ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

Тема занятия: Основы аналитики данных с помощью Pandas

Аннотация к занятию: на данном уроке обучающиеся знакомятся с библиотекой Pandas. В первой части урока они учатся получению информации о данных и их обработке. Во второй части урока обсудят библиотеки для визуализации данных. Вспомнят некоторые полезные функции из Matplotlib, а также познакомятся с новой библиотекой Seaborn.

Цель занятия: знакомство обучающихся с библиотекой Pandas.

Задачи занятия:

- научить загружать данные в Python с помощью библиотеки Pandas;
- познакомить с основными функциями библиотеки Pandas для получения и обработки информации;
- освоить процесс работы с двумя таблицами сразу;
- познакомить с основной функцией визуализации.

Ход занятия

Этап занятия	Время	Деятельность педагога	Комментарии, рекомендации для педагогов
Организационный этап	2 мин.	Здравствуйте! Мы продолжаем знакомство с языком программирования Python	Приветствие. Создание в классе атмосферы психологическог о комфорта
Постановка цели и задач занятия. Мотивация учебной деятельности обучающихся	10 мин.	**Вопросы для обсуждения** Все ли работали с электронными таблицами? Представляете, как выглядит таблица? Как вы думаете, есть ли в Python библиотека, которая позволяет работать с табличными данными? **Ответы обучающихся** Pandas — одна из популярных библиотек для работы с данными и их удобного представления в виде таблиц. Эта библиотека сопровождается подробной документацией, где можно найти множество примеров её использования.	Способствовать обсуждению мотивационных вопросов

Изучение нового материала	50 мин.	Начнём с импорта библиотек. Импортируем знакомые нам NumPy и Pandas. Обратите внимание, что для импорта Pandas, как и NumPy, есть общепринятое сокращение: мы импортируем Pandas как pd. Все, кто занимается машинным обучением, импортирует эту библиотеку именно так. В этом ноутбуке мы будем работать с данными о пассажирах «Титаника». Для каждого пассажира в данных есть информация о том, в каком классе и какой каюте ехал пассажир, в каком порту он сел на корабль, как зовут пассажира, сколько ему лет и пара других характеристик. Также для каждого пассажира известно, выжил ли он при крушении «Титаника» или нет. Данные о пассажирах хранятся в двух файлах. В одном из файлов — titanik_surv — для каждого пассажира хранится информация о том, выжил он при крушении или нет. Во втором файле — titanik_data — хранится вся остальная информация о пассажирах. Обычно данные хранятся в файлах в виде таблиц формата .csv или .xlsx. Но чаще всего данные для машинного обучения хранятся в формате csv. Наши файлы titanik_data и titanik_surv именно такие. Csv — это сокращение от comma separated values. По-русски — значения, разделённые запятыми. Это значит, что если мы откроем файл формата csv в обычном блокноте, мы увидим, что это таблица. В первой строке через запятую записаны названия колонок таблицы. Каждая строка таблицы — это одна строка этого файла. Значения в строках также разделены между собой запятыми. Вот это — значение в первой строке и первом столбце таблицы, вот это — значение в первой строке и втором столбце таблицы и так далее.	Для справки: Сайт: https://pythonim.ru/libraries/biblioteka-pandas-python Перед уроком рекомендуется ознакомиться с материалами, представленным и на сайте.

		Но так смотреть на таблицу, конечно, неудобно. Вернёмся к коду. Чтобы загрузить таблицу из файла csv и работать с ней в Python, нам понадобится библиотека Pandas и её функция read_csv. Во-первых, давайте загрузим файл в Google Colab. Делается это так: открываем меню файлов, нажимаем загрузить, выбираем на компьютере нужный файл. Файл загружен, теперь он отображается у нас здесь. Вернёмся к коду. Чтобы загрузить файл в Python, нужно в функцию pd.read_csv подать путь к файлу. У нас он лежит прямо тут, поэтому просто пишем название файла. Ещё передаем аргумент index_col: сейчас мы поймём, зачем он нужен. Запускаем. Ячейка исполнилась: теперь у нас в переменной titanik_passengers находится таблица с информацией о пассажирах «Титаника». Давайте посмотрим, как таблица выглядит. Это можно сделать так: позвать у переменной, в которой находится таблица, метод head(). Этот метод выводит первые несколько строк таблицы на экран. Внутрь передаём количество строк, которые мы хотим увидеть. Пусть будет десять. Запускаем. Вот как выглядит наша таблица. Очень похоже на Excel-таблицу. Здесь каждая строка — это один пассажир «Титаника». По столбцам — характеристики, признаки пассажиров. Названия столбцов выделены жирным. Что за информацию содержит каждый из столбцов, написано в ячейке ниже. Обратим внимание на столбец PassengerId. Мы указывали название этого столбца в функции чтения файла как index_col. Index_col — это название той колонки в данных, которая	

