ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

Тема занятия: Решающие деревья.

Аннотация к занятию: обучающиеся познакомятся с важным классом алгоритмов машинного обучения — решающими деревьями. Узнают, что такое логические алгоритмы анализа данных. Изучат, как из простых решающих правил составлять решающие деревья. Обсудят, как обучить решающие деревья для задач классификации и регрессии.

Цель занятия: сформировать у обучающихся представление об устройстве и обучении решающего дерева, показать преимущества и недостатки алгоритма для задач классификации и регрессии. Познакомить с выбором решающего правила в решающем дереве.

Задачи занятия:

- познакомить обучающихся с классом алгоритмов машинного обучения — решающими деревьями;
- сформировать понятия «решающее дерево», «решающее правило»;
- проанализировать решающие деревья с точки зрения недообучения и переобучения;
- познакомить с преимуществами и недостатками алгоритма для задач классификации и регрессии.

Ход занятия

Этап занятия	Время	Деятельность педагога	Комментарии, рекомендации для педагогов
Организационный этап	5 мин.	Добрый день! Давайте подумаем, как мы с вами принимаем реальные решения в жизни. Чем мы руководствуемся? **Возможные ответы обучающихся:** ● правилами; ● рассуждениями; ● алгоритмами. Наши рассуждения редко напоминают линейные алгоритмы. Правила, которыми мы руководствуемся, зачастую более простые. Приведём примеры таких решающих правил, сформулированных на простом языке. Предположим, что вы сотрудник банка, который принимает решение о выдаче кредита. Кредитный скоринг, то есть определение, стоит ли выдать клиенту кредит, — это классическая задача классификации. В роли признаков здесь могут выступать разные характеристики клиента, а в качестве обучающей выборки — данные о том, вернули ли кредит предыдущие клиенты.	Приветствие. Создание в классе атмосферы психологического комфорта.

		Как может выглядеть алгоритм принятия решения о выдаче кредита? Например, так: «Если в анкете клиента указан телефон, зарплата клиента превышает $1000, а размер кредита не превышает $3000, то кредит выдать». «Если клиент уже имеет просроченные кредиты, кредит не выдавать», и так далее. Другой пример: предположим, что вы врач, принимающий решение о проведении операции на сердце. Пример решающего правила для вас может быть такой: если возраст пациента по крайней мере 60 лет и пациент перенёс инфаркт, то операцию не делать. Итак, основная идея алгоритма решающего дерева заключается в том, чтобы каким-то образом смоделировать реальный процесс принятия решений человеком	
Постановка цели и задач занятия. Мотивация учебной деятельности обучающихся	7 мин.	Какая сегодня тема урока? **Возможные ответы обучающихся:** ● алгоритмы; ● решающие правила. Тема занятия — «Решающие деревья». Как вы думаете, какие задачи мы сможем сегодня решить? **Возможные ответы обучающихся:** ● узнать, что такое решающие деревья;	Способствовать обсуждению мотивационных вопросов

		• как создать решающие деревья. **Мы узнаем:** • устройство и способы обучения решающего дерева; • преимущества и недостатки решающих деревьев.	
Изучение нового материала	50 мин.	Вернёмся к алгоритмам. А какие алгоритмы мы уже знаем? **Ответы обучающихся** Мы с вами уже изучили несколько алгоритмов машинного обучения: метод ближайших соседей и линейные алгоритмы. Алгоритм ближайших соседей очень редко используется на практике сразу по ряду причин: во-первых, он склонен к переобучению, а во-вторых, для его применения в памяти необходимо хранить всю обучающую выборку, чего почти никогда нельзя себе позволить. Наконец, он довольно медленно работает и, как мы с вами обсуждали, неустойчив к масштабу признаков. Напротив, линейные алгоритмы часто применяются на практике благодаря своей простоте и устойчивости. Но в некоторых задачах нижележащие закономерности в данных, которые мы хотим восстановить, решив, например, задачу регрессии, оказываются слишком сложными для линейных алгоритмов. В самом деле, линейные алгоритмы по самой своей структуре не могут восстанавливать произвольные закономерности	Для справки: Сайт: https://ml-handbook.ru/chapters/decision_tree/intro Перед уроком рекомендуется ознакомиться с материалами, представленными на сайте.

