассо-регрессия). Также используется для отбора признаков $a_{linreg_{lasso}}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \lambda ||\boldsymbol{w}||_1$

2) L_2 - регуляризация (гребневая регрессия). Применяется, когда независимые переменные коррелируют друг с другом $a_{linreg_{ridge}}(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \lambda ||\boldsymbol{w}||_2^2$

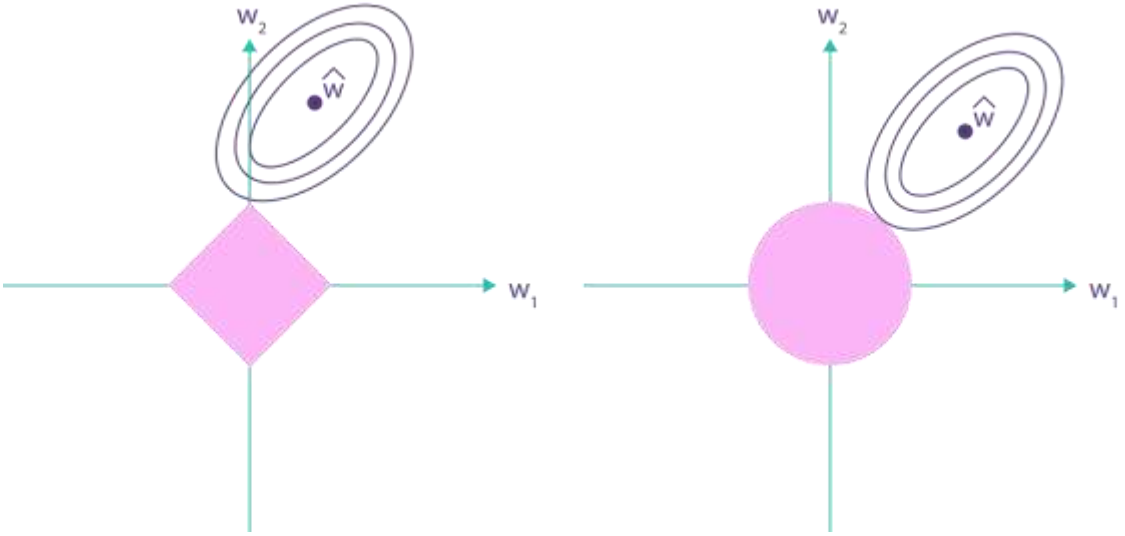


Рис. 1. Сравнение регрессий Лассо (слева) и гребневой (справа), пример для двумерного пространства независимых переменных. Фиолетовые области изображают ограничения на коэффициенты w, эллипсы — некоторые значения функции наименьшей квадратичной ошибки

Рассмотрим подробнее Рис. 1. Поскольку рассматривается функция квадратичной ошибки, которая является выпуклой, задача оптимизации в случае регрессии Лассо сводится к следующей:

$$\begin{cases} \frac{1}{d}\sum_{i=1}^{d}(w_i x_i - y_i)^2 \to \min\limits_{w} \\ \|w\|_1 \le C \end{cases}$$

где C — некоторая константа. Для гребневой регрессии оптимизационная задача выглядит похожим образом, только вместо L_1-нормы используется L_2-норма. На Рис. 1 изображены линии уровня функционала квадратичной ошибки, а также множество, определяемое ограничением $||\boldsymbol{w}||_1 \le C$ для регрессии Лассо и $||\boldsymbol{w}||_2^2 \le C$ для гребневой регрессии.

