МАШИННОЕ ОБУЧЕНИЕ

Практикум

Ставрополь, 2023

Машинное обучение. Практикум [Электронный ресурс]: учебно-методическое пособие. – СтГАУ, 2023. – 41 с.

Машинное обучение. Учебно-методическое пособие для студентов и, которые хотят овладеть навыками машинного обучения и применять их на практике. В пособии подробно рассмотрены основные методы машинного обучения, приведены примеры их использования в различных приложениях и

представлены практические задания. Особое внимание уделено выбору признаков, регуляризации моделей, работе с несбалансированными данными

и управлению ошибками. Пособие также содержит описание основных инструментов и библиотек машинного обучения, таких как Python, NumPy,

Pandas, Scikit-learn и TensorFlow, а также практические советы по работе с данными. В целом, это полезный ресурс для тех, кто хочет научиться применять методы машинного обучения на практике.

**ЧАСТЬ IV. МНОГОВАРИАНТНОЕ ОБУЧЕНИЕ**

**ЗАНЯТИЕ 1 (2 часа)**

**Шаг 1. Описание проекта**

Вы работаете в интернет-магазине «Игротека», который продаёт по всему миру компьютерные игры.

Из открытых источников доступны исторические данные о продажах игр, оценки пользователей и экспертов, жанры и платформы (например, Xbox или PlayStation).

Вам нужно выявить определяющие успешность игры закономерности. Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании.

Перед вами, допустим, данные до 2016 года. Представим, что сейчас декабрь 2016 г., и вы планируете кампанию на 2017-й. Нужно отработать принцип работы с данными. Не важно, прогнозируете ли вы продажи на 2017 год по данным 2016-го или же 2027-й — по данным 2026 года.

**Описание данных games.csv**

• Name — название игры

• Platform — платформа

• Year\_of\_Release — год выпуска

• Genre — жанр игры

• NA\_sales — продажи в Северной Америке (миллионы долларов)

• EU\_sales — продажи в Европе (миллионы долларов)

• JP\_sales — продажи в Японии (миллионы долларов)

• Other\_sales — продажи в других странах (миллионы долларов)

• Critic\_Score — оценка критиков (от 0 до 100)

• User\_Score — оценка пользователей (от 0 до 10)

• Rating — рейтинг от организации ESRB (англ. Entertainment Software Rating Board). Эта ассоциация определяет рейтинг компьютерных игр и присваивает им подходящую возрастную категорию.

Данные за 2016 год могут быть неполными.

**ИНСТРУКЦИЯ ПО ВЫПОЛНЕНИЮ ПРОЕКТА**

**Шаг 2. Откройте файл с данными и изучите общую информацию**

Путь к файлу: /datasets/games.csv или games.csv в папке Anaconda3

**Шаг 3. Подготовьте данные**

• Замените названия столбцов (приведите к нижнему регистру);

• Преобразуйте данные в нужные типы. Опишите, в каких столбцах заменили тип данных и почему;

• Обработайте пропуски при необходимости:

- Объясните, почему заполнили пропуски определённым образом или почему не стали это делать;

- Опишите причины, которые могли привести к пропускам;

- Обратите внимание на аббревиатуру 'tbd' в столбцах с рейтингом. Поясните, как обработать это значение.

• Посчитайте суммарные продажи во всех регионах и запишите их в отдельный столбец.

**Шаг 4. Проведите исследовательский анализ данных**

• Посмотрите, сколько игр выпускалось в разные годы. Важны ли данные за все периоды?

• Посмотрите, как менялись продажи по платформам. Выберите платформы с наибольшими суммарными продажами и постройте распределение по годам. Найдите популярные в прошлом платформы, у которых сейчас продажи на нуле. За какой характерный период появляются новые и исчезают старые платформы?

• Определите, данные за какой период нужно взять, чтобы исключить значимое искажение распределения по платформам в 2016 году.

• Далее работайте только с данными, которые вы определили. Не учитывайте данные за предыдущие годы.

• Какие платформы лидируют по продажам, растут или падают? Выберите несколько потенциально прибыльных платформ.

• Постройте график «ящик с усами» по глобальным продажам каждой игры и разбивкой по платформам. Велика ли разница в продажах? А в средних продажах на разных платформах? Опишите результат.

• Посмотрите, как влияют на продажи внутри одной популярной платформы отзывы пользователей и критиков. Постройте диаграмму рассеяния и посчитайте корреляцию между отзывами и продажами. Сформулируйте выводы и соотнесите их с продажами игр на других платформах.

• Посмотрите на общее распределение игр по жанрам. Что можно сказать о самых прибыльных жанрах? Выделяются ли жанры с высокими и низкими продажами?

**Шаг 5. Составьте портрет пользователя каждого региона**

Определите для пользователя каждого региона (NA, EU, JP):

• Самые популярные платформы (топ-5). Опишите различия в долях продаж.

• Самые популярные жанры (топ-5). Поясните разницу.

• Влияет ли рейтинг ESRB на продажи в отдельном регионе?

**Шаг 6. Проведите исследование статистических показателей**

• Как изменяется пользовательский рейтинг и рейтинг критиков в различных жанрах? Посчитайте среднее количество, дисперсию и стандартное отклонение. Постройте гистограммы. Опишите распределения

**Шаг 7. Проверьте гипотезы**

• Средние пользовательские рейтинги платформ Xbox One и PC одинаковые;

• Средние пользовательские рейтинги жанров Action (англ. «действие») и Sports (англ. «виды спорта») разные. Задайте самостоятельно пороговое значение alpha. Поясните: • Как вы сформулировали нулевую и альтернативную гипотезы; • Какой критерий применили для проверки гипотез и почему.

**Шаг 8. Напишите общий вывод**

Оформление: Выполните задание в Jupyter Notebook. Заполните программный код в ячейках типа code, текстовые пояснения — в ячейках типа markdown. Примените форматирование и заголовки.

Как будут проверять мой проект? Мы подготовили критерии оценки проекта. Прежде чем решать кейс, внимательно изучите их. На что обращают внимание наставники, когда проверяют ваш проект:

• Как вы описываете выявленные в данных проблемы?

• Как готовите датасет к анализу?

• Какие строите графики для распределений и как их объясняете?

• Как рассчитываете стандартное отклонение и дисперсию?

• Формулируете ли альтернативную и нулевую гипотезы?

• Какие методы применяете, чтобы их проверить?

• Объясняете результат проверки гипотезы или нет?

• Соблюдаете ли структуру проекта и поддерживаете аккуратность кода?

• Какие выводы делаете?

• Оставляете ли комментарии к шагам?

**Цель работы**

Выявить определяющие успешность игры закономерности. Это позволит сделать ставку на потенциально популярный продукт и спланировать рекламные кампании.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import scipy

from collections import Counter

from scipy.stats import ttest\_1samp, ttest\_ind

from scipy.stats import levene

**Шаг 2. Откройте файл с данными и изучите общую информацию**

df **=** pd**.**read\_csv('/datasets/games.csv')

df**.**head(5)

или

df **=** pd**.**read\_csv('games.csv')

df**.**head(5)

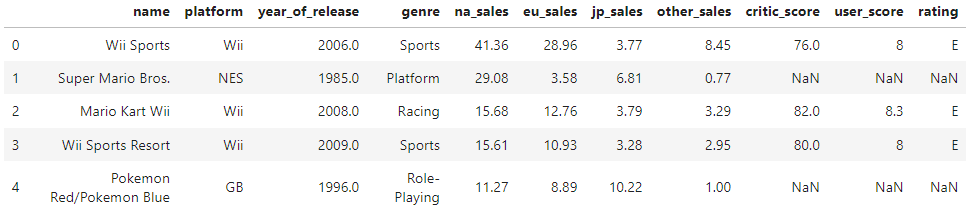
**Шаг 3. Подготовьте данные**

Замените названия столбцов (приведите к нижнему регистру)

df.columns = df.columns.map(lambda x: x.lower())

df.head()

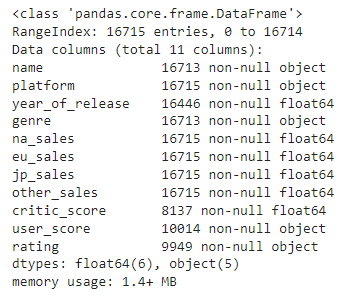
Результат:



Преобразуйте данные в нужные типы. Опишите, в каких столбцах заменили тип данных и почему.

df.info()

Возможный результат:



Кандидаты на изменение типа:

year\_of\_release – все года целочисленны, а float64 требует больше памяти

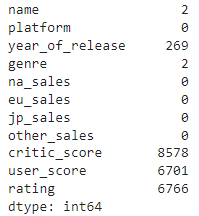
user\_score – там находятся строковые значения, которые пока надо обработать.

Пока надо обработать столбцы. Потом привести к нужному типу

Обработайте пропуски при необходимости

#df.info()

df.isnull().sum()



**Работаем с name**

print('До выполнения работ:', df['name'].isnull().sum())

До выполнения работ: 2

df = df.dropna(subset=['name']).reset\_index(drop=True)

#df.isnull().sum()

print('После выполнения работ:', df['name'].isnull().sum())

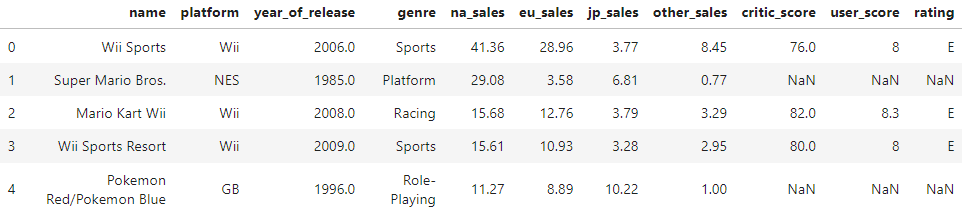
После выполнения работ: 0

**Работаем с year\_of\_release**

Мы просмотрели датасет и заметили, что игры обычно выходят в один год для разных платформ. Поэтому можно заполнить пропущенные данные в этом столбце просто сопоставив с другими платформами тех же игр:

df.head()

Результат:



print('До выполнения работ:', df['year\_of\_release'].isnull().sum())

До выполнения работ: 269

df['year\_of\_release'] = df.groupby('name')['year\_of\_release'].transform(lambda x: x.fillna(x.mean()))

print('После выполнения работ:', df['year\_of\_release'].isnull().sum())

После выполнения работ: 146

Таким путем мы смогли заполнить 123 пропуска в данном столбце.

