

Big Data and Data Science

Логинов Сергей

НФИМд-01-22

link: <https://stepik.org/lesson/584152/step/1?unit=578918>

Образовательная организация:

Русская школа программирования

Трудоемость:

26-28 часов

Блок SQL

Все примеры реализуются в БД Oracle.

В данном блоке изучаются

- Подзапросы в операторах SELECT, FROM, WHERE и их применение для решения практических задач
- Дополнительные операторы - ANY, ALL, IN, их использование в подзапросах
- СТЕ - табличные выражения, помогающие эффективнее работать при большом количестве подзапросов.
- Различные виды JOIN - INNER, LEFT, RIGHT
- Альтернативные виды объединения - UNION, UNION ALL, INTERSECT, MINUS
- Оконные функции - агрегатные и ранжирующие, смещения и аналитические
- Представления

Блок Big Data

В данном блоке представлены азы архитектуры MapReduce и файловой системы HDFS, а также Apache Hadoop.

Конспект первого видео

Большие данные - структурированные и неструктурированные данные в больших объемах.

Структурированные данные - имеют четкий формат, всегда имеют однозначное обращение к ним. Например, таблицы в БД.

Неструктурированные данные - более гибкий формат, в котором информации не представлена в однозначном виде. Например, тексты, картинки, аудио и т.д.

Суть больших данных выражена в модели 3V(VVV):

- Объем - обработка больших массивов информации, неформально от 1 Гбайт
- Скорость - быстрая генерация информации
- Разнообразие - данные в различных форматах поступают из разных источников и имеют разные сущности

Существует вертикальное и горизонтальное масштабирование

Вертикальное - увеличение и наращивание производительности

Горизонтальное - увеличение количества вычислительных центров

Технология MapReduce - разработка Google. Краткое введение.

В ее основе лежат процедуры функционального программирования:

- map - применение заданной функции к каждому элементу списка
- reduce - объединение результатов работы map

На основе данного подхода разработано множество продуктов для обработки больших данных:

- Apache Hadoop
- Apache CouchDB
- MongoDB

В действительности данная модель состоит из последовательной комбинации трех функций:

1. map - обрабатывает большое количество входных данных.

Главный узел (мастер) получает список данных, делит его на части для рабочих узлов.

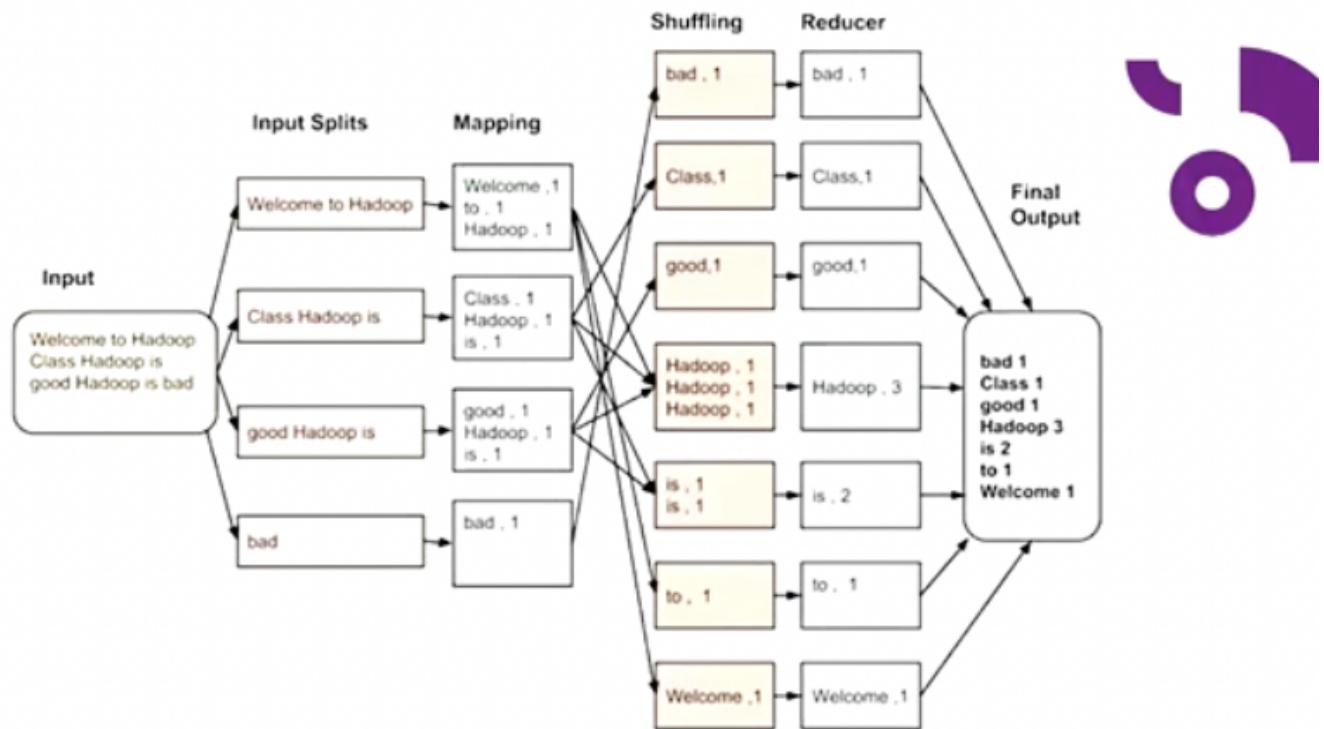
Далее рабочие узлы применяют функцию map на части данных и записывают результат в виде "ключ-значение"

2. shuffle - перераспределяет данные на основе ключей, созданных функцией map, данные одного ключа находятся в отдельном узле

3. reduce - параллельная обработка данных по порядку следования ключей и объединение результатов в мастере.

Мастер передает свободным узлам промежуточные результаты выполнения следующего шага. Результат = решение поставленной задачи

Примерная схема работы модели:



Распределенная файловая система HDFS

Данная файловая система, используемая в Hadoop, предназначена для хранения файлов больших размеров, которые блоками распределены на

разных узлах вычислительного кластера.

Файловая система имеет архитектуру master/slave. Она состоит из (утрировано):

NameNode - сервер, который управляет пространством имен файловой системы и регулирует клиентский доступ к файлам. Другие функции:

- Выполняет операции на открытие, закрытие, переименовывание файлов и каталогов
- Определяет отображение блоков в DataNodes
- Дает команды DataNodes на создание, удаление и репликацию блоков

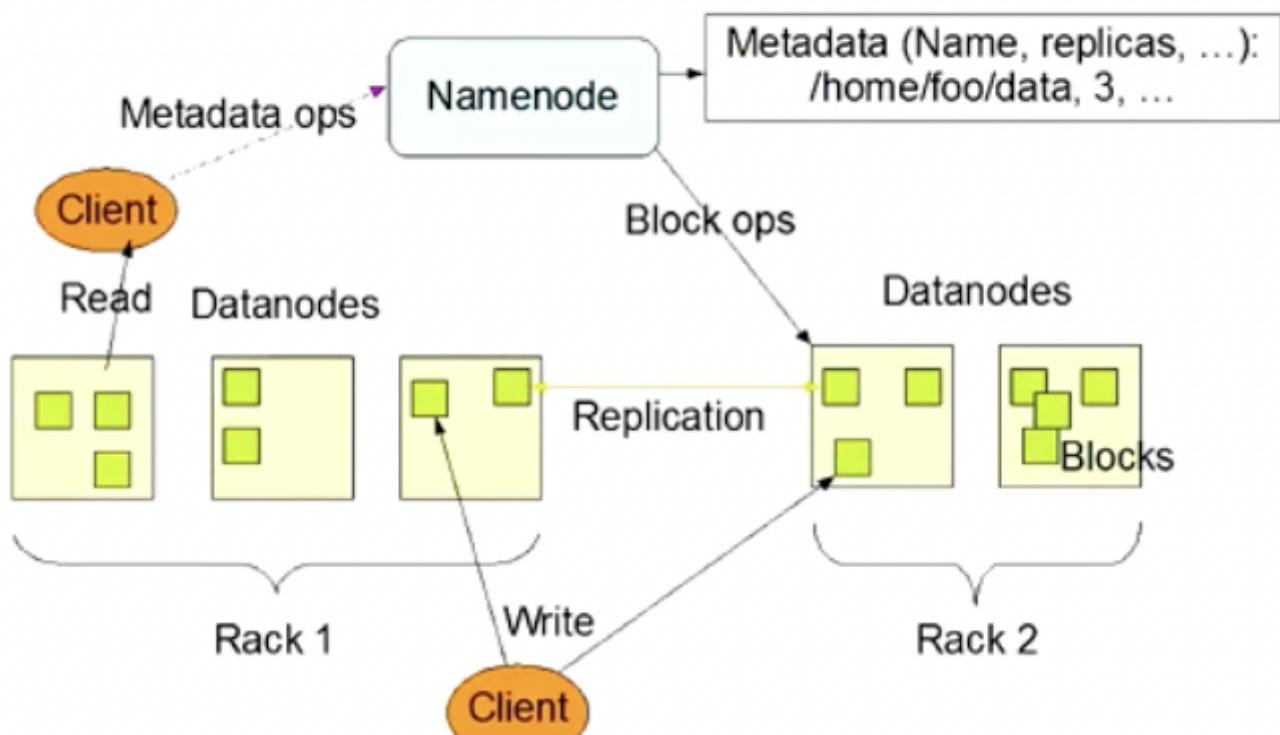
DataNodes - управляют хранилищем данных, хранят внутри блоки данных.

Другие функции:

- Обычно их несколько и по одному на каждом узле кластера
 - Отвечает за обслуживание запросов чтение/запись с клиентской файловой системы
- Client** - пользователь и приложение, взаимодействующий через API с файловой системой

Схема:

HDFS Architecture



Что такое Hadoop ?

Apache Hadoop - набор библиотек, утилит и фреймворк для работы с большим объемом данных.

Его основные модули:

- Hadoop Common - набор утилит и библиотек, используемых, в том числе, для управления распределенными файлами
- Hadoop MapReduce - платформа для программирования и выполнения MapReduce-вычислений
- YARN - система планирования заданий и управления кластером
- HDFS - распределенная файловая система

Полезные ссылки:

<https://stepik.org/course/150/promo> - Бесплатный курс по Hadoop

<http://hadoop.apache.org/> - Официальный сайт Apache Hadoop

<https://www.bigdataschool.ru/wiki/mapreduce> - Что такое MapReduce

<https://medium.com/edureka/hadoop-tutorial-24c48fbf62f6> - Туториал по экосистеме Hadoop

<https://medium.com/xnewdata/do-you-know-what-is-hadoop-mapreduce-technology-ed3b341401c7> - Серия статей по Hadoop. Очень полезно для более глубокого погружения

Инструменты обработки, анализа и визуализации данных

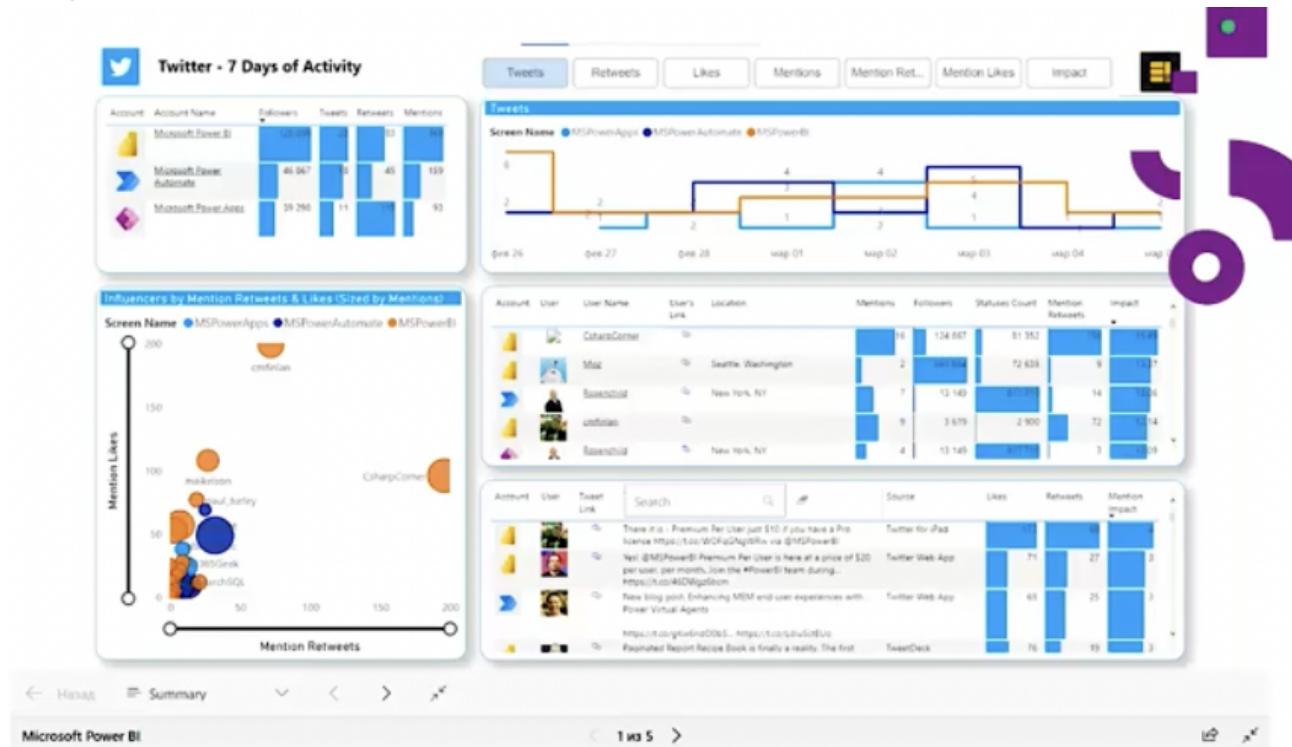
Анализ данных в Power BI

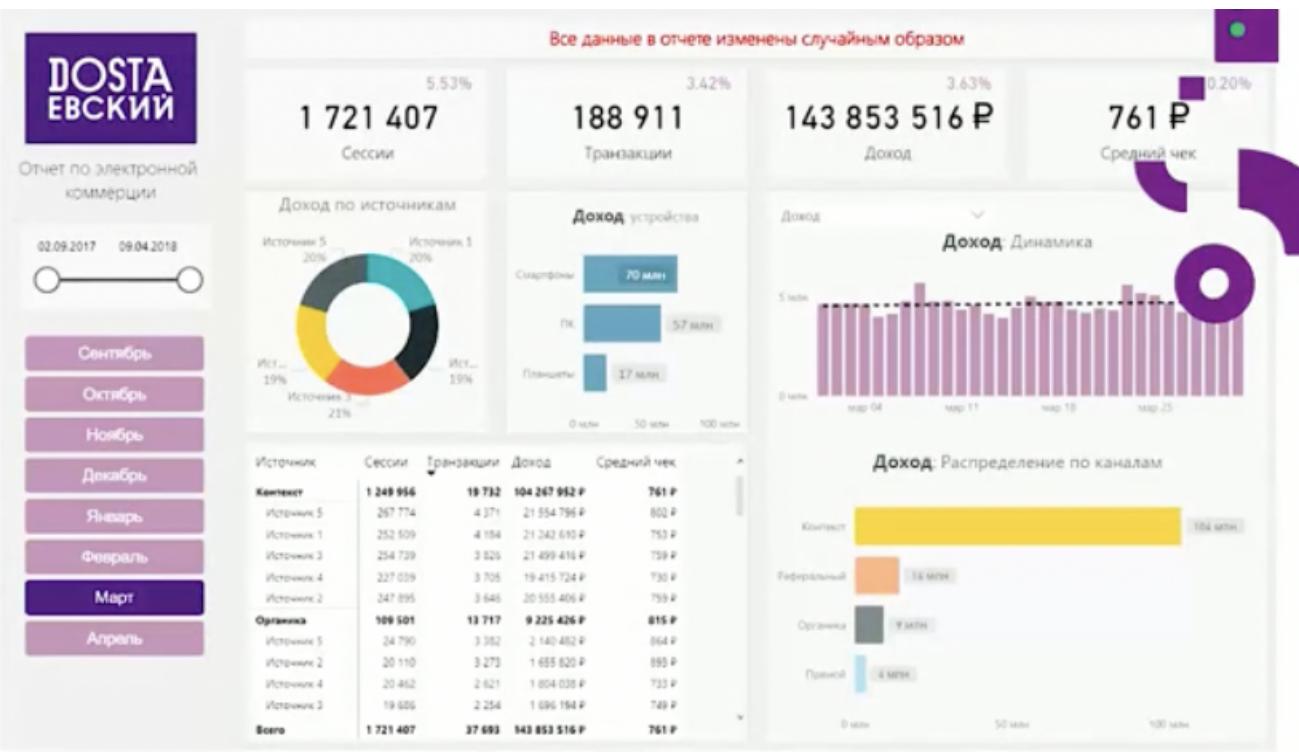
Power BI - набор продуктов от Microsoft для бизнес-аналитики - изменения в компаниях через определение потребностей и рекомендацию решений, которые обеспечивают пользу для стейкхолдеров (заинтересованных сторон).

Назначение Power BI

- Получение данных
- Визуализация
- Предоставление доступа к данным

Результат работы в Power BI - отчет. Его примеры:





Плюсы Power BI:

- Большое комьюнити
- Лидер отрасли
- Встроенные коннекторы
- Простота использования
- Сходство с Excel
- Набор визуализаций

Power BI Desktop

Возможности:

- Подключение к данным
- Преобразование и очистка данных
- Создание визуальных элементов
- Создание отчетов
- Предоставление доступа к отчетам

Компоненты:

- Power Query - извлечение, загрузка, преобразование

- Power Pivot – моделирование
- Power View – визуализация

Power Pivot использует "язык" DAX – Data Analysis eXpressions. С помощью них создаются:

- Меры – вычисляются во время использования в визуализации
- Вычисляемые столбцы – вычисляются при создании

Категории функций DAX:

- Фильтры
- Математические
- Статистические
- Логические
- Даты и времени
- Текстовые
- Информационные

Пример DAX:

Год добавления фильма на Netflix:

```
--date_add_year = year('netflix_data'[date_added])
```

Кол-во лет от создания фильма до
добавления на Netflix:

```
--date_diff_add_release =
'netflix_data'[_date_add_year_]-
'netflix_data'[release_year]
```

Примеры мер:



Среднее кол-во оценивших:

```
!avg_num_votes = AVERAGE(imdb_data[num_votes])
```

Средний рейтинг:

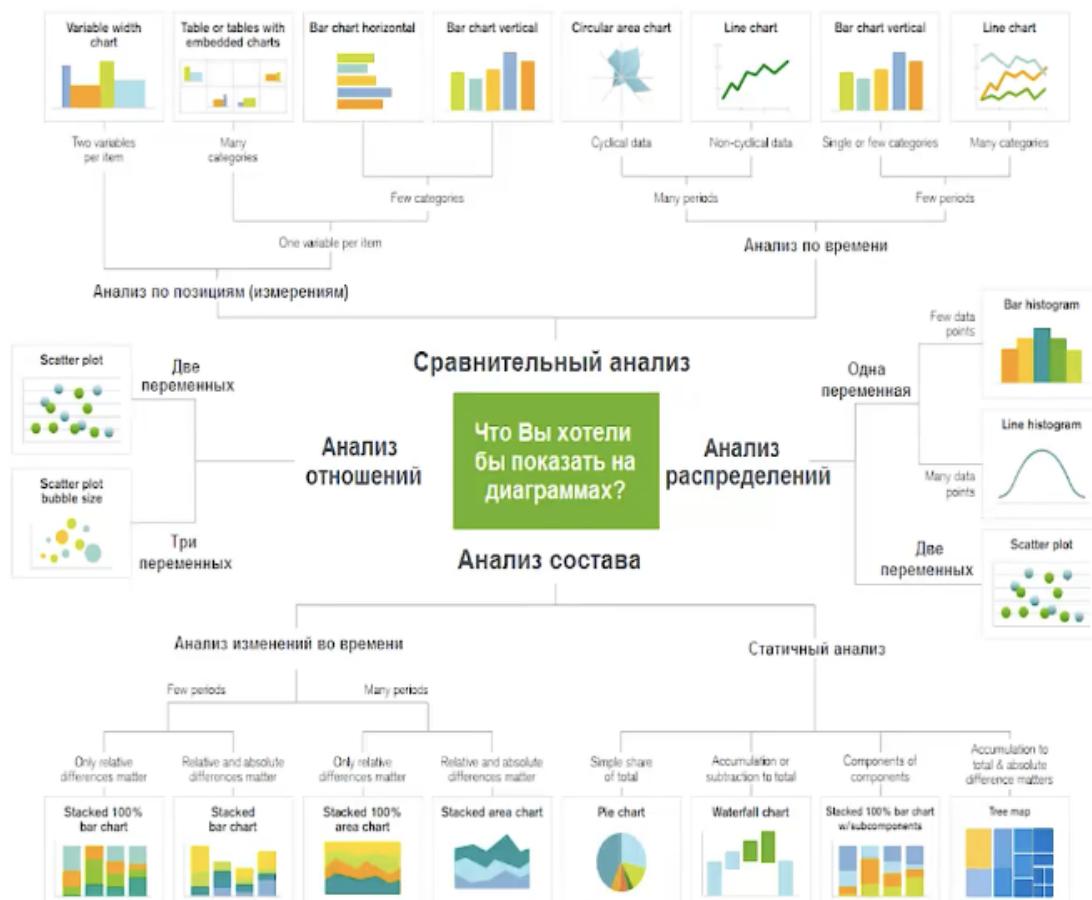
```
!avg_rating = AVERAGE(imdb_data[average_rating])
```

