Титульный лист материалов по дисциплине

(заполняется по каждому виду учебного материала)

ДИСЦИПЛИНА	**Технологии извлечения знаний из больших данных** (полное наименование дисциплины без сокращений)
ИНСТИТУТ	ИКБ
КАФЕДРА	**КБ-4 «Интеллектуальные системы информационной безопасности»** полное наименование кафедры)
ВИД УЧЕБНОГО МАТЕРИАЛА	**Лекция** (в соответствии с пп.1-11)
ПРЕПОДАВАТЕЛЬ	**Никонов В.В.** (фамилия, имя, отчество)
СЕМЕСТР	**3 семестр 2023/2024 уч. года** (указать семестр обучения, учебный год)

Базовая работа с табличными данными.

Ансамблевые методы: стекинг, бэггинг, бустинг

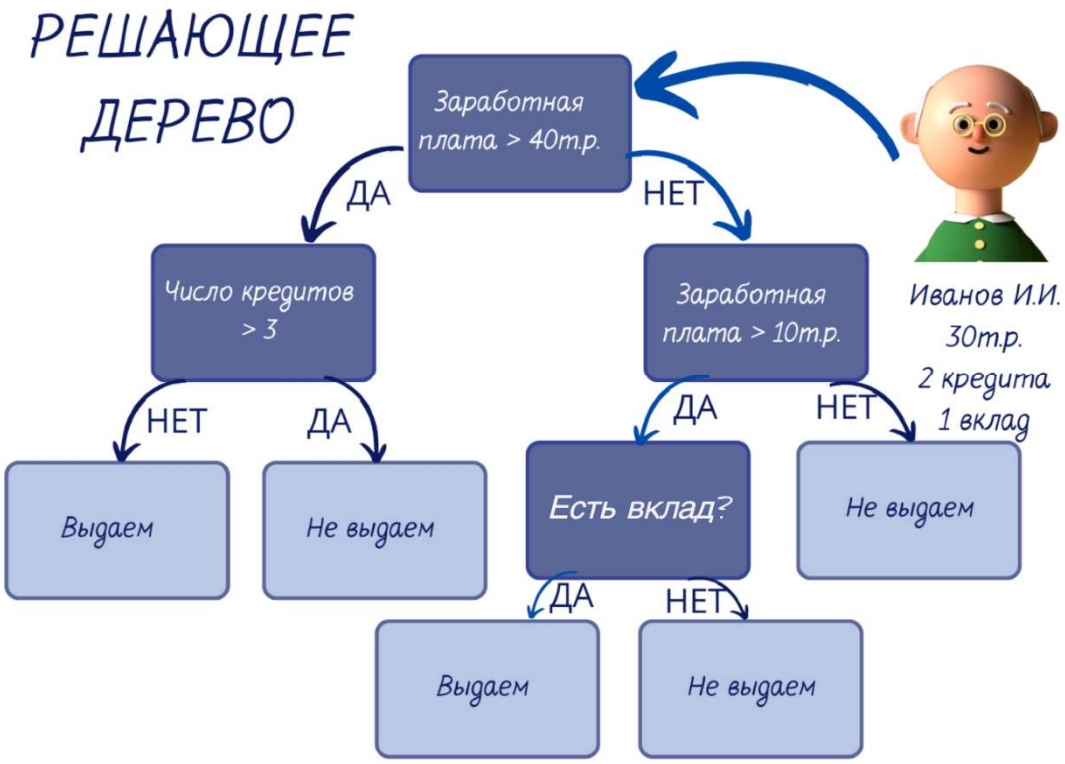
Ансамбли — методы, наиболее часто используемые в машинном обучении при работе с табличными данными. Ансамбли обычно используют в задачах классификации и регрессии, но их можно использовать и в других задачах, которые сводятся к этим двум. Например, в блоке про рекомендательные системы мы обсуждали контентный подход, который решает задачу выполнения рекомендаций как задачу регрессии. Сначала мы поговорим про основу ансамблей - решающие деревья.

Вспомним постановку задач регрессии и классификации. В обеих задачах требуется набор табличных данных: по строкам таблицы заданы объекты, например клиенты, по столбцам — признаки, то есть некоторые характеристики объектов (возраст, заработная плата, стаж работы клиента и т. д.). Кроме того, нужна разметка данных: для каждого объекта в данных должно быть известно значение целевой переменной (класс в задаче классификации или число в задаче регрессии). Имея набор размеченных данных, мы обучаем алгоритм, который будет предсказывать значение целевой переменной для новых объектов на стадии внедрения.

Ансамблирование чаще всего используют применительно к решающим деревьям, поэтому начнем блок с разбора этого метода.

Решающее дерево — это алгоритм, который делает предсказания на основе серии вопросов об объекте.

Например, если нужно научиться предсказывать, вернет ли клиент кредит, решающее дерево может выглядеть так:



Обработка каждого объекта (клиента) начинается с верхнего вопроса. Если ответ «да», то идем влево, если ответ «нет» — вправо. Далее снова решаем, идти влево или вправо, и снова повторяем процесс. В какой-то момент мы придем к ответу. У каждого объекта (клиента) свой путь в дереве и свой итоговый ответ.

Как и во всех алгоритмах машинного обучения, решающее дерево составляет не человек, а компьютер на основе данных. Поскольку решающее дерево состоит из вопросов и ответов, именно их алгоритму и нужно найти.

Сначала составляется первый вопрос. Для этого компьютер пробует все возможные вопросы вида «n-й признак больше числа x?» и выбирает тот, с которым получается наименьшая ошибка на обучающих данных.

Далее данные делятся на две части: те, которые «ушли влево» и те, которые «ушли вправо». Для каждой части снова перебираются все вопросы и выбирается лучший.

Далее процесс снова повторяется.

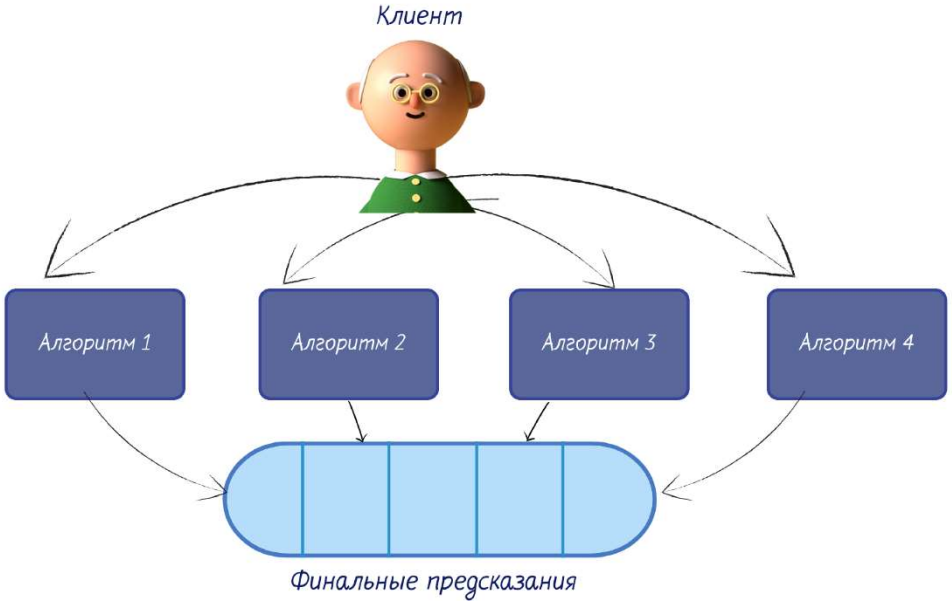
Чтобы решающие деревья работали лучше, можно устанавливать разные ограничения. Например, можно ограничить длину пути объекта в дереве: ответ должен быть получен не более, чем за 10 вопросов. А можно ограничить обучение: чтобы выбрать вопрос, можно попробовать не более, чем 100 вариантов вопросов. Какие именно выбрать ограничения, зависит от задачи и данных. Обычно делают так: пробуют разные варианты ограничений, оценивают качество на отложенных данных (тех, которые не участвовали в обучении) и выбирают ограничения, с которыми получилось наивысшее качество.

Процесс называется подбором гиперпараметров, а сами ограничения называются гиперпараметрами. «Гипер» — потому что их нельзя настроить по обучающим данным и нужно обязательно использовать отложенные данные. А вот вопросы в решающем дереве называются параметрами — их компьютер настраивает по обучающим данным. Без аккуратного подбора гиперпараметров не обходится ни один проект по машинному обучению.

С помощью описанной процедуры компьютер может построить решающее дерево, выполняющее идеальные предсказания для практически любых обучающих данных. Но на новых данных оно может работать как хорошо, так и плохо, и чаще всего выполняет предсказания с не очень высоким уровнем качества. Зато в ансамблях деревья часто достигают наивысших возможных значений качества.

ансамблирование — это простая техника, позволяющая значительно повысить качество решения задачи.

Ансамблирование заключается в том, чтобы обучить несколько алгоритмов и усреднять их предсказания.



Например, мы решаем задачу предсказания стоимости дома и обучаем ансамбль из четырех алгоритмов. Для нового дома первый алгоритм предсказывает стоимость 3 млн рублей, второй — 2,8 млн рублей, третий — 3,2 млн рублей, а четвертый — 3,5 млн рублей. Тогда предсказание ансамбля будет вычисляться как среднее предсказаний алгоритмов:

$\frac{3+2,8+3,3+3,5}{4} = 3,15$ млн рублей.

Аналогичная схема применима в задаче классификации, если алгоритмы предсказывают вероятности классов. Например, в задаче предсказания, согласится ли клиент оформить кредитную карту, алгоритмы в ансамбле могут предсказать для нового клиента вероятности оформления 40%, 80% и 85%, тогда предсказание ансамбля будет вычисляться как

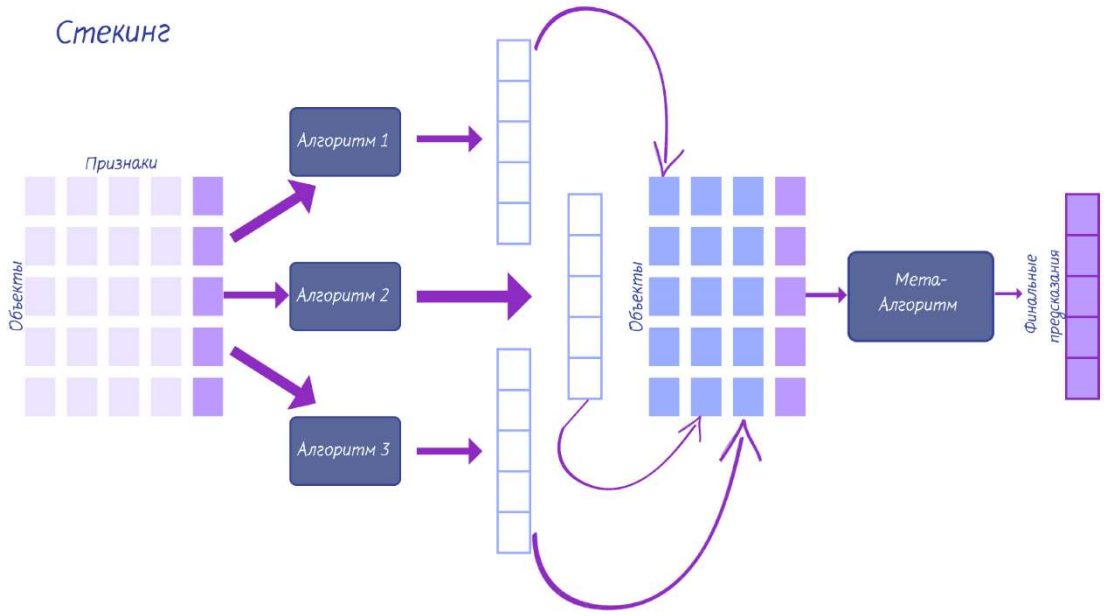
$\frac{40+80+85}{3} = 68,3\%$ Ансамбль предсказал вероятность оформления больше 50%, то есть предсказал, что клиент оформит карту.

