

Лекция 5. Google Colab(Jupyter). Знакомство с аналитикой.

**Курс**

[**https://docs.google.com/document/d/1wmPgo-AOM1su0VqSj3L3C77tUUiF2BtiFQwKEgkjN\_c/edit#**](https://docs.google.com/document/d/1wmPgo-AOM1su0VqSj3L3C77tUUiF2BtiFQwKEgkjN_c/edit#)

**Оглавление**

[Введение](#_e1ww5mjf8m1) **2**

[Термины, используемые в лекции](#_30j0zll) **3**

[Чтение и предварительный просмотр данных](#_hpthp5lo1c45) **3**

[Выбор данных](#_ua36ubr05zd) **5**

[Простая статистика](#_fhgoegyilwy4) **7**

[Изображаем статистические отношения](#_fimmp0k49wz4) **8**

[Линейные графики](#_q9v0ptbn35zz) **10**

[Гистограмма](#_ujlt60dxosg3) **11**

[Выводы](#_kwazm2cm1g73) **13**

# Введение

Знакомство с аналитикой. Мы будем пользоваться таким инструментом как Google colab

На лекции мы познакомимся с инструментом для работы с табличными данными(**pandas**) и способами визуализации данных с помощью библиотек **matplotlib** и **seaborn**. Прежде, чем приступать непосредственно к машинному обучению, важно произвести **EDA**(Exploratory Data Analysis) - Разведочный анализ данных.

Он состоит в анализе основных свойств данных, нахождения в них общих закономерностей, распределений и аномалий, построение начальных моделей, зачастую с использованием инструментов визуализации.

Понятие введено математиком **Джоном Тьюки**, который сформулировал цели такого анализа следующим образом:

1. Максимальное «проникновение» в данные
2. Выявление основных структур
3. Выбор наиболее важных переменных
4. Обнаружение отклонений и аномалий
5. Проверка основных гипотез

# Термины, используемые в лекции

**Перцентиль** - это показатель, используемый в статистике, показывающий значение, ниже которого падает определенный процент наблюдений в группе наблюдений

**Scatterplot (Точечный график) -** Математическая диаграмма, изображающая значения двух переменных в виде точек на декартовой плоскости.

**Медиана набора чисел** — число, которое находится в середине этого набора, если его упорядочить по возрастанию, то есть такое число, что половина из элементов набора не меньше него, а другая половина не больше.

**Базовые функции для работы с данными**

Библиотека **pandas** может читать многие форматы, включая: **.csv, .xslx, .xls, .txt, sql** и многие другие. Полный список по [ссылке](https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html)

Чтобы подключить библиотеку к Вашей программе необходимо написать следующее:

|  |
| --- |
| import pandas as pd |

**Напоминание:** as(alias) - псевдоним. Мы можем сократить название все библиотеки до 2-х букв.

Прочтем файл .csv(он находится в Google Colab в папке **sample\_data**) с помощью библиотеки **pandas**

|  |
| --- |
| df = pd.read\_csv('sample\_data/california\_housing\_train.csv') |

Для того чтобы прочитать первые n строк таблицы, необходимо воспользоваться следующей функцией:

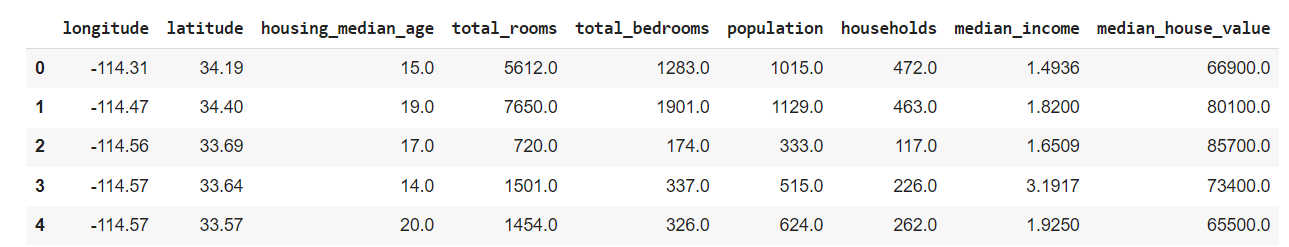
|  |
| --- |
| DataFrame.head(n=5) |

Где DataFrame - это таблица с данными, которая предварительно была открыта. Мы ее открыли и записали в переменную **df.** Необязательно указывать n=5, вместо 5 мы можем указать любое число(число не должно превосходить количество строк в таблице). Если Вы ничего не укажете в круглых скобках, то ошибка не вылезет, по умолчанию будут выведены первые 5 строк таблицы.