**Работаем с critic\_score**

print('До выполнения работ:', df['critic\_score'].isnull().sum())

До выполнения работ: 8576

Заполним аналогично предыдущему варианту.

df['critic\_score'] = df.groupby('name')['critic\_score'].transform(lambda x: x.fillna(x.mean()))

print('После выполнения работ:', df['critic\_score'].isnull().sum())

После выполнения работ: 7615

#df = df.dropna(subset = ['critic\_score']).reset\_index(drop=True)

**Работаем с user\_score**

У user\_score есть значения tbd. Надо их преобразовать

df.loc[df.user\_score=='tbd', 'user\_score'] = np.nan

df.user\_score = df.user\_score.astype('float')

print('До выполнения работ:', df['user\_score'].isnull().sum())

До выполнения работ: 9123

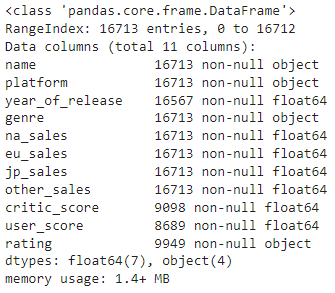
df['user\_score'] = df.groupby('name')['user\_score'].transform(lambda x: x.fillna(x.mean()))

print('После выполнения работ:', df['user\_score'].isnull().sum())

После выполнения работ: 8024

**Работаем с critic\_score - 2**

df.info()

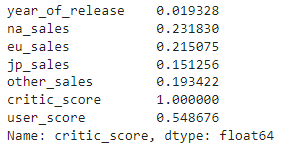


Как мы можем видеть из таблицы корреляций ниже, critic\_score тесно связан с user\_score

df.corr()['critic\_score'] - не работает

df['user\_score']. corr(df['critic\_score']) - поэтапно

df['na\_sales']. corr(df['critic\_score'])

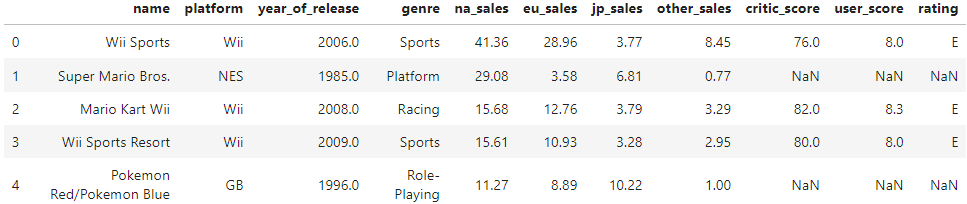


**Задумка:** **разобьем user\_score на когорты, а затем заполним critic\_score как среднее у каждой когорты**

print('До выполнения работ:', df['critic\_score'].isnull().sum())

До выполнения работ: 7615

df.head()

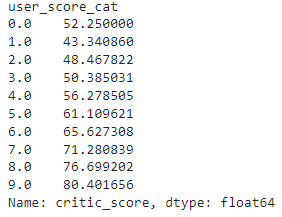


# создаем столбец когорт

df['user\_score\_cat'] = df['user\_score'].apply(lambda x: x//1)

# смотрим на среднее каждой когорты

df.groupby('user\_score\_cat').mean()['critic\_score']



# Я не особо умный, зато трудолюбывый :)

# Заполняем пропуски

df.loc[df['user\_score\_cat']==0, 'critic\_score'] = df.loc[df['user\_score\_cat']==0, 'critic\_score'].fillna(52.250000)

df.loc[df['user\_score\_cat']==1, 'critic\_score'] = df.loc[df['user\_score\_cat']==1, 'critic\_score'].fillna(43.340860)

df.loc[df['user\_score\_cat']==2, 'critic\_score'] = df.loc[df['user\_score\_cat']==2, 'critic\_score'].fillna( 48.46782)

df.loc[df['user\_score\_cat']==3, 'critic\_score'] = df.loc[df['user\_score\_cat']==3, 'critic\_score'].fillna( 50.38503)

df.loc[df['user\_score\_cat']==4, 'critic\_score'] = df.loc[df['user\_score\_cat']==4, 'critic\_score'].fillna(56.278505)

df.loc[df['user\_score\_cat']==5, 'critic\_score'] = df.loc[df['user\_score\_cat']==5, 'critic\_score'].fillna( 61.10962)

df.loc[df['user\_score\_cat']==6, 'critic\_score'] = df.loc[df['user\_score\_cat']==6, 'critic\_score'].fillna( 65.62730)

df.loc[df['user\_score\_cat']==7, 'critic\_score'] = df.loc[df['user\_score\_cat']==7, 'critic\_score'].fillna( 71.28083)

df.loc[df['user\_score\_cat']==8, 'critic\_score'] = df.loc[df['user\_score\_cat']==8, 'critic\_score'].fillna( 76.69920)

df.loc[df['user\_score\_cat']==9, 'critic\_score'] = df.loc[df['user\_score\_cat']==9, 'critic\_score'].fillna( 80.4016)

print('После выполнения работ:', df['critic\_score'].isnull().sum())

После выполнения работ: 7279

df.drop('user\_score\_cat', axis='columns', inplace=True)

**Работаем с rating**

print('До выполнения работы')

df.rating.isnull().sum()

До выполнения работы 6764

**Заполняем рейтинг модой**

df['rating'] = df.groupby('name')['rating'].transform(lambda x: x.fillna(Counter(x).most\_common()[0][0]))

print('После выполнения работы')

df.rating.isnull().sum()

После выполнения работы 6485

**Работаем с user\_score – 2**

Поскольку зависимости рейтинга от жанра нет, то у меня остается наиболее очевидный вариант для заполнения пропусков. Заполнить из значением "Нет рейтинга"

df.rating.fillna('No rating', inplace=True)

print('После выполнения работ:', df['rating'].isnull().sum())

После выполнения работ: 0

**Работаем с year\_of\_release – 2**

print('До выполнения работ:', df['year\_of\_release'].isnull().sum())

До выполнения работ: 146

Здесь я решил заполнить данные исходя из идеи того, что игра для PS4 врядли бы выпускалась в 2005 году.

df['year\_of\_release'] = df.groupby('platform')['year\_of\_release'].transform(lambda x: x.fillna(np.mean(x)))

print('После выполнения работ:', df['year\_of\_release'].isnull().sum())

После выполнения работ: 0

df.year\_of\_release = df.year\_of\_release.astype('int')

**-- Итог**

print('ИТОГО')

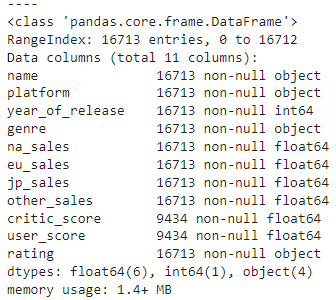
print('Количество пропущенных значений\n',df.isnull().sum(), sep='')

print('----')

print('Форма датасета',df.shape)

print('----')

df.info()



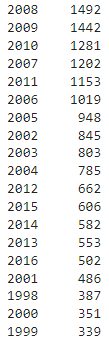
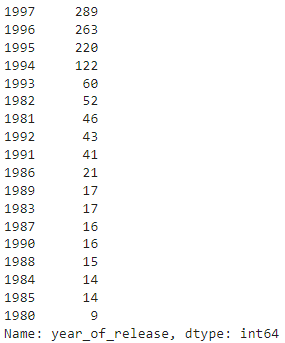
**Посчитайте суммарные продажи во всех регионах и запишите их в отдельный столбец.**

df['world\_sales'] = df['na\_sales'] + df['eu\_sales'] + df['jp\_sales'] + df['other\_sales']

**Шаг 4. Проведите исследовательский анализ данных**

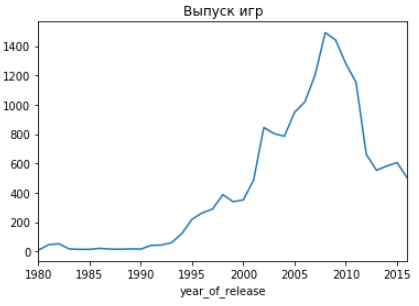
Посмотрим, сколько игр выпускалось в разные годы. Важны ли данные за все периоды?

df.year\_of\_release.value\_counts()

df.groupby('year\_of\_release')['name'].count().plot(title='Выпуск игр')

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f70f2feae90>



Обратим внимание на спад выпуска игр перед 2010 годом.

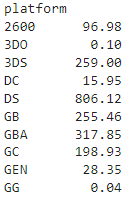
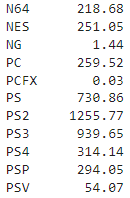
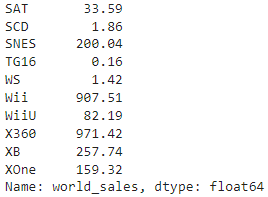
Был выполнен поиск информации в Яндексе и пришли к выводу, что на это также оказал влияние кризис 2008 года.

Допустим в октябре 2008 года цена акций EA упала на 75% с 50 до 15 долларов и оставалось на этом уровне ещё пять лет. А ведь EA была самым крупным издателем в мире. Акции Activision Blizzard, Take-Two, THQ, Gamestop и Nintendo провалились на 50%.