		содержит уникальный номер строки. Смотрите, тут значения идут от 1 и далее. Получается, эта колонка нумерует пассажиров Титаника, каждому человеку ставит в соответствие его уникальный id. Если в таблице присутствует такая колонка, её название нужно указывать тут в качестве index_col. Хорошо, мы считали нашу таблицу и даже на неё посмотрели. Теперь переходим к изучению того, как с этой таблицей работать. Для начала научимся выводить общую информацию о данных в таблице. Во-первых, можно узнать, сколько в нашей таблице строк и столбцов. Нужно вывести атрибут shape у переменной, в которой лежит таблица. Видим, что у нас восемьсот девяносто одна строка и десять столбцов. Получается, у нас данные о восемьсот девяносто одном пассажире, и для каждого есть десять признаков. Далее: Можно вывести общую информацию о таблице с помощью метода info. Запустим. Что видим: во-первых, здесь вверху написано то, что мы уже узнали в ячейке выше: что в данных восемьсот девяносто одна строка и десять столбцов. Далее появилась информация для каждого из столбцов в таблице. В этой колонке для каждого столбца указано, сколько в нём ненулевых значений. Что это значит: в табличных ячейках иногда бывают пропуски. Они в Pandas обозначаются словом NaN: это сокращение от not a number. Если мы вернёмся к нашей таблице выше, то увидим, что в столбце Cabin есть много значений NaN. Это значит, что для мистера Owen Harris неизвестно значение каюты, в которой он ехал на «Титанике»: тут пусто. Видно, что в столбце Cabin довольно много пропусков.	

		Вернёмся обратно. Значение non-null count показывает, сколько значений в том или ином столбце ненулевые, то есть, не пропущены. Для Cabin мы видим, что не пропущены всего двести четыре значения. Учитывая, что строк у нас всего восемьсот девяносто одна, то пропущенных значений в колонке Cabin намного больше половины: для большинства пассажиров «Титаника» мы не знаем, в какой каюте они ехали. Также здесь видно, что пропуски есть и в колонках Age и Embarked. В остальных столбцах пропусков нет, количество ненулевых значений в них равно количеству строк всего в таблице. Колонка Dtype показывает тип данных в столбце. Int и float значит, что значения в столбце представлены числами. Например, PClass и Age — это числовые столбцы. В этом можно убедиться, посмотрев на таблицу. Dtype object значит, что тип данных колонки — строка. Например, Name — это строковая колонка. Имя человека — это строка, а не число. Вот такую информацию выводит нам info. Ещё есть метод describe. Запустим его. Тут выводится другая информация о колонках. А именно: для всех числовых колонок выводятся статистики: сколько всего ненулевых значений в колонке, минимальное значение в колонке, максимальное значение, среднее и дисперсия, а также некоторые квантили. Понятно, почему эта информация выводится только для числовых колонок: для тех колонок, данные в которых представлены строками, просто нельзя посчитать среднее, дисперсию, минимальное и максимальное значения. Ну какое среднее может быть у колонки Name?	

		Что мы тут видим: например, что средний возраст пассажиров «Титаника» — 29 лет. Максимальный возраст — восемьдесят лет. А минимальный возраст, смотрите, ноль целых сорок два. Это интересно. Это может значить две вещи: либо то, что на борту «Титаника» был ребенок возрастом около полугода, либо что в данных есть ошибка. Например, человек, который заполнял эту таблицу, мог случайно написать ноль целых сорок два вместо сорок два. Подобные ошибки в данных бывают очень часто, полезно уметь их находить и от них избавляться. Дело в том, что модели машинного обучения лучше обучать на данных, в которых ошибок нет. И вот такой первичный анализ данных, которым мы сейчас занимаемся, помогает ошибки находить. Более подробно о том, как именно бороться с ошибками в данных, вы узнаете позже, когда будете обучать модели машинного обучения. А мы пока пойдём дальше. Сейчас мы выводили общую информацию о таблице. Но можно работать и с отдельными столбцами. Взять из таблицы отдельный столбец просто: нужно записать его название в квадратных скобках. С этим столбцом можно делать разные вещи. С ним можно работать, как с массивом. Например, можно узнать, какое минимальное значение в этом столбце: просто напишем .min(). Получили те же ноль целых сорок два. Можно вывести максимальное значение в столбце, среднее, дисперсию значений столбца. Понятно, что все эти величины можно считать только для числовых столбцов. И вообще, столбец можно перевести в массив типа NumPy и работать с ним через NumPy. Делается это просто: в numpy array подаём столбец.	