**Пример:**

|  |
| --- |
| df.head() |

**Результат:**

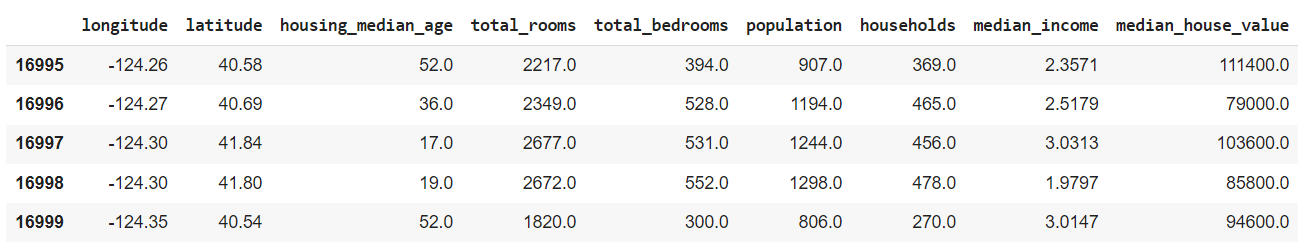
****

Как мы знаем, в нашем мире почти все симметрично, есть отрицательные числа, а есть положительные и тд. Значит, если есть функция, которая показывает первые 5 строк таблицы, то и есть функция, которая показывает последние 5 строк таблицы.   
Да, действительно, это так. Давайте с ней познакомимся

**Пример:**

|  |
| --- |
| df.tail() |

**Результат:**

****

Данная функция работает аналогично с head(). Необязательно выводить последние 5 строчек, можно указать сколько угодно.

Иногда заранее неизвестно сколько строк и столбцов находится внутри таблицы, чтобы это сделать необходимо воспользоваться специальной функцией.

**Пример:**

|  |
| --- |
| df.shape |

**Результат:**

|  |
| --- |
| (17000, 9) |

Функция shape возвращает размеры таблицы: кортеж из 2 значений, 1 - количество строк, 2 - количество столбцов.

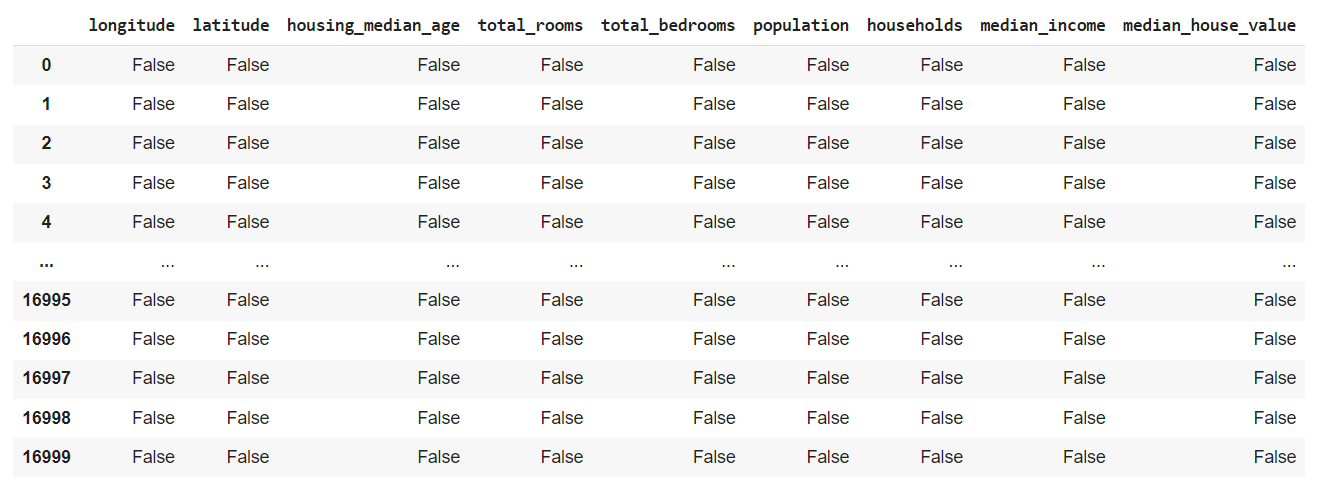
Согласитесь, что все не раз делали заказ на каком-нибудь маркетплейсе. И когда мы заполняли поле **“Email”**, то могли его пропустить, потому что указанно, что оно необязательное и не хотели видеть лишнего спама. Вы когда-нибудь задумывались, как в этом случае эти данные будут выглядеть внутри базы данных(таблице)? Пропуск? Пустая ячейка? Нет. Когда нужно указать, что в данной ячейки таблицы ничего нет указывается значение **null**.