		в данных: с этим мы с вами сталкивались, когда изучали линейную и логистическую регрессию. Возникает желание использовать такие решающие правила для создания алгоритма машинного обучения. Так вот, алгоритм, использующий логические закономерности в данных, вроде тех, что мы с вами рассмотрели, называется логическим алгоритмом. Самый простой способ организовать логические правила в алгоритм — это построить решающее дерево. Решающее дерево очень похоже на блок-схему алгоритма. По сути, оно и является блок-схемой. В каждой вершине решающего дерева находится некоторый вопрос, а рёбра из вершины соответствуют ответам на этот вопрос: да или нет. В листах этого дерева, то есть в конечных или, как говорят, терминальных вершинах, записаны ответы алгоритма. Если мы говорим про задачу классификации, то в листах записаны классы, к которым мы относим объект. Кстати, на этом занятии мы будем говорить исключительно о решающих деревьях для задачи классификации. Деревья регрессии мы обсудим отдельно. Перед вами пример решающего дерева для решения задачи классификации клиентов на два класса: добросовестные и недобросовестные заёмщики. При поступлении на вход объекта — в данном случае, клиента — мы стартуем из корня и начинаем задавать ему соответствующие вопросы. В данном примере мы спрашиваем у клиента, верно ли, что его возраст	

		превосходит 40 лет. Если это верно, мы идём в правое поддерево, если нет — в левое. И так мы продолжаем отвечать на вопросы и спускаться в самый низ дерева. В конце концов мы попадаем в один из листов, в котором стоит метка класса — выдать кредит или отказать в выдаче. Для простоты давайте считать, что признаки объектов являются действительными числами. Мы также ограничимся лишь вопросами вида «Верно ли, что тот или иной признак больше или равен некоторому фиксированному числу z?». Например, верно ли, что зарплата клиента больше или равна $2000? Оказывается, таких простых вопросов с ответами да/нет вполне хватает для построения эффективных решающих деревьев. А вот вопросы типа «Какое у вас образование?» с тремя возможными ответами — высшее, среднее, специальное — в реальных решающих деревьях не встречаются. Отлично, мы с вами разобрались, что такое решающее дерево. Как и всегда, основная сложность заключается в том, как искать такое решающее дерево для конкретной обучающей выборки. Нужно научиться обучать решающие деревья. Я опишу алгоритм обучения решающего дерева в общих чертах, а затем расскажу о том, как написать точные формулы для обучения. Итак, давайте считать, что мы решаем задачу классификации заёмщиков на хороших — это класс 1, и	

		плохих — это класс 0. В нашем распоряжении есть обучающая выборка — информация о старых заёмщиках: их признаки, то есть персональные данные, и целевая переменная — вернули ли они кредит. Строить наше решающее дерево мы будем поэтапно. На каждом шаге алгоритма мы будем брать некоторый лист дерева и искать решающее правило, которое гарантирует наилучшее разбиение обучающей выборки в этой вершине. Давайте посмотрим, как это работает на практике. Итак, мы видим, что у нашего дерева уже построено пять вершин, из которых три являются листами. Мы находимся в красном листе и хотим выбрать для него решающее правило. Посмотрим на обучающую выборку. Прогоним её через текущее дерево и в каждой вершине запишем, сколько объектов первого класса и сколько объектов нулевого класса дошло до этой вершины. Скажем, всего в обучающей выборке у нас было 100 объектов первого класса и 80 объектов нулевого класса. Решающее правило в корневой вершине разделило их на две группы: в левое поддерево ушло 70 объектов первого класса и 20 объектов нулевого класса. В правое поддерево ушло 30 объектов первого класса и 60 объектов нулевого класса. Вершина слева тоже разбивается на две вершины и так далее. Итак, в красную вершину, которую мы хотим сейчас подразбить, попала часть объектов обучающей	

