Comprehensive AI

پنجشنبه, ۲۸ اسفند ۱۳۹۹ ۰۱:۱۰ ب.ظ

نکات مربوط به آموزش جامع تاپ لرن

تمام فایل های گذاشته شده در oneDriveذخیره شده
https://1drv.ms/u/s!Anu7yTwLSfA7bODnKuQxLn02G5M?e=QaBteN

دو معادله و دو مجهول
شیب نمودار
الگاریتمlog
واریانس
انتگرال
زیگما
واریانس
نمودار توزیع نرمال

**
≃ x ->
یعنی در یک بار اجرا xدر آمده که لزوما جواب
ما هم دوباره ایکس در نمیاد ولی حول و حوش ایکسه
**

---

install anaconda
>> conda install pydotplus

---

install tensorflow
if you system has NVIDIA gpu
>> conda install tensorflow-gpu
if not:
>> conda install tensorflow

---

update tensorflow
python -m pip install --upgrade https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.12.0-py3-none-any.whl

---

open jupyter
>> cd .... -> open folder you want
>> jupyter notebook
now jupyter run in you browser localHost

لگاریتم

$4^n = 16$
X = log(16, 4) #عدد ۱۶ بالا و ۴ را پایین لوگ مینویسیم
= 2
https://blog.faradars.org/logarithms-introduction/

---

read data from Excel file (.csv)

in excel csv file:
title count
test1 2
test2 5
test3 9
test4 4

___
%matplotlib inline
import numpy as np
import pandas as pd
result = pd.read_csv('Excel-CSV File Address')
result.head()
->
  title count
0 test1 2
1 test2 5
2 test3 9
3 test4 4

فقط دوتای اول
result.head(2)
->
  title count
0 test1 2
1 test2 5

فقط دوتای آخر
result.tail(2)
->
  title count
2 test3 9
3 test4 4

انواع داده ها

<- کمّی
  int
    1, 2, 3 ,...
  flout
    1.1 , 1.5, 24.3
<- کیفی
  مقدماتی، پیشرفته

ترتیبی ->
مثلا ستاره های امتیازی مربوط به فیلم

میانه اعداد

median
2,4,5,7,4,8,9
--sort-- > 2,4,4,5,7,8,9
median = 5
وسط عدد در تعداد اعداد فرد
و ۲ تای وسط یا میانگین دوتای وسط در تعداد اعداد زوج

مود
mode

2,4,4,5,7,8,9
mode = 4
بیشترین تکرار

میانگین
mean

مجموع
_____÷
تعداد

Tolerance بازه تغییر -

لیستی از دیتا که از یک بازه ای داره استفاده میکند
کوچک ترین عدد و بزرگترین عدد میشود بازه تغییر

تا عدد 20000
10000 ..... ---------- 45000
min - 10000
max - 45000
count = 20000
mean = 25000 # با میانگین ۲۵۰۰۰
Tolerance = 20000
25000 + 20000 = 45000
25000 - 20000 = 10000

ممکن اعدادی که به ما میدهد کمی بالا پایین تر از ۱۰۰۰۰ و ۴۵۰۰۰ هزار باشد#
ولی تفاوت کم
به این دلیل در پایین میانگین کمی بالا پایین میشود

بازسازی مباحث در numpy

import numpy as np
numbers = np.random.normal(Mean, انحراف معیار, Count) #(25000, 15000, 20000)
np.mean(numbers)
-> ≃ 25145
هر بار اجرا یک میانگین می دهد که حول بیست و
پنج هزار تا است و اختلاف زیادی با بیست و پنج هزار ندارد

np.mean([3,4,5])
-> 4.0

num = [1,2,3,4,5,6,7]
np.median(num)
->
4.0

نمودار گرافیکی اعداد

```
%matplotlib inline
import matplotlib.pyplot as plt
numbers = np.random.normal(25000, 15000, 20000)
plt.hist(numbers,50)  #50 is Y Axis
plt.show()
```

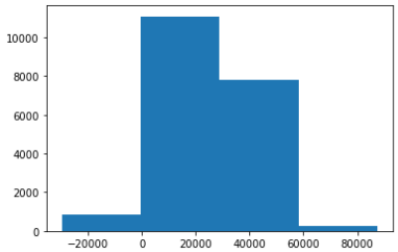
np.median(numbers)
-> ≃ 25217.466125555038

point

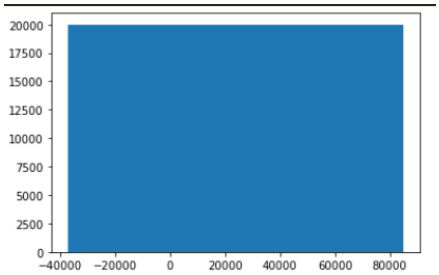
بالا در مورد تولرانس صحبت شد
Tolerance = 20000
25000 + 20000 = 45000
25000 - 20000 = 10000
ولی اگر نمودار و ببینیم بازه ی بیشتر
از ۴۵۰۰۰ و بازه ی کمتر از ۱۰۰۰۰ هم داریم.
ما در اعداد نرمال شکل زنگوله ای داریم و
داده ها در یک بازه می رسند به میانگین و
مانند آینه و تقارن به پایین دوباره بر میگردند
به این نمودار توزیع نرمال گفته میشه
این ساختار حدود ۹۵% دیتا رو نشون میده
plt.hist(datas, bins)
bins -> تعداد داده ای که در بازه نمایش داده شوند مثلا اگر یک بذاریم فقط یک داده را نشان میدهد
که یک داده کل نمودار رو فرا میگیره، ما در بینز ۳ میتوانیم توزیع نرمال بودن یا نبودنشو تا حد زیادی بفهمیم

Description

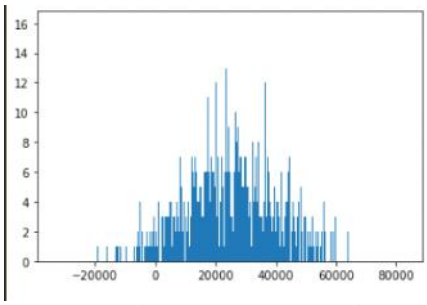
```
plt.hist(numbers,50)
```
50 در اصل تعداد محور ها است مثلا



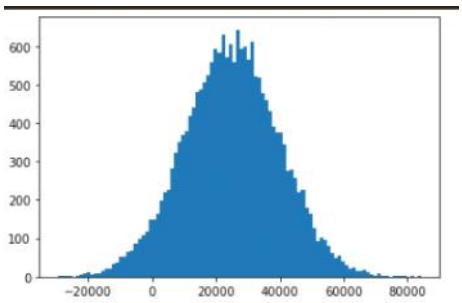

```
plt.hist(numbers,4)
```




```
plt.hist(numbers,1)
```




```
plt.hist(numbers,10000)
```




```
plt.hist(numbers,100)
```

انواع نمودار ها و پلات ها در متپلات لیب

plots
https://7learn.com/ai/machine-learning/matplotlib-and-seaborn-library-tutorial

---

درک تاثیر داده بر روی نتیجه

import ⇶
numbers = np.random.normal(25000, 15000, 20000)
np.median(numbers)
-> ≃ 25217.466125555038

np.mean(numbers)
-> ≃ 25153.931429364045

numbers = np.append(numbers,[100000000])
np.median(numbers)
-> ≃ 25218.286816999014 #Almost same

np.mean(numbers)
-> ≃ 30152.42380817364 #Very changed

point
باید داده هارو مبنای یکسان بگیریم
مثلا در بازه ی ۰ تا ۱
تا نتیجه گیری درستی از داده داشته باشیم
مثلا در آمار درآمد ممکن است یکی در آمد نجومی
داشته باشد و این باعث میشود تا نتیجه گیری
غلطی در مورد داده ها داشته باشیم

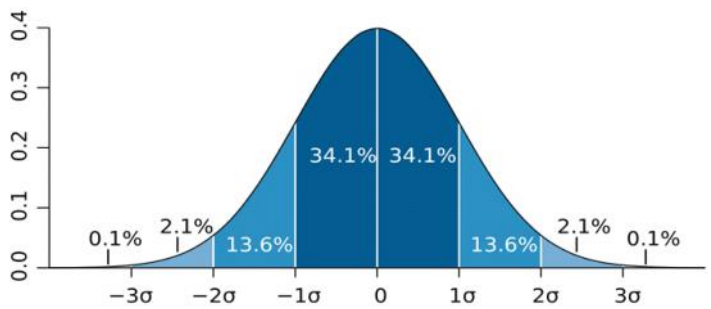
تولید لیست عدد رندوم

ages = np.random.randint(18, high= 90, size= 500) #500 :min18 max90

بیشترین تکرار داخل مجموعه

from scipy import stats
stats.mode(ages)
-> ModeResult(mode=array([42]), count=array([13]))
عدد ۴۲ سیزده بار تکرار شده
در ضمن ممکنه آرایه چند عدد باشه

میزان انحراف معیار در توزیع نرمال



u = میانگین
σ = مثلا انحراف معیار

اتحراف معیار، در اصل پراکندگی داده را نشان میدهد

اگر میانگین و انحراف معیار رو داشته باشیم میتوانیم یک
نمودار توزیع نرمالمون رو داشته باشیم
u +- σ -> 68%
u +- 2σ -> 95%
u +- 3σ -> 99.7%

یعنی از میانگین منهای انحراف معیار تا میانگین به علاوه انحراف معیار
برابر با ۶۸ درصد از داده هایمان میشود

در ادامه نحوه محاسبه اش را میگوییم

واریانس

میانگین اختلاف مربعات از مینانگین

1. میانگین گیری

μ -> mean میانگین
(1,4,5,4,8)
μ -> 4.4

2. تمام اعضا تک تک منهای میانگین

(-3.4, -0.4, 0.6, -0.4, 3.6)

3. تک تک اعضا، به توان دو

(11.56, 0.16, 0.36, 0.16, 12.96)

4. میانگین از کل اعضا

μ_2 = 5.2
σ^2 = 5.2 : variance واریانس
σ -> انحراف معیار

محاسبه انحراف معیار

= انحراف معیارstd = STandard Deviation
انحراف معیار رادیکال واریانس میباشد
$\sqrt{\sigma^2} = \sigma \simeq 2.24$ -> <- انحراف معیارstd
هر چقدر انحراف معیار به صفر نزدیک تر باشه
متوجه میشویم اون مجموعه ای که داریم
اعدادش به هم نزدیک تر هستند
یعنی ۶۸ درصد از داده های ما بین
4.4 + 2.24 , 4.4 - 2.24

___
مثال۲:
میانگین نمره ی دانش آموزان در آزمون ۱ برابر ۱۲
و در آزمون دوم هم برابر با ۱۲ شده است

mean_1 = 12 , σ = 3
mean_2 = 12 , σ = 0.5
در آزمون اول واریانس با ۳ شده
در نتیجه پراکندگی نمرات بسیار زیاد هست
یعنی ۶۸ درصد مجموعه برابر با
12 +- 3 -> 9 , 15
ولی در آزموون دوم
12 +- 0.5 -> 11.5 , 12.5
نمرات بالا کمی پایین تر آمده ولی در عوض
نمرات پایین هم بسیار بالا آمده
در نتیجه مدیر میفهمد که این دبیر برای مدرسه مفید است
چون بچه ها پیشرفت داشتند

