ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

Тема занятия: Введение в машинное обучение

Аннотация к занятию: обучающиеся продолжат знакомиться с машинным обучением. Обсудят проблему переобучения и то, как измерять качество модели машинного обучения. На примере переобучения многочленов узнают, почему для машинного обучения нельзя использовать слишком сложные модели. Научатся правильно оценивать качество обученных моделей. Изучат первый алгоритм машинного обучения — KNN, обсудят его преимущества и недостатки.

Цель занятия: сформировать у учеников представление о проблемах переобучения, познакомить с методами решения проблем с переобучением, научить правильно оценивать качество обученных моделей.

Задачи занятия:

- выяснить проблемы переобучения;
- разобрать алгоритм машинного обучения KNN;
- закрепить знания о машинном обучении на примерах.

Ход занятия

Этап занятия	Время	Деятельность педагога	Комментарии, рекомендации для педагогов
Организационный этап	5 мин.	Добрый день, ребята! Вы готовы к началу урока?	Проверить готовность детей к уроку. Настроить на работу.
Постановка цели и задач занятия. Мотивация учебной деятельности обучающихся	7 мин.	Мы продолжаем говорить о машинном обучении. На этом уроке мы обсудим проблему переобучения и то, как измерять качество модели машинного обучения. На примере переобучения многочленов вы узнаете, почему для машинного обучения нельзя использовать слишком сложные модели. И научитесь правильно оценивать качество обученных моделей. Также мы изучим первый алгоритм машинного обучения — KNN, обсудим его преимущества, недостатки и способы их исправить.	Учитель мотивирует, предлагает изучить материал.
Изучение нового материала	50 мин.	Посмотрим на стандартный алгоритм действий для создания модели машинного обучения. Эту последовательность мы описывали на предыдущем уроке. Она нарисована на слайде в виде диаграммы. Мы начинаем с того, что формируем матрицу объекты-признаки. Обучаем модель и после делаем	Объяснение материала: беседа с использованием презентации;

		предсказания на данных, которые модель ещё не видела. Если нас не устраивает качество, что происходит почти всегда, мы возвращаемся в начало и меняем модель или процесс обучения. Естественно, нас интересует качество модели на тех данных, которые она не видела в процессе обучения. Именно для этого перед обучением мы должны первым делом разделить данные на те, которые покажем нашей модели, и те, которые показывать не будем. То есть обучающую выборку — train, и тестовую выборку — test. Приведу пример, который покажет, что качество работы модели машинного обучения на обучающих и тестовых данных может очень сильно различаться. Для этого рассмотрим очень простую задачу регрессии, в которой у каждого объекта всего один признак. На графике изображён наш датасет. Он состоит всего из 14 точек: 7 элементов в обучающей выборке и 7 элементов в тестовой. Единственный признак объектов изображён по оси абсцисс, а целевая переменная, которую необходимо предсказать, отложена по оси ординат. Давайте пока что полностью уберём тестовую часть, потому что её мы для обучения алгоритма использовать не будем. Данные, которые видит модель на стадии обучения, будут выглядеть так. Цель — построить функциональную зависимость между признаком и целевой переменной. То есть провести некоторую функцию, на которую наши точки лягут более-менее точно и красиво. **Вопрос для обсуждения** Давайте подумаем, какая модель в данном случае может быть самой простой из возможных?	фронтальная работа.

		Ответ школьников Самая простая функция — линейная, которая на графике изображается прямой линией. Несмотря на свою простоту, это тоже алгоритм машинного обучения. Такой подход к решению задачи называется линейной регрессией. О ней вы узнаете больше в следующих модулях нашего курса. Проведём прямую через зелёные точки. Пройти через все она не сможет. Видно, что приближение получается не очень хорошее. Кажется, что мы могли бы сделать предсказание получше. Кстати, предсказания на тестовом датасете по качеству почти не отличаются от предсказаний на обучающем датасете. Можно сказать, все настолько же плохо. Придумаем что-то более качественное. Возникает естественная идея. Если прямой не хватило для хорошего качества, давайте искать более сложные зависимости. Одна из таких зависимостей — это квадратный трёхчлен. Попробуем подобрать многочлен второй степени, который наиболее близко проходит через наши точки. Иными словами, мы подгоним под точки параболу. Проанализируем это решение. Во-первых, качество на обучающем датасете стало немного лучше. Более того, кажется, что и на тестовом датасете ничего не испортилось. Во-вторых, для тестового и обучающего датасета качество все так же почти не отличается друг от друга.	Педагог вступает в разговор с обучающимися, обсуждают, дополняют друг друга.

