ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

Тема занятия: Знакомство с библиотеками Pandas и Matplotlib

Аннотация к занятию: на данном уроке обучающиеся знакомятся с библиотеками Pandas, Matplotlib и Seaborn. В первой части урока они учатся использовать специальную библиотеку Pandas для работы с большим объёмом данных. Во второй части урока знакомятся с двумя самыми популярными библиотеками для создания графиков на Python: Matplotlib и Seaborn.

Цель занятия: формирование у обучающихся представления о библиотеках Pandas, Matplotlib и Seaborn.

Задачи занятия:

- познакомить с библиотекой Pandas и рассмотреть, какой тип данных она использует;
- научить загружать в Python большие объёмы информации;
- узнать о способах первичного анализа данных в Pandas;
- познакомить с библиотеками Matplotlib и Seaborn.

Ход занятия

Этап занятия	Время	Деятельность педагога	Комментарии, рекомендации для педагогов
Организационный этап	2 мин.	Добрый день! Очень рада вас видеть на занятии!	Приветствие. Создание в классе атмосферы психологическог о комфорта
Постановка цели и задач занятия. Мотивация учебной деятельности обучающихся	10 мин.	**Вопросы для обсуждения:** Анализ входных данных — важный этап, без которого в машинном обучении не обойтись. Для корректного выбора модели и её параметров нужно хорошо разбираться в том, ● кто является источником данных: человек или машина, ● какая форма зависимости может возникать в данных: линейная или нелинейная, ● есть ли необходимость в интерпретации обученной модели и многом другом.	Способствовать обсуждению мотивационных вопросов

		 Почему для анализа данных нам необходима сторонняя библиотека? Ведь в самом Python существуют массивы, которые хорошо сохраняют большие объёмы информации. **Ответы обучающихся** Ранее мы говорили, что машинное обучение работает не просто с большими данными, а с гигантскими объёмами, поэтому классические типы данных Python нам не подойдут. Мы будем использовать для этого специальную библиотеку Pandas и её встроенные типы данных. Давайте с ними познакомимся.	
Изучение нового материала	55 мин.	Для начала импортируем библиотеку Pandas. `[1] import pandas as pd` Теперь познакомимся с новым типом данных Series. Он напоминает одномерный массив, вот только в качестве индекса могут выступать как цифры, так и строки. Например, сейчас мы создадим серию из 4 месяцев, в качестве индекса выступят строки.	Для справки: Сайт: https://habr.com/ru/company/otus/blog/540526/ Перед уроком рекомендуется ознакомиться с

```
[2]  data = pd.Series(["Январь", "Февраль", "Март", "Апрель"],
                      index = ['Первый', "Второй", "Третий", "Четвёртый"])
```

```
[3]  data
```

```
Первый        Январь
Второй       Февраль
Третий         Март
Четвёртый     Апрель
dtype: object
```

Важно: чтобы получить несколько значений метода .loc, необходимо передавать индексы массивом и указывать дополнительные квадратные скобки.

```
[ ]  data[["Первый", "Третий"]]
```

```
Первый    Январь
Третий      Март
dtype: object
```

У метода .loc есть брат-близнец .iloc. Он позволяет выбрать конкретную ячейку набора данных по индексу, что напоминает привычную работу с массивами.

материалами, представленным и на сайте

```
[11] data[[0,3]]

        Первый      Январь
        Четвёртый   Апрель
        dtype: object
```

Хотя у .loc и .iloc схожий принцип работы, они не заменяют друг друга.

Тип данных Series используется не так часто. Обычно это результат обработки другого типа данных — DataFrame.

```
[3] df = pd.DataFrame({'col1': [1, 2], 'col2': [3, 4]})

    df
```

DataFrame — это двумерная сущность, которая по своей форме напоминает таблицу в Excel. В ней есть строки и столбцы, которые создают уникальное положение элементов.
Создать Dataframe можно несколькими способами: из массива или Series.

Если не указывать индексы строк явно, то Pandas самостоятельно присвоит им числовую последовательность.

Для доступа к элементами в DataFrame используются те же методы, что и у Series. Для получения доступа к конкретному столбцу достаточно указать его в квадратных скобках, а для

вызова строк указать метод .iloc и записать индекс требуемой строки.
DataFrame можно воспринимать как двумерный массив, поэтому для доступа к конкретной ячейке достаточно записать комбинацию из индекса строки и столбца.