نکته ای درباره واریانس

(1,4,5,4,8)

Population Variance

$$\mu = \frac{1+4+4+5+8}{5} = 4.4$$

ولی وقتی داده ی ما، بخشی کوچک از یک دیتای
مثلا ۱۰۰ تایی است و اومدیم ۵ تا نمونه ازش بدست
اوردیم. اینجا دیگه تقسیم بر تعداد که ۵ تا است نمیکنیم
بلکه تقسیم بر تعداد - ۱ میکنیم
5-1 = 4 -> μ = 22÷4 = 5.5
در نتیجه ساختار واریانس و جواب واریانس کامل عوض میشود

Sample Variance

$$\mu = \frac{1+4+4+5+8}{5 - 1} = 5.5$$

این هم فرقی با واریانس ندارد ولی از بخشی از کل دیتا
داریم استفاده میکنیم به جای کل دیتا

___

Population Variance

$$\sigma^2 = \frac{\sum(x - \mu)^2}{N}$$

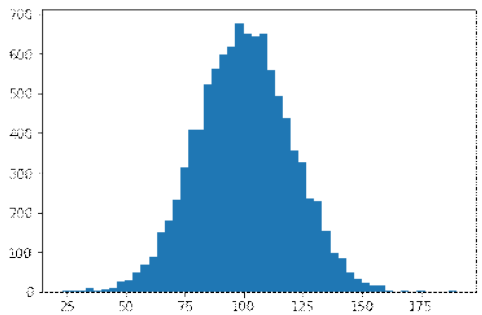
Sample Variance

$$S^2 = \frac{\sum(x - \mu)^2}{N - 1}$$

محاسبه اتوماتیک ی انحراف معیار و واریانس

```
import numpy as np
import matplotlib.pyplot as plt
numbers = np.random.normal(100.0, 20.0, 10000) #mean100, σ = 20, 10000number
plt.hist(numbers,50)
plt.show()
```

out →



numbers.std() # انحراف معیار
≃ 20.039245675158945

numbers.var()
≃ 401.57136722937645 -> std * std = 20 * 20 ≃ 400

هر چقدر *انحراف معیار* *کمتر* باشه بازه ی اعداد کوچکتر میشود
یعنی محور افقی کمتر میشود
مثلا : بازه اعداد ، در مثال های زیر
np.random.normal(100.0, 20.0, 10000)
-> ≃ 40 , 160

np.random.normal(100.0, 2.0, 10000)
-> ≃ 94 , 106

در اصل هرچقدر کمتر باشه اعداد ما به میانگین نزدیک تر است
که در اینجا میانگین ۱۰۰ است و اگر انحراف معیار ۰
بگذاریم محدوده فقط ۱۰۰ میشود یک خط صاف میانگین

بدست آوردن احتمال وجود داشتن عدد توی یک بازه در نمودار توزیع نرمال

ما برای این کار باید مساحت زیر قسمت نموداری
که میخواهیم عدد درونش باشد را محاسبه میکنیم
که مساحت درصد احتمال را نشان می دهد
نهایت احتمال صد درصد است که میشود ۱
یعنی مساحت کل زیر نمودار ۱ است
این کار را میتوانیم با توابعی انجام دهیم مانند
تابع جرم احتمال
تابع چگالی احتمال

تابع جرم نرمال

PMF = Probability Mass Function
برای اعداد گسسته
به وسیله ی خود عدد درصد محاسبه میشود

تابع چگالی احتمال

PDF = Probability Density Function
برای اعداد پیوسته
بر اساس مساحت زیر نمودار درصد محاسبه میشود

انتگرال

زیگما

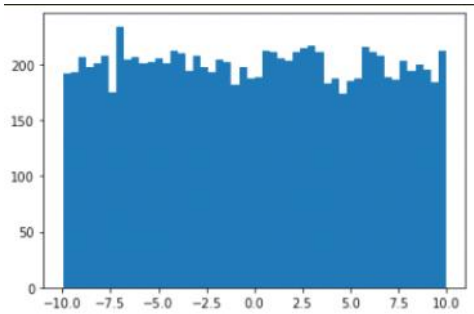
یک مثال

مثال:
پنجاه تا خیابان داریم و کرایه اتوبوس میانگین ۱۰۰۰۰۰ که تقریبا کرایه اتوبوس توی
هر کدام از خیابان ها +- ۲۰۰۰ می شود
np.random.normal(10000, 2000, 100000)
این در تابع توزیع نرمال است

تابع توزیع یکنواخت

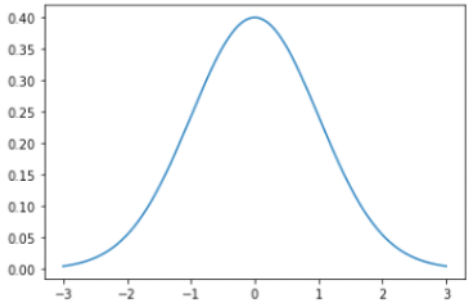
```
import numpy as np
import matplotlib.pyplot as plt
values = np.random.uniform(-10.0, 10.0, 10000)
plt.hist(values,50 )
plt.show()
```

همه ی اعداد حدود ۲۰۰ اند
چون پنجاه ضرب در ۲۰۰ میشه ۱۰۰۰۰
در اصل ۱۰۰۰۰ را تقسیم بر ۵۰ میکنیم
و میگوییم بازه ده تا منفی ده را به ۵۰ قسمت تقسیم کن
در اینجا دیگر نمودار زنگوله ای نیست و مربعی است

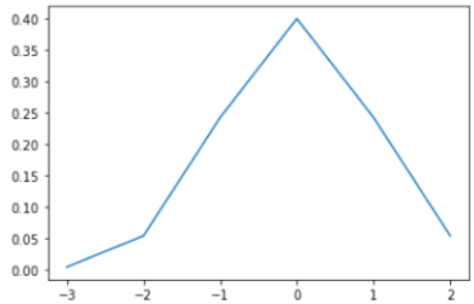


نمودار تابع توزیع نرمال با یک نوع دیگر

```
from scipy.stats import norm
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(-3, 3, 0.001)
plt.plot(x, norm.pdf(x))  #pdf -> احتمال چگالی تابع
```



```
x = np.arange(-3, 3, 1)
```

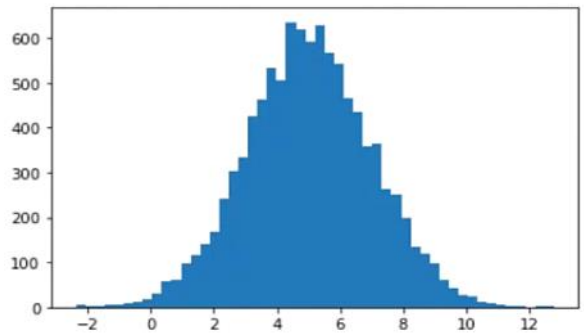


میبینیم اگر یک دهیم، در اصل داده هارو کم کردیم و نرمی نمودار رو کم کردیم

نمایش خود دیتا

```
import numpy as np
import matplotlib.pyplot as plt
mu = 5.0
sigma = 2.0
values = np.random.normal(mu, sigma, 10000)
plt.hist(values, 50)
plt.show
```

Out>



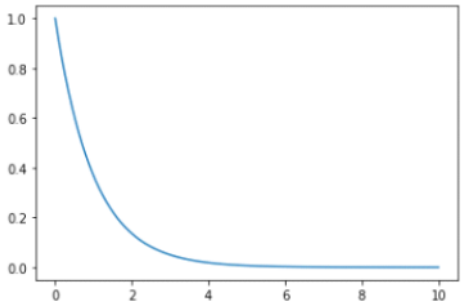
نمودار تابع توزیع نمایی

#expon -> exponential -> نمایی

```
from scipy.stats import expon
import matplotlib.pyplot as plt
```

```
x = np.arange(0, 10, 0.001)
plt.plot(x, expon.pdf(x))
```

این تابع نزولی است



_____

point

توضیح دوباره

arange(0, 10, 1)

عدد یک رو که میبینیم در اصل فاصله اعداده

یعنی بین ۰ تا ۱۰ با فاصله ی ۱ ده تا عدد میده

یا مثلا با فاصله ی ۰/۵ بیست تا عدد میده

این رو با تغییر اعداد روی نمودار بهتر میشه فهمید

مرور کلی توابع

تابع چگالی احتمالPDF
تابع توزیع نرمال
تابع توزیع یکنواخت
تابع توزیع نمایی
....
تابع جرم احتمالPMF
تابع توزیع برنولی
تابع پوآسون یا پاسان

تکمیل نشده $$$ $$$

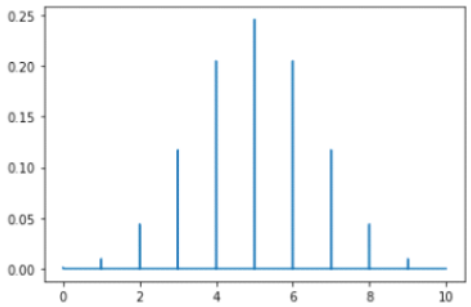
موارد احتمال زیر در مورد برنولی #
چولگی
کشیدگی
آنتروپی
تابع مولد گشتاور
تابع توزیع تجملی

کتاب دکتر عادل آذر آمار و کاربرد آن در مدیریت
جان فروند آمار و احتمال

نشان دادن تابع توزیع برنولی
تابع جرم احتمال
PMF

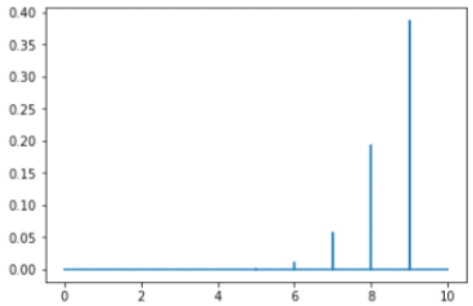
```
from scipy.stats import binom
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(0, 10, 0.001)
plt.plot(x, binom.pmf(x, 10, 0.5))
```

Out>

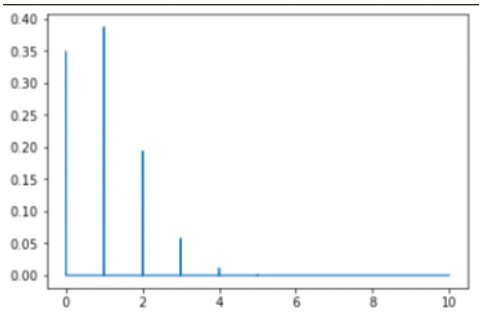


---

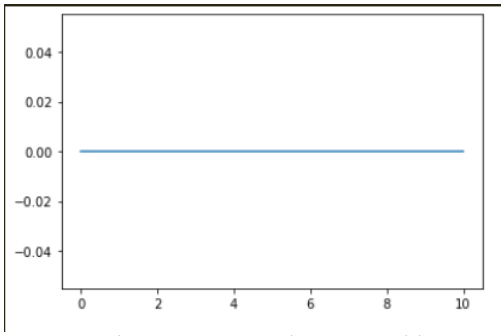
point
درصد اختمال یا شانس موفقیت → 0.5
مثلا اگر روی ۰/۹ باشه گرایش پیدا میکنه
روی ۱۰ و اگر مثلا ۰/۱ باشه گرایش پیدا میکنه
روی ۱ و وقتی میانه میذاریم یا همان ۰/۵ اعداد
مانند توزیع نرمال میشوند و پخش میشوند