Средний кол-во лет до добавления на Netflix:

```
!avg_year_before_add =  
AVERAGE(netflix_data[_date_diff_add_release])
```



Подходы к выбору визуализации:



Полезные ссылки:

Данные

Netflix: https://github.com/NastyaSNK/pbi_lecture/raw/master/netflix_data.csv

Данные IMDB: https://github.com/NastyaSNK/pbi_lecture/raw/master/imdb_data.csv

Файл с итоговым

результатом: https://github.com/NastyaSNK/pbi_lecture/raw/master/imdb_data.pbix

Любопытные источники на русском:

[Как Microsoft спрятала целый сервер и как его найти / Хабр \(habr.com\)](#) - Статья на Хабре о внутренностях PowerBI

[Документация по Power BI - Power BI | Microsoft Docs](#) - Документация Microsoft

[PowerBIBook.ru — документация PowerBIBook.ru](#) - Адаптированный учебник

[Power BI в Microsoft Learn | Microsoft Docs](#) - Обучение Microsoft

[Блог NeedForData, обучающие статьи по Power BI](#) - Блог о PowerBI и Excel

[Telegram: Contact @PBI_Rus](#) - Телеграм-канал для вопросов и обсуждений по теме Power BI

<https://github.com/needforadata/PublicSharables.pdf> - Краткая методичка по применению визуализаций для Power BI

Инструменты работы с большими данными

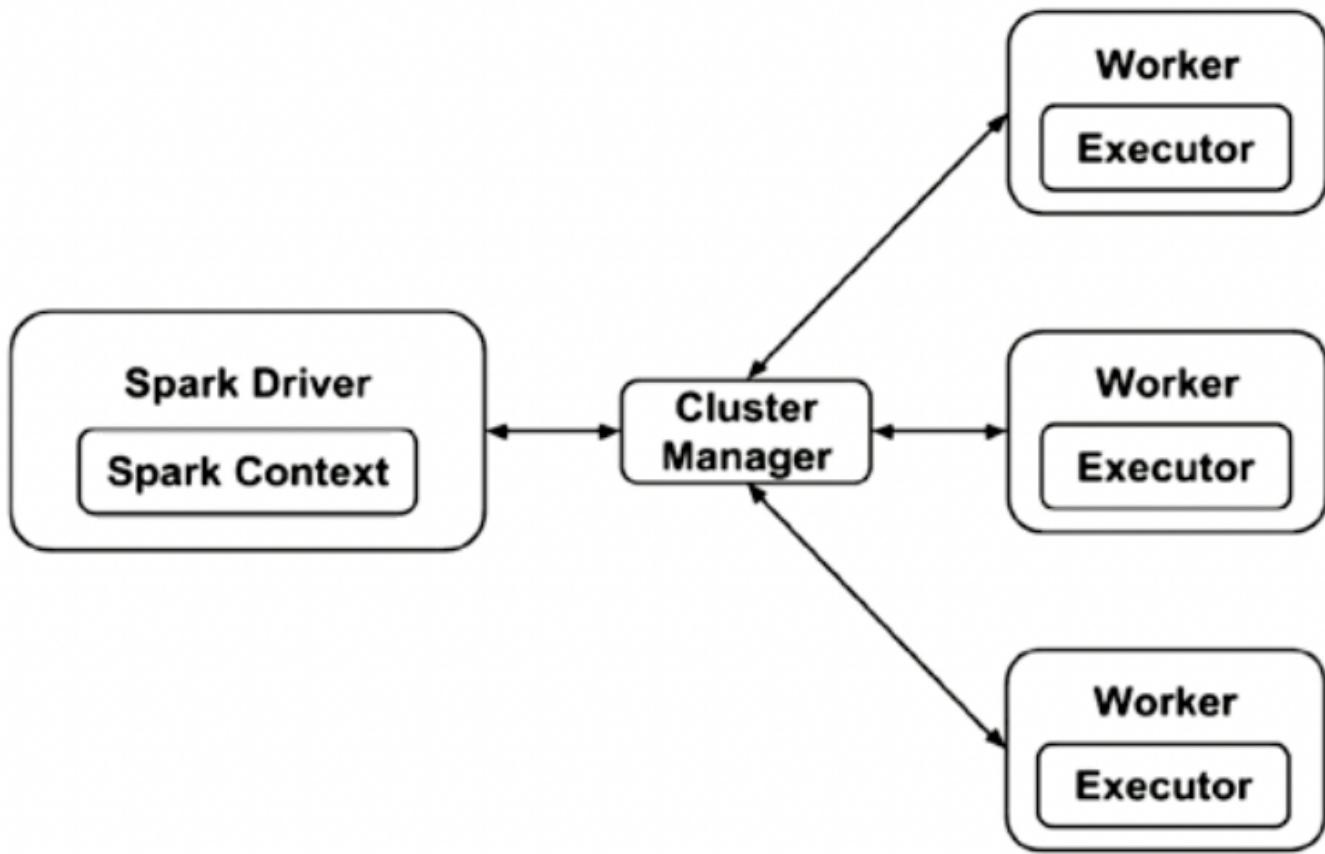
Фреймворк Apache Spark

pandas - работа с большими данным, которые помещаются в оперативную память

Spark - работа с большими данными, которые не помещаются в оперативную память

Особенность Spark - ленивые вычисления. Все манипуляции с данными подразделяются на два общих - трансформации и действия.

Схема работы:



Основой является Driver. Он запускает функцию main и разделяет приложение на задачи.

Задачи передаются исполнителям (executor), каждый из них работает над своей задачей.

Для управления кластером существует Cluster Manager, их существует несколько типов

Основные форматы хранение данных:

		
.xlsx	.csv	.json
Не открывается в текстовых редакторах и может быть зашифрован паролем	Открывается любым текстовым редактором и читаем	Открывается любым текстовым редактором и легко читаем
Потребляет много памяти	Потребляет мало памяти	Потребляет много памяти
Поддерживает вложенность	Не поддерживает вложенность	Поддерживает вложенность
Плохая масштабируемость	Плохая масштабируемость	Хорошая масштабируемость

В Spark используется особенный способ хранения данных: Parquet

- Записать один раз, считывать по потребности
- Загружаются только необходимые данные, а не весь файл
- Позволяет использовать вложенный формат данных
- Хранится по столбцам



Row Storage

Last Name	First Name	E-mail	Phone #	Street Address

Columnar Storage

Last Name	First Name	E-mail	Phone #	Street Address

Системы машинного обучения

Статистика

A/B тесты

Идея:

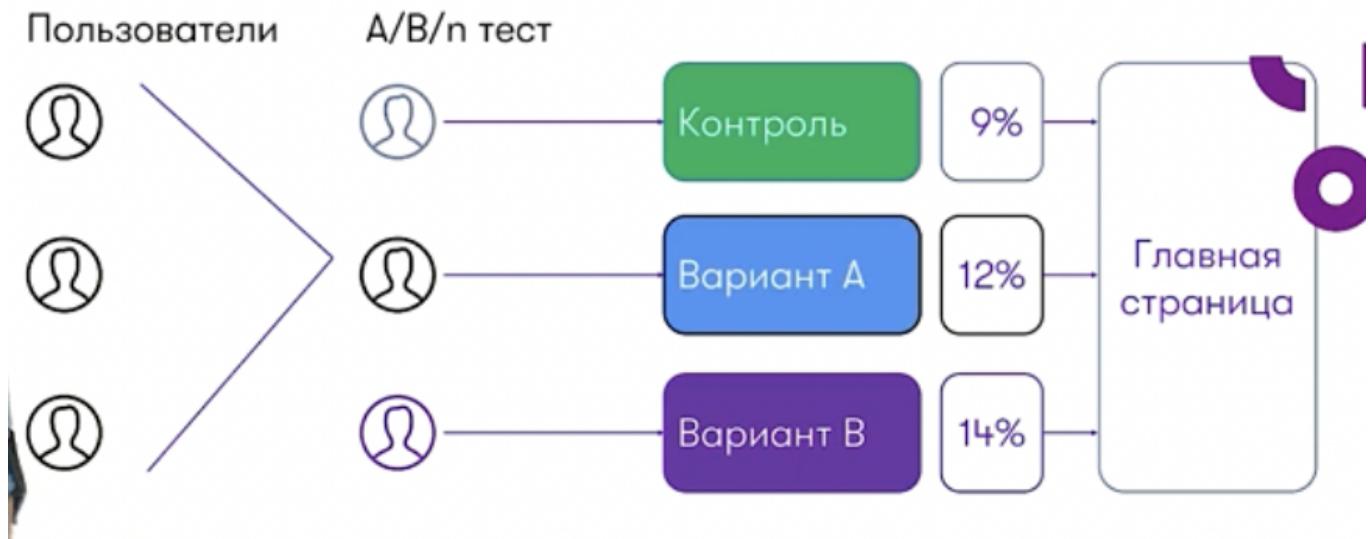


Требования к тестам:

- Понимание метрики/цели

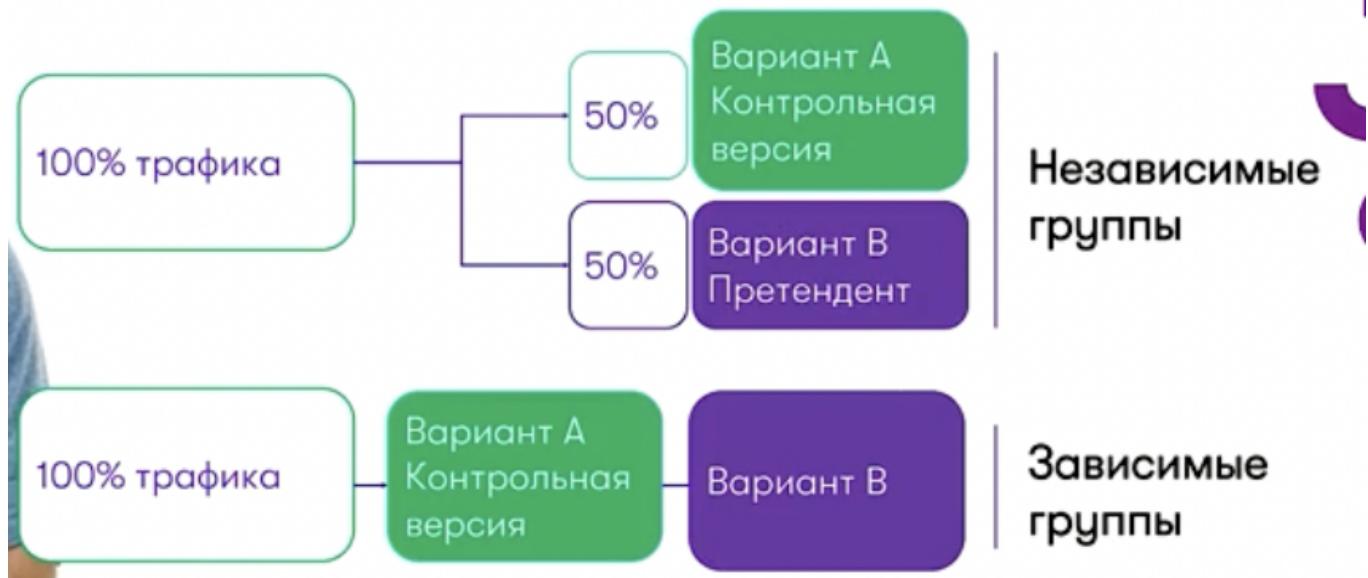
- Одновременность
- Случайность
- Достаточный объем выборки
- Независимость

A/B тестирование не ограничивается делением на 2 группы, существуют Multi A/B тесты:



Если нет деления на группы или группы зависимы, то в таком случае также можно применять тестирование, но измененное:

А/В тесты при зависимых группах

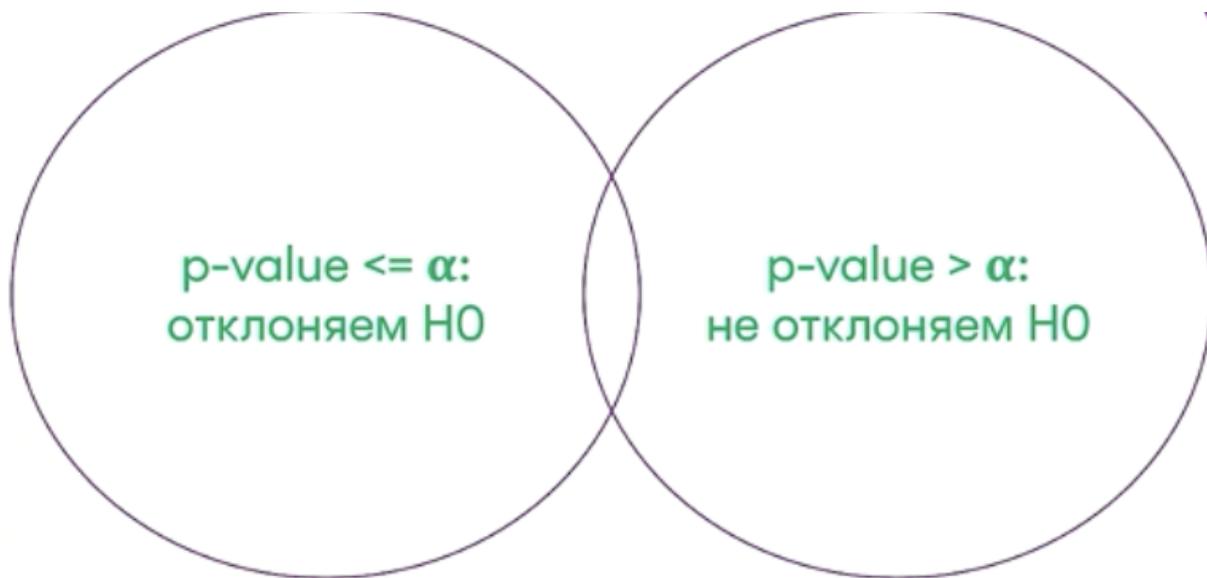


Основная задача тестов - статистическая проверка выгодных и полезных гипотез.

Основные понятия:

p-value - вероятность ошибки при условии, что эффект случайный;
вероятность ошибочно заметить различия между контрольной и тестовой группой

Гипотеза - наше предположение



p - value - это вероятность ошибки,
при условии, что эффект случайный

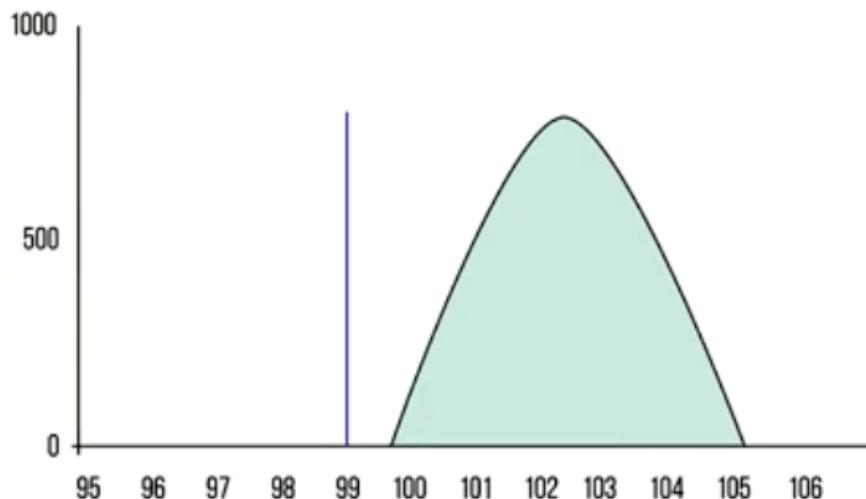
Как принять решение?

Сравнить p-value с альфа (обычно берется 5%)

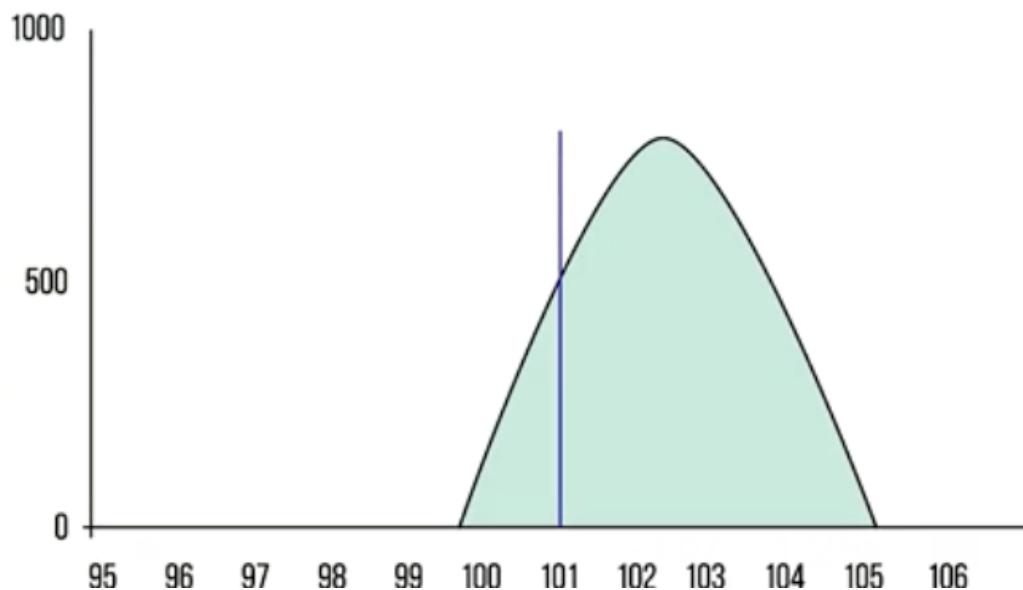
контрольной группы



p-value < 5%



p-value > 5%



p-value получаем с помощью статистических критериев. Основная цель критерия - дать величину возможной ошибки.



Сравнение двух выборок За исключением номинальных данных

Нормальное распределение
(только количественные)

Распределение ненормальное
(количественные или
категориальные)

t-тест Стьюдента

Зависимые/
Независимые

Категориальные

Критерий
Уилкоксона

U-
критерий
Манна-
Уитни

Критерий
Фишера/Пирсона

t-test/Критерий Стьюдента

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Легко интерпретируемый
критерий.
Мы просто смотрим на
различие **средних**

Критерий Вилкоксона и Манна-Уитни

Используйте для проверки между
двумя выборками (зависимыми или независимыми)

Невозможно отклонить НО:
распределение выборок равны.
Отклонить НО: распределения
выборок не равны

Критерий Вилкоксона для независимых выборок называется критерием
Манна-Уитни

В отличие от критерия Стьюдента, критерий Манна-Уитни не требует
нормального распределения, но является трудно интерпретируемым.

Дисперсионный анализ – для сравнения большого количества групп.
Идея: сравнение дисперсии внутри группы и дисперсии между группами.
Исходя из этого вычисляются оценки того, насколько группы отличаются.
Вычисляется F-значение Фишера.

Машинное обучение в бизнесе

В лекции рассказывается о том, что на данный момент машинное обучение и
нейросети нельзя назвать полноценной научной дисциплиной. Они часто и,
что важно, очень успешно применяются в бизнесе, однако нет
фундаментальных теорий об их работе.

В этой области существует два лагеря - сторонники научного подхода, и сторонники бизнес-подхода. Первые хотят глубоко разобраться в теме и все подробно изучить, вторые же считают, что рабочие конструкции/решения, если они будут полезны, нужно брать и применять, не вдаваясь в детали. Для примера взят google-переводчик. Для перевода с точки зрения науки очень сложно реализовать какие-то модели и алгоритмы. Также сложно описать перевод формулами и логическими выражениями. Заниматься формализацией всех правил разных языков долго и дорого. Поэтому здесь используется рекуррентная нейросеть, которая обучается на десятках миллионов пар вида "слово-перевод". Да, такой вариант переводчика не будет 100% точным, однако его реализация во много раз дешевле, а польза не намного меньше.