		выборки, скажем, 9 объектов первого класса и 11 нулевого. Напомню, что наша цель — это выбрать в данной вершине решающее правило, то есть некоторый признак X, а также порог, по которому мы будем делить объекты обучающей выборки на две части — в левое поддерево и в правое поддерево. Ну, например, пусть X — это суммарное количество просрочек по всем предыдущим кредитам. Ясно, что чем больше X, тем хуже. Так вот, мы хотим выбрать некоторый порог на количество просрочек. В данном случае порог равен 12: если просрочек больше 12, идём в правое поддерево, если меньше или равно — в левое поддерево. Давайте подумаем, какие условия необходимы для решающего правила, чтобы оно было хорошим. Было бы идеально, если бы существовало такое решающее правило, чтобы в левую вершину уходили все 9 объектов первого класса, а в правую — 11 объектов нулевого класса. Тогда разбиение окажется окончательным и потомков красной вершины дальше можно не разбивать: все объекты, попадающие влево, мы будем относить к первому классу, а попадающие вправо — к нулевому. В терминах нашей задачи кредитного скоринга, для нас было бы идеально, если бы существовало решающее правило, которое позволяет сразу в данной вершине отфильтровать 11 «хороших» заёмщиков от 9 «плохих»: хорошие попали бы в одно поддерево, а плохие — в другое.	Для справки: Ирисы Фишера — это набор данных для задачи классификации, на примере которого Рональд Фишер в 1936

		Такая идеальная ситуация, скорее всего, не произойдёт. Тем не менее, хочется подобрать правило так, чтобы по максимуму разнести объекты двух классов в разные поддеревья. Например, для данного признака X — количества просрочек по кредитам — мы нашли правило X меньше или равно 12, поставляющее такое разбиение: 8 на 5 в левой вершине и 1 на 6 в правой. Нас устраивает это правило, и мы двигаемся дальше. Что же мы делаем дальше? Повторяем процесс до тех пор, пока все листы не станут терминальными. Конечно, к алгоритму, который мы описали, возникает масса вопросов. Во-первых, по какому критерию выбирать наилучшее решающее правило? Во-вторых, как понимать, что какую-то вершину можно сделать терминальной и дальше не дробиться? Наконец, как технически перебрать все доступные решающие правила и выбрать оптимальное? Итак, мы в общих чертах описали алгоритм обучения решающего дерева для задачи классификации. Теперь давайте посмотрим, как решающие деревья применяются на практике. Делать мы это будем на известнейшем датасете под названием «Ирисы Фишера». Этот датасет был собран известным английским математиком Рональдом Фишером в первой половине XX века. Датасет содержит данные о 150 экземплярах цветов ириса, по 50 экземпляров трёх видов: Iris setosa, Iris virginica и Iris versicolor. Каждый	году продемонстрировал работу разработанного им метода дискриминантного анализа. Иногда его также называют ирисами Андерсона, так как данные были собраны американским ботаником Эдгаром Андерсоном. Этот набор данных стал уже классическим и часто используется в литературе для иллюстрации работы различных статистических алгоритмов

		экземпляр имеет 4 характеристики: длина наружной доли околоцветника, ширина наружной доли околоцветника, длина лепестка, ширина лепестка. В данном случае для классификации достаточно лишь двух признаков: длины и ширины лепестка. Визуализируем выборку, отложив по оси x ширину лепестка, а по оси y — длину. Точки зелёного, жёлтого и красного цветов соответствуют объектам трёх классов. Построим для этой задачи решающее дерево. Оказывается, чтобы построить хорошее решающее дерево только на этих двух признаках, достаточно глубины 3. Решающее дерево вы видите на картинке, в вершинах используется всего два признака. Заметим, что наша плоскость разбилась горизонтальными и вертикальными линиями на три участка: зелёного, жёлтого и красного цветов. Это разбиение соответствует нашему решающему дереву: если точка плоскости окрашена, например, в зелёный цвет, это означает, что если пропустить объект, соответствующий этой точке, через решающее дерево, то он попадёт в зелёную вершину. Посмотрим на корневую вершину. В ней мы спрашиваем, верно ли, что Petal Length — PL, длина лепестка — меньше 2,5. Если меньше, то сразу приходим в терминальную вершину, окрашенную красным цветом. Смотрите: на картинке появилась горизонтальная полоса красного цвета, причём эта полоса отделяется от всех остальных точек плоскости прямой, на которой Petal Length равно 2,5. И так далее. В конце концов плоскость разбивается на	