Решение упомянутой выше оптимизационной задачи определяется точкой пересечения допустимого множества с линией уровня, ближайшей к безусловному минимуму. Из Рис. 1 следует, что эллиптические области могут (в случае регрессии

Лассо) касаться углов ромба, и при этом один из коэффициентов будет равен 0 (что невозможно в гребневой регрессии). Поэтому именно регрессия Лассо может быть использована для отбора признаков: когда из множества всех признаков выбирается такое его подмножество, что признаки в данном подмножестве будут иметь наибольшую важность для обучения и предсказания модели. Также можем сравнить изменения весов для двух типов регуляризации. Из выборки были взяты четыре признака, на которых обучались модели с L_1 и L_2-регуляризацией. Качество модели без регуляризации и с L_1 и L_2-регуляризацией примерно одинаковое $R^2 \approx 0,62$.

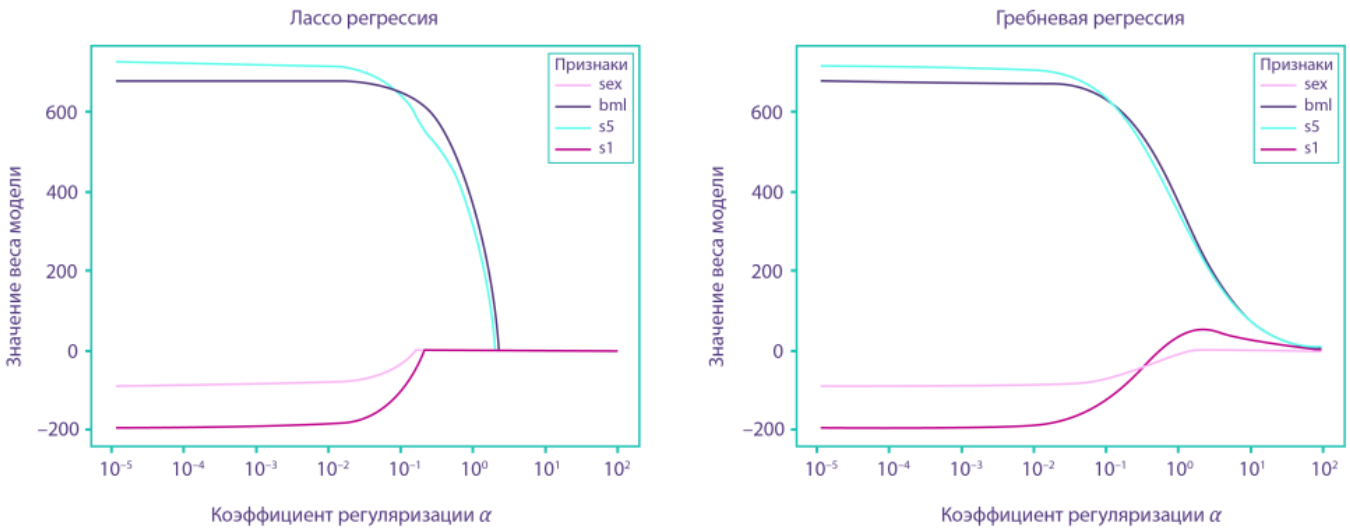


Рис. 2. Зависимость весов модели от силы регуляризации

Метод k-ближайших соседей

Данный метод используется при решении задач классификации и регрессии. Если задано расстояние между объектами, мы можем делать предсказания для какого-то конкретного объекта, используя значения целевой функции его соседей.

Алгоритм прогноза: для нового *объекта x требуется построить прогноз по K-ближайшим к нему объектам* ($x_1, ..., x_K$). Предсказания модели $a_{knn}(x)$ при этом будут иметь следующий вид:

$a_{knn}(x) = \hat{y} = \frac{\sum_{k=1}^{K} y_k}{K}$. Число ближайших соседей *K* является гиперпараметром, который нужно подбирать для каждой задачи.

Для выбора числа *k* особенно актуальна дилемма смещения и дисперсии в машинном обучении. Она заключается в следующем:

- Если модель идеально описывает все данные, она переобучилась. Есть вероятность, что она не сохранит предсказательную способность на других данных (обобщающая способность)

- Если модель плохо описывает данные, то она не «переобучилась», но, возможно, и не обучилась совсем

Примеры недообучения и переобучения метода *k*-ближайших соседей представлены на Рис. 3, зависимость ошибки от сложности модели — на Рис. 4.