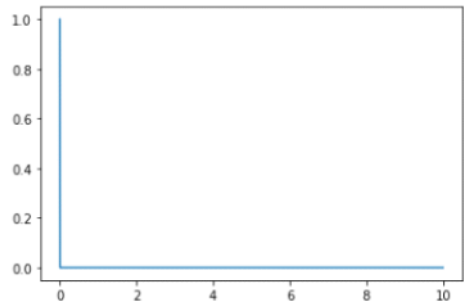

```
plt.plot(x, binom.pmf(x, 10, 0.9))
```
میبینیم اینجا نزدیک به یک (همان ۱۰ ای که بهش دادیم) شد




```
plt.plot(x, binom.pmf(x, 10, 0.1))
```
میبینیم اینجا نزدیک به ۰ شد




```
plt.plot(x, binom.pmf(x, 10, 1))
```




```
plt.plot(x, binom.pmf(x, 10, 0))
```

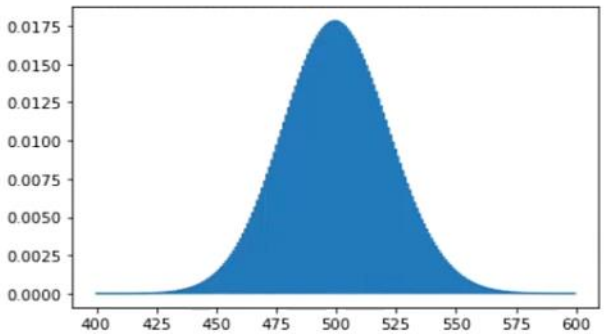
و ۰/۵ که اول دادیم بودیم چون میانه بود، مثل توزیع نرمال شد

نشان دادن تابع پاسان

یا پو آسون / poisson

```
import numpy as np
from scipy.stats import poisson
import matplotlib.pyplot as plt

mu = 500  # میانگین
x = np.arange(400,600, 0.5) #
plt.plot(x, poisson.pmf(x, mu))
plt.show()
```

Out>



Description

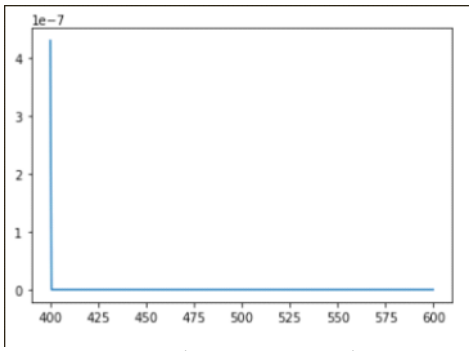
مثلا میانگین روزانه بازدید یک سایت ۵۰۰ نفر است
از ۴۰۰ تا ۶۰۰ نفر بازدید کننده ممکنه باشه

این نمودار غلط انداز است و شبیه توزیع نرمال است
ولی در اصل توزیع نرمال نیست

مثلا میخواهیم ببینیم عدد ۵۵۰ چقدر احتمال داره
با توجه به نمودار عدد ۵۵۰ احتمال ۰/۰۰۲ احتمال وقوع دارد

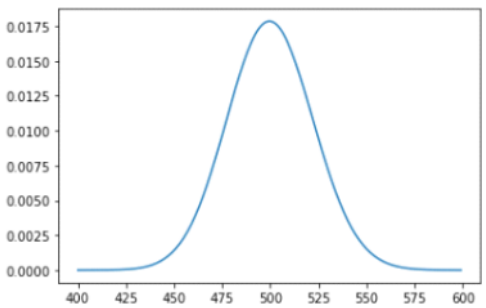
این نمودار برای اینه که ببنیم، احتمال تعداد بازدید چقدر هست
مثلا چقدر احتمال داره که در روز بازدید کننده داشته باشد

وقتی روی نمودار عددی مثل ۰/۶ را بذاریم نمایش داده نمیشه و اعداد
خاصّی مثل ۰/۵ و ۱ و ۵ و ۱۰ و ... نمایش داده میشه




```
x = np.arange(400,600, 0.6)
```

اعداد یک هم به صورت عمودی و آبی نمایش داده میشن و نمودار
عدد ۱ بسیار نرم است و اون رنگ آبی را ندارد



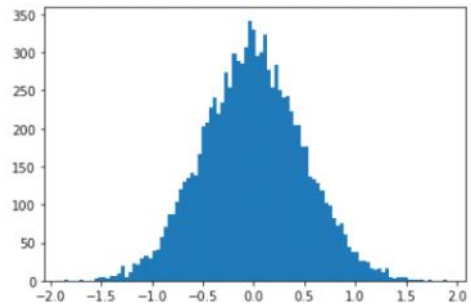

```
x = np.arange(400,600, 1)
```

اگر بخواهیم احتمال وقوع حادثه به
میرود کار به ثابت که یا زمانی فاصله توی تعداد مشخص
به شرط اینکه اتفاقات نرخ میانگین مشخصی داشته باشند
مثلا در بازار بورس سهمی روزانه ۳۰۰۰۰ تا ازش خرید و فروش میشه
یعنی به صورت میانگین میدونیم حدود ۳۰۰۰۰ تاست، و ویک چیز مشخص است
مثلا برای بورس و پیشبینی بازار های مالی
برای اعداد گسسته به کار میرود، چون جرم احتمال است

- پرسنتایلPercentileصدک -

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
values = np.random.normal(0, 0.5, 10000)
plt.hist(values, 100)
plt.show()
```

Out>



نمودار توزیع نورمال#

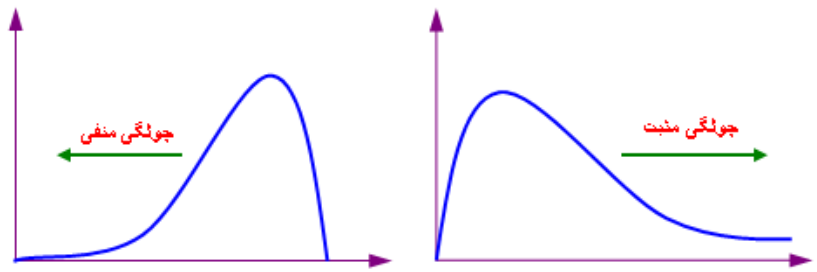
در نمودار میتوانیم ببینیم که حدود ۳۰۰ تا از داده های ما حدود ۰ است یعنی ۳۰۰ تا داده نزدیک ۰ است
ما در صدک چیزی برعکس این را محاسبه میکنیم
به اینگونه که ما درصد میدهیم و به ما میگوید که
مثلا ۵۰ درصد اعداد از ۰/۱۲ که خودش داده کمتر است

np.percentile(values, 30) #30%
-> -0.25... # سی درصد اعداد از ۰/۲۵ کمتر است
np.percentile(values, 50)
-> -0.001...
np.percentile(values, 90)
-> -0.64...
np.percentile(values, 99)
-> -1.15...
یعنی ۹۰ درصد اعدادی که در داده است از ۰/۶۴ صدم کمتر است

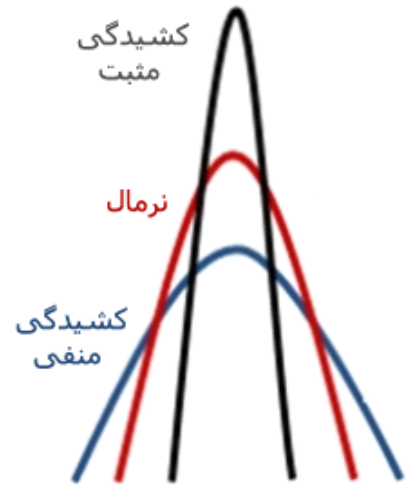
---

moments - گشتاور

چولگی
کشیدگی
نمودار توزیع نرمال متقارنه
ولی در دیتای واقعی اینجور نیست و کمی متمایل به چپ و راست است
این ها یک ضرایبی هستند که دوتای آنها
<- چولگیSkewness
<- کشیدگیKurtosis



نمودار چولگی مثبت و منفی



نمودار کشیدگی مثبت و منفی

در کشیدگی منفی پراکندگی داده بیشتری نسبت به نرمال داریم
وقتی هم که کشیدگی مثبت باشه یعنی به عدد میانگین نزدیک تر است و پراکندگی کمتر هست

گشتاور شامل ۴ بخش بسیار مهم هست
میانگین
واریانس
چولگی
کشیدگی

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as sp

values = np.random.normal(0, 0.5, 10000)
plt.hist(values, 50)
plt.show()
np.mean(values)  #میانگین
```
≃ -0.0018...

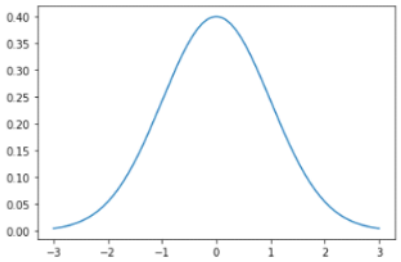
```
np.var(values)    #واریانس
```
≃ 0.248...

```
sp.skew(values)  #چولگی
```
≃ -0.018... #چولگی منفی یا مثبت

```
sp.kurtosis(values)  #کشیدگی
```
≃ -0.0219... # کشیدگی مثبت یا منفی

آشنایی کاملتر با matplotlib

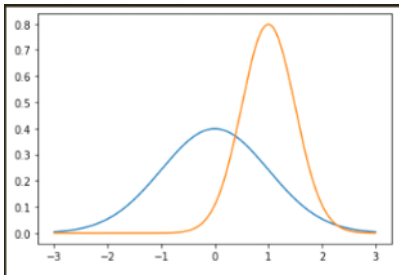
```
from scipy.stats import norm
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(-3, 3, 0.001)
plt.plot(x, norm.pdf(x))
plt.show()

Out>
```



برای مقایسه و کنار هم قرار گذاشتن دو نمودار ->

```
x = np.arange(-3, 3, 0.001)
plt.plot(x, norm.pdf(x))
plt.plot(x, norm.pdf(x,1.0,0.5))
plt.show()
```

Out>

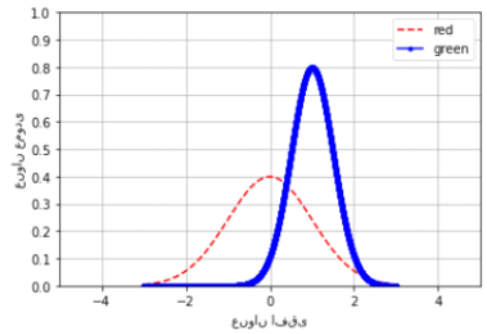


سیو کردن نمودار ->

```
plt.savefig('\\c:ML\\res.png', format='png') #or png
```

نشان دادن قسمت خاصی از نمودار

```
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt

x= np.arange(-3, 3, 0.001)
axes = plt.axes()
axes.set_xlim([-5, 5])
axes.set_ylim([0, 1 ])
axes.set_yticks([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
axes.grid()  # بندی جدول

color1= 'r--'  #چین خط قرمز   #:
color2= 'b.-'  #چین خط و چین نقطه آبی

plt.plot(x, norm.pdf(x), color1)
plt.plot(x, norm.pdf(x, 1.0, 0.5), color2)
plt.legend(['red', 'green'], loc= 1) # میکند تغییر موقعیتش لوک عدد تغییر با / است مهم ها رنگ تعریف ترتیب
#پلات اولین برای آیتم اولین

plt.xlabel('افقی عنوان')
plt.ylabel('عمودی عنوان')
plt.show()
```