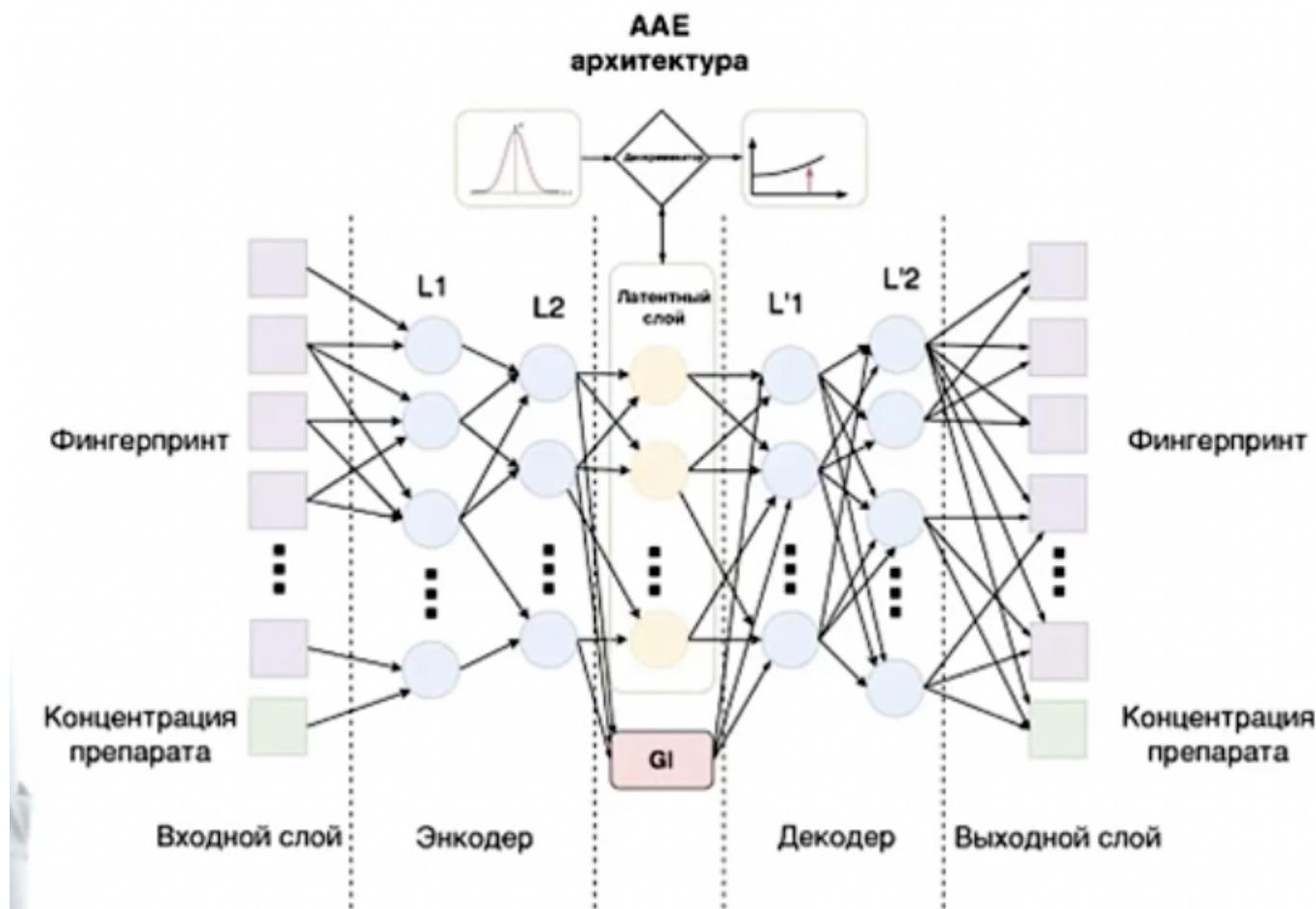
Работу таких нейросетей сложно полностью интерпретировать, а также сложно логически описать их работу после обучения.

Благодаря развитию технологий и многократному снижению стоимости комплектующих и компьютеров в целом, нейросети и машинное обучение переживают третью и самую большую волну популярности.

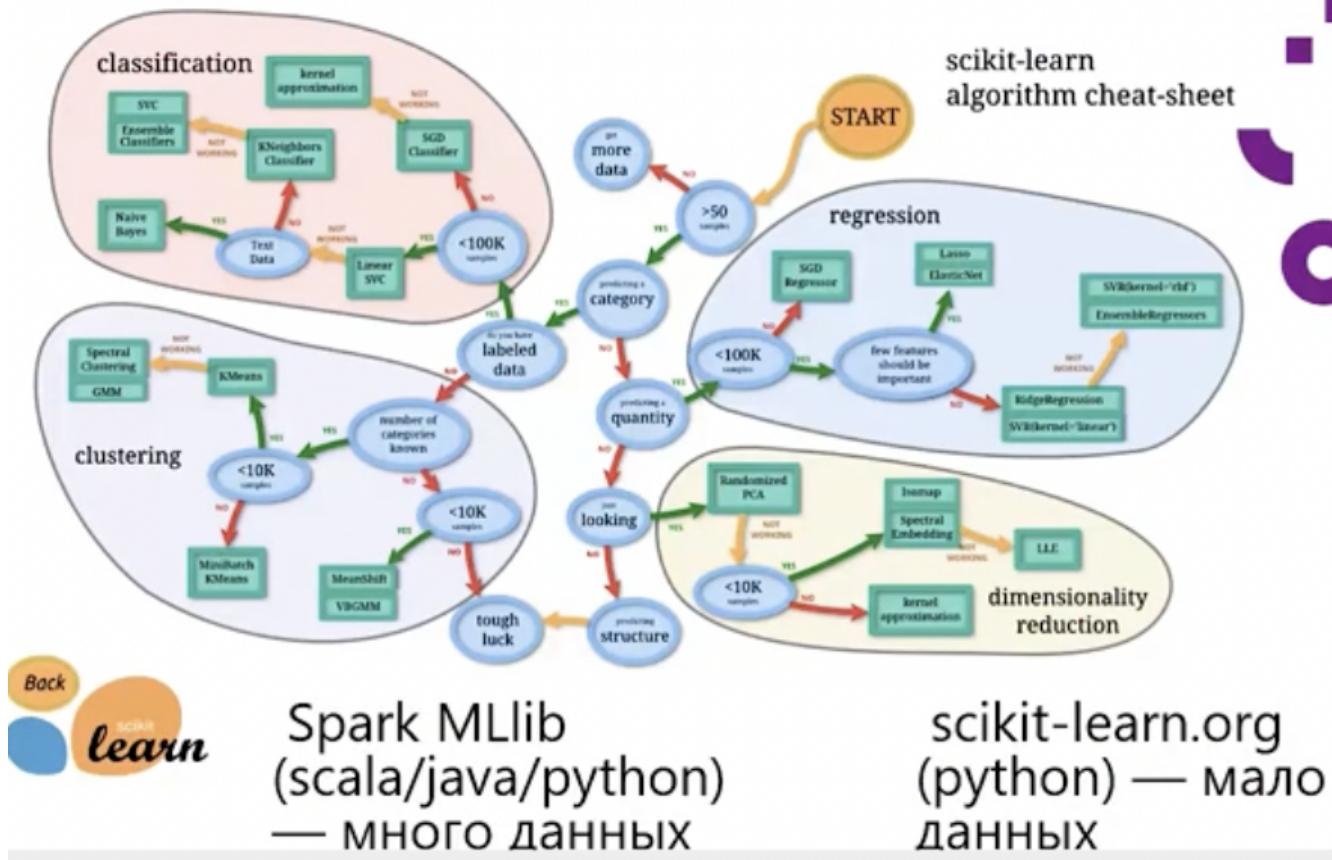
В данный момент имеются нейросети для самых различных задач. Например, для генерации интерьеров (генеративная), восстановления части изображений и т. д.

Пример нейросети для поиска способа борьбы с онкологией:

Борьба с заболеваниями



Некоторые полезные библиотеки/фреймворки:



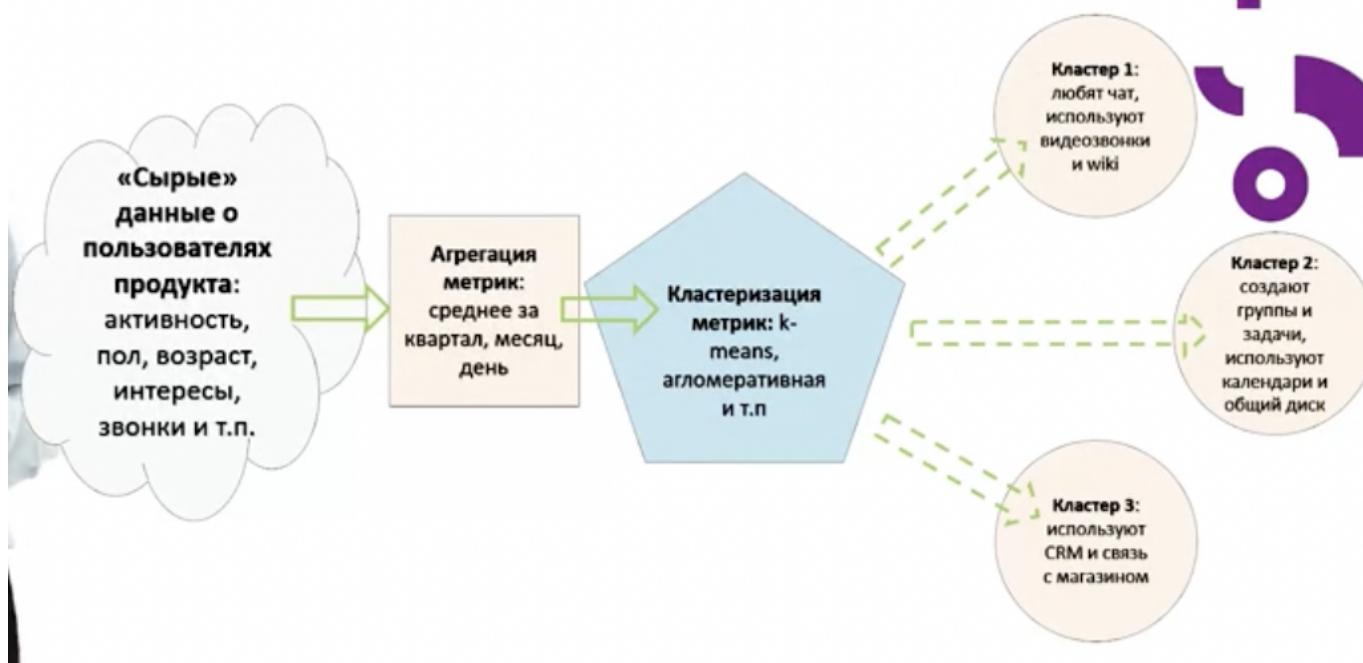
"Никогда не пытайтесь в рабочее время за деньги инвесторов повторить какой-то алгоритм с нуля"

При внедрении машинного обучения в компании начать стоит с простой аналитики - базовая статистика, крі и т.д.

Также стоит добавить визуализацию результатов, но она должна быть полезной.

После этого можно переходить к более сложным вещам, например, к кластерному анализу. Пример его использования:

Алгоритм использования кластеризации в продукте-1



Алгоритм использования кластеризации в продукте-2



Алгоритм использования кластеризации в продукте-3



Кластерный анализ: бизнес-кейсы

Сегментация клиентов, типов
использования сервиса, ...

Кластеризация «общего» товарного
каталога

Кластеризация графа связей сайтов
(пересечение аудитории)

Маркетинг работает с целевыми
группами, информация разбита на
«смысловые облака»

На следующем этапе можно внедрять персонализацию (похожа на
кластеризацию, но работает на размеченных данных) и классификацию. О ней

подробнее.

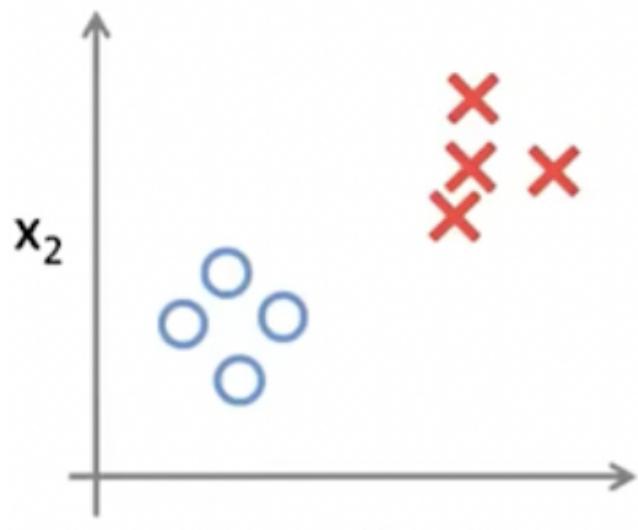
Классификация

Разбиваем по группам. Обучение

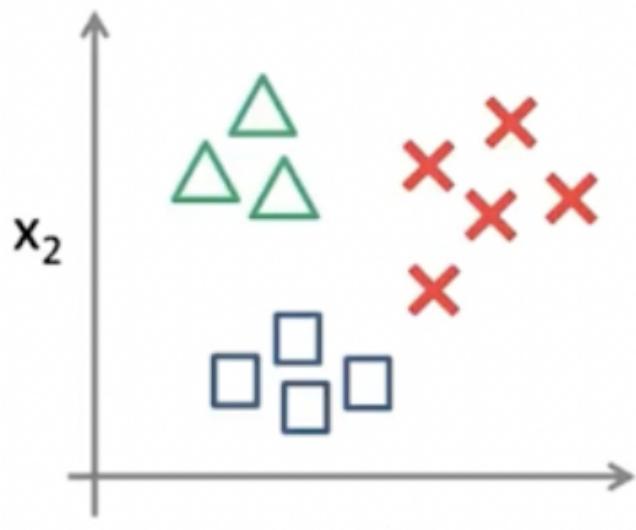
Бинарная

Мультиклассовая

Binary classification:



Multi-class classification:



Классификация: бизнес-кейсы

Удержание: найти клиентов, которые скоро уйдут (churn-rate)

Найти клиентов, готовых стать платными

Найти клиентов, которые готовы купить новую услугу

Найти готовых уволиться

Определить у клиента пол

Совет - попробовать работать с градиентным бустингом вместо более привычных/старых алгоритмов.

Как запустить проект ML-проект в облаке?

Главный совет - воспользоваться готовым решением. Например, Amazon Machine Learning

Сервис "Amazon Machine Learning"

- Только логистическая регрессия
- Работа через API
- Простой язык для feature engineering
- Простая визуализация и контроль качества классификаторов (binary, multiclass) и регрессии
- Масштабирование

Ссылка на сертификат:

<https://stepik.org/cert/1767781>



РУССКАЯ
ШКОЛА
ПРОГРАММИРОВАНИЯ

Настоящий сертификат подтверждает, что

Sergey Loginov

успешно завершил/а курс

Big Data и Data Science: перейди на новый уровень

<https://stepik.org/course/101689> | <https://stepik.org/cert/1767781>

13.11.2022

Анализ данных (Введение в Python и обработку таблиц)

Логинов Сергей НФИмд-01-22

Сылка на курс: <https://stepik.org/course/126333?search=1440407514>

Образовательная организация: ТюмГУ

Трудоемкость: 7 часов

В данном курсе не выдается сертификат от платформы, но имеется возможность получить удостоверение от Тюменского государственного университета. Информация по теме из курса:

Уважаемые слушатели курса!

Вы можете бесплатно получить **удостоверение государственного образца о повышении квалификации** по курсу «Анализ данных» от Тюменского государственного университета!

Пройдите регистрацию до 1 декабря 2022 года, и получите удостоверение уже в этом году.

[Ссылка на регистрацию](#)

Вам понадобятся документы:

- паспорт;
- снилс;
- документ об образовании;
- подписанное вами согласие на обработку персональных данных ([форма Согласия на обработку ПД](#))

Сохранено

× разработки Google Colab. Она похожа на jupyter, но использует облачные вычисления, а значит не требует установки каких-либо программ или пакетов и не ест ресурсы компьютера. Есть интеграция с Google диском, что тоже удобно.

Основы обработки таблиц с данными на языке Python

```
import pandas as pd
```

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/students.csv',  
                 delimiter=',')
```

Повторим вычисление статистических характеристик.

Для начала некоторые функции суммарной статистики по всему датасету

```
df.head()
```

	Age	Growth	Shoe size	Course number	Year of birth	Friend number	Russian rating	Maths rating	Physics rating	Computer science rating
0	20	170	40	2	2002	100	85	86	0	8
1	22	191	43	7	2000	307	69	74	64	7
2	21	172	41	4	2000	186	78	62	0	
3	19	168	38	2	2003	604	72	0	0	
4	23	159	36	1	1998	144	0	0	0	

5 rows × 48 columns

```
df.describe()
```

	Age	Growth	Shoe size	Course number	Year of birth	Friend number	F
count	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000	186.000000
mean	20.688172	170.661290	39.715054	3.500000	1991.107527	374.032258	81
std	3.468713	9.055881	2.814920	3.147285	132.099550	2199.541507	14
min	17.000000	153.000000	34.000000	0.000000	200.000000	0.000000	0
25%	19.000000	164.000000	38.000000	2.000000	2000.000000	74.500000	76
50%	20.000000	169.000000	39.000000	3.000000	2001.500000	130.000000	85
75%	21.000000	178.000000	42.000000	4.000000	2003.000000	236.000000	91
max	55.000000	197.000000	48.000000	37.000000	2004.000000	30000.000000	100

8 rows × 33 columns

```
# df.value_counts() – вычисляет значения, но выглядит громоздко,  
# просто закомментируем и запомним
```

```
df['Growth'].mean() # средний рост = математическое ожидание
```

170.66129032258064

```
df['Growth'].median() # медиана
```

169.0

```
df.Growth.std() # СКО
```

9.055880959226972

Далее в курсе проводится введение в фильтрацию/селекцию данных в датасете, получение слайсов и т.д. Тема простая и заранее изученная, идет как базовая, поэтому считаю, что не обязательно полностью отражать ее в отчете. Однако, для повторения или если встретятся спорные или неизвестные моменты, они будут зафиксированы ниже

```
# базовая селекция как образец  
df[df.Sex == 'женский'].head()
```

	Age	Growth	Shoe size	Course number	Year of birth	Friend number	Russian rating	Maths rating	Physics rating	Comput scien rati
0	20	170	40	2	2002	100	85	86	0	0
3	19	168	38	2	2003	604	72	0	0	0
4	23	159	36	1	1998	144	0	0	0	0
9	22	168	38	6	2000	297	71	5	0	0
10	20	158	35	4	2001	1000	85	72	0	0

5 rows × 48 columns

```
df[(df.Growth > 165) & (df.Growth < 175)].head() # двойная
```

	Age	Growth	Shoe size	Course number	Year of birth	Friend number	Russian rating	Maths rating	Physics rating	Comput scien rati
0	20	170	40	2	2002	100	85	86	0	0
2	21	172	41	4	2000	186	78	62	0	0
3	19	168	38	2	2003	604	72	0	0	0
9	22	168	38	6	2000	297	71	5	0	0
12	21	169	39	4	2001	150	68	27	0	0

5 rows × 48 columns

```
df[df.Age + df['Year of birth'] == 2022].shape # количество людей,  
# которые отпраздновали  
# день рождения в этом году
```

(102, 48)

```
df[df.Age == df.Age.max()]
```

	Age	Growth	Shoe size	Course number	Year of birth	Friend number	Russian rating	Maths rating	Physics rating	Comput scien rati
81	55	160	37	7	1960	2000	0	0	0	0

1 rows × 48 columns

Влияние строк и столбцов друг на друга

Группировка, сортировка и корреляция

```
df_cut = df[['Age', 'Growth', 'Weight']]  
df_cut.head()
```

	Age	Growth	Weight
0	20	170	64.0
1	22	191	73.0
2	21	172	60.0
3	19	168	59.0
4	23	159	57.0

Сортировка

```
df_cut.sort_values(by='Age', ascending=False)
```

	Age	Growth	Weight
81	55	160	3.0
63	35	180	78.0
99	31	165	53.0
173	29	176	70.0
82	27	168	58.0
...
127	18	165	46.0
61	18	182	75.0
159	18	168	70.0
115	17	175	72.0
126	17	168	71.0

186 rows × 3 columns

```
# множественная сортировка  
df_cut.sort_values(by=['Age', 'Growth'], ascending=[False, True])
```

	Age	Growth	Weight
81	55	160	3.0
63	35	180	78.0
99	31	165	53.0
173	29	176	70.0
82	27	168	58.0
...
112	18	183	65.0
16	18	185	64.0
42	18	185	68.0
126	17	168	71.0
115	17	175	72.0

186 rows × 3 columns

```
df_sorted = df_cut.sort_values(by='Growth', ascending=False)
```

Обращение по индексам и слайсы

```
df_sorted.iloc[0, ]
```

```
Age      19.0  
Growth   197.0  
Weight   100.0  
Name: 103, dtype: float64
```

Добавление столбцов в датасет

```
df_cut['BMI'] = 10000 * df_cut['Weight'] / (df_cut['Growth'] * df_cut['Growth'])
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/entries/ipython.html>

```
df_cut.head()
```

	Age	Growth	Weight	BMI
0	20	170	64.0	22.145329
1	22	191	73.0	20.010416
2	21	172	60.0	20.281233
3	19	168	59.0	20.904195
4	23	159	57.0	22.546576

Коэффициент корреляции

```
df_cut.corr()
```

	Age	Growth	Weight	BMI
Age	1.000000	-0.101982	-0.185722	-0.200305
Growth	-0.101982	1.000000	0.544528	0.146201
Weight	-0.185722	0.544528	1.000000	0.904678
BMI	-0.200305	0.146201	0.904678	1.000000

Группировка

```
df.groupby('Sex')[['Growth', 'Weight']].mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning:  
    """Entry point for launching an IPython kernel.
```

Sex	Growth	Weight
женский	165.725000	59.105263
мужской	179.636364	71.580645

```
df.groupby(['Sex', 'Chocolate'])['Russian rating', 'Maths rating'].mean()

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning
"""\Entry point for launching an IPython kernel.
```

Sex	Chocolate	Russian rating	Maths rating
женский	M&Ms	79.400000	51.600000
	Баунти	81.181818	56.318182
	КитКат	85.782609	61.782609
	Марс	91.166667	36.666667
	Милки Вей	79.583333	45.250000
	Натс	84.636364	53.818182
	Скиттлс (хотя это и не шоколадка)	86.000000	67.000000
	Сникерс	80.375000	61.562500
	Твикс	86.478261	55.217391
	M&Ms	83.666667	76.666667
мужской	Баунти	78.375000	68.000000
	КитКат	74.416667	51.083333
	Марс	85.600000	55.400000
	Милки Вей	72.800000	61.800000
	Натс	70.500000	72.000000
	Скиттлс (хотя это и не шоколадка)	84.000000	77.000000
	Сникерс	77.666667	59.055556
	Твикс	79.818182	75.000000