Вы познакомились с двумя новыми типами данных библиотеки Pandas, на которых строится логика её работы. Рассмотрим, как их применять, на примере реального набора данных.

Для этого изучим функцию read_csv. Она отвечает за загрузку больших наборов данных и автоматически создаёт из них тип данных DataFrame. Кроме того, эта функция умеет автоматически скачивать наборы данных по ссылке, чем мы и воспользуемся.

```
[38] df = pd.read_csv("https://ai-academy.ru/upload/files/football.csv") #index_col = 0
```

```
[39] type(df)
```

Обращаем ваше внимание на то, что функция вернула в программу DataFrame. Загруженная нами таблица была очень тяжёлой, поэтому визуализация с помощью функции print() нам не подходит. Вместо неё в библиотеке Pandas есть метод head(). По умолчанию он выводит первые 5 строчек датафрейма, что позволяет оценить корректность данных и увидеть все столбцы.

	Unnamed: 0	Name	Age	Nationality	Club	Value	Wage	
0	0	L. Messi	31	Argentina	FC Barcelona	110500000	565000	
1	1	Cristiano Ronaldo	33	Portugal	Juventus	77000000	405000	
2	2	Neymar Jr	26	Brazil	Paris Saint-Germain	118500000	290000	
3	3	De Gea	27	Spain	Manchester United	72000000	260000	
4	4	K. De Bruyne	27	Belgium	Manchester City	102000000	355000	

Посмотрим на нашу таблицу. В ней есть столбец Unnamed, значения которого дублируют индексы строк. Это говорит о том, что в изначальных данных уже присутствовали индексы. Добавим в функцию read_csv новый параметр index_col = 0 и понаблюдаем за результатом.

Продолжим анализ загруженного набора данных. В этом нам помогут следующие методы и параметры анализа:

- .shape возвращает кортеж из числа строк и столбцов у DataFrame;
- .columns возвращает коллекцию с названиями колонок;
- .info() — базовая информация о всех строках в датафрейме.

Вопрос для обсуждения
Как же нам посмотреть всю информацию из большого объема данных?

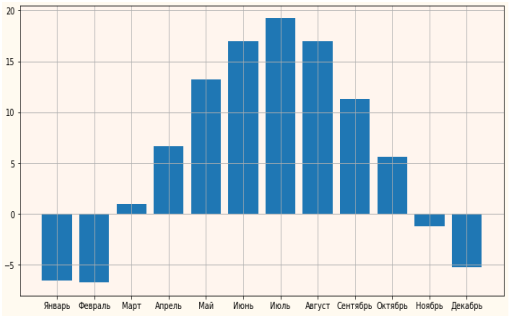
Ответы обучающихся

Визуализация — отличный инструмент для анализа большого объёма информации.

Вопрос для обсуждения
Зачем нам графики?

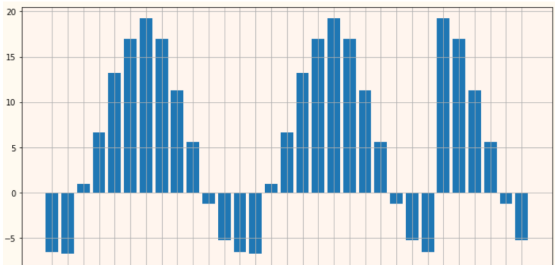
Ответы обучающихся
Для демонстрации методов машинного обучения часто используют графики. Это связано с их наглядностью. Посмотрим, как выглядят данные о среднемесячной температуре в Москве в таблице и на графике:

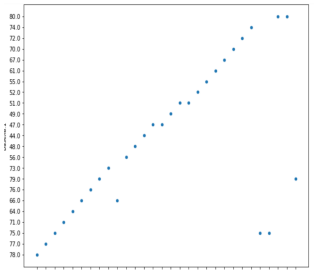
Средняя погода в Москве	
Январь	-6,5
Февраль	-6,7
Март	1
Апрель	6,7
Май	13,2
Июнь	17
Июль	19,2
Август	17
Сентябрь	11,3
Октябрь	5,6
Ноябрь	-1,2
Декабрь	-5,2



В таблице заключена текстовая информация, которую необходимо «читать», перемещаясь по строкам и столбцам. Графики — это визуальное представление данных, их можно проанализировать быстрее, чем таблицу. Если вас попросят назвать самый жаркий месяц, то при помощи графика это сделать намного быстрее.