**Посмотрите, как менялись продажи по платформам. Выберите платформы с наибольшими суммарными продажами и постройте распределение по годам. Найдите популярные в прошлом платформы, у которых сейчас продажи на нуле. За какой характерный период появляются новые и исчезают старые платформы?**

Суммарные продажи у каждой платформы

df.groupby('platform').sum()['world\_sales']

Вот такой график продаж у 10 самых популярных платформ

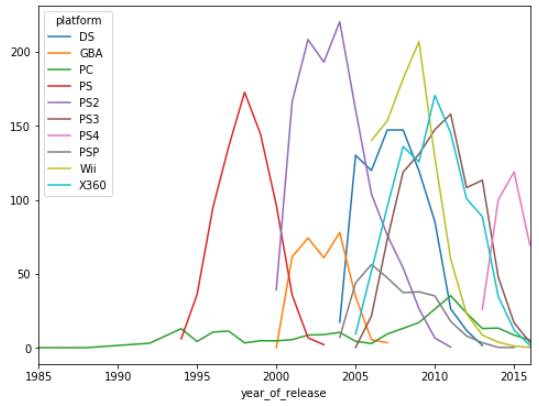
most\_popular = df.groupby('platform').sum()['world\_sales'].sort\_values(ascending=False).index.values[:10]

popular\_df = df[df.platform.isin(most\_popular)]

popular\_df = popular\_df.pivot\_table(index='year\_of\_release', columns='platform', values='world\_sales', aggfunc='sum')

popular\_df.plot(figsize=(8,6))

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f70f2f46910>



**Найдите популярные в прошлом платформы, у которых в 2016 году продажи на нуле**

Судя по графику выше это:

PS2

PS

Wii

X360

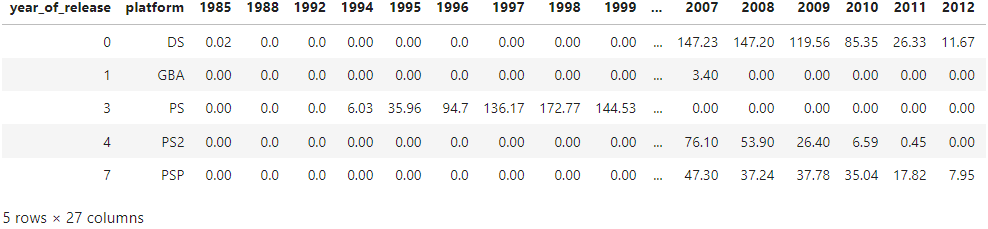
PS3

DS

XB

temp = popular\_df.T.fillna(0).reset\_index()

temp[temp[2016]==0]



**А теперь делаем окончательный вывод. Платформы с нулевым продажами в 2016 году это:**

DS

GBA

PS

PS2

PSP

**Судя по верхнему графику в среднем продолжительность жизни каждой платформы составляет 10 лет**

Хотя лично моё мнение в том, что PC будут еще долго в обороте.

Ниже мы решили провести более точные расчеты, а не опираться на данные графика

temp.set\_index('platform',inplace=True)

Здесь мы подсчитали количество годов с ненулевыми продажами для каждой популярной платформы. Из данного списка стоит исключить PC, так как он явно является выбросом.

life\_of\_platform = temp[temp!=0].count(axis=1)

print('Среднее время жизни платформы:',life\_of\_platform.drop('PC').mean())

Среднее время жизни платформы: 10.222222222222221

**Определите, данные за какой период нужно взять, чтобы исключить значимое искажение распределения по платформам в 2016 году.**

Почему должно быть искажение в 2016 году?

Я так понял имеется ввиду сильно смещение среднего арифметического из-за того, что до 1996 года продажи были мизерными

df.groupby('year\_of\_release').sum().world\_sales

|  |  |  |
| --- | --- | --- |
| year\_of\_release  1980 11.38  1981 35.68  1982 36.85  1983 16.80  1984 50.35  1985 53.95  1986 37.08  1987 21.70  1988 47.21  1989 73.45  1990 49.37  1991 32.23 | 1992 76.16  1993 45.99  1994 81.76  1995 89.15  1996 199.15  1997 201.07  1998 259.79  1999 251.25  2000 201.68  2001 333.30  2002 401.42  2003 372.37  2004 428.17 | 2005 460.21  2006 523.40  2007 606.16  2008 688.42  2009 664.32  2010 600.07  2011 510.59  2012 360.52  2013 362.58  2014 331.79  2015 267.98  2016 129.94  Name: world\_sales, dtype: float64 |

**Далее работайте только с данными, которые вы определили. Не учитывайте данные за предыдущие годы.**

Уменьшаем наш датасет. Теперь в новом датафрейме будут данные начиная с 2008 года

df\_striped = df[(df['year\_of\_release']>=2008)].copy()

**Какие платформы лидируют по продажам, растут или падают? Выберите несколько потенциально прибыльных платформ.**

Ниже мы можем видеть тот же самый график, который был наверху, но здесь сделали срез начиная от 2008 года.

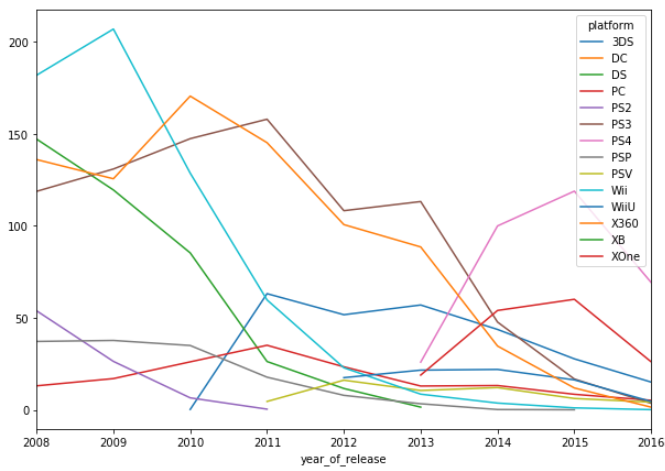
df\_striped.pivot\_table(index='platform',

columns='year\_of\_release',

values='world\_sales',

aggfunc='sum').T.plot(figsize=(10,7))

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f70f2deced0>



Перспективной платформой можно выделить:

PS4

PC

XOne

WiiU

Несмотря на то, что у ПК не особо большие продажи она демонстрирует стабильность

Судя по графику можно видеть отрицательный тренд почти на всем рынке видеоигр.

К 2016 году упал продажи у некогда очень продаваемых платформ:

PS3

X360

Wii

PC изначально не демонстрировал больших продаж, но все равно заметно снижение

**Постройте график «ящик с усами» по глобальным продажам каждой игры и разбивкой по платформам. Велика ли разница в продажах? А в средних продажах на разных платформах? Опишите результат.**

df\_striped.boxplot(by='platform', column='world\_sales', figsize=(10,8))

Результат:

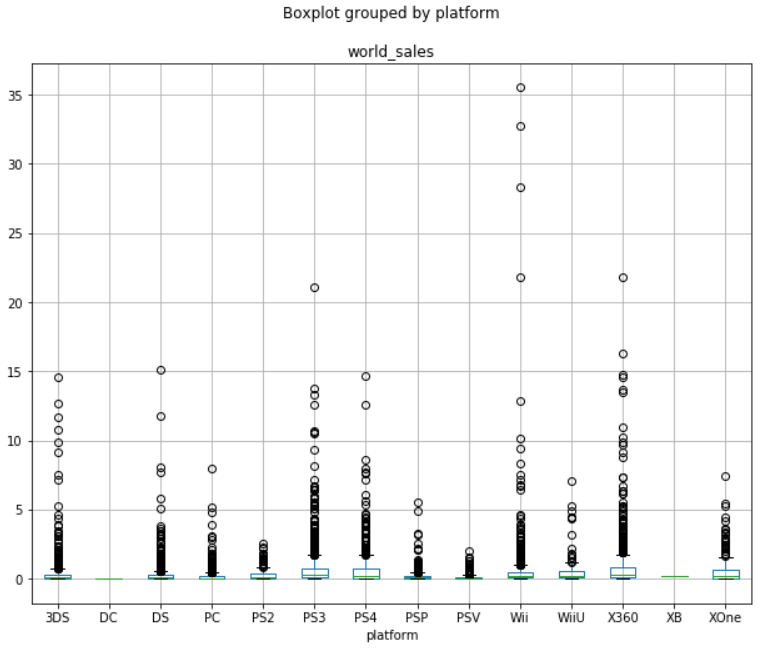
/opt/conda/lib/python3.7/site-packages/numpy/core/\_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray

return array(a, dtype, copy=False, order=order)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f70f2e5e490>

Далее вывод графика

plt.show()

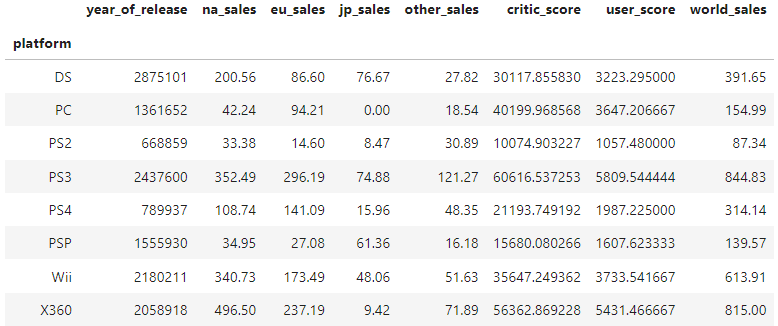


Разница в основном незначительна. С 1 по 3 квартиль разные платформы демонстрируют похожие продажи. То есть медианные продажи не будут сильно различаться, чего не скажешь о средних. На боксплотах наверху можно наблюдать какой серьезный выброс есть у Wii. Это игра Wii Sports, которая далеко обогнала все остальные игры

