In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
```

Hypothesis Testing

- 1 Define Ho and Ha
- 2 Check the distribution
- 3 Check right tail vs left tail vs 2 tail
- 4 Calculate pvalue
- 5 Compare pvalue with alpha

In [2]:
```python
#ztest
def calc_ztest(mu,obs,sigma,n):
    std_error = sigma/np.sqrt(n)
    z = (obs-mu)/std_error
    prob_less = norm.cdf(z)
    prob_greater = 1-norm.cdf(z)
    return prob_less,prob_greater
```

In [3]:
```python
#Shampoo Sales
'''
Retail has 2000 outlets weekly sales of shampoo bottles.
Avg = 1800
std_dev = 100.
Team A,tried out on 50 stores. The results are good and apply it on all
2000 outlets.On those 50 stores the avg sales = 1850.
Did the marketing team had an effect?alpha = 0.01
'''
```

Out[3]:
```
'\nRetail has 2000 outlets weekly sales of shampoo bottles.\nAvg = 1800\nstd_dev = 100.\nT
eam A,tried out on 50 stores. The results are good and apply it on all\n2000 outlets.On th
ose 50 stores the avg sales = 1850.\nDid the marketing team had an effect?alpha = 0.01\n'
```

In [4]:
```python
Ho = 'The marketing team did not have any effect'
Ha = 'The marketing team had effect'
alpha = 0.01
std_error = 100/np.sqrt(50)
z = (1850-1800)/std_error
pval = 1-norm.cdf(z)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.00020347600872250293
Reject Ho
```

In [5]:
```python
'''
Retail has 2000 outlets weekly sales of shampoo bottles.
Avg = 1800
std_dev = 100.
```

```
Team B,tried out on 5 stores. The results are good and apply it on all
2000 outlets.On those 50 stores the avg sales = 1900.
Did the marketing team had an effect?alpha = 0.01
'''
```

Out[5]:

```
'\nRetail has 2000 outlets weekly sales of shampoo bottles.\nAvg = 1800\nstd_dev = 100.\nTeam B,tried out on 5 stores. The results are good and apply it on all\n2000 outlets.On those 50 stores the avg sales = 1900.\nDid the marketing team had an effect?alpha = 0.01\n'
```

In [6]:

```
Ho = 'The marketing team did not have any effect'
Ha = 'The marketing team had effect'
alpha = 0.01
std_error = 100/np.sqrt(5)
z = (1900-1800)/std_error
pval = 1-norm.cdf(z)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.0126736593387341
Failed to reject Ho
```

Critical Value

A critical value is a point on the distribution of the test statistic under the null hyothesis which determines a whether a particular region is significant or not.

- if zscore > critcial value: Reject Ho

In [7]:

```
'''
A country has a population average height of 65 inches with standard deviation of 2.5.
A person feels people from his state are shorter.
He takes the average of 20 people, and sees that it is 64.5.
At a 5% significance level (or 95% confidence level),
can we conclude that people from his state are shorter, using the Z-test? What is the p-va
'''
```

Out[7]:

```
'\nA country has a population average height of 65 inches with standard deviation of 2.5.\nA person feels people from his state are shorter.\nHe takes the average of 20 people, and sees that it is 64.5.\nAt a 5% significance level (or 95% confidence level), \ncan we conclude that people from his state are shorter, using the Z-test? What is the p-value?\n'
```

In [8]:

```
Ho = 'People are not short'
Ha = 'People are short'
mu = 65
sigma = 2.5
obs = 64.5
n = 20
alpha = 0.05
#Now
std_error = 2.5/np.sqrt(20)
z = (obs-mu)/std_error
pval = norm.cdf(z)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.185546684761348 78
Failed to reject Ho
```

In [9]:
```python
'''
The verbal reasoning in GRE has an average score of 150,
and a standard deviation of 8.5.
A coaching center claims that their students are better.
An average of 10 people showed that the students
from this coaching center have an average of 155.
At a 5% significance level (or 95% confidence level),
can we conclude that students from the coaching center are better?
Use the Z-test, and compute the p-value.
'''
```

Out[9]:
```
'\nThe verbal reasoning in GRE has an average score of 150, \nand a standard deviation of 8.5. \nA coaching center claims that their students are better.\nAn average of 10 people showed that the students \nfrom this coaching center have an average of 155.\nAt a 5% significance level (or 95% confidence level), \ncan we conclude that students from the coaching center are better?\nUse the Z-test, and compute the p-value.\n'
```

In [10]:
```python
Ho = 'The students are not better'
Ha = 'The students are better'
mu = 150
sigma = 8.5
n = 10
obs = 155
alpha = 0.05
std_error = sigma/np.sqrt(n)
z = (obs-mu)/std_error
pval = 1-norm.cdf(z)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.031431210741779014
Reject Ho
```

In [11]:
```python
'''
A french cake shop claims that the average number of pastries
they can produce in a day exceeds 500.

The average number of pastries produced per day over a 70 day period was found to be 530.
Assume that the population standard deviation for the pastries produced per day is 125.