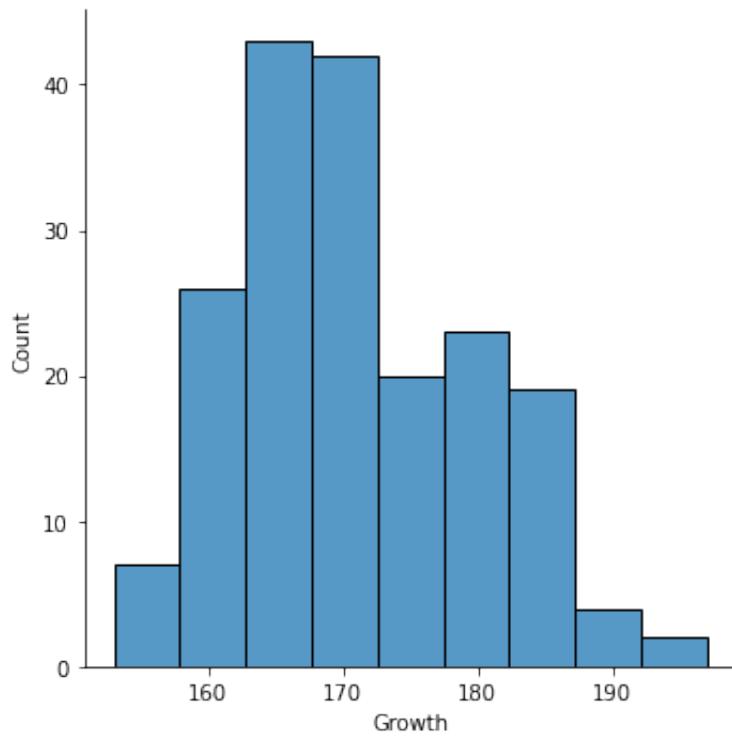
▼ Базовые операции с данными

Визуализация данных

```
import seaborn as sns
```

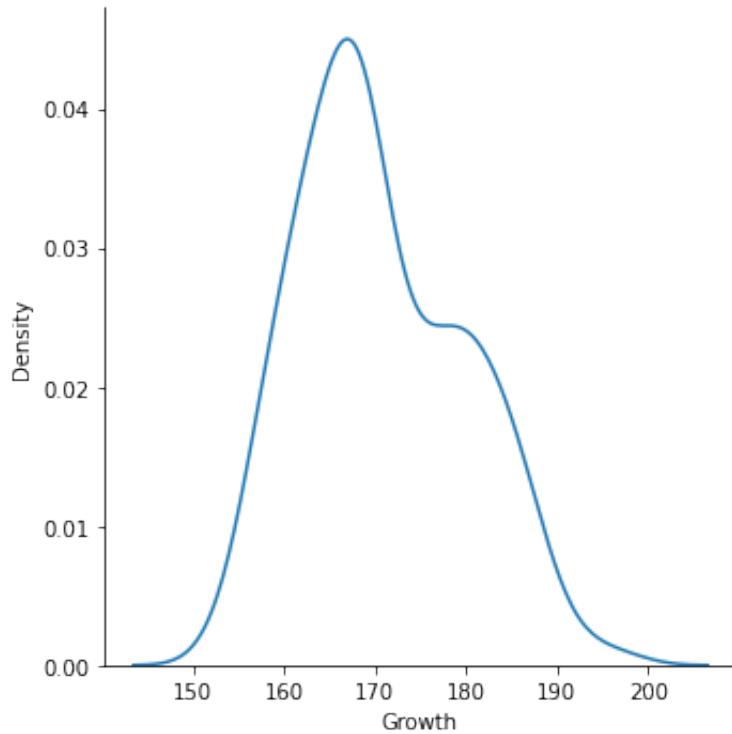
```
# гистограмма распределения  
sns.displot(data=df, x='Growth')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f9acb806c10>
```



```
sns.displot(data=df, x='Growth', kind='kde')
```

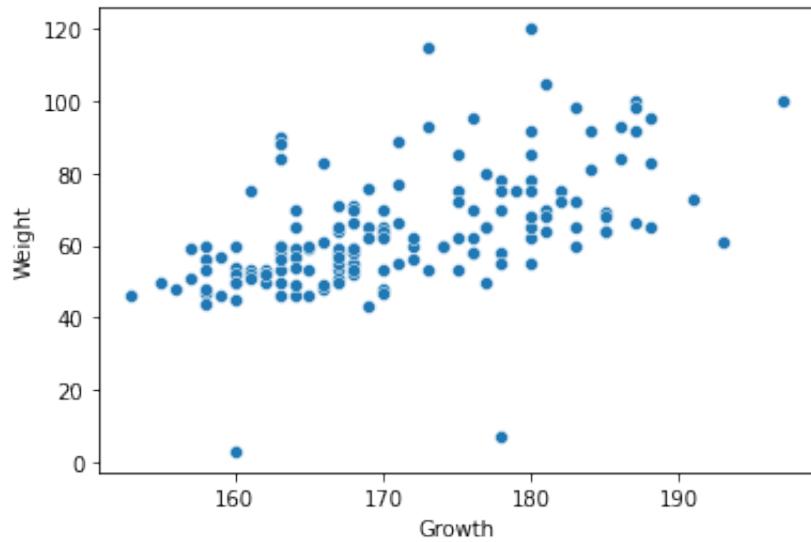
```
<seaborn.axisgrid.FacetGrid at 0x7f9acb162b90>
```



```
# точечный график
```

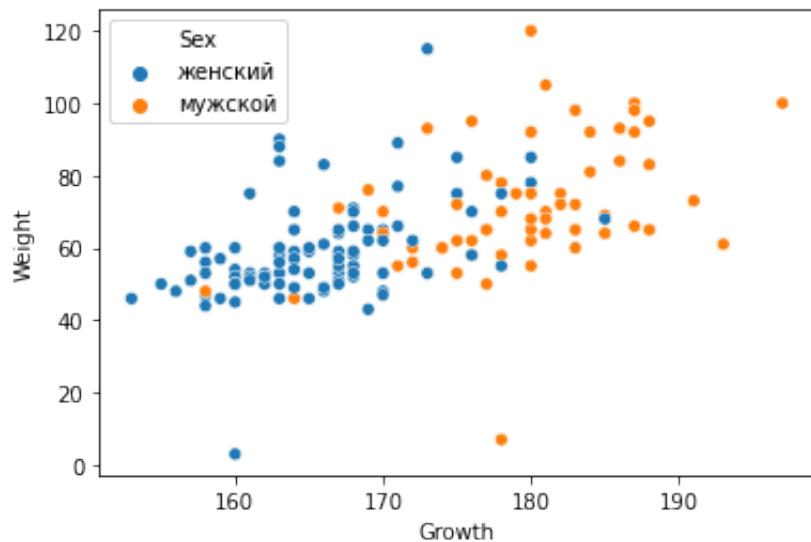
```
sns.scatterplot(data=df, x='Growth', y='Weight')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac7b45810>
```



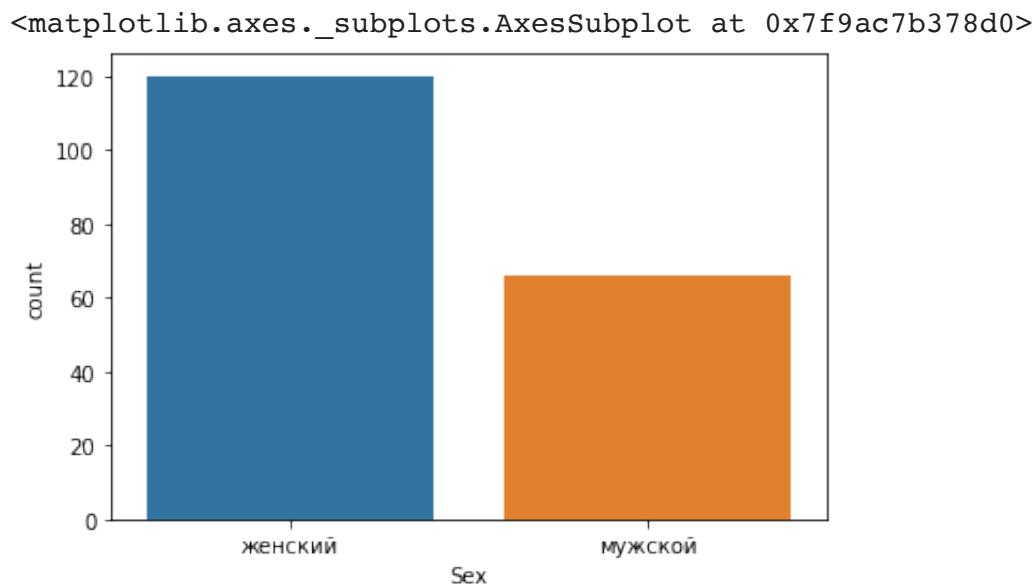
```
sns.scatterplot(data=df, x='Growth', y='Weight', hue='Sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac7ac0110>
```

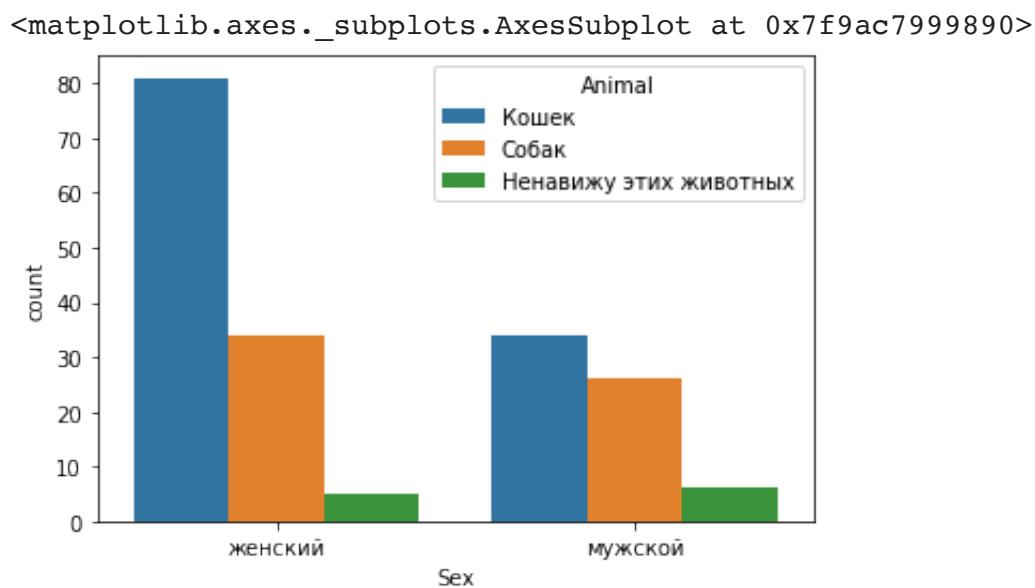


Визуализация категориальных признаков

```
sns.countplot(x=df['Sex']) # or data=df, x='Sex'
```



```
sns.countplot(data=df, x='Sex', hue='Animal')
```



Построение нескольких графиков

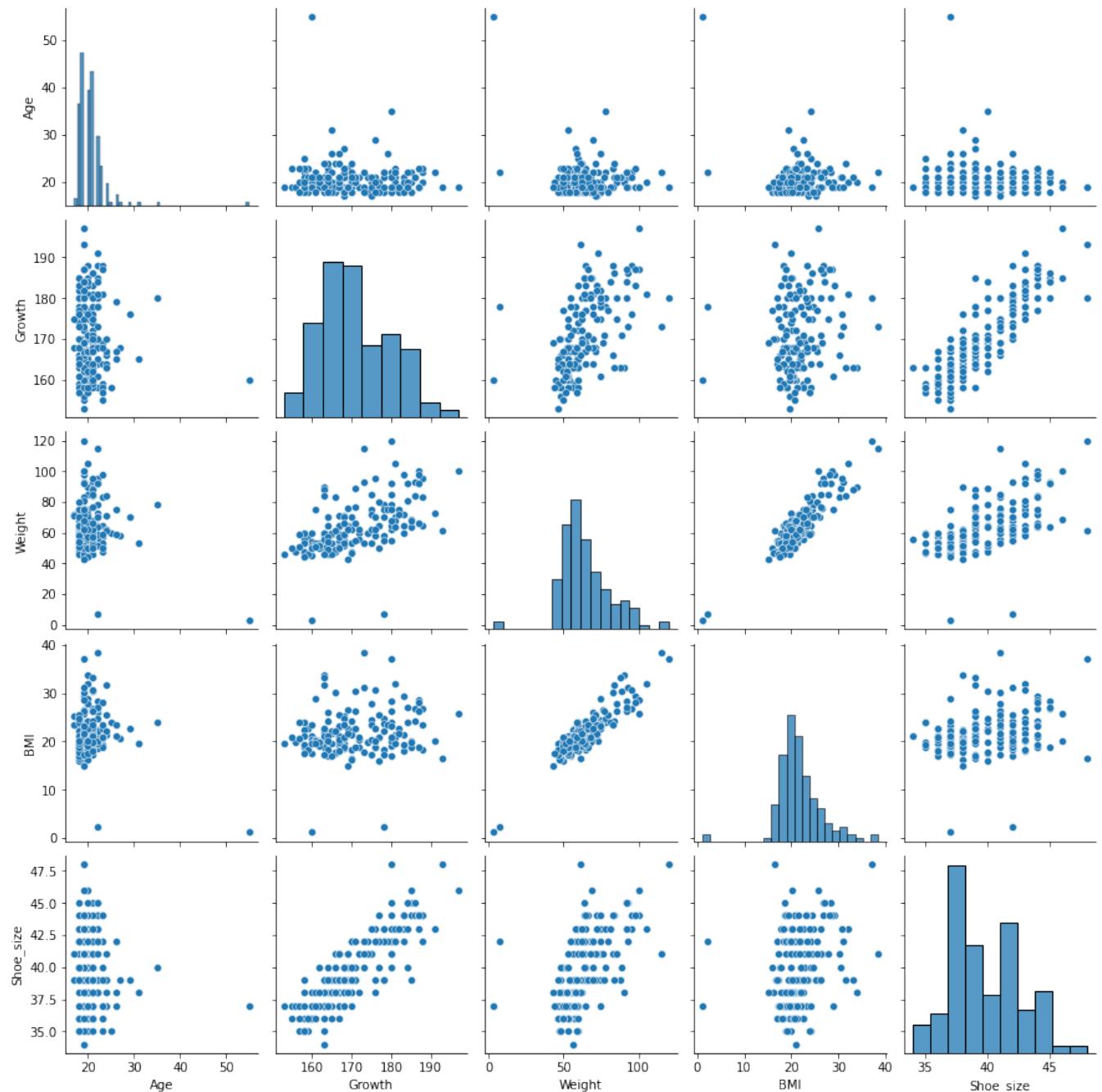
```
df_cut['Shoe_size'] = df['Shoe size']
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/using_hands_on/entry_point.html

```
sns.pairplot(df_cut)
```

<seaborn.axisgrid.PairGrid at 0x7f9ac79176d0>



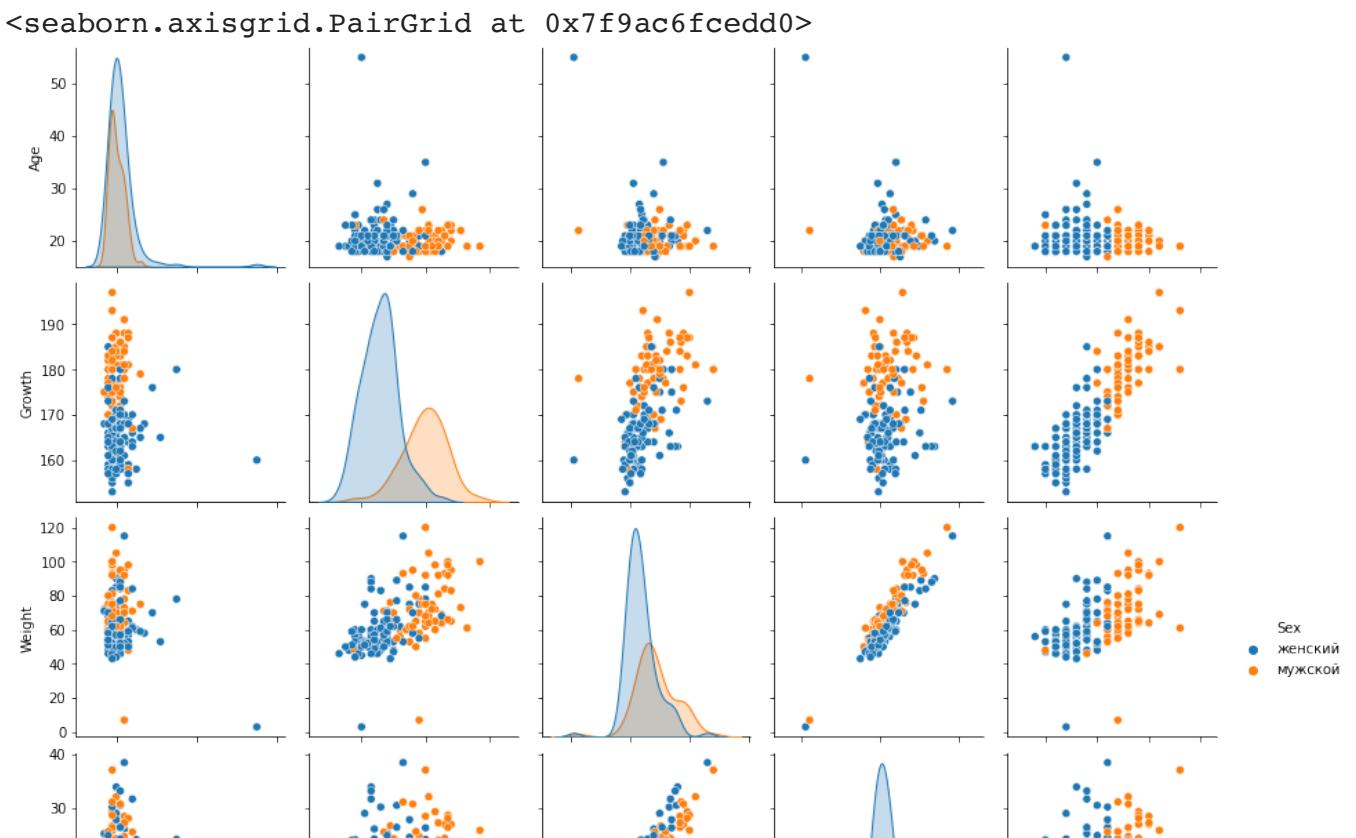
Здесь по диагонали находится гистограмма признака, а в остальных ячейках - парные точечные диаграммы

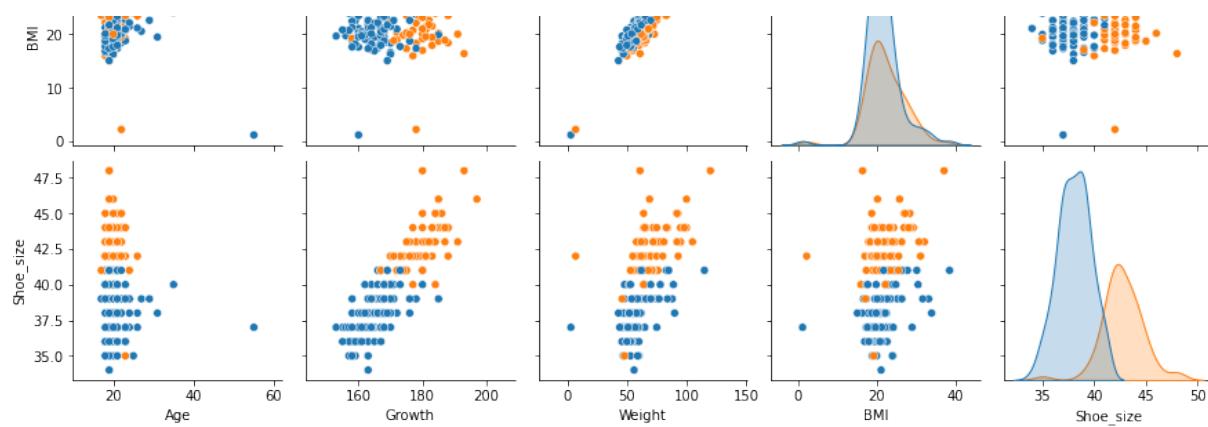
```
df_cut['Sex'] = df.Sex
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWith
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs>
"""\nEntry point for launching an IPython kernel.

```
sns.pairplot(df_cut, hue='Sex')
```