Чтобы обнаружить пустые значения в таблице данных необходимо воспользоваться функцией **.isnull()**.

**Пример:**

|  |
| --- |
| df.isnull() |

**Результат:**

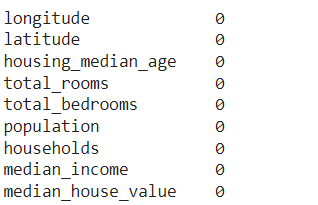


Функция привела нашу таблицу к следующему виду **True-False,** где **True** - это пустая ячейка, **False** - это заполненная ячейка. Но это неудобно, то есть нам надо просматривать 17 000 \* 9 = 153 000 ячеек. Вау… Это займет слишком много времени. Однако, мы можем воспользоваться еще одной функцией .**sum()**. Данная функция выведет количество null-значений в каждой ячейке по столбцам.

**Пример:**

|  |
| --- |
| df.isnull().sum() |

**Результат:**

****

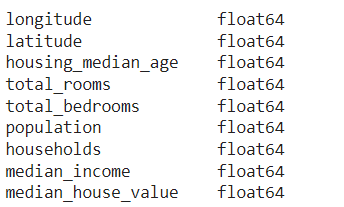
Можно сделать следующий вывод: пустые значения в нашей таблицы отсутствуют.

Еще при работе с C#, мы поняли, что у каждой переменной есть свой тип данных. Также и здесь, у каждого столбца есть свой тип данных, чтобы это узнать, нужно применить функцию **.dtypes**.

**Пример:**

|  |
| --- |
| df.dtypes |

**Результат:**



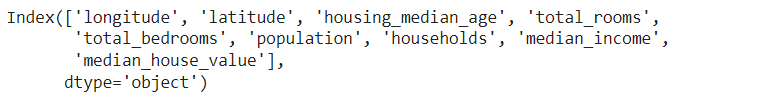
Делаем вывод: у всей таблицы данных один тип **float(дробное число)**

Чтобы узнать название всех столбцов в таблице, воспользуйтесь функцией **.columns.**

**Пример:**

|  |
| --- |
| df.columns |

**Результат:**

****

# Выборка данных

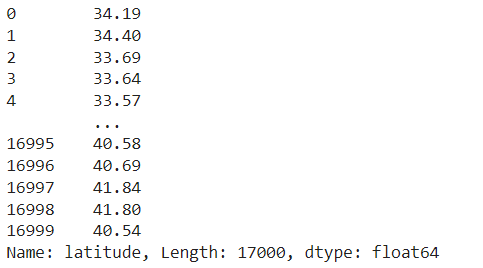
|  |
| --- |
| * **Медиана набора чисел** — число, которое находится в середине этого набора, если его упорядочить по возрастанию, то есть такое число, что половина из элементов набора не меньше него, а другая половина не больше. |

Если Вы хотите вывести 1 столбец на экран, то можно указать следующее выражение, которое позволит это сделать.

**Пример:**

|  |
| --- |
| df['latitude'] |

**Результат:**

****

Что мы будем делать, если нам потребуется вывести на экран сразу несколько столбцов? Не очень удобно будет это прописывать вот таким образом:

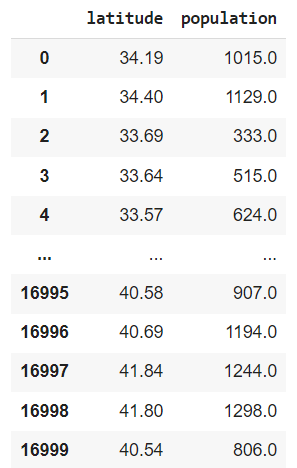
|  |
| --- |
| print(df['latitude'])  print(df['population']) |

Конечно есть решение данного вопроса

**Пример:**

|  |
| --- |
| df[['latitude', 'population']] |

**Результат:**

****



**Задание:** Необходимо вывести столбец **total\_rooms**, у которого медианный возраст здания(**housing\_median\_age**) меньше **20**.

