==========

Coding: Random number generation: Give you a dice with 7 sides, how do you generate random numbers between 1 and 10 with equal probability?

def dice():

return np.random.randint(1,7+1)

def r():

def roll():

return dice()-1, dice()-1

while True:

a,b=roll()

k=b+a\*7

if k<40:

break

return k%10+1

==========

Coding: 某个小动物可以一次跳上一个台阶或者两个台阶，问爬上第N级台阶可以有几种方法，写个简单的code出来。

f(N) = f(N-1)+f(N-2)

f=[0,0,1]

def fib(n):

if n==1:

return 0

elif n==2:

return 1

else:

for i in range(3,n+1):

f.append(f[i-1]+f[i-2])

return f[-1]

==========

X ~ N(0, 1), Y ~ N(0, 1), X & Y independent. P(X > 2Y)=? P(X > |2Y|)=?

X-2Y ~ N(0, 5) so P(X > 2Y) = 0.5

==========

Optimal Dice Roll, you have a 6-sided fair dice, your reward is value of very last roll. You have a max of 3 rolls. What is optimal stopping strategy and expected reward amount

Dynamic program

S = {(t,d)}

where t is the number of rolls left, t=1,2,3, and d is dice outcome, d=1,2,3,4,5,6

A= {a=reroll, a`=stop}

Value is outcome of the last roll

or

==========

停车位题。比如你要去上班，公司在一条街道的最底端，你从街道的最上端向公司驶去，在街道的一侧有连续的N 个停车位，问用怎样的策略可以停车后到公司的步行距离最近

Dynamic programming

S = {(d,x)}

d is distance to the company

X is status of a spot

A = {a=continue, a`=stop}

Value is d

假设停车位的availability 是i.i.d.的， 然后用的策略是停在第一个available的车位，问停车后步行到公司距离的期望

Geometric distribution

==========

捕捉标记题，具体数字记不清了，用字母代替，比如第一次捕到a只熊，标记之后放回， 第二次捕了b只其中c只有标记， 问一共多少熊

Suppose there are n bears. Then there are a bears are marked.

If n is small, hypergeometric

If n is large, binomial

Find n that maximizes P(X=c)

==========

They collected a sample of 50 users from each algorithm. The number of users that said they were satisfied was: 45 in the new algorithm and 40 in old algorithm. How would you help that team to interpret these results? Which questions would you ask that team

p1 = 45/50 = 0.9

p2 = 40/50 = 0.8

n1 = 50

n2 = 50

h0: p1 = p2

h2: p1 > p2

Unpooled

var1 = 0.9\*0.1/50 = 0.0018

var2 = 0.8\*0.2/50 = 0.0032

SE = sqrt(0.0018+0.0032) = 0.0707

Z = (p1-p2)/SE = 1.414 < 1.96

Pooled

P\_pool = (45+40)/100 = 0.85

SE = sqrt(0.85\*0.15\*(1/50+1/50)) = 0.0714

Z = (p1-p2)/SE = 1.4 < 1.96

Not significant

==========

7局4胜，打七局的概率是多少

P(7 games) = P(3 wins in first 6 games) = (Binomial)

P(win series in 7 games) = P(See 4 wins with 3 loses) = 5/32 (Negative-binomial)

P(A wins game k|A wins game k-1)=p11, P(A wins game k|A wins game k-1)=p01,

Use simulation to compute P(game 7 is played)

Use Markov chain

p=0.5

p\_01=0.6

p\_11=0.4

def who\_wins(p):

u=np.random.rand()

return u>p

def game():

n=0

A\_win=0

B\_win=0

A\_last\_game=None

while True:

if A\_last\_game is None:

A\_last\_game = who\_wins(p)

elif A\_last\_game==1:

A\_last\_game = who\_wins(p\_01)

else:

A\_last\_game = who\_wins(p\_11)

n=n+1

A\_win+=A\_last\_game

B\_win+=1-A\_last\_game

if A\_win==4 or B\_win==4:

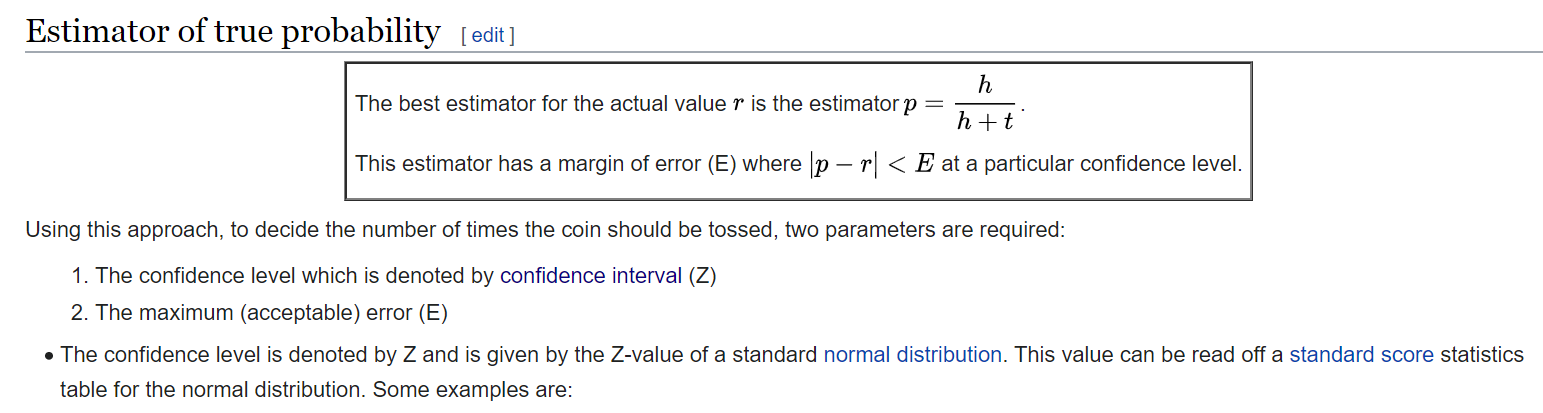
return n

sum([game()//7 for i in range(1000)])/1000

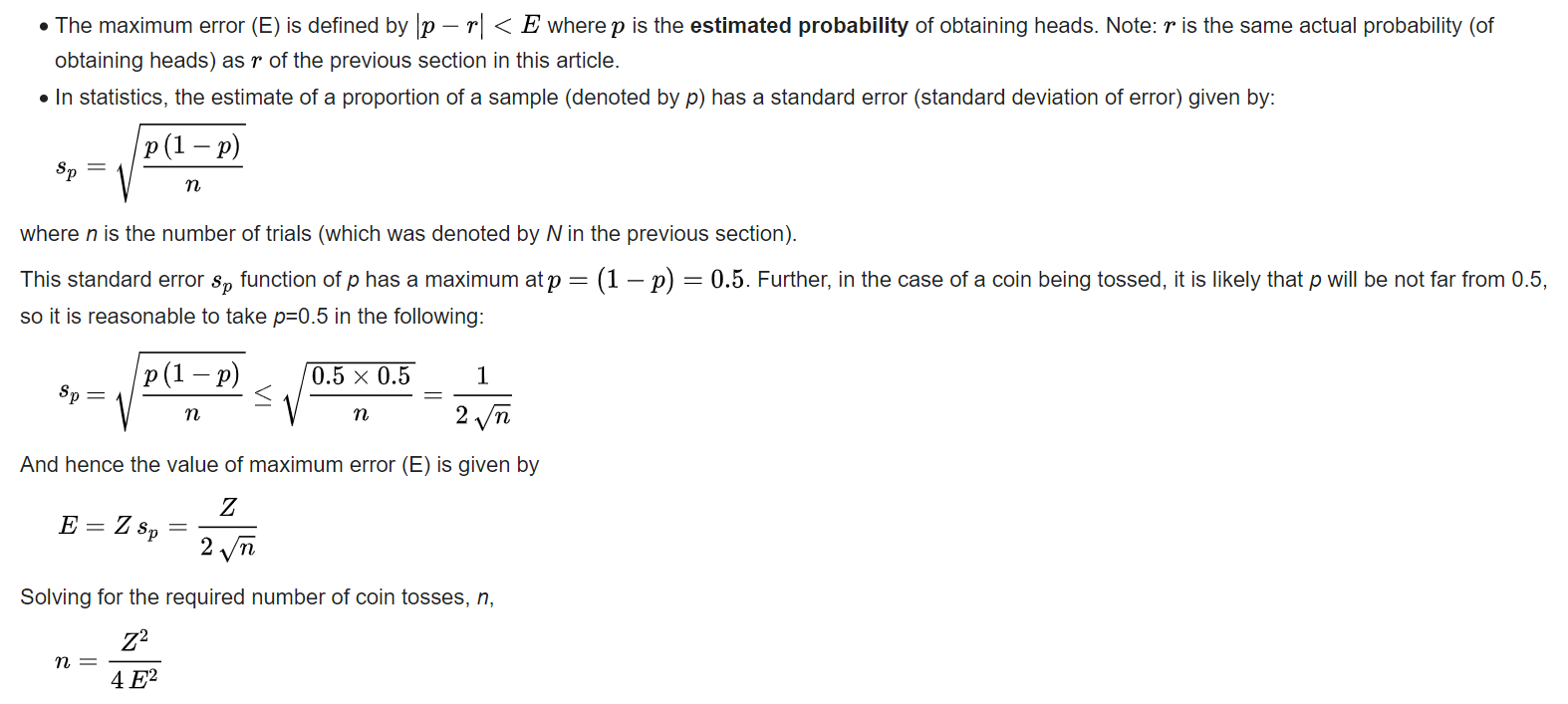
==========

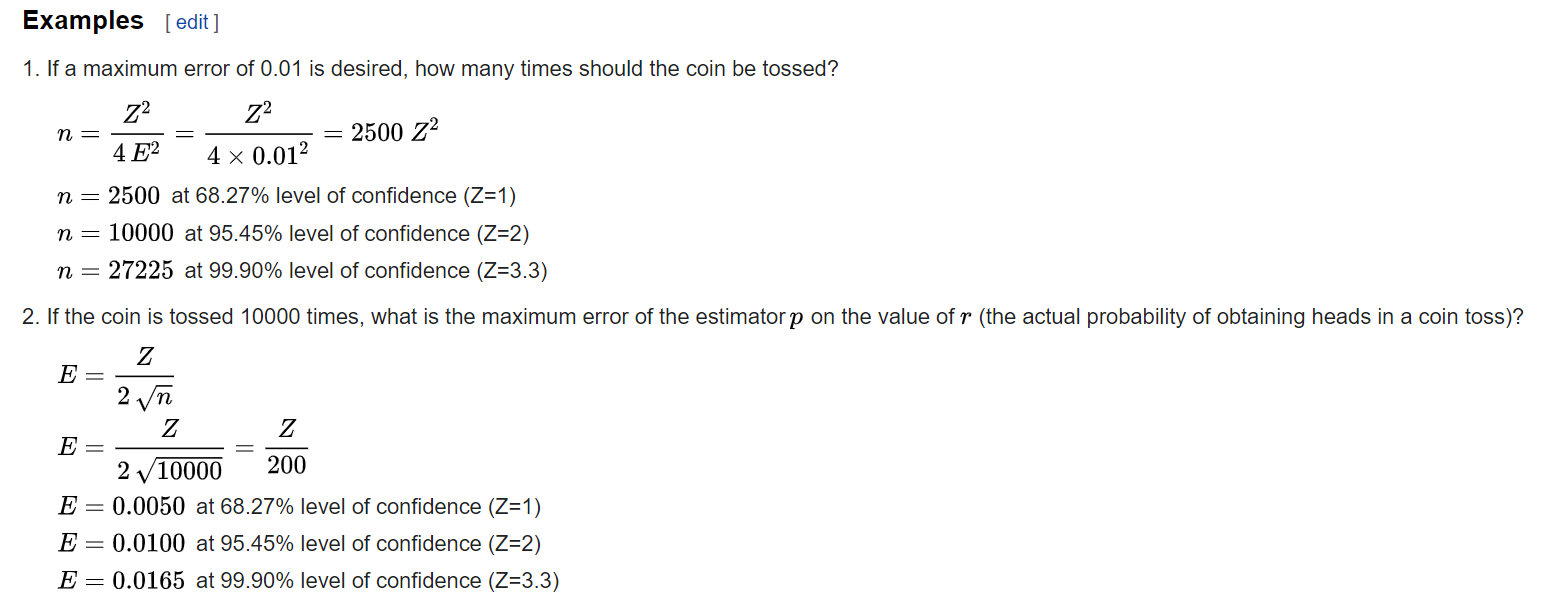
How to test if a coin is fair

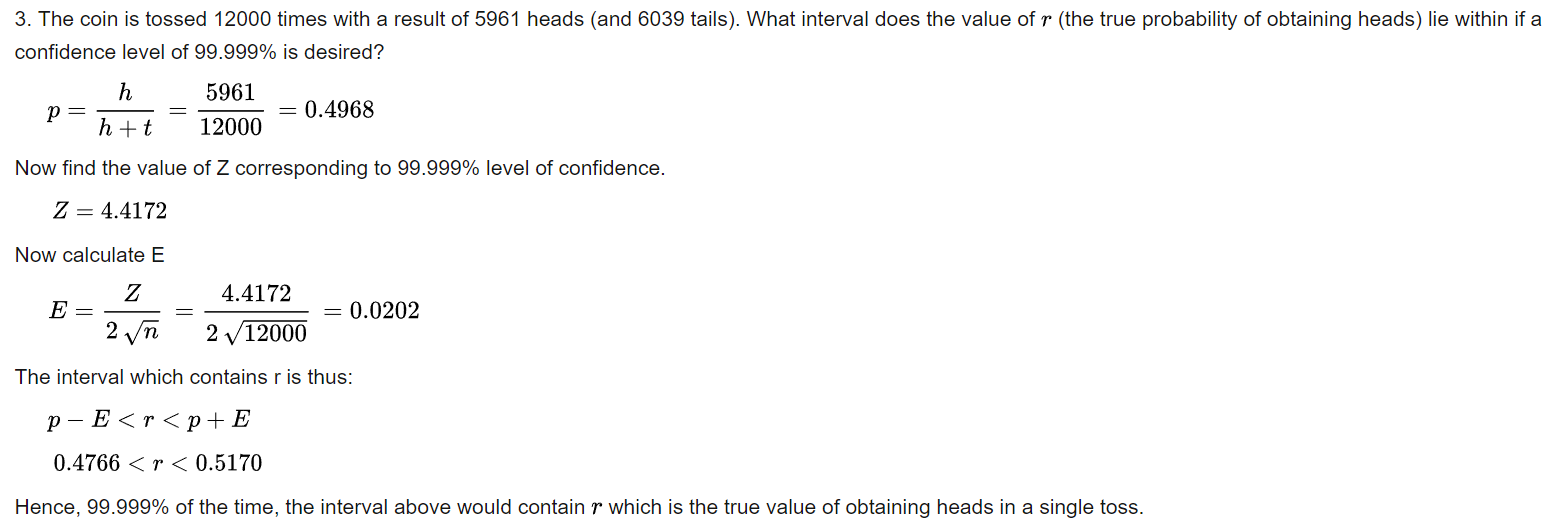
Frequentist











One sample z-test

Cannot reject the null hypothesis

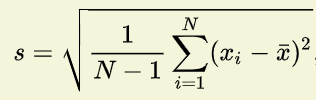
==========

什么是SD？ 如何算SD？

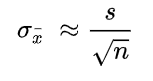
什么是SE？如何算SE？

如何算median 的SE？ 如果sample 过大的话如何算？sample很小的话呢？

SD



SE



SE of median using bootstrap

Take our original sample of n observations, and sample from it, with replacement to create new samples. Each new sample contains n elements. We create B bootstrap samples, where B is a number of 1000 or more. These procedures draw at least 1000 bootstrap samples, and can draw as many as 50,000. For each bootstrap sample we compute the sample median (denoted Med\*), and when we have drawn all of our samples, these values of