		прямоугольные области, стороны которых параллельны осям координат. Мы видим, что решающее дерево получилось весьма качественным: на двух классах из трёх ошибок нет вообще, а на третьем мы получили всего три ошибки. Мы с вами в общих чертах описали алгоритм построения решающего дерева. Напомню, мы строим дерево, по очереди разбивая вершины на две части с помощью некоторого решающего правила. Наша цель при выборе решающего правила заключается в том, чтобы объекты из обучающей выборки, дошедшие до красной вершины, хорошо разделялись по классам. Так мы делаем до тех пор, пока все листы не станут терминальными вершинами, то есть пока во всех листах не будет преобладания того или иного класса. Напомню, пусть в красной вершине 9 объектов первого класса и 11 объектов нулевого класса. Давайте через Q обозначим суммарное количество объектов в вершине, в которой происходит ветвление, а через p малое с индексом Q — долю объектов первого класса в этой вершине. В данном случае Q = 20, а p_Q равно 9/20. Мы выбрали некоторое решающее правило, например, правило X <= 12, и видим, что в левое поддерево ушло 8 объектов первого класса и 5 объектов нулевого класса, а в правое — 1 первого и 6 нулевого. Давайте через L обозначим количество объектов в левом поддереве, а через p_L — долю объектов нулевого класса в левом поддереве. В данном случае L равно 13, а p_L равно	

		8/13. По аналогии определяются R, равное 7, и p_R, равное 1/7. Итак, мы хотим записать некоторый критерий ветвления, то есть числовую характеристику, которую необходимо максимизировать для поиска оптимального решающего правила. Давайте посмотрим на значения p_L и p_R. Напомню, хорошее решающее правило, как мы с вами решили, это такое правило, по итогам которого разбиение будет «чистым», то есть в одной вершине будет ярко выражено преимущество первого класса, а в другой — нулевого. Что же означает, что в левой вершине есть преимущество одного из классов? Это означает, что, внимание, p_L близко либо к 0, либо к 1. То же самое можно сказать и про p_R. Наша цель — подобрать решающее правило так, чтобы p_L и p_R были близки к концам отрезка от нуля до единицы. Давайте запишем функцию, которую мы хотим минимизировать в данной вершине. Пусть H(x) — это некоторая функция, которая на краях отрезка [0,1] принимает значение 0, а чем ближе значение x к середине отрезка, тем большее значение принимает H. Примеры подходящих функций H я нарисовал на картинке. Нам необходимо найти решающее правило, которое минимизирует выражение L / Q * H(p_L) + R / Q * H(p_R). Давайте разберёмся, что здесь написано и почему эта задача минимизации имеет смысл. Во-	

		первых, чем меньше значение H(p), тем лучше правило фильтрует объекты в правой и левой вершинах. Во-вторых, разумеется, чем больше объектов попало в вершину, тем более значимо нам соответствующее слагаемое в сумме. Поэтому мы домножаем H(p_L) на L/Q, а H(p_R) — на R/Q. Итак, задача минимизации написана, теперь нужно понять, какую конкретно функцию H выбрать. Необходимые условия для неё мы написали. На самом деле, почти любая функция, удовлетворяющая условиям, которые мы написали, подошла бы. На практике обычно используются две функции. Первая функция — это так называемая энтропия, которая записывается формулой H(q) = −q log q − (1−q) log(1−q). Логарифм здесь обычно берётся по основанию два, чтобы в точке q=½ значение энтропии было равно 1. Пожалуйста, проверьте самостоятельно, что, действительно, H(1/2) = 1, а H(0) = H(1) = 0. Также используют функцию H(q) = 4q(1−q), которая называется индексом Джини. Для неё тоже можно проверить, что H(1/2) = 1, а на концах отрезка H = 0. Более того, в принципе, индекс Джини очень похож на энтропию, в этом можно убедиться на графике. Итак, дорогие друзья, давайте резюмируем. Мы записали выражение, которое необходимо минимизировать, чтобы найти оптимальное решающее правило в данной вершине. Теперь, чтобы искать оптимальное решающее правило, нам необходимо	