Работа с пропусками в данных и отсутствующими значениями

```
# Простое удаление пропусков
```

```
df1 = df.dropna()
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 118 entries, 0 to 185
Data columns (total 48 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Age              118 non-null    int64  
 1   Growth            118 non-null    int64  
 2   Shoe size         118 non-null    int64  
 3   Course number    118 non-null    int64  
 4   Year of birth   118 non-null    int64  
 5   Friend number   118 non-null    int64  
 6   Russian rating  118 non-null    int64  
 7   Maths rating    118 non-null    int64  
 8   Physics rating  118 non-null    int64  
 9   Computer science rating  118 non-null    int64
```

```
9 Computer science rating    118 non-null   int64
10 Chemistry rating        118 non-null   int64
11 Literature rating       118 non-null   int64
12 History rating          118 non-null   int64
13 Geography rating        118 non-null   int64
14 Biology rating          118 non-null   int64
15 Foreign language rating 118 non-null   int64
16 Social science rating    118 non-null   int64
17 Distance to home km     118 non-null   int64
18 Minutes to first class  118 non-null   int64
19 Children number          118 non-null   float64
20 Removed teeth            118 non-null   float64
21 Weight                  118 non-null   float64
22 Glasses                 118 non-null   object
23 Sex                      118 non-null   object
24 Problems in last semester 118 non-null   object
25 Coin                     118 non-null   object
26 Rock paper scissors     118 non-null   object
27 Animal                  118 non-null   object
28 Month of birthday       118 non-null   int64
29 Your rating in university 118 non-null   object
30 Fastfood                118 non-null   object
31 Height of 5000 mm        118 non-null   int64
32 Width of 5000 mm         118 non-null   int64
33 Putin age                118 non-null   int64
34 Army                     118 non-null   object
35 Hostel                   118 non-null   object
36 Hair length              118 non-null   float64
37 Floor number             118 non-null   int64
38 Social network duration min 118 non-null   int64
39 Chocolate                118 non-null   object
40 City population           118 non-null   float64
41 Strange people            118 non-null   object
42 Your insitute            118 non-null   object
43 Brother-sister            118 non-null   object
44 Plane seat                118 non-null   object
45 Middle and index finger  118 non-null   int64
46 Middle and ring finger   118 non-null   float64
47 Middle and little finger 118 non-null   float64
dtypes: float64(7), int64(26), object(15)
memory usage: 45.2+ KB
```

Удалили много теоретически полезной информации, лучше воспользоваться заменой пропусков

```
df.isnull().sum()
```

Age	0
Growth	0
Shoe size	0
Course number	0
Year of birth	0
Friend number	0
Russian rating	0
Maths rating	0
Physics rating	0
Computer science rating	0
Chemistry rating	0
Literature rating	0
History rating	0
Geography rating	0
Biology rating	0
Foreign language rating	0
Social science rating	0
Distance to home km	0
Minutes to first class	0
Children number	36
Removed teeth	38
Weight	29
Glasses	2
Sex	0
Problems in last semester	0
Coin	0
Rock paper scissors	0
Animal	0
Month of birthday	0
Your rating in university	0
Fastfood	0
Height of 5000 mm	0
Width of 5000 mm	0
Putin age	0
Army	0
Hostel	0
Hair length	0
Floor number	0
Social network duration min	0
Chocolate	0
City population	0
Strange people	0
Your insitute	0
Brother-sister	0
Plane seat	0
Middle and index finger	0
Middle and ring finger	0
Middle and little finger	0
dtype:	int64

```
df2 = df.fillna(0)
```

```
df2.info()
```

<class 'pandas.core.frame.DataFrame'>			
RangeIndex: 186 entries, 0 to 185			
Data columns (total 48 columns):			
#	Column	Non-Null Count	Dtype
0	Age	186 non-null	int64
1	Growth	186 non-null	int64
2	Shoe size	186 non-null	int64
3	Course number	186 non-null	int64
4	Year of birth	186 non-null	int64
5	Friend number	186 non-null	int64
6	Russian rating	186 non-null	int64
7	Maths rating	186 non-null	int64
8	Physics rating	186 non-null	int64
9	Computer science rating	186 non-null	int64
10	Chemistry rating	186 non-null	int64
11	Literature rating	186 non-null	int64
12	History rating	186 non-null	int64
13	Geography rating	186 non-null	int64
14	Biology rating	186 non-null	int64
15	Foreign language rating	186 non-null	int64
16	Social science rating	186 non-null	int64
17	Distance to home km	186 non-null	int64
18	Minutes to first class	186 non-null	int64
19	Children number	186 non-null	float64
20	Removed teeth	186 non-null	float64
21	Weight	186 non-null	float64
22	Glasses	186 non-null	object
23	Sex	186 non-null	object
24	Problems in last semester	186 non-null	object
25	Coin	186 non-null	object
26	Rock paper scissors	186 non-null	object
27	Animal	186 non-null	object
28	Month of birthday	186 non-null	int64
29	Your rating in university	186 non-null	object
30	Fastfood	186 non-null	object
31	Height of 5000 mm	186 non-null	int64
32	Width of 5000 mm	186 non-null	int64
33	Putin age	186 non-null	int64
34	Army	186 non-null	object
35	Hostel	186 non-null	object
36	Hair length	186 non-null	float64
37	Floor number	186 non-null	int64
38	Social network duration min	186 non-null	int64
39	Chocolate	186 non-null	object
40	City population	186 non-null	float64
41	Strange people	186 non-null	object
42	Your insitute	186 non-null	object
43	Brother-sister	186 non-null	object
44	Plane seat	186 non-null	object
45	Middle and index finger	186 non-null	int64

```
46 Middle and ring finger      186 non-null    float64
47 Middle and little finger   186 non-null    float64
dtypes: float64(7), int64(26), object(15)
memory usage: 69.9+ KB
```

Можно также заменить на медиану/среднее значение и тд

```
df2.isnull().sum()
```

Age	0
Growth	0
Shoe size	0
Course number	0
Year of birth	0
Friend number	0
Russian rating	0
Maths rating	0
Physics rating	0
Computer science rating	0
Chemistry rating	0
Literature rating	0
History rating	0
Geography rating	0
Biology rating	0
Foreign language rating	0
Social science rating	0
Distance to home km	0
Minutes to first class	0
Children number	0
Removed teeth	0
Weight	0
Glasses	0
Sex	0
Problems in last semester	0
Coin	0
Rock paper scissors	0
Animal	0
Month of birthday	0
Your rating in university	0
Fastfood	0
Height of 5000 mm	0
Width of 5000 mm	0
Putin age	0
Army	0
Hostel	0
Hair length	0
Floor number	0
Social network duration min	0
Chocolate	0
City population	0
Strange people	0
Your insitute	0
Brother-sister	0
Plane seat	0
Middle and index finger	0
Middle and ring finger	0
Middle and little finger	0
dtype: int64	

Замена по столбцу

```
df['Weight'].fillna(df['Weight'].mean())  
  
0      64.0  
1      73.0  
2      60.0  
3      59.0  
4      57.0  
     ...  
181     59.0  
182     43.0  
183     51.0  
184     62.0  
185     54.0  
Name: Weight, Length: 186, dtype: float64
```

Замена пропусков в категориальных признаках

```
df['Glasses'] = df['Glasses'].fillna('да')
```

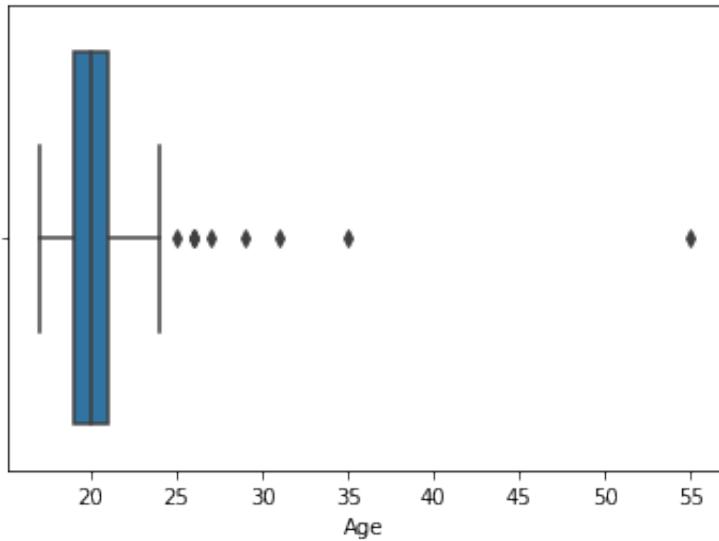
Выбросы и аномалии

Очень часто их можно найти при парном анализе двух признаков

Можно проверить с помощью визуализации (box plot)

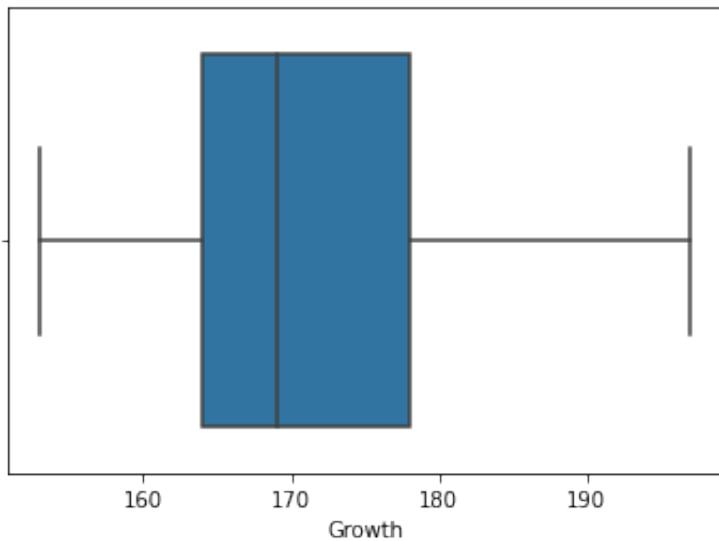
```
sns.boxplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning  
  FutureWarning  
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac4a8d510>
```



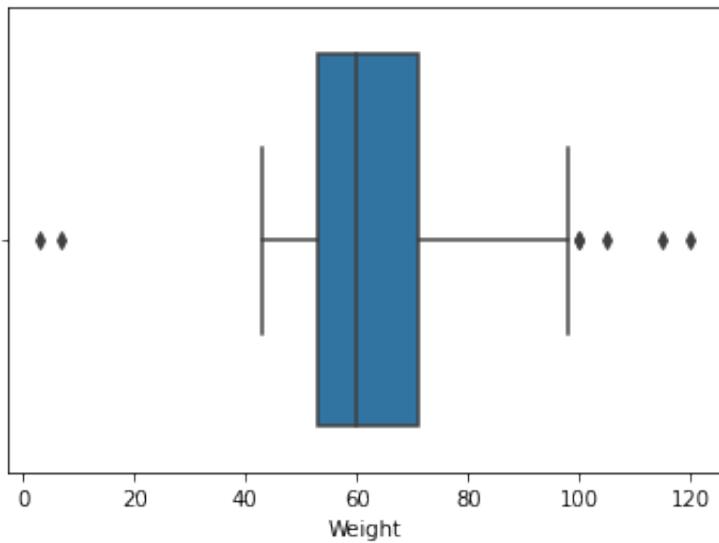
```
sns.boxplot(df['Growth'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning  
  FutureWarning  
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac4a3e450>
```



```
sns.boxplot(df['Weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning  
  FutureWarning  
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac49979d0>
```



1 метод удаления выбросов

Найти среднее значение m

Найти СКО s

Построить интервал $m-3s$; $m+3s$, все наблюдения за пределами интервала удалить из таблицы

То есть используется простейшее правило трех сигм, которое покрывает 99% значений

```
m = df['Weight'].mean()
```

```
s = df['Weight'].std()
```

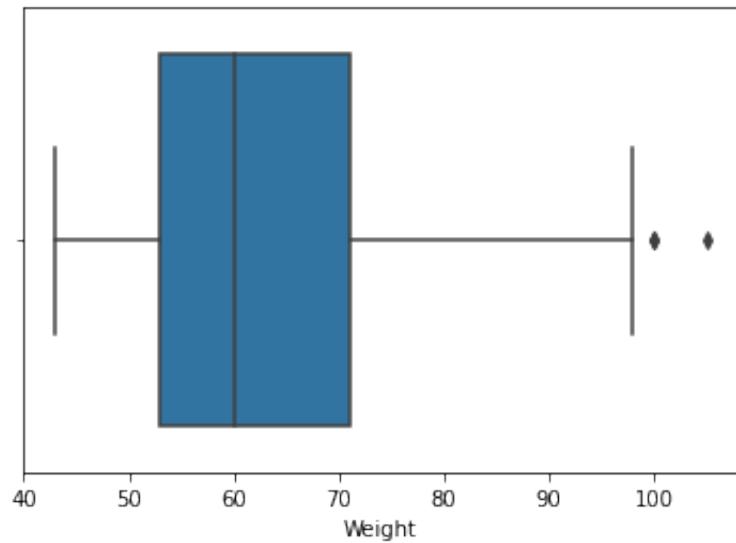
```
m - 3 * s, m + 3 * s
```

```
(14.52494740231539, 113.53874686520054)
```

```
df = df[(df['Weight'] > m - 3 * s) & (df['Weight'] < m + 3 * s)]
```

```
sns.boxplot(df['Weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning  
  FutureWarning  
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac4916190>
```



2 способ

a - 25 процентиль

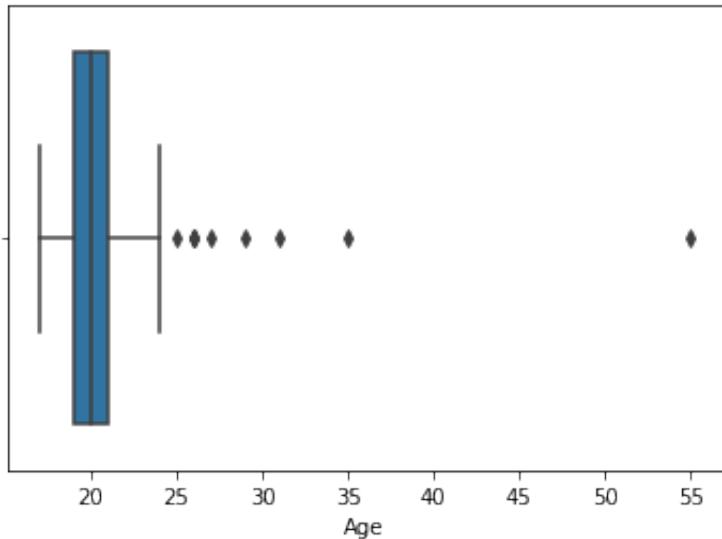
b - 75 процентиль

Строится интервал вида $a - 1.5(b - a); b + 1.5(b - a)$

Наблюдения за пределами также удаляются

```
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/students.csv',
                 delimiter=',')
sns.boxplot(df.Age)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac4a25910>
```

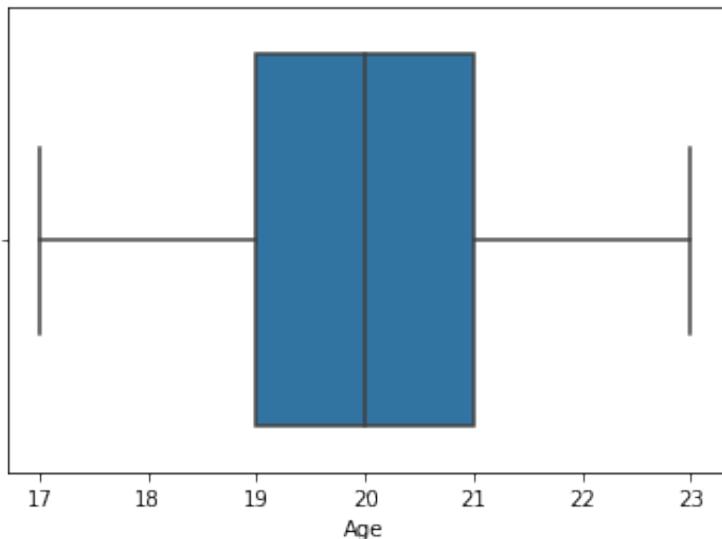


```
a = df['Age'].quantile(0.25)
b = df['Age'].quantile(0.75)
```

```
df = df[(df['Age'] > a - 1.5 * (b - a)) & (df['Age'] < b + 1.5 * (b - a))]
```

```
sns.boxplot(df['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning  
  FutureWarning  
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac486cf50>
```



Кластеризация

```
from sklearn.cluster import KMeans
```

```
df_cut = df[['Weight', 'Growth', 'Sex']]
```

```
df_cut = df_cut.dropna()
```

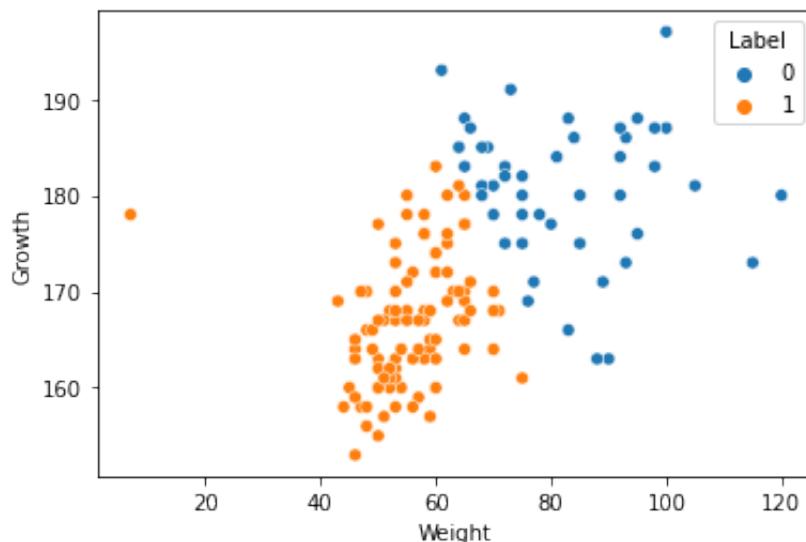
```
kmeans = KMeans(n_clusters=2, random_state=0)  
kmeans.fit(df_cut[['Weight', 'Growth']])  
df_cut['Label'] = kmeans.labels_
```

```
df_cut.head()
```

	Weight	Growth	Sex	Label
0	64.0	170	женский	1
1	73.0	191	мужской	0
2	60.0	172	мужской	1
3	59.0	168	женский	1
4	57.0	159	женский	1

```
sns.scatterplot(data=df_cut, x='Weight', y='Growth', hue='Label')
```

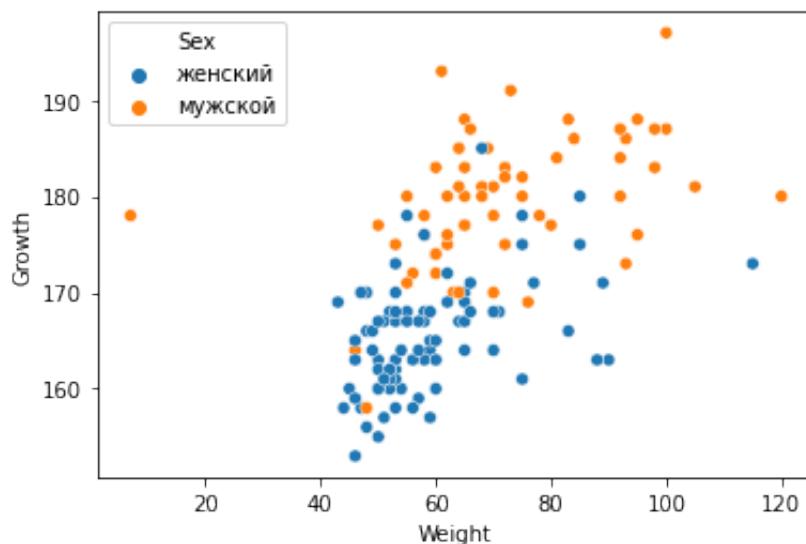
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac4240490>
```



Сравним с разбиением по полу

```
sns.scatterplot(data=df_cut, x='Weight', y='Growth', hue='Sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac2c1cf90>
```



Пример простейшей проверки точности кластеризации

```
sum((df_cut['Sex'] == 'мужской') & df_cut['Label'] == 1)
```

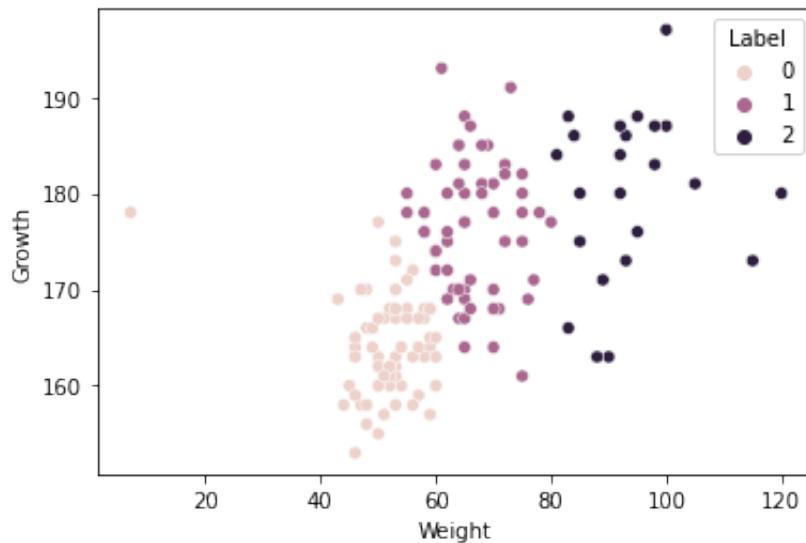
24

24 ошибки в определении мужчин

```
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(df_cut[['Weight', 'Growth']])
df_cut['Label'] = kmeans.labels_
```