Также бывают взвешенные ансамбли, когда каждому алгоритму присваивается вес, отвечающий за вклад этого алгоритма в предсказание целого ансамбля. Иногда устраивают «голосование» алгоритмов: за какой класс «проголосовало» больше алгоритмов, тот класс и предсказывает ансамбль.

Виды ансамблей. Чтобы составить ансамбль, важно, чтобы алгоритмы выполняли различные предсказания. Если все алгоритмы будут предсказывать одно и то же число, то после усреднения получится такое же число, и смысла в ансамблировании не будет.

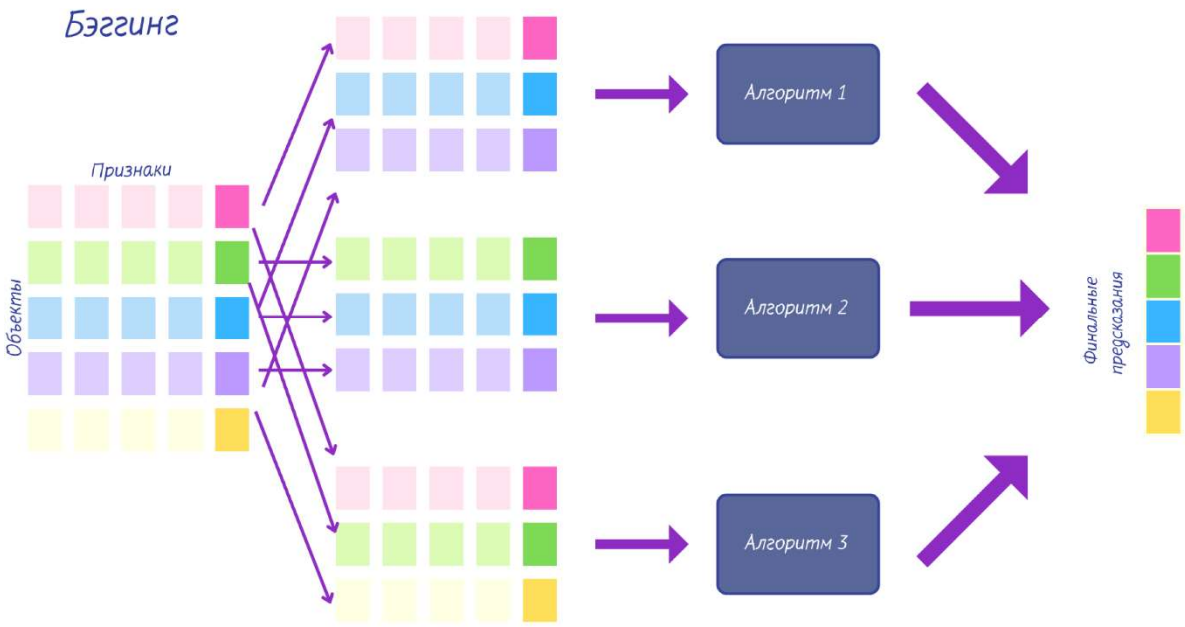
В простейшем случае можно использовать разные виды алгоритмов. Например, можно составить ансамбль из линейной модели, нейронной сети и решающего дерева (о них поговорим далее): логично ожидать, что разные алгоритмы выполняют различные предсказания. Настройка алгоритма одного вида может занимать много времени, поэтому такие ансамбли обычно состоят из небольшого числа алгоритмов.

Стэкинг. Чтобы еще больше повысить качество ансамбля алгоритмов разного вида, часто применяют дополнительный метод, называемый стекинг (Stacking). У этого метода интересная идея: использовать предсказания алгоритмов в качестве признаков. Разберемся подробнее. Сначала алгоритмы (в нашем примере линейная модель, нейросеть и решающее дерево) обучаются на стандартных данных, с настоящими признаками. Затем эти алгоритмы выполняют предсказания. Как мы обсудили выше, в обычном ансамбле, чтобы получить итоговое предсказание, предсказания отдельных алгоритмов усредняются. А в стекинге эти предсказания используются как признаки для нового алгоритма (его обычно называют мета-алгоритмом). Название метода идет от английского stack (складывать в стопку) — мета-алгоритм будто бы ставится сверху отдельных алгоритмов. Итоговый ансамбль сможет обнаруживать еще более сложные зависимости в данных, чем отдельные алгоритмы.

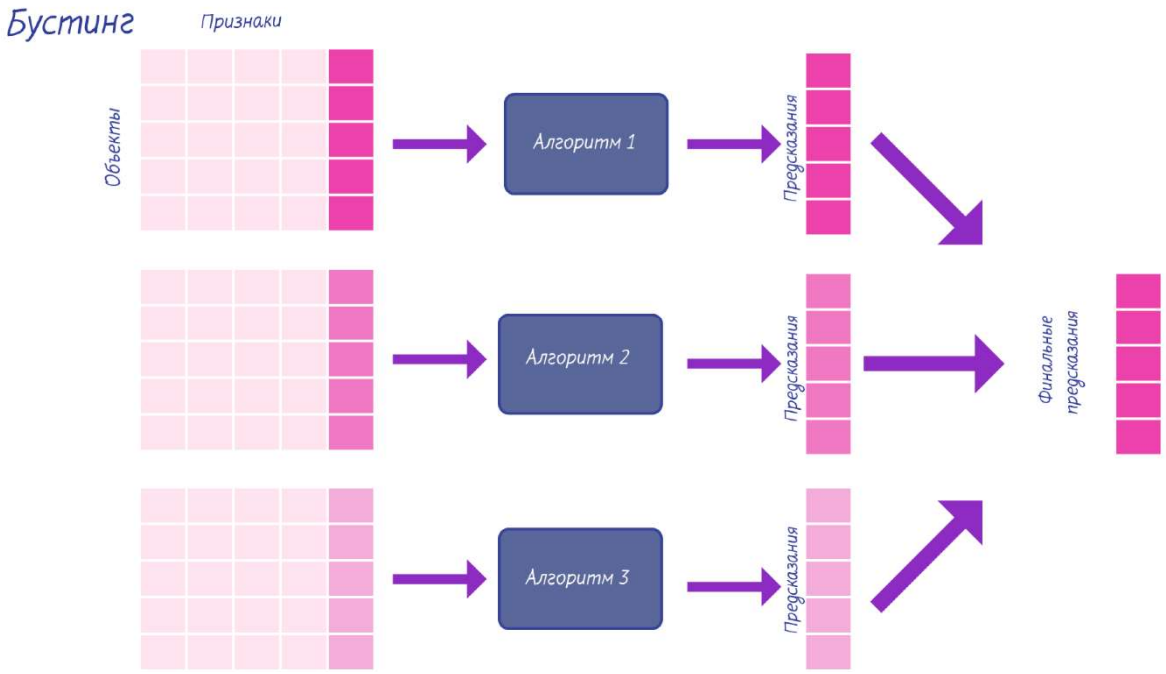


Бэггинг.Итак, различные алгоритмы для ансамбля можно получить, используя разные виды алгоритмов. Но можно составить ансамбль и из алгоритмов одного вида (чаще всего используют решающие деревья), правда, для этого потребуются специальные методы добавления разнообразия в алгоритмы. Часто используемым ансамблем такого вида является бэггинг (Bagging). Чтобы повысить разнообразие алгоритмов, в бэггинге каждый алгоритм обучается на своем наборе объектов. Например, если объекты — это клиенты, то для обучения каждого алгоритма будет случайно выбрана часть клиентов. Название алгоритма идет от английского bag (мешок) — мы как будто вытаскиваем часть содержимого мешка при обучении каждого алгоритма. С помощью бэггинга можно обучать ансамбли из сотен и тысяч алгоритмов, при этом качество будет сначала расти при увеличении числа алгоритмов, а с какого-то момента стабилизируется.

Немного усовершенствованный бэггинг над решающими деревьями называется случайный лес: лес — потому что много деревьев, а случайный — потому что деревья специально делают максимально различными с помощью внесения дополнительной случайности.



Бустинг. Еще один широко используемый метод ансамблирования называется бустинг. Этот метод тоже создает ансамбли из алгоритмов одного вида, например, из решающих деревьев. Основная идея бустинга заключается в том, что каждый следующий алгоритм обучается с целью исправить ошибки в предсказаниях предыдущих алгоритмов. Сначала по данным обучается первый алгоритм. Затем для каждого объекта вычисляется, насколько первый алгоритм ошибся в предсказании. Из этих ошибок составляется новый столбец, который нужно предсказывать. Второй алгоритм обучается предсказывать ошибки. Далее снова вычисляются ошибки, третий алгоритм обучается их предсказывать и так далее. Получается, что второй и последующие алгоритмы учатся решать не исходную задачу, а новую, придуманную задачу — задачу исправления ошибок. Название бустинг идет от английского boost (форсировать, способствовать росту) — каждый следующий алгоритм как бы совершенствует предыдущие. Но в какой-то момент нужно остановиться: начиная с некоторого алгоритма все последующие будут переобучаться, и качество начнет падать.



Ансамблирование значительно повышает качество работы алгоритмов, поэтому на практике разными ансамблями пользуются как кубиками, комбинируя их и создавая еще более сложные алгоритмы. Например, можно применить стекинг поверх бэггинга и бустинга!

Среди библиотек для обучения ансамблей решающих деревьев стоит выделить LightGBM, Xgboost и CatBoost.

Ансамблирование — это процедура усреднения предсказаний нескольких алгоритмов. Ансамблирование значительно повышает качество алгоритмов, но замедляет предсказания и делает их менее интерпретируемыми. Среди алгоритмов ансамблирования наиболее популярны стекинг, бэггинг, бустинг и случайные леса. Чаще всего используется бустинг на решающих деревьях — именно с его помощью обычно удается достичь наивысшего качества. Для этого нужно обязательно подобрать гиперпараметры!