Test the claim using a z-test with the critical z-value = 1.64
at the alpha (significance level) = 0.05, and state your interpretation.
'''
```

Out[11]:
```
'\nA french cake shop claims that the average number of pastries \nthey can produce in a day exceeds 500.\n\nThe average number of pastries produced per day over a 70 day period was found to be 530.\nAssume that the population standard deviation for the pastries produced per day is 125.\n\nTest the claim using a z-test with the critical z-value = 1.64 \nat the alpha (significance level) = 0.05, and state your interpretation.\n'
```

In [12]:
```python
Ho = 'The average number of pastries produced doesnot exceed 500'
Ha = 'The average number of pastries produced doesnot exceed 500'
mu = 500
n = 70
sigma = 125
obs = 530
alpha = 0.05
```

```python
std_error = sigma/np.sqrt(n)
z = (obs-mu)/std_error
print(z)
pval = 1-norm.cdf(z)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
2.007984063681781
0.022322492581293485
Reject Ho
```

In [13]:

```python
'''
The Chai Point stall at Bengaluru airport estimates that each person visiting the store dr
of 1.7 small cups of tea.

Assume a population standard deviation of 0.5 small cups. A sample of 30 customers
collected over a few days averaged 1.85 small cups of tea per person.

Test the claim using a z-test at an alpha = 0.05 significance value,
with a critical z-score value of ±1.96.
'''
```

Out[13]:

```
'\nThe Chai Point stall at Bengaluru airport estimates that each person visiting the store
drinks an average \nof 1.7 small cups of tea.\n\nAssume a population standard deviation of
0.5 small cups. A sample of 30 customers\ncollected over a few days averaged 1.85 small cu
ps of tea per person.\n\nTest the claim using a z-test at an alpha = 0.05 significance val
ue,\nwith a critical z-score value of ±1.96.\n'
```

In [14]:

```python
Ho = 'mu = 1.7'
Ha = 'mu != 1.7'
mu = 1.7
n = 30
sigma = 0.5
obs = 1.85
alpha = 0.05
c_val = 1.96
std_error = sigma/np.sqrt(n)
z = (obs-mu)/std_error
print(z)
#Right tailed test
print(norm.ppf(0.95))
if z>c_val:
    print('Reject Ho')
else:
    print('Fail to reject Ho')
```

```
1.6431676725155
1.6448536269514722
Fail to reject Ho
```

In [15]:

```python
'''
The student hostel office at IIT Madras estimates that each student
uses more than 3.5 buckets of water per day.
45 students in a certain wing averaged 3.72 buckets of water per day.

Assume that the population standard deviation is 0.7 buckets.
What is the critical sample mean for this population, assuming a critical z- value of 1.28
'''
```

Out[15]: '\nThe student hostel office at IIT Madras estimates that each student\nuses more than 3.5 buckets of water per day.\n45 students in a certain wing averaged 3.72 buckets of water pe r day.\n\nAssume that the population standard deviation is 0.7 buckets. \nWhat is the crit ical sample mean for this population, assuming a critical z- value of 1.28?\n'

In [16]:
```
mu = 3.5
n = 45
obs = 3.72
sigma = 0.7
z = 1.28
std_error = 0.7/np.sqrt(n)
c_val = (z*std_error)+mu
print(c_val)
```

```
3.6335677938559874
```

Ttest

When population std deviation is not given we cannot use ztest,so we use ttest.

- n vs c and c is having 2 categories only than we perform ttest

In [17]:
```
from scipy.stats import ttest_1samp,ttest_ind
```

In [18]:
```
#One sample ttest(one set of sample vs constant)
iq_scores = [110,105,98,102,99,104,115,95]
alpha = 0.01
Ho = 'mu = 100'
Ha = 'mu > 100'
tstat, pval = ttest_1samp(iq_scores,100)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.1754994493585011
Failed to reject Ho
```

Two tailed

When we have 2 samples we use ttest_ind

- Ho: mu1 = mu2
- Ha: mu1 != mu2 ## Greater
- Ho: mu1 = mu2
- Ha: mu1 > mu2 ## Less
- Ho: mu1 = mu2
- Ha: mu1 < mu2

In [19]:
```
#Iq across 2 schools
df_iq = pd.read_csv('iq_two_schools.csv')
df_iq
```

Out[19]:

	School	iq
0	school_1	91

	School	iq
1	school_1	95
2	school_1	110
3	school_1	112
4	school_1	115
5	school_1	94
6	school_1	82
7	school_1	84
8	school_1	85
9	school_1	89
10	school_1	91
11	school_1	91
12	school_1	92
13	school_1	94
14	school_1	99
15	school_1	99
16	school_1	105
17	school_1	109
18	school_1	109
19	school_1	109
20	school_1	110
21	school_1	112
22	school_1	112
23	school_1	113
24	school_1	114
25	school_1	114
26	school_2	112
27	school_2	115
28	school_2	95
29	school_2	92
30	school_2	91
31	school_2	95
32	school_2	91
33	school_2	99
34	school_2	111
35	school_2	115
36	school_2	108

	School	iq
37	school_2	109
38	school_2	109
39	school_2	114
40	school_2	115
41	school_2	116
42	school_2	117
43	school_2	117
44	school_2	128
45	school_2	129
46	school_2	130
47	school_2	133
48	school_2	95
49	school_2	90

In [20]:

```
df_iq.groupby('School')['iq'].mean()
```

Out[20]:

```
School
school_1    101.153846
school_2    109.416667
Name: iq, dtype: float64
```

In [21]:

```
iq_1 = df_iq[df_iq['School']=='school_1']['iq']
iq_2 = df_iq[df_iq['School']=='school_2']['iq']
```

In [22]:

```
iq_1.head()
```

Out[22]:

```
0     91
1     95
2    110
3    112
4    115
Name: iq, dtype: int64
```

In [23]:

```
iq_2.head()
```

Out[23]:

```
26    112
27    115
28     95
29     92
30     91
Name: iq, dtype: int64
```

In [24]:

```
Ho = 'mu1 = mu2'
Ha = 'mu1 != mu2'
alpha = 0.05
tstat,pval = ttest_ind(iq_1,iq_2)
print(pval)
if pval<alpha:
    print('Reject Ho')
```

```
else:
    print('Failed to reject Ho')
```

```
0.020045527109362l7
Reject Ho
```

In [25]:

```
#order of the samples is important
Ho = 'mu1  = mu2'
Ha = 'mu1 > mu2'
alpha = 0.05
tstat,pval = ttest_ind(iq_1,iq_2,alternative = 'greater')
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.989977236445319
Failed to reject Ho
```

In [26]:

```
Ho = 'mu1 = mu2'
Ha = 'mu1 < mu2'
alpha = 0.05
tstat,pval = ttest_ind(iq_1,iq_2,alternative = 'less')
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.010022763554681085
Reject Ho
```

In [27]:

```
Ho = 'mu1 = mu2'
Ha = 'mu1 < mu2'
alpha = 0.05
tstat,pval = ttest_ind(iq_2,iq_1,alternative = 'greater')
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.010022763554681085
Reject Ho
```

In [28]:

```
df = pd.read_csv("aerofit.csv")
df
```

Out[28]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	Male	14	Single	3	4	29562	112
1	KP281	19	Male	15	Single	2	3	31836	75
2	KP281	19	Female	14	Partnered	4	3	30699	66
3	KP281	19	Male	12	Single	3	3	32973	85
4	KP281	20	Male	13	Partnered	4	2	35247	47
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

180 rows × 9 columns

In [29]:

```
income_male = df.loc[df['Gender']=='Male','Income']
income_female = df.loc[df['Gender']=='Female','Income']
```

In [30]:

```
df.groupby('Gender')['Income'].mean()
```

Out[30]:

```
Gender
Female    49828.907895
Male      56562.759615
Name: Income, dtype: float64
```

In [31]:

```
alpha = 0.05
Ho = 'Income of both the genders is same'
Ha = 'Income of male > income of females'
tstat,pval = ttest_ind(income_male,income_female, alternative = 'greater')
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Fail to reject Ho')
```

```
0.003263631548607129
Reject Ho
```

In [32]:

```
'''
Based on field experiments, a new variety green gram is expected to given an yield of 12.(

The variety was tested on 10 randomly selected farmers fields. The yield (quintals/hectare

[14.3,12.6,13.7,10.9,13.7,12.0,11.4,12.0,12.6,13.1]

With 5% significance level can we conclude that average yield is more than the expected yi
'''
```

Out[32]:

```
'\nBased on field experiments, a new variety green gram is expected to given an yield of 1
2.0 quintals per hectare.\n\nThe variety was tested on 10 randomly selected farmers field
s. The yield (quintals/hectare) were recorded as\n\n[14.3,12.6,13.7,10.9,13.7,12.0,11.4,1
2.0,12.6,13.1]\n\nWith 5% significance level can we conclude that average yield is more th
an the expected yield?\n'
```

In [33]:

```
a = [14.3,12.6,13.7,10.9,13.7,12.0,11.4,12.0,12.6,13.1]
print(np.mean(a))
Ho = 'mu = 12'
Ha = 'mu > 12'
alpha = 0.05
#Since we have one sample here we use ttest_1samp
tstat,pval = ttest_1samp(a,12,alternative = 'greater')
print(pval)
if pval<alpha:
    print('Reject Ho')
```

```
else:
    print('Failed to reject Ho')
```

```
12.629999999999999
0.04979938002326663
Reject Ho
```

In [34]:
```
'''
The one sample T-test is used when we want to compare a sample mean to a population mean.

The average British man is 175.3 cm tall.
A survey recorded the heights of 10 UK men and we want to know whether
the mean of the sample is different from the population mean.

survey_height = [177.3, 182.7, 169.6, 176.3, 180.3, 179.4, 178.5, 177.2, 181.8, 176.5]
'''
```

Out[34]:
```
'\nThe one sample T-test is used when we want to compare a sample mean to a population mea
n.\n\nThe average British man is 175.3 cm tall.\nA survey recorded the heights of 10 UK me
n and we want to know whether \nthe mean of the sample is different from the population me
an.\n\nsurvey_height = [177.3, 182.7, 169.6, 176.3, 180.3, 179.4, 178.5, 177.2, 181.8, 17
6.5]\n'
```

In [35]:
```
Ho = 'mu = 175.3'
Ha = 'mu != 175.3'
alpha = 0.05
survey_height = [177.3, 182.7, 169.6, 176.3, 180.3, 179.4, 178.5, 177.2, 181.8, 176.5]
tstat,pval = ttest_1samp(survey_height,175.3)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.04734137339747034
Reject Ho
```

In [36]:
```
'''
We have the potato yield from 12 different farms.