```
sns.scatterplot(data=df_cut, x='Weight', y='Growth', hue='Label')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac2b8a910>
```



▼ Базовые методы машинного обучения

Предсказание (классификация - категориальный признак) и регрессия (числовой признак)

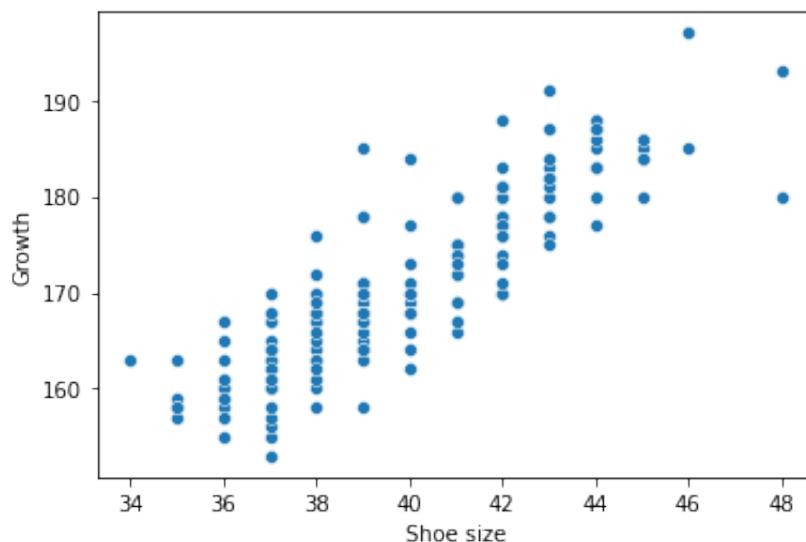
Линейная регрессия

```
df_cut = df[['Growth', 'Shoe size']]  
df_cut = df_cut.dropna()
```

```
from sklearn.linear_model import LinearRegression
```

```
sns.scatterplot(data=df_cut, x='Shoe size', y='Growth')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac2b25210>
```



Просматривается линейная зависимость, значит можно применять линейную регрессию.

Целевой признак - рост

```
linearRegression = LinearRegression()  
results = linearRegression.fit(df_cut['Shoe size'].values.reshape(-1, 1),  
                               y=df_cut['Growth'].values)
```

```
results.coef_, results.intercept_ # коэффициент и свободный член
```

```
(array([2.77966583]), 60.26465138793836)
```

```
df_cut['Predicted Growth'] = results.predict(df_cut['Shoe size'].values.reshape(
```

```
df_cut.head()
```

	Growth	Shoe size	Predicted Growth
0	170	40	171.451284
1	191	43	179.790282
2	172	41	174.230950
3	168	38	165.891953
4	159	36	160.332621

Проверим точность предсказаний, посчитаем среднюю абсолютную ошибку

```
from sklearn.metrics import mean_absolute_error  
mean_absolute_error(df_cut['Growth'], df_cut['Predicted Growth'])  
  
3.4686666590527797
```

Теперь множественная линейная регрессия

```
df_cut = df[['Middle and index finger',  
            'Middle and ring finger',  
            'Middle and little finger']]
```

```
df_cut
```

	Middle and index finger	Middle and ring finger	Middle and little finger
0	20	10.0	40.0
1	5	5.0	20.0
2	13	10.0	26.0
3	12	13.5	35.0
4	10	11.0	22.0
...
181	13	23.0	45.0
182	8	9.0	15.0
183	9	16.0	35.0
184	9	15.0	25.0
185	7	10.0	35.0

171 rows × 3 columns

```
results = linearRegression.fit(df_cut[['Middle and ring finger', 'Middle and lit
y=df_cut['MIddle and index finger'].values)
results.coef_, results.intercept_
(array([0.11527745, 0.04296171]), 7.4077839882224925)
```

Итоговая модель:

Middle and index finger = 0.115 * Middle and ring finger + 0.043 * Middle and little finger + 7.41

```
df_cut['Predicted'] = results.predict(df_cut[['Middle and ring finger', 'Middle
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWith
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: 
```

```
mean_absolute_error(df_cut['MIddle and index finger'], df_cut['Predicted'])
```

3.1885190843328126

Проверка точности модели на новой выборке Предыдущие тесты точности алгоритма были не совсем верными, так как модель обучалась и проверялась на одних и тех же данных

Далее рассказывается о стандартном подходе разделения на тренировочную и тестовую выборку

Здесь будут использоваться два датасета

```
df_test = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/students_test.csv'  
                     delimiter=',')
```

```
df_test_cut = df_test[['MIddle and index finger',  
                      'Middle and ring finger',  
                      'Middle and little finger']]
```

Используем функцию predict на тестовую выборку

```
df_test_cut['Predicted'] = results.predict(df_test_cut[['Middle and ring finger']])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWith  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs  
"""\nEntry point for launching an IPython kernel.
```

```
mean_absolute_error(df_test_cut['Predicted'], df_test_cut['MIddle and index finger'])
```

4.68045833086758

Алгоритмы классификации

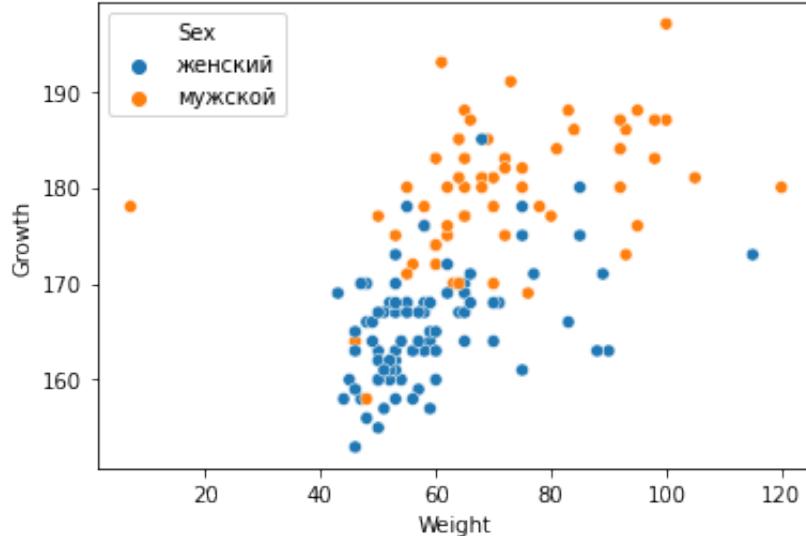
Метод k ближайших соседей (kNN)

```
from sklearn.neighbors import KNeighborsClassifier
```

```
df_cut = df[['Growth', 'Weight', 'Sex']]  
df_cut = df_cut.dropna()
```

```
sns.scatterplot(data=df_cut, x='Weight', y='Growth', hue='Sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac2b25450>
```



Для корректного подсчета расстояний необходимо провести нормировку данных

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
# обучение нормировщика
```

```
scaler.fit(df_cut[['Weight', 'Growth']].values.reshape(-1, 2))
```

```
StandardScaler()
```

```
# нормировка
```

```
arr = scaler.transform(df_cut[['Weight', 'Growth']].values.reshape(-1, 2))
```

В итоге данные превращаются в обезличенный массив

```
arr
```

```
array([[-0.01667202, -0.1499425 ],  
       [ 0.54119943,  2.09919504],  
       [-0.26461489,  0.06426107],  
       [-0.3266006 , -0.36414608].
```

```
[ -0.45057203, -1.32806217],  
[ -0.88447205,  0.59977001],  
[ -0.57454347,  0.92107538],  
[  3.45452809,  0.92107538],  
[  0.29325656,  1.45658432],  
[ -0.38858632, -0.36414608],  
[ -1.0704292 , -1.43516396],  
[  2.09084234,  1.24238074],  
[ -1.00844348, -0.57834966],  
[ -0.01667202,  1.45658432],  
[ -0.26461489,  1.24238074],  
[ -0.6985149 , -0.47124787],  
[ -0.3266006 , -0.79255323],  
[  1.78091375,  1.5636861 ],  
[ -0.57454347, -0.36414608],  
[ -0.51255775, -1.43516396],  
[  0.0453137 ,  1.77788968],  
[  0.0453137 ,  0.59977001],  
[  0.35524228, -0.79255323],  
[ -0.26461489, -0.89965502],  
[ -0.20262917,  2.31339862],  
[  1.5949566 , -0.89965502],  
[  0.35524228,  1.02817716],  
[  0.0453137 , -0.1499425 ],  
[  1.90488519,  1.77788968],  
[ -0.01667202,  1.45658432],  
[  0.0453137 ,  0.59977001],  
[ -0.6985149 , -1.00675681],  
[  1.22304231,  1.5636861 ],  
[ -0.3266006 , -0.68545144],  
[  0.66517086, -1.11385859],  
[  0.23127085,  1.45658432],  
[ -1.00844348, -0.1499425 ],  
[  0.23127085,  1.02817716],  
[  0.47921371,  1.24238074],  
[ -0.14064345,  0.92107538],  
[ -0.38858632,  0.49266822],  
[ -0.14064345,  0.06426107],  
[  0.35524228,  0.7068718 ],  
[  0.66517086,  0.38556644],  
[ -0.6985149 , -0.89965502],  
[ -1.00844348, -1.64936753],  
[  1.71892804,  0.92107538],  
[ -0.38858632, -0.89965502],  
[  0.72715658, -0.25704429],  
[  0.0453137 , -0.25704429],  
[ -0.07865774, -0.1499425 ],  
[ -0.6985149 , -0.1499425 ],  
[  0.66517086,  1.13527895],  
[  0.10729941,  1.67078789],  
[ -0.38858632,  0.7068718 ],  
[  0.85112801,  0.7068718 ],  
[  0.0453137 ,  0.92107538],  
[  2.52474235,  1.02817716].
```

```
[ -0.88447205, -0.89965502],  
[ 1.29562802, 0.29566441]
```

Сама классификация

```
model = KNeighborsClassifier(n_neighbors=1) # инициализация  
model.fit(arr, y=df_cut['Sex'].values)  
  
KNeighborsClassifier(n_neighbors=1)
```

Модель обучена, можно проверять ее на тестовой выборке

```
df_test_cut = df_test[['Growth', 'Weight', 'Sex']]  
df_test_cut = df_test_cut.dropna()  
  
arr_test = scaler.transform(df_test_cut[['Weight', 'Growth']].values.reshape(-1,  
  
df_test_cut['Predicted'] = model.predict(arr_test)  
  
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWith  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row_indexer,col_indexer] = value instead  
  
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs  
"""\nEntry point for launching an IPython kernel.
```

```
df_test_cut.head()
```

	Growth	Weight	Sex	Predicted
0	180.0	78.0	мужской	мужской
1	167.0	50.0	женский	женский
2	178.0	70.0	женский	мужской
3	156.0	47.0	женский	женский
4	186.0	94.0	мужской	мужской

Метрики качества алгоритма классификации

Матрица сопряженности

```
pd.crosstab(df_test_cut['Sex'], df_test_cut['Predicted'])
```

Predicted женский мужской

Sex			
женский	39	8	
мужской	3	27	

Закодируем результаты

```
df_test_cut['Code'] = 0
```

```
df_test_cut.loc[(df_test_cut['Sex'] == 'мужской') & (df_test_cut['Predicted'] == 'мужской')] = 1
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWith
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

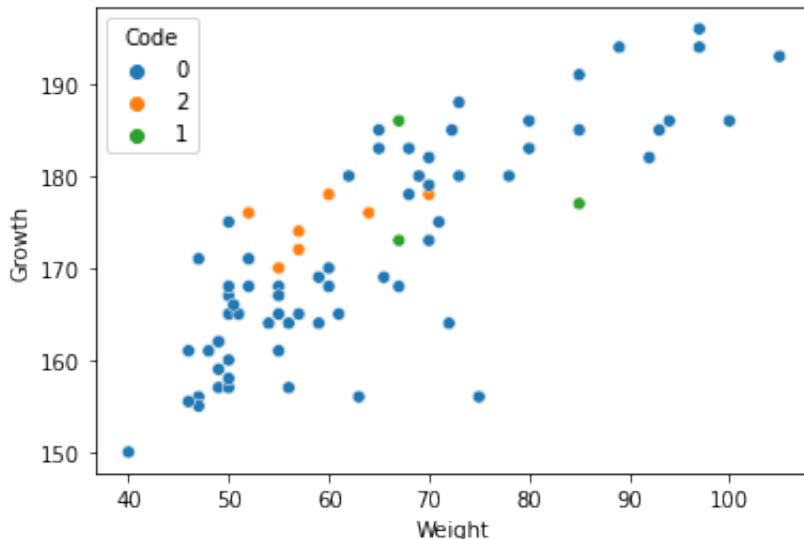
See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/indexing.html#setting-with-index>

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1817: SettingWith
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/stable/indexing.html#setting-with-index>

```
sns.scatterplot(data=df_test_cut, x='Weight', y='Growth', hue='Code')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac2996cd0>
```

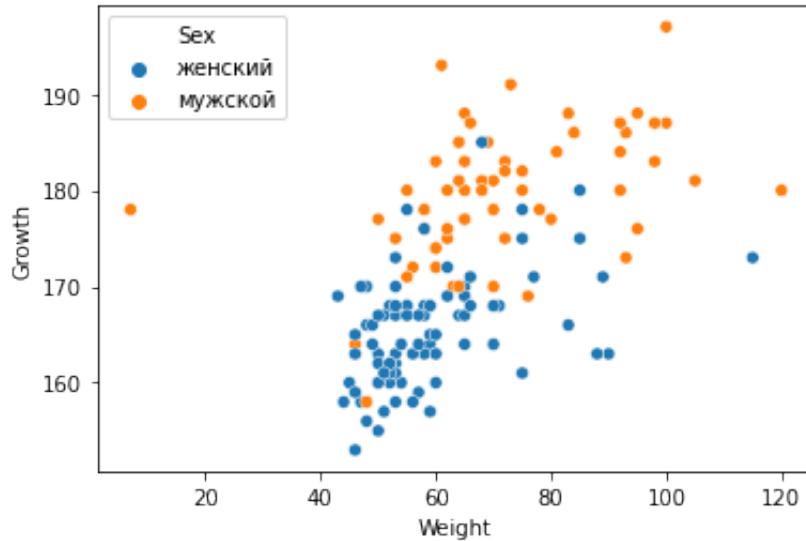


Линейный классификатор

```
from sklearn.linear_model import SGDClassifier
```

```
sns.scatterplot(data=df_cut, x='Weight', y='Growth', hue='Sex')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac2929bd0>
```



Задача линейного классификатора - разделить данные на группы прямой линией

```
scaler.fit(df_cut[['Weight', 'Growth']].values.reshape(-1, 2))
arr = scaler.transform(df_cut[['Weight', 'Growth']].values.reshape(-1, 2))
```

```
model = SGDClassifier()
```

```
model.fit(arr, y=df_cut['Sex'].values)
```

```
SGDClassifier()
```

```
df_test_cut = df_test[['Growth', 'Weight', 'Sex']]
```

```
df_test_cut = df_test_cut.dropna()
```

```
arr_test = scaler.transform(df_test_cut[['Weight', 'Growth']].values.reshape(-1,
```

```
df_test_cut['Predicted'] = model.predict(arr_test)
```

```
pd.crosstab(df_test_cut['Predicted'], df_test_cut['Sex'])
```

Sex женский мужской

Predicted

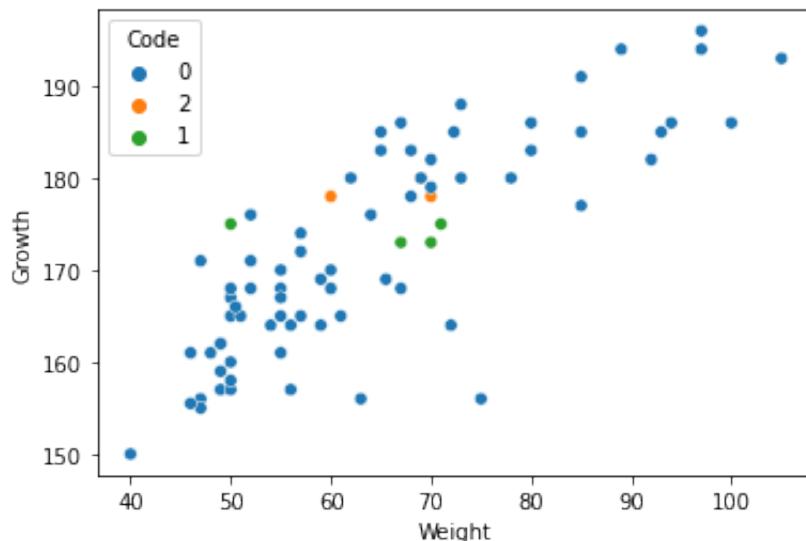
Predicted	женский	мужской
женский	44	4
мужской	3	26

```
df_test_cut['Code'] = 0
```

```
df_test_cut.loc[(df_test_cut['Sex'] == 'мужской') & (df_test_cut['Predicted'] == df_test_cut['Sex'])] = 1
```

```
sns.scatterplot(data=df_test_cut, x='Weight', y='Growth', hue='Code')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9ac2a31690>
```



Вероятностные алгоритмы

Вместо предсказания дают вероятность принадлежности к одному или другому классу

```
from sklearn.ensemble import RandomForestClassifier
```

```
df_cut = df[['Growth', 'Weight', 'Sex', 'Hair length', 'Children number']]  
df_cut = df_cut.dropna()
```

```
model = RandomForestClassifier(max_depth=2, random_state=0)

model.fit(df_cut[['Growth', 'Weight', 'Hair length', 'Children number']].values,
          y=df_cut['Sex'].values)

RandomForestClassifier(max_depth=2, random_state=0)

df_test_cut = df_test[['Growth', 'Weight', 'Sex', 'Hair length', 'Children number']]
df_test_cut = df_test_cut.dropna()
```

Для получения вероятности используем метод **predict_proba**

```
result = model.predict_proba(df_test_cut[['Growth', 'Weight', 'Hair length', 'Children number']])
print(result)
```

```
[[0.05717479 0.94282521]
 [0.95703983 0.04296017]
 [0.92220778 0.07779222]
 [0.94430864 0.05569136]
 [0.04557862 0.95442138]
 [0.95326172 0.04673828]
 [0.95058413 0.04941587]
 [0.20051871 0.79948129]
 [0.08046935 0.91953065]
 [0.60783415 0.39216585]
 [0.70492419 0.29507581]
 [0.06705782 0.93294218]
 [0.6689992 0.3310008]
 [0.96331532 0.03668468]
 [0.96513351 0.03486649]
 [0.05959696 0.94040304]
 [0.14395969 0.85604031]
 [0.03395659 0.96604341]
 [0.96193671 0.03806329]
 [0.05959696 0.94040304]
 [0.03654891 0.96345109]
 [0.03654891 0.96345109]
 [0.10687531 0.89312469]
 [0.95703983 0.04296017]
 [0.92891486 0.07108514]
 [0.68630715 0.31369285]
 [0.07115209 0.92884791]
 [0.10142245 0.89857755]
 [0.93072869 0.06927131]
 [0.96513351 0.03486649]
 [0.95507991 0.04492009]
 [0.93724458 0.06275542]
 [0.9646143 0.0353857]
 [0.0361200 0.0638701 1]
```

```
[0.12028684 0.87971316]
[0.04479506 0.95520494]
[0.0434047 0.9565953 ]
[0.06923173 0.93076827]
[0.96279612 0.03720388]
[0.9646143 0.0353857 ]
[0.9646143 0.0353857 ]
[0.93359305 0.06640695]
[0.95703983 0.04296017]
[0.12121668 0.87878332]
[0.96513351 0.03486649]
[0.56588125 0.43411875]
[0.0434047 0.9565953 ]
[0.9646143 0.0353857 ]
[0.88054449 0.11945551]
[0.93241863 0.06758137]
[0.95703983 0.04296017]
[0.9646143 0.0353857 ]
[0.82292556 0.17707444]
[0.96193671 0.03806329]
[0.93259282 0.06740718]
[0.89663671 0.10336329]
[0.72626932 0.27373068]
[0.9347861 0.0652139 ]
[0.03395659 0.96604341]
```

... 20051071 0.700401201

Получили вероятности принадлежности к классу

```
df_test_cut['pr class 0'] = result[:, 0]
df_test_cut['pr class 1'] = result[:, 1]
df_test_cut.head()
```

	Growth	Weight	Sex	Hair length	Children number	pr class 0	pr class 1
0	180.0	78.0	мужской	1.2	2.0	0.057175	0.942825
1	167.0	50.0	женский	30.0	2.0	0.957040	0.042960
3	156.0	47.0	женский	20.0	2.0	0.922208	0.077792
5	150.0	40.0	женский	30.0	2.0	0.944309	0.055691
7	183.0	80.0	мужской	2.0	0.0	0.045579	0.954421

Селекция признаков

Использование нецелевых категориальных признаков

```
from sklearn import tree
```

Заменим признак категориальный признак пола числами, для этого просто кодируем разные метки нулями и единицами

```
from sklearn import preprocessing
```

```
coder = preprocessing.LabelEncoder()
```

```
coder.fit(df['Sex'])
```

LabelEncoder()

```
coder.transform(df['Sex'])
```

```
df1 = df
```

```
df1['Sex'] = coder.transform(df1['Sex'])
```

```
for name in ['Coin', 'Animal', 'Army']:
```

```
coder.fit(df1[name])
```

```
df1[name] = coder.transform(df1[name])
```

```
df_cut = df1[['Growth', 'Weight', 'Sex', 'Hair length',  
             'Coin', 'Children number', 'Animal', 'Army']]
```