Ошибки модели: недообучение и переобучение. Дилемма о смещении и дисперсии

Ошибки в предсказаниях могут возникать из-за качества данных. Например, даже хорошая модель может показать плохие результаты, если тестовая выборка окажется нерепрезентативной. Ошибки могут возникать и

из-за того, что модель построена плохо. С данными мы разберемся позже, а о проблемах модели поговорим сейчас.

Две основные проблемы — переобучение и недообучение. Они изображены на рисунке 1:

- Модель не выучила закономерности, присутствующие в данных, т. е. она ***недообучилась***
- Модель «запомнила ответы» из обучающей выборки, но не выучила закономерности в данных, т. е. ***переобучилась***

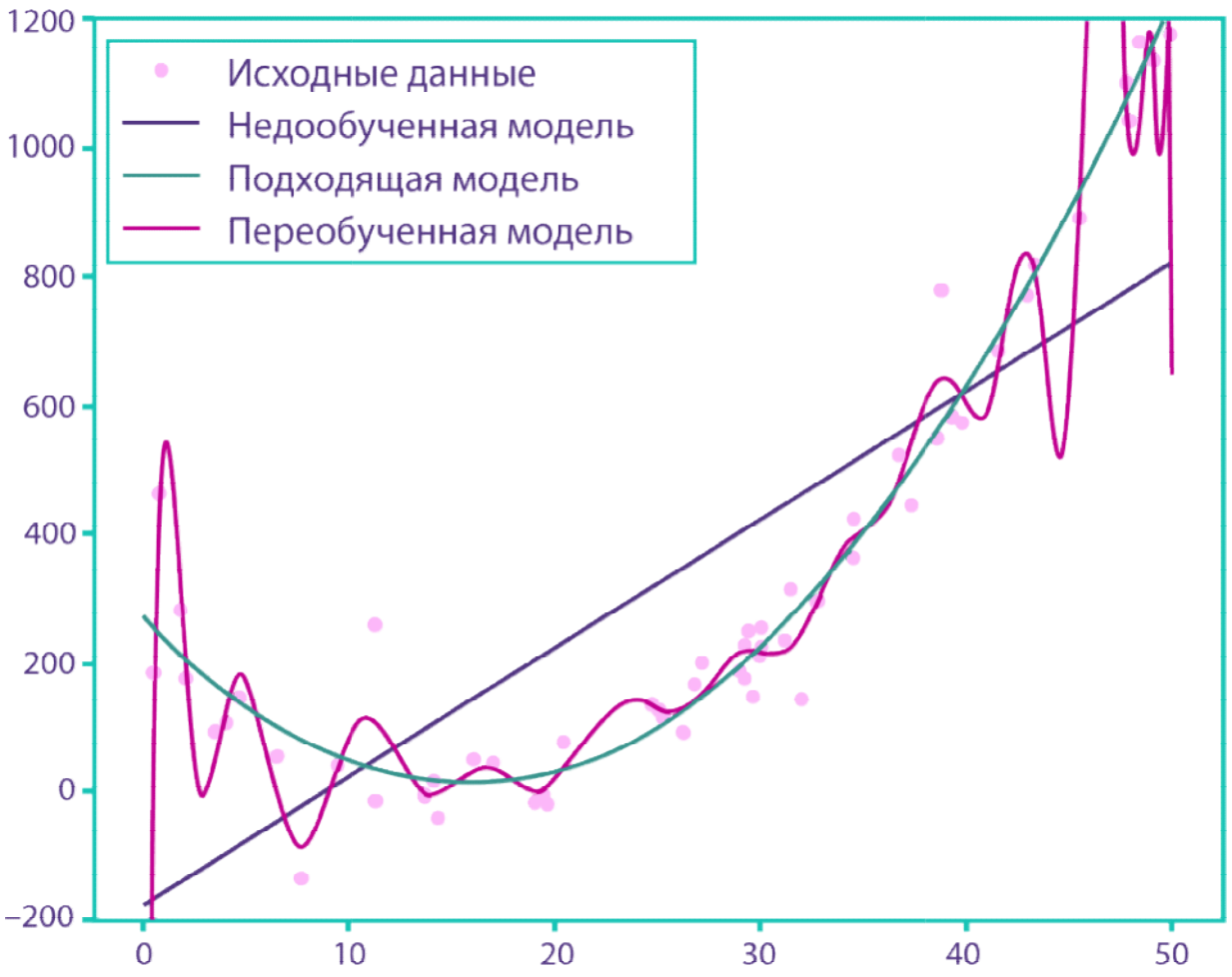


Рис. 1. Недообучение и переобучение модели

Определить причину плохого предсказания модели помогают методы валидации и общее понимание того, как строятся модели. О них мы поговорим подробнее далее и в JupyterNotebook по теме: «Беггинг, градиентный бустинг, XGBoost, Catboost».

Недообучение или переобучение модели также дает представление о степени соответствия ее сложности данной задаче: если модель недообучилась, то, вероятно, она слишком простая, чтобы выявить истинную зависимость в данных; если же модель переобучилась, она может быть слишком сложной для данной задачи, т. е. зависимость более простая, чем модель. Ответ на вопрос, как правильно сбалансировать сложность модели и ее предсказательную силу, может дать ***дилемма о смещении и дисперсии.***

Для наглядной демонстрации дилеммы о смещении и дисперсии посмотрим на результаты обучения четырех произвольных моделей для задачи классификации в Таблице 1. Напомним, что ROC AUC у идеальной модели равен 1, а у случайной модели — 0,5.

Модель №	Обучающая выборка		Тестовая выборка		Анализ
	ROC AUC	Ошибка	ROC AUC	Ошибка	
1	**0,98**	Низкая	**0,97**	Низкая	Модель точная
2	0,58	Высокая	0,57	Высокая	Модель недообучена
3	**0,98**	Низкая	0,57	Высокая	Модель переобучена
4	0,58	Высокая	**0,97**	Низкая	Нерепрезентативная тестовая выборка (данные для обучения и тестирования модели подобраны некорректно)

Табл. 1. Связь метрики, ошибки на обучающей/тестовой выборке с качеством модели

Таким образом, из Рисунка 1 и Таблицы 1 видно, что модель является адекватной, если:

- Модель описывает обучающую выборку хорошо (не обязательно идеально)
- Метрика для такой модели лучше, чем предсказание постоянным средним для задачи регрессии и случайным выбором метки класса для задачи классификации

- Качество предсказания модели на обучающей выборке сохраняется и на тестовой выборке при разных корректных разбиениях внутри выборки

Рассмотрим дилемму о смещении и дисперсии в контексте скользящего контроля, который еще называют перекретсной проверкой, или кросс-валидацией. Вспомним, как он происходит. Схематично ее можно изобразить следующим образом (см. рис. 2). Подробное описание скользящего контроля расмотрено ранее.

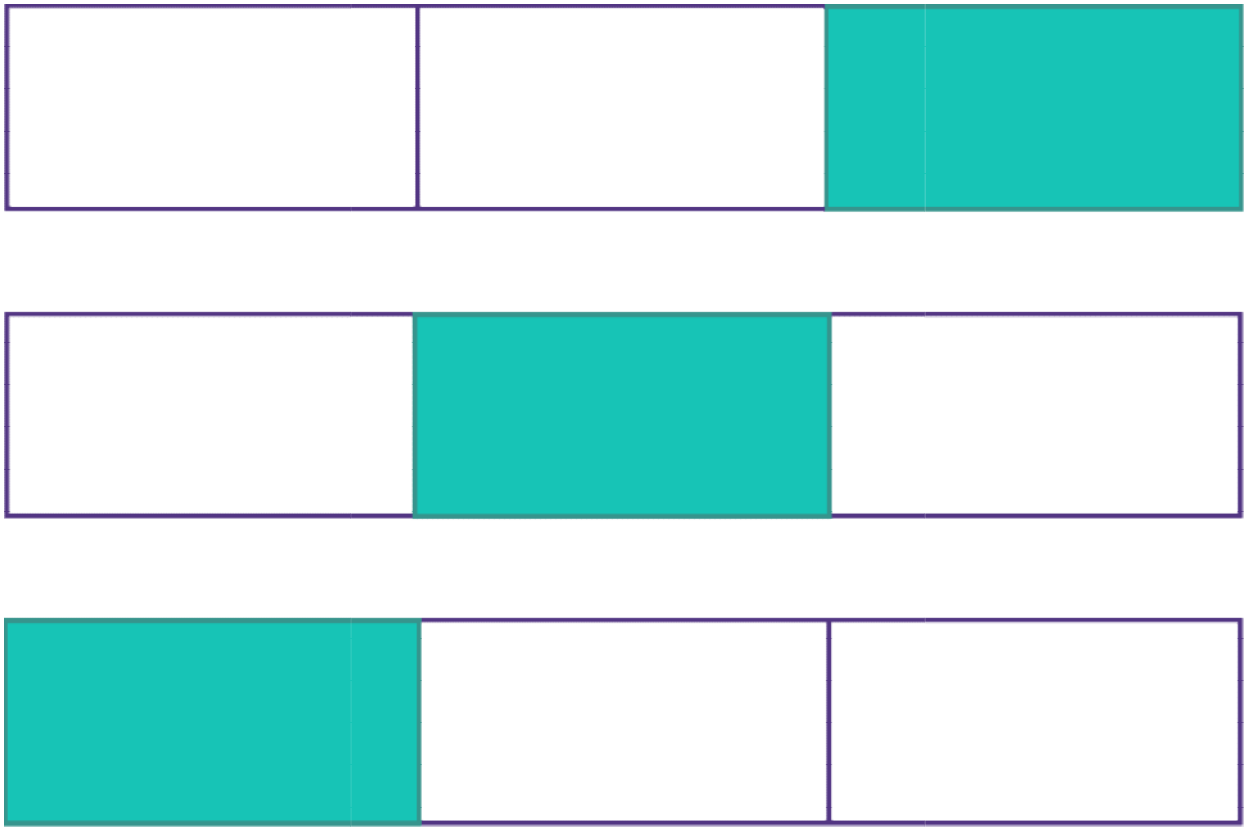


Рис. 2. Скользящий контроль. Пример для трехкратного разбиения. Зеленым цветом обозначена тестовая выборка, остальные данные используются для обучения модели

Рассмотрим результаты тестирования трех разных моделей на трех разбиениях выборки (см. Табл. 2).

Модель №	ROC AUC, фолд 1	ROC AUC, фолд 2	ROC AUC, фолд 3	Среднее ± отклонение	Анализ
1	0,98	0,96	0,93	0,96 ± 0,02	Высокий ROC AUC, малое

					отклонение => модель хорошая
2	0,98	0,54	0,79	0,77 ± 0,18	Большая дисперсия результатов => модель переобучена
3	0,56	0,52	0,59	0,56 ± 0,03	ROC AUC, близкий к 0,5 (случайное предсказание), малое отклонение => модель недообучена

Табл. 2. Анализ результатов тестирования трех моделей на кросс-валидации

С помощью скользящего контроля можно оценить стабильность результатов модели. Как видно из Таблицы 2, если результаты сильно меняются от разбиения к разбиению, т. е. дисперсия результатов большая, то модель переобучилась как модель № 2 из таблицы; если результат стабильно плохой, как у модели № 3, то модель недообучилась; а если результат стабильно хороший, как у модели № 1, то модель нормальная.