		Для числовых столбцов также можно построить гистограммы. Делается это с помощью метода hist. Построим гистограмму для столбца Age со значением bins равно восемь. Что вообще такое гистограмма и как её получить: берём столбец Age, вычисляем минимальное и максимальное значения. Это ноль целых сорок восемь десятых и восемьдесят, мы это вычисляли выше. Далее делим этот промежуток от минимального до максимального значения на восемь равных отрезков. Давайте будем считать, что минимальное значение примерно равно нулю, и тогда промежуток от нуля до восьмидесяти делится на восемь отрезков длиной десять. От нуля до десяти, от десяти до двадцати и так далее. Эти отрезки называются бинами. Теперь берём наш столбец Age и для каждого отрезка считаем, сколько значений из столбца Age попадает в этот отрезок. Рисуем на графике прямоугольник со сторонами вдоль концов отрезка и такой высоты, сколько значений из столбца Age попало в этот отрезок. Вот, например, прямоугольник для отрезка от десяти до двадцати. Он имеет высоту около ста десяти. Это означает, что примерно сто десять человек из нашей таблицы были в возрасте между десятью и двадцатью. Мы можем менять значение bins. Например, увеличить его, и тогда промежуток от минимального до максимального значения будет поделён на большее количество отрезков. Поставим, например, bins равно двадцать. Чем больше bins, тем более детально гистограмма отражает информацию о значениях в колонке. Можно сказать, что гистограмма отражает распределение значений колонки: в нашем случае, распределение возраста пассажиров. Вы, наверное, помните термин «распределение» из модуля по теории вероятностей и статистике. Гистограмма имеет непосредственное отношение к распределению: точнее, к его	

		плотности. Смотрите, если бы возраст пассажира был случайной величиной, то гистограмма была бы похожа на график плотности распределения этой случайной величины. И увеличивая bins, то есть, уменьшая длины отрезков, на которые мы делим промежуток, мы бы получали график, всё больше напоминающий плотность распределения случайной величины. Итак, это гистограмма. Что она нам даёт: на самом деле, по ней видно, каково распределение возраста пассажиров «Титаника». Например, видно, что у нас есть очень много людей в возрасте около двадцати-тридцати лет. А ещё у нас почему-то много пассажиров в возрасте около нуля лет. То ли это значит, что на «Титанике» было много детей, то ли у нас ошибки в данных. Таким образом, гистограмма позволяет лучше увидеть, как устроены значения в столбце. Гистограммы также можно строить с помощью библиотеки Matplotlib. В ней тоже есть функция hist, в которую нужно передать значения, по которым затем строить гистограмму, и количество бинов. Видим, что построилось то же самое. Всё, что мы сейчас рассмотрели — min, max, std, гистограмма — применимо к числовым столбцам. Для столбцов, значения в которых строковые, можно вывести value_counts. Давайте выведем для столбца пол. Value counts выводит все значения, которые встречаются в столбце, и рядом пишет, сколько раз это значение встречается. Тут видно, что женщин в нашей таблице триста четырнадцать, а мужчин — пятьсот семьдесят семь. Для числовых колонок тоже можно вывести value_counts. Выведем, например, для Pclass: эта колонка показывает класс каюты, в которой ехал пассажир.	

		Однако для числовых колонок, у которых очень много различных значений, например, для Age, выводить value_counts будет бесполезно: мы получим кучу информации, которую неудобно анализировать. Поэтому для числовых колонок чаще полезнее строить гистограммы. Что ж, вот мы и разобрали основные методы получения общей информации о данных в таблице с помощью Pandas. Сейчас мы познакомимся с индексированием колонок по условиям. Индексирование по условиям — это выделение из таблицы набора строк, которые соответствуют определённым условиям. Пример. В таблице «Титаник» есть мужчины и женщины. Я хочу выделить из таблицы только те строки, которые соответствуют мужчинам. Это сделать очень просто. Нужно взять таблицу titanik_passengers и в квадратных скобках записать условие. Строки, которые соответствуют этому условию, будут выбраны. Здесь записано условие «взять те строки таблицы titanik_passengers, у которых значение в столбце sex равно male». То, что получится из этого выражения, сохраним в переменную titanik_male. Titanik_male — тоже будет таблицей. Давайте выведем первые пять её строк. Проверим, что в этой таблице все строки соответствуют мужчинам, то есть, что у всех строк в колонке sex написано male. Выведем value_counts колонки Sex. Видим, что, действительно, в таблице всего пятьсот семьдесят семь строк, и у всех в колонке пол написано male.	