Что если информации станет ещё больше? Тогда мы изменим форму самого графика. На смену гистограмме, где у каждого значения есть свой столбец, придёт точечная диаграмма. Это позволит охватить больше информации.



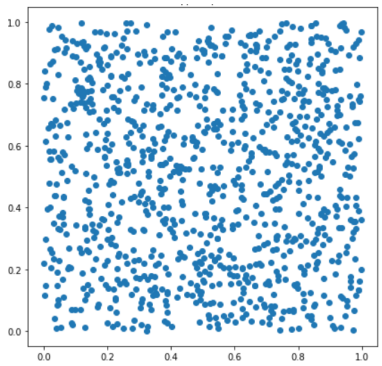


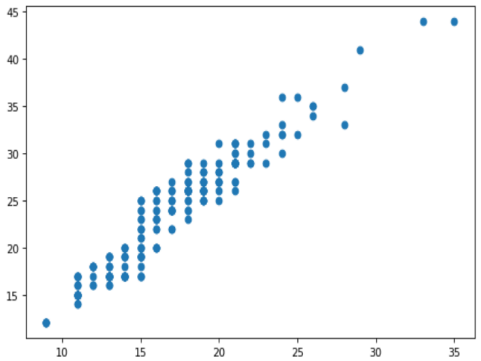
Как работает точечная диаграмма

Точечная диаграмма (Scatter plot) — частый гость в машинном обучении и анализе данных. Иногда её называют диаграммой рассеяния. Посмотрим, что выделяет этот тип визуализации среди других графиков.

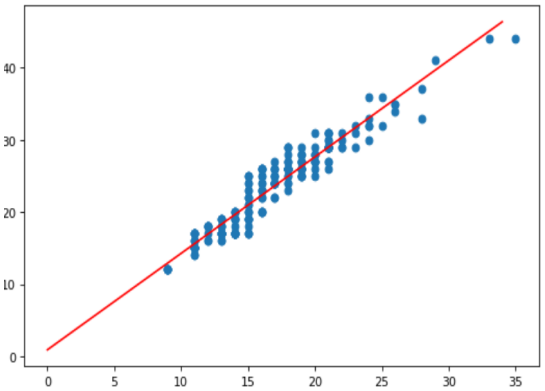
Диаграмма рассеяния использует координаты для отображения зависимости двух переменных. Данные

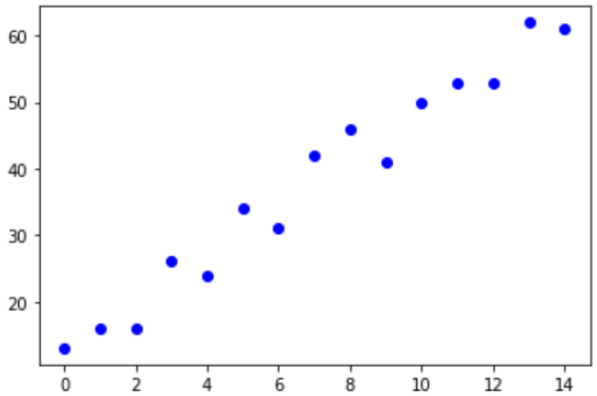
отображаются в виде набора точек, и по форме их распределения можно сделать интересные выводы.