Рассмотрим описанные выше в Таблице 2 модели в контексте типов ошибок. Ошибки бывают:

- Неустранимые — ошибки, которые совершила нормальная модель. Такие ошибки, как правило, вызваны шумом в данных и тем, что реальные данные почти невозможно описать на 100 %, т. е. в принципе нельзя получить идеальную точность предсказания
- Ошибки с высокой дисперсией — ошибки, совершаемые переобученной моделью, которая сильно чувствительна к данным. Переобученная модель очень хорошо описывает одни данные и почти неприменима к другим
- Ошибки с высоким смещением — ошибки, которые совершает

недообученная модель. Недообученная модель стабильно показывает низкий результат, независимо от тестовой выборки

Перейдем к разложению ошибки на смещение и дисперсию с математической точки зрения на примере задачи регрессии. Пусть исходная выборка $\boldsymbol{x} = (x_1, \ldots, x_d)$, целевая переменная зависит от входных признаков следующим образом: $y = f(\boldsymbol{x}) + \epsilon$,

где $f(\boldsymbol{x})$ — некоторая функция, ϵ распределено по нормальному закону с нулевым математическим ожиданием и дисперсией $\sigma^2 - \epsilon \sim N(0, \sigma^2)$. Пусть у нас имеется алгоритм $a(\boldsymbol{x})$. Посмотрим, чему равно мат. ожидание квадрата отклонения предсказания алгоритма от значения целевой переменной:

$$\mathbb{E}(y - a(\mathbf{x}))^2 = \mathbb{E}(y^2 + (a(\mathbf{x}))^2 - 2ya(\mathbf{x}) =$$
$$= \mathbb{E}y^2 - (\mathbb{E}y)^2 + (\mathbb{E}y)^2 + \mathbb{E}(a(\mathbf{x}))^2 - (\mathbb{E}a(\mathbf{x}))^2 + (\mathbb{E}a(\mathbf{x}))^2 - 2\mathbb{E}[y \cdot a(\mathbf{x})] =$$
$$= Dy + Da(\mathbf{x}) + (\mathbb{E}y)^2 + (\mathbb{E}a(\mathbf{x}))^2 - 2\mathbb{E}[y \cdot a(\mathbf{x})] \overset{y=f(\mathbf{x})+\epsilon}{=}$$
$$= Dy + Da(\mathbf{x}) + (f(\mathbf{x}))^2 + (\mathbb{E}a(\mathbf{x}))^2 - 2f(\mathbf{x})\mathbb{E}a(\mathbf{x}) =$$
$$= Dy + Da(\mathbf{x}) + (\mathbb{E}(f(\mathbf{x}) - a(\mathbf{x})))^2 \equiv \sigma^2 + Var(a(\mathbf{x})) + bias^2(f(\mathbf{x}), a(\mathbf{x}))$$

Таким образом, разброс *(Var(a(**x**)))* — дисперсия ответов алгоритма *a(**x**)*, а смещение *(bias(f(**x**), a(**x**)))* — мат. ожидание разности истинного ответа и предсказания. Получается, ошибка раскладывается на три составляющие:

- σ^2 — связана с шумом в исходных данных
- *Var(a(**x**))* — характеризует разнообразие алгоритмов
- *bias(f(**x**), a(**x**))* — характеризует способность алгоритмов настраиваться на целевую зависимость

Ошибки разброса и смещения можно визуализировать в виде мишени для игры в дартс (Рис. 3):

- В левом верхнем случае модель попадает точно в цель с малым разбросом (дисперсией) и малым смещением. Пример — модель № 1 из Таблицы 2
- В левом нижнем углу — алгоритм с большой дисперсией и высокой средней точностью. Пример — модель № 2 из Таблицы 2
- В правом верхнем углу — алгоритм с малым разбросом (дисперсией) и

большим смещением. Пример — модель № 3 из Таблицы 2

- В правом нижнем углу — модель с большой дисперсией и большим смещением. Работа очень плохого аналитика данных.

Рис. 3. Визуализация разброса и смещения на примере игры в дартс

Ошибка складывается из первых трех пунктов, упомянутых выше. С увеличением сложности модели и добавлением новых степеней свободы ошибка смещения уменьшается, однако ошибка дисперсии возрастает (см. Рис. 4).

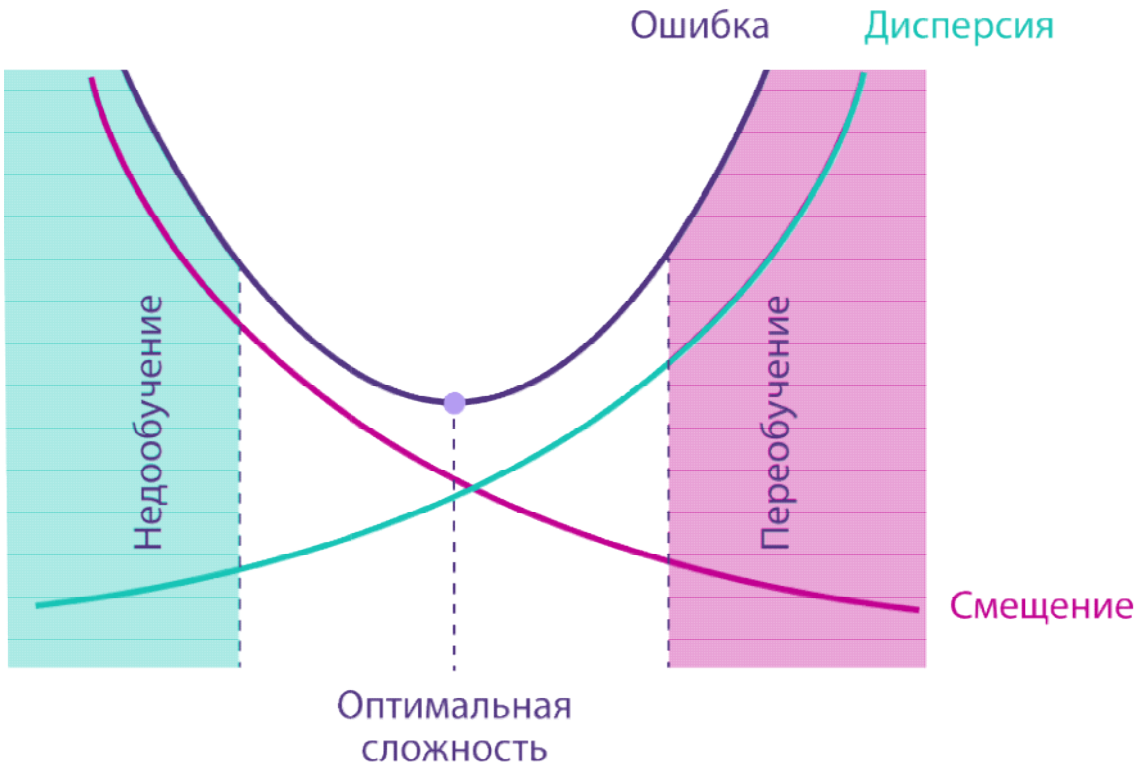


Рис. 4. Классическая иллюстрация изменения разброса и смещения

Компромиссом рассмотренной выше дилеммы о смещении и дисперсии являются *ансамбли моделей*. Они объединяют в себе несколько простых моделей, каждая из которых является, скорее, недообученной. Объединение слабых моделей позволяет добиться высокого качества ансамбля. Подобно тому, как усреднение нескольких наблюдений снижает оценку дисперсии данных, так и увеличение числа простых алгоритмов, обучающихся на некоторых порциях данных из исходной выборки с последующим усреднением полученных предсказаний, помогает снизить дисперсию прогноза. Также ансамбли работают и над снижением смещения

Беггинг, градиентный бустинг, XGBoost, CatBoost

Бэггинг

Бэггинг — усреднение предсказания. В большинстве способов бэггинг (или усреднение) применяется для моделей, построенных на подвыборках основной выборки. Тогда каждая из моделей обучится на своей версии

данных, и усредненное предсказание с большой вероятностью будет отражать общие зависимости в данных.

Пусть обучается *M* алгоритмов, а данные содержат *N* объектов. Тогда для каждого алгоритма из всего множества объектов равновероятно выбираются *N* объектов с возвращением, т. е. в каждом таком подмножестве могут быть повторяющиеся объекты (такой процесс генерации подвыборок с помощью выбора объектов с возвращением называется **бутстрепом (bootstrap)**). Предположим, что ошибки базовых алгоритмов не скоррелированы (на самом деле это предположение неверно, поскольку подвыборки формировались из общего множества объектов), и их дисперсии равны между собой. Тогда бэггинг уменьшает разброс алгоритма в *M* раз. В терминах смещения и дисперсии это означает, что бэггингне изменяет смещение базовых алгоритмов, но уменьшает их дисперсию.

Бэггинг позволяет реализовать OOB-подход (out-of-bagestimation): после обучения для каждого объекта из выборки получается усредненное предсказание тех алгоритмов, в обучающую подвыборку которых данный объект не попал. Получив подобные предсказания, можно посчитать метрику, которая будет оценивать качество финальной модели.

Рассмотрим бэггинг на примере задачи классификации. Пусть исходная выборка $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, где $\boldsymbol{x}_i$ — входные признаки, y_i — целевая переменная для всех объектов $i \in [1, n]$, $a_S(\boldsymbol{x})$ — классификатор, обученный на подвыборке*S*. С помощью бутстрепа формируем *M*подвыборок $S^{*,1}, \ldots, S^{*,M}$ выборки *D*, и пусть каждая подвыборка имеет размер *n*. При формировании выборок выбирается *d* элементов исходной выборки с возвращением, в каждой подвыборке используется подмножество случайно выбранных признаков. Обучаем классификаторы $a_{S^{*,m}}(\boldsymbol{x})$, где $m \in [1, M]$. Тогда итоговое предсказание будет осуществляться следующим образом:

$$\hat{a}(\boldsymbol{x}) = \text{большинство\_голосов}\{[a_{S^{*,m}}(\boldsymbol{x})]_{m=1}^{M}\}.$$

При решении задачи регрессии подход выглядит так же, только ответ определяется не большинством голосов, а усреднением предсказаний всех моделей.