**Посмотрите, как влияют на продажи внутри одной популярной платформы отзывы пользователей и критиков. Постройте диаграмму рассеяния и посчитайте корреляцию между отзывами и продажами. Сформулируйте выводы и соотнесите их с продажами игр на других платформах.**

df\_popular = df\_striped[df\_striped.platform.isin(most\_popular)]

df\_popular.head()



popular\_platforms\_list = df\_popular\_grouped.index.values

popular\_platforms\_list

На экран выводится:

array(['DS', 'PC', 'PS2', 'PS3', 'PS4', 'PSP', 'Wii', 'X360'],

dtype=object)

fig, ax = plt.subplots(nrows=1, ncols=8, sharey=True, figsize=(30,6))

ax[0].scatter(data=df\_popular[df\_popular.platform=='DS'], x='user\_score',y='world\_sales')

ax[1].scatter(data=df\_popular[df\_popular.platform=='PC'], x='user\_score',y='world\_sales')

ax[2].scatter(data=df\_popular[df\_popular.platform=='PS2'], x='user\_score',y='world\_sales')

ax[3].scatter(data=df\_popular[df\_popular.platform=='PS3'], x='user\_score',y='world\_sales')

ax[4].scatter(data=df\_popular[df\_popular.platform=='PS4'], x='user\_score',y='world\_sales')

ax[5].scatter(data=df\_popular[df\_popular.platform=='PSP'], x='user\_score',y='world\_sales')

ax[6].scatter(data=df\_popular[df\_popular.platform=='Wii'], x='user\_score',y='world\_sales')

ax[7].scatter(data=df\_popular[df\_popular.platform=='X360'], x='user\_score',y='world\_sales')

fig.suptitle('Зависимость между user score и продажами у каждой платформы')

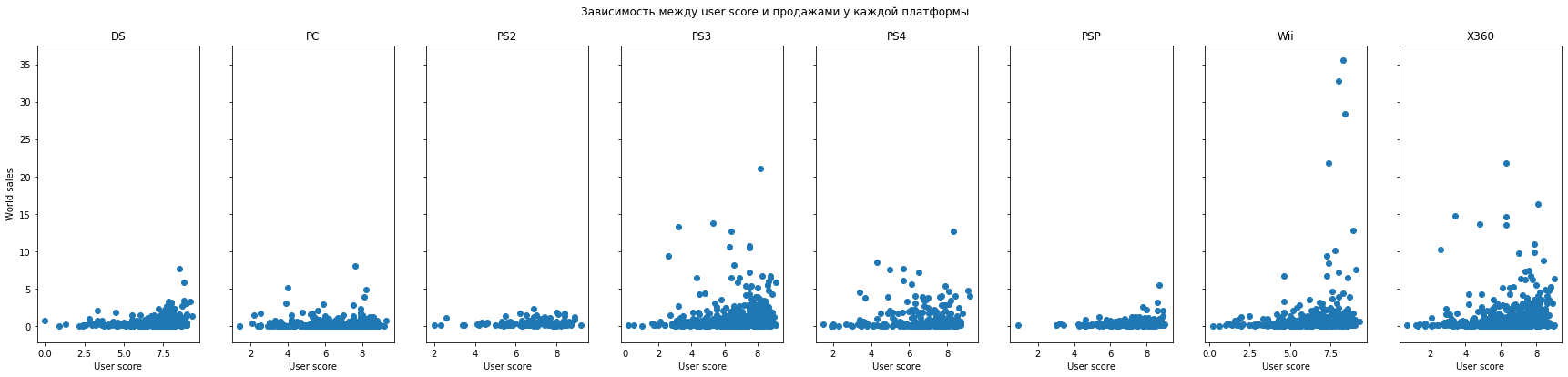
ax[0].set\_ylabel('World sales')

for i,e in enumerate(ax[:]):

e.set\_xlabel('User score')

platform\_name = popular\_platforms\_list[i]

e.set\_title(platform\_name)



fig, ax **=** plt**.**subplots(nrows**=**1, ncols**=**8, sharey**=True**, figsize**=**(30,6))

ax[0]**.**scatter(data**=**df\_popular[df\_popular**.**platform**==**'DS'], x**=**'critic\_score',y**=**'world\_sales')

ax[1]**.**scatter(data**=**df\_popular[df\_popular**.**platform**==**'PC'], x**=**'critic\_score',y**=**'world\_sales')

ax[2]**.**scatter(data**=**df\_popular[df\_popular**.**platform**==**'PS2'],x**=**'critic\_score',y**=**'world\_sales')

ax[3]**.**scatter(data**=**df\_popular[df\_popular**.**platform**==**'PS3'],x**=**'critic\_score',y**=**'world\_sales')

ax[4]**.**scatter(data**=**df\_popular[df\_popular**.**platform**==**'PS4'],x**=**'critic\_score',y**=**'world\_sales')

ax[5]**.**scatter(data**=**df\_popular[df\_popular**.**platform**==**'PSP'],x**=**'critic\_score',y**=**'world\_sales')

ax[6]**.**scatter(data**=**df\_popular[df\_popular**.**platform**==**'Wii'],x**=**'critic\_score',y**=**'world\_sales')

ax[7]**.**scatter(data**=**df\_popular[df\_popular**.**platform**==**'X360'],x**=**'critic\_score',y**=**'world\_sales')

fig**.**suptitle('Зависимость между critic score и продажами у каждой платформы')

ax[0]**.**set\_ylabel('World sales')

**for** i,e **in** enumerate(ax[:]):

e**.**set\_xlabel('Critic score')

platform\_name **=** popular\_platforms\_list[i]

e**.**set\_title(platform\_name)

Вывод на экран:

Text(0.5, 36.72222222222221, 'Critic score')

Text(0.5, 1.0, 'DS')

Text(0.5, 36.72222222222221, 'Critic score')

Text(0.5, 1.0, 'PC')

Text(0.5, 36.72222222222221, 'Critic score')

Text(0.5, 1.0, 'PS2')

Text(0.5, 36.72222222222221, 'Critic score')

Text(0.5, 1.0, 'PS3')

Text(0.5, 36.72222222222221, 'Critic score')

Text(0.5, 1.0, 'PS4')

Text(0.5, 36.72222222222221, 'Critic score')

Text(0.5, 1.0, 'PSP')

Text(0.5, 36.72222222222221, 'Critic score')

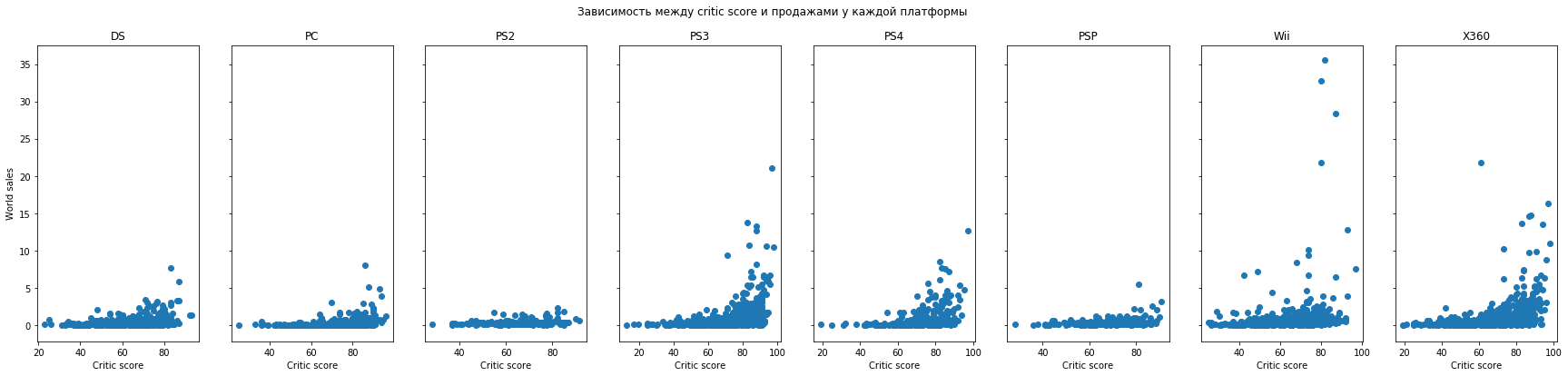
Text(0.5, 1.0, 'Wii')

Text(0.5, 36.72222222222221, 'Critic score')

Text(0.5, 1.0, 'X360')

plt.show()

Результат:



Для большинства платформ заметна положительная корреляция между critic\_score/user\_score и суммарным количеством продаж

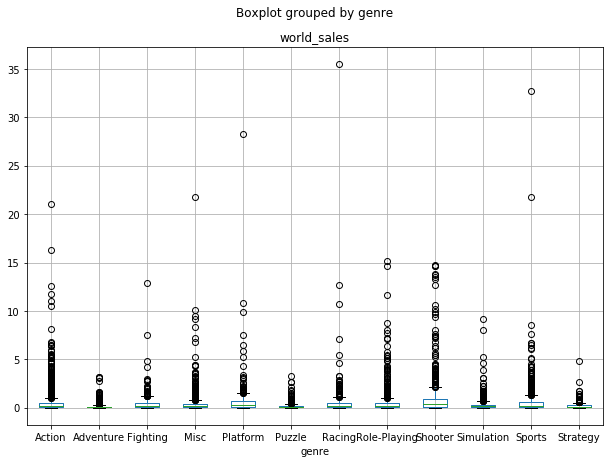
Примечателен тот факт, что для PS2 почти не заметна корреляция судя по графику.

**ЗАНЯТИЕ 2 (2 часа)**

Мы продолжаем обработку данных для машинного обучения.