		Воодушевлённые успехом, мы идём дальше. Если многочлен второй степени сработал лучше многочлена первой степени, то, может, построить многочлен ещё более высокой степени, например, четвёртой? В самом деле, у многочлена четвёртой степени, как известно, больше степеней свободы, чем у параболы, поэтому и исходную зависимость можно приблизить лучше. Провели. Что ж, это не совсем то, чего мы хотели. Видно, что кривая уже хорошо проходит через точки обучающего датасета, а вот на точках из тестового множества предсказания стали значительно хуже. Для крайней правой красной точки предсказание совсем испортилось. А что если повысить степень многочлена, вдруг случайно получится хорошо? Оказывается, нет. Если мы перейдем к ещё более сложной модели — многочлену 6 степени, то кривая пройдет через все точки обучающего датасета, и наши предсказания на нём будут идеальными. В самом деле, известный математический факт заключается в том, что для любых семи точек можно придумать многочлен не более чем 6 степени, который через них все проходит. А вот на данных, которые наша модель не видела — точках из тестового множества — предсказания станут совсем плохими. Для крайней правой точки мы предсказываем что-то, не имеющее отношения к реальности. Такой эффект называется переобучением. Он хорошо описывается графиком на слайде. На этом графике по оси x отложена сложность модели — в нашем случае, степень многочлена. По оси y отложена	

		величина ошибки модели. На графике изображены две кривые: синяя соответствует обучающей выборке, красная — тестовой. Вначале, когда мы усложняем модель, ошибка падает и на обучающей, и на тестовой выборке. Это происходит, когда мы переходим от прямой к параболе. Но с какого-то момента при дальнейшем усложнении модели ошибка на обучающей выборке продолжает падать, а на тестовой возрастает. Это мы заметили, когда перешли к более высокой степени многочлена. Какая ошибка нас интересует на самом деле — на обучающей выборке или на тестовой? Конечно же, на тестовой, ведь когда мы будем использовать модель в реальном мире, никто не будет давать ей данные, которые она видела в ходе обучения. Поэтому реальное качество будет совпадать с качеством на тестовой выборке, а не на обучающей. Итак, в машинном обучении сложность модели — это то, насколько сложные зависимости она может описывать. Многочлен 4 степени может описать более сложную зависимость между x и y, но при недостаточном количестве обучающих данных он может увидеть в них что-то, чего нет в настоящей зависимости. Таким образом, необходимо находить баланс между простотой модели и её выразительностью. Эта идеальная точка на графике изображена восклицательным знаком. **Вопрос для обсуждения** Как её найти? Есть только один способ — измерять качество на тестовом множестве и останавливаться тогда, когда оно достигает своего оптимума.	

		Именно из-за эффекта переобучения нам и нужно разделение на тестовую и обучающую выборки. Без этого разделения любая оценка качества, которую мы делаем, будет нечестной, и мы будем выбирать сложные модели, которые плохо работают в реальности. Вернём небольшой должок, который у нас остался. Мы рассуждали о качестве в задаче регрессии, но не определили, как именно мы будем оценивать это качество. Действительно, на практике мы не можем просто посмотреть на графики и выбрать лучшую модель. Вместо этого необходимо выражать качество наших моделей в численном виде. Для этого существуют функционалы качества или, как их ещё называют, метрики. Также иногда используют термин «функция потерь». Формально эти термины немного отличаются друг от друга, но сейчас мы не будем обсуждать разницу. Для разных задач и моделей используются разные метрики. В будущем мы познакомимся со множеством различных метрик машинного обучения, а пока остановимся на стандартных. Для задачи классификации в качестве метрики часто используют процент верно угаданных ответов. Эту метрику называют accuracy, причём это название обычно с английского не переводят, чтобы не спутать с другой метрикой, precision. Например, если речь идёт о предсказании выживших пассажиров «Титаника», то accuracy вычисляется как количество пассажиров, для которых модель верно предугадала исход, делить на общее количество пассажиров. Для задачи регрессии используют две основные функции потерь, которые называются MSE и MAE — mean squared error,	Для справки: Accuracy — это метрика, которая характеризует качество модели, агрегированное