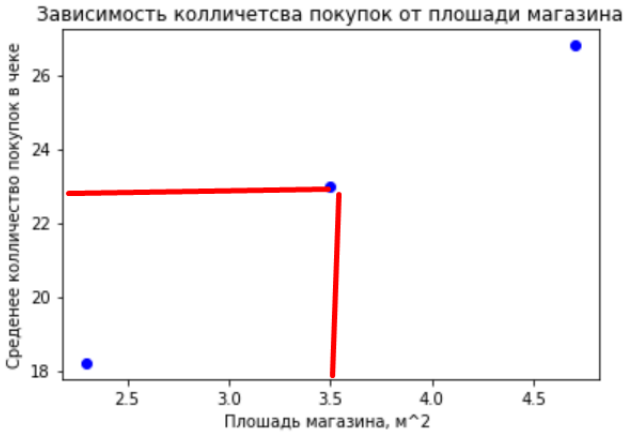




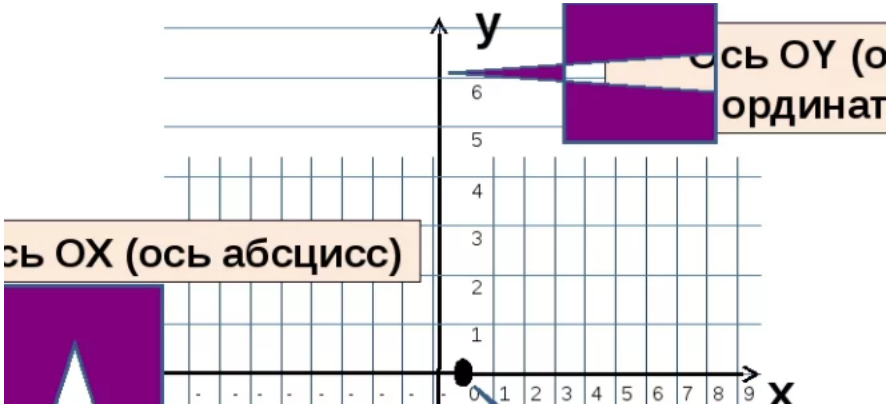
Заметно, как данные на правом графике расположились вдоль одной прямой. Между ними существует связь, которую можно отобразить:



		Если связь между данными можно описать прямой линией, как в нашем случае, то это линейная зависимость (linear dependence). И наоборот, если данные не имеют чётко выраженной зависимости, их называют хаотичными (chaotic distribution). Что могут сказать точки на графике? Что скрывается за значениями, которые мы отображаем на графике? Это могут быть две любые числовые характеристики. В точечной диаграмме не существует иерархии и последовательности, все точки равны между собой. Рассмотрим график зависимости товаров в корзине покупателя от суммы чека:  Сократим количество значений на графике и присмотримся к их положению. Для этого проведём перпендикулярные линии к основанию графика:	Для справки: Сайт: https://colab.research.google.com/drive/1OzwncrLx0HFh_p9pAR09XWXqf5Bcp0rP?usp=sharing



Значения по горизонтали называют значениями оси X, а по вертикали — значениями оси Y. По графику видно, что если сумма чека составит 3,5 тысячи, то средний покупатель приобретёт 23 товара. С увеличением денег растёт количество товаров, а значит и объём возможных покупок. Это говорит о том, что между осью X и осью Y существует зависимость.

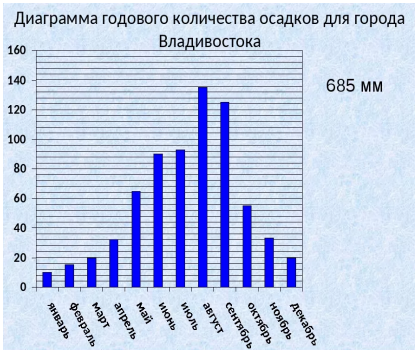
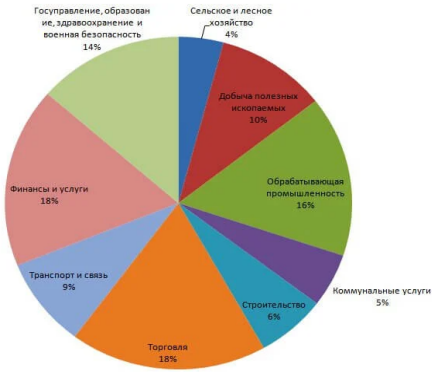


Типы данных

Не все данные можно представить в виде точечной диаграммы. Перед тем как нанести значения на график, необходимо определить их тип. Существует три типа данных:

- количественные (quantitative) — значения некоторого признака, которые можно измерить (возраст, рост, температура);
- качественные (categorical) — значения одного фиксированного положения из возможных значений (семейное положение, этническая принадлежность);
- ранговые (rank) — значения порядковой шкалы, не имеющие численного определения (место на соревнованиях, этаж в доме).

Диаграмма рассеяния демонстрирует зависимость двух количественных переменных. Именно поэтому она не сможет отобразить зависимость среднего балла по математике у мальчиков и девочек, так как пол является качественным типом данных.