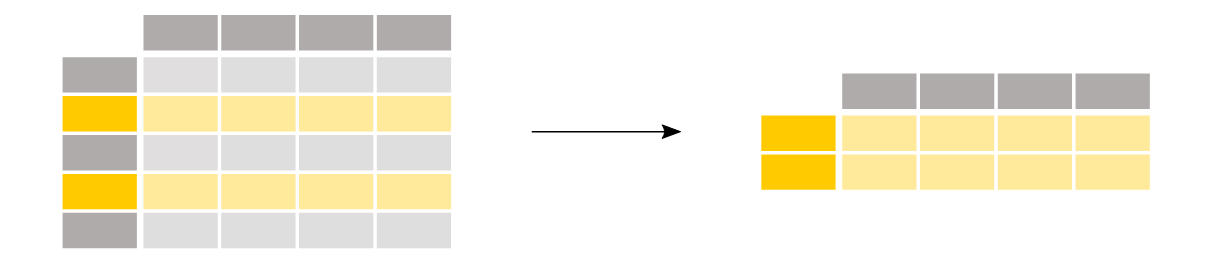
Для того чтобы решить это задание, давайте познакомимся с синтаксисом выборки данных. На самом деле, это ничем не отличается от операторов ветвления.

**Решение:**

|  |
| --- |
| df[df['housing\_median\_age'] < 20] |

**Результат:**

****



Если Вам нужно поставить другое условие, то аналогично.

Мы помним с C#, что иногда приходится проверять несколько условий сразу. Чтобы проверить несколько условий внутри Google Colab, указывается так:

|  |
| --- |
| df[(df['housing\_median\_age'] > 20) & (df['total\_rooms'] > 2000)] |

& - выполнение одновременно **всех** условий.

| - выполнение **хотя бы одного** из условия.

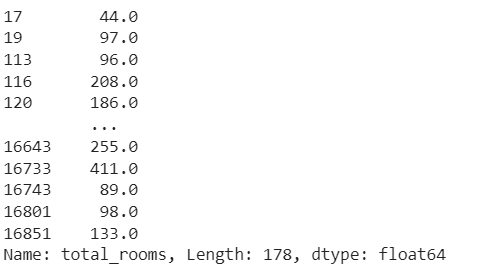


Первую часть задания мы успешно выполняли! Только загвоздка… Нам не нужна вся таблица, а лишь один столбец, как это сделать?

**Решение:**

|  |
| --- |
| df[df['housing\_median\_age'] < 20, 'total\_rooms']  # или (если необходимо вывести 2 и более столбцов  df[df['housing\_median\_age'] < 20, ['total\_bedrooms', 'total\_rooms']] |

**Результат:**

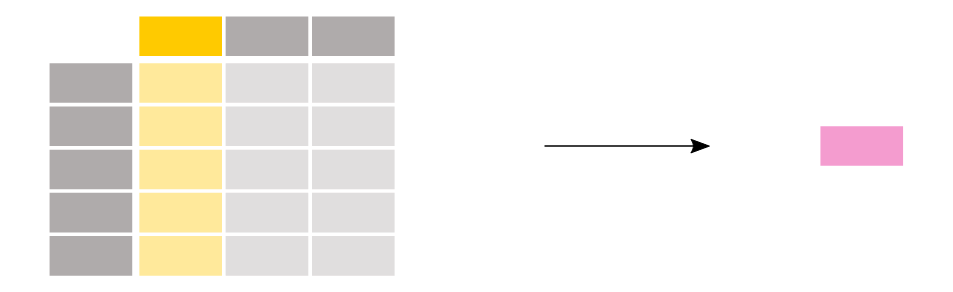
****



# 

# Простая статистика

**Pandas** позволяет получить основные простые данные для описательной статистики. Такие как минимальное значение в столбце, максимальное значение, сумма всех значений, среднее значение



**Максимальное значение:**

|  |
| --- |
| print(df['population'].max()) |

**Результат:**

**35682.0**

**Минимальное значение:**

|  |
| --- |
| print(df['population'].min()) |

**Результат:**

**3.0**

**Среднее значение:**

|  |
| --- |
| print(df['population'].mean()) |

**Результат:**

**1429.5739411764705**

**Сумма:**

|  |
| --- |
| print(df['population'].sum()) |

**Результат:**

**24302757.0**

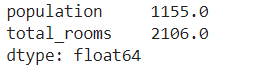
Эту же статистику можно рассчитывать сразу для нескольких столбцов