We know that the standard potato yield for the given variety is µ = 20.

x = [21.5, 24.5, 18.5, 17.2, 14.5, 23.2, 22.1, 20.5, 19.4, 18.1, 24.1, 18.5]

Test if the potato yield from these farms is significantly higher than the standard yield
'''
```

Out[36]:
```
'\nWe have the potato yield from 12 different farms.\n\nWe know that the standard potato y
ield for the given variety is µ = 20.\n\nx = [21.5, 24.5, 18.5, 17.2, 14.5, 23.2, 22.1, 2
0.5, 19.4, 18.1, 24.1, 18.5]\n\nTest if the potato yield from these farms is significantly
higher than the standard yield with 5% significance level.\n'
```

In [37]:
```
ho = 'mu = 20'
ha = 'mu > 20'
x = [21.5, 24.5, 18.5, 17.2, 14.5, 23.2, 22.1, 20.5, 19.4, 18.1, 24.1, 18.5]
alpha = 0.05
tstat,pval = ttest_1samp(x,20,alternative = 'greater')
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.4223145946526807
Failed to reject Ho
```

In [38]:

```
'''
Samples of Body fat percentages of few gym going men and women are recorded.

men = [13.3, 6.0, 20.0, 8.0, 14.0, 19.0, 18.0, 25.0, 16.0, 24.0, 15.0, 1.0, 15.0]
women = [22.0, 16.0, 21.7, 21.0, 30.0, 26.0, 12.0, 23.2, 28.0, 23.0]

Perform 2 sample T-test to check if mean body fat percentage of men and women are statisti

Assume significance level to be 5%.
'''
```

Out[38]:

```
'\nSamples of Body fat percentages of few gym going men and women are recorded.\n\nmen =
[13.3, 6.0, 20.0, 8.0, 14.0, 19.0, 18.0, 25.0, 16.0, 24.0, 15.0, 1.0, 15.0]\nwomen = [22.
0, 16.0, 21.7, 21.0, 30.0, 26.0, 12.0, 23.2, 28.0, 23.0]\n\nPerform 2 sample T-test to che
ck if mean body fat percentage of men and women are statistically different.\n\nAssume sig
nificance level to be 5%.\n'
```

In [39]:

```
men = [13.3, 6.0, 20.0, 8.0, 14.0, 19.0, 18.0, 25.0, 16.0, 24.0, 15.0, 1.0, 15.0]
women = [22.0, 16.0, 21.7, 21.0, 30.0, 26.0, 12.0, 23.2, 28.0, 23.0]
Ho = 'Body percentages are same'
Ha = 'Body fat percentages are different'
alpha = 0.05
tstat,pval = ttest_ind(men,women)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.010730607904197957
Reject Ho
```

In [40]:

```
'''
IQ score samples from two schools are collected.
Perform T-test with 5% significance level to check if there is any
statistically significant difference in mean IQ's of two schools.

school_1 = [115, 111, 112, 101, 95, 98, 100, 90, 89, 108]
school_2 = [107, 103, 91, 99, 104, 98, 117, 113, 92, 96, 108, 115, 116, 88]
'''
```

Out[40]:

```
"\nIQ score samples from two schools are collected.\nPerform T-test with 5% significance l
evel to check if there is any\nstatistically significant difference in mean IQ's of two sc
hools.\n\nschool_1 = [115, 111, 112, 101, 95, 98, 100, 90, 89, 108]\nschool_2 = [107, 103,
91, 99, 104, 98, 117, 113, 92, 96, 108, 115, 116, 88]\n"
```

In [41]:

```
school_1 = [115, 111, 112, 101, 95, 98, 100, 90, 89, 108]
school_2 = [107, 103, 91, 99, 104, 98, 117, 113, 92, 96, 108, 115, 116, 88]
alpha = 0.05
Ho = 'Iq is same'
Ha = 'Iq is different'
tstat,pval = ttest_ind(school_1,school_2)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.7154458095186707
Failed to reject Ho
```

Chi square

In [42]:
```
from scipy.stats import chisquare,chi2_contingency
```

Degree of Freedom:

- Given n numbers and their average is known, totally how many of the n values we shall know (n-1)

2 Arrays

- Height and weight
- Suppose I have averages of height and weight
- How many values can be unknown? Answer is 1 of Height and 1 of weight can be unknown.
- n1 is number of values for height
- n2 is number of values for weight
- n1-1+n2-1 number of known values
- n1+n2-2
- Degree of freedom is 2

Win Century F T F 160 154 T 176 184
Both row and column sum are known the number of unknown values is 3. Degree of freedom is 1, because even if 1 value is known we can figure out the rest.

A B C D x y z Both rown and column sum are known the number of unknown values is 6 Degree of freedom = (rows-1)*(columns-1)

Chisquare is used in case of cat vs cat.

A coin is tossed 50 times H T Expected 25 25 Acutal 28 22

Chisquare formula:

- Ho:Fair coin
- Ha:Biased coin
- chi**2 = (28-25)**2/25 + (22-25)**2/25

In [43]:
```
alpha = 0.05
chi_stat,pval = chisquare([28,22],#Observed
                          [25,25]#Expected
                          )
print(pval)
print(chi_stat)
```