**Посмотрите на общее распределение игр по жанрам. Что можно сказать о самых прибыльных жанрах? Выделяются ли жанры с высокими и низкими продажами?**

df\_striped.boxplot(by='genre', column='world\_sales', figsize=(10,7))



Если считать медианы, то наиболее прибыльными жанрами являются платформеры, шутеры и спорт. Однако некоторые жанры имеют приблизительно одинаковые медианные продажи

**Шаг 5. Составьте портрет пользователя каждого региона**

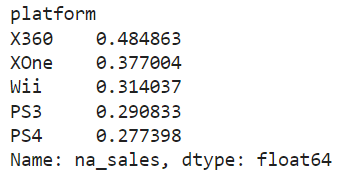
**Определите для пользователя каждого региона (NA, EU, JP):**

Самые популярные платформы (топ-5). Опишите различия в долях продаж.

**- NA**

na\_popular\_platforms = df\_striped.groupby('platform').na\_sales.mean().sort\_values(ascending=False)[:5]

na\_popular\_platforms

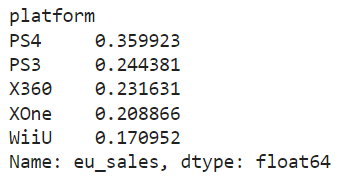


- EU

Самые популярные платформы у европейцев

eu\_popular\_platforms = df\_striped.groupby('platform').eu\_sales.mean().sort\_values(ascending=False)[:5]

eu\_popular\_platforms

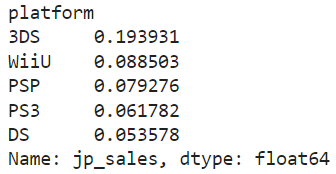


- JP

Самые популярные платформы у японцев

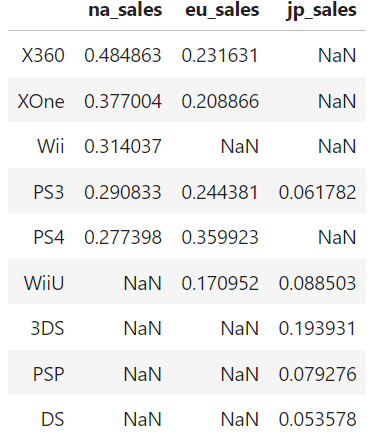
jp\_popular\_platforms = df\_striped.groupby('platform').jp\_sales.mean().sort\_values(ascending=False)[:5]

jp\_popular\_platforms

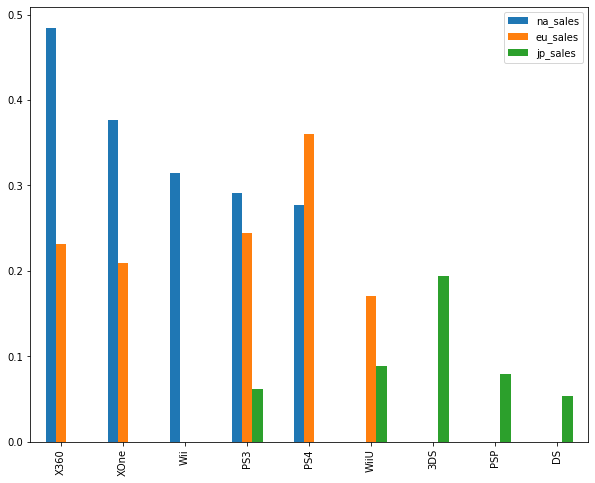


data\_to\_plot = pd.DataFrame(data=[na\_popular\_platforms,eu\_popular\_platforms, jp\_popular\_platforms]).T

data\_to\_plot

****

data\_to\_plot.plot(kind='bar',figsize=(10,8))



Выводы:

* Азиатский рынок наиболее обособлен от остальных.
* Европейский и американский во многом схожи.
* Судя по графику платформа Wii была нацелена именно на американский рынок, учитывая то, что на других рынках она оказалась в аутсайдерах.
* Самой популярной линейкой-платформ в США является - XBox
* Самой популярной линейкой-платформ в Европе является - Playstation
* Самой популярной линейкой-платформ в Японии является - DS(Nintendo)
* В платформе XBOX азиатские геймеры видимо вообще не заинтересованы

**Самые популярные жанры (топ-5). Поясните разницу.**

- NA

Самые популярные жанры у американцев

na\_popular\_genres = df\_striped.groupby('genre').na\_sales.mean().sort\_values(ascending=False)[:5]

na\_popular\_genres

- EU

Самые популярные жанры у европейцев

eu\_popular\_genres = df\_striped.groupby('genre').eu\_sales.mean().sort\_values(ascending=False)[:5]

eu\_popular\_genres

- JP

Самые популярные жанры у японцев

jp\_popular\_genres = df\_striped.groupby('genre').jp\_sales.mean().sort\_values(ascending=False)[:5]

jp\_popular\_genres

Сделайте выводы.

data\_to\_plot = pd.DataFrame(data=[na\_popular\_genres,eu\_popular\_genres, jp\_popular\_genres]).T

data\_to\_plot.plot(kind='bar')



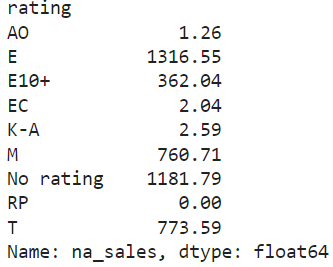
**Влияет ли рейтинг ESRB на продажи в отдельном регионе?**

- NA

Популярность рейтингов у американцев

na\_popular\_ratings = df.groupby('rating').sum().na\_sales

na\_popular\_ratings



Далее популярность у европейцев и азиатов.

**Вывод:**

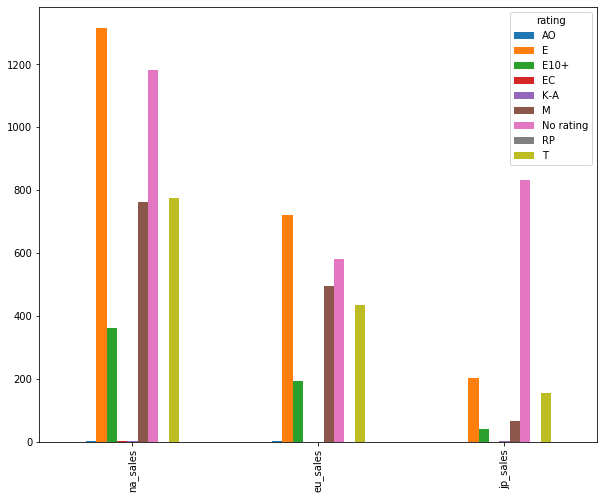
На всёх 3 рынках доминируют игры с рейтингом E.

Очень много данных у нас с неразмеченными рейтингами, что намного усложняет анализ продаж

data\_to\_plot = pd.DataFrame([na\_popular\_ratings, eu\_popular\_ratings, jp\_popular\_ratings])

data\_to\_plot.plot(kind='bar', figsize=(10,8))

Результат:



**Шаг 6. Проведите исследование статистических показателей**

**Как изменяется пользовательский рейтинг и рейтинг критиков в различных жанрах? Посчитайте среднее количество, дисперсию и стандартное отклонение. Постройте гистограммы. Опишите распределения**

**Пользовательский рейтинг**

**Медианный пользовательский рейтинг у каждого жанра**

Сначала для графика хотели взять сумму рейтинга у каждого жанра, но из-за огромного различия в количестве наблюдений каждого жанра взяли медиану.

#Гистограммы распределения оценок пользователей у каждой платформы

genres\_list = df\_striped.genre.unique().tolist()

fig, axes = plt.subplots(nrows=3, ncols=4, figsize=(20,15))

fig.suptitle('Гистограммы распределения оценок пользователей у каждой платформы')

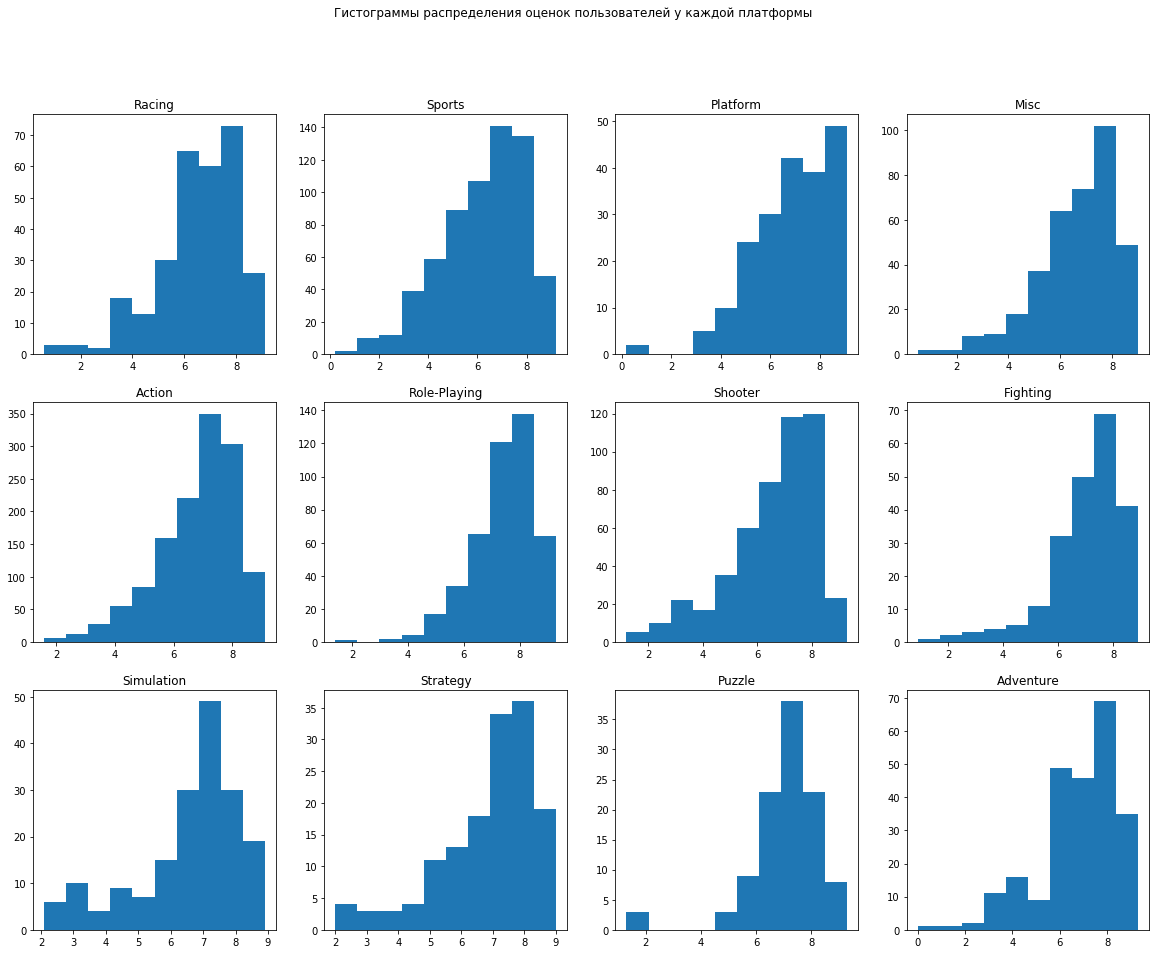
axes = axes.flatten()