Med\* represent the sampling distribution of the median. The program then sorts this sampling distribution for low to high and calculates the relevant percentiles. For our purposes here, these will be the 2.5th and 97.5th percentile. For B = 1000, these will be the 25th and 975th order statistics (values from the ordered series). The percentile method would take these to be the upper and lower cutoffs for the 95% confidence interval.

==========

x~ N(mu, 1)，H0: mu = 0，取一个数，啥时候rej H0？

|x|>1.96

从以上的distribution draw sample x，如果 x<1 就discard data，如果还用上面的标准，新的type 1 error是多少？

P(x>1.96|x>1)

==========

Coding. 生成 1000\*2 矩阵: 第一列是按一定概率分布的categorical variables，第二列是正态分布的随机数。然后去除所有第二列小于1的行，再把第二列的数按第一列的categorical variable分类做归一化。不涉及算法

data={

'a':np.random.binomial(1,0.2,1000),

'b':np.random.normal(0,1,1000)

}

# Or use np.random.choice(3,10,[0.1,0.2,0.7])

df=pd.DataFrame(data=data)

df=df.loc[df.b>=1].reset\_index(drop=True)

# Normalization: rescale to range of [0,1]

df.groupby('a').transform(lambda x: (x-x.min())/(x.max()-x.min()))

# Standardization: rescale to have mean=0 and std=1

df.groupby('a').transform(lambda x: (x-x.mean())/x.std())

* Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.
* Standardization, on the other hand, can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

However, at the end of the day, the choice of using normalization or standardization will depend on your problem and the machine learning algorithm you are using. There is no hard and fast rule to tell you when to normalize or standardize your data. You can always start by fitting your model to raw, normalized and standardized data and compare the performance for best results.

==========

怎么用biased coin生成uniform discrete number

def flip(p=0.7):

if np.random.rand()<=p:

return 1

else:

return 0

def rand():

while True:

a=flip()

b=flip()

if a^b:

return a

def du(n):

M=math.ceil(math.log(n,2))

while True:

r=0

for i in range(M):