		то есть среднеквадратичная ошибка, и mean absolute error, то есть средняя абсолютная ошибка. Формулы для их вычисления вы видите на слайде. В первом случае мы усредняем квадраты отклонений правильного ответа от ответа, который предсказала модель. Во втором случае усредняем модули отклонений. Мы продолжаем знакомство с машинным обучением. **Вопрос для обсуждения** Помните, мы пытались предсказать стоимость одного ноутбука, опираясь на стоимость всех остальных? **Ответы школьников** Не знаю, как вам, а первое, что хотелось сделать мне — выбрать несколько похожих ноутбуков (например, два последних) и предположить, что первый ноутбук по цене должен быть где-то посередине между теми двумя. Так получается довольно точный ответ — мы ошибаемся всего на несколько тысяч. **Что мы сделали?** Выбрали 2 ближайших к нашему ноутбуку объекта из обучающей выборки и, на основе ответов на тех объектах, предсказали ответ на нужном нам объекте. Эта идея и лежит в алгоритме k ближайших соседей (по английски — k nearest neighbors, сокращённо KNN). Сформулируем, как работает этот алгоритм, на примере задачи классификации на три класса: красный, зелёный и жёлтый. Объекты изображены точками на картинке.	по всем классам. Это полезно, когда классы для нас имеют одинаковое значение. В случае, если это не так, accuracy может быть обманчивой.

		Сначала зафиксируем натуральное число k — не очень большое, скажем, k=5. Это количество соседей, которых мы будем рассматривать. Пусть к нам приходит некоторый объект x, на котором нужно сделать предсказание. Давайте найдём k ближайших точек среди всего обучающего датасета. В качестве предсказания отдаём класс, который чаще всего встречается среди этих k точек. На картинке объект, про который мы спрашиваем, находится в центре. Классов всего три: красный, зелёный и жёлтый. Видно, что из 5 ближайших соседей на картинке 4 красных. Мы описали применение уже готовой модели. Обсудим, как обучать модель. Если число k фиксировано, то обучение не требуется, потому что вся сложность — поиск соседей — происходит на стадии применения. Можно сказать, что во время обучения k ближайших соседей мы просто запоминаем все точки из обучающей выборки. Для примера посмотрим на алгоритм одного ближайшего соседа. Пусть у нас есть некоторая выборка. Она изображена на картинке. Раз k=1, то новый объект, который нужно классифицировать, мы относим к классу, которому принадлежит наиболее близкий объект из обучающей выборки. Сделаем вот что: покрасим все точки нашей плоскости в цвет ближайшего к ней объекта из обучающей выборки. Тогда плоскость разделится на три части: красную, синюю и зелёную. Теперь, если понадобится классифицировать новый объект, мы сможем изобразить его на нашей плоскости. Цвет точки, соответствующей данному объекту, определяет класс объекта, который предсказал алгоритм.	Для справки: https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19

		Этот приём визуализации мы будем часто использовать на протяжении курса. Мы обсудили алгоритм k ближайших соседей для задачи классификации. Обобщение алгоритма на случай задачи регрессии придумать несложно. Вместо выбора наиболее частого класса, мы находим среднее значение целевых переменных и отдаём это как предсказание. Так же, как мы делали это в случае с ноутбуками. Давайте подумаем, почему алгоритм KNN работает. Когда мы создаём модель для истинной функции, мы делаем допущения о наших данных. Можно легко понять, почему без допущений ничего не выйдет. Чтобы сделать предсказание, нужно предположить, что находящиеся близко точки имеют близкие друг к другу значения целевой функции. Без этого предположения мы ничего не можем сказать о точке, не лежащей в нашем обучающем датасете. В предположении, которое мы делали раньше, мы использовали расстояние между точками. Близкие точки имеют близкие ответы. Но что такое близкие точки? Близость можно определить по-разному. Например, существует самое обычное расстояние между точками в пространстве. Это корень из суммы квадратов разностей координат — длина отрезка между точками в многомерном пространстве. Но расстояния можно задать и множеством других способов. Например, 1 метр по вертикали можно считать менее значимым, чем 1 метр по горизонтали. А ещё можно	