num\_of\_bins = 10

for genre,ax in zip(genres\_list,axes):

ax.hist(df\_striped[df\_striped['genre']==genre].user\_score, bins=num\_of\_bins)

ax.set\_title(genre)



**Рейтинг критиков**

fig, axes = plt.subplots(nrows=3, ncols=4, figsize=(20,15))

fig.suptitle('Гистограммы распределения оценок критиков у каждой платформы')

axes = axes.flatten()

num\_of\_bins = 10

for genre,ax in zip(genres\_list,axes):

ax.hist(df\_striped[df\_striped['genre']==genre].critic\_score, bins=num\_of\_bins)

ax.set\_title(genre)

Сделать выводы.

# Шаг 8. Общий вывод

**ЗАНЯТИЕ 3 (2 часа)**

**Корреляция столбцов Pandas**

Корреляция имеет дело с силой и направленностью связи между двумя или более переменными. В pandas функция DataFrame.corr() может использоваться для получения корреляции между двумя или более столбцами в DataFrame. Корреляция используется для анализа силы и направления между двумя количественными переменными. Она обозначается R и значения между -1 и +1. А положительное значение для R указывает на позитивные ассоциации и отрицательное значение Р указывает на негативные ассоциации

**Примеры корреляции столбцов**

# Correlation between two columns of DataFrame.

corr=df['Fee'].corr(df['Discount'])

# Correlation between all the columns of DataFrame.

df2=df.corr()

**Пример 1.**

# Создать DataFrame.

import pandas as pd

import numpy as np

technologies= {

'Courses':["Spark","PySpark","Spark","Python","PySpark"],

'Fee' :[22000,25000,23000,24000,26000],

'Discount':[1500,1000,1200,800,1300],

'Duration':['30days','50days','30days','35days','40days']

}

df = pd.DataFrame(technologies)

print(df)

**Корреляция между двумя столбцами фрейма данных**

Вы можете увидеть корреляцию между двумя столбцами фрейма данных pandas с помощью DataFrame.corr() функции. pandas.DataFrame.corr() Используется для нахождения попарной корреляции всех столбцов в фрейме данных. Например, давайте посмотрим, какова корреляция между Fee и Discount.

# Correlation between two columns of DataFrame.

corr=df['Fee'].corr(df['Discount'])

print(corr)

**Задание 1.**

1.Создать фрейм данных.

2.Получить корреляцию между переменными.

Найти корреляцию между столбцами или сериями с помощью pandas. Мы можем использовать функцию pandas corr() для нахождения корреляций столбцов чисел или корреляции между несколькими рядами.

У нас есть следующий фрейм данных.

df = pd.DataFrame({'Name': ['Jim', 'Sally', 'Bob', 'Sue', 'Jill', 'Larry'],

'Weight': [160.20, 160.20, 209.45, 150.35, 187.52, 187.52],

'Height': [50.10, 68.94, 71.42, 48.56, 59.37, 63.42] })

**Задание 2.**

Чтобы вычислить корреляцию между двумя переменными в Python, мы можем использовать функцию Numpy corrcoef()

1.Создать массив из 50 случайных целых чисел от 0 до 10.

2.Создать положительно коррелированный массив с некоторым случайным шумом.

3.Определить тип корреляции.

4.Проверить, является ли эта корреляция статистически значимой, рассчитать p-значение.

Генерация случайных чисел

Numpy random seed() в Python используется для генерации псевдослучайных чисел на основе начального значения. Псевдослучайное число - это число, которое сортирует случайные числа, но на самом деле они не являются случайными числами.

Numpy random seed - это числовое значение, которое генерирует новый набор или повторяет псевдослучайные числа. Значение в numpy random seed сохраняет состояние случайности. Если мы вызываем начальную функцию, используя значение 1 несколько раз, компьютер отображает одни и те же случайные числа. Если значение не указано в numpy random seed, то компьютер будет использовать текущее системное время в секундах или миллисекундах в качестве начального значения для генерации другого набора случайных чисел.

Функция NumPy random() не генерирует ‘по-настоящему’ случайные числа, но мы использовали ее для генерации псевдослучайных чисел.

import numpy as np

np.random.seed(100)

#создать массив из 50 случайных целых чисел от 0 до 10

var1 = np.random.randint(0, 10, 50)

#создать положительно коррелированный массив с некоторым случайным шумом

var2 = var1 + np.random.normal(0, 10, 50)

#рассчитать корреляцию между двумя массивами

np.corrcoef(var1, var2)

# [[ 1. 0.335]

# [ 0.335 1. ]]

По умолчанию эта функция создает матрицу коэффициентов корреляции. Если бы мы только хотели вернуть коэффициент корреляции между двумя переменными, мы могли бы используйте следующий синтаксис:

np.corrcoef(var1, var2)[0,1]

#0.335

Чтобы проверить, является ли эта корреляция статистически значимой, мы можем рассчитать p-значение, связанное с коэффициентом корреляции Пирсона, с помощью Scipy pearsonr(), которая возвращает коэффициент корреляции Пирсона вместе с двусторонним p-значением.

from scipy.stats.stats import pearsonr

pearsonr(var1, var2)

#(0.335, 0.017398)

**Задание 3.**

1.Сгенерировать DataFrame с 3 переменными.

2.Вычислить корреляцию между переменными в Pandas DataFrame используя data.corr().

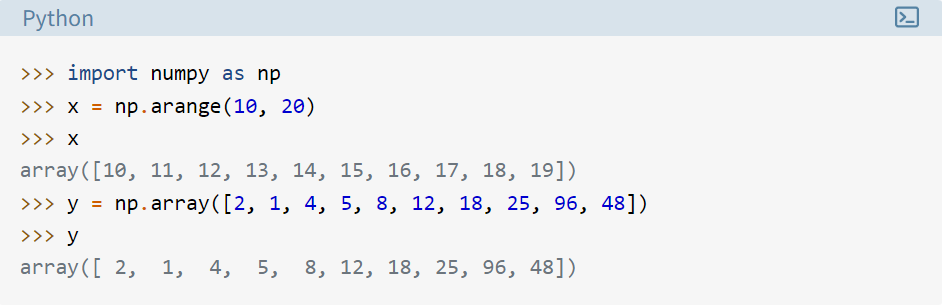
Если вас интересует вычисление корреляции между несколькими переменными в Pandas DataFrame, вы можете просто использовать функцию .corr()

**Пример: Вычисление корреляции NumPy**

Существует несколько статистических данных, которые вы можете использовать для количественной оценки корреляции. В этом руководстве вы узнаете о трех коэффициентах корреляции:

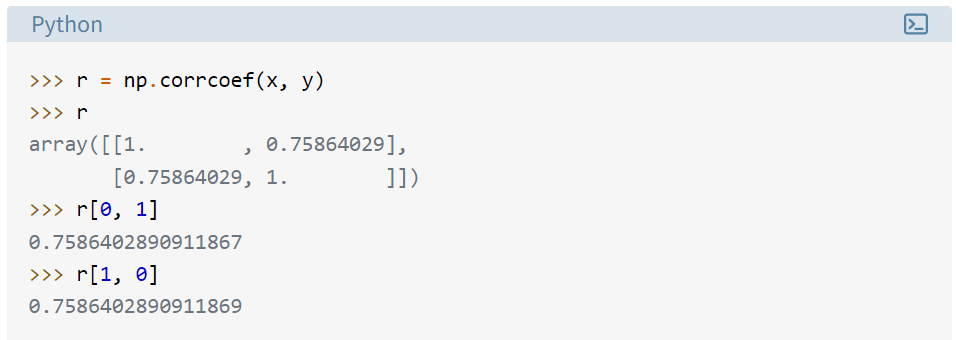
* Pearson's r
* Rho Спирмена
* Тау Кендалла

Коэффициент Пирсона измеряет линейную корреляцию, в то время как коэффициенты Спирмена и Кендалла сравнивают ранги данных. Существует несколько функций и методов корреляции NumPy, SciPy и pandas, которые вы можете использовать для вычисления этих коэффициентов. Вы также можете использовать Matplotlib для удобной иллюстрации результатов.