**Медианное значение:**

|  |
| --- |
| df[['population', 'total\_rooms']].median() |

**Результат:**

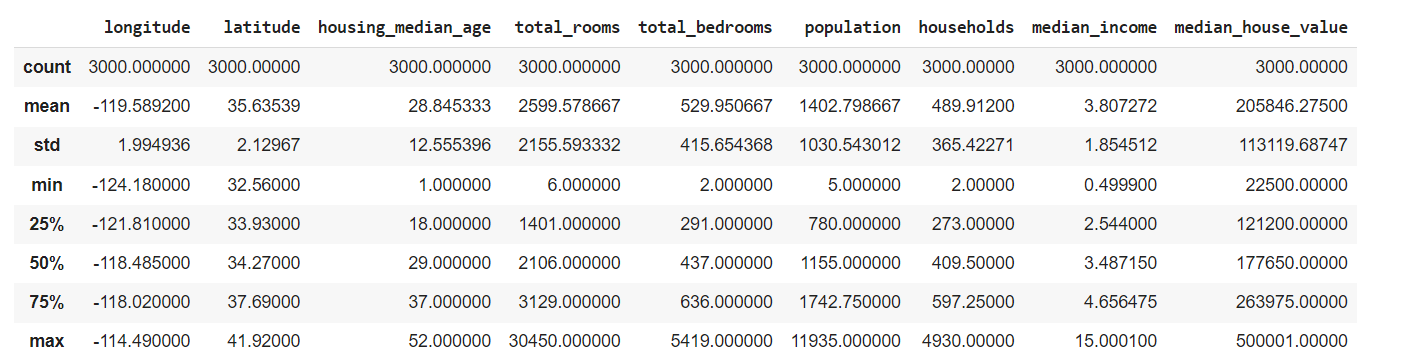
****

|  |
| --- |
| * **Перцентиль** - это показатель, используемый в статистике, показывающий значение, ниже которого падает определенный процент наблюдений в группе наблюдений |

Получить общую картину можно простой командой **describe**

|  |
| --- |
| df.describe() |

**Результат:**

****

**count** - Общее кол-во не пустых строк

**mean** - среднее значение в столбце

**std** - стандартное отклонение от среднего значения

**min** - минимальное значение

**max** - максимальное значение

Числа **25%, 50%, 75%** - перцентили

# Изображаем статистические отношения

**Scatterplot (Точечный график)**

Математическая диаграмма, изображающая значения двух переменных в виде точек на декартовой плоскости. Библиотека **seaborn** без труда принимает **pandas** **DataFrame**(таблицу). Чтобы изобразить отношения между двумя столбцами достаточно указать, какой столбец отобразить по оси **x**, а какой по оси **y**.

Для того чтобы начать работу с библиотекой seaborn, ее необходимо импортировать к себе в программу:

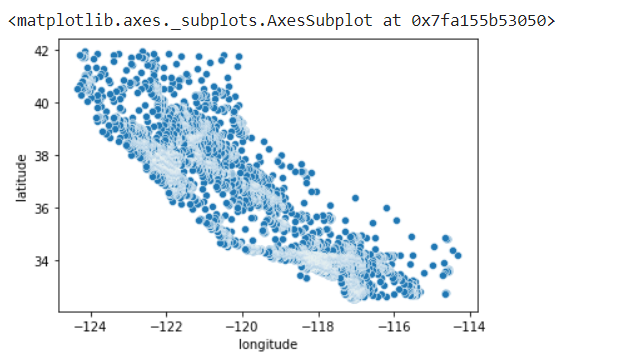
|  |
| --- |
| import seaborn as sns |

Вернемся к нашей таблице. Можно заметить, что дома расположены в определенной "полосе" долготы и широты.