Вотинг

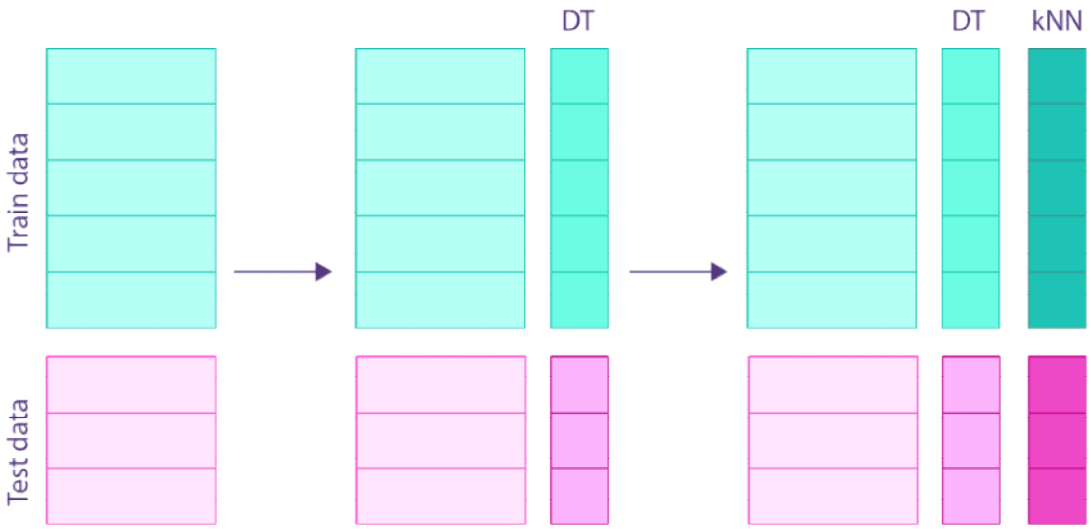
Вотинг — голосование. В некоторых случаях ансамблей на основе голосования голос отдельной модели может иметь больше веса, чем другой. Эти коэффициенты расставляет аналитик по точностным результатам каждой модели. Часто, например, такой подход применяется для использования отдельных моделей, чувствительных к определенному типу данных (например, к выбросам), наряду с основной моделью.

Стекинг

Стекинг — наложение предсказаний. Стекинг моделей можно описать несколькими этапами:

- Обучаем несколько разных моделей
- Производим предсказания для тестовой выборки для каждой модели
- Данное предсказание используем как признак в дополнение к изначальной выборке
- Получившийся датасет используем как новую обучающую выборку

Таким образом, в стекинге моделей происходит обучение на предсказаниях разных моделей, т. е. каждая последующая модель учитывает предыдущие предсказания как новые признаки. Пример стекинга решающего дерева и метода k ближайших соседей представлен на Рис. 1.



Стекинг решающего дерева и метода k ближайших соседей

Бустинг

Бустинг — итеративный стекинг. Один из наиболее эффективных методов машинного обучения в задачах обучения с учителем на численных данных. Такие ансамбли сочетают в себе алгоритмы бэггинга и стекинга, обучая слабые модели на отдельных частях выборки и итеративно обучаясь на своих ошибках. Для моделей бустинга используются, как правило, деревья решений — алгоритмы с низким разбросом, но высоким смещением. Такие модели просты в реализации и менее чувствительны к изменениям данных. Наиболее популярны именно деревья решений (с небольшой глубиной) благодаря своей нелинейности и вычислительной эффективности.

Линейные модели никогда не используются для бустинга, потому что, по сути, агрегирование предсказаний моделей в ансамбле есть их линейная комбинация. Если предсказания получаются путем обучения линейной модели, то это ничем не будет отличаться от использования одной линейной модели, но с другими коэффициентами.

Рассмотрим пример задачи регрессии. Пусть имеется обучающая выборка $\boldsymbol{x} = (x_1, \ldots, x_d)$ и M линейных алгоритмов:

$$a_i(\boldsymbol{x}) = w_1^i x_1 + \ldots + w_d^i x_d, \text{для любого } i \in [1, M].$$

Тогда их взвешенная сумма будет выглядеть следующим образом (ρ_i — коэффициенты, с которыми учитываются базовые модели $a_i(\boldsymbol{x})$):

$$\hat{a}(\boldsymbol{x}) = \sum_{i=1}^{M} \rho_i a_i(\boldsymbol{x}) = \sum_{i=1}^{M} \rho_i (w_1^i x_1 + \ldots + w_d^i x_d) = \sum_{i=1}^{M} (\rho_i w_1^i x_1 + \ldots + \rho_i w_d^i x_d) =$$

$$= \sum_{i=1}^{M} (w'^i_1 x_1 + \ldots + w'^i_d x_d),$$

где $w'^i_j = \rho_i w_j^i$, т. е. мы получили ту же модель линейной регрессии, но с другими коэффициентами, что означает, что смысла делать ансамбль линейных моделей нет (потому что фактически это будет эквивалентно линейной модели с другими коэффициентами).

Наиболее известные алгоритмы бустинг-моделей:

- Адаптивный бустинг

Наиболее известная реализация — AdaBoost (классификация, регрессия). AdaBoost итеративно обновляет веса объектов в выборке и добавляет нового слабого ученика на каждом этапе, чтобы усилить предсказания ансамбля.

Рассмотрим алгоритм адаптивного бустинга для бинарной классификации. Пусть целевая переменная $Y = \{-1, +1\}$, выборка $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_d, y_d)\}$, и мы хотим обучить ансамбль из M моделей. Нашу результирующую модель обозначим $\hat{f}_M(\boldsymbol{x}) = sign(\sum_{m=1}^{M} \rho_m a_m(\boldsymbol{x}))$,

где ρ_m — коэффициенты, с которыми учитываются базовые модели $a_m(\boldsymbol{x})$. Также введем следующие обозначения:

$w_{i,M} = exp(-y_i \sum_{m=1}^{M} \rho_m a_m(\mathbf{x}_i))$ и $\widetilde{w_i}, M = \frac{w_{i,M-1}}{\sum_j w_{j,M-1}}$. Тогда алгоритм адаптивного бустинга будет выглядеть следующим образом:

1. Инициализируем веса $w_{i,1}$, где $i \in [1, d]$: пусть каждый вес принимает значение $\frac{1}{d}$, т.е. $w_{i,1} \leftarrow \frac{1}{d}$

2. Обучим базовых моделей:

a. Пусть a_m $(m \in [1, M])$ – базовая классификационная модель с маленькой ошибкой $N(a_m, \widetilde{w}_m) \leftarrow \sum_{i=1}^{d} \widetilde{w}_{i,m} 1(y_i a_m(\mathbf{x}_i) \leq 0)$. Коэффициенты базовой модели a_m: $\rho_m \leftarrow \frac{1}{2}\log\frac{1-N(a_m,\widetilde{w}_m)}{N(a_m,\widetilde{w}_m)}$

b. Для каждого элемента из выборки, т.е. в цикле по *i* от 1 до *d*, обновим коэффициенты $w_{i,m+1}$ и $\widetilde{w}_{i,m+1}$: $w_{i,m+1} \leftarrow w_{i,m}\exp(-\rho_m y_i a_m(\mathbf{x}_i))$ $\widetilde{w}_{i,m+1} \leftarrow \frac{w_{i,m+1}}{\sum_{j=1}^{d} w_{j,m+1}}$

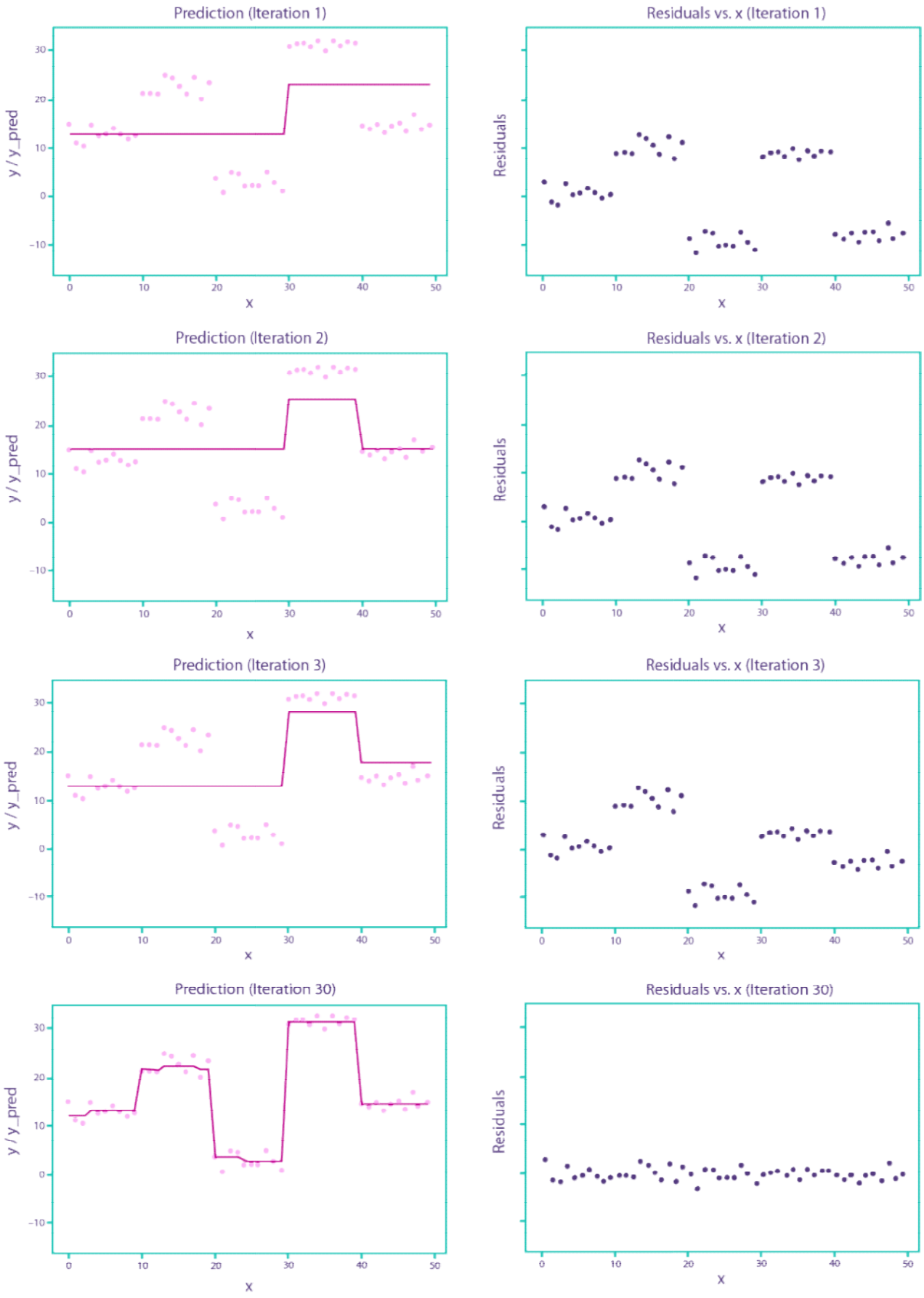
c. Обновим ансамбль моделей: $f_m \leftarrow \sum_{s=1}^{m} \rho_m a_m(\mathbf{x}_i)$

3. Итоговая модель:

$$\hat{f}_M = sign(f_m)$$

- Градиентный бустинг

Сводит задачу поиска оптимальных параметров к задаче градиентного спуска, минимизируя остатки (Рис. 1) предсказаний на каждой итерации. Градиентный спуск — численный метод нахождения локального минимума (или максимума) функции с помощью движения вдоль градиента. Основная идея метода заключается в том, чтобы идти в направлении наискорейшего спуска, а это направление задается антиградиентом.