		Точно так же можно выделить из таблицы строки, у которых, например, значение PClass равно трём, или embarked равно S, и любые другие условия. Зачем может понадобиться выделять какие-то определённые строки таблицы? Это может быть полезно, чтобы отдельно исследовать строки, которые соответствуют тому или иному условию. Например, я могу захотеть исследовать пассажиров мужского пола «Титаника». Выделив строки, соответствующие мужчинам, в отдельную переменную titanik_male, я могу эту таблицу анализировать. Например, узнать минимальный и максимальный возраст мужчин на лайнере, узнать, в каютах какого класса сколько мужчин жило и так далее. Потом, например, могу сделать то же самое для строк, которые соответствуют женщинам, и сравнить, чей средний возраст выше: женщин или мужчин. Сравнить распределение возрастов женщин и мужчин, построив две гистограммы, и так далее. Часто бывает полезно отдельно анализировать ту или иную группу объектов данных, и для этого нам и нужно уметь группы выделять. Вместо чистого равенства можно ставить знаки больше или меньше. Например, получим строки таблицы, в которых значение колонки Age больше либо равно восемнадцати. Иначе говоря, выделим всех совершеннолетних пассажиров «Титаника». Идём дальше: усложним задачу. Зададим сложное условие: пусть я хочу вывести всех пассажиров мужского пола, возраст которых от восемнадцати до двадцати двух лет. Тогда в квадратных скобках нужно записать три этих условия, и каждое взять в круглые скобки. Между ними я ставлю союз «и»: я же хочу тех людей, которые и мужчины, и в возрасте больше восемнадцати и в возрасте меньше двадцати двух. «И» выражается амперсандом.	

		Запустим. Видим, что в полученной таблице и правда только мужчины и только в возрасте от восемнадцати до двадцати двух. Аналогично можно задавать условия с «или». Выведем тех пассажиров, у которых класс каюты либо второго, либо первого класса. «Или» обозначается прямой палочной. Отлично, идём дальше. Это же условие: класс каюты либо второго, либо первого класса можно записать по-другому. Здесь написано, что мы выбираем те строки таблицы titanik_passengers, у которых значение колонки PClass находится среди значений этого массива: 1 и 2. Видим, что на выходе мы получили ту же таблицу, что и в ячейке выше. Мы рассмотрели несколько примеров индексирования по условиям. Различные условия можно комбинировать и таким образом выделить в отдельную таблицу строки, соответствующие любым критериям. Теперь давайте поговорим о том, как можно модифицировать таблицу: создавать новые колонки. Посмотрим на столбцы SibSp и Parch. SibSp — это количество братьев, сестёр и супругов пассажира, которые также плывут на корабле. Parch — это количество родителей и детей пассажира, которые также плывут на «Титанике». Сумма значений в этих двух колонках — это общее количество родственников пассажира на борту. Давайте сделаем новую колонку, в которой и будет это записано: для каждого пассажира в ней будет сумма значений его колонок SibSp и Parch.	

		Делается это очень просто. Мы можем взять и сложить эти две колонки. Действительно, колонка таблицы имеет тип Pandas Series, как вы помните из предыдущего модуля, а с ними можно проводить арифметические операции. Получили в итоге новый Series, который назовем total_relatives. Теперь добавим эту новую колонку в нашу таблицу. Нужно выбрать ей название: пусть это тоже будет totalrelatives. Пишем, что в таблице titanik_passengers будет новая колонка с названием totalrelatives, — она будет равна нашей переменной. Выводим теперь первые строки нашей таблицы, и видим, что в конец добавилась наша колонка. И её значения для каждого пассажира, каждой строки, равны сумме значений колонок SibSp и Parch в этой строке. Так, добавлять колонки мы научились. Но их иногда нужно ещё и удалять. Удалять колонки тоже просто: с помощью метода drop. Давайте, например, удалим колонки SibSp и Parch: у нас уже есть колонка totalrelatives с общим количеством родственников пассажира, так что SibSp и Parch уже не нужны. Пишем titanik_passengers drop и в качестве аргумента columns подаём список названий колонок, которые мы хотим удалить. Запишем то, что получится, обратно в переменную titanik_passengers и выведем первые пять строк. Видим, что колонки удалились. Итак, мы научились добавлять новые колонки и удалять старые. Можно еще делать вот что: изменять часть значений колонки по некоторому условию. Например, пусть я хочу переделать колонку Sex в числовую: вместо male написать ноль, вместо female — один. Я могу заменить все значения male колонки Sex на ноль. Делается это так.	