		Тем не менее, в статистике есть множество других графиков. Например, для представления числовых данных и их изменений во времени используют гистограмму. А если нужно показать отношение частей данных друг к другу, то стоит воспользоваться круговой диаграммой.   	

Для обозначения подборки графиков, диаграмм или изображений с минимумом сопроводительного текста даже есть специальный термин — инфографика. Визуальный анализ данных помогает лучше представить информацию, чтобы она была понятна не только дата-сайентисту, но и людям без специальной подготовки. Проще посмотреть на график и сразу представить характер данных, чем листать несколько страниц отчета, согласны?

Вы уже понимаете, что настоящему аналитику данных без визуализации не обойтись. Посмотрим, как именно работать с ней в Python. Нужны ли особые библиотеки? Всё просто: если нужно быстро визуализировать результаты анализа, используйте уже знакомую библиотеку Pandas, — её возможностей хватит, чтобы построить основные типы графиков и диаграмм.

Если нужно оформить свой график нестандартно, добавить интерактивные элементы, то потребуются профессиональные библиотеки визуализации данных, такие как Matplotlib, Seaborn или Plotly. Они подходят, когда задача визуализации стоит на первом месте, например, при составлении годового отчёта крупной компании.

```
[ ]  import matplotlib.pyplot as plt

     %matplotlib inline
```

Гистограмма — это диаграмма, которая группирует числовые данные в ячейки, отображая их в виде сегментированных столбцов. Они используются для отображения распределения набора данных: как часто значения попадают в диапазоны.

```
[ ]  df_rest.total_bill.hist()
```

```
[ ]  df_rest.total_bill.hist(bins = 20)
```

Конечно, графики можно подписывать, менять параметры отображения и видоизменять их на своё усмотрение для дальнейшего использования в отчётах.

```
ax = df_rest.tip.hist(bins = 20)

ax.set_title('График распределение чаевых')
ax.set_ylabel('Количество заказов')
ax.set_xlabel('Сумма чаевых, USD')
```

Точечная диаграмма — это тип математической диаграммы, использующей декартовы координаты для отображения значений двух переменных для набора данных.

Данные отображаются в виде набора точек, каждая из которых имеет значение одной переменной, определяющей положение на горизонтальной оси, и значение другой переменной для вертикальной оси.

```
[ ]  axes = df_rest.plot.scatter("total_bill", "tip")
```

```
[ ]  axes = df_rest.plot.scatter("total_bill", "tip", c='DarkBlue')

     axes.set_title('Общая сумма счёта Vs Сумма чаевых')
     axes.set_xlabel('Сумма чека')
     axes.set_ylabel('Размер чаевых')
```

Seaborn — вторая по популярности библиотека визуализации после Matplotlib. Она расширяет функционал последней за счёт новых типов графиков и дополнительных параметров.

```
[ ]  import seaborn as sns
```

Метод jointplot() чем-то напоминает scatter, но помимо распределения параметров он визуализирует на одном

		графике гистограммы обоих признаков и показывает связь между ними. `[ ] sns.jointplot(x = 'total_bill', y = 'tip', data = df_rest, kind = 'reg')`	
Закрепление изученного материала	10 мин.	**Вопросы для обсуждения** • Для чего нам нужна библиотека Pandas? • С помощью чего можно визуализировать данные и составить о них первое впечатление?	Педагог организует беседу по вопросам
Этап подведения итогов занятия (рефлексия)	8 мин.	**Вопросы для обсуждения** • О чём был этот урок? • Какие вопросы остались? Что осталось непонятным?	Педагог способствует размышлению обучающихся над вопросами
Информация о домашнем задании, инструктаж по его применению	5 мин.	-	

Рекомендуемые ресурсы для дополнительного изучения:

1. ПИТОНТЬЮТОР. [Электронный ресурс] – Режим доступа: http://pythontutor.ru/.
2. Библиотека Pandas в Python. [Электронный ресурс] – Режим доступа: https://pythonim.ru/libraries/biblioteka-pandas-python.
3. Библиотека Matplotlib в Python. [Электронный ресурс] – Режим доступа: https://pythonim.ru/libraries/biblioteka-matplotlib-v-python.
4. Как строить графики. [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/otus/blog/540526/.