		будет перебрать все возможные признаки объектов и все возможные пороги для разбиения. Таким образом, мы с вами полностью описали процесс выбора решающего правила в вершине. Собственно, этот процесс мы всегда будем использовать при построении решающего дерева. Итак, дорогие друзья, теперь давайте более глубоко поговорим о том, какие свойства есть у уже построенных решающих деревьев. В частности, нам важно проанализировать решающие деревья с точки зрения недообучения и переобучения. Напомню, у каждой модели машинного обучения есть гиперпараметры. Например, у линейных алгоритмов гиперпараметры — это параметры регуляризации. Для алгоритма k ближайших соседей — это количество соседей. У решающего дерева также есть гиперпараметры. Мы уже выяснили с вами, что выбор функции H для выбора решающего правила — это один из гиперпараметров решающего дерева. Но, безусловно, у решающих деревьев есть и другие гиперпараметры. Нам необходимо ответить на вопрос, какие гиперпараметры решающих деревьев влияют на различные свойства этих самых решающих деревьев. Для этого анализа нам необходима задача. Вообще говоря, мы пока рассмотрели только одно применение решающего дерева: к задаче ирисов Фишера. Проблема в том, что эта задача была очень лёгкая. В примере с ирисами Фишера наши три класса были почти линейно	Для справки: Регрессия — способ выбрать из семейства функций ту, которая минимизирует функцию потерь. Последняя характеризует, насколько сильно пробная функция отклоняется от значений в заданных точках. Если точки получены в эксперименте, они неизбежно содержат ошибку измерений, шум, поэтому разумнее требовать, чтобы функция передавала общую тенденцию, а не точно проходила через все точки. В каком-то смысле регрессия — это «интерполирующая аппроксимация»: мы хотим провести кривую как

		разделимыми. Давайте посмотрим, как решающие деревья работают на заданиях посложнее. Перед нами датасет «Игрушка дьявола». Он основан на коде Стенфордского курса по нейронным сетям. Закономерность в данных понятна: точки каждого класса образуют закрученную спираль в сторону центра картинки. Теперь давайте построим несколько решающих деревьев для решения этой задачи. Как мы будем это делать? Мы введём гиперпараметр «максимально возможная глубина дерева» и будем строить деревья с различными значениями этого гиперпараметра. Когда мы будем строить решающее дерево для решения этой задачи, в зависимости от максимальной возможной глубины дерева у нас будут получаться решающие деревья с такими классами. Вот такие классы для глубины 3 (пауза), для глубины 7 (пауза) и для глубины 12. Для глубины 3 разделяющие поверхности получились слишком неточными: они лишь поверхностно улавливают спиральную структуру точек и допускают множество ошибок. Для глубины 7 и 12 спиральная структура классов заметна, но сами классы получились очень неестественными: есть множество вкраплений одного класса внутри другого там, где они вряд ли должны быть. Всё это происходит из-за того, что алгоритм переобучается, то есть подстраивается не под общую закономерность в данных, а реагирует на небольшие погрешности и выбросы.	можно ближе к точкам и при этом сохранить её максимально простой, чтобы уловить общую тенденцию. За баланс между этими противоречивыми желаниями как-раз отвечает функция потерь (в английской литературе «loss function» или «cost function»)