		Пишем titanik_passengers.loc, внутри — условие для тех строк, в которых мы хотим заменить значение колонки. Условие пишется так же, как мы это делали выше: чтобы взять все строки, у которых в колонке Sex значение male, мы пишем так. Затем через запятую пишем ту колонку, в которой мы хотим поменять значение в строках, которые соответствуют условию. Мы пишем Sex. Далее закрываем квадратные скобки, равно и пишем то значение, на которое хотим заменить значения вот в этих строках и этой колонке. Мы хотим заменить на ноль. Итак, что выходит: выходит, что в таблице titanik_passengers в строках, которые соответствуют вот этому условию, значение в столбце Sex будет заменено на ноль. Давайте в этом убедимся: видим, что вместо male здесь теперь нули. Можно задавать условие по одному столбцу, а заменять значение в другом. Например, мы могли бы выбрать всех мужчин и заменить у них всех значение в столбце Pclass на ноль. Тогда сюда нужно было бы вписать Pclass. Следующее, что мы рассмотрим — это слияние двух таблиц. Помните, в самом начале мы говорили, что наши данные разделены на две таблицы: в той, с которой мы сейчас работали, находятся характеристики пассажиров, а во второй — titanic_surv — находится информация, кто из пассажиров выжил при крушении, а кто — нет. Давайте скачаем вторую таблицу и загрузим в Colab. Теперь откроем её в коде. Здесь для каждого пассажира из таблицы titanik_passengers записано один или ноль: один, если пассажир выжил, ноль, если нет. Теперь: как понять, какому пассажиру какая строка соответствует? Вот по этой колонке индексов PassengerId. В этом файле эта колонка соответствует колонке passenger_id нашей большой таблицы.	

		Например, мы видим, что мистер Owen Harris на «Титанике», к сожалению, не выжил. При загрузке файла в Python мы тоже указывали колонку PassengerId как колонку индексов. Давайте объединим эти две таблицы в одну. Делается это очень просто. Берём одну из таблиц, пишем join и в качестве аргумента передаём вторую таблицу. В итоге получится одна таблица. Запишем её обратно в переменную titanik_passengers. Посмотрим, что вышло. Видим, что получилась одна большая таблица со всеми колонками. Обратите внимание: колонка PassengerId в итоговой таблице всего одна. Именно эта колонка, колонка индексов, использовалась при слиянии двух таблиц функцией join, чтобы определить, какие строки таблицы titanic_surv должны соответствовать определённым строкам таблицы titanik_passengers. И именно за этим мы указывали эту колонку как index_col при загрузке обеих таблиц: чтобы Pandas считал эту колонку как колонку-идентификатор строк таблиц и понимал, какая строка одной таблицы соответствует какой строке другой. При слиянии двух таблиц по этой колонке pandas join оставляет в полученной таблице только одну колонку индексов. Отлично. Мы научились объединять таблицы. Следующее, о чём мы поговорим — это индексация строк. Смотрите, мы научились получать из таблицы отдельные столбцы: для этого нужно просто написать название столбца в квадратных скобках. А вот одну строку таким же способом получить не выйдет. Чтобы получить строку по её номеру, нужно добавить iloc. И после, в квадратных	

		скобках, записать номер строки. Мы получим строку вот такого вида. У этой строки уже можно брать значения в каждой из колонок. Например, получим значение первой строки колонки Name: вот оно. Есть и второй способ взять значение из определённой ячейки таблицы. Можно воспользоваться методом loc. В квадратных скобках через запятую написать номер строки и название столбца, который хотим взять. Смотрите, однако, что вышло: казалось бы, мы взяли один и тот же номер строки, но имена отличаются. Дело вот в чём. iloc берёт первую по счёту строку с начала таблицы. Нумерация при этом идёт с нуля. Смотрите, Мистер Джон Бредли действительно находится на первой по счёту строке в таблице, если нумеровать строки с нуля. А loc берёт не первую строку с начала, а ту строку, в чьей колонке индексов записана единица. То есть через loc можно обращаться к строкам, используя их индексы. Это ещё одна причина, по которой мы указали колонку индексов при загрузке файлов в Питон: с помощью этой колонки можно очень удобно обращаться к отдельным строкам таблицы через метод loc. Значение в ячейках таблицы можно менять. Например, давайте изменим имя пассажира в строке с индексом один. Просто присвоим элементу новое значение Петя. Выведем таблицу. Видим, что имя человека тут действительно изменилось на Петя. Итак, мы научились получать отдельные строки и элементы	