Алгоритм работы бустинг-ансамблей на трех первых итерациях и на 30-й итерации (слева) оптимизации ошибок предсказаний (справа)

Существует много реализаций градиентного бустинга, каждая из которых обладает своими достоинствами, однако самыми распространенными являются **XGBoost, LightGBM** и **CatBoost**.

Таким образом:

- Вотинг редко используется «в чистом виде», поскольку обычно является частью метода ансамблирования
- Бэггинг уменьшает дисперсию, а стекинг и бустинг, наоборот, уменьшают смещение. Поэтому, если одна какая-то модель переобучается, справиться с этим поможет бэггинг. Если одна модель не обучается, бустинг на основе данной модели или стекинг помогут улучшить качество
- Стекинг:
  - часто используется на соревнованиях, например на kaggle, для извлечения признаков, которые могут использоваться другими ансамблями
  - если для бэггинга и бустинга обычно базовыми моделями являются одинаковые модели, чаще всего решающие деревья, то стекинг будет лучше работать при использовании разных моделей, которые будут как бы дополнять друг друга
  - лучше не использовать на очень маленьких выборках, для них лучше подойдет бэггинг или бустинг

Оценка неопределенности модели с помощью ансамблей

Некоторые модели машинного обучения должны быть более надежными, чем другие. Например, в медицинской диагностике или автономном вождении цена ошибки модели может быть очень большой и измеряться в больших суммах денег или даже человеческих жизнях. У точности моделей есть свои пределы: редко удается построить модель, которая никогда не ошибается. Одним из вариантов работы в таком случае является дополнительная оценка уверенности модели в таком прогнозе. Если модель недостаточно уверенная, она может делегировать принятие окончательного решения человеку.

Обычно говорят об уверенности модели в своем прогнозе и об оценке неопределенности: чем больше уверенность, тем меньше неопределенность, и наоборот. Перейдем к типам неопределенности, которые помогут разобраться, почему модель может быть неуверенной в своих предсказаниях и ошибаться.

Неопределенность бывает двух типов:

- **Алеаторная неопределенность** — неопределенность данных или, другими словами, естественно возникшая неопределенность, которую мы не можем уменьшить из-за того, что она возникает, когда существует несколько почти идентичных комбинаций входных признаков с разными метками классов. На практике редко рассматривается, поскольку ее нельзя уменьшить
- **Эпистемическая неопределенность** — неопределенность модели, которая возникает из-за недостатка знаний и может быть уменьшена за счет получения дополнительных данных. На практике может быть уменьшена путем добавления большего числа параметров в модель, увеличения размера и охвата обучающей выборки и т. д.
- **Полная неопределенность** — сумма неопределенности данных и модели

Примеры алеаторной и эпистемической неопределенности изображены на рисунке ниже:

- Слева — пример алеаторной неопределенности: прогноз в точке знака вопроса алеаторно не определен, поскольку классы пересекаются
- Справа — пример эпистемической неопределенности: в области знака вопроса нам недостаточно знаний о правильной гипотезе расположения разделяющей плоскости классификатора, что вызвано недостатком данных

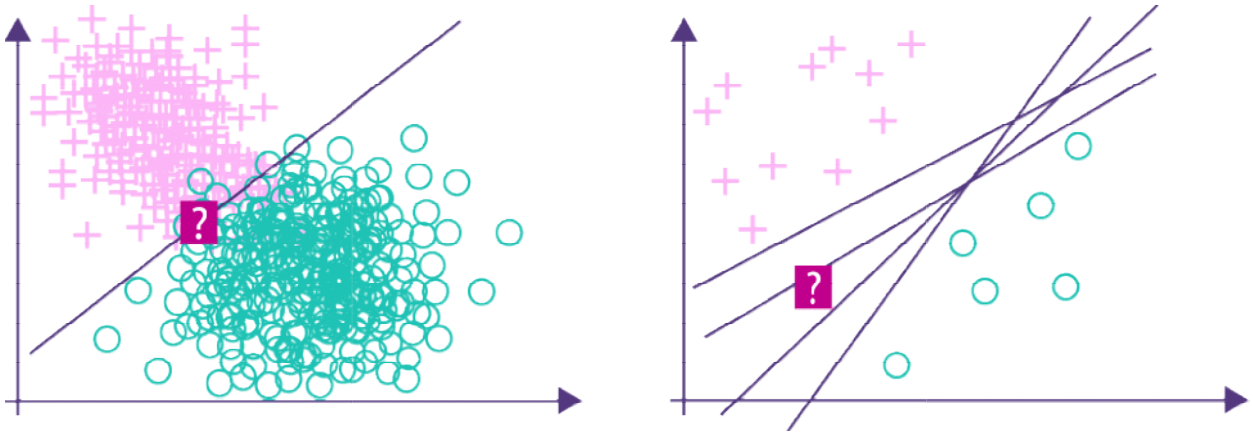


Рис. 1 Пример алеаторной (слева) и эпистемической неопределенностей (справа)

Алеаторная неопределенность (неопределенность данных)

Возникает из-за пересечения классов в задачах классификации и шума в целевой переменной в задачах регрессии. Рассмотрим подробнее алеаторную неопределенность в контексте данных задач.

Задача регрессии

Рассмотрим подробнее байесовскую линейную регрессию, которая, в отличие от обычной линейной регрессии, предсказывает распределение по целевому значению y путем выравнивания распределения по весам модели w. Введем также обозначения входных признаков x и обучающей выборки (X_{train}, y_{train}), где X_{train} — данные, y_{train} — целевая переменная. Распределение по целевой переменной $p(y|x, w)$ — прогнозирующее распределение — может быть получено путем маргинализации по w. Мы берем среднее значение прогнозов бесконечно многих моделей:

$$\underbrace{p(y|\mathbf{x}, \mathbf{X}_{\mathbf{train}}, \mathbf{y}_{\mathbf{train}}, \alpha, \beta)}_{\text{прогнозирующее распределение}} =$$

$$= \int d\mathbf{w} \quad \underbrace{p(y|\mathbf{x}, \mathbf{w}, \beta)}_{\text{распределение по целевой переменной}} \quad \underbrace{p(\mathbf{w}|\mathbf{X}_{\mathbf{train}}, \mathbf{y}_{\mathbf{train}}, \alpha, \beta)}_{\text{распределение параметров}}$$

где α, β — два гиперпараметра. Оба распределения правой части имеют явный вид, и можно получить явную формулу для прогнозного распределения для некоторых моделей.

Рассмотрим распределения из уравнения выше подробнее на примере линейной регрессии.

- Условное распределение по целевой переменной — гауссовское распределение со средним значением, определяемым точечным произведением весов и признаков (как в обычной линейной регрессии), и фиксированной дисперсией, определяемой параметром точности β: $p(y|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(y|\mathbf{x}^T\mathbf{w}, \beta^{-1})$
- Распределение параметров также предполагается нормальным, но со средним значением $\mathbf{m}_n$ и дисперсией $\mathbf{S}_n$: $p(\mathbf{w}|\mathbf{X}_{\mathbf{train}}, \mathbf{y}_{\mathbf{train}}, \alpha, \beta) = \mathcal{N}(\mathbf{w}\,\mathbf{m}_n, \mathbf{S}_n)$

Тогда параметры $\mathbf{m}_n$ и $\mathbf{S}_n$апостериорного распределения будут заданы явными формулами: $\mathbf{m}_n = \beta\mathbf{S}_n{\mathbf{X}_{\mathbf{train}}}^T\mathbf{y}_{\mathbf{train}}$ $\mathbf{S}_n^{-1} = \alpha\mathbf{I} + \beta{\mathbf{X}_{\mathbf{train}}}^T\mathbf{X}_{\mathbf{train}}$

где α — параметр, определяющий точность предыдущего распределения параметров *p(w)*. Само распределение будет нормальным. То есть для каждого вектора параметров мы знаем, насколько он вероятен с учетом нашего исходного ожидания от параметров модели и наблюдений.

Теперь пойдем дальше и получим прогнозирующее распределение уже не для вектора параметров модели линейной регрессии, а для целевой переменной в конкретной точке. Дисперсия этого распределения будет характеризовать неопределенность прогноза модели в точке.

Прогнозирующее распределение — результат нашей модели: для заданного ***x*** она предсказывает распределение вероятностей по целевой переменной ***y***. Поскольку как распределение по параметрам, так и условное распределение по целевой переменной являются гауссовыми, прогнозирующее распределение принимает следующую форму:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{X}_{\mathbf{train}}, \mathbf{y}_{\mathbf{train}}, \alpha, \beta) = \mathcal{N}(y|\mathbf{m}_n^T\mathbf{x}, \sigma_n^2(\mathbf{x}))$$

Среднее значение прогнозируемого распределения задается точечным произведением среднего значения распределения по весам $\mathbf{m}_n$ и признакам

x. Интуитивно понятно, что мы выполняем процедуру, похожую на линейную регрессию, но используя средние веса и игнорируя дисперсию распределения по весам на текущий момент, что учитывается отдельно в дисперсии прогнозируемого распределения:

$$\sigma_n^2(\mathbf{x}) = \underbrace{\beta^{-1}}_{\text{алеаторная неопределенность}} + \underbrace{\mathbf{x}^T \mathbf{S}_n \mathbf{x}}_{\text{эпистемическая неопределенность}}$$

Таким образом, мы получили алеаторную и эпистемическую неопределенности для модели линейной регрессии. Более детальный вывод результата выше можно прочитать в книге(секция 3.3).

Задача классификации

Модели машинного обучения, решающие задачи классификации, как уже упоминалось ранее в курсе, могут работать в двух режимах:

1) predict — модель предсказывает метку класса, т. е. одно число
2) predict_proba — модель предсказывает вероятности принадлежности объекта к каждому из классов, т. е. вектор длины K, где K — количество классов, причем каждая из этих вероятностей лежит в промежутке от 0 до 1, и сумма элементов вектора равна 1 (в общем случае это не всегда так, но вероятности можно откалибровывать таким образом, чтобы это правило выполнялось)

Если модель работает во втором режиме, то вектор вероятностей может характеризовать неопределенность данных. Рассмотрим задачу многоклассовой классификации с K классами, и пусть модель в качестве предсказания для какого-то объекта выдала вектор, каждое число которого очень близко к 1/K . Если пример действительно относился к какому-то среднему классу, (например, если при решении задачи классификации изображений котов и собак попалась картинка котопса, и модель в качестве предсказания выдала (0,5, 0,5)), то мы действительно можем говорить об алеаторной неопределенности. Если, например, модель учится различать картины Ван Гога и Рембрандта, и для картины Рембрандта модель выдала вероятности (0,5, 0,5), тогда это типичная ситуация

эпистемическойнеопределенности, так как ее можно уменьшить путем добавления в обучающую выборку картин Рембрандта. Если выборка не сбалансирована, то алеаторная неопределенность будет выше для класса, который чаще встречается в выборке (из-за большего спектра ситуаций, где он мог встретиться).