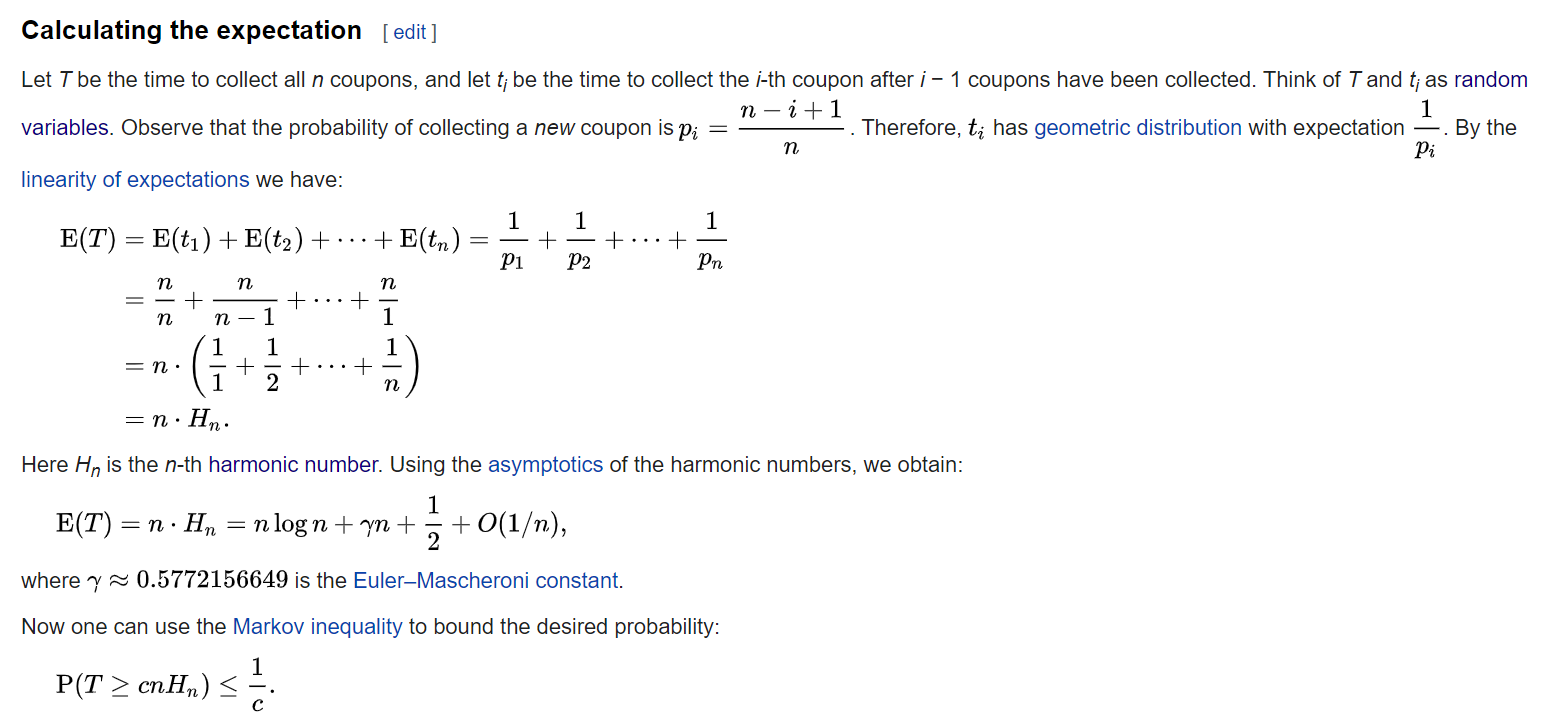
r+=rand()\*2\*\*i

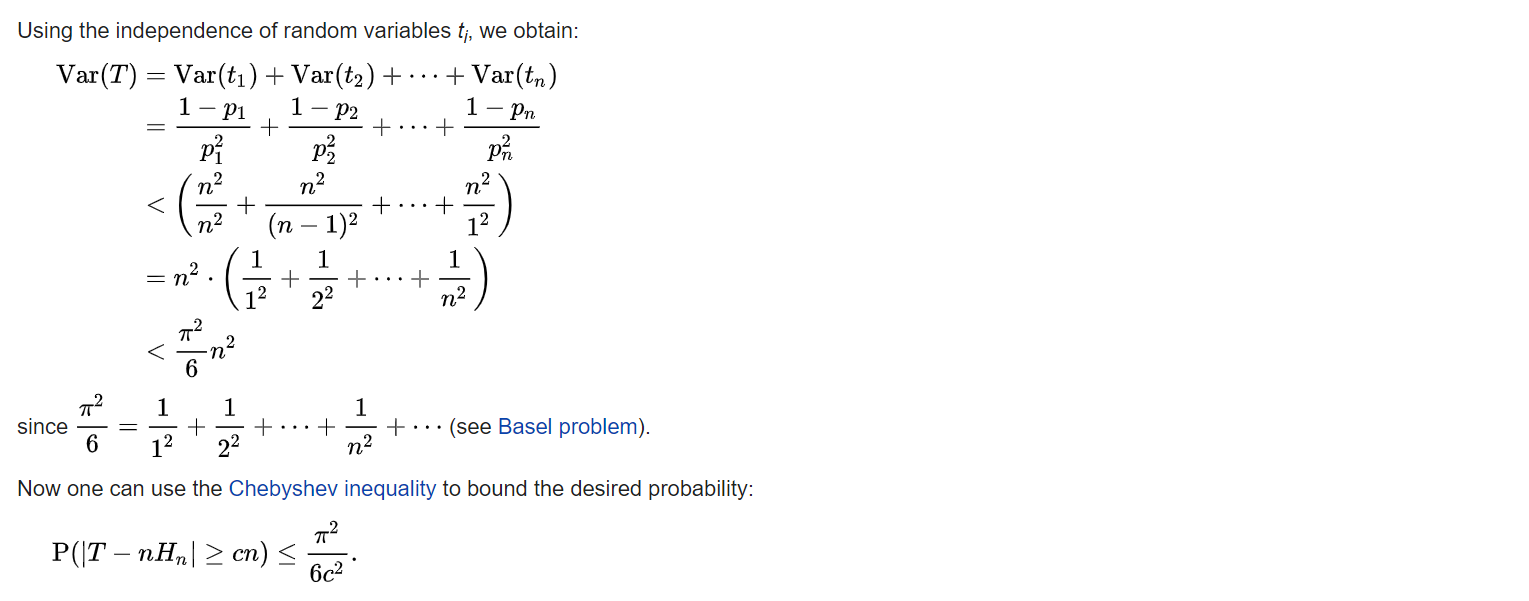
if r<=n:

return r

==========

Coupon collector problem





==========

1000个reviewer, 1000个movie，然后sample rating，问能想到什么distribution, rating可能出现的哪些bias，建模的话需要怎么建，最后可以推导到MF建模

Recommender system overview

Collaborative filtering

Content-based filtering

Matrix factorization algorithm

==========

How to deal with missing data

3 types of missing

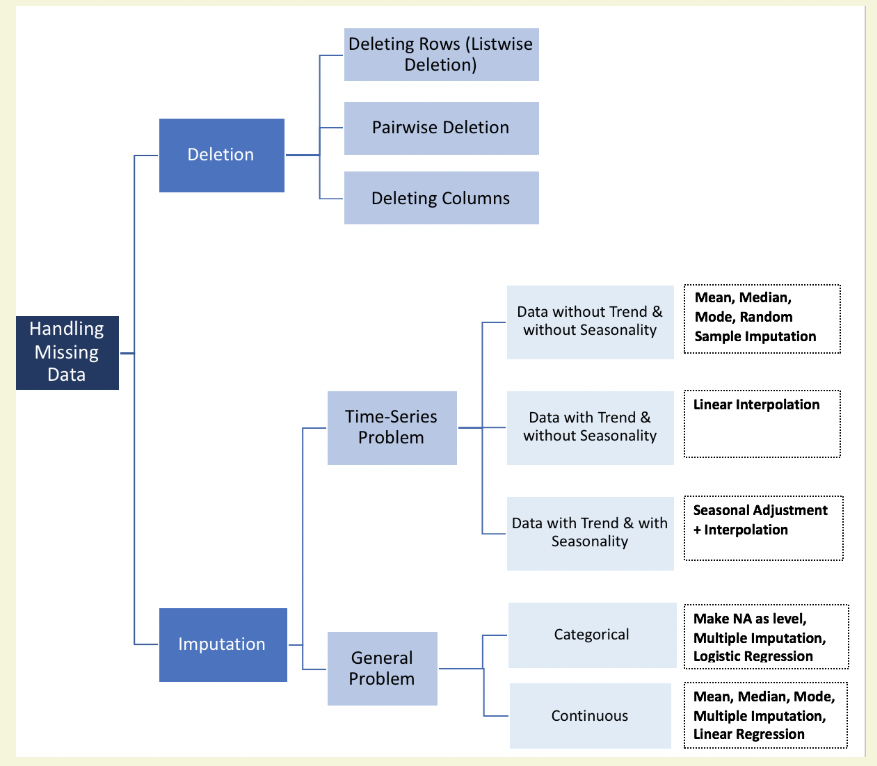
* MCAR: missing completely at random. Missing value y does not depend on y or x
  + e.g. randomly missing the entire record
* MAR: missing at random. Missing value y does not depend on y but depends on x
  + e.g. certain occupation is less likely to report their income
* MNAR: missing not at random. Missing value y depends on y
  + e.g. high-income people are less likely to report their income

Remove data

* Listwise: drop entire row if relevant fields are missing
* Drop variable: drop entire column if variable is not important and has many missing values