		таблицы. Кажется, мы изучили практически все основные методы для работы с таблицами в библиотеке Pandas. Конечно, в следующих модулях вы изучите ещё больше возможностей библиотеки Pandas, когда будете учиться обрабатывать реальные датасеты, чтобы потом на них обучать модели машинного обучения. Прежде чем мы продолжим работу с библиотеками для визуализации, скажу ещё одно важное замечание относительно Pandas. Pandas, как мы убеждаемся, очень удобная библиотека для работы с табличными данными: в ней есть много удобных функций для анализа и изменения таблиц. Но есть один нюанс. Возможно, вам когда-то захочется пройтись по таблице циклом. Например, сделать цикл по строкам таблицы, внутри которого вы для каждой строки что-то делаете. Скажем, посчитать количество людей в таблице, которым больше двадцати лет, и сделать это именно с помощью цикла. Вот как это может выглядеть. Заводим переменную, в которой будем хранить количество людей старше двадцати. Далее циклом проходимся по каждой строке таблицы и внутри цикла сравниваем значение в колонке Age этой строки с двадцатью. Если человек старше двадцати, добавляем к переменной единицу. Давайте замерим время работы такого кода. Импортируем библиотеку time, замерим время до начала работы цикла и после. И выведем разницу в секундах этих двух моментов времени. Запускаем. Получим вот что: тридцать четыре сотых секунды. На самом деле это очень много. Наша таблица довольно маленькая, в ней меньше тысячи строк, и уже время прохождения цикла по её строкам занимает около трети секунды. А	

		представьте себе, если бы у нас была таблица с реальными данными, в которой было бы миллион строк или больше. И при этом мы бы внутри цикла делали что-то сложнее, чем просто сравнение значения ячейки таблицы с числом. Тогда время работы такого цикла было бы очень большим. В целом, любые циклы по элементам Pandas-таблиц работают очень долго. Так делать не нужно. Но что же тогда делать? Как писать код, если хочется сделать подобный цикл? Вариантов два. Первый вариант — пользоваться возможностями Pandas. Выше мы с вами разбирали индексирование по условиям. Вы, наверное, уже догадались, что посчитать количество людей в таблице, которые старше двадцати, можно и без цикла, а как раз с помощью индексирования по условиям. Нужно просто выделить из таблицы те строки, у которых в колонке Age значение больше двадцати, и взять длину, то есть, количество строк полученной таблицы. Давайте замерим время работы такого кода. Видим, что он работает менее чем за сотую долю секунды: в сорок раз быстрее цикла в ячейке выше. И всегда, когда вам захочется написать цикл по элементам Pandas-таблицы, вспоминайте, что то же самое, что вы хотели написать циклом, скорее всего, можно написать по-другому, без цикла, с помощью Pandas. Практически в ста процентах случаев цикл можно заменить на более быстрый и даже удобный код, используя различные инструменты Pandas. У Pandas намного больше возможностей и методов, чем мы разобрали в этом модуле. Там есть практически все функции для любых манипуляций с таблицами. Если вы хотите что-то сделать с	

		таблицей, но не знаете, как это сделать в Pandas, вы можете воспользоваться поиском в интернете или поиском по документации Pandas. Гарантирую, что вы найдете, как реализовать нужную вам функцию с помощью этой библиотеки. Но всё же бывает, что очень хочется сделать цикл по строкам таблицы. И тогда лучше делать это так: перевести таблицу Pandas в формат NumPy. Это делается очень просто: подаём нашу таблицу в качестве аргумента в numpy array. И вот в этой переменной окажется NumPy-версия нашей таблицы. Давайте на неё посмотрим. Видим, что это массив из строк нашей таблицы, двумерный массив. К его элементам можно обращаться по индексам строки и столбца. Например, выведем первую строку. Видим, что первый элемент — это значение столбца PClass таблицы, второй — Name, и так далее. Age — это четвёртый элемент этой строки. Или, если нумеровать с нуля, то третий. В строках NumPy-версии таблицы нет информации PassengerId, потому что это колонка индексов. Она теряется при переводе таблицы в NumPy. В NumPy сохраняется начинка таблицы: всё, кроме колонки индексов и названий столбцов. Тогда тот же цикл подсчёта пассажиров старше двадцати лет можно записать таким образом: проходимся циклом по строкам NumPy-версии нашей таблицы. В каждой строке сравниваем третий по счёту элемент строки с двадцатью. И прибавляем единицу к переменной, если больше двадцати. Запустим этот цикл с замером времени. Видим, насколько этот цикл отработал быстрее, чем цикл по строкам Pandas-таблицы.	