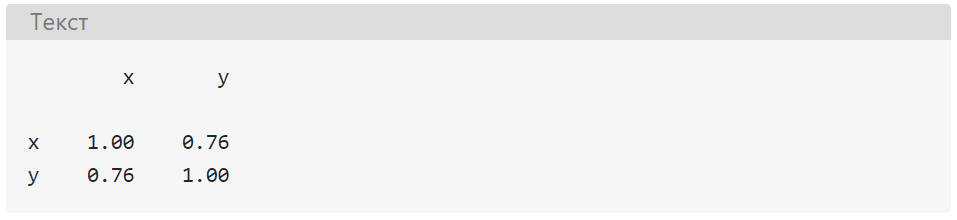


Здесь вы используете np.arange() для создания массива x целых чисел от 10 (включительно) до 20 (исключительных). Затем вы используете np.array() для создания второго массива y, содержащего произвольные целые числа.

Если у вас есть два массива одинаковой длины, вы можете вызвать np.corrcoef() с обоими массивами в качестве аргументов:

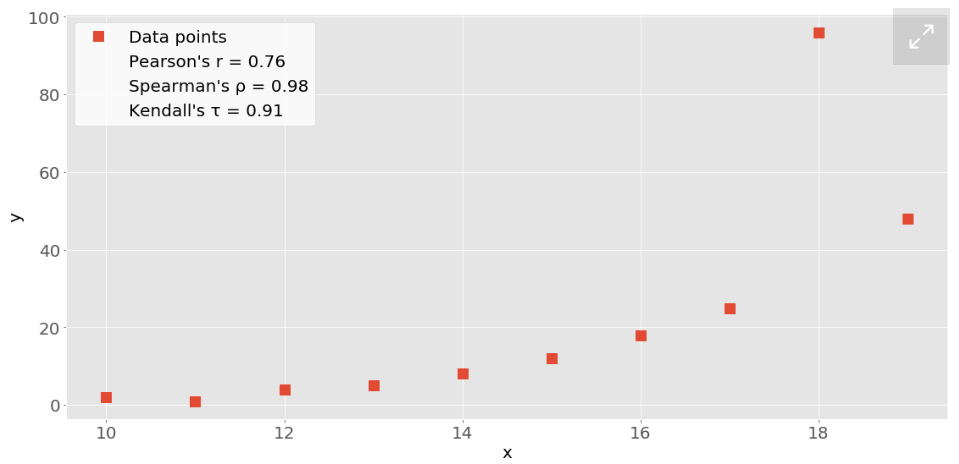


corrcoef() возвращает корреляционную матрицу, которая представляет собой двумерный массив с коэффициентами корреляции. Вот упрощенная версия корреляционной матрицы, которую вы только что создали:



Значения на главной диагонали корреляционной матрицы (верхний левый и нижний правый) равны 1. Верхнее левое значение соответствует коэффициенту корреляции для x и x, в то время как нижнее правое значение является коэффициентом корреляции для y и y. Они всегда равны 1.

Однако обычно вам нужны нижние левые и верхние правые значения корреляционной матрицы. Эти значения равны и оба представляют собой коэффициент корреляции Пирсона для x и y. В данном случае он равен приблизительно 0,76.



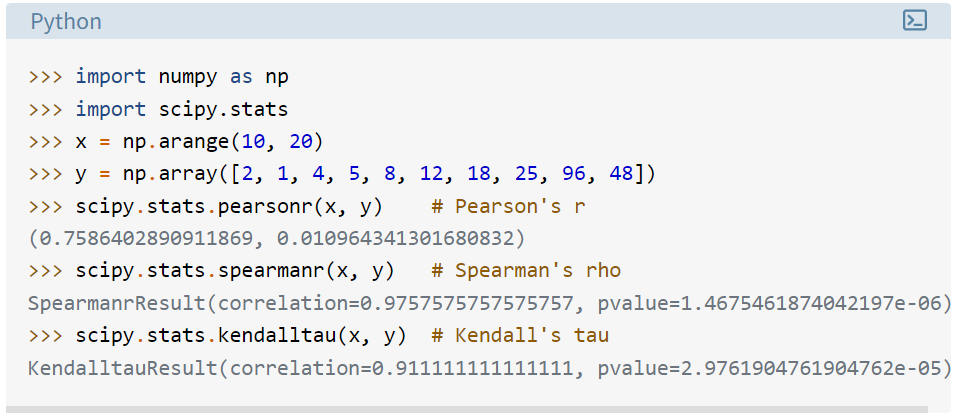
На этом рисунке показаны точки данных и коэффициенты корреляции для приведенного выше примера. Красные квадраты – это точки данных. Как вы можете видеть, на рисунке также показаны значения трех коэффициентов корреляции.

**Пример: Вычисление корреляции SciPy**

В SciPy также есть множество статистических процедур, содержащихся в scipy.stats. Вы можете использовать следующие методы для вычисления трех коэффициентов корреляции, которые вы видели ранее:

* pearsonr()
* spearmanr()
* kendalltau()

Вот как вы могли бы использовать эти функции в Python:



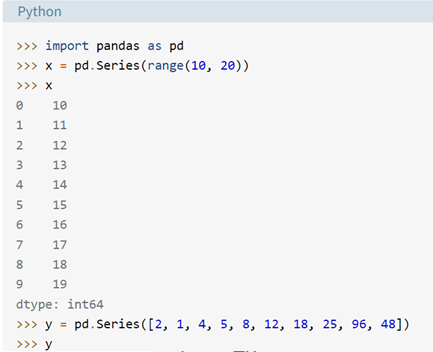
Обратите внимание, что эти функции возвращают объекты, содержащие два значения:

1. Коэффициент корреляции
2. p-значение

P-значение используется в статистических методах при проверке гипотезы. P-значение является важным показателем, для интерпретации которого требуются глубокие знания вероятности и статистики.

**Пример: Вычисление корреляции pandas**

Pandas в некоторых случаях более удобен для вычисления статистики, чем NumPy и SciPy. Он предлагает статистические методы для Series и DataFrame экземпляров. Например, учитывая два Series объекта с одинаковым количеством элементов, вы можете вызвать .corr() один из них с другим в качестве первого аргумента:





Здесь вы используете .corr() для вычисления всех трех коэффициентов корреляции. Вы определяете желаемую статистику с помощью параметра method, который может принимать одно из нескольких значений:

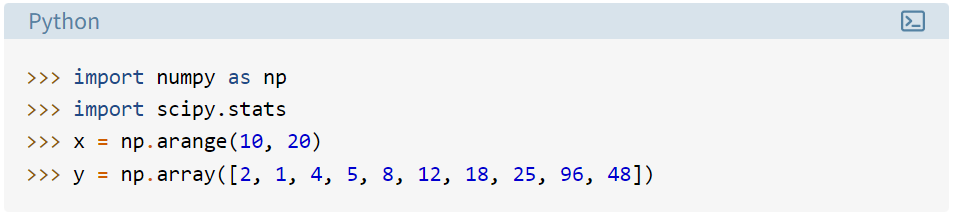
* 'pearson'
* 'spearman'
* 'kendall'
* Вызываемый

**Вызываемым** может быть любая функция, метод или объект с.\_\_call\_\_(), который принимает два одномерных массива и возвращает число с плавающей запятой.

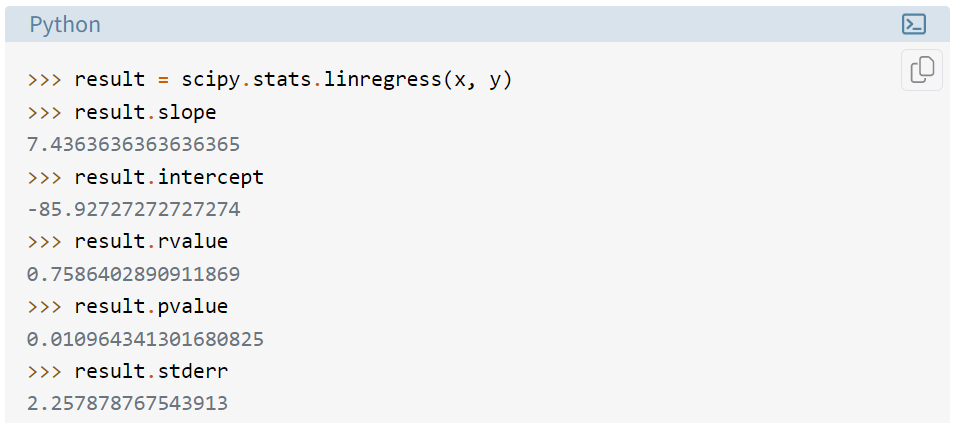
**Линейная регрессия: реализация SciPy**

Линейная регрессия - это процесс нахождения линейной функции, которая максимально приближена к фактической взаимосвязи между объектами. Другими словами, вы определяете линейную функцию, которая наилучшим образом описывает связь между объектами. Эта линейная функция также называется линией регрессии.

Вы можете реализовать линейную регрессию с помощью SciPy. Вы получите линейную функцию, которая наилучшим образом аппроксимирует взаимосвязь между двумя массивами, а также коэффициент корреляции Пирсона. Для начала вам сначала нужно импортировать библиотеки и подготовить некоторые данные для работы:



Здесь импортируем numpy and scipy.stats и определяете переменные x and y. Вы можете использовать scipy.stats.linregress() для выполнения линейной регрессии для двух массивов одинаковой длины. Вы должны предоставить массивы в качестве аргументов и получить выходные данные, используя точечную нотацию:



Вот и все! Мы завершили линейную регрессию и получили следующие результаты:

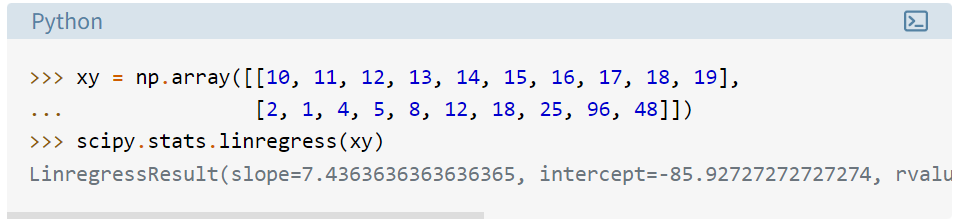
slope: наклон линии регрессии

intercept: перехват линии регрессии

pvalue: значение p

stderr: стандартная ошибка оцененного градиента

Вы узнаете, как визуализировать эти результаты, в более позднем разделе. Вы также можете предоставить один аргумент для linregress(), но это должен быть двумерный массив с одним измерением длины:



Результат точно такой же, как в предыдущем примере, потому что xy содержит те же данные, что и x и y вместе взятые. linregress() взял первую строку xy как одну функцию, а вторую строку - как другую функцию.