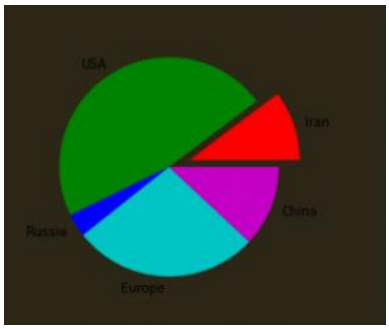
Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center	9
'center'	10

نمودار های دایره ای

نیاز ها برای ایجاد نمودار
values
colors
labels

```
values= [12, 55, 4, 32, 14] # ها قسمت مقدار
colors= ['r','g','b','c','m']  # قسمت هر رنگ
labels = ['Iran','USA','Russia','Europe','China']  #ها قسمت اسم
explode = [0.2, 0, 0, 0, 0] # زدگی بیرون
plt.pie(values, colors= colors, labels= labels, explode= explode)
plt.show()
```

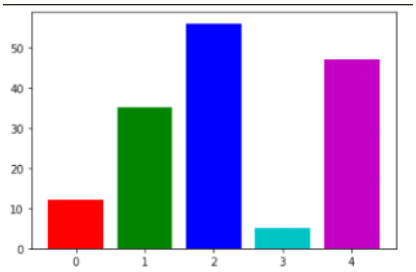
Out>



نمودار میله ای

```
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
values = [12, 35, 56, 5, 47]
colors = ['r', 'g', 'b', 'c', 'm']
plt.bar(range(0,5), values, color = colors) # range(0, تعداد آیتم ها)
plt.show()
```

Out>



نمودار نقطه ای

```
import numpy as np

x = np.random.randn(500)
y = np.random.randn(500)
plt.scatter(x, y)
plt.show()
```

اکثر اعداد در بازه ی ۰ است
میانگین <- 0
این در اصل همون توزیع نرمال است
چون این توزیع نرمال است ۶۸ درصد اعداد
در بازه ی -۱ و +۱ هست
تمام آرگومان های نمودار <- https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html

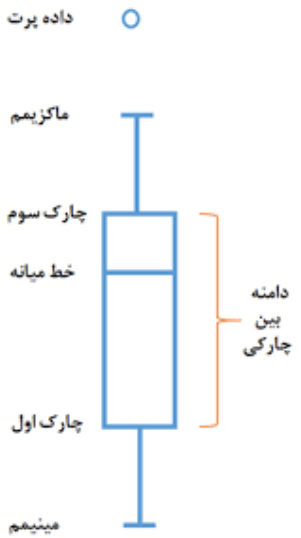
عوض کردن سایز مارکر یا همان نقطه ها در scatter

```
# doubling the width of markers
x = [0,2,4,6,8,10]
y = [0]*len(x)
s = [20*4**n for n in range(len(x))]
plt.scatter(x,y,s=s)
plt.show()
```

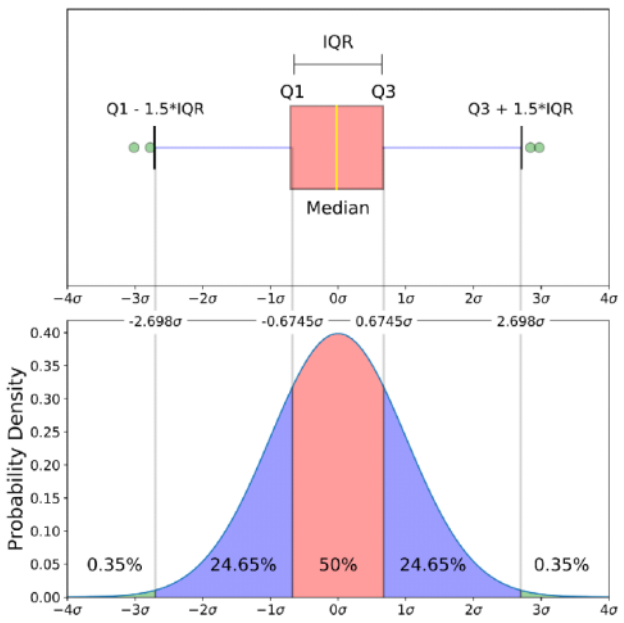
مباحث نمودار جعبه ای / چارک

در محاسبات آمار داده هایی داریم به نام داده ی پرت
این داده ها را باید از مجمع اصلی داده هایمان کم کنیم
مثلا توی یک شرکت همه حدود ۵ میلیون حقوق میگیرند
و یک نفر ۱۰۰ میلیون، این ۱۰۰ میلیون یک داده ی پرت
حساب میشود که اگر در محاسبات آماری این را در نظر بگیریم
محاسبات به مشکل برخورد میکند، پس در محاسبات نباید این
رو در نظر بگیریم

چارک اول ->
چارک ها داده هایی هستند که ۲۵ درصد
اعدادی که در نمودار داریم از اون کوچک تر هستند



چارک سوم ->
مانند جارک اول است ولی ۷۵ درصد اعداد ازش کوچک تر اند



چارک اول و سوم <- Q1, Q3
منطقه ای که بین چارک ها تشکیل میشود در اصل همون داده های اصلی مون
که در عکس صورتی است را :
IQR دامنه ی میان چارکی مینامیم ->
این دامنه شامل ۵۰ درصد داده ها میشود
میانگین +- انحراف معیار -> ۶۸ درصد اعداد
پس بازه ی میان چارکی از بازه ی میانگین مثبت منفی انحراف معیار میشود
کوچک تر است

Maximum -> Q3 + 1.5*IQR

minimum -> Q3 - 1.5*IQR

وقتی فاصله ی بازه میان چارکی تا حداکثر و حداقل
فاصله ی زیادی داشته باشد نشان دهنده ی چولگی است

کشیدن نمودار جعبه ای

```
import numpy as np
values = np.random.rand(100) *100 -40
# اعداد ما به 0 تا 1 وقتی و میدهد * 100 ما به میکنیم
اعداد 0 تا 100 ، منهای وقتی میدهد 40 ما به میکنیم -40 تا 60 میدهد عدد
high = np.random.rand(10) *50 +100   # ماکس یا بالا بازه
low = np.random.rand(10) *-50 -100   # مین یا پایین بازه
data = np.concatinate((values, high, low)) # میکند یکی را همه
plt.boxplot(data)
plt.show()
```

---

seabornکتابخانه

DATA =

datas

```
%matplotlib inline  #magic function
import pandas as pd
import seaborn as sb
data = pd.read_csv('cars.csv')
num_counter = data['# Gearse'].value_counts()   # شده تکرار که هایی آیتم تعداد
num_counter.plot(kind= 'bar')
sb.set()  # میشود اعمال پلات مت به بورن سی تنظیمات
num_counter.plot(kind= 'bar')
sb.distplot(data[''])  # خطی هم ای نرده هم
```

```
.value_counts()  Out>
```

6	224
7	110
9	57
10	28
1	17
5	16

CSV جدا کردن قسمت نیاز از فایل

```
...
data2 = data[['Cylinders','CityMPG','HwyMPG','CombMPG']]
```

```
data2.head()
```

Out>

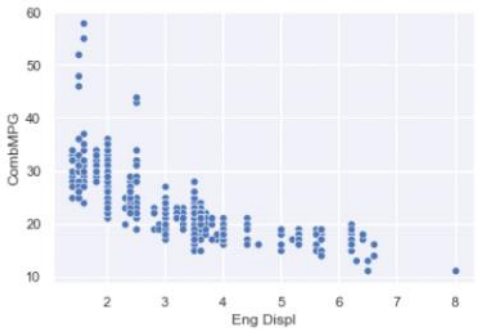
	Cylinders	CityMPG	HwyMPG	CombMPG
0	8	18	25	21
1	16	9	14	11
2	8	12	20	15
3	8	15	25	18
4	8	14	23	17

تاثیر داده ها روی یکدیگر و مقایسه با یکدیگر

```
sb.pairplot(data=data2, hue='Cylinders', height=2.5)
```

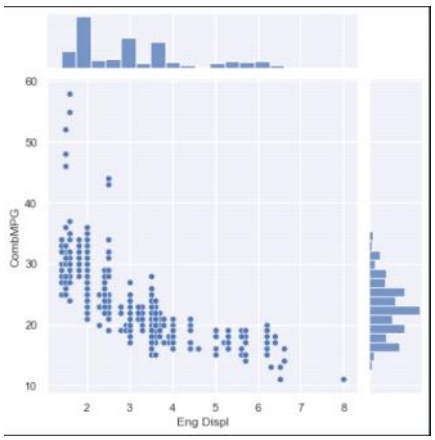
به ما یک جدول ۳ در ۳ میدهد که این ۳ تا در اصل
'cityMPG','HwyMPG','CombMPG'
هستند که داده های جدول تغییرات سیلندر هست نسبت به
افقی و عمودی که در اصل همان سه عامل دیگر هستند
اینا رو دونه دونه میاد و نشون میده و میشه باهم مقایسه کرد

```
sb.scatterplot(x= 'Eng Displ', y= 'CombMPG', data= data)
```



این برای **مقایسه دو ستون** به کار میره
اینجا مثلا ما
CombMPG را با Eng Displ مقایسه کردیم

```
sb.jointplot(x= 'Eng Displ', y= 'CombMPG', data= data)
```

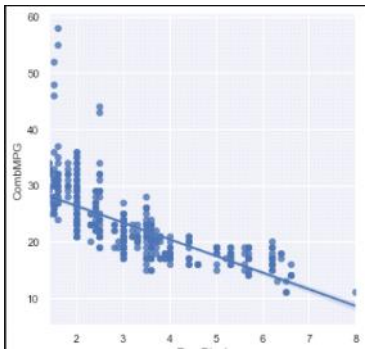


مثل قبلی است ولی نمودار میله ای هم یکجا داره که کار ما راحت شود

نمودار رگرسیون خطی

رگرسیون
فراوانی تجمع داده ها حول یک خط در محور افقی عمودی است
رگرسیون خطی یم معادله خطی است، اگر ما به x , yعدد بدیم
میتونیم نتیجه ی آینده را پیش بینی
توضیحات کامل در ادامه
Article Link

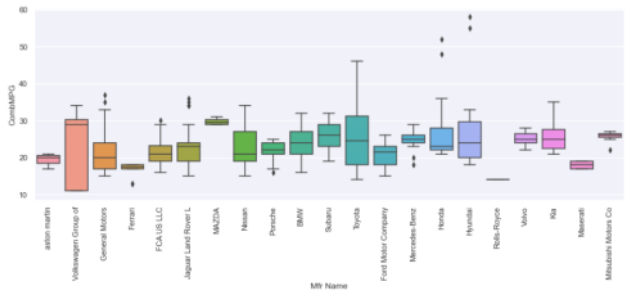
```
data = pd.read_csv('cars.csv')
sb.lmplot(x='Eng Displ', y='CombMPG', data=data)
```