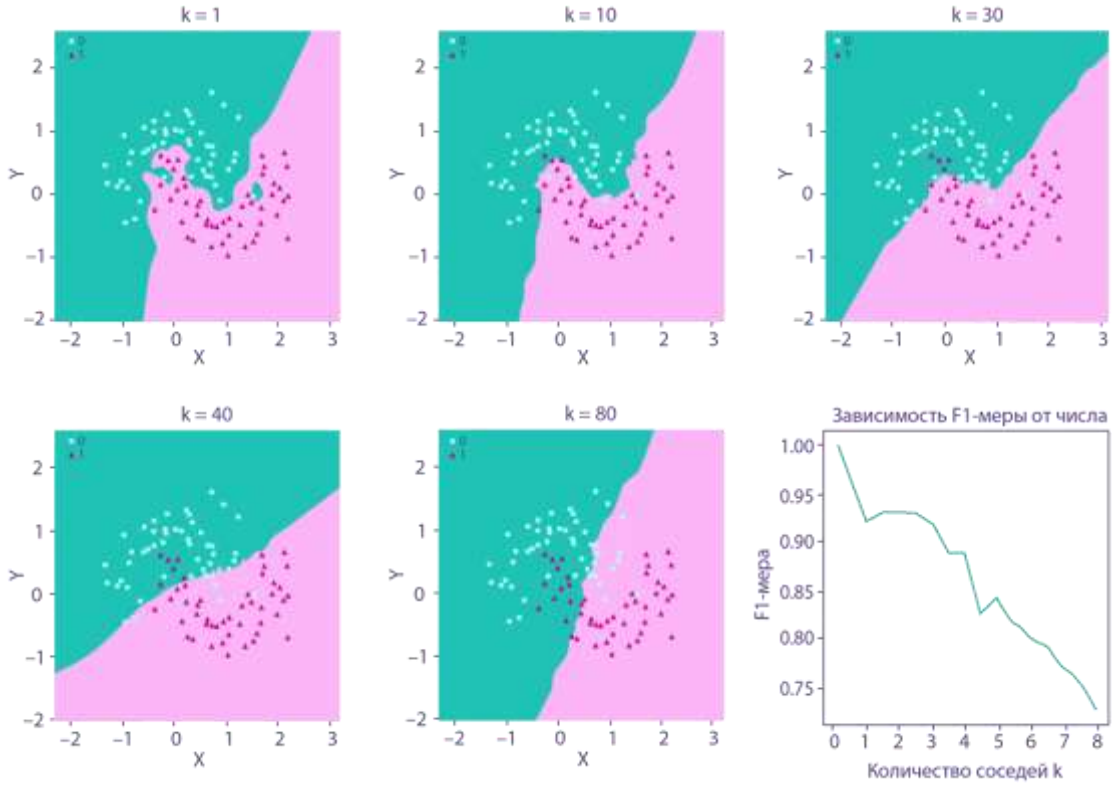


Рис. 3. Зависимость предсказания модели от числа ближайших соседей k. В исходной выборке содержится 100 точек