		считать углы между отрезками, которые идут из центра координат до точек. Посмотрим на k ближайших соседей в действии. У нас есть датасет с признаками для квартир. Для некоторых квартир мы знаем ответ, а для одной нет. Как бы сработал метод 1 ближайшего соседа? Чтобы применить его, нам нужно посчитать расстояния до всех точек в обучающем датасете, найти k самых близких точек к нашей и усреднить целевые переменные для них. Так как k у нас равно 1, мы выбираем только одну самую близкую точку и берём её целевую переменную. Это точка 3. Перейдём к преимуществам и недостаткам метода k ближайших соседей. Первое. Он интерпретируемый, то есть результаты его работы можно объяснить человеку. Когда мы получили какое-то предсказание, можно посмотреть на k ближайших соседей, которые использовались для его получения, и проверить, что мы усредняли. Интерпретируемость часто важна на практике. Представьте, что вы с помощью машинного обучения поставили пациенту тяжёлый диагноз, но не можете объяснить, почему алгоритм так решил. Пациент хочет удостовериться на сто процентов, что машина не ошиблась и можно начинать дорогостоящее лечение. Кроме того, интерпретируемость позволяет отлавливать ошибки в поведении алгоритма. Это плюс.	

		Второе. Метод требует функцию расстояния. Теоретически, если задать хорошую функцию расстояния, ваш KNN станет лучшим предсказателем среди возможных. Проблема в том, что задать такую функцию — тоже очень сложная задача, как и само предсказание. Третье. На KNN лежит проклятие размерности. Это означает, что чем больше признаков у наших объектов, тем сложнее становится делать предсказания. Это связано как со скоростью работы, так и с некоторыми свойствами геометрии n-мерного пространства. Со второй проблемой можно справиться, используя нормирование признаков. Мы научимся бороться с одной конкретной аномалией, которая мешает KNN хорошо работать. Эта аномалия — разный масштаб разных признаков. Пусть нам нужно определить стоимость квартиры. У нас есть датасет с двумя признаками. Первый — расстояние до метро в метрах, а второй — количество комнат. Нам нужно предсказать стоимость с помощью KNN. Понятно, что при поиске ближайших соседей самый большой вклад в расстояние между точками внесёт именно признак «расстояние до метро». Просто потому, что оно обычно исчисляется сотнями или тысячами метров, в то время как количество комнат — это единицы. На картинке график таких квартир. Видно, что переменную x1 можно исключить: в поиске ближайших соседей ничего не поменяется. Для борьбы с такой проблемой используют стандартизацию: из каждого признака мы вычитаем среднее и делим на стандартное отклонение (мы обсуждали эти понятия в модуле про основы теории вероятностей). Тогда масштаб всех признаков получается примерно одинаковым.	

Закрепление изученного материала	15 мин.	**Вопрос для обсуждения** Что вы узнали сегодня? **Возможные ответы школьников:** Мы узнали, что такое переобучение и как оно себя проявляет. Выяснили, что переобучение возникает, когда мы увеличиваем сложность модели. Без разделения на тестовую и обучающую выборки мы будем получать нечестные оценки качества моделей и не заметим переобучения. Поняли, что качество моделей оценивают с помощью функционалов качества или, иначе, метрик, и познакомились с метриками классификации и регрессии. Поговорили о принципах работы алгоритма k ближайших соседей. KNN ищет k ближайших соседей и усредняет их значения целевой переменной в случае регрессии. В случае классификации отдаёт наиболее частый класс. Мы обсудили преимущества и недостатки KNN и узнали, что перед подачей в алгоритм все признаки нужно привести к одному масштабу.	Проверить степень усвоения учебного материала. Учитель на этом этапе фиксирует внимание, задавая уточняющие вопросы.
Этап подведения итогов занятия (рефлексия)	8 мин.	**Вопросы для обсуждения** ● 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.	Посмотреть дополнительную литературу на тему машинного обучения	

Рекомендуемые ресурсы для дополнительного изучения:

1. Машинное обучение: просто о сложном. [Электронный ресурс] – Режим доступа: https://sbercloud.ru/ru/warp/blog/machine-learning-about.
2. Простыми словами о методах решения проблем с переобучением. [Электронный ресурс] – Режим доступа: https://newtechaudit.ru/overfitting/.
3. Машинное обучение для начинающих. [Электронный ресурс] – Режим доступа: https://proglib.io/p/mashinnoe-obuchenie-dlya-nachinayushchih-osnovnye-ponyatiya-zadachi-i-sfera-primeneniya-2021-08-29.
4. Метод k-ближайших соседей. [Электронный ресурс] – Режим доступа: https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19.