**Изображение точек долготы по отношению к широте:**

|  |
| --- |
| sns.scatterplot(data=df, x="longitude", y="latitude") |

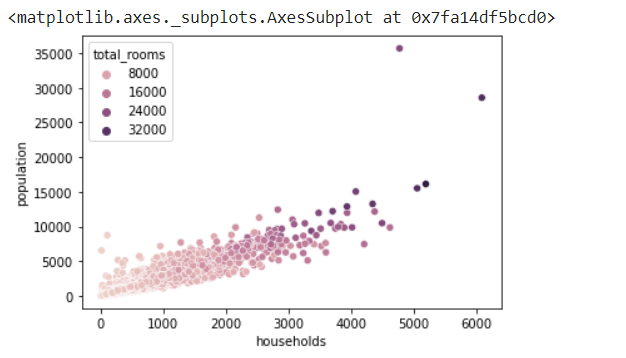
**Результат:**

****

Помимо двумерных отношений, мы можем добавить "дополнительное измерение" с помощью цвета. В данном случае опять же достаточно очевидное отношение, чем выше кол-во семей, тем выше кол-во людей и соответственно комнат.

|  |
| --- |
| sns.scatterplot(data=df, x="households", y="population", hue="total\_rooms") |

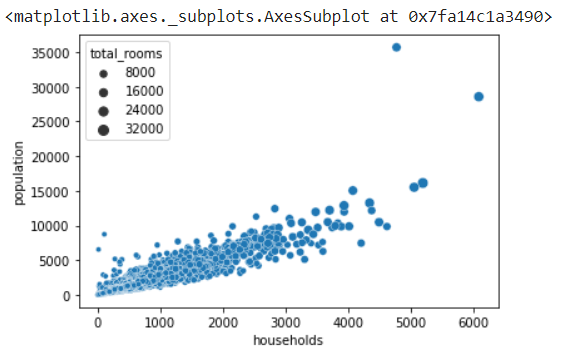
**Результат:**

****

Помимо обозначения дополнительного измерения цветом мы можем использовать **size**.

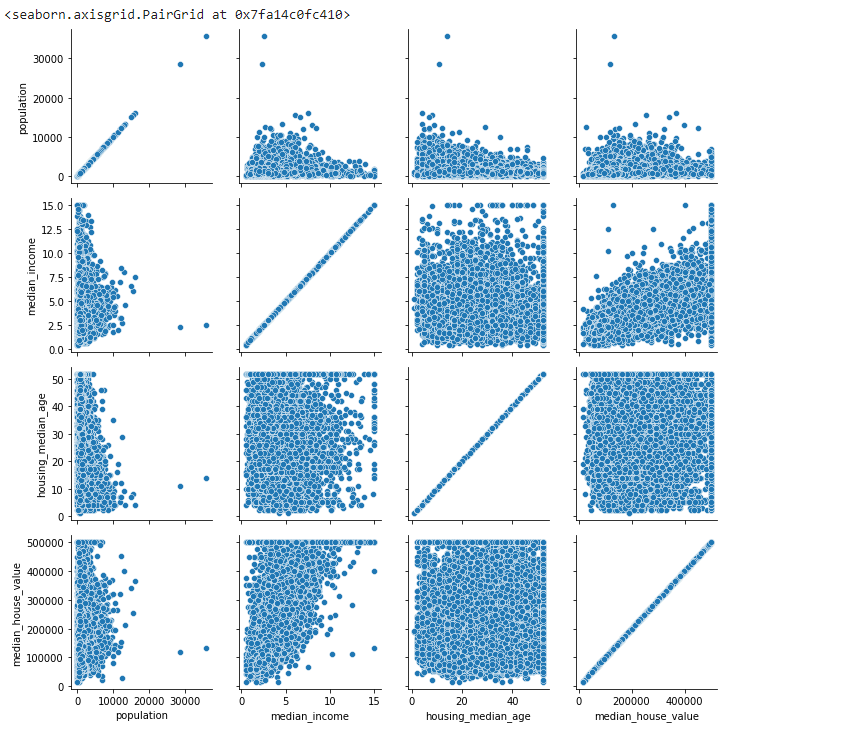
|  |
| --- |
| sns.scatterplot(data=df, x="households", y="population", hue="total\_rooms") |

**Результат:**

****

Мы можем визуализировать сразу несколько отношений, используя класс **PairGrid** внутри **seaborn. PairGrid** принимает как аргумент pandas **DataFrame** и визуализирует все возможные отношения между ними, в соответствии с выбранным типом графика.

|  |
| --- |
| cols = ['population', 'median\_income', 'housing\_median\_age', 'median\_house\_value']  g = sns.PairGrid(df[cols])  g.map(sns.scatterplot) |

**Результат:**

Как Вы думаете, чем вызвана линейная зависимость по диагонали?