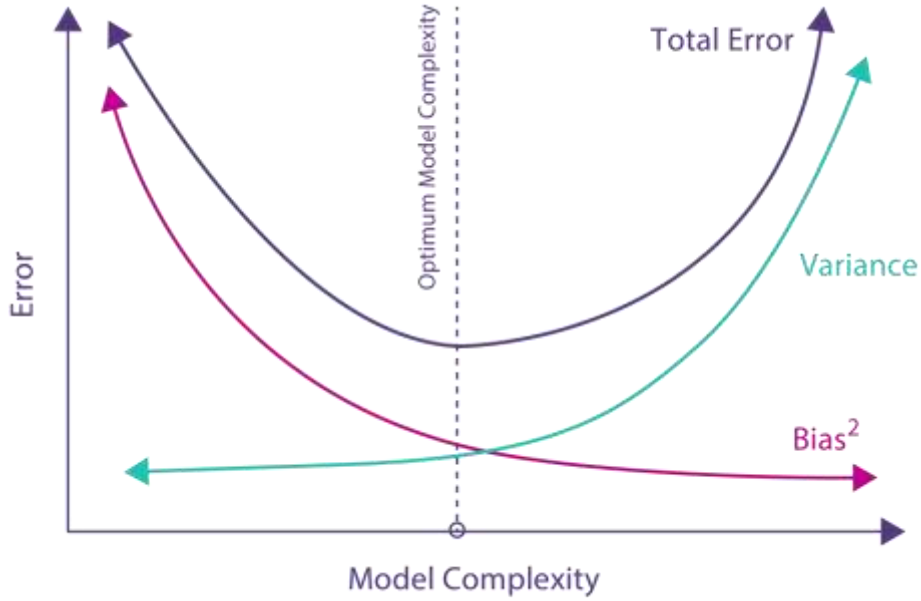


Рис. 4. Зависимость ошибки от сложности модели — демонстрация недообучения модели (левая область), переобучения (правая область) и оптимальной модели (вертикальная пунктирная линия)

Преимуществом данного метода является то, что он интерпретируемый: мы можем сказать, почему модель принимает решение, предъявив похожие объекты из обучающей выборки.

Недостатки:

- Требует задания расстояния между объектами, а также числа соседей, которые используются для принятия решения для нового объекта
- Плохо работает, если входных признаков или объектов много (для каждого объекта выборки необходимо посчитать расстояние между ним и всеми остальными объектами, что требует высокой вычислительной сложности) либо если расстояние между объектами не отражает их близость с точки зрения целевого свойства
- Требует нормализации входных данных

Взвешивание объектов в методе ближайших соседей

Существует вариация метода *k*-ближайших соседей со взвешенным учетом объектов. Пусть обучающая выборка задается объектами $(x_1, ..., x_n)$. Упорядочим их относительно рассматриваемого объекта *x*: $\rho(x, x_{i_1}) \leq \rho(x, x_{i_2}) \leq \ldots \leq \rho(x, x_{i_n})$.

Обозначим $z_1 = x_{i_1}, \ldots, z_K = x_{i_K}$.

Тогда модель для решения задачи регрессии будет иметь вид:

$$a_{knn_{weighted}}(x) = \hat{y} = \frac{\sum_{k=1}^{K} y_i w(k, \rho(x, x_k))}{\sum_{k=1}^{K} w(k, \rho(x, x_k))}.$$

Примеры весов:

1. Веса, зависящие от индекса:

   a) $w_k = \alpha^k, \alpha \in (0, 1)$

   b) $w_k = \frac{K+1-k}{K}$

2. Веса, зависящие от расстояния:

   a) $w_k = \frac{\rho(z_K, x) - \rho(z_k, x)}{\rho(z_K, x) - \rho(z_1, x)}$, если $\rho(z_K, x) \neq \rho(z_1, x)$; иначе 1

   b) $w_k = \frac{1}{\rho(x, z_k)}$

Решающие деревья

Решающие деревья используются при решении задач классификации и регрессии. В отличие от линейной регрессии не ищут линейные закономерности в данных. Решающее дерево чаще всего строится по принципу наилучшего разделения или, в математических терминах, минимизации энтропии. Процесс построения **решающих деревьев** заключается в последовательном, рекурсивном разбиении обучающего множества на подмножества с применением решающих правил в узлах. Процесс разбиения продолжается до тех пор, пока все узлы в конце

всех ветвей не будут объявлены листьями (либо пока не будет достигнута максимальная заданная глубина дерева). Пример решающего дерева представлен на Рис. 5.