نمودار جعبه ای تک تک داده ها برای مقایسه

```
data = pd.read_csv('cars.csv')
sb.set(rc= {'figure.figsize': (15,5)})
ax = sb.boxplot(x='Mfr Name', y='CombMPG', data=data)
ax.set_xticklabels(ax.get_xticklabels(), rotation= 90)
```

با این کار نمودار جعبه ای تک تک شرکت هارو با اسم بدست میاریم
https://b2n.ir/565791 #d_ai_نمودار جعبه ای



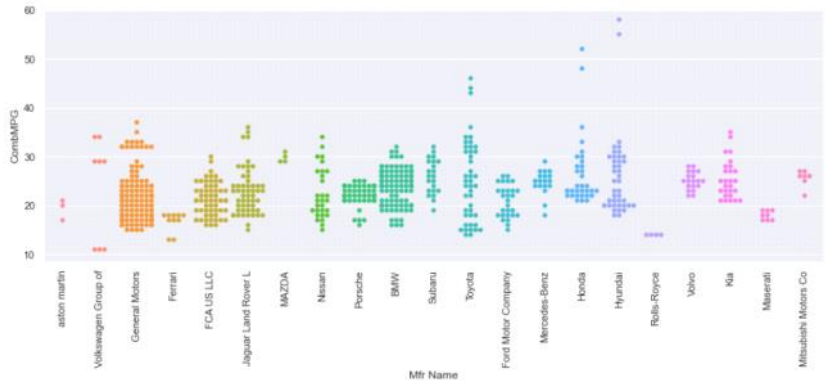
Description

زاویه قرار گیری اسم ها در محور افقی <- rotation
اگر ۹۰ باشد به صورت عمودی است اما اگر ۴۵ باشد مانند
عکس بالا، اسم ها مورب چیده میشود

نشون دادن جعبه ای به شکل دیگیر

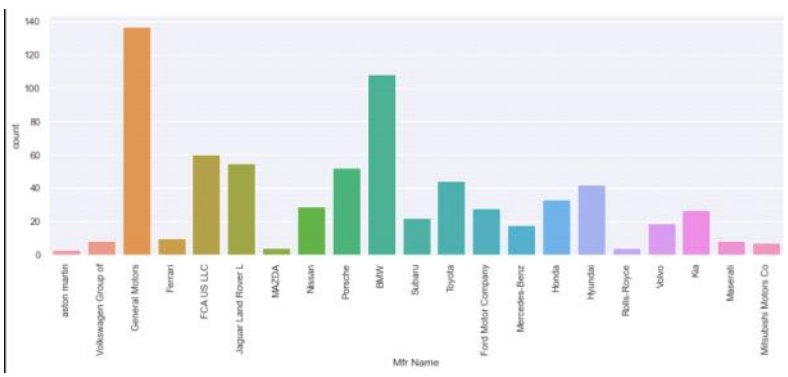
```
sb.set(rc= {'figure.figsize': (15,5)})
ax = sb.swarmplot(x='Mfr Name', y='CombMPG', data=data)
ax.set_xticklabels(ax.get_xticklabels(), rotation= 90)
```

این همون نمودار جعبه ای است که فراوانی تجمعی را نیز نشان میدهد



نمودار میله ای تکرار

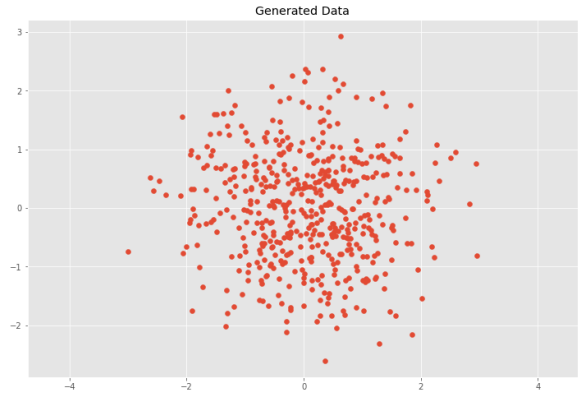
```
sb.set(rc= {'figure.figsize': (15,5)})
ax = sb.countplot(x='Mfr Name',data=data)
ax.set_xticklabels(ax.get_xticklabels(), rotation= 90)
```



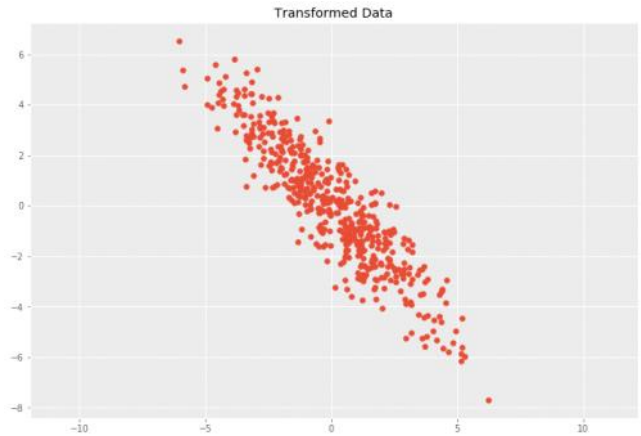
برای اینکه ببینیم در دیتایی که داریم از داده چند بار تکرار شده

Covariance / کوواریانس

فرض کنید یک دیتایی داریم
داده میزان درآمد و ضریب میزان سن است
یعنی میخوایم ببینیم هرچقدر سن این افراد بیشتر میشه
آیا به درآمدشون هم اضافه میشه یا نه
یعنی میزان درآمد به میزان سن آیا وابستگی یا همبستگی وجود داره یا خیر

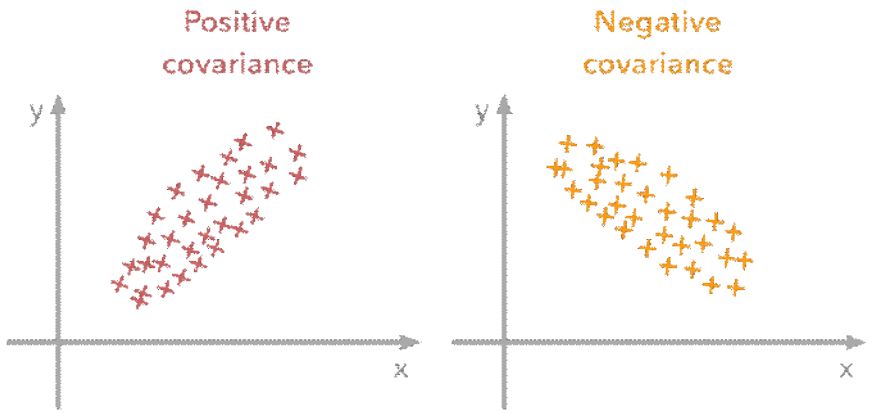


مثلا فرض میکنیم نمودار بالا داده های ماست،
میبینیم که هیچ ارتباتی بین داده ها وجود ندارد
کوواریانس میاد چک میکنه آیا بین داده هایی که دادیم
همبستگی یا پیوستگی وجود دارد یا خیر



اگر مانند عکس دوم ارتباطی وجود داشته باشد
و به صورت خطی بود همبستگی وجود دارد

همبستگی مثبت و منفی



محاسبه این را پایتون برایمان خودش انجام میدهد
ولی حالا چگونه این بدست می آید
بالاخره داده های ما یک میانگین دارند
و میانگین درآمدی و میانگنی سنی را میتوانیم
اینجا بدست بیاوریم، اگر اینجا میزان واریانس
تک نقاط (داده ها) محاسبه کنیم
که یک سری تحلیل ها داره ما به مقدار کوواریانس میرسیم
البته این عملیات برای هر اتریبیوت انجام میشه
نحوه محاسبه کوواریانس به مانند محاسبه واریانس است با این تفاوت که به جای
مجموع مربعات اختلاف از میانگین در واریانس، از
مجموع حاصلضرب‌های اختلاف از میانگین هر دو متغیر در کوواریانس استفاده می‌شود.
یعنی این پروسه محاسبه برای تک تک داده ها انجام میشه
این کوواریانس که ما به دست میاریم یک عددی هست که
یا صفر یا بزرگ تر از صفر یا کوچیک تر از صفر
وقتی صفر باشه یعنی هیچ گونه وابستگی بین داده ها وجود نداره
یعنی هرچقدر به صفر نزدیک باشه وابستگی کمتر
و هر چقدر از صفر فاصله داشته باشه وابستگی بیشتری دارد

$$Cov_{xy} = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{(n-1)} = \frac{\Sigma xy - n\bar{x}\bar{y}}{(n-1)}$$

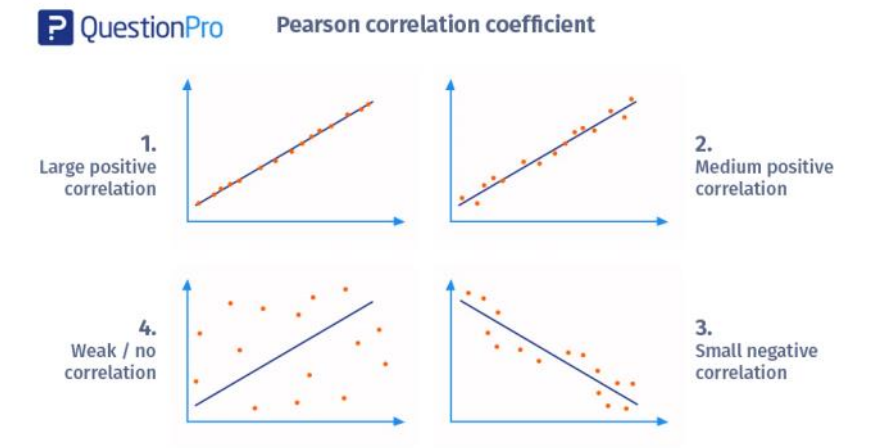
دو مجموعه ایکس و ایگرگ با تعداد یکسان

به عنوان مثال دو مجموعه داریم
x = [1,2,3]
y = [4,5,6]
1) میانگین هر دو را حساب میکنیم
mu_x = 2
mu_y = 5
2) تک تک آیتم هارو منهای میانگین میکنیم
x' = [-1, 0, 1]
y' = [-1, 0, 1]
3) حالا دو مجموعه را ضرب میکنیم
mult = [-1,0,1] × [-1,0,1] = 1 + 0 + 1 = 2

4) تقسیم بر تعداد آیتم ها منهای یک میکنیم
res = 2 ÷ (n-1) = 2 ÷ (3-1) = 2 ÷ 2 = 1 #n = تعداد آیتم ها

۱ دلیل -nاین است که ما از
Sample Variance
استفاده کردیم

Correlation / ضریب هم بستگی



در عکس بالا میبینیم که هرچه پراکندگی بیشتر باشه
هم بستگی کمتر میشه
نتیجه ی این عددی هست بین -۱ تا ۱
وقتی که ۱ باشد یعنی ما یک همبستگی کاملااثبات شده را داریم

Article Link

1. ضریب همبستگی و کاربرد آن در علم داده
   1.1.) کوواریانسCovariance(
      1.1.1. محاسبه کوواریانس
      1.1.2. محدودیت‌های کوواریانس
   1.2.) ضریب همبستگیCorrelation(
      1.2.1. محاسبه ضریب همبستگی
      1.2.2. ویژگی‌های ضریب همبستگی