```
df_cut = df_cut.dropna()
```

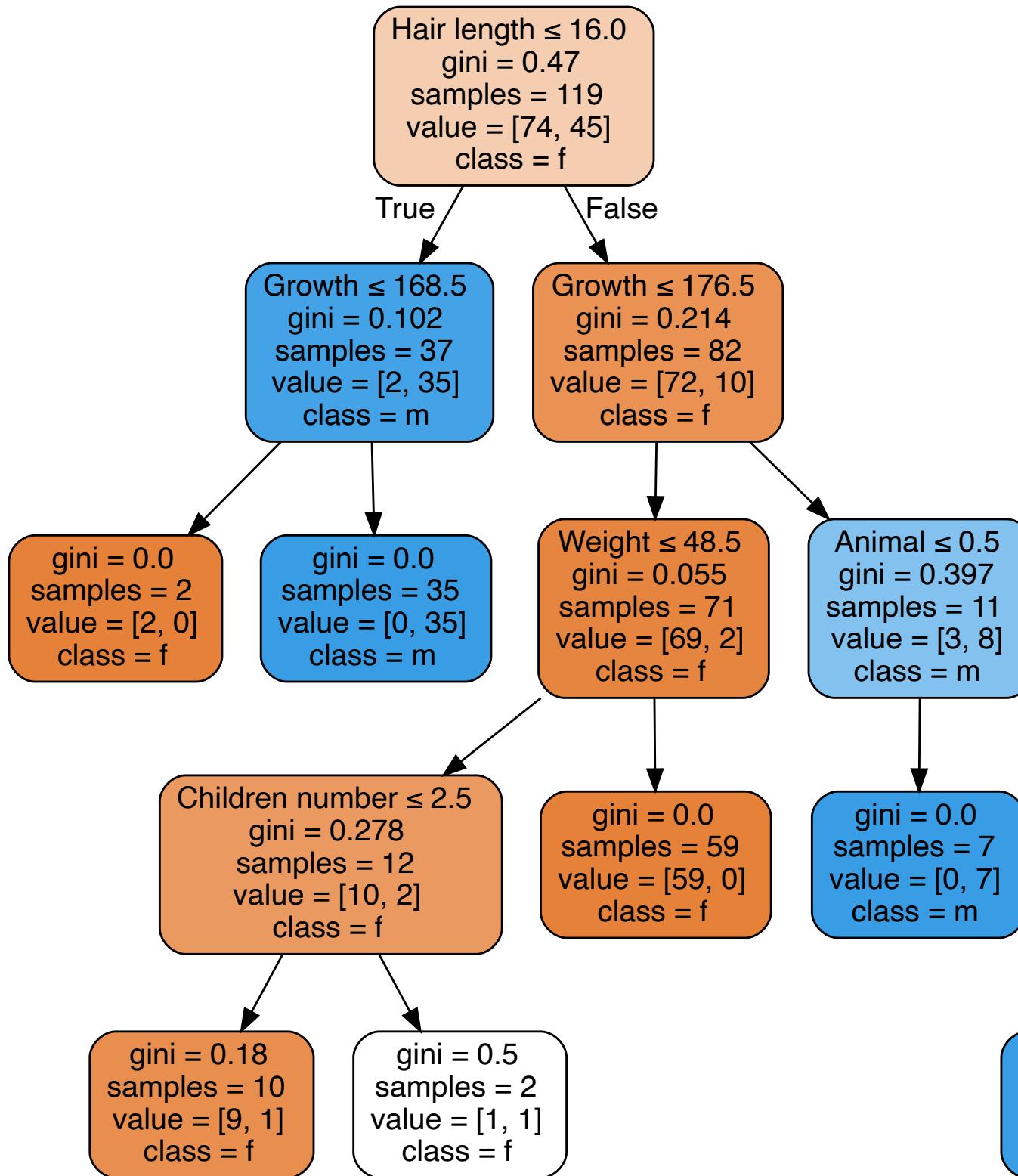
```
model = tree.DecisionTreeClassifier(max_depth=4)
model.fit(df_cut[['Growth', 'Weight', 'Hair length', 'Children number',
                  'Coin', 'Animal', 'Army']].values.reshape(-1, 7),
          y=df_cut['Sex'].values)

DecisionTreeClassifier(max_depth=4)
```

Визуализация результатов

```
import graphviz

dot_data = tree.export_graphviz(model, out_file=None,
                                feature_names=['Growth', 'Weight', 'Hair length',
                                               'Children number', 'Coin', 'Animal',
                                               'Army'],
                                class_names=['f', 'm'],
                                filled=True,
                                rounded=True,
                                special_characters=True
                               )
graph = graphviz.Source(dot_data)
graph
```



```
df_test_cut = df_test[['Growth', 'Weight', 'Sex',  
                      'Hair length', 'Children number',  
                      'Coin', 'Animal', 'Army']]
```

```
df_test_cut = df_test_cut.dropna()
```

```
for name in['Sex', 'Animal', 'Army', 'Coin']:
    coder.fit(df_test_cut[name])
    df_test_cut[name] = coder.transform(df_test_cut[name])

df_test_cut['Predict'] = model.predict(df_test_cut[['Growth', 'Weight', 'Coin',
                                                    'Hair length', 'Army', 'Anim
                                                    'Children number']].values.r

quality = pd.crosstab(df_test_cut['Predict'], df_test_cut['Sex'])
quality
```

Sex	0	1
Predict		
0	30	0
1	12	26

```
# recall (ж) – доля правильно предсказанных женщин
quality[0][0] / sum(quality[0])
```

0.7142857142857143

```
# recall (м)
quality[1][1] / sum(quality[1])
```

1.0

```
# precision (ж) – доля истинных женщин среди людей, которых классификатор отнес
quality[0][0] / (quality[0][0] + quality[1][0])
```

1.0

```
# precision (м)
quality[1][1] / (quality[1][1] + quality[0][1])
```

0.6842105263157895

Кодирование категориальных признаков и их полезность

```
coder = preprocessing.LabelEncoder()

df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/students.csv',
                 delimiter=',')
```

Переведем все категории в числа

```
df1 = df.dropna()
for name in ['Sex', 'Coin', 'Animal', 'Army', 'Glasses',
             'Your rating in university', 'Fastfood',
             'Hostel', 'Chocolate', 'Brother-sister',
             'Plane seat', 'Problems in last semester',
             'Rock paper scissors', 'Strange people',
             'Your insitute']:
    coder.fit(df1[name])
    df1[name] = coder.transform(df1[name])
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9: SettingWith
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs>
if __name__ == '__main__':

Проверим важность признаков

```
from sklearn.ensemble import ExtraTreesClassifier

selector = ExtraTreesClassifier()
result = selector.fit(df1[df1.columns], df1['Sex'])
result.feature_importances_

array([0.00486312, 0.05612654, 0.1077762 , 0.00277149, 0.004382 ,
       0.00374471, 0.01107933, 0.00502965, 0.01358079, 0.02781332,
       0.00137147, 0.00303761, 0.0038468 , 0.00117425, 0.0059186 ,
       0.00166235, 0.00488792, 0.00341772, 0.00588856, 0.00385442,
       0.00341795, 0.0133458 , 0.0073797 , 0.41186935, 0.00735933,
       0.02651898, 0.00213353, 0.00393967, 0.00334157, 0.00258795,
       0.00384484, 0.00422083, 0.00554578, 0.0071112 , 0.09920971,
       0.00242854, 0.06129814, 0.0059842 , 0.0042773 , 0.00561605,
       0.00609938, 0.00336402, 0.00399035, 0.00661712, 0.00565334,
       0.00348775, 0.01189274, 0.00523805])
```

```
features_table = pd.DataFrame(result.feature_importances_, index=df1.columns,
                               columns=['Importance'])

features_table.sort_values(by='Importance', ascending=False)
```

	Importance
Sex	0.411869
Shoe size	0.107776
Army	0.099210
Hair length	0.061298
Growth	0.056127
Computer science rating	0.027813
Coin	0.026519
Physics rating	0.013581
Weight	0.013346
Middle and ring finger	0.011893
Russian rating	0.011079
Glasses	0.007380
Problems in last semester	0.007359
Putin age	0.007111
Brother-sister	0.006617
City population	0.006099
Floor number	0.005984
Biology rating	0.005919
Minutes to first class	0.005889
Plane seat	0.005653
Chocolate	0.005616
Width of 5000 mm	0.005546
Middle and little finger	0.005238
Maths rating	0.005030
Social science rating	0.004888

Age	0.004863
Year of birth	0.004382
Social network duration min	0.004277
Height of 5000 mm	0.004221
Your insitute	0.003990
Animal	0.003940
...	...

Видим убывающий список наиболее эффективных признаков, описывающих пол человека

Оставим только часть значимых признаков

Middle and index finger 0.003488

```
df_cut = df[['Army', 'Shoe size', 'Hair length', 'Growth',
             'Coin', 'Computer science rating', 'Weight', 'Sex']]
df_cut = df_cut.dropna()
```

Strange people 0.003364

Другой способ кодирования категориальных признаков

```
df_cut = pd.get_dummies(df_cut)
df_cut.head()
```

	Shoe size	Hair length	Growth	Computer science rating	Weight	Army_могут призвать	Army_не призовут (по разным причинам)	Coin_Oрел
0	40	50.0	170	84	64.0	0	1	1
1	43	7.0	191	72	73.0	0	1	1
2	41	4.0	172	0	60.0	1	0	0
3	38	20.0	168	0	59.0	0	1	0

Получили дополнительные столбцы с различными вариантами ответов

Теперь удалим лишние столбцы, тк, например, в случае пола, нам хватит только столбца с одним полом, чтобы знать данные по каждому полу

```
df_cut = df[['Army', 'Shoe size', 'Hair length', 'Growth',
             'Coin', 'Computer science rating', 'Weight', 'Sex']]
df_cut = df_cut.dropna()
df_cut = pd.get_dummies(df_cut, drop_first=True)
df_cut.head()
```

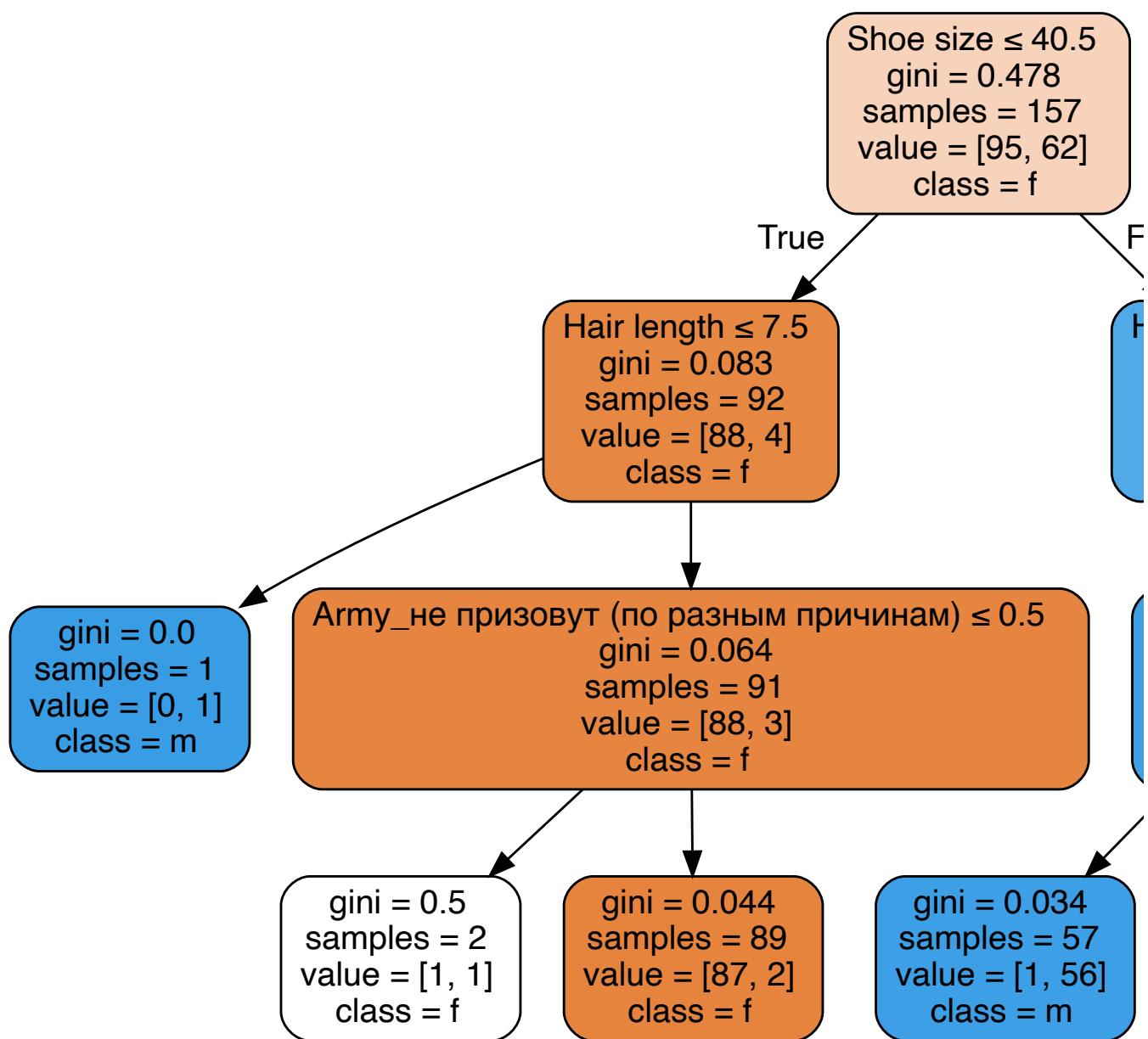
	Shoe size	Hair length	Growth	Computer science rating	Weight	Army_не призовут (по разным причинам)	Coin_Решка	Sex_мужской
0	40	50.0	170	84	64.0	1	0	0
1	43	7.0	191	72	73.0	1	0	1
2	41	4.0	172	0	60.0	0	1	1

Удалили избыточность

```
model = tree.DecisionTreeClassifier(max_depth=3)
model.fit(df_cut[['Army_не призовут (по разным причинам)',
                  'Shoe size',
                  'Growth',
                  'Hair length',
                  'Coin_Решка',
                  'Computer science rating',
                  'Weight']].values.reshape(-1, 7),
          y=df_cut['Sex_мужской'])
```

DecisionTreeClassifier(max_depth=3)

```
dot_data = tree.export_graphviz(model, out_file=None,
                                 feature_names=['Army_не призовут (по разным прич',
                                                'Shoe size',
                                                'Growth',
                                                'Hair length',
                                                'Coin_Решка',
                                                'Computer science rating',
                                                'Weight'],
                                 class_names=['f', 'm'],
                                 filled=True, rounded=True,
                                 special_characters=True)
graph = graphviz.Source(dot_data)
graph
```



```
df_test_cut = df_test[['Army', 'Shoe size', 'Hair length', 'Coin',
                      'Computer science rating', 'Weight', 'Sex',
                      'Growth']]
df_test_cut = df_test_cut.dropna()
```

```
df_test_cut = pd.get_dummies(df_test_cut, drop_first=True)
df_test_cut.head()
```

	Shoe size	Hair length	Computer science rating	Weight	Growth	Army_не призовут (по разным причинам)	Coin_Решка	Sex_мужской
0	44.0	1.2	88	78.0	180.0	1	0	1
1	38.0	30.0	0	50.0	167.0	1	0	0
2	41.0	50.0	0	70.0	178.0	1	0	0

```
df_test_cut['Predict'] = model.predict(df_test_cut[['Army_не призовут (по разным
                                                    'Shoe size',
                                                    'Growth',
                                                    'Hair length',
                                                    'Coin_Решка',
                                                    'Computer science rating',
                                                    'Weight']]].values.reshape(-1, 7))
```

```
pd.crosstab(df_test_cut['Predict'], df_test_cut['Sex_мужской'])
```

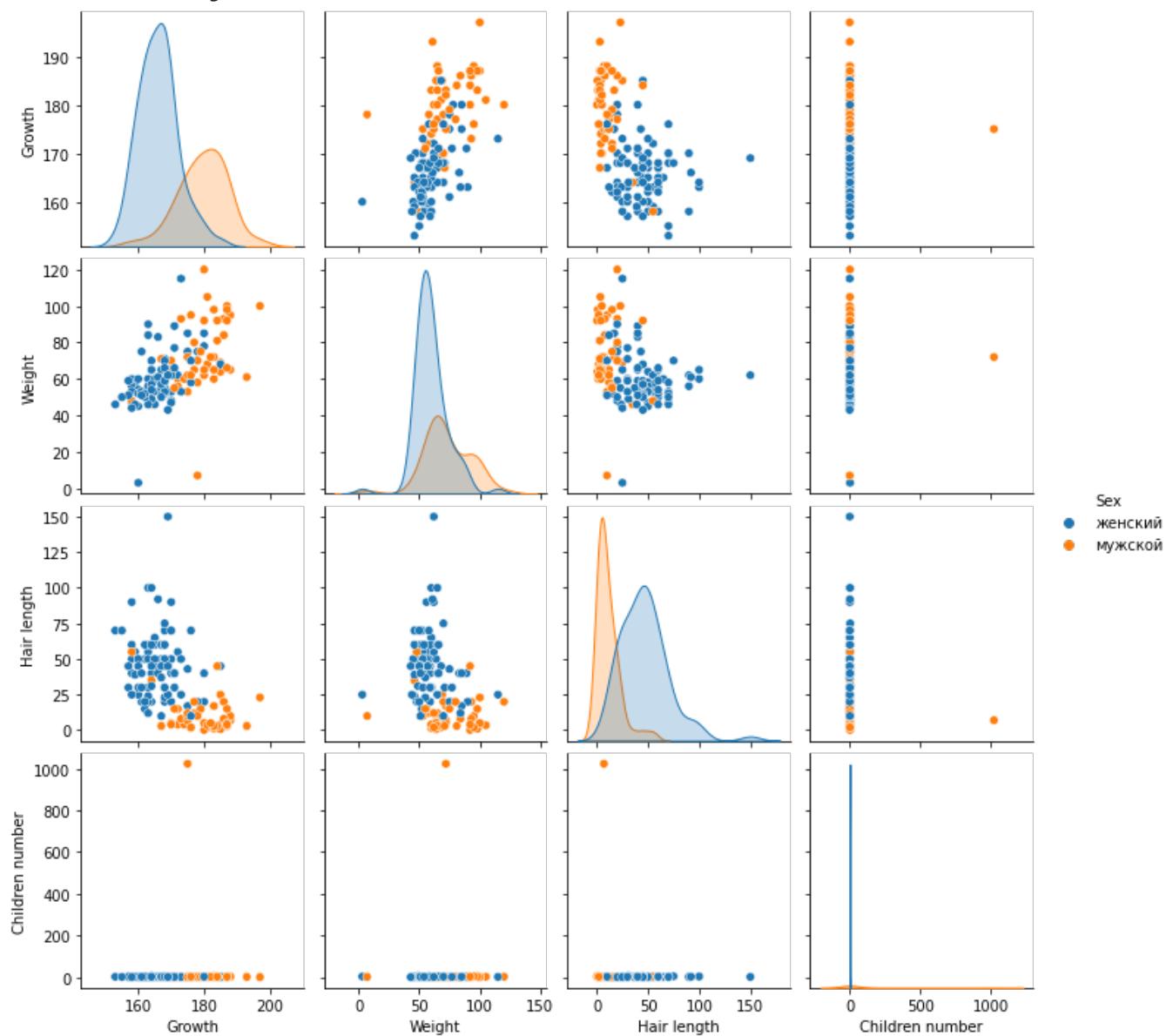
Sex_мужской	0	1
Predict		
0	46	1
1	1	29

▼ Решающие деревья

```
df_cut = df[['Growth', 'Weight', 'Sex', 'Hair length', 'Children number']]
df_cut = df_cut.dropna()
```

```
sns.pairplot(df_cut, hue='Sex')
```