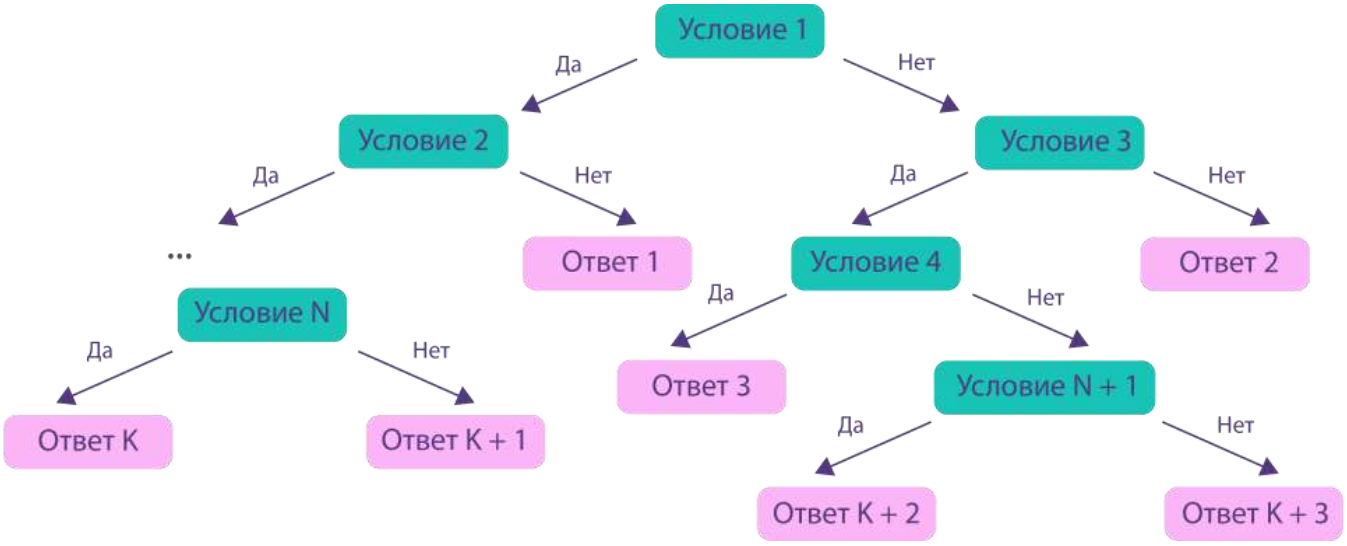


Рис. 5. Решающее дерево.

Преимущества данного метода:

- Интерпретируемый: на каждом шаге алгоритм рекурсивно проверяет, удовлетворяет ли объект тому или иному условию, и в зависимости от результата переходит к новому условию, пока не станет очевиден ответ (т. е. пока объект не окажется в листе дерева)
- Не требует нормализации входных данных
- Позволяет оценить модель при помощи статистических тестов, что дает возможность оценить надежность модели

Недостатки:

- Не могут выявить линейную зависимость
- Проблема получения оптимального дерева является NP-полной задачей — практическое применение алгоритма деревьев решений основано на эвристических алгоритмах, таких как алгоритм «жадности», где единственно оптимальное решение выбирается локально в каждом узле. То есть такие алгоритмы не могут обеспечить оптимальность всего дерева в целом

Чаще всего используется не одно дерево, а несколько, т. е. *ансамбль* (лес решающих деревьев).

Ансамбли моделей

Ансамбли объединяют множество простых моделей, которые по отдельности были бы слабыми и недообученными. Ансамбли комбинируют решения нескольких моделей с целью улучшения качества обобщающих и предсказательных свойств. Ансамбли можно собирать несколькими способами, которые будут подробнее рассмотрены далее в курсе.

Преимуществом и в какой-то мере недостатком ансамблевых алгоритмов служит эффективность работы только на выборках большого размера, от тысячи объектов. На малых выборках они будут обычно давать слишком сложную модель, склонную к переобучению. Ансамбли также не требуют нормализации входных данных.