Однако общих, более четких рекомендаций по определению неопределенности данных на основе векторов вероятностей, предсказанных моделью, нет. Каждый конкретный случай требует тщательного анализа. Одинаковые, близкие к 1/К вероятности — это, скорее, индикатор, который указывает на необходимость исследования как самого объекта, так и модели.

Эпистемическая неопределенность (неопределенность модели)

Для определения эпистемической неопределенности возьмем ансамбль моделей. Пусть нам доступно апостериорное распределение параметров модели. Тогда неопределенность параметров θ определяется апостериороммодели $p(\theta|\mathcal{D})$: $p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$

В целом модели, выбранные из данного распределения, должны иметь одинаковые предсказания на данных, которые они видели, на которых они обучались, и разные — на данных, которые они не видели. Можно рассматривать следующий ансамбль моделей (решающий задачу классификации): $\{P(y|\mathbf{x};\theta^{(m)})\}_{m=1}^{M}$, где M — количество моделей в ансамбле, $\theta^{(m)} \sim p(\theta|\mathcal{D})$.

Рассмотрим визуальное сравнение неопределенности при разных входных данных. При понятных входных данных для моделей (т. е. при данных, аналогичных обучающей выборке) их предсказания будут одинаковыми, а значит, и неопределенность будет низкой (Рис. 2). Из этого же Рис. 2 можно сделать вывод, что модели уверены при предсказании класса 1. На Рис. 3 тоже показаны данные, которые понятны моделям, однако эпистемическая неопределенность (неопределенность модели) является

очень высокой, поскольку вероятности для всех классов небольшие, модели не уверены ни в одном из них. Если тестовая выборка будет сильно отличаться от обучающей, то предсказания обученных моделей будут отличаться, а значит, и эпистемическая неопределенность будет высокой (Рис. 4).

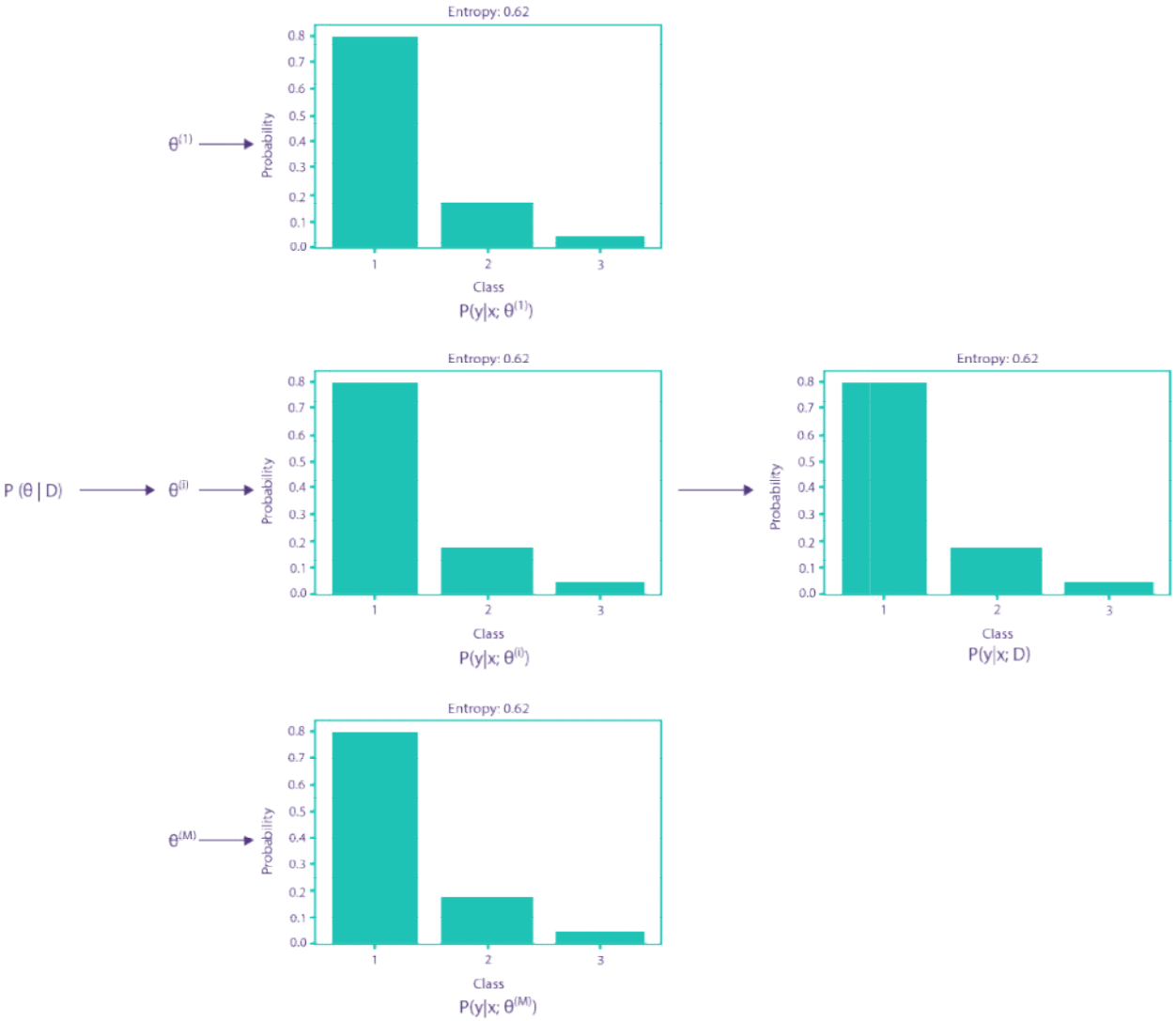


Рис 2. Неопределенность ансамбля для знакомых данных. Низкая алеаторная неопределенность

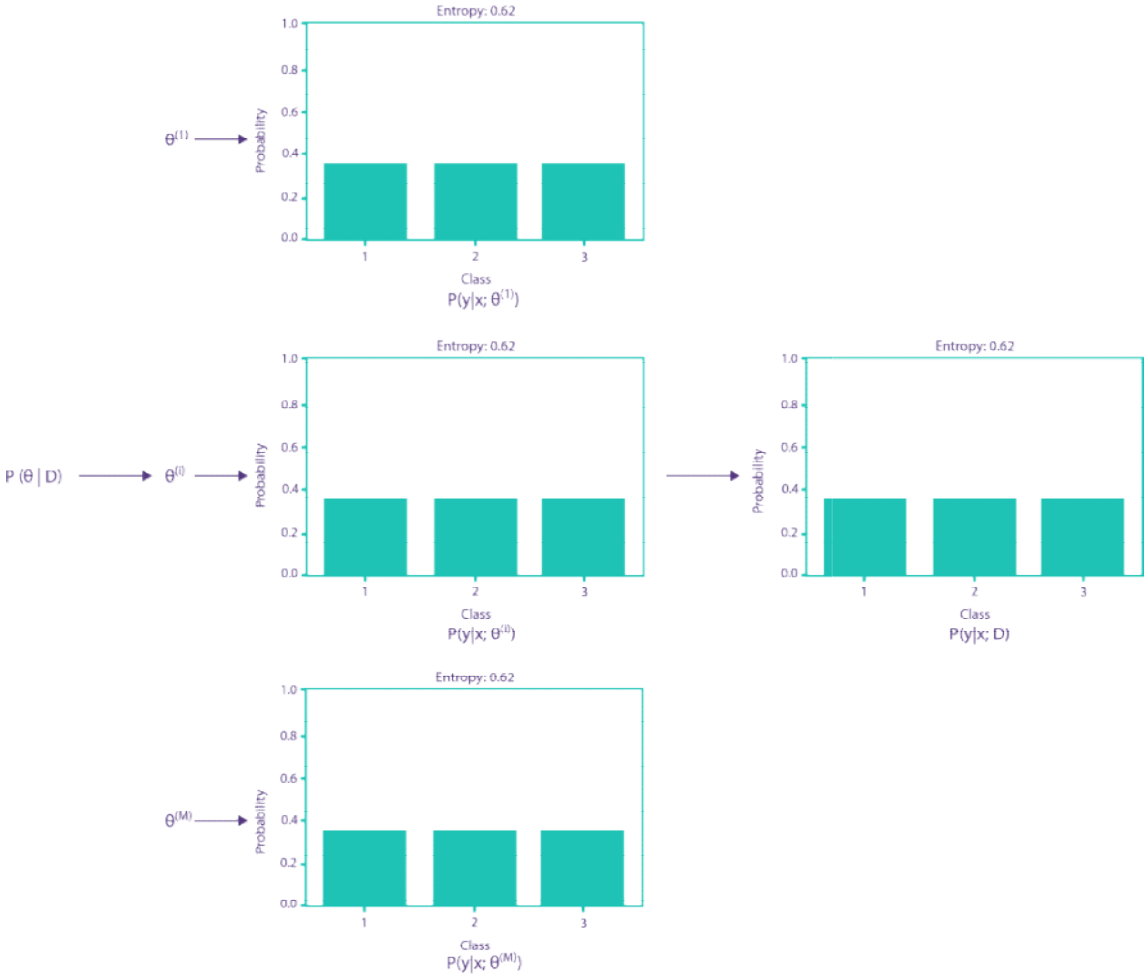


Рис. 3. Неопределенность ансамбля для знакомых данных. Высокая эпистемическая неопределенность и энтропия