<seaborn.axisgrid.PairGrid at 0x7f9ac2ab1ad0>

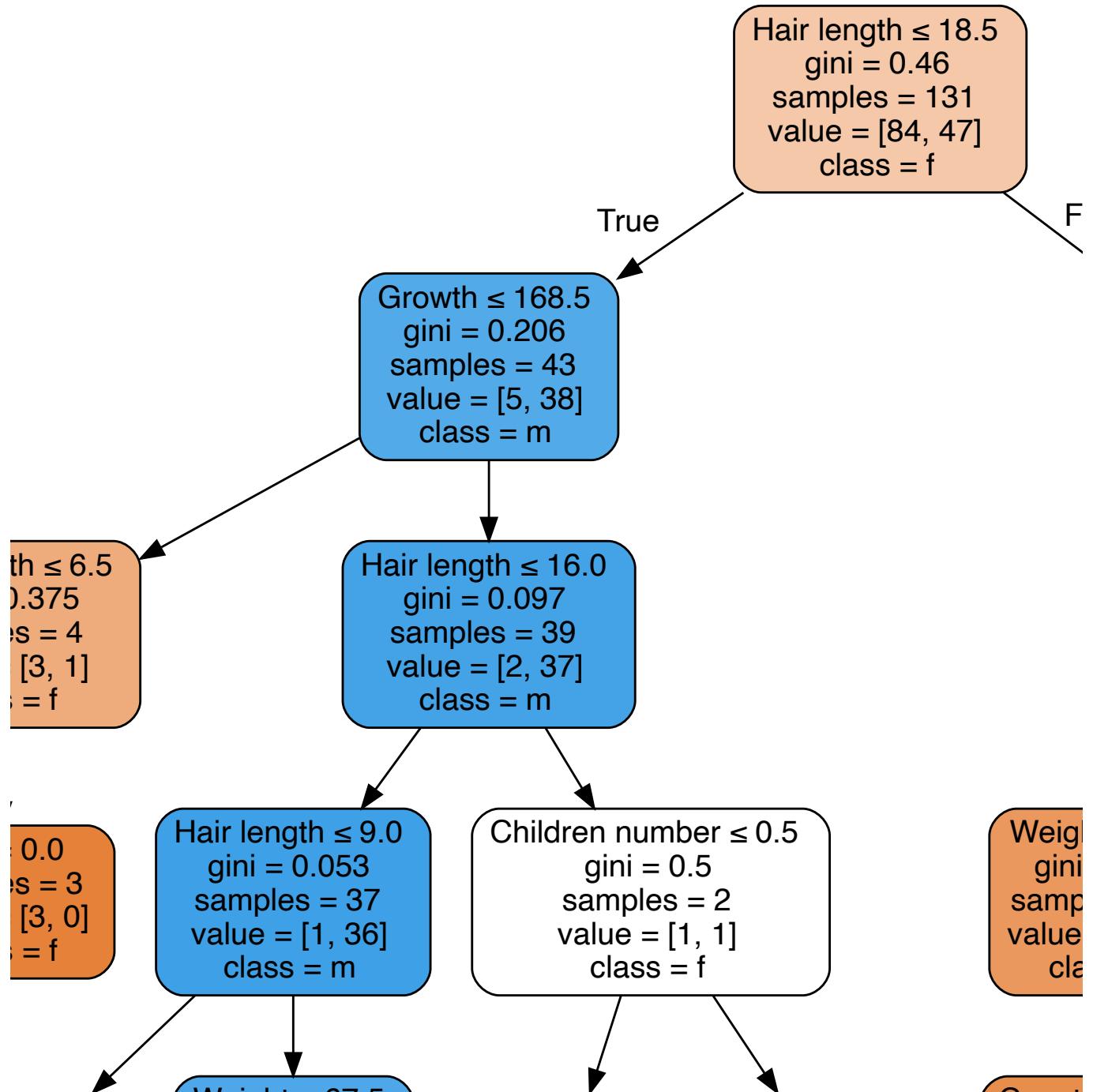


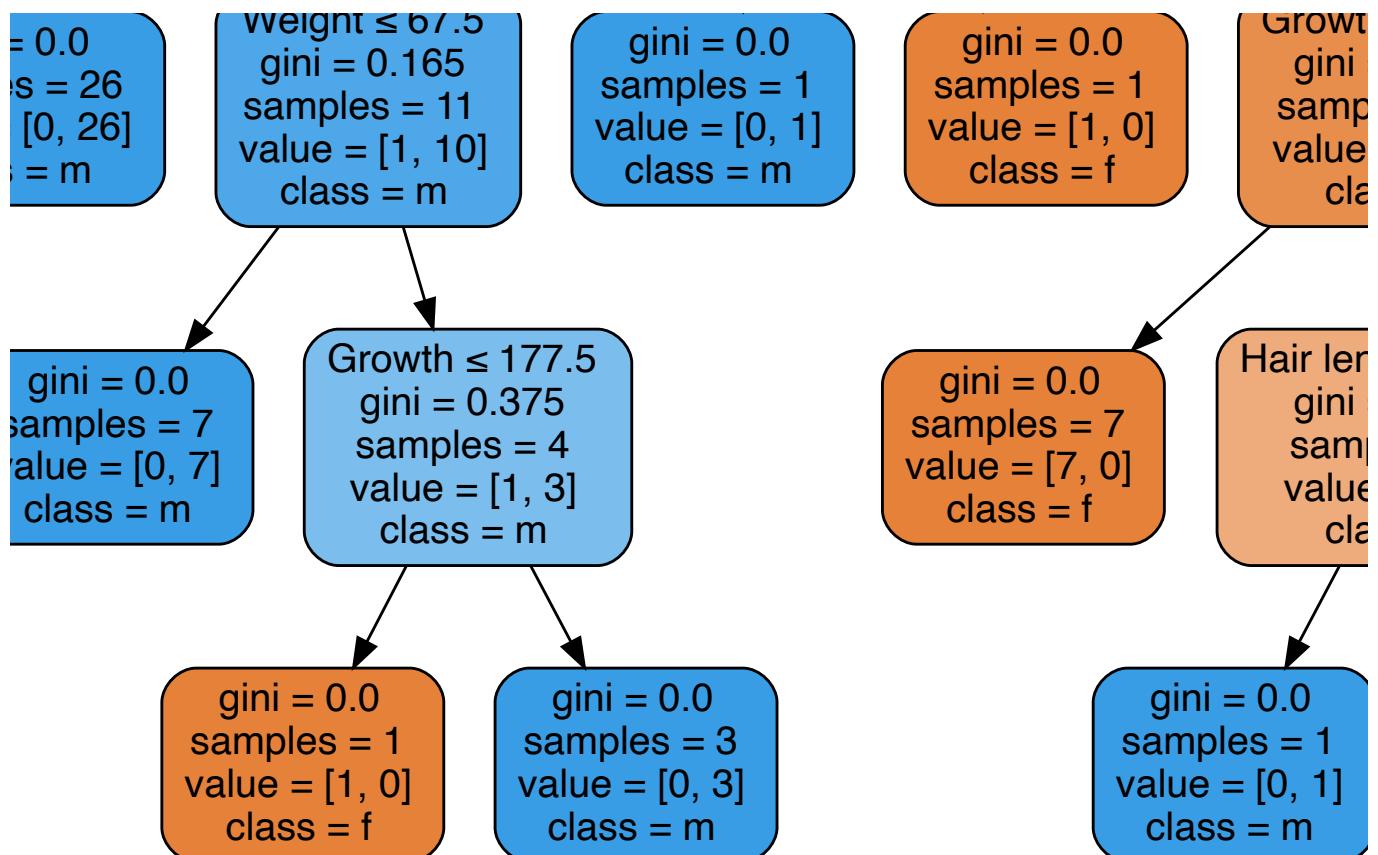
```
model = tree.DecisionTreeClassifier()
```

```
model.fit(df_cut[['Growth', 'Weight', 'Hair length', 'Children number']].values,
          y=df_cut['Sex'].values)
```

```
DecisionTreeClassifier()
```

```
dot_data = tree.export_graphviz(model, out_file=None,
                                 feature_names=['Growth', 'Weight', 'Hair length'],
                                 class_names=['f', 'm'],
                                 filled=True, rounded=True,
                                 special_characters=True)
graph = graphviz.Source(dot_data)
graph
```





```

df_test_cut = df_test[['Growth', 'Weight', 'Sex', 'Hair length',
                      'Children number']]
df_test_cut = df_test_cut.dropna()
  
```

Большой плюс - для деревьев не нужно нормировать данные

```

df_test_cut['Predicted'] = model.predict(df_test_cut[['Growth', 'Weight',
                                                       'Hair length', 'Children n
  
```

```

pd.crosstab(df_test_cut['Predicted'], df_test_cut['Sex'])
  
```

Sex женский мужской

Predicted	женский	мужской
женский	41	1
мужской	1	25

Минусы - громоздкая визуализация. Выйти из ситуации можно с помощью установки параметра max_depth, дерево станет более компактным. Однако точность классификатора может снизиться.

Еще раз про показатели качества классификации

Основных два - precision и recall

precision - доля верных наблюдений в предсказанной группе

recall - доля верных наблюдений среди всех наблюдений данной группы (изначальных, не предсказанных)

Для их подсчета есть функция

```
from sklearn.metrics import precision_recall_fscore_support
```

Здесь важен порядок, первый аргумент - истинные метки, второй - предсказанные.

Только в таком случае результаты будут верные

```
precision_recall_fscore_support(df_test_cut['Sex'], df_test_cut['Predicted'])  
  
(array([0.97619048, 0.96153846]),  
 array([0.97619048, 0.96153846]),  
 array([0.97619048, 0.96153846]),  
 array([42, 26]))
```

Первая строка - precision для каждого пола

Вторая строка - recall для каждого пола

Третья строка - F1 метрика

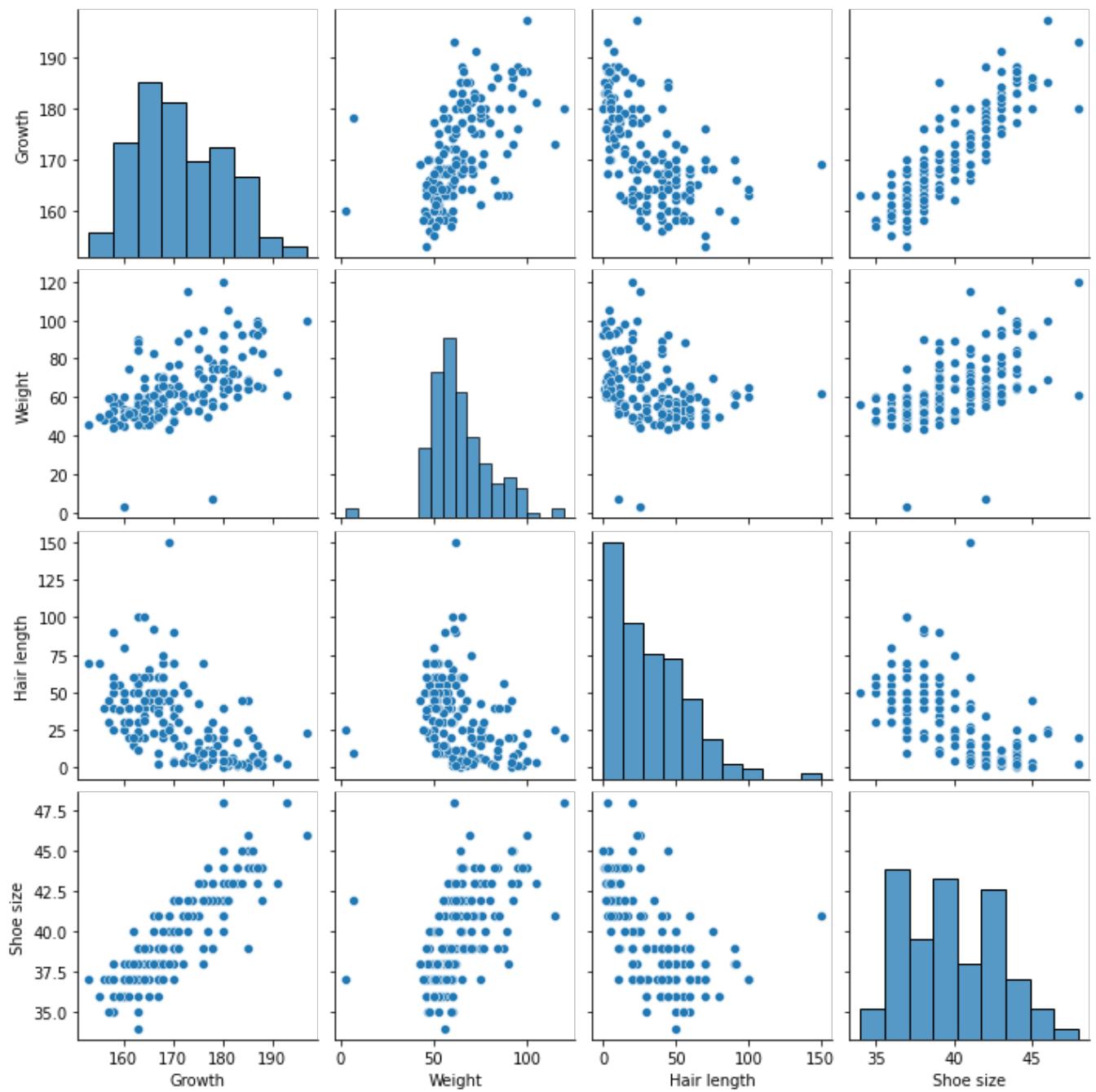
4 - количество наблюдений каждого класса

Задача регрессии. Дерево решений

```
df_cut = df[['Growth', 'Weight', 'Hair length', 'Shoe size']]  
df_cut = df_cut.dropna()
```

```
sns.pairplot(df_cut)
```

<seaborn.axisgrid.PairGrid at 0x7f9ac1ec1d90>



Будем предсказывать рост по всем остальным признакам с помощью
DecisionTreeRegressor

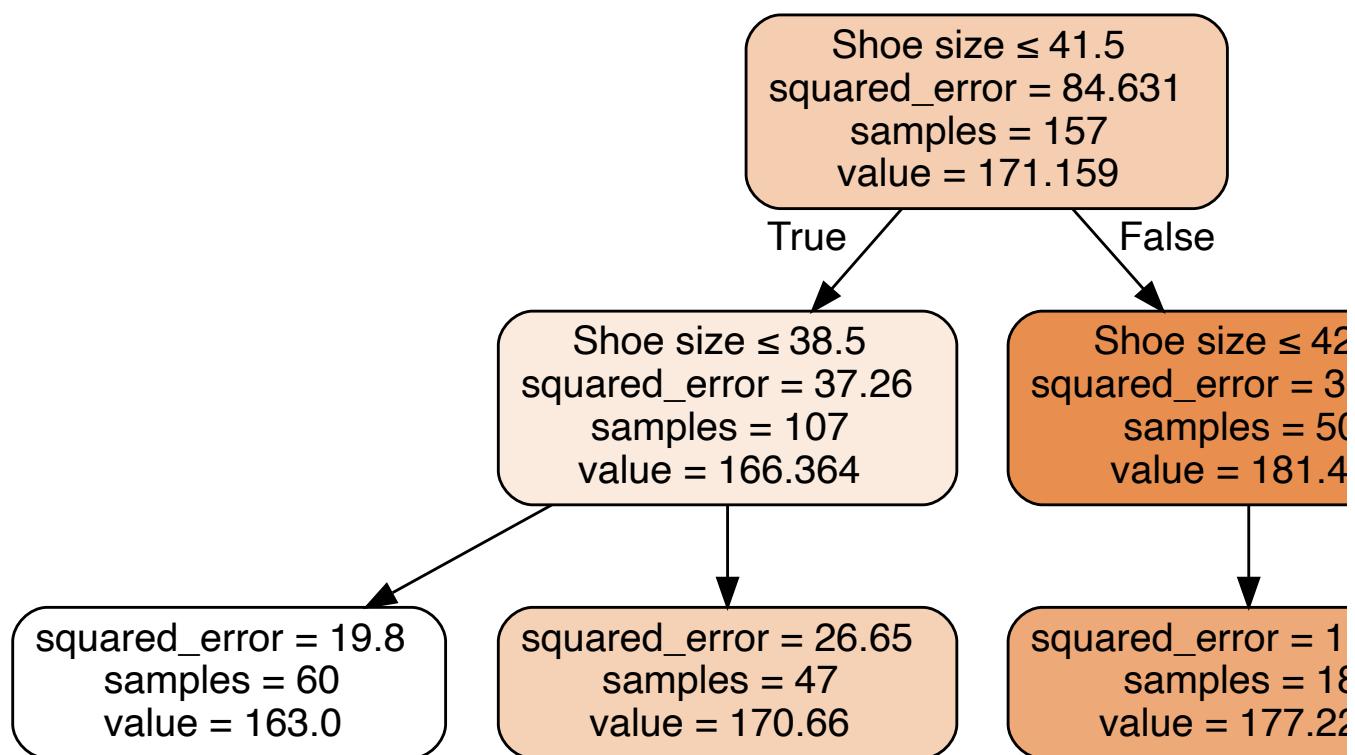
```

model = tree.DecisionTreeRegressor(max_depth=2)
model.fit(df_cut[['Weight', 'Hair length', 'Shoe size']].values.reshape(-1, 3),
          y=df_cut['Growth'].values)

DecisionTreeRegressor(max_depth=2)

dot_data = tree.export_graphviz(model, out_file=None,
                                feature_names=['Weight', 'Hair length', 'Shoe size'],
                                class_names='Growth',
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph

```



```

df_test_cut = df_cut[['Growth', 'Weight', 'Shoe size', 'Hair length']]
df_test_cut = df_test_cut.dropna()

df_test_cut['Predicted'] = model.predict(df_test_cut[['Weight', 'Hair length', 'Shoe size']])

```

В отличие от классификации, здесь для проверки качества используем среднюю абсолютную ошибку

```
mean_absolute_error(df_test_cut['Growth'], df_test_cut['Predicted'])
```

3.7618918552649414

Для разных значений глубины поиска можно получать разные значения точности. Но не всегда большее значение ведет к возрастанию точности. Здесь можно столкнуться с проблемой переобучения.

Бустинг и ансамбли алгоритмов

С точки зрения статистики, совместное использование (ансамбль) нескольких алгоритмов машинного обучения будет давать большую точность, чем эти алгоритмы по-отдельности

Пример - RandomForest, составленный из различных деревьев

```
df_cut = df[['Growth', 'Weight', 'Sex', 'Hair length', 'Children number']]  
df_cut = df_cut.dropna()
```

```
model = RandomForestClassifier(max_depth=2, random_state=0)  
model.fit(df_cut[['Growth', 'Weight', 'Hair length', 'Children number']].values,  
y=df_cut['Sex'].values)
```

```
RandomForestClassifier(max_depth=2, random_state=0)
```

```
df_test_cut = df_test[['Growth', 'Weight', 'Sex', 'Hair length', 'Children numbe  
df_test_cut = df_test_cut.dropna()
```

```
df_test_cut['Predicted'] = model.predict(df_test_cut[['Growth', 'Weight', 'Hair l
```

```
pd.crosstab(df_test_cut['Sex'], df_test_cut['Predicted'])
```

Predicted женский мужской

Sex	
женский	42
мужской	1
	25

Градиентный бустинг

```
from sklearn.ensemble import GradientBoostingClassifier

model = GradientBoostingClassifier(random_state=0)
model.fit(df_cut[['Growth', 'Weight', 'Hair length', 'Children number']].values,
          y=df_cut['Sex'].values)

GradientBoostingClassifier(random_state=0)

df_test_cut = df_test[['Growth', 'Weight', 'Sex', 'Children number',
                      'Hair length']]
df_test_cut = df_test_cut.dropna()

df_test_cut['Predicted'] = model.predict(df_test_cut[['Growth',
                                                       'Weight',
                                                       'Hair length',
                                                       'Children number']].values)

pd.crosstab(df_test_cut['Sex'], df_test_cut['Predicted'])
```

Predicted женский мужской

Sex		
женский	42	0
мужской	1	25

```
precision_recall_fscore_support(df_test_cut['Sex'], df_test_cut['Predicted'])

(array([0.97674419, 1.        ]),
 array([1.        , 0.96153846]),
 array([0.98823529, 0.98039216]),
 array([42, 26]))
```

В данном месте повторяется раздел "Вероятностные алгоритмы"

Повторим итог

```
model = RandomForestClassifier(max_depth=2, random_state=0)
model.fit(df_cut[['Growth', 'Weight', 'Hair length', 'Children number']].values,
          y=df_cut['Sex'])

RandomForestClassifier(max_depth=2, random_state=0)

df_test_cut = df_test[['Growth', 'Weight', 'Sex', 'Children number',
                      'Hair length']]
df_test_cut = df_test_cut.dropna()

result = model.predict_proba(df_test_cut[['Growth', 'Weight',
                                         'Hair length',
                                         'Children number']])

df_test_cut['pr class 0'] = result[:, 0]
df_test_cut['pr class 1'] = result[:, 1]
df_test_cut.head()
```

	Growth	Weight	Sex	Children number	Hair length	pr class 0	pr class 1
0	180.0	78.0	мужской	2.0	1.2	0.111984	0.888016
1	167.0	50.0	женский	2.0	30.0	0.960954	0.039046
3	156.0	47.0	женский	2.0	20.0	0.930384	0.069616
5	150.0	40.0	женский	2.0	30.0	0.954804	0.045196
7	183.0	80.0	мужской	0.0	2.0	0.072297	0.927703

```
df_test_cut[(df_test_cut['Sex'] == 'мужской') & (df_test_cut['pr class 1'] > 0.9
```

	Growth	Weight	Sex	Children number	Hair length	pr class 0	pr class 1
7	183.0	80.0	мужской	0.0	2.0	0.072297	0.927703
23	194.0	97.0	мужской	0.0	5.0	0.057225	0.942775
25	185.0	72.3	мужской	100.0	4.0	0.098636	0.901364
27	185.0	93.0	мужской	2.0	10.0	0.094359	0.905641
28	193.0	105.0	мужской	2.0	11.0	0.094359	0.905641
33	196.0	97.0	мужской	2.0	15.0	0.096930	0.903070
45	185.0	65.0	мужской	2.0	4.0	0.076419	0.923581
48	185.0	85.0	мужской	2.0	4.0	0.079070	0.920930
60	191.0	85.0	мужской	2.0	3.0	0.079070	0.920930
82	182.0	92.0	мужской	0.0	4.0	0.057225	0.942775
93	183.0	68.0	мужской	0.0	8.0	0.076349	0.923651
97	188.0	73.0	мужской	2.0	3.0	0.079070	0.920930

[Платные продукты Colab](#) - [Отменить подписку](#)