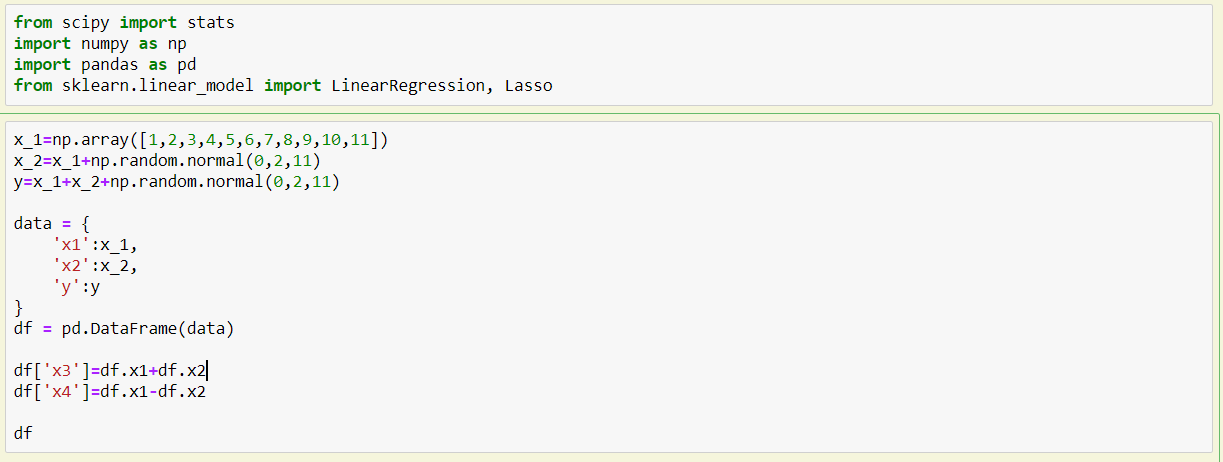
Imputation

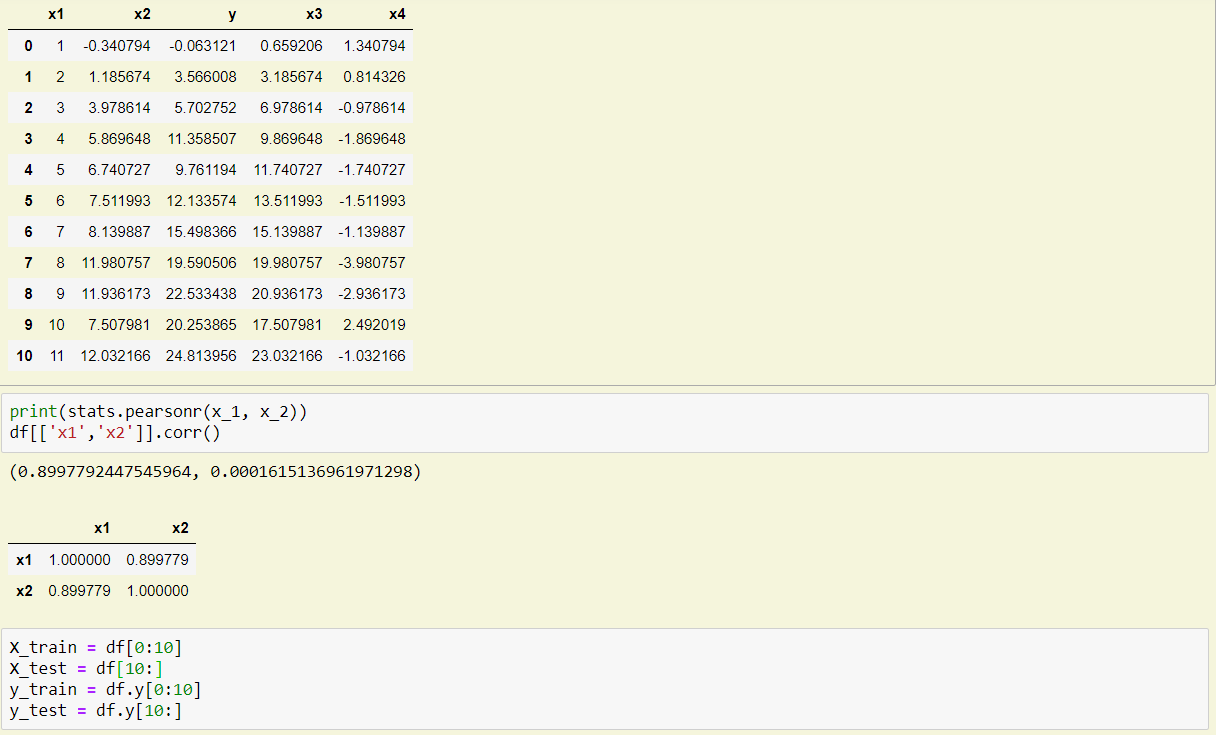
* Another category or -1: tree algorithm handles missing variable as encoded variable
* Mean/median/mode
* Predict missing value based on other features: KNN, regression

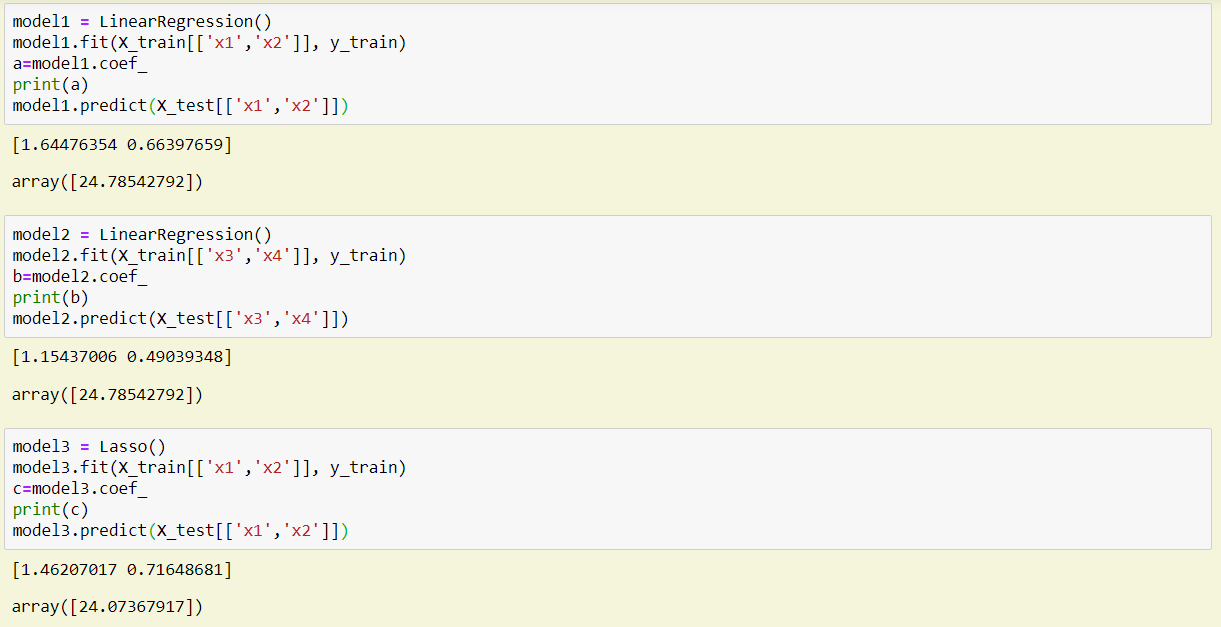


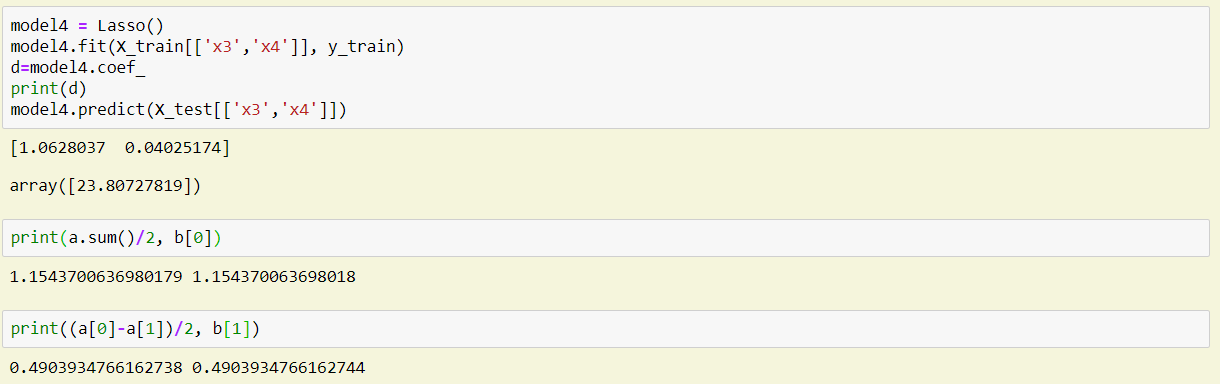
==========

线性回归: X1和X2 correlated, Y~a\*X1+b\*X2 和 Y~c\*(X1+X2)+d\*(X1-X2), 两个模型的预测值是否一样，coef是否一样，如果用lasso回归，预测值和coef会怎么样？









==========

Simpson paradox

Ad ROI for mobile and desktop are both decreased but overall ROI has increased. How to interpret?

ROI is a ratio, so Simpson paradox can happen.

This is because the ROI for mobile and desktop is very different, and investment is switched from one to another.

e.g.

ROI(mobile) decreases from 15/100 to 6/50

ROI(desktop) decreases from 40/50 to 70/100

ROI(total) = 55/150 to 76/13