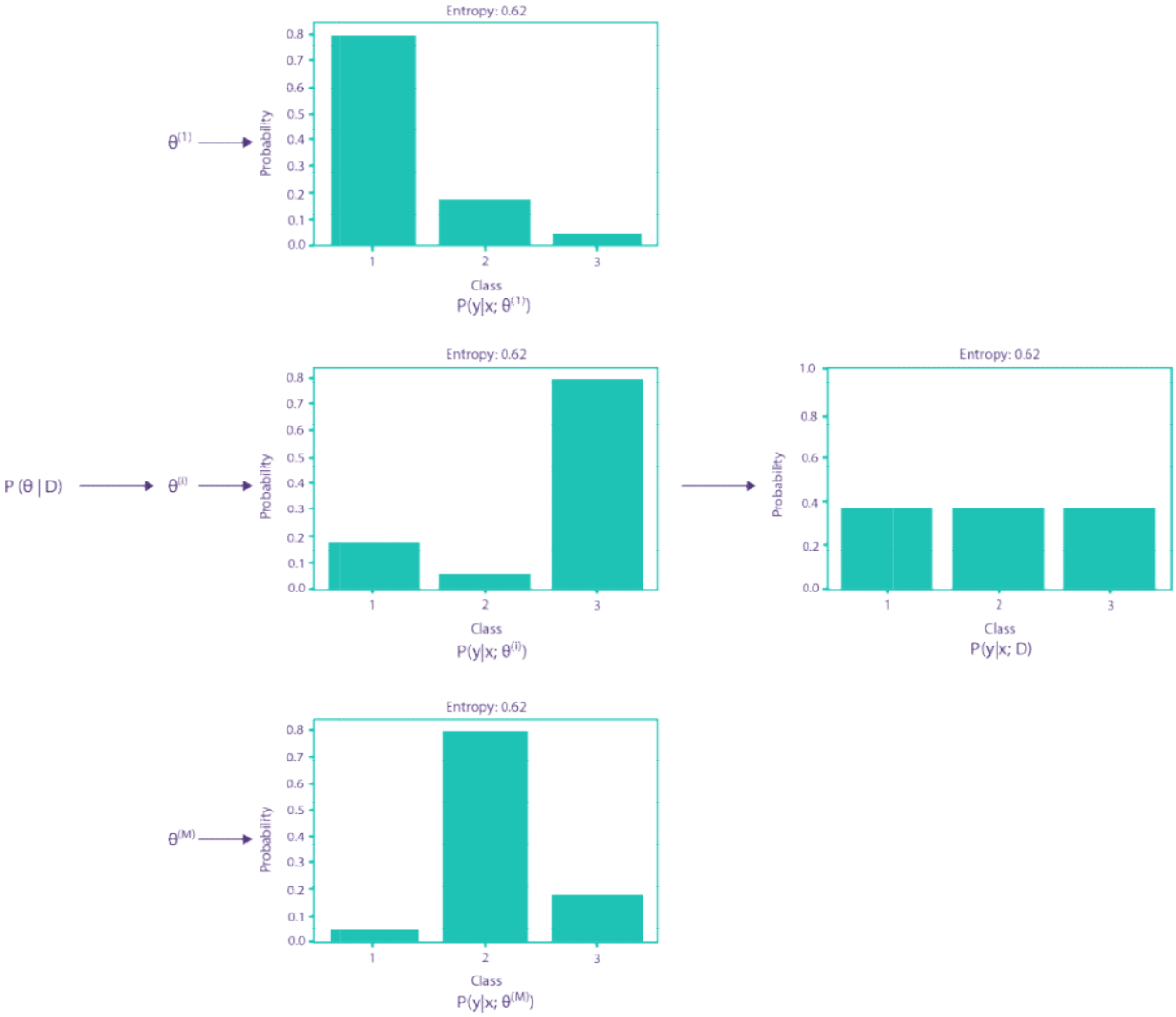


Рис. 4. Неопределенность ансамбля для незнакомых данных

Каждую модель из ансамбля $\{P(y|\mathbf{x};\theta^{(m)})\}_{m=1}^{M}$ можно представить как точку на симплексе (Рис. 5).

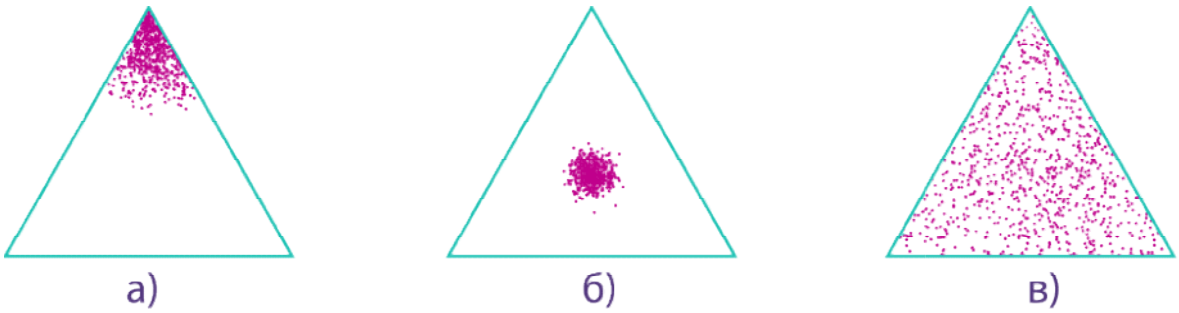


Рис. 5. Визуализация: а) низкой неопределенности; б) неопределенности данных; в) неопределенности модели на симплексе

Мерой эпистемической неопределенности в задаче классификации может служить **взаимная информация (mutualinformation, MI)**. Рассмотрим разницу между $\mathcal{H}[P(y|\mathbf{x};\mathcal{D})]$ и $\mathbb{E}_{p(\theta|\mathcal{D})}[\mathcal{H}[P(y|\mathbf{x};\mathcal{D})]]$:

$$\underbrace{\mathcal{I}[y,\theta|\mathbf{x},\mathcal{D}]}_{\text{эпистемическая неопределенность}} = \underbrace{\mathcal{H}[P(y|\mathbf{x};\mathcal{D})]}_{\text{полная неопределенность}} - \underbrace{\mathbb{E}_{p(\theta|\mathcal{D})}[\mathcal{H}[P(y|\mathbf{x};\mathcal{D})]]}_{\text{ожидаемая неопределенность данных}}$$

- Если предсказания моделей одинаковы, то эпистемическая неопределенность равна нулю:

$$\mathcal{I}[y,\theta|\mathbf{x},\mathcal{D}] = \mathcal{H}[P(y|\mathbf{x};\mathcal{D})] - \mathbb{E}_{p(\theta|\mathcal{D})}[\mathcal{H}[P(y|\mathbf{x};\mathcal{D})]] = 0$$

- Если предсказания моделей различны, то эпистемическая неопределенность больше нуля:

$$\mathcal{I}[y,\theta|\mathbf{x},\mathcal{D}] = \mathcal{H}[P(y|\mathbf{x};\mathcal{D})] - \mathbb{E}_{p(\theta|\mathcal{D})}[\mathcal{H}[P(y|\mathbf{x};\mathcal{D})]] > 0$$

Данная разница и есть **взаимная информация (mutualinformation, MI)**.

Определение неопределенности в задаче регрессии аналогично задаче классификации, однако мы не можем получить точную формулу полной неопределенности, но вместо нее можем использовать **полную дисперсию**. Рассмотрим ансамбль регрессионных моделей:

$$\{p(\mathbf{y}|\mathbf{x};\theta^{(m)})\}_{m=1}^{M} \rightarrow \{\{\mu^{(m)},\sigma^{(m)}\} = f(\mathbf{x};\theta^{(m)})\}_{m=1}^{M}$$

Для этого ансамбля полная дисперсия выглядит следующим образом:

$$\mathbb{V}[y] = \mathbb{V}_{ens}[\mu] + \mathbb{E}_{ens}[\sigma^2] \approx \underbrace{\frac{1}{M}\sum_{m=1}^{M}(\bar{\mu} - \mu^{(m)})^2}_{\text{эпистемическая неопределенность}} + \underbrace{\frac{1}{M}\sum_{m=1}^{M}(\sigma^{(m)})^2}_{\text{алеаторная неопределенность}}$$

Концептуально похоже на взаимную информацию, но неинвариантно к масштабированию.

Трудности при определении неопределенности моделей:

- Требует больших вычислительных ресурсов и объемов памяти. Во-первых, само по себе обучение ансамбля — ресурсоемкий процесс. Кроме того, если ансамбль состоит из M моделей, время предсказания увеличивается в M раз

- Сложно составить ансамбль таким образом, чтобы модели в нем отличались друг от друга

Таким образом, эпистемическая неопределенность показывает уровень «несогласия» моделей друг с другом. Алеаторную неопределенность можно посчитать следующим образом:

- В задачах классификации:
  - обучаем модель с функцией потерь (отрицательное логарифмическое правдоподобие, кросс-энтропия) (negativelog-likelihood (cross-entropy) loss)
  - для каждого примера получаем распределение по меткам классов
  - неопределенность данных — энтропия этих распределений
- В задачах регрессии:
  - делаем предположение о распределении данных
  - оптимизируем функцию потерь (если предполагаем, что данные подчиняются нормальному закону распределения, то функция потерь — отрицательное логарифмическое правдоподобие (negativelog-likelihoodloss)
  - оцениваем среднее и дисперсию в случае нормального распределения, и тогда неопределенность данных — дисперсия данного распределения

Подробнее посмотреть вывод всех формул можно здесь.

Модели градиентного бустинга

Рассмотрим несколько моделей градиентного бустинга:

- Стохастический градиентный бустинг (StochasticGradientBoosting, SGB)

В его основе лежит стохастический градиентный спуск, который отличается от обычного градиентного спуска тем, что градиент оптимизируемой функции считается на каждом шаге не как сумма градиентов от каждого элемента выборки, а как градиент от одного, случайно выбранного элемента.

- Стохастический градиентный бустинг Ланжевена (StochasticGradientLangevinBoosting, SGLB)

Основан на специальной форме уравнения диффузии Ланжевена, специально разработанной для градиентного спуска. Подробнее об особенностях и реализации можно почитать здесь.

Если рассматривать ансамбль нескольких моделей стохастического градиентного бустинга или стохастического градиентного бустинга Ланжевена, то в задачах:

- Регрессии:
  - алеаторная неопределенность — средняя прогнозируемая дисперсия
  - эпистемическая неопределенность — дисперсия прогнозируемых средних значений
- Классификации:
  - алеаторная неопределенность — средняя энтропия по моделям
  - полная неопределенность — энтропия среднего прогноза

В библиотеке CatBoost содержатся следующие инструменты для оценки неопределенности:

- **Функция потерь RMSEWithUncertainty** — предсказывает среднее значение и дисперсию для задач регрессии
- **Опция PosteriorSampling** — включает режим SGLB и устанавливает параметры обучения, гарантирующие апостериорное сэмплирование
- **Тип предсказаний VirtEnsembles** — возвращает предсказания виртуального ансамбля для любой заданной модели

Нейронные сети

Для нейронных сетей можно действовать похожим образом. Наши подходы к оценке требовали лишь оценки вероятности и ее разброса. Поэтому если у нас есть ансамбль нейронных сетей — задача решена.

Для создания ансамбля мы можем обучить с нуля несколько нейронных

сетей. В результате мы получим набор нейросетей, обученных на одинаковых данных, но с разными параметрами. Дисперсия предсказаний сетей в ансамбле и будет оценкой неопределенности.

Существуют более специфические способы, предназначенные именно для нейронных сетей. С помощью метода вариационного дропаута (variationaldropout) мы можем получить несколько разных предсказаний для одной модели! Данная техника дает возможность получать большее количество предсказаний, чем для ансамбля, хотя в случае вариационного дропаута предсказания могут быть сильно скоррелированы.