کد کوواریانس در پایتون

```
import numpy as np
x = [1, 2, 3]
y = [4, 5, 6]
def de_mean(values):
    values_mean = np.mean(values)
    return [xi - values_mean for xi in values]

print(de_mean(x))
print(de_mean(y))

out>
     [-1.0, 0.0, 1.0]
     [-1.0, 0.0, 1.0]
```

حالا باید این دو مجموعه در هم ضرب کنیم
ضرب دو مجموعه را با کمک نام پای میتوانیم انجام دهیم
با کمک، فانکشنی به نام dot که دو لیست یا ارایه میگیرد و در هم ضرب میکند

```
result = np.dot(de_mean(x), de_mean(y))

def covariance(a, b):
    n = len(a)
    return np.dot(de_mean(x), de_mean(y)) / (n-1)
```

```
print(covariance(x, y))
```

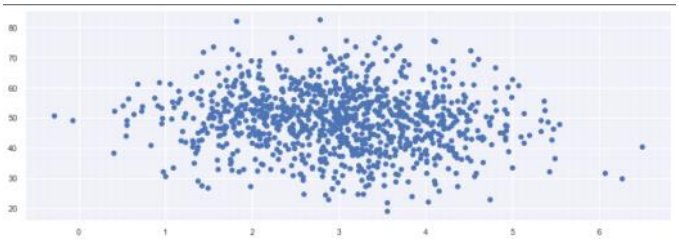
در توضیح کوواریانس در بالاتر توضیح این عملیات ها و دلیل جواب ها داده شده

مثال واقعی تر از کوواریانس

```
import numpy as np
import matplotlib.pyplot as plt
def covariance(a, b):
    n = len(a)
    return np.dot(de_mean(a), de_mean(b)) / (n-1)
page_speed = np.random.normal(3.0, 1.0, 1000)
page_requests = np.random.normal(50.0, 10.0, 1000)
page_visit = np.random.normal(50.0, 10.0, 1000)/ page_speed

print(covariance(page_speed, page_requests))
plt.scatter(page_speed, page_requests)
```

out>

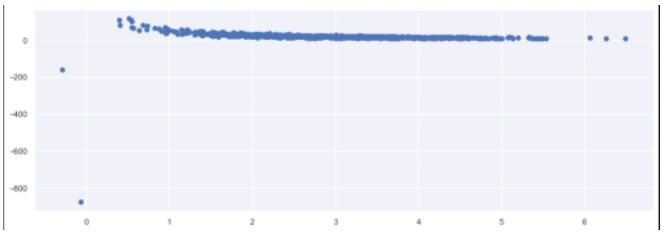
-0.9064826100979432



```
plt.scatter(page_speed, page_visit)
print(covariance(page_speed, page_visit))
```

out>

-4.597823447165747



مثلا حدود عدد منفی ۵ را میدهد
این عدد منفی است یعنی ارتباط زیادی بین داده ها وجود دارد
نمیتوانیم بگوییم این منفی ۵ چه چیز یرا نشان میدهد
این جا باید از ضریب هم بستگی یا کورلیشن

محاسبه ضریب همبستگی

```
def corralation(x, y):
    std_x = x.std()
    std_y = y.std()
    return covariance(x,y) / std_x / std_y

corralation(page_speed, page_requests)
```

در ضمن فانکشن de_mean و covariance رو از قبل تعریف کرده بودیم

عددی بین ۰ تا ۱ میدهد که این عدد هرچی به ۱ نزدیک تر باشد
ارتباط بین این دو لیست به هم مرتبط هستند از نوع مثبت
یعنی افزایش اولی روی افزایش دومی تاثیر مستقیم داره
اما خود نام پای این را محاسبه میکند و نیازی نیست ما محاسبه کنیم

np.corrcoef(page_speed, page_requests)

point

دلیل اون فرمول هایی که در محاسبه در فانکشن استفاده کردیم

آکادمی تحلیل آماری ایران

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\,n\Sigma x^2 - (\Sigma x)^2\,]\,[\,n\Sigma y^2 - (\Sigma y)^2\,]}}$$

r = ضریب هم بستگی
صورت کسر همان کوواریانس هست
حالا مخرج در اصل همان فرمول std
std(x) , std(y)
σ^2 -> واریانس
$\sqrt{\sigma^2} = \sigma \simeq 2.24$ -> <- انحراف معیار std

---

احتمال شرطی

احتمال معمولی
مثلا یک کیسه داریم ۵ مهره آبی قرمز صورتی سبز زرد داخلش داریم
احتمال اینکه اولین مهره ای درمیاریم قرمز باشد چند است

$$p(s) = \frac{n(A)\ \text{-> پیشامد}}{n(S)\ \text{-> فضای نمونه}} \rightarrow \text{احتمال}$$

$$\frac{1}{5} = 20\%$$

حالا چیزی داریم به نام احتمال شرطی
یک کیسه داریم ۵ مهره
قرمز - قرمز - آبی- سبز- زرد
دو مهره برمیداریم، احتمال اینکه مهره دوم قرمز باشد
فرض میکنیم مهره اول آبی را برداشته ایم
B = 2/4
R = 1/4
Y = 2/4
G = 2/4
این بستگی دارد و اینجای چیزی به نام احتمال شرطی وسط می آید

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

کد نویسی احتمال شرطی

```
import numpy as np
np.random.seed(0)   #ندهد تصادفی اعداد
#دهیم انجام مطالعه فروشگاهی سایت یک های داده روی میخواهیم ما مثلا
totals= {20:0, 30:0, 40:0, 50:0, 60:0, 70:0}  #سنی بازه در افراد تعداد
purchases = {20:0, 30:0, 40:0, 50:0, 60:0, 70:0} #سنی بازه در افراد خرید میزان
totalpurchases = 0
for person in range(100000):
    age = np.random.choice([20, 30, 40, 50, 60, 70])
    purchaseProbability= float(age)/ 100  #سن به نسبت خرید احتمال
    totals[age] += 1

    if np.random.random() < purchaseProbability:
        totalpurchases += 1
        purchases[age] += 1
print(totals)
#-> {20: 16576, 30: 16619, 40: 16632, 50: 16805, 60: 16664, 70: 16704}

print(purchases)
 # هر چه سن بیشتر میشه میزان خرید هم بیشتر میشه
#-> {20: 3392, 30: 4974, 40: 6670, 50: 8319, 60: 9944, 70: 11713}

print(totalpurchases)
#-> 45012
```

حالا میخواهیم توی بازه سنی خاصی مطالعات انجام دهیم

```
PEF = float(purchases[30]/ float(totals[30])) #P,Probability   E,purchases   F,people
print(f"%{str(PEF)[2:4]}")
```

#4974 ÷ 16619 = 0.29... احتمال افرادی که در بازه سن ۳۰ سال خرید کرده اند به کل افرادی که در بازه سی سال اند

#p(E | F)

out> %29

حالا میخواهیم ببینیم چند درصد از کل در بازه سی سال اند

```
PF = float(totals[30]/ 100000.0)
print(f"%{str(PF)[2:4]}")   #هستند 30 سنی بازه توی نفره هزار صد بازه داخل که افرادی کل از درصد
```

out> %16

```
PE = float(totalpurchases/ 100000.0)
print(f"%{str(PE)
[2:4]}")   #اند داده انجام خرید نفر هزار صد از درصد 45 مثلا یعنی ، کل به کنندگان خرید درصد
```

out> %45

ما نتیجه میگیریم که بین سن خرید رابطه وجود دارد میبینیم

PEFتقریبا شد ۲۹ درصد

PEF -> خرید به شرط سن

اگر سن و خرید با هم رابطه نداشتن اون موقع باید

PE = PEF

اما الان میبینیم که

PEF -> %29

PE -> %45

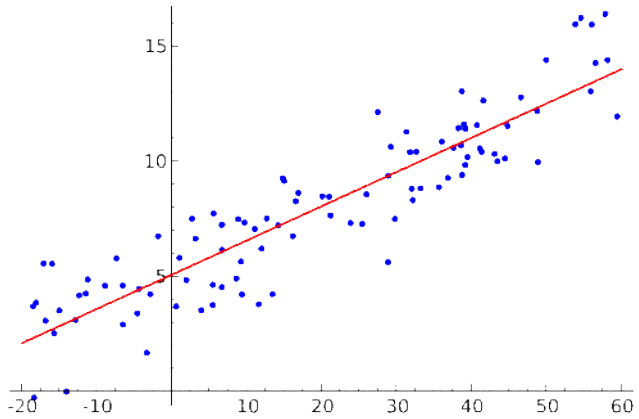
و این رو خودمان اینجاد کردیم که با زیاد شدن سن خرید هم زیاد شه

<کد منبع

conditional-
probability

رگرسیون خطی
Linear Regression

داده های مربوط حول یک خط بالا میروند
آن خط رگرسیون خطی است
رگرسیون خطی عملا کارکردش به این صورت است که
یه مجموعه دیتا میدن و ما میایم معادله خط رو بدست میاریم
وقتی یک داده رو مثلا x اش رو میدیم yرو به ما میدهد
و برای پیشبینی کاربرد دارد
مثلا با زیاد شدن قد میدانیم که وزن زیاد میشود



f(x) -> m*x + b

حالا یکجا ما وزن رو داریم میخواهیم پیشبینی کنیم قد چقدر است
حالا از میخواهیم بدانیم چقدر این عدد گفته شده خطا دارد
برای محاسبه خطای پیشبینی و اعتبار سنجی ما یک مبحثی تحتRsquareداریم
که در جلوتر بهش میرسیم

/ رگرسیون چند جمله ایpolynomial Regression

به عنوان مثال
f(x) -> m*x + b
توان xیک هست و به خاطره همین معادله خطی که میکشیم یک خط راست است
اما همیشه به این شکل نیست و بعضی از مواقع متغیر توان دارد
f(x) -> $a*x^2 + b*x + c$ -> معادله درجه ۲
معادله ای که میتوانیم پیاده سازی کنیم میتواند بیشتر باشد
معمولا رگرسیون کامل صاف نیست و اکثرا حالت های پیچیده دارد

/ رگرسیون چند متغیرهMultiple Regression

مثلا برای خرید یک ماشین ما فقط یک آیتم برای قیمت نداریم

مثلا تعداد در، سرعت، صفر تا صد، رنگ، مصرف و... میتواند روی خرید ما و البته قیمت تاثیر بذارد
یا مثلا ما میخواهیم چندین مورد رو پیشبینی کنیم

Price = a + x*speed + y*age + ...
a → ضریب ثابت یا قیمت پایه

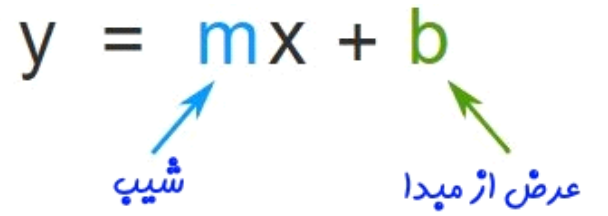
اگر بتوانیم ضرایب x, y, z رو پیشبینی کنیم عملا میتوانیم به این نتیجه برسیم
که کدام یک از پارامتر ها توی پیشبینی ما تاثیر بیشتری دارد
هر چه سرعت بیشتر باشد تاثیر بیشتری دارد
هرچه سن کمتر باشد تاثیر بیشتری دارد