		Итог такой: когда вам хочется сделать цикл по элементам Pandas-таблицы, не делайте этого. Либо ищите удобные функции Pandas, которые позволят вам реализовать то, что вы хотите, без цикла, либо делайте цикл по NumPy-версии таблицы. Таким образом, Pandas и NumPy отлично дополняют друг друга: совместно они позволяют очень эффективно работать с данными. Поэтому эти две библиотеки — основные для работы с данными в машинном обучении. Далее вы ещё встретитесь со многими другими примерами применения одной и другой библиотеки. Почти все функции библиотек Matplotlib и Seaborn, которые мы будем обсуждать сегодня, работают с числовыми колонками в данных. Поэтому давайте переведём две текстовые колонки — Sex и Embarked — в числовой вид, чтобы их тоже можно было визуализировать. Остальные колонки в числовой вид переводить не будем. Колонка Sex. Давайте заменим в ней значения male на один, а female — на ноль. Это очень просто сделать вот этим кодом. После этого, если мы посмотрим на таблицу, увидим, что колонка Sex стала числовой. Заменим на числа и колонку Embarked. Выведя value_counts, видим, что в этой колонке встречается три разных значения: S, C и Q. Создадим словарь, где каждой букве присвоим число. И теперь заменим значения в колонке на соответствующие числа с помощью метода map. Нужно просто вызвать map у нужной колонки и передать на вход словарь значений: что нужно поменять на что. Теперь снова выведем head и убедимся, что значения в колонке поменялись на нужные числа. Видим, что так и есть.	Для справки: Корреляция - это статистическая взаимосвязь двух или нескольких случайных величин

		Итак, переходим к визуализации. С библиотекой Matplotlib мы уже немного работали выше: строили гистограмму. Также вы с ней работали в прошлом модуле. Давайте вспомним одну из самых часто используемых функций Matplotlib: scatter. Возьмём два столбца данных: например, Age и Fare, и визуализируем их зависимость между собой. Передадим в функцию plt.scatter эти два столбца. Что мы здесь видим: каждая точка на графике — это одна строка, один пассажир нашей таблицы. По оси X отложено значение колонки Age пассажира, по оси Y — значение колонки Fare. Давайте подпишем оси: делается это просто. В plt.xlabel подаём название оси X, в plt.ylabel подаём название оси Y. Получается, на графике мы видим то, как соотносятся между собой признаки Age и Fare. Очень часто по таким визуализациям отношений двух признаков можно получить полезную информацию для обучения модели машинного обучения. Об этом вы подробнее узнаете в следующих модулях. Можно сделать визуализацию ещё более информативной. Добавим в функцию scatter аргумент c. C — это сокращение от color, цвет. Положим c равным колонке survived нашей таблицы. Что значит, что цвет равен колонке survived? Это значит, что каждая точка на графике покрасится в один из двух цветов, в зависимости от того, какое у пассажира значение в колонке survived. Точки выживших пассажиров будут покрашены в один цвет, а тех, кто не выжил — в другой. Запустим. Вот так это выглядит. По таким визуализациям можно видеть, как отличаются объекты с разными значениями колонки	

		survived, и часто эта информация помогает обучать модели машинного обучения. Точно так же можно построить такие и для других колонок таблицы. В качестве первого аргумента в scatter подаём одну из колонок, значения которой будут координатами оси X, а в качестве второго аргумента подаём другую колонку. Однако есть более быстрый и удобный способ получить такие визуализации для всех пар числовых признаков в таблице сразу. Делается это с помощью библиотеки Seaborn. Импортируем эту библиотеку. Она, как и другие библиотеки, имеет общепринятое сокращение: sns. Нужная нам функция из этой библиотеки — pairplot. В эту функцию подаём сразу всю нашу таблицу. В качестве аргумента hue указываем ту колонку, которая будет отвечать за цвета точек. Подаём сюда колонку survived: точки будут покрашены в два цвета в зависимости от того, какое значение соответствующего точке пассажира в колонке survived. Запускаем. Вот что видим: для каждой пары числовых признаков здесь есть визуализация такого же типа, что мы делали выше с помощью scatter. Здесь есть пояснение, какому цвету точек соответствует какое значение колонки Survived. Например, вот визуализация отношения колонок Fare и Age: действительно выглядит как то, что у нас получалось выше. Бежевые точки соответствуют людям, которые не выжили на Титанике, фиолетовые — тем, кто выжили.	