		Итак, мы с вами видим, что решающее дерево при большой возможной глубине является нестабильным алгоритмом, склонным к переобучению: небольшое изменение в исходных данных может повлечь за собой полную перестройку дерева. Такого эффекта мы не наблюдали, например, в логистической регрессии. Вместе с тем, в решающем дереве скрыт огромный потенциал в силу вариативности возможных решающих правил. Давайте теперь поговорим о других гиперпараметрах решающего дерева. Мы с вами в целом поняли, что чем больше глубина получившегося дерева, тем больше это будет влиять на переобучение модели. Есть и другие способы контролировать глубину решающего дерева. Это, например, минимальное количество объектов в листе. В самом деле, мы можем при ветвлении запретить выбирать решающие правила, которые будут дробить обучающую выборку слишком мелко, оставляя в одной из вершин слишком мало объектов. Это довольно разумно, потому что если в вершине остаётся очень мало объектов, то мы не можем с надёжностью сказать, что дробить эту вершину дальше имеет смысл. В самом деле, некоторые из этих объектов могли оказаться шумными и попасть в эту вершину случайно, а не вследствие каких-то скрытых закономерностей в данных. Обратите внимание, что чем больше минимальное количество объектов в листе, тем меньше высота получающегося дерева. Ещё один гиперпараметр, который служит похожую роль, это максимальное количество листьев в дереве.	

		Этот параметр означает, что при достижении деревом слишком большого количества листьев, мы просто перестаём дальше дробить наше решающее дерево. Соответственно, чем больше этот параметр, тем больше глубина и, следовательно, тем больше решающее дерево склонно к переобучению. Итак, мы с вами поняли, что у решающего дерева большое количество гиперпараметров. Мы с вами перечислили лишь основные из них. Некоторые гиперпараметры могут сильно влиять на качество получившегося алгоритма, другие — не так сильно. Одно можно сказать определённо: оптимальные гиперпараметры будут очень сильно зависеть от конкретной задачи, которую вы решаете. Таким образом, ответить на вопрос, а какие же гиперпараметры выбирать для обучения решающего дерева, никто вам дать не может. Единственное, что вы можете делать — это подбирать оптимальные параметры, измеряя качество алгоритма на заранее отложенной валидационной выборке. Кстати, очень важно не пытаться определять лучшую модель по обучающей выборке. В самом деле, в таком случае наилучшим деревом всегда будет дерево с наибольшей глубиной, а мы с вами, дорогие друзья, только что убедились, что это не так. **Вопрос для обсуждения** Как вы думаете, какие преимущества и недостатки есть у алгоритма для задачи классификации? **Ответы обучающихся**	

		Начнём с преимуществ. Во-первых, как мы с вами выяснили, решающее дерево способно восстанавливать закономерность любой сложности с любой наперёд заданной точностью. В самом деле, несложно показать, что с помощью решающего дерева можно приблизить любое разбиение пространства на части. Таким свойством не обладают линейные алгоритмы, что выгодно отличает решающие деревья. Далее, решающие деревья довольно просты в понимании и построении. Конечно, ничего не бывает проще, чем knn или линейная регрессия, но и ничего жутко сложного в решающих деревьях нет. Третье свойство, о котором мы с вами не говорили, это автоматическая возможность обрабатывать пропуски в данных. В практических заданиях мы с вами видели, что в данных могут возникать пропуски: некоторые признаки некоторых объектов неизвестны. Решающее дерево крайне естественно обрабатывает пропущенные значения. Если в вершине нам нужно знать значение признака, которого мы не знаем, мы можем просто отправить наш объект в любое поддерево из двух. Ну, например, в то, в которое на стадии обучения попало большинство объектов. Из недостатков обучающего дерева мы выделим два. Во-первых, решающее дерево склонно к переобучению, особенно с ростом глубины. Так как с ростом глубины растёт и сложность восстанавливаемых закономерностей, получается, что на сложных задачах решающее дерево сильно переобучится. Во-вторых, решающее дерево нестабильно и сильно зависит от гиперпараметров модели и от обучающей выборки. Это	Для справки: Задача классификации — это задача присвоения меток объектам. Например, если объекты — это фотографии, то метками может быть содержание фотографий: содержит ли изображение пешехода или нет, изображен ли мужчина или женщина, какой породы собака изображена на фотографии. Обычно есть набор взаимоисключающих меток и сборник объектов, для которых эти метки известны. Имея такую коллекцию данных, необходимо автоматически расставлять метки на произвольных объектах того же типа, что были в изначальной коллекции.