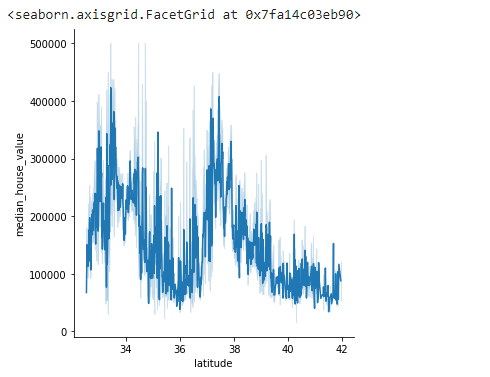
# Линейные графики

Хорошо подойдут, если есть временная или какая-либо иная последовательность и значения, которые могут меняться в зависимости от нее. Для генерации линейных графиков в **seaborn** используется **relplot** функцию. Она также принимает **DataFrame**, **x**, **y** - столбцы.

Для визуализации выбирается тип **line**:

|  |
| --- |
| sns.relplot(x="latitude", y="median\_house\_value", kind="line", data=df) |

**Результат:**

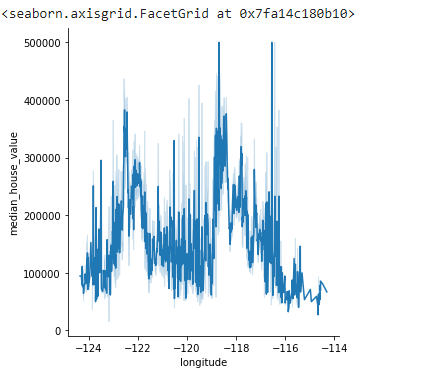
****

Можно видеть, что в определенных местах долготы цена за дома резко подскакивает.

Попробуем визуализировать longitude по отношения к median\_house\_value и поймем в чем же дело, почему цена так резко подскакивает.

|  |
| --- |
| sns.relplot(x = 'longitude', y = 'median\_house\_value', kind = 'line', data = df) |

**Результат:**

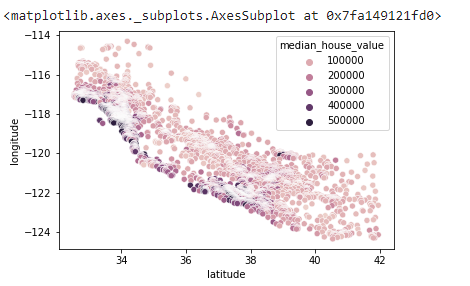
****

Можно видеть, что в определенных местах широты цена за дома также очень высока.

Используя точечный график можно визуализировать эти отношения с большей четкостью. Скорее всего резкий рост цен связан с близостью к ценному объекту, повышающему качество жизни, скорее всего побережью океана или реки.

|  |
| --- |
| sns.scatterplot(data=df, x="latitude", y="longitude", hue="median\_house\_value") |

**Результат:**

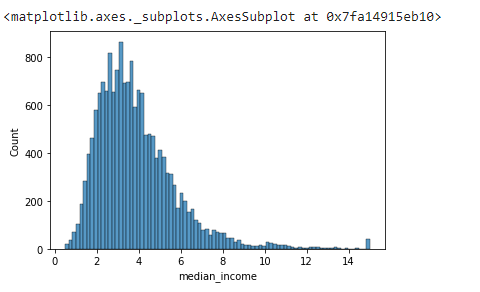
****

# Гистограмма

Способ представления табличных данных в графическом виде — в виде столбчатой диаграммы. По оси **x** обычно указывают значение, а по оси **y** - встречаемость(кол-во таких значений в выборке)

|  |
| --- |
| sns.histplot(data=df, x="median\_income") |

**Результат:**

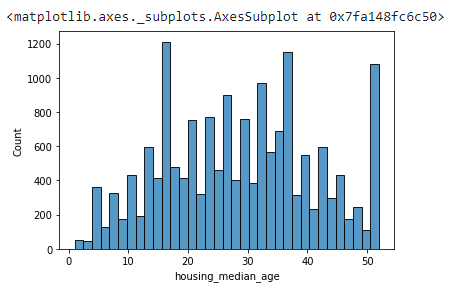


Можно видеть что у большинства семей доход находится между значениями 2 и 6. И только очень небольшое количество людей обладают доходом > 10.