```
0.3961439091520741
0.72
```

In [44]:
```
Ho = 'Gender and preference independent'
Ha = 'Gender and preference is dependent'
observed = [[527,72],
            [206,102]]
chi_stat,pval,df,exp_freq = chi2_contingency(observed)
print(chi_stat)
print(pval)
print(df)
print(exp_freq)
```

```
57.04098674049609
```

```
4.268230756875865e-14
1
[[484.08710033 114.91289967]
 [248.91289967  59.08710033]]
```

When to use chisquare

- Cat vs Cat
- Type 1: To compare expected vs actual results.
- Type 2: To check independence.

Assumptions of chisquare

- 1) Both variable are categorical
- 2) Observations are independent i.e the value of one observation does not affect any value of the other observation.
- 3) Each cell is mutually exclusive(0 intersection) in the contingency table i.e one individual cell cannot belong to more than one cell.
- 4) Expected value in each cells in the contingency table should be 5 or greater in at least 80% of cells that no cell should have an expected value less than 1. Expected value = (row_Sum * col_sum)/Table_sum

Sampling Methods

- Probablity Sampling Methods
  - 1) Simple Random Sample:
    - Here every member in a population has an equal probablity of being selected to be in the sample.Randomly seleceted members.Sinle random samples are usually repersentative of the population we are interested in sice every member has an equal chance of being included in the sample.
  - 2) Stratified random sample:
    - Split a population into groups.Random;ly select some members from each group to be in the sample. Statified random samples ensure that members from each group in the population are included in the survey.
  - 3) Cluster Random Sample:
    - Split a population into clusters.Randomly select some of the clusters and include all members from those clusters in the sample. Cluster random sample gets every member from some of the groups, which is useful when each group is reflective of the population as a whole.
  - 4) Systematic random sample:
    - Put every member of a population into some order. Choosing a random starting point and select every nth member to be in the sample. Systematic random sample are usually representative of the population we are interested in since every memeber has an equal chance of being included in the sample.

Anova: Analysis of Variance

- Numerical vs Categorical -----> more than 2 categories.

A one way anova compares the means of 3 or more independent groups to determine if there is a statistically significant difference between the corresponding population means.

Why not Pairwise ttest for categorical values having more than 2 variables?

- Too many tests
- Error compounding
- Every time you conduct a ttest there is a chance that you will make a Type 1 error. This error is usally 5%. By running two ttest on the same data you wil have increased your chance of 'making a mistake' to 10%.

One way Anova: Assumtions:

- 1) Normality - each sample was drawn from a normallyl distributed population.
  - QQ test and Shapiro test
- 2) Row independence
- 3) Equal variance in different groups.
  - Levene Test
- 4)If any of these assumptions don't hold we use kruskal wallis test.
  - Kruskal wallis us used only when anova fials and is used for n vs c where c is more than 2 categories.

In [45]:
```
from scipy.stats import f_oneway
```

In [46]:
```
df = pd.read_csv('aerofit.csv')
df
```

Out[46]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	Male	14	Single	3	4	29562	112
1	KP281	19	Male	15	Single	2	3	31836	75
2	KP281	19	Female	14	Partnered	4	3	30699	66
3	KP281	19	Male	12	Single	3	3	32973	85
4	KP281	20	Male	13	Partnered	4	2	35247	47
...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200
176	KP781	42	Male	18	Single	5	4	89641	200
177	KP781	45	Male	16	Single	5	5	90886	160
178	KP781	47	Male	18	Partnered	4	5	104581	120
179	KP781	48	Male	18	Partnered	4	5	95508	180

180 rows × 9 columns

In [47]:
```
df['random_group'] = np.random.choice(['g1','g2','g3'],size = len(df))
df
```

Out[47]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	random_group
0	KP281	18	Male	14	Single	3	4	29562	112	g3
1	KP281	19	Male	15	Single	2	3	31836	75	g2
2	KP281	19	Female	14	Partnered	4	3	30699	66	g3

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	random_group
3	KP281	19	Male	12	Single	3	3	32973	85	g1
4	KP281	20	Male	13	Partnered	4	2	35247	47	g1
...	...	...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416	200	g2
176	KP781	42	Male	18	Single	5	4	89641	200	g2
177	KP781	45	Male	16	Single	5	5	90886	160	g1
178	KP781	47	Male	18	Partnered	4	5	104581	120	g3
179	KP781	48	Male	18	Partnered	4	5	95508	180	g1

180 rows × 10 columns

In [48]:

```
Ho = 'All means are equal'
Ha = 'Some means are different'
income_g1 = df[df['random_group']=='g1']['Income']
income_g2 = df[df['random_group']=='g2']['Income']
income_g3 = df[df['random_group']=='g3']['Income']
alpha = 0.01
f_stats,pval = f_oneway(income_g1,income_g2,income_g3)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.9788576649244534
Failed to reject Ho
```

Kruskal

In [49]:

```
from scipy.stats import kruskal
Ho = 'All means are equal'
Ha = 'Some means are different'
alpha = 0.01
k_stat,pval = kruskal(income_g1,income_g2,income_g3)
print(pval)
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.9372864163698341
Failed to reject Ho
```

QQ-Plot: quantile quantile plot

A graph follows normal distribution only and only if it follows emperical formula.And by looking at the graph we cannot figure out whether it is normal distribution or not. If the data is normal gaussian plot it will be linear.

In [50]:

```
from statsmodels.graphics.gofplots import qqplot
```

In [51]:

```
df_1 = pd.read_csv('weight-height.csv')
```

```
df_1
```

Out[51]:

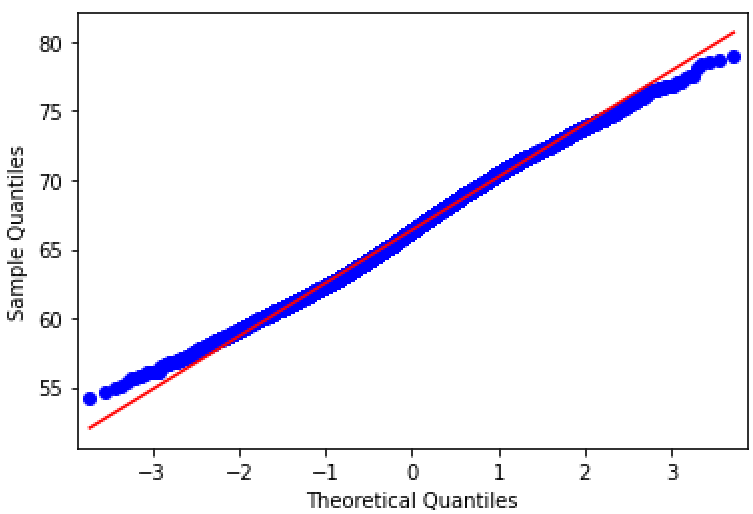
	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801
...	...	...	...
9995	Female	66.172652	136.777454
9996	Female	67.067155	170.867906
9997	Female	63.867992	128.475319
9998	Female	69.034243	163.852461
9999	Female	61.944246	113.649103

10000 rows × 3 columns

In [52]:

```
height = df_1['Height']
qqplot(height,line = 's')
plt.show()
```

```
C:\Users\debas\anaconda3\lib\site-packages\statsmodels\graphics\gofplots.py:993: UserWarni
ng: marker is redundantly defined by the 'marker' keyword argument and the fmt string "bo"
(-> marker='o'). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```



In [53]:

```
df_2 = pd.read_csv('waiting_time.csv')
df_2
```

Out[53]:

	time
0	184.003075
1	36.721521
2	29.970417

	time
3	75.640285
4	61.489439
...	...
90041	135.885984
90042	15.223970
90043	207.839528
90044	140.488418
90045	50.719544

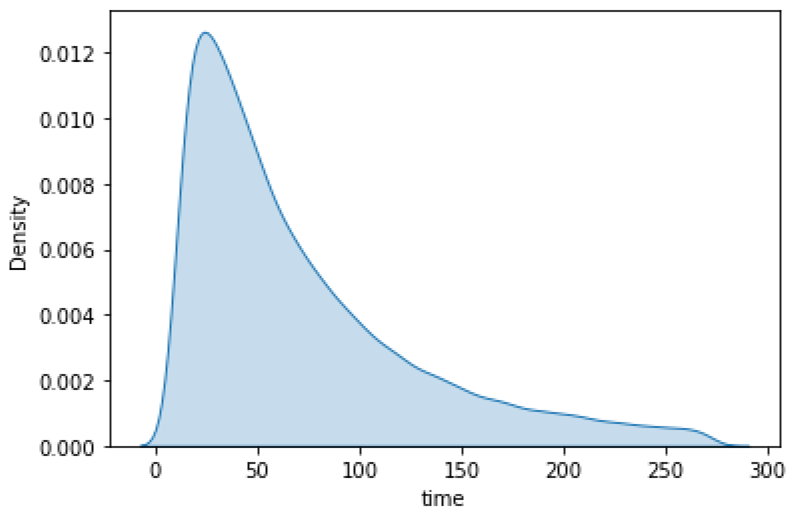
90046 rows × 1 columns

In [54]:

```
sns.kdeplot(df_2['time'],fill = True)
```

Out[54]:

```
<AxesSubplot:xlabel='time', ylabel='Density'>
```

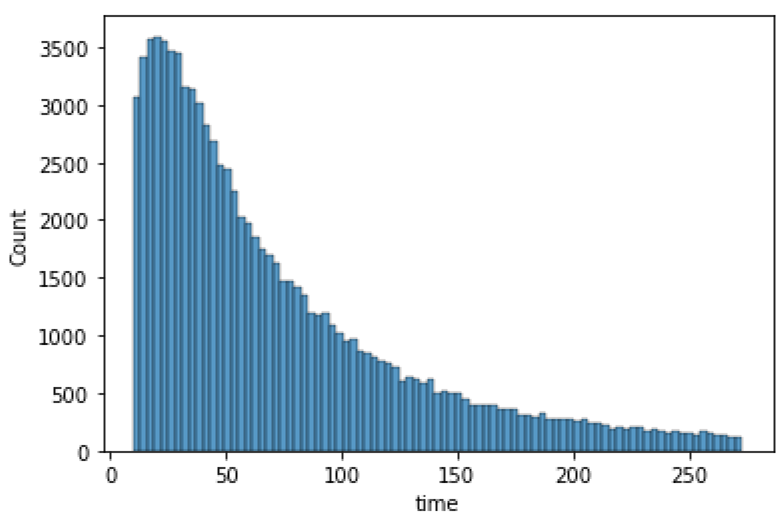


In [55]:

```
sns.histplot(df_2['time'])
```

Out[55]:

```
<AxesSubplot:xlabel='time', ylabel='Count'>
```

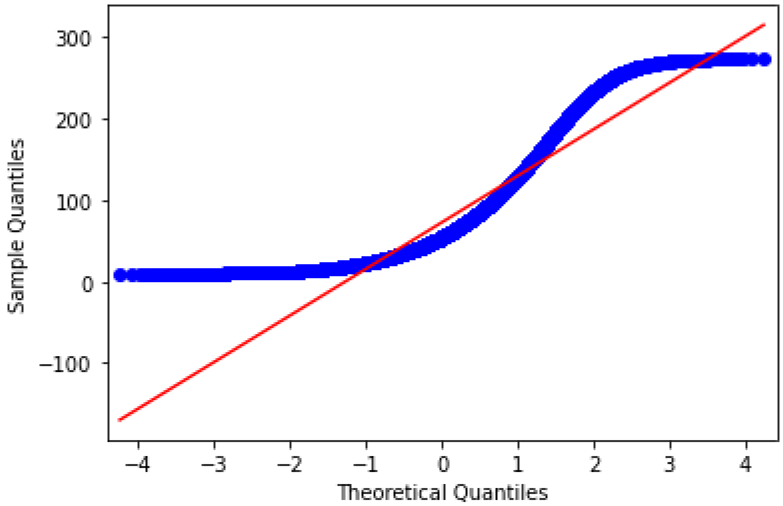


In [56]:

```
qqplot(df_2['time'],line = 's')
```

```
plt.show()
```

```
C:\Users\debas\anaconda3\lib\site-packages\statsmodels\graphics\gofplots.py:993: UserWarning: marker is redundantly defined by the 'marker' keyword argument and the fmt string "bo" (-> marker='o'). The keyword argument will take precedence.
  ax.plot(x, y, fmt, **plot_style)
```



Shapiro Test

Take a few samples of data(50 to 200). This may not work if data is too large.

In [57]:
```
from scipy.stats import shapiro
height_subset = height.sample(100)
Ho = 'Data is gaussian'
Ha = 'Data is not gaussian'
shapiro_test,pval = shapiro(height_subset)
print(pval)
alpha = 0.05
if pval<alpha:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.354047759645462036
Failed to reject Ho
```

Levene Test

In [58]:
```
height_man = df_1[df_1['Gender']=='Male']['Height']
height_women = df_1[df_1['Gender']=='Female']['Height']
```

In [59]:
```
from scipy.stats import levene
Ho = 'Variance are equal'
Ha = 'Variance are not equal'
levene_stat,pval = levene(height_man,height_women)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Fail to reject Ho')
```

```
0.0004586349895436178
Reject Ho
```

Correlation Test

Numeric vs Numeric

Population Covariance Formula

$$Cov(x,y) = \frac{\Sigma(x_i-\bar{x})(y_i-\bar{y})}{N}$$

Sample Covariance

$$Cov(x,y) = \frac{\Sigma(x_i-\bar{x})(y_i-y)}{N-1}$$

Disadvantage of covariance:

Covariance can only measure the directional relationship between two assets. It cannot show the strength of the relationship between assets.

In [60]:
```
df = pd.read_csv('weight-height.csv')
df.head()
```

Out[60]:

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

Correlation coefficient

$$r_{xy} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\Sigma(x_i - \bar{x})^2 \, \Sigma(y_i - \bar{y})^2}}$$

r_{xy} = correlation coefficient between x and y

x_i = the values of x within a sample

y_i = the values of y within a sample

$\bar{x}$ = the average of the values of x within a sample

$\bar{y}$ = the average of the values of y within a sample

In [61]:

```
df[['Height','Weight']].corr()
```

Out[61]:

	Height	Weight
Height	1.000000	0.924756
Weight	0.924756	1.000000

Pearson

In [62]:

```
from scipy.stats import pearsonr,spearmanr
Ho = 'No correlation'
Ha = 'There is correlation'
pearsonr(df['Height'],df['Weight'])
```

Disadvantage of Pearson

- It cannot determine the nonlinear relationship between variables.
- It cannot distinguish between dependent and independent variables.

In [63]:

```
## Spearman
Ho = 'No correlation'
Ha = 'There is correlation'
spearmanr(df['Height'],df['Weight'])
```

Out[63]:

```
SpearmanrResult(correlation=0.9257076644210767, pvalue=0.0)
```

Poisson Distribution

A poisson distribution is a discrete distribution. It gives the probablity of an event happening a certain number of times(k) within a given interval of time or space.The poisson distribution has only one parameter lambda.

Rate or lambda: Average number of events occuring in a given time interval.

Rules deciding Poisson Distribution

- 1 The problem that you are dealing with should be a counting problem (counting number of occurences)
- 2 Row independence
- 3 Rate at which events occur is independent of any occurancy
- 4 No simulataneous events

In [64]:
```
from scipy.stats import poisson,binom
'''
A city sees 3 accidents per day on average. No of accidents per day.
What is the probablity that there will be 5 accidents tommorrow?
'''
poisson.pmf(k = 5,mu = 3)
```

Out[64]:
```
0.10081881344492458
```

PMF Formula:

P(x = k) = (lambda **k + e**-lambda)/k!