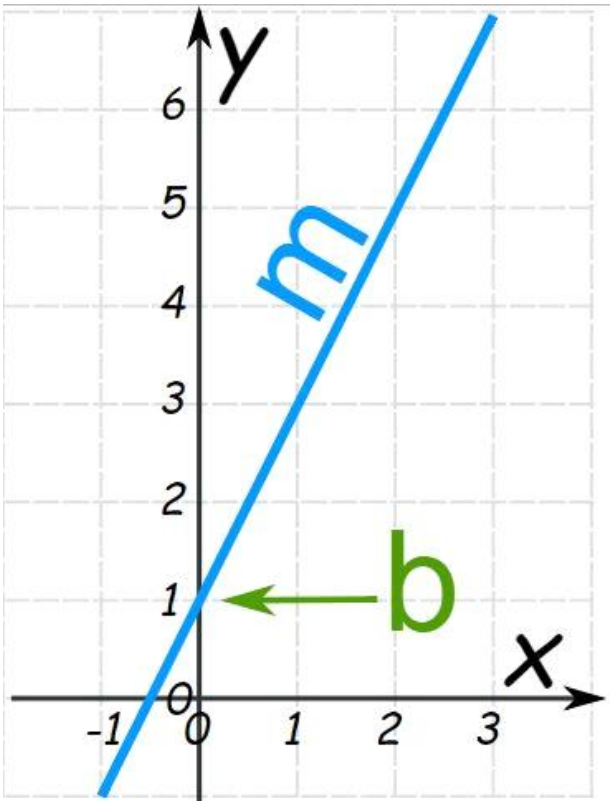
خطا یا همان درصد اتکا یا Rsquare هم اینجا هست

درضمن ضریب ها نباید با هم رابطه یا دیپندنسی Dependency داشته باشند
و باید هر کدام مستقل از بقیه باشند مثلا سن با تعداد در ها هیچ رابطه ای ندارد
ممکنه بعضی از این متغیر ها با هم تعامل و وابستگی داشته باشند اما سیستم ؟؟

معادله خط

به روابط توصیف کننده ی خط راست گفته میشود
y = mx + b
m -> شیب
b -> عرض از مبدا
y = 2 * x + 1
x<-نسبت به y
x : 1 -> y : 2
x : 2 -> y : 4
x : 3 -> y : 6
از ۳ تا ۱ دو تا فاصله است و از ۶ تا ۲ هم چهار فاصله است
میبینیم که هرچه xرو بیشتر کنیم
yدوبرابر اون بیشتر میشه
پس ضریب پشت xدر اصل شیب خط است
و اون جمع هم عرض از مبدا است
x : -1 -> y : -1
x : 0 -> y : 1
x : 1 -> y : 3
x : 2 -> y : 5
معادلات خط اصلا توان ۲ و ۳ و رادیکال و... ندارد
<- اشکال درست معادله خط
y = 3x -6
f(x) = 3x -6
w(u) = 3x -6
h(z) = 3x -6
y -2 = 3(x + 1)
y +2x -2 = 0
5x = 6
y ÷ 2 = 3
<- اشکال غلط معادله خط
y^2 -2 = 0 غلط
$3\sqrt{x}$ - y = 6 غلط
x^3 ÷ 2 = 16 غلط





m <- شیب
??

b <- عرض از مبدا
نقطه اتصال یا تقاطع به محور y
اگر مثلا ۳ باشد در محور عمودی برای شروع سه تا بالاتر میرویم
یعنی مثلا از ۳ به جای ۰ شروع میشود

<- تابع همانی
f(x) = x
y = x
ویژگی مهم تابع همانی این است که ورودی و خروجی یکسان اند
است 45°و شیب همان
یعنی در اصل ضریب xیک است

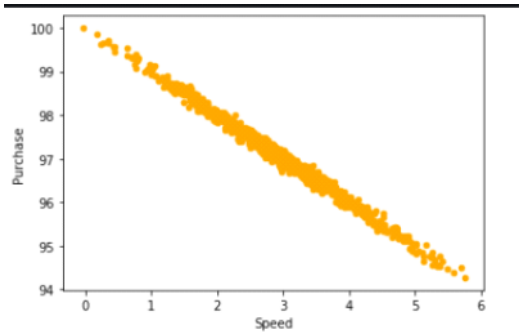
constant function <- تابع ثابت
f(x) = c
یک خط کاملا افقی رو محور ایجاد میکنند
۳ مثلا = f(x)هر ورودی رو بگیره خروجی برابر با ۳ است
https://b2n.ir/f00505

تابع های پیچیده ای هم هستند که درجه ی متغیر بالا تر است مثل
$f(x) = ax^2 + bx + c$

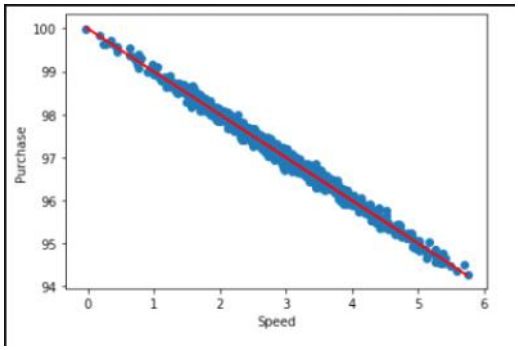
رگرسیون خطی

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

np.random.seed(0)
pageSpeeds = np.random.normal(3, 1, 1000)
purchaseAmount = 100 - (pageSpeeds + np.random.normal(0, 0.1, 1000))
plt.scatter(pageSpeeds , purchaseAmount, color= "orange", s = 20)
plt.ylabel('Purchase')
plt.xlabel('Speed')
plt.show()
slope, intercept, r_value, p_value, std_err = stats.linregress(pageSpeeds, purchaseAmount)
#slope -> خط شیب
#intercept -> مبدا از عرض
#r_value -> آید می بدست این از خطا ضریب
#p_value ->
#std_err ->
# f(x) = m * x + b
# f(x) = sloop * x + intercept
print(slope, intercept, r_value, p_value)

def predict(x):
    return slope * x + intercept
#نمودار به توجه با تست
print("predict -1 :" ,predict(-1)) #-> correct
#پیشبینی
print("predict 10 :" ,predict(10))

#رگرسیون نمایش
fitline = predict(pageSpeeds)
plt.scatter(pageSpeeds, purchaseAmount)
plt.plot(pageSpeeds, fitline, c= 'r')
plt.ylabel('Purchase')
plt.xlabel('Speed')
plt.show()
```

out>




```
-0.9969094198397863 99.98950643497004 -0.9951991104092179 0.0
predict -1 : 100.98641585480982
predict 10 : 90.02041223657217
```

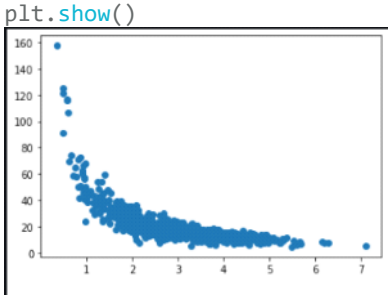


ضریب خطا

r_value ** 2
بین ۰ تا ۱ که درست ترین حالت ۱ است
یعنی اگر ۰/۹۸ باشد، ۹۸ درصد میتوان به جواب اعتماد کرد

کد نویسی رگرسیون چند جمله ای / Polynomial Regression

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score

np.random.seed(2)
pageSpeeds = np.random.normal(3, 1, 1000)
purchaseAmount = np.random.normal(50, 10, 1000)/ pageSpeeds
plt.scatter(pageSpeeds, purchaseAmount)
plt.show()
```

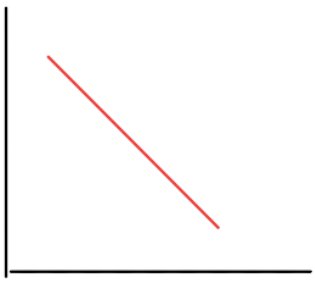


```
#خط معادله آوردن بدست
#بگیم رو متغیر درجه باید
#ای جمله چند رگرسیون
x = np.array(pageSpeeds)
y = np.array(purchaseAmount)
linear_equation = np.poly1d(np.polyfit(x, y, 4))
# چهار درجه معادله x میگیرد ورودی فانکشن یک مانند که ها داده به باتوجه
# میشود خطی رگرسیون اصل در بدیم یک بهش اگر

#خط معادله نمایش
xp = np.linspace(0, 7 , 100) # بین ۰ و ۷ صد عدد میسازد چون اعداد ما در بازه ۰ و ۷ هست
#0.7, 5.5 , 100
plt.scatter(x, y)
plt.plot(xp, linear_equation(xp), c = 'red')
plt.show()
#دقیقه پلات توان رابطه توضیحات 4:10

#خطا درصد و پیشبینی
print(linear_equation(1)) #-> میگوید خط معادله به توجه با را مقدار
r2 = r2_score(y, linear_equation(x))
print(r2) #-> میاید افزایش یا کاهش دقتش و میکند تغییر معادله درجه به توجه با
```

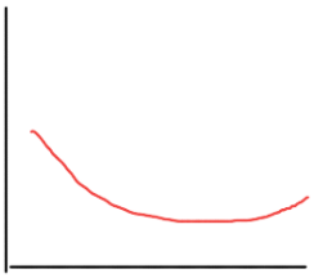
```
#خطا میزان مقابل در معادله درجه ->
#linear_equation : 1 -> R Square : 0.50
#linear_equation : 2 -> R Square : 0.69
#linear_equation : 3 -> R Square : 0.78
#linear_equation : 4 -> R Square : 0.82
#linear_equation : 5 -> R Square : 0.85
#linear_equation : 6 -> R Square : 0.87
#linear_equation : 7 -> R Square : 0.87
#linear_equation : 8 -> R Square : 0.88
#linear_equation : 9 -> R Square : 0.88
#linear_equation : 15 -> R Square : 0.88
#linear_equation : 25 -> R Square : 0.88
#linear_equation : 100 -> R Square : 0.88
#linear_equation : 1000 -> R Square : 0.88
#میدهد ارور کمی بعد به 15 حدود از البته
```

description

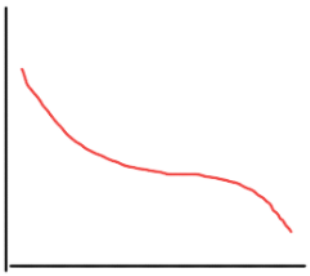
با توجه به درجه معادله شکل کلی معادله ی خط



شکل بالا مربوط به معادله درجه یک است



بالایی هم درجه ۲ است که مانند حرفUاست

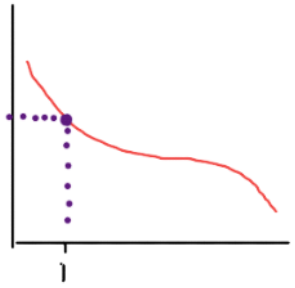


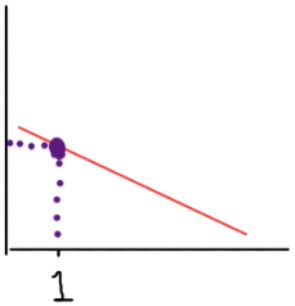
این هم درجه ۳ است که شبیهSاست که ۹۰ درجه چرخیده