Изобразим гистограмму по **housing\_median\_age**.

|  |
| --- |
| sns.histplot(data = df, x = 'housing\_median\_age') |

**Результат:**

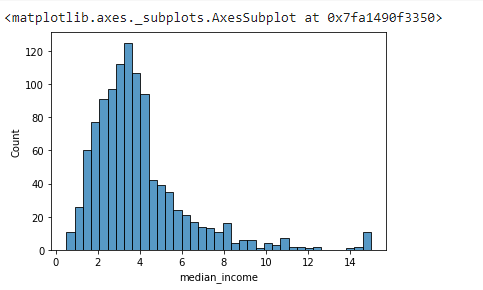


Распределение по возрасту более равномерное. Большую часть жителей составляют люди в возрасте от 20 до 40 лет. Но и молодежи не мало. Также очень много пожилых людей > 50 лет медианный возраст.

Давайте посмотрим медианный доход у пожилых жителей.

|  |
| --- |
| sns.histplot(data=df[df['housing\_median\_age']>50], x="median\_income") |

**Результат:**

****

Большого отличия от популяции в целом не наблюдается. Скорее всего это местные жители.

Давайте разобьем возрастные группы на 3 категории те кто моложе 20 лет, от 20 до 50 и от 50, чтобы посмотреть влияет ли это на доход.

|  |
| --- |
| df.loc[df['housing\_median\_age'] <= 20, 'age\_group'] = 'Молодые'  df.loc[(df['housing\_median\_age'] > 20) & (df['housing\_median\_age'] <= 50), 'age\_group'] = 'Ср. возраст'  df.loc[df['housing\_median\_age'] > 50, 'age\_group'] = 'Пожилые' |

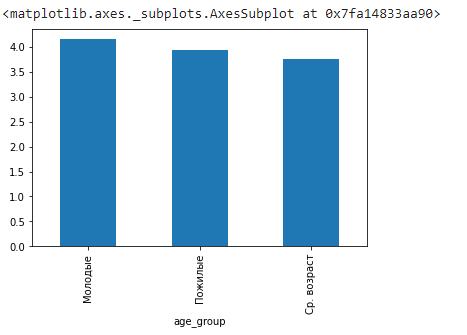
Что в этом случае происходит внутри таблицы? Добавился новый столбец **age\_group**, в котором будет указана соответствующая категория.



Применим **group\_by**, чтобы получить среднее значение.

|  |
| --- |
| df.groupby('age\_group')['median\_income'].mean().plot(kind='bar') |

**Результат:**

****

Молодые оказываются самой богатой группой населения. Но отличие в доходе не значительное.

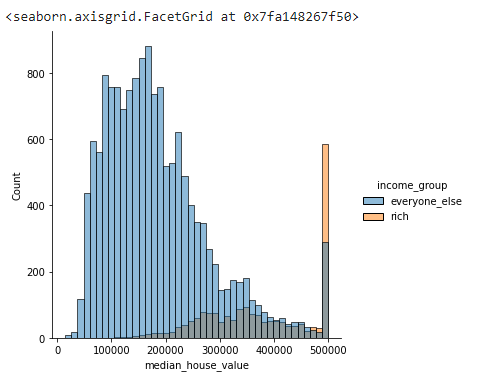
**Seaborn** так же позволяет нам смотреть распределение по многим параметрам. Давайте поделим группы по доходам на 2. Те у кого медианный доход выше 6 и те у кого меньше. Изобразим дополнительное измерение с помощью оттенка в виде возрастных групп и групп по доходам.

|  |
| --- |
| df.loc[df['median\_income'] > 6, 'income\_group'] = 'rich'  df.loc[df['median\_income'] < 6, 'income\_group'] = 'everyone\_else' |



|  |
| --- |
| sns.displot(df, x="median\_house\_value", hue="income\_group") |

**Результат:**

****

# Итоги:

Анализ данных должен предоставлять информацию и инсайт, которые не видны невооруженным взглядом. В этом и есть красота аналитики. В данном случае можно сделать следующий выводы. Стоимость домов напрямую зависит от их расположения, в определенной полосе(скорее всего побережье) цена на дома высокая. Чем выше доход, тем больше шанс, что человек проживает в богатом районе. Распределение по возрастам примерно одинаковое во всех группах доходов. Ну и очевидно чем больше людей, тем больше семей, и соответственно комнат и спален.