**Примечание:** В приведенном выше примере scipy.stats.linregress() строки рассматриваются как объекты, а столбцы - как наблюдения. Это потому, что строк две.

Обычная практика в машинном обучении противоположна: строки - это наблюдения, а столбцы - объекты. Многие библиотеки машинного обучения, такие как pandas, Scikit-Learn, Keras и другие, следуют этому соглашению.

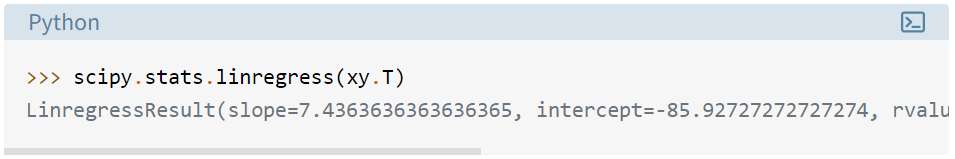
Вы должны внимательно следить за тем, как отображаются наблюдения и функции всякий раз, когда вы анализируете корреляцию в наборе данных.

linregress() вернет тот же результат, если вы предоставите transpose of xy или массив NumPy с 10 строками и двумя столбцами. В NumPy вы можете транспонировать матрицу многими способами.

Вот как вы могли бы транспонировать xy:

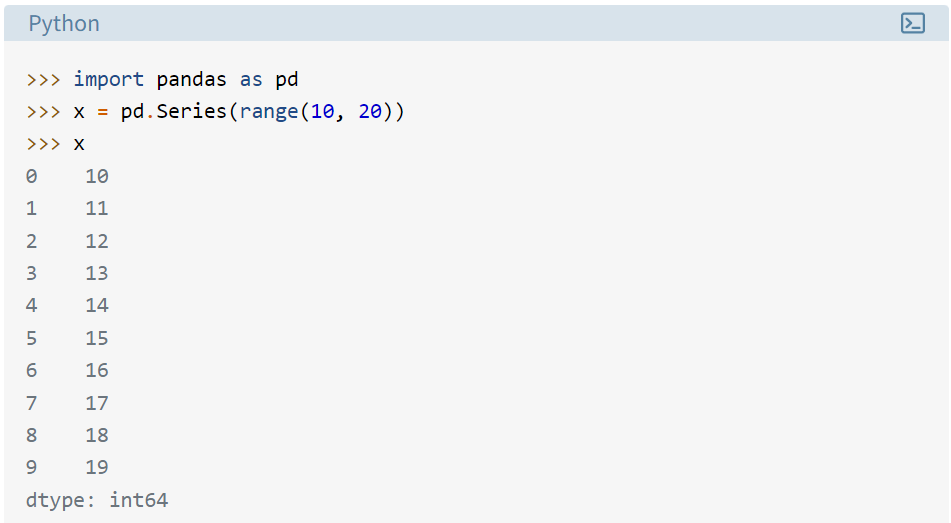


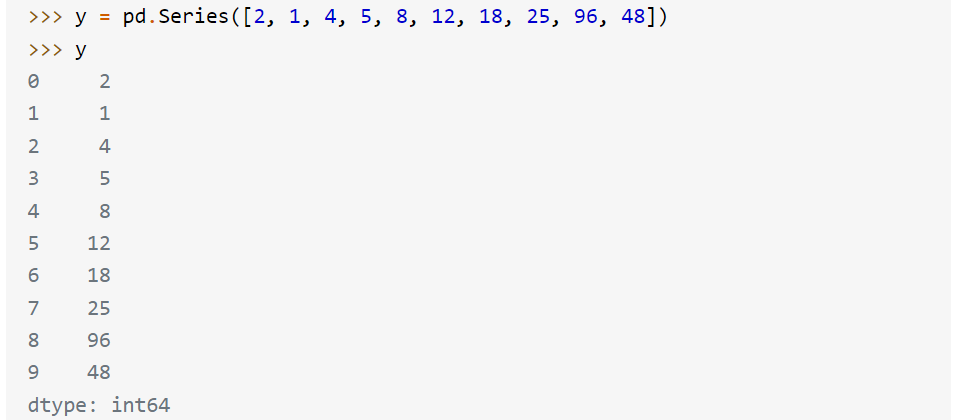
Теперь, когда вы знаете, как получить транспонирование, вы можете передать его в linregress(). В первом столбце будет одна функция, а во втором столбце - другая функция:

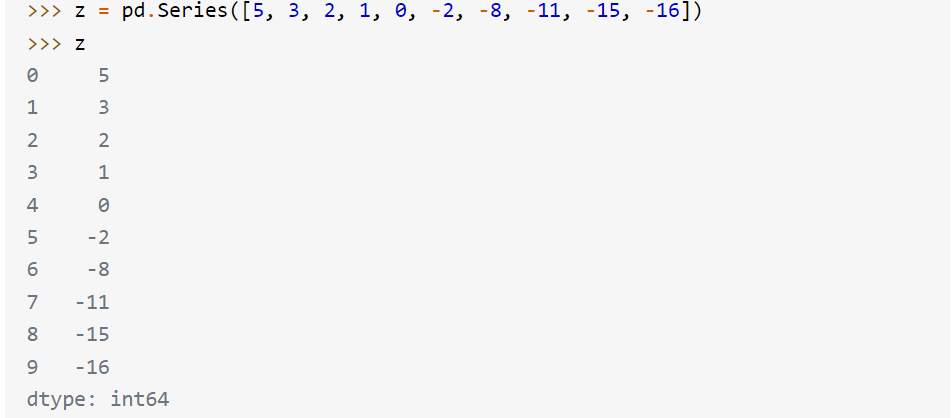


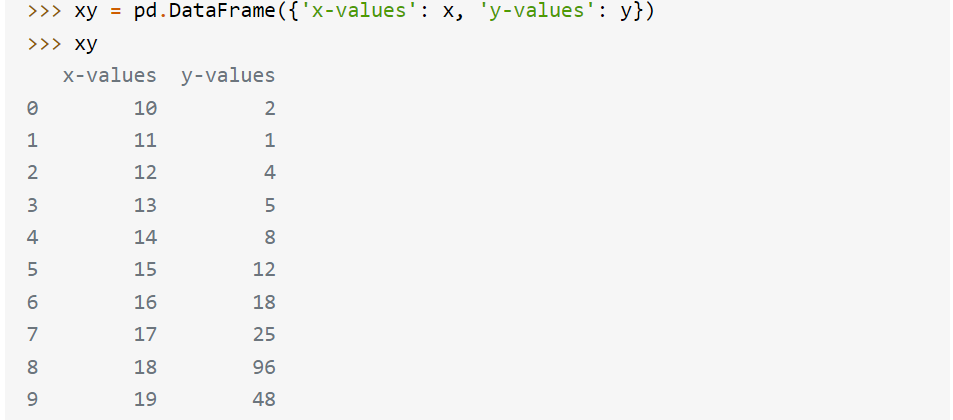
**Корреляция Пирсона: реализация pandas**

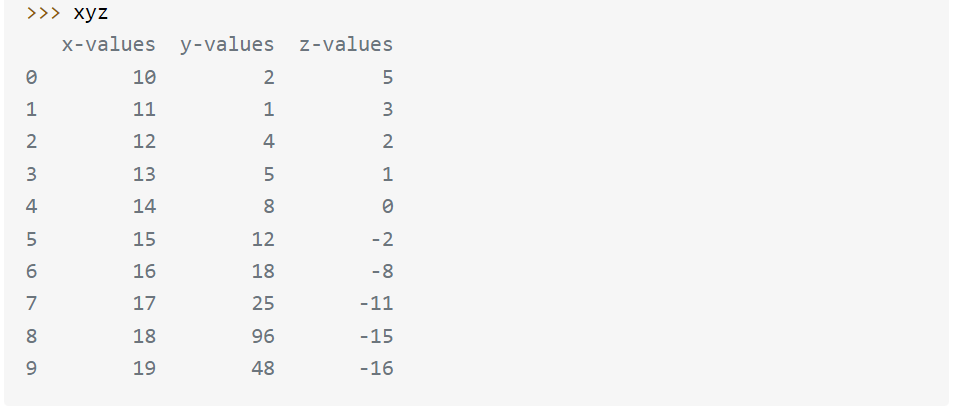
До сих пор вы использовали методы Series и DataFrame object для вычисления коэффициентов корреляции. Давайте рассмотрим эти методы более подробно. Сначала вам нужно импортировать pandas и создать несколько экземпляров Series и DataFrame:











Теперь у вас есть три Series объекта с именами x, y и z. У вас также есть два DataFrame объекта, xy и xyz.

**Примечание:** При работе с DataFrame экземплярами вы должны знать, что строки являются наблюдениями, а столбцы - объектами. Это согласуется с обычной практикой машинного обучения.



Если вы укажете nan значение, то .corr() оно все равно будет работать, но при этом будут исключены наблюдения, содержащие nan значения:

