به نام آفریننده هستی

دوره آموزشی تحلیل و مصورسازی داده با پایتون(پودمان دو)

Python data analysis and visualization

آنچه در این آموزش خواهید یافت:

پودمان دو

- آشنایی با مفهوم کتابخانه ها
- معرفی کتابخانه پانداس،کاربرد ها و ویژگی ها
- شناخت انواع داده های ساختاریافته
- روش های ایجاد دیتافریم
- خواندن فایل داده
- معرفی متدهای پرکاربرد برای کار با دیتافریم
- ادغام دیتافریم ها
- ساخت زیر مجموعه هایی از یک دیتا فریم
- روش های فیلتر کردن دیتا فریم
- گروهبندی داده ها
- پاکسازی داده ها(Data Cleaning)
- تشخیص همبستگی(Correlation)
- مصور سازی داده ها (Data visualization)

آشنایی با کتابخانه ها در پایتون

مفهوم کتابخانه

کتابخانه مجموعه ای از ماژول هاست. ماژول مجموعه ای از کدها است که برای یک هدف خاص استفاده می شود و می توان در برنامه های گوناگونی از آن استفاده کرد. در پایتون کتابخانه های بسیار زیادی وجود دارد. به این معنا که شما می توانید به منبع وسیعی از کد های آماده دسترسی داشته باشید و به آسانی و با سرعت و کیفیت بالا عمل کد نویسی را انجام دهید.

کتابخانه های متنوع پایتون برای علم داده

Pandas : یک کتابخانه قدرتمند در زبان برنامه‌نویسی پایتون است که برای تحلیل و مدیریت داده‌های ساختار یافته استفاده می‌شود. این کتابخانه به شما امکان می‌دهد که با داده‌های ساختار یافته مانند جداول یا دیتافریم‌ها به راحتی کار کنید و انواع عملیات مختلفی را بر روی آن‌ها انجام دهید.

با استفاده از پانداس، می‌توانید داده‌های خود را بخوانید و به فرمت مناسبی برای تحلیل و پردازش تبدیل کنید. این کتابخانه ابزارهای مختلفی را برای تمیز کردن و پیش‌پردازش داده‌ها ارائه می‌دهد و امکانات گسترده‌ای برای انجام عملیات ریاضی و آماری بر روی داده‌ها فراهم می‌کند.

چند مورد از کاربرد های کتابخانه پانداس در حوزه های مختلف

پردازش زبان طبیعی: پردازش زبان طبیعی به معنی رمز گشایی کلمات برای کامپیوتر ها به شیوه ای که بتوانند کلمات را درک کنند می باشد افراد متخصص درحوزه هوش مصنوعی همواره به دنبال آن هستند تا سیستم هایی بسازند که بتوانند متن را معنی کنند و کارهایی مانند تشخیص غلط املایی،ترجمه یا طبقه بندی متن را به صورت خودکار انجام دهند اما این کار به آسانی امکان پذیر نیست زیرا زبان انسان پیچیدگی زیادی دارد و در طول زمان نیز همواره دچار تغییر و تحول می شود کتابخانه پانداس در این راه به متخصصان خدمت زیادی کرده است و با کمک آن ساخت مدل های مختلف آسان شده است

سیستم های توصیه گر: حتما شما هم هنگام خرید از فروشگاه های اینترنتی با توصیه های سایت در مورد خرید محصول یا محصولاتی خاص رو برو شده اید این همان سیستم توصیه گر است که با کمک پانداس ایجاد شده است توصیه گرها براساس علایق وتاریخچه جستجوی کاربران پیشنهاداتی را برای آنها ارسال می کنند از آنجا که این سیستم ها برای پاسخگویی مناسب به حجم زیادی از دیتا ها نیازمند می باشند بدون کمک بیگ دیتا ها موفق به ساخت یک توصیه گر مناسب نخواهیم بود باید بگوییم که پانداس مهم ترین و اصلی ترین کتابخانه پایتون می باشد که برای مدیریت داده های با حجم بالا مورد استفاده قرار می گیرد و از آنجا که پایه و اساس آن زبان برنامه نویسی C می باشد بنابر این در مواجهه با بیگ دیتا ها دارای سرعت فوق العاده ای است.

پیش بینی قیمت سهام: پیش بینی بازار پر تلاطم سهام برای کارشناسان به آسانی امکان پذیر نیست زیرا عوامل زیادی از جمله سیاست های مالی و ارزی، وضعیت سیاسی و اقتصادی، میزان سود آوری شرکت ها و بسیاری از مسائل دیگر در تعیین صعودی یا نزولی بود بازار سهام نقش دارند اما با کمک پانداس این مشکل نیز به راحتی قابل حل است. پانداس با تجزیه و تحلیل داده های قبلی سهام، قادر است مدل هایی را بسازد که بتوانند بازار سهام را پیش بینی کند.

ویژگی های پانداس

-**مدیریت داده های ازدست رفته:**یکی از مشکلات اساسی هنگام کار با داده ها، برخورد با ویژگی های گم شده می باشد که به دلایل زیادی می تواند رخ دهد مثلا شخص تمایلی به اعلام میزان در آمد خود یا ثبت شماره تلفن خود نداشته و یا در جمع آوری داده ها بی دقتی صورت گرفته است الگوریتم های یادگیری ماشین قابلیت مقابله با داده های گم شده را ندارند و هنگام پیاده سازی آنها با خطا مواجه خواهیم شد پانداس دارای چندین راه کار مفید برای شناسایی داده های گم شده و سپس حذف و یا جایگزین کردن مقدار دیگری برای این نوع داده ها است.

-**تمیز کردن داده ها:**

از آنجا که داده های موجود در دنیای اطراف ما به صورت داده های خام و تمیز نشده هستند ، ممکن است حاوی مطالب بی ارزشی باشند که درنهایت منجر به ارائه نتایج نادرست و به دنبال آن تصمیم گیری های اشتباه خواهند شد . فرایند پاک سازی داده ها شامل شناسایی و حذف موارد اشتباه و نادرست و بیهوده از پایگاه داده است پانداس امکان تمیز کردن داده ها به منظور دستیابی به نتایج دقیق تر را فراهم می سازد.

-**پشتیبانی از فرمت های مختلف فایل ها :**

شما به عنوان یک تحلیل گر داده با انواع مختلف فایل ها مواجه خواهید شد پانداس امکان استفاده از انواع مختلف فایل ها شامل Comma-separated values (CSV)، XLSX، JSONو..... فراهم کرده است این مورد از مهم ترین ویژگی های پانداس می باشد.

منظور از داده‌های ساختار یافته پانداس چیست؟

داده‌های ساختار یافته به داده‌هایی میگویند که به صورت منظم و ساختارمند در قالب مشخصی ذخیره شده‌اند. این داده‌ها معمولاً به صورت جداول، ماتریس‌ها، دیتافریم‌ها، یا هر نوع دیگری از ساختارهای داده‌ای است که دارای ستون‌ها و ردیف‌ها هستند و هر ستون معمولاً یک نوع داده‌ای خاص دارد.

ویژگی اصلی داده‌های ساختار یافته، قابلیت سازماندهی و دسته‌بندی دقیق داده‌ها است. با ساختارمند کردن داده‌ها، می‌توان به سادگی به داده‌ها دسترسی داشته و عملیات مختلفی روی آن‌ها انجام داد. همچنین، داده‌های ساختار یافته امکان تحلیل و استفاده بهینه از اطلاعات را فراهم می‌کنند.

به عنوان مثال، یک دیتافریم در کتابخانه Pandas یک نمونه از داده‌های ساختار یافته است. این دیتافریم اطلاعات را به صورت جدولی سازماندهی کرده و به کاربر امکان می‌دهد تا عملیات مختلفی را بر روی داده‌ها اعمال کند، مانند فیلتر کردن، مرتب‌سازی، ترکیب و تحلیل آماری. این ساختار داده‌ای بسیار مفید است زیرا به شما امکان می‌دهد تا به سادگی و به شکل منظم با داده‌های خود کار کنید و از اطلاعات بهتری برای تصمیم‌گیری‌ها و تحلیل‌های خود استفاده کنید.

برای استفاده از هر یک از کتابخانه ها باید با دستور import آن را فراخوانی نمایید. زیرا تا زمانی که در فایل پایتون، یک کتابخانه خاص را وارد (import) نکرده باشید، نمی توانید از توابع(functions) ، کلاس ها (classes) و روش های (methods) تعریف شده در آن کتابخانه، در کدنویسی استفاده نمایید.

ساختار داده‌ها در کتابخانهPandas

به طور کلی کتابخانه Pandas دارای دو دسته ساختار داده است:

- سری (Series)
- دیتافریم (DataFrame)

Series در پانداس:

سری یک آرایه‌ی یک بُعدی و برچسب‌گذاری شده است (دارای ایندکس برای سطرهاست) که می‌تواند انواع مختلف داده (عدد صحیح، اعشاری، رشته و...) را در خود نگه دارد.

مقادیری که در series قرار می‌گیرند قابل تغییر هستند؛ اما اندازه series پانداس، غیر قابل تغییر است. به اولین عنصر در series ، اندیس تخصیص داده خواهد شد و اندیس آخرین عنصر در series برابر با N-1 است که در آن، Nتعداد کل عنصرهای موجود در سری است.

درصورتی که اندیس گذاری را از قبل مشخص نکنیم به صورت پیش فرض با عدد صفر شروع می شود اما می توانیم آن را با کاراکتر ها نیز نمایش دهیم با کمک اندیس گذاری می توانیم به یک مقدار مشخص دسترسی پیدا کنیم.

برای ساخت series پانداس، ابتدا باید بسته پانداس را با استفاده از دستور import پایتون، واردکرد.

```
Import pandas  as pd
```

یک سری در پانداس به شکل زیر تعریف می‌شود:

```
pandas.Series( data, index, dtype, copy)
```

ساخت سری با استفاده از یک لیست

```
import pandas as pd
list = [11, 12, 13,14,15]
ser = pd.Series(list)
print(ser)
```

```
0    11
1    12
2    13
3    14
4    15
dtype: int64
```

ایندکس گذاری با استفاده از کاراکتر ها:

```
list = [11, 12, 13,14,15]
i=["a" ,"b" ,"c" ,"d" ,"e"]
ser2 = pd.Series(list,index=i)
print(ser2)
```

```
a    11
b    12
c    13
d    14
e    15
dtype: int64
```

ساخت سری با استفاده از آرایه های Numpy

نکته: کتابخانه‌ی Numpy یکی از ابزارهای قوی و محبوب در زمینه‌ی پردازش عددی و علم داده‌ها در پایتون است. این کتابخانه برای کار با آرایه‌ها و ماتریس‌ها بهینه شده و امکانات بسیاری برای انجام عملیات ریاضی، جبر خطی، تبدیلات، انتخاب و فیلتر کردن داده‌ها و بسیاری از عملیات دیگر فراهم می‌کند.

در این مثال با استفاده از کتابخانه Numpy آرایه ای از داده ها ساخته شده است.سپس این آرایه به یک سری تبدیل شده است.

نکته : قبل از اعمال دستورات می بایست کتابخانه Numpy را فراخوانی کنید.

Import Numpy as np

```
data = np.array(["ali","erfan","sadra"])

i = ["a","b","c"]

myser = pd.Series(data, index=i)
myser
```

```
a      ali
b    erfan
c    sadra
dtype: object
```

ساخت سری با استفاده از دیکشنری

```
dic = {'a': "ali", 'b': "erfan", 'c': "sadra"}
myser3 = pd.Series(dic, index=['a', 'b', 'c'])
myser3
```

```
a      ali
b    erfan
c    sadra
dtype: object
```

```
dic = {'1': "ali", '2': "erfan", '3': "sadra"}
myser3 = pd.Series(dic, index=['1', '2', '3',])
myser3
```

```
1      ali
2    erfan
3    sadra
dtype: object
```

```
dic = {'1': "ali", '2': "erfan", '3': "sadra"}
myser3 = pd.Series(dic, index=['1', '2', '3','4'])
myser3
```

```
1      ali
2    erfan
3    sadra
4      NaN
dtype: object
```

کتابخانه‌های پایتون داده‌های گمشده را به صورت عبارت NaN نشان می‌دهند که مخفف "Not a Number" است. به کمک توابع مرتبط با این کتابخانه‌ها می‌توانید مشخص کنید که کدام سلول‌ها مقادیر گمشده دارند.

فراخوانی عناصر سری

برای دسترسی به هر کدام از عناصر موجود در سری شماره اندیس یا کاراکتر مربوط به آن را داخل [ ] قرار می دهیم

```
dic = {'1': "ali", '2': "erfan", '3': "sadra"}
myser3 = pd.Series(dic, index=['1', '2', '3','4'])
myser3
```

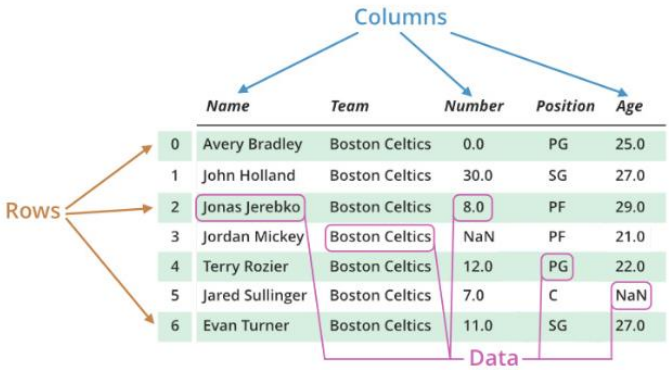
```
1      ali
2    erfan
3    sadra
4      NaN
dtype: object
```

```
print(myser3["1"])
```

```
ali
```

DataFrame :

دیتا فریم نوعی ساختار داده ای دو بعدی است که داده ها را به صورت جدول در سطر ها و ستون های ایندکس دار قرار می دهد .برای هرکدام از سطرها درصورتی که index ها را از قبل تعیین نکنیم به صورت پیش فرض مقادیر ۰ تا n-1 در نظر گرفته می شود. دیتا فریم ها را می توان با استفاده از یک یا چند لیست یا دیکشنری ایجاد کرد اندازه دیتا فریم ها نیز به سادگی قابل تغییر می باشد.



ساخت دیتا فریم از یک لیست:

```
list = [["ali","20","A"],["reza","19","A"],["sara","15","C"]]

df = pd.DataFrame(list,columns=["Name","Grade","Class"])
df
```

	Name	Grade	Class
0	ali	20	A
1	reza	19	A
2	sara	15	C

ساخت دیتا فریم با کمک یک تاپل(Tuple):

```
data = [(101, 'Ali'), (1022, 'Sara'), (103, 'Sina'), (104, 'Dara')]
df4=pd.DataFrame(data, columns=['Cod', 'Name'])
df4
```

	Cod	Name
0	101	Ali
1	1022	Sara
2	103	Sina
3	104	Dara

```
data = [(101, 'Ali','A'), (1022, 'Sara','B'), (103, 'Sina','A'), (104, 'Dara','C')]
df5=pd.DataFrame(data, columns=['Cod', 'Name','Grade'])
df5
```

	Cod	Name	Grade
0	101	Ali	A
1	1022	Sara	B
2	103	Sina	A
3	104	Dara	C

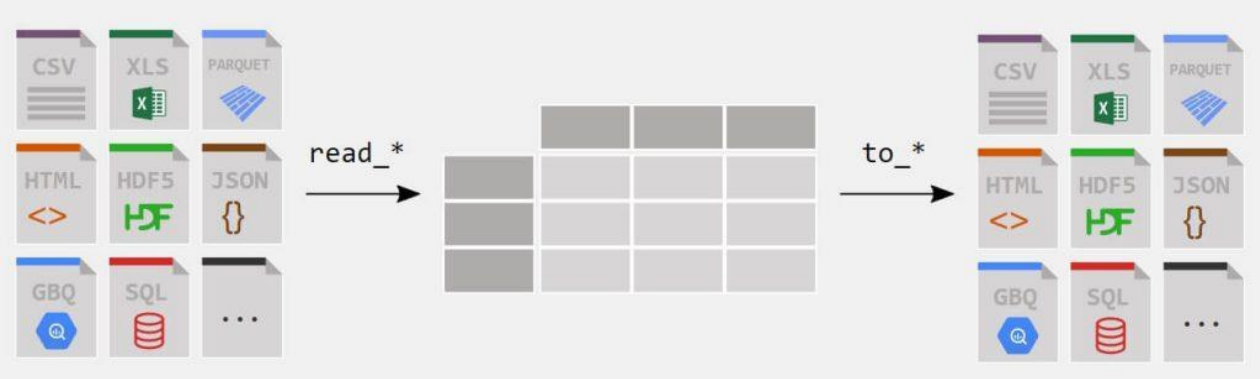
ساخت دیتا فریم با استفاده از دیکشنری :

```
dict={"Name":["ali","reza","sara"],"Grade":[20,19,15] ,"Class":["A","A","C"]}
df = pd.DataFrame(dict)
df
```

	Name	Grade	Class
0	ali	20	A
1	reza	19	A
2	sara	15	C

خواندن و نوشتن داده در جدول

Pandas از انواع فرمت‌های فایل یا منابع داده مانندcsv ، excel، sql، json، parket و... پشتیبانی می‌کند. وارد کردن داده ها از هر یک از این منابع داده توسط یک تابع با پیشوند read_* انجام می‌شود. به طور مشابه، از پیشوند to_* نیز می‌توان برای ذخیره داده‌ها استفاده کرد.



خواندن فایل های CSV :

فایل های CSV مخفف عبارت Comma Separated Values است که به معنی مقادیر جدا شده با ویرگول می باشد. این فایل حاوی یک لیست از اطلاعات است که اغلب برای تبادل داده بین برنامه های مختلف استفاده می شود.

ساختار فایل هایCSV

فایل CSV ساختار نسبتا ساده ای دارد. این فایل یک لیست از داده هایی را ارائه می دهد که توسط کاما یا ویرگول جدا شده اند. به عنوان مثال شما چندین مخاطب در یک برنامه مدیریت تماس و مخاطب دارید که محتوای آن به این شکل است:

Name,Email,Phone Number,Address

AnzalWeb,mail@anzalweb.ir,013-123-4567,Anzali-Khiban-1

Ehsan,ehsan.sez@gmail.com,087-123-4567,Bijar-Khiaban-2

این نمونه خلاصه و مثالی از ۲ داده در این فایل است که می تواند پیچیده تر از این و شامل هزاران خط ورودی و رشته های طولانی باشد. ممکن است علامت ها فرق کند اما فرمت اصلی به همین شکل است. این سادگی در فایل های CSV باعث شده است تا بتوانید به آسانی از داده های خود خروجی و ورودی بگیرید. این داده ها به آسانی با برنامه های معمولی مانند NotePad و یا اکسل قابل خواندن توسط انسان هستند.

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df)
```

ذخیره سازی دیتافریم در یک فایل csv

```
import pandas as pd
df = pd.DataFrame({'daneshkadeh': ['معماري','هنر', 'فني' ,'رياضي'],
                   'daneshjoo': ['520', '460','270','660'],
                   'ostad': ['12', '10' ,'22','26']})
df.to_csv(r"C:\Users\kh.hanani\Desktop\out.csv")
```

```
df
```

	daneshkadeh	daneshjoo	ostad
0	رياضي	520	12
1	فني	460	10
2	هنر	270	22
3	معماري	660	26

مشاهده داده ها

head() و tail() :متد هايhead() و head() :به ما این امکان را می‌دهند که نمونه‌ای از داده خود را ببینیم head اولِ داده‌ها و tail آخرِ داده‌ها را نمایش می‌دهد. تعدادِ پیش فرضِ نمایش داده‌ها ۵ می‌باشد. اگر بخواهید تعداد دلخواهی از نمونه‌ها را ببینید، اولین آرگومانِ این توابع را عدد مد نظر خود را می‌دهیم

.مثال:

با چاپ ۱۰ ردیف اول DataFrame یک نمای کلی سریع دریافت کنید:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.head(10))
```

خروجی: ۱۰ سطر اول داده ها را از جدول data نمایش میدهد.

Iloc :

اگر از قبل می دانیم که کدام ردیف ها را می خواهیم، می توانیم به سادگی از متد ilocبرای تعیین ردیف ها از روی ایندکس آنها استفاده کنیم.

دیتای همه سطر ها و همه ستون ها را نمایش دهید.

```
df.iloc[: , : ]
```

دیتای سطر اول را نمایش دهید.

```
df.iloc[[0] , :]
```

معادل دستور df.head(3) را بنویسید.

```
df.iloc [[0,1,2],:]
```

```
df.iloc [0:3 , :]
```

دیتای همه جدول منهای ستون آخر را نمایش دهید.

```
df.iloc[ : , : -1]
```

دیتای ستون دوم را نمایش دهید.

```
df.iloc[ : , [1]]
```

دیتا فریم با سه ستون **Name ,Grade ,Class** بسازید و مشخصات ۵ نفر را وارد کنید .

```
[35]: Dict={'Name':["Ali", "Mohammad", "Maryam", "zahra" , "Amir"  ],'Grade': [18,19,19,17,20], 'Class':['A','B','C', 'D','E']}

[44]: df = pd.DataFrame({'Name':["Ali", "Mohammad", "Maryam", "zahra" , "Amir"  ],'Grade': [18,19,19,17,20], 'Class':['A','B','C', 'D','E']})

[45]: df
```

[45]:

	Name	Grade	Class
0	Ali	18	A
1	Mohammad	19	B
2	Maryam	19	C
3	zahra	17	D
4	Amir	20	E

```
[ ]:
```

دسترسی به یک ستون از دیتا فریم:

می خواهیم ستون Name از دیتافریم فوق را به تنهایی نمایش دهیم . برای این منظور دو روش وجود دارد :

```
df.Name
```

```
0         Ali
1    Mohammad
2      Maryam
3       zahra
4        Amir
Name: Name, dtype: object
```

```
df['Name']
```

```
0         Ali
1    Mohammad
2      Maryam
3       zahra
4        Amir
Name: Name, dtype: object
```

دو ستون **Name , Grade** را با استفاده از نام ستون نمایش دهید .

```
df[['Name','Grade']]
```

	Name	Grade
0	Ali	18
1	Mohammad	19
2	Maryam	19
3	zahra	17
4	Amir	20

اعمال برخی از توابع روی داده ها:

max() ,min(), sum(), mean(, ...

میانگین ستون نمرات را حساب کنید.

```
df["Grade"].mean()
```

```
18.6
```

Describe():

این متد خلاصه‌ای از اطلاعاتِ عددیِ DataFrame را به ما می‌دهد:

```
df
```

	Name	Grade	Class
0	Ali	18	A
1	Mohammad	19	B
2	Maryam	19	C
3	zahra	17	D
4	Amir	20	E

```
df.describe()
```

	Grade
count	5.000000
mean	18.600000
std	1.140175
min	17.000000
25%	18.000000
50%	19.000000
75%	19.000000
max	20.000000

: info()

این متد مشخصات کلی DataFrame را ارائه می دهد.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Name    5 non-null      object
 1   Grade   5 non-null      int64 
 2   Class   5 non-null      object
dtypes: int64(1), object(2)
memory usage: 252.0+ bytes
```

: index

نحوه index گذاری دیتافریم را بیان می کند.

```
df.index
```

```
RangeIndex(start=0, stop=5, step=1)
```

:Columns

لیستی از ستون‌ها را نمایش می‌دهد.

```
df.columns
```

```
Index(['Name', 'Grade', 'Class'], dtype='object')
```

Count:

مقادیر غیر null را شمارش می کند .

```
df
```

	Name	Grade	Class
0	Ali	16	A
1	Mohamad	17	B
2	Maryam	15	A
3	Zahra	14	C
4	Amir	13	B
5	Hasan	15	A

```
df.count()
```

```
Name     6
Grade    6
Class    6
dtype: int64
```

```
df['Name'].count()
```

```
6
```

Len() : کل مقادیر را شمارش می کند

```
df
```

	Name	Grade	Class
0	Ali	16	A
1	Mohamad	17	B
2	Maryam	15	A
3	Zahra	14	C
4	Amir	13	B
5	Hasan	15	A

```
len(df)
```

```
6
```

```
len(df['Name'])
```

```
6
```

اضافه کردن سطر و ستون

برای اضافه کردن سطر از loc استفاده می‌کنیم و index مورد نظر را به آن می‌دهیم. در نهایت مقادیرِ آن را به صورت لیست تعریف می کنیم .

```
df.loc[5]=['Hasan', '15','A']
```

```
df
```

	Name	Grade	Class
0	Ali	18	A
1	Mohammad	19	B
2	Maryam	19	C
3	zahra	17	D
4	Amir	20	E
5	Hasan	15	A

در نظر داشته باشید که تعداد و نوعِ elemnt- های لیستی که می‌دهیم، باید مطابق با ستون‌ها باشد.

اگر مقداری که به loc می‌دهیم تکراری باشد و مشابه آن را در DataFrame داشته باشیم، مقادیری که به آن داده شده، جایگزینِ مقادیر قبلی می‌شود.

حذف سطر و ستون

برای حذف سطر و ستون از متد ()drop استفاده می‌کنیم. متدِ drop آرگومان‌های مختلفی می‌گیرد، که پرکاربردترینِ آن اولین آرگومان است، که سطر یا ستون مورد نظر ماست. آرگومان دیگر axis است که برای سطر ۰ و برای ستون ۱ می‌باشد (به صورت پیشفرض ۰ است.) آرگومان بعدی inplace است، که در صورت True بودن تغییرات به همان DataFrame اعمال می‌شود .به صورت پیشفرض ۰ است مگر مواقعی استفاده می‌شود که بخواهیم تغییرات را در DataFrame جدید ذخیره کنیم.

سطر دارای index ۴ را حذف کنید.

```
df
```

	Name	Grade	Class
0	Ali	18	A
1	Mohammad	19	B
2	Maryam	19	C
3	zahra	17	D
4	Amir	20	E
5	Hasan	15	A

```
df.drop(4)
```

	Name	Grade	Class
0	Ali	18	A
1	Mohammad	19	B
2	Maryam	19	C
3	zahra	17	D
5	Hasan	15	A

ستون Name را حذف کنید.

```
df
```

	Name	Grade	Class
0	Ali	18	A
1	Mohammad	19	B
2	Maryam	19	C
3	zahra	17	D
4	Amir	20	E
5	Hasan	15	A

```
df.drop('Name',axis=1 )
```

	Grade	Class
0	18	A
1	19	B
2	19	C
3	17	D
4	20	E
5	15	A

برای اینکه ستون Name به طور کامل از دیتافریم حذف شود دستور را چگونه تغییر میدهید؟

```
df.drop('Name',axis=1 ,inplace=True )
```

	Grade	Class
0	18	A
1	19	B
2	19	C
3	17	D
4	20	E
5	15	A

تغییر نام ستون

برای تغییر نام ستون از متد ()rename استفاده می‌کنیم. متد rename یک دیکشنری به عنوان آرگومان می‌گیرد، که کلیدهای آن نامِ فعلی ستون‌هاست و مقادیرِ آن، اسامیِ جدید است.

```
new_col={'Name':'Nam','Grade':'Nomre','Class':'Group'}
```

```
df.rename(columns=new_col )
```

	Nam	Nomre	Group
0	Ali	18	A
1	Mohammad	19	B
2	Maryam	19	C
3	zahra	17	D
4	Amir	20	E
5	Hasan	15	A

برای اینکه تغییر نام ستون ها در دیتافریم ماندگار شود دستور را چگونه تغییر میدهید؟

```
new_col={'Name':'Nam','Grade':'Nomre','Class':'Group'}
```

```
df.rename(columns=new_col ,Inplace=True )
```

	Nam	Nomre	Group
0	Ali	18	A
1	Mohammad	19	B
2	Maryam	19	C
3	zahra	17	D
4	Amir	20	E
5	Hasan	15	A

دستکاری داده ها در پایتون:

Broadcasting : به انجام یک عملیات بر روی تک تک اعضا ستون های یک دیتا فریم گفته می شود .

مثلا می خواهیم به نمره تک تک افراد در دیتافریم یک مقدار ثابت اضافه شود (۲ نمره)

```
df
```

	Name	Grade	Class
0	Ali	18	A
1	Mohamad	19	B
2	Maryam	17	A
3	Zahra	16	C
4	Amir	15	B
5	Hasan	17	A

```
df['Grade']+2
```

```
0    20
1    21
2    19
3    18
4    17
5    19
Name: Grade, dtype: int64
```

برای اینکه این افزایش در ستون نمره در دیتافریم ماندگار شود دستور را چگونه تغییر میدهید؟

```
df['Grade']=df['Grade']+2
```

```
df
```

	Name	Grade	Class
0	Ali	20	A
1	Mohamad	21	B
2	Maryam	19	A
3	Zahra	18	C
4	Amir	17	B
5	Hasan	19	A

علاوه بر عملگرهای + و – و * و / می توان از توابع Add و Sub وMul و Div استفاده کرد .

مثال :

```
df['Grade'].add(2)
```

```
0    22
1    23
2    21
3    20
4    19
5    21
Name: Grade, dtype: int64
```

```
df['Grade']=df['Grade'].sub(4)
```

```
df
```

	Name	Grade	Class
0	Ali	16	A
1	Mohamad	17	B
2	Maryam	15	A
3	Zahra	14	C
4	Amir	13	B
5	Hasan	15	A

ادغام دیتا فریم ها

عمل ادغام در دیتافریم‌ها همان عمل پیوستن (Join) در SQL است که از آن برای پیوستن دو دیتافریم در یک یا چند ستون استفاده می‌کنیم. همچنین اگر بخواهید دو دیتافریم را ادغام کنید باید از روش **pd.merge()** استفاده نمائید.

ابتدا مثالی از ادغام معادل inner join در SQL را بررسی می کنیم .

```
df1 = pd.DataFrame({'a': ['foo', 'bar'], 'b': [1, 2]})
df2 = pd.DataFrame({'a': ['foo', 'baz'], 'c': [3, 4]})
df1
```

	a	b
0	foo	1
1	bar	2

```
df2
```

	a	c
0	foo	3
1	baz	4

```
df1.merge(df2, how='inner', on='a')
```

	a	b	c
0	foo	1	3

در اینجا مثالی از ادغام ، معادل Left join در SQL را خواهیم داشت:

```
df1 = pd.DataFrame({'a': ['foo', 'bar'], 'b': [1, 2]})
df2 = pd.DataFrame({'a': ['foo', 'baz'], 'c': [3, 4]})
df1
```

	a	b
0	foo	1
1	bar	2

```
df2
```

	a	c
0	foo	3
1	baz	4

```
df1.merge(df2, how='left', on='a')
```

	a	b	c
0	foo	1	3.0
1	bar	2	NaN

و نیز مثالی از ادغام ، معادل Right join در SQL را خواهیم داشت:

```
df1 = pd.DataFrame({'a': ['foo', 'bar'], 'b': [1, 2]})
df2 = pd.DataFrame({'a': ['foo', 'baz'], 'c': [3, 4]})
df1
```

	a	b
0	foo	1
1	bar	2

```
df2
```

	a	c
0	foo	3
1	baz	4

```
df1.merge(df2, how='right', on='a')
```

	a	b	c
0	foo	1.0	3
1	baz	NaN	4

مرتب سازی دیتافریم

اگر بخواهید یک دیتافریم را بر اساس مقادیر موجود در یک ستون خاص مرتب کنید باید از روش **sort_values()** استفاده کنید.

مرتب کردن بر اساس یک یا چند فیلد:

```
df.sort_values(by="Grade")
```

	Name	Grade	Class
4	Amir	15	B
3	Zahra	16	C
2	Maryam	17	A
5	Hasan	17	A
0	Ali	18	A
1	Mohamad	19	B

```
df.sort_values(by=["Grade","Name"])
```

	Name	Grade	Class
4	Amir	15	B
3	Zahra	16	C
5	Hasan	17	A
2	Maryam	17	A
0	Ali	18	A
1	Mohamad	19	B

نکته: به طور پیش فرض داده ها به صورت صعودی مرتب می شود در صورتیکه بخواهیم مرتب سازی نزولی Sort Descending داشته باشیم از ascending=False استفاده می کنیم .

```
df.sort_values(by="Grade" , ascending=False)
```

	Name	Grade	Class
1	Mohamad	19	B
0	Ali	18	A
2	Maryam	17	A
5	Hasan	17	A
3	Zahra	16	C
4	Amir	15	B

Concatenation یا الحاق داده ها:

در واقع به معنای افزودن یک مجموعه از داده‌ها به دیگری است، به وسیله فراخوانی متد concat() قابل انجام است.

```
df1=pd.read_csv(r"C:\Users\kh.hanani\Desktop\class.csv")
df2=pd.read_csv(r"C:\Users\kh.hanani\Desktop\information.csv")
df1
```

	Name	Grade	Class
0	Ali	18	A
1	Mohamad	19	B
2	Maryam	17	A
3	Zahra	16	C
4	Amir	15	B
5	Hasan	17	A

```
df2
```

	Name	Age	sex
0	Ali	32	M
1	Mohamad	22	M
2	Maryam	45	F
3	Zahra	31	F
4	Amir	19	M
5	Hasan	25	M

```
pd.concat((df1,df2),axis=1)
```

	Name	Grade	Class	Name	Age	sex
0	Ali	18	A	Ali	32	M
1	Mohamad	19	B	Mohamad	22	M
2	Maryam	17	A	Maryam	45	F
3	Zahra	16	C	Zahra	31	F
4	Amir	15	B	Amir	19	M
5	Hasan	17	A	Hasan	25	M

اگر بخواهیم دیتاست ها به صورت سطری با هم ترکیب شوند مقدار axis را برابر صفر قرار می دهیم.

```
pd.concat((df1,df2),axis=0)
```

	Name	Grade	Class	Age	sex
0	Ali	18.0	A	NaN	NaN
1	Mohamad	19.0	B	NaN	NaN
2	Maryam	17.0	A	NaN	NaN
3	Zahra	16.0	C	NaN	NaN
4	Amir	15.0	B	NaN	NaN
5	Hasan	17.0	A	NaN	NaN
0	Ali	NaN	NaN	32.0	M
1	Mohamad	NaN	NaN	22.0	M
2	Maryam	NaN	NaN	45.0	F
3	Zahra	NaN	NaN	31.0	F
4	Amir	NaN	NaN	19.0	M
5	Hasan	NaN	NaN	25.0	M

مشخص است که مقادیر ستون هایی که داده متناظری در دیتافریم مقابل ندارند با NaN پر شده است.

فیلتر کردن دیتافریم

Pandas این قابلیت را دارد که می‌توان سطرها و یا ستون هایی خاص از یک جدول را انتخاب و یا فیلتر نمود.

فیلتر کردن دیتافریم بر اساس یک شرط(روش اول):

```
df1
```

	Name	Grade	Class
0	Ali	18	A
1	Mohamad	19	B
2	Maryam	17	A
3	Zahra	16	C
4	Amir	15	B
5	Hasan	17	A

```
df[df1["Grade"]>= 17]
```

	Name	Grade	Class
0	Ali	18	A
1	Mohamad	19	B
2	Maryam	17	A
5	Hasan	17	A

```
df[df1["Grade"]== 17]
```

	Name	Grade	Class
2	Maryam	17	A
5	Hasan	17	A

روش دوم : فیلتر کردن داده ها با توابع nlargest() و nsmallest()

با اجرای متد nlargest() و nsmallest() می‌توانیم سطرهایی که مقدارشان در یک ستون خاص بیشترین و کمترین است را فیلتر کنیم.

در این مثال می خواهیم ۳ سطری که در ستون Grade بزرگترین مقادیر را به خود اختصاص داده‌اند، بینیم.

```
df1
```

	Name	Grade	Class
0	Ali	18	A
1	Mohamad	19	B
2	Maryam	17	A
3	Zahra	16	C
4	Amir	15	B
5	Hasan	17	A

```
df1.nlargest(3,"Grade")
```

	Name	Grade	Class
1	Mohamad	19	B
0	Ali	18	A
2	Maryam	17	A

حال می خواهیم ۳ سطری که در ستون Grade کمترین مقادیر را به خود اختصاص داده‌اند، بینیم.

```
df1
```

	Name	Grade	Class
0	Ali	18	A
1	Mohamad	19	B
2	Maryam	17	A
3	Zahra	16	C
4	Amir	15	B
5	Hasan	17	A

```
df1.nsmallest(3,"Grade")
```

	Name	Grade	Class
4	Amir	15	B
3	Zahra	16	C
2	Maryam	17	A

روش سوم : استفاده از متد query()

متد query یکی دیگر از متدهای فیلتر کردن است که انعطاف‌پذیری بالایی دارد .query این امکان را برای شما فراهم

می‌کند تا یک حالت را در قالب یک رشته (string) تعیین کنید.

```
df1.query('16 < Grade < 19')
```

	Name	Grade	Class
0	Ali	18	A
2	Maryam	17	A
5	Hasan	17	A

```
df1.query('Grade < 19')
```

	Name	Grade	Class
0	Ali	18	A
2	Maryam	17	A
3	Zahra	16	C
4	Amir	15	B
5	Hasan	17	A

```
df1.query('Class == "A" ')
```

	Name	Grade	Class
0	Ali	18	A
2	Maryam	17	A
5	Hasan	17	A

روش چهارم: فیلتر کردن داده ها بر اساس شرایط با استفاده از Loc :

```
class_A = df.loc[(df["Class"] == "A")]
```

```
class_A
```

	Name	Grade	Class
0	Ali	18	A

```
class_A_lower = df.loc[(df["Class"] == "A") & (df["Grade"] < 19)]
```

```
class_A_lower
```

	Name	Grade	Class
0	Ali	18	A

گروهبندی داده ها در پایتون:

متد groupby() به ما امکان می‌دهد داده‌ها را بر اساس یک یا چند ستون گروه‌بندی کنیم و توابع مختلفی مانند مجموع، میانگین، شمارش و غیره را انجام دهیم.

```
df1
```

	Name	Grade	Class
0	Ali	18	A
1	Mohamad	19	B
2	Maryam	17	A
3	Zahra	16	C
4	Amir	15	B
5	Hasan	17	A

```
df1.groupby(["Class"]).mean("Grade")
```

	Grade
Class	
A	17.333333
B	17.000000
C	16.000000

```
df1.groupby(["Class"]).max("Grade")
```

	Grade
Class	
A	18
B	19
C	16

شمارش مقادیر یکتا (یونیک) در یک ستون

اگر بخواهید مقادیر یونیک را در یک ستون از دیتافریم شمارش کنید باید از **nunique()** استفاده کنید .

```
df2
```

	Name	Age	sex
0	Ali	32	M
1	Mohamad	22	M
2	Maryam	45	F
3	Zahra	31	F
4	Amir	19	M
5	Hasan	25	M

```
df2["sex"].nunique()
```

```
2
```

اعمال توابع روی یک ستون:

اگر می‌خواهید یک تابع را در یک ستون اعمال کنید باید از متد apply() استفاده نمائید.

روش شمارش مقادیر:(Value Counts)

اگر می‌خواهید فراوانی هر مقدار را در یک ستون پیدا کنید باید از روش **value_counts()** استفاده کنید .

پاکسازی داده (Data Cleaning)

پاکسازی داده یا تمیزکاری داده به معنای شناسایی و اصلاح مشکلاتی است که در داده وجود دارد. این مشکلات شامل اشتباهات مختلف در ورود داده‌ها، اطلاعات ناقص یا نادرست، اختلاف و تضاد در داده‌ها، یا حتی اطلاعات تکراری هستند.
هدف اصلی پاکسازی داده این است که داده‌ها به‌گونه‌ای تنظیم و بهینه شود که برای تحلیل، پیش‌بینی و تصمیم‌گیری‌های موثر، قابل استفاده باشند.

برای مثال، فرض کنید یک دیتابیس دارید که در آن اطلاعات افراد شامل نام، سن، و آدرس آن‌ها ذخیره شده است. اما ممکن است نام یا آدرس، به شکل نادرست وارد شده باشد یا سن به‌صورت یک عدد منفی یا غیرمنطقی وارد شده باشد .با پاکسازی داده این مشکلات به‌راحتی شناسایی و اصلاح می‌شوند.

پاکسازی داده وظیفه چه کسی است؟

تحلیل‌گران داده :(Data analysts) این افراد به داده‌ها نگاه دقیقی دارند تا اطمینان حاصل کنند که گزارش‌ها و نتایجی که تولید می‌کنند دقیق و قابل اعتماد هستند. برای این کار، داده‌ها باید پاکسازی شوند تا اشتباهات و ناهماهنگی‌های موجود در آن‌ها پاک شوند.

دانشمندان داده :(Data scientists) این افراد با مدل‌های ماشین لرنینگ سروکار دارند، یعنی مدل‌های ماشین لرنینگ را بر اساس داده‌ها آموزش می‌دهند. اگر داده‌ها پاکسازی نشده باشند، ممکن است عملکرد این مدل‌ها کاهش می‌یابد.

برنامه‌نویسان :برنامه‌نویسان در بسیاری از پروژه‌های خود به داده‌های تمیز نیاز دارند. مثلاً، یک برنامه‌نویس وب که برای نمایش اطلاعات به کاربران از داده استفاده می‌کند، اگر این داده‌ها پاکسازی نشده باشند، ممکن است در برنامه مشکلاتی ایجاد شود.

اهمیت پاکسازی داده

-افزایش دقت داده‌ها

داده‌های نادرست، آمارها و نتایج را تحت تاثیر قرار می‌دهد. با پاکسازی داده، سازمان‌ها تصمیمات مبتنی بر داده را با اطمینان بیشتری می‌گیرند.

-افزایش قابلیت استفاده از داده‌ها

داده‌های پاکسازی‌شده و قابل اعتماد به‌شکل گسترده‌تر و در بخش‌های مختلفی از سازمان قابل استفاده هستند.

-تحلیل آسان‌تر

پاکسازی داده کمک می‌کند تحلیل داده‌ها آسان‌تر انجام شود .

-ذخیره‌سازی بهتر داده‌ها

پاکسازی داده و حذف داده‌های غیرضروری و تکراری، هزینه‌ی ذخیره‌سازی داده را کاهش می‌دهد. در نتیجه سازمان‌ها می‌توانند استفاده از منابع داده‌ها را بهینه کنند.

-کیفیت داده

با حذف اشتباهاتی مانند املای نادرست، اصلاح فرمت تاریخ، و رفع مشکلات متداول، کیفیت داده‌ها افزایش می‌یابد.

-کارایی

پاکسازی داده باعث بهینه‌سازی فرآیند پردازش داده می‌شود، که این به صرفه‌جویی در زمان و منابع کمک می‌کند و فرآیند آنالیز داده را سریع‌تر و کارآمدتر می‌کند.

بررسی و کاوش در داده ها

پس از وارد کردن کتابخانه های لازم و بارگذاری فایل داده لازم است به بررسی و کاوش در داده ها بپردازیم. یکی از روش های بررسی داده ها استفاده از متد ()info می باشد. با استفاده از خروجی به دست آمده می توانید اطلاعات داده ها از قبیل تعداد ستون ها ، تعداد سطر های غیر خالی هر ستون ، نوع داده های هر ستون و... را بررسی کنید. همچنین توسط متد ()describe ، آمار توصیفی برای ستون‌های عددی ارائه می‌شود، توجه داشته باشید که ستون‌های حاوی کاراکتر توسط این تابع در نظر گرفته نمی‌شوند. نتیجه ظاهر شده، شامل مولفه‌های، تعداد(Count) ، میانگین(mean) ، انحراف استاندارد (std)، حداقل(min) ، چارک اول (۲۵٪)، چارک دوم یا میانه (۵۰٪) ، چارک سوم (۷۵٪) و حداکثر (max) است که با استانداردهای نرم‌افزارهای محاسبات آماری مطابقت دارد.

چه زمانی پاکسازی داده (Data Cleaning) انجام می‌شود؟

پاکسازی داده یا Data Cleaning در فرایند تحلیل داده، مرحله‌ای بسیار مهم است. چون در همین مرحله میزان دقت و قابل اعتماد بودن داده بررسی می‌شود .داده ها ممکن است در یکی از حالت های ذیل نیاز به پاکسازی داشته باشد .

۱ .داده‌های خالی یا از دست رفته (Empty cells)

در مجموعه داده‌های بزرگ، ممکن است مقادیر خالی یا از دست رفته وجود داشته باشد. برای حل این مشکل، دانشمندان داده از تکنیک‌های پاکسازی داده استفاده می‌کنند تا این مقادیر را با تخمین مناسب پر کنند. به عنوان مثال، اگر یک داده به نام "مکان" وجود دارد که خالی است، دانشمندان داده می‌توانند آن را با میانگین داده‌های "مکان" آن مجموعه داده جایگزین کنند یا یک داده از منبع دیگری پیدا کنند تا آن را پر کنند.

۲ .داده‌های پرت(Outliers)

در مجموعه‌های داده ممکن است داده‌هایی وجود داشته باشند که در مقایسه با سایر داده‌ها از نظر مقدار یا رفتار، دورافتاده و پرت باشند. داده‌های پرت باعث اشتباه در تحلیل می‌شود که به نتایج غلط یا تصمیم‌های نادرست منجر می‌شود. به همین دلیل

شناسایی داده‌های پرت بسیار مهم است. برای حل این مشکل، دانشمندان داده از تکنیک‌های پاکسازی داده استفاده می‌کنند تا داده‌های پرت را در مجموعه‌های داده شناسایی و حذف کنند.

۳. داده های با فرمت ناصحیح (Data in wrong format)

۴. داده های ناصحیح (Wrong data)

۵. داده های تکراری (Duplicates)

۶. قالب‌بندی مجدد داده(Data formatting)

منظور از قالب‌بندی داده، تبدیل داده‌های موجود به فرمت یا نوعی خاصی از داده است که برای تحلیل‌های آینده مناسب باشد.

برای مثال، اگر داده‌ها شامل انواع مختلف متنی، عددی و دسته‌ای باشند، ممکن است برای انجام تحلیل‌های مختلف نیاز باشد تمام این داده‌ها به یک فرمت خاص تبدیل شوند. در این صورت، باید داده‌های دسته‌ای به عددی یا متنی تبدیل شوند تا به‌طور موثر قابل استفاده باشند.

همچنین، ممکن است یک مجموعه داده از چند منبع مختلف به دست آمده باشد و هر کدام دارای ساختارهای متفاوتی باشند. در این حالت، با استفاده از فرآیند فرمت‌بندی داده، می‌شود این داده‌ها را به یک ساختار یکپارچه تبدیل کرد تا تحلیل آن‌ها به شکل موثرتری انجام شود.

برای حل هر یک از مشکلات فوق الذکر استراتژی هایی پیشنهاد شده است که در ادامه به بررسی آن می پردازیم.

۱ .داده‌های خالی یا از دست رفته (Empty cells)

-حذف کامل سطر مربوط به این داده ها

یکی از راه های مقابله با سلول های خالی حذف ردیف هایی است که حاوی سلول های خالی هستند.این کار معمولا زمانی که با داده های بزرگ کار می کنید مشکلی ایجاد نمی کند و حذف چند ردیف تأثیر زیادی در نتیجه نخواهد داشت.

به منظور حذف مقادیر NAN از دیتافریم (Dataframe) از متد dropna استفاده می‌کنیم. این تابع سطرها و ستون‌های NAN را با شیوه‌های مختلف حذف می‌کند .

Example

Return a new Data Frame with no empty cells:

```
import pandas as pd

df = pd.read_csv('data.csv')

new_df = df.dropna()

print(new_df.to_string())
```

-جایگزین کردن سلول های خالی با یک مقدار

متد fillna() به ما اجازه می دهد تا سلول های خالی را با یک مقدار جایگزین کنیم. به این ترتیب شما مجبور نیستید تمام ردیف ها را فقط به دلیل چند سلول خالی حذف کنید.

Example

Replace NULL values with the number 130:

```
import pandas as pd

df = pd.read_csv('data.csv')

df.fillna(130, inplace = True)
```

درمثال فوق همه سلول های خالی در کل دیتافریم با مقدار ۱۳۰ جایگزین می شود .اگر بخواهید فقط سلول های خالی یک ستون (ستون Calories) از دیتافریم با مقدار ۱۳۰ جایگزین شود ، کد شما به شکل ذیل تغیر خواهد کرد .

Example

Replace NULL values in the "Calories" columns with the number 130:

```
import pandas as pd

df = pd.read_csv('data.csv')

df["Calories"].fillna(130, inplace = True)
```

-جایگزین کردن سلول های خالی با مقدار میانگین،میانه یا مد

یک روش رایج جایگزین کردن سلول های خالی با محاسبه میانگین ،میانه یا مد از بقیه داده های آن ستون است .می توان از توابع mean() ، median() و mode() برای انجام محاسبات مربوطه استفاده کرد .

Example

Calculate the MEAN, and replace any empty values with it:

```
import pandas as pd

df = pd.read_csv('data.csv')

x = df["Calories"].mean()

df["Calories"].fillna(x, inplace = True)
```

۲. داده های با فرمت ناصحیح (Data in wrong format)

برای تصحیح این مدل از داده ها دو راه پیشنهاد می شود.یا سطر حاوی داده با فرمت ناصحیح را حذف کنید یا فرمت کل سلول های آن ستون را به فرمت صحیح تغییر دهید .

در دیتافریم زیر دو سلول با فرمت ناصحیح مشاهده می شود.یکی در سطر ۲۲ و دیگری سطر ۲۶ که ستون `Date` باید حاوی یک تاریخ باشد .

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3

```
14         60  '2020/12/14'    104        132     379.3
15         60  '2020/12/15'     98        123     275.0
16         60  '2020/12/16'     98        120     215.2
17         60  '2020/12/17'    100        120     300.0
18         45  '2020/12/18'     90        112       NaN
19         60  '2020/12/19'    103        123     323.0
20         45  '2020/12/20'     97        125     243.0
21         60  '2020/12/21'    108        131     364.2
22         45           NaN    100        119     282.0
23         60  '2020/12/23'    130        101     300.0
24         45  '2020/12/24'    105        132     246.0
25         60  '2020/12/25'    102        126     334.5
26         60      20201226    100        120     250.0
27         60  '2020/12/27'     92        118     241.0
28         60  '2020/12/28'    103        132       NaN
29         60  '2020/12/29'    100        132     280.0
30         60  '2020/12/30'    102        129     380.3
31         60  '2020/12/31'     92        115     243.0
```

از متد to_datetime() در پانداس برای تبدیل فرمت ستون Date به فرمت تاریخ استفاده می کنیم.

Pandas has a `to_datetime()` method for this:

Example

Convert to date:

```
import pandas as pd

df = pd.read_csv('data.csv')

df['Date'] = pd.to_datetime(df['Date'])
```

```
print(df.to_string())
```

و نتیجه را به این صورت مشاهده خواهید کرد.

همانطور که از نتیجه می بینید، تاریخ در ردیف ۲۶ تصحیح شد، اما تاریخ خالی در ردیف ۲۲ با یک مقدار NaT جایگزین شده است.که برای تصحیح آن می بایست از یکی از روش هایی که قبلا توضیح داده شد استفاده کنید.

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        450  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
```

```
14         60  '2020/12/14'    104        132     379.3
15         60  '2020/12/15'     98        123     275.0
16         60  '2020/12/16'     98        120     215.2
17         60  '2020/12/17'    100        120     300.0
18         45  '2020/12/18'     90        112       NaN
19         60  '2020/12/19'    103        123     323.0
20         45  '2020/12/20'     97        125     243.0
21         60  '2020/12/21'    108        131     364.2
22         45           NaT    100        119     282.0
23         60  '2020/12/23'    130        101     300.0
24         45  '2020/12/24'    105        132     246.0
25         60  '2020/12/25'    102        126     334.5
26         60  '2020/12/26'    100        120     250.0
27         60  '2020/12/27'     92        118     241.0
28         60  '2020/12/28'    103        132       NaN
29         60  '2020/12/29'    100        132     280.0
30         60  '2020/12/30'    102        129     380.3
31         60  '2020/12/31'     92        115     243.0
```

در اینجا سطر مربوطه را با با متد ()dropna حذف می کنیم.

Example

Remove rows with a NULL value in the "Date" column:

```
df.dropna(subset=['Date'], inplace = True)
```

۳-داده های ناصحیح (Wrong data)

داده های اشتباه لزومی ندارد که حتما سلول های خالی یا قالب اشتباه باشد، بلکه ممکن است در ثبت داده اشتباهی رخ داده باشد، مثلاً ممکن است کاربر به جای "۱,۹۹" "۱۹۹" را ثبت کرده باشد.

گاهی اوقات می توانید با نگاه کردن به مجموعه داده ها ، داده های اشتباه را تشخیص دهید، زیرا با شناختی که وجود دارد می توان محدوده قابل انتظار داده ها را دریافت. اگر به مجموعه داده های ذیل نگاهی بیندازید، می بینید که در ردیف ۷، مدت زمان ۴۵۰ است، اما برای تمام ردیف های دیگر مدت زمان بین ۳۰ تا ۶۰ است.با توجه به اینکه در این ستون زمان جلسات تمرینی یک نفر ثبت شده است، نتیجه می گیریم که این فرد ۴۵۰ دقیقه تمرین نکرده است.

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0

```
 21        60  '2020/12/21'    108       131     364.2
 22        45           NaN    100       119     282.0
 23        60  '2020/12/23'    130       101     300.0
 24        45  '2020/12/24'    105       132     246.0
 25        60  '2020/12/25'    102       126     334.5
 26        60      20201226    100       120     250.0
 27        60  '2020/12/27'     92       118     241.0
 28        60  '2020/12/28'    103       132       NaN
 29        60  '2020/12/29'    100       132     280.0
 30        60  '2020/12/30'    102       129     380.3
 31        60  '2020/12/31'     92       115     243.0
```

یکی از روش های تصحیح داده های ناصحیح جایگزین کردن آن با مقادیر دیگر می باشد.از آنجا که مثال فوق یک اشکال تایپی به نظر می رسد و می توان حدس زد که مقدار صحیح ۴۵ باشد.

Example

Set "Duration" = 45 in row 7:

```
df.loc[7, 'Duration'] = 45
```

این روش در مورد داده های کوچک می تواند عملی باشد که داده های اشتباه را یکی یکی بررسی و اصلاح نماییم اما در خصوص داده های بزرگ باید راه حل دیگری اتخاذ نمود. برای جایگزینی داده های اشتباه در مجموعه داده های بزرگتر، می توانید قوانینی ایجاد کنید، به عنوان مثال برخی از مرزها را برای مقادیر قانونی یک داده تعیین کنید و هر مقداری را که خارج از مرزهای تعریف شده باشد جایگزین کنید.

Example

Loop through all values in the "Duration" column.

If the value is higher than 120, set it to 120:

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.loc[x, "Duration"] = 120
```

روش دیگر مدیریت داده های ناصحیح حذف سطری است که حاوی آن داده می باشد.در این روشما اجباری به پیدا کردن مقدار جایگزین ندارید و از آن داده در انجام تحلیل ها صرف نظر می نمایید.

Example

Delete rows where "Duration" is higher than 120:

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.drop(x, inplace = True)
```

۴- داده های تکراری (Duplicates)

ممکن است دیتافریم حاوی سطر های داده تکراری باشد.مانند آنچه در سطر های ۱۱ و ۱۲ از دیتاست ذیل مشاهده می شود.

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1

```
 9          60  '2020/12/10'     98        124     269.0
 10         60  '2020/12/11'    103        147     329.3
 11         60  '2020/12/12'    100        120     250.7
 12         60  '2020/12/12'    100        120     250.7
 13         60  '2020/12/13'    106        128     345.3
 14         60  '2020/12/14'    104        132     379.3
 15         60  '2020/12/15'     98        123     275.0
 16         60  '2020/12/16'     98        120     215.2
 17         60  '2020/12/17'    100        120     300.0
 18         45  '2020/12/18'     90        112       NaN
 19         60  '2020/12/19'    103        123     323.0
 20         45  '2020/12/20'     97        125     243.0
 21         60  '2020/12/21'    108        131     364.2
 22         45           NaN    100        119     282.0
 23         60  '2020/12/23'    130        101     300.0
 24         45  '2020/12/24'    105        132     246.0
 25         60  '2020/12/25'    102        126     334.5
 26         60      20201226    100        120     250.0
 27         60  '2020/12/27'     92        118     241.0
 28         60  '2020/12/28'    103        132       NaN
 29         60  '2020/12/29'    100        132     280.0
 30         60  '2020/12/30'    102        129     380.3
 31         60  '2020/12/31'     92        115     243.0
```

برای کشف ردیف های تکراری از متد ()duplicated استفاده می کنیم. این متد یک مقدار بولین برای هر سطر برمی گرداند که برای سطر های تکراری برابر True و سطر های غیر تکراری برابر False می باشد.

Example

Returns `True` for every row that is a duplicate, otherwise `False`:

```
print(df.duplicated())
```

برای حذف ردیف های تکراری از متد drop_duplicates() استفاده می کنیم .

Example

Remove all duplicates:

```
df.drop_duplicates(inplace = True)
```

نکته: به یاد داشته باشید inplace = True سبب می شود تغییرات در دیتافریم اصلی ذخیره شود.

همبستگی(Correlation)

همبستگی (Correlation) یک رابطه آماری بین دو متغیر تصادفی یا دو دسته داده است که لزوما به معنای ارتباط علی و معلولی آنها نیست. همبستگی روابط متغیرها را به صورت دوبه‌دو و جدا از تاثیر همزمان سایر متغیرها بررسی می‌کند.

عمده ترین روش‌های شناخته شده در این زمینه توسط پیرسون (برای داده‌های نرمال) و اسپیرمن (برای داده‌های غیرنرمال) ارائه شد. همبستگی برای بررسی نوع و میزان رابطه متغیرها استفاده می‌شود. در حالیکه رگرسیون پیش‌بینی روند آینده یک متغیر وابسته براساس یک مجموعه روابط بین متغیر وابسته با یک چند متغیر پیش‌بین (مستقل) است که در گذشته ثبت و ضبط شده است .

پیدا کردن همبستگی بین ستون های داده در پانداس

متد corr() یکی از قابلیت های مهم کتابخانه پانداس را ارائه میدهد.این متد رابطه بین هر ستون در دیتاست را با سایر ستون ها مشخص می کند.به مثال زیر توجه کنید.

Example

Show the relationship between the columns:

```
df.corr()
```

Result

```
          Duration     Pulse  Maxpulse  Calories
Duration  1.000000 -0.155408  0.009403  0.922721
Pulse    -0.155408  1.000000  0.786535  0.025120
Maxpulse  0.009403  0.786535  1.000000  0.203814
Calories  0.922721  0.025120  0.203814  1.000000
```

توجه داشته باشد که این متد ستون های غیر عددی را در نظر نمی گیرد.

تفسیر جدول خروجی

نتیجه متد corr() جدولی ست که با یک سری اعداد نشان می دهد چقدر بین مقادیر ستون های عددی در دیتاست ارتباط برقرار است.این اعداد بین ۱- و ۱ هستند .

- مقدار ۱ نشان میدهد که یک رابطه ۱ به ۱ وجود دارد (یک همبستگی کامل) و برای این مجموعه داده ها، هر بار که یک مقدار در ستون اول بالا می رفت، مقدار دیگر نیز بالا می رفت.
- ۰,۹ نیز رابطه خوبی است و اگر مقدار در ستون اول را افزایش دهید احتمالاً مقدار دیگر نیز افزایش خواهد یافت.
- ۰,۹- به اندازه ۰,۹ رابطه خوبی خواهد بود، اما اگر یک مقدار را افزایش دهید، مقدار دیگر احتمالاً کاهش می یابد.

- ۰,۲ به معنای یک رابطه خوب نیست، به این معنی که اگر یک مقدار بالا رفت به این معنی نیست که مقدار دیگر افزایش می یابد.

همبستگی خوب : بستگی به کاربرد دارد، اما می‌توان گفت که مقدار همبستگی باید حداقل ۰,۶ (یا ۰,۶-) باشد تا آن را یک همبستگی خوب بدانیم.

همبستگی کامل :

به طور مثال وقتی ستون "Duration" عدد ۱,۰۰۰۰۰۰ را نشان میدهد ، که منطقی است، هر ستون همیشه یک رابطه و همبستگی کامل با خودش دارد.

"Duration" و "Calories " همبستگی ۰,۹۲۲۷۲۱ دارند، که همبستگی بسیار خوبی است، و ما می توانیم پیش بینی کنیم که هر چه بیشتر ورزش کنید، کالری بیشتری می سوزانید و برعکس: اگر کالری زیادی سوزانده اید، احتمالا تمرین طولانی انجام داده اید.

همبستگی ضعیف:

"Duration" و "Maxpulse" یک همبستگی ۰,۰۰۹۴۰۳ دارند که همبستگی بسیار ضعیفی است، به این معنی که ما نمی توانیم حداکثر پالس را فقط با نگاه کردن به مدت زمان انجام کار پیش بینی کنیم و بالعکس.

مصور سازی داده ها (Data visualization)

تجسم داده‌ها تکنیکی است که به دانشمندان داده (Data Scientists) اجازه می‌دهد تا داده‌های خام را به گراف و دیاگرام تبدیل کنند. چنین تصاویری خواندن و درک داده‌ها را آسان‌تر می‌کنند .بسیاری از کتابخانه‌های پایتون اجازه تغییر یا دست‌کاری داده‌ها را می‌دهندNumpy :، Pandas، Matplotlib، Tensorflowو.... در میان این اسامی، Matplotlibو Seaborn از سایر موارد کارآمدتر و متداول‌تر هستند .

Matplotlib یک کتابخانه قدرتمند برای رسم نمودار در پایتون است که برای ایجاد انواع نمودارهای ثابت، متحرک و تعاملی استفاده می‌شود. هدف اصلی Matplotlib ارائه ابزارها و عملکرد به کاربران برای نمایش داده‌ها به‌صورت گرافیکی است تا تجزیه‌وتحلیل و درک هریک از آن‌ها آسان‌تر شود .

انواع نمودار در پایتون

- نمودار میله‌ای(Bar Chart)
- نمودار خطی(Line Plot)
- نمودار پراکندگی(Scatter plot)
- نمودار هیستوگرام(Histogram)
- نمودار جعبه‌ای(Box Plot)
- نمودار دایره‌ای(Pie Chart)

برای تصویرسازی داده ها از Pyplot که یک زیر ماژول از کتابخانه Matplotlib است استفاده میکنیم. متد ()plot برای رسم انواع نمودار به کار می رود.

Example

Import pyplot from Matplotlib and visualize our DataFrame:

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv(r"C:\Users\kh.hanani\Downloads\data (1).csv")
```

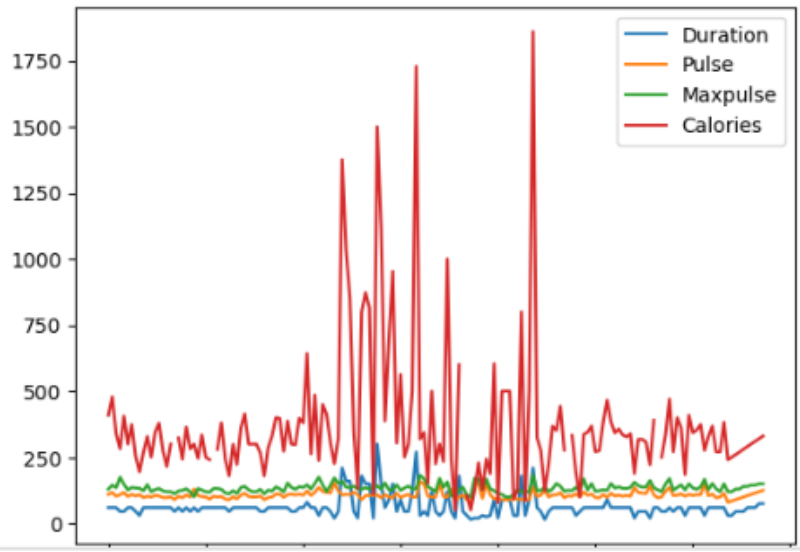
```
df
```

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
...	...	...	...	...
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

169 rows × 4 columns

```
df.plot()
```

```
<Axes: >
```

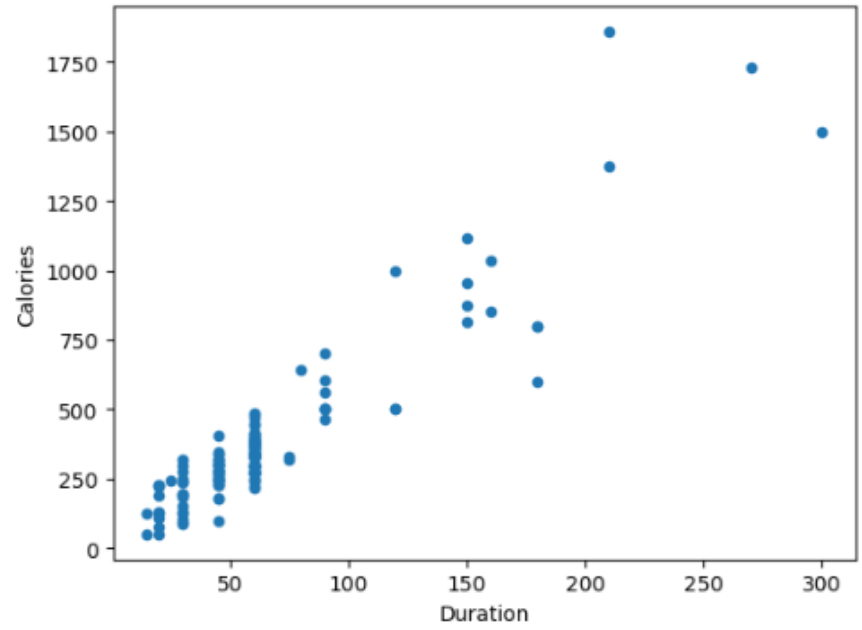


نمودار پراکندگی(Scatter plot)

در متد plot() می توان از آپشن هایی برای تعیین نوع نمودار رنگ و بسایر تنظیمات در داخل پرانتز استفاده کرد.به طور مثال در صورتیکه بخواهید نمودار پراکندگی داشته باشید از kind = 'scatter' استفاده نمایید.برای رسم نمودار scatter دو متغیر برای محور x و y مورد نیاز می باشد.در مثال ذیل از ستون "Duration" برای محور x و "Calories" برای محور y استفاده شده است.پس مقادیر x و y را x = 'Duration', y = 'Calories' قرار می دهیم.

```
df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

plt.show()
```



در مثال های پیشتر مشاهده کردیم که همبستگی بین دو متغیر "Duration" و "Calories" مقدار 0.922721 بود و ما به این نتیجه رسیدیم که مدت زمان تمرین بیشتر به معنای کالری سوزی بیشتر است.که نمودار فوق نیز بیانگر همین واقعیت است.

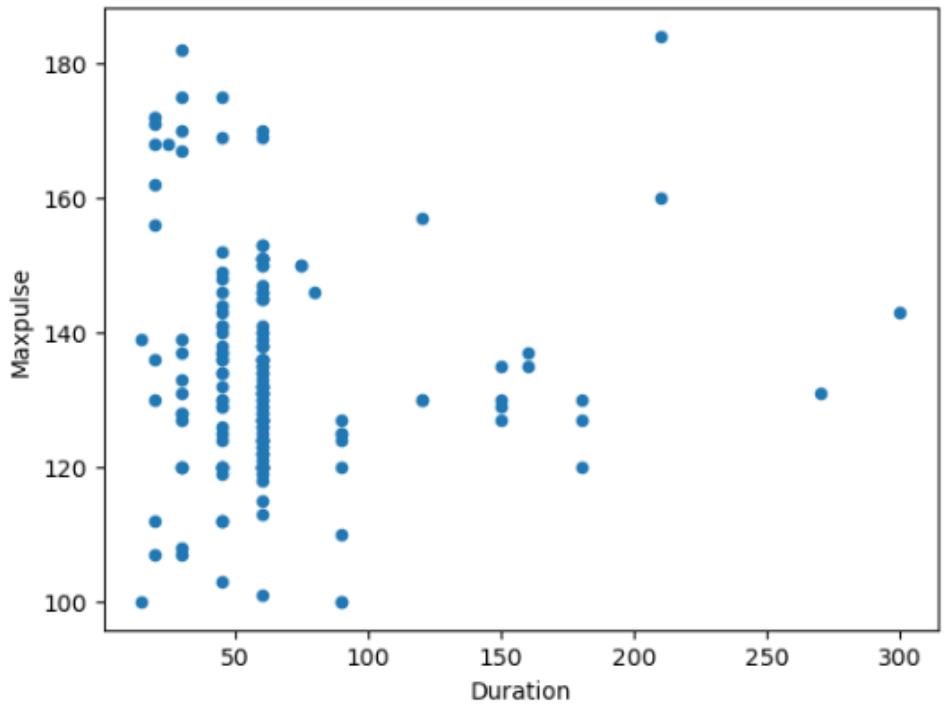
حال scatterplot دیگری رسم می کنیم که یک همبستگی بد بین دو ستون "Duration"و"Maxpulse" را نشان می دهد .مقدار این همبستگی 0.009403 محاسبه شده بود.

Example

A scatterplot where there are no relationship between the columns:

```
df.plot(kind = 'scatter', x = 'Duration', y = 'Maxpulse')
```

```
<Axes: xlabel='Duration', ylabel='Maxpulse'>
```



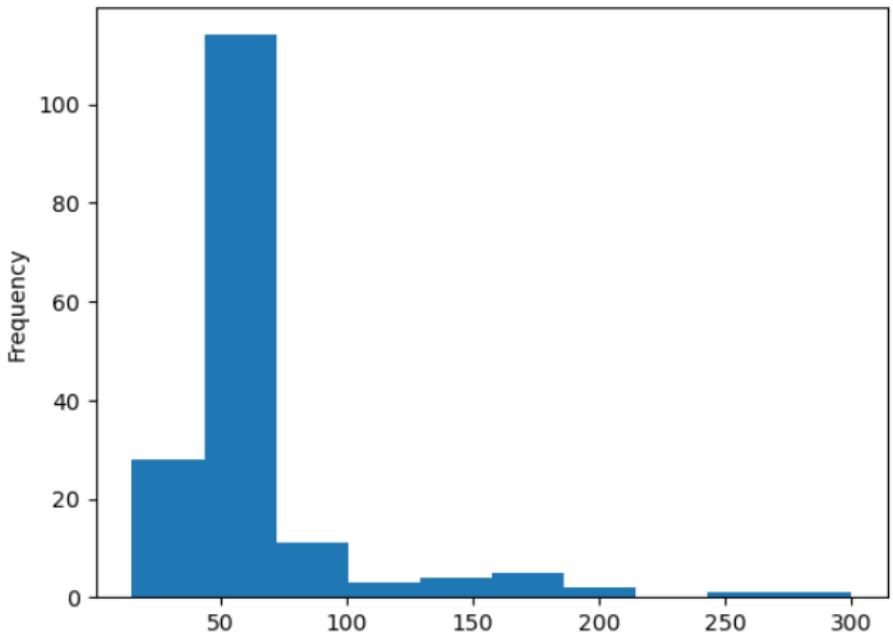
نمودار هیستوگرام(Histogram)

برای رسم نمودار هیستوگرام کافیست kind = 'hist' قرارداده شود.این نمودار فقط به یک متغیر نیاز دارد. یک هیستوگرام فراوانی هر بازه را به ما نشان می دهد، به عنوان مثال. چند تمرین بین ۵۰ تا ۶۰ دقیقه طول کشیده است.در نمودار زیر ما از ستون "Duration" استفاده کرده ایم.

Example

```
df["Duration"].plot(kind = 'hist')
```

```
<Axes: ylabel='Frequency'>
```



نمودار گویای این است که در دیتاست ما بیشتر از ۱۰۰ تمرین وجود دارد که بین ۵۰ تا ۶۰ دقیقه طول کشیده است .

توضیح: سایر انواع نمودار و تنظیمات پیشرفته تر جهت رسم و تحلیل نمودار ها در پایتون ، در پودمان ۳ آموزش داده خواهد شد.

منابع:

Python data analysis and visualization course in Udemy

Machine Learning with Python course in Udemy

https://www.w3schools.com/python/

https://www.hackerrank.com/domains/python