		мы с вами видели на примере датасета игрушка дьявола. Чтобы по максимуму использовать достоинства этого алгоритма, а также нивелировать недостатки, учёные придумали строить композиции решающих деревьев, но об этом мы с вами поговорим в следующих модулях. А сейчас мы очень кратко обсудим решающие деревья для задачи регрессии. Немного скажем о том, как выглядит решающее дерево для задачи регрессии. В целом оно устроено так же, только в листовых вершинах дерева стоят не метки классов, а действительные числа — ответы алгоритма на данных объектах. При поступлении на вход объекта мы прогоняем его по дереву в соответствии с решающими правилами в вершинах, которые устроены так же, как и для задачи классификации. Когда объект приходит в листовую вершину, алгоритм предсказывает то число, которое находится в этой листовой вершине. Получается, что в задаче регрессии количество ответов решающего дерева ограничено количеством листовых вершин. О том, как обучать решающее дерево для регрессии, мы с вами говорить не будем. Вместо этого приведём пример работы решающего дерева для простой задачи одномерной регрессии. Чёрным отмечены экспериментальные точки: по оси x находится единственный признак, по оси y — целевая переменная, таргет. Обратите внимание, что истинная закономерность в данных здесь — это просто закон	

		синуса: видите, точки в целом ложатся на синусоиду. Но есть небольшое количество случайных выбросов. Соответственно, здесь построено два решающих дерева: глубиной 2 и глубиной 5. Зелёным отмечены ответы для глубины 2, а красным — для глубины 5. Как видите, зелёное дерево вообще ничего не поняло про нашу закономерность: ему просто не хватило глубины, чтобы хоть что-то разумное выучить. Красная кривая, хоть и больше смахивает на синус, имеет большое количество выбросов. Заметно, что дерево переобучилось. **Вопрос для обсуждения** Как вы думаете, какие преимущества и недостатки есть у алгоритма для задачи регрессии? **Ответы обучающихся** Во-первых, это все преимущества и недостатки дерева для классификации. При этом дерево для регрессии ещё более склонно к переобучению, особенно учитывая, что для регрессии, как мы с вами убедились, нужна большая глубина, чем для классификации. Всё это делает решающее дерево само по себе довольно слабым алгоритмом для регрессии. Впрочем, его тоже можно использовать как кирпичик для построения композиции алгоритмов, но об этом мы будем говорить в следующем модуле.	

Закрепление изученного материала	15 мин.	**Вопросы для обсуждения** ● Что такое решающее дерево? ● Как обучить решающее дерево? ● Какие преимущества и недостатки есть у алгоритма для задач классификации и регрессии?	Педагог организует беседу по вопросам
Этап подведения итогов занятия (рефлексия)	8 мин.	**Вопросы для обсуждения** ● Чему я научился? ● С какими трудностями я столкнулся? ● Каких знаний мне не хватает для более глубокого понимания изученного материала? ● Достиг ли я поставленных целей и задач?	Педагог способствует размышлению обучающихся над вопросами
Информация о домашнем задании, инструктаж по его применению	5 мин.	Дома повторите основные определения, приведите примеры использования нейронных сетей из повседневной жизни	

Рекомендуемые ресурсы для дополнительного изучения:

1. Решающие деревья. [Электронный ресурс] – Режим доступа: https://ml-handbook.ru/chapters/decision_tree/intro.
2. Модели классификации для нескольких классов. [Электронный ресурс] – Режим доступа: https://ranalytics.github.io/data-mining/071-Multicla...
3. Машинное обучение: от Ирисов до Телекома. [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/billing/blog/334738/.
4. Вероятностная интерпретация классических моделей машинного обучения. [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/343800/.
5. Основы линейной регрессии. [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/514818/.