		Также здесь есть вот такие визуализации на пересечении колонки с самой собой. Это две гистограммы. Гистограмма бежевого цвета — это гистограмма, или, можно сказать, распределение значений колонки Age для пассажиров, которые не выжили на «Титанике». Фиолетовая гистограмма — распределение значений колонки Age для пассажиров, которые выжили. По этим распределениям можно понимать, как отличаются значения тех или иных колонок для выживших и не выживших пассажиров. Часто бывает, например, что распределения каких-то колонок сильно отличаются для объектов одного и другого класса. Из этого можно сделать вывод, что значение в этих колонках сильно влияет на то, к какому класса относится объект. Ещё одна очень полезная функция библиотеки seaborn — это визуализация корреляций признаков. Возьмём нашу таблицу и напишем corr. Это выведет нам таблицу, где для каждой пары числовых признаков будет написан коэффициент корреляции этих двух признаков. Это очень важная информация, которая помогает в анализе данных: по ней мы понимаем, насколько различные признаки связаны между собой и как значение одного признака влияет на другие. Также для настройки некоторых моделей машинного обучения очень важно понимать, насколько признаки в данных коррелируют между собой. С помощью Seaborn можно визуализировать эту таблицу в более информативном виде. Подаём матрицу корреляций в функцию sns.heatmap. Получаем вот такую визуализацию: здесь ячейки покрашены в цвета в зависимости от значения корреляции признаков. Теперь визуально видно, какие признаки коррелируют сильно, а какие слабо.	

		Аргумент annotate True задаёт, чтобы внутри ячеек были написаны значения корреляции. Можете убрать этот аргумент и посмотреть, что будет. Последняя функция, которую мы рассмотрим, — это countplot. Давайте подадим этой функции на вход колонку Pclass и в качестве параметра hue снова зададим колонку Survived. Получим вот что. Здесь на оси X расположены значения, которые бывают в колонке Pclass. Для каждого значения есть два столбика: этот столбик — сколько в таблице есть строк со значением Pclass 1 и значением survived 0. Этот столбик — сколько в таблице есть строк со значением Pclass 1 и значением survived 1. Тут те же пары для Pclass 2 и 3. Можно сказать, что это визуализация value_counts, но ещё и с разделением по survived. Опять же, здесь наглядно можно видеть, как отличаются значения признаков для разных пассажиров: которые выжили и которые нет. Эта функция хороша для визуализации признаков, у которых немного разных значений. Если передать Age в функцию countplot, получится не очень визуально понятная картина. Это потому, что в Age много разных уникальных значений, и каждое из них отображается на оси X отдельно. Для визуализации признаков типа Age, у которых много различных значений, лучше использовать гистограммы, как тут.	

Закрепление изученного материала	15 мин.	**Вопросы для обсуждения** • Как загрузить данные в Python с помощью библиотеки Pandas? • Какие операции можно выполнять с ячейками и таблицами? • С помощью каких функций можно визуализировать данные?	Педагог организует беседу по вопросам
Этап подведения итогов занятия (рефлексия)	8 мин.	**Вопросы для обсуждения** • Чему я научился? • С какими трудностями я столкнулся? • Какие вопросы остались? Что осталось непонятным?	Педагог способствует размышлению обучающихся над вопросами
Информация о домашнем задании, инструктаж по его применению	5 мин.	В этом задании вы проанализируете данные из Appstore. Не бойтесь гуглить и заглядывать в «Полезные ссылки» для того, чтобы выполнить какие-то задания. Возможно, на семинаре не было какого-то нужного метода, но он находится в поисковике за 2 минуты. Больше про pandas можно найти по этом полезным ссылкам: Официальные туториалы: http://pandas.pydata.org/pandas-docs/stable/tutorials.html Статья на Хабре от OpenDataScience сообщества: https://habr.com/company/ods/blog/322626/ Подробный гайд: https://media.readthedocs.org/pdf/pandasguide/latest/pandasguide.pdf	

Рекомендуемые ресурсы для дополнительного изучения:

1. ПИТОНТЬЮТОР. [Электронный ресурс] – Режим доступа: http://pythontutor.ru/.
2. Онлайн курс «Поколение Python»: курс для начинающих. [Электронный ресурс] – Режим доступа: https://stepik.org/course/58852/syllabus.
3. Библиотека Pandas в Python. [Электронный ресурс] – Режим доступа: https://pythonim.ru/libraries/biblioteka-pandas-python.
4. Библиотека Matplotlib в Python. [Электронный ресурс] – Режим доступа: https://pythonim.ru/libraries/biblioteka-matplotlib-v-python.