یا مثلا درجه ۴ مانند یک Wاست که دسته سمت چپش کمی بلند تر است
و به همین ترتیب یک حالت سینوسی دارد و یکی در میان سرش به سمت بالا پایین میرود
در زوج ها صعودی و در فرد ها نزولی

_____

print(linear_equation(1))





یعنی مقدار عدد با توجه به معادله خط داده میشود
و مقادیر با توجه به معادله خط متغیر میشوند

دادن داده های ماشین و پیشبینی با کمک رگرسیون چند متغیر Multiple Regression

ما داده های یک ماشین رو بهش میدیم و میخواهیم ضریب مایل طی شده، سیلندر، تعداد در رو بدست بیاریم تا رابطه اش با قیمت رو بدست بیاریم

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler

result = pd.read_excel('cars_data.xls')
df_1 = result[['Mileage','Price']]
bins = np.arange(0,50000,10000)
groups = df_1.groupby(pd.cut(df_1['Mileage'],bins)).mean()
print(groups.head())
groups['Price'].plot.line()

scale = StandardScaler()
X = result[['Mileage','Cylinder','Doors']]
y = result[['Price']]
X[['Mileage','Cylinder','Doors']] = scale.fit_transform(X[['Mileage','Cylinder','Doors']])
print(X)

est = sm.OLS(y, X).fit()
est.summary()

y.groupby(result.Doors).mean()

scaled = scale.transform([[45000, 8, 4]])
print(scaled)
predict = est.predict(scaled[0])
```

```
print(predict)
```

OutPut

part of output

	coef	std err
Mileage	-1272.3412	804.623
Cylinder	5587.4472	804.509
Doors	-1404.5513	804.275

میبینیم که در خروجی ضریب هارو تحت coef میبینیم
میبینیم که ضریب مایل منفی است یعنی هرچه مایل بیشتر باشد قیمت کمتر میشود
ضریب سیلندر بین این سه تا بیشتری هست و ضریب مثبت دارد

Doors	Price
2	23807.135520
4	20580.670749

تعداد المان هایی که اینجا داریم خیلی کمه، ما نمیتونیم به همین دو دیتا اکتفا کنیم؟؟
متوسط برای خوردرو های ۲ در باید ۲۳۸۰۷ پول داد و برای ۴ در باید ۲۰۵۸۰ پول داد

[[3.07256589 1.96971667 0.55627894]]
[6315.01330583]
اولی اسکیل شده ی داده هایمان است
دومی قیمتی هست با توجه به داده ها

description

scale.fit_transform :

میبینیم که خروجی این بخش اعدادی بین -۱ و ۱ هستند
دقیق تر میانگین ۰ و انحراف معیار ۱ پس ۶۸ درصد از داده ها
دقیقا بین -۱ و ۱ هستند و بقیه داده ها کمی بالا پایین تر هستند
در اصل این تابع توزیع نرمال هست ما این کار رو روی داده ها کردیم
چون مطالعه روی تابع توزیع نرمال خیلی ساده تر از زمانی هست که بیایم روی دیتا های واقعی و پراکنده انجام بدیم

StandardScaler :

یکی از شش پیش پردازش یا preprocess در sklearn است
مقاله مدیم
استاندارد اسکیلر میاد و همه ی آیتم هارو از میانگین کم میکنه و اسکیل میکنه نسبت به یونیت واریانس
در اصل میاد همه ی آیتم هارو (بعد از کم کردن از میانگین) تقسیم بر std یا انحراف معیار میکنه
که در مقاله هم این رو توضیح کامل داده

Files

cars_data

cars_data

cars_data.xls

point

باید توجه داشت که برای گرفتن داده از اکسل باید نرم افزار اکسل نصب شده باشد
اگر در سیستمی اکسل نصب نشده باشد باید کتابخونه پایتونی نصب کرد که این مشکل رو هندل کند و بتوان فایل اکسل رو در کد دید

```
>>> pip install xlrd
```

Machine learning برسی مفاهیم

مجموعه ای از الگوریتم ها که یک سری داده رو میگیره و بر اساس اون داده تحلیل و پیشبینی میکنه

muchin learning :

supervised learning :

ما مجموعه ای از دیتا جواب رو داریم

Unsuppervised learning:

مجموعه ای از دیتا رو داریم و سیستم خودش طبقه بندی میکنه

train test :

مجموعه ای از داده ها که مدل بر اساس این ایجاد میشود
مثلا ۱۰۰۰۰ تا داریم، میگوییم ۸۰۰۰ تا برای ترین کردن و ۲۰۰۰ تا برای تست کردن مدلی که با ۸۰۰۰ تا ساخته شده
tain : 8000
test : 2000

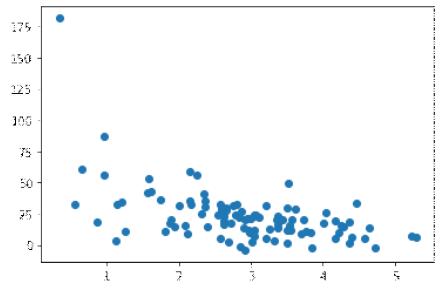
میزان اتکا و درصد خطا : ??english

با توجه به ترین تست ما باید میزان اتکا به مدل یعنی همون درصد خطا رو داشته باشیم که راه های مختفی دارد مثل
Rsquare
root mean
squared error
که درصد خطا با توجه به مدل و از طریق داده های تست به دست میاد

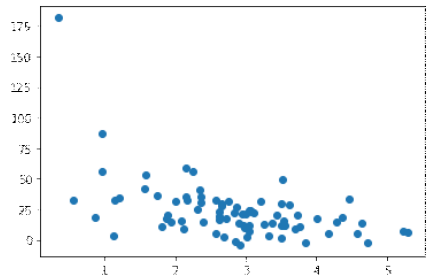
---

supervised پیاده سازی

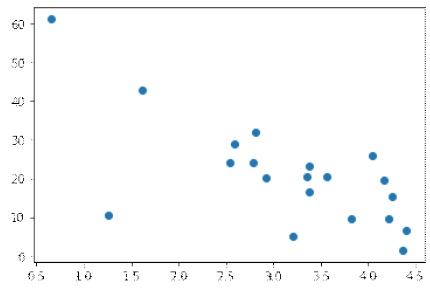
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score


np.random.seed(2)


pageSpeed = np.random.normal(3.0, 1.0, 100)
purchaseAmount = np.random.normal(50.0, 30.0, 100) / pageSpeed


plt.scatter(pageSpeed, purchaseAmount)
```



```
train_X = pageSpeed[:80]    #80% 100 = 80
test_X = pageSpeed[80:]
train_y = purchaseAmount[:80]
test_y = purchaseAmount[80:]


plt.scatter(train_X, train_y)
```



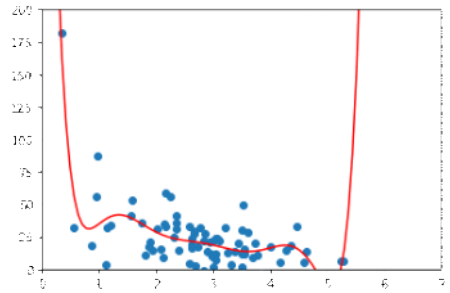
```
plt.scatter(test_X, test_y)
```



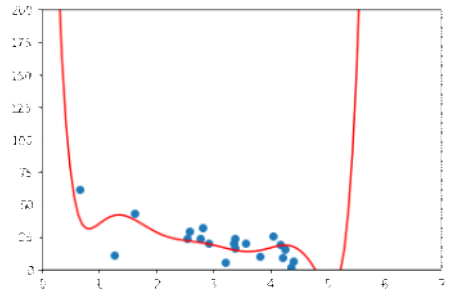
```
train_X_arr = np.array(train_X)
train_y_arr = np.array(train_y)
p4 = np.poly1d(np.polyfit(train_X_arr, train_y_arr, 8))
print(p4)
```

```
test_X_arr = np.array(test_X)
test_y_arr = np.array(test_y)
```

```
xp = np.linspace(0, 7, 100)
axes = plt.axes()
axes.set_xlim([0, 7])
axes.set_ylim([0, 200])
plt.scatter(train_X_arr, train_y_arr)
plt.plot(xp, p4(xp), c='r')
plt.show()
```



```
xp = np.linspace(0, 7, 100)
axes = plt.axes()
axes.set_xlim([0, 7])
axes.set_ylim([0, 200])
plt.scatter(test_X_arr, test_y_arr)
plt.plot(xp, p4(xp), c='r')
plt.show()
```



```
r2 = r2_score(test_y_arr, p4(test_X))
print('Rsquare test : ',str(r2)[2:4] + '.' + str(r2)[4:6] + '%')
# Rsquare test :  30.01%
```

```
r2 = r2_score(train_y_arr, p4(train_X))
print('Rsquare train : ',str(r2)[2:4] + '.' + str(r2)[4:6] + '%')
# Rsquare test :  64.27%
```

point

ما باید داده هایمان را در انتخاب ترین و تست رندوم انتخواب کنیم، اما چون وقتی که داده های فیکمون رو با نام پای میساختیم به صورت رندوم ساخته شده دیگر لازم نیست به صورت رندم انتخاب کنیم

description

ما اگر بخواهیم یک داده جدید پیشبینی کنیم، اگر داده ی ما در بازه ی اعداد ترینمون نباشه که با توجه به رگرسیون هم معلوم است مدل ما اصلا کارامد نیست، میبینیم که بعد از حدود داده های ترین رگرسیون به بیراهه به سمت بالا حرکت کرده

Source Code Link

پیاده سازی مدل تشخیص اسپم ایمیل

```
import numpy as np
import os
import io
from pandas import DataFrame
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB


def readFiles(path):
  for root, dirNames, fileNames in os.walk(path):
    for name in fileNames:
      path = os.path.join(root, name)

      in_body = False
      lines = []
      f = io.open(path, 'r', encoding='latin1')
      for line in f:
        if in_body:
          lines.append(line)
        elif line == '\n':
          in_body = True
      f.close()
      message = '\n'.join(lines)
      yield path, message


def DataFrameFromDir(path, classification):
  rows = []
  index = []

  for file_name, message in readFiles(path):
    rows.append({'message':message, 'class':classification})
    index.append(file_name)

  return DataFrame(rows, index= index)



data = DataFrame({'message':[], 'class':[]},  )    #class -> spam OR not_spam
data = data.append(DataFrameFromDir('…/emails/spam', 'spam'))
data = data.append(DataFrameFromDir('…/emails/not_spam', 'not_spam'))


print(data.head())


vectorizer = CountVectorizer()
counts = vectorizer.fit_transform(data['message'].values)
```

```
targets = data['class'].values
print(targets)
print(counts)



classifier = MultinomialNB()
classifier.fit(counts, targets)


examples = ['free now now free free now product']
example_count = vectorizer.transform(examples)
predictions = classifier.predict(example_count)
print(predictions)
```

description

os.walk(path) :

این فانکشن تمام آدرس فایل های داخل یک دایرکتوری را بر میگرداند

root : آدرسی که بهش دادیم

dirs : آدرس دایکتوری های موجود در آدرس

files : آدرس فایل های موجود در آدرس

Email Data Link

Source Code Link