In [65]:
```
'''
Shop is open for 8 hours a day , the avg no of customres in 8 hours is 74
What is the prob that in 2 hours , there will be at most 15 customers?
'''
```

Out[65]:
```
'\nShop is open for 8 hours a day , the avg no of customres in 8 hours is 74\nWhat is the
prob that in 2 hours , there will be at most 15 customers?\n'
```

In [66]:
```
rate = (74/8) * 2
poisson.cdf(mu = rate,k = 15)
```

Out[66]:
```
0.24902769151284776
```

In [67]:
```
'''
Probablity of atleast 7 customers in 2 hrs
'''
```

Out[67]:
```
'\nProbablity of atleast 7 customers in 2 hrs\n'
```

In [68]:
```
1-poisson.cdf(k = 6,mu = rate)
```

Out[68]:
```
0.9992622541111789
```

When n is large and p is small, np = mu ----> Binomial is very close to Poisson Distribution.

In [69]:
```
'''
```

```
There are 80 students in a kinder garden class.
Each one of them has 0.015 probablity of foregeting their lunch on
any given
day . What is the average or expected number of students who forgot
lunch in the class?
What is the probablity that exactly 3 of them will forget their lunch today?
'''
```

Out[69]:

```
'\nThere are 80 students in a kinder garden class. \nEach one of them has 0.015 probablity of foregeting their lunch on \nany given \nday . What is the average or expected number of students who forgot \nlunch in the class?\nWhat is the probablity that exactly 3 of them will forget their lunch today?\n'
```

In [70]:

```
rate = 80*0.015
poisson.pmf(k = 3,mu = rate )
```

Out[70]:

```
0.08674393303071422
```

Exponential Distribution

The exponential distribution that often concerns the amount of time until some specific event happens. It is a process in which events happen continuously and independently at a constant average rate. The exponenetial distribution has the key property of being memory less. It is memory less because, we don't need to remember the time when the process has started.

In [71]:

```
'''
240  message per hour on an avg, Assume it follows poisson dist
What is the avg no of messages in 30 seconds.
Prob of 1 message in the next 30 secs.
'''
```

Out[71]:

```
'\n240  message per hour on an avg, Assume it follows poisson dist\nWhat is the avg no of messages in 30 seconds.\nProb of 1 message in the next 30 secs.\n'
```

In [72]:

```
rate = (240/(60*60))*30
poisson.pmf(mu = rate,k = 1)
```

Out[72]:

```
0.2706705664732254
```

In [73]:

```
'''
Prob that there are 3 messages in 20 secs
'''
```

Out[73]:

```
'\nProb that there are 3 messages in 20 secs\n'
```

In [74]:

```
rate = (240/(60*60))*20
rate
```

Out[74]:

```
1.3333333333333333
```

In [75]:

```
poisson.pmf(mu = rate,k = 3)
```

Out[75]:

```
0.10413714098399081
```

In [76]:

```
'''
What is the average wait time between 2 messages?
'''
```

Out[76]: '\nWhat is the average wait time between 2 messages?\n'

In [77]:
```
scale = (60*60)/240
scale
```

Out[77]: 15.0

In [78]:
```
'''
240  message per hour on an avg, Assume it follows poisson dist
What is the probablity of waiting for more than 10 secs
for the next message
'''
```

Out[78]: '\n240 message per hour on an avg, Assume it follows poisson dist\nWhat is the probablity of waiting for more than 10 secs \nfor the next message\n'

In [79]:
```
scale = (60*60)/240
scale
```

Out[79]: 15.0

In [80]:
```
from scipy.stats import expon
1-expon.cdf(x = 10,scale = scale)
```

Out[80]: 0.513417119032592

In [81]:
```
'''
What is the probablity  for waiting for less than 6 seconds to
receive the message?
'''
```

Out[81]: '\nWhat is the probablity for waiting for less than 6 seconds to\nreceive the message?\n'

In [82]:
```
scale = (60*60)/240
expon.cdf(x = 6,scale = scale)
```

Out[82]: 0.3296799539643607

In [83]:
```
'''
Average 5 mins to debug, Assume poisson set up , Find the prob of
debugging in 4 to 5 mins.
'''
```

Out[83]: '\nAverage 5 mins to debug, Assume poisson set up , Find the prob of \ndebugging in 4 to 5 mins.\n'

In [84]:
```
expon.cdf(x = 5,scale = 5) - expon.cdf(x = 4,scale = 5)
```

Out[84]: 0.08144952294577923

In [85]:
```
'''
Find the prob of needing more than 6 mins to debug
'''
```
Out[85]:
```
'\nFind the prob of needing more than 6 mins to debug\n'
```
In [86]:
```
1-expon.cdf(x = 6,scale = 5)
```
Out[86]:
```
0.3011942119122022
```
In [87]:
```
'''
Given that you already spent 3 mins and not found
the bug, what is the probablity of needing more than 9 mins overall
'''
```
Out[87]:
```
'\nGiven that you already spent 3 mins and not found\nthe bug, what is the probablity of n
eeding more than 9 mins overall\n'
```
In [88]:
```
'''
P(T>9|T>3) = P(T>9 intersection T>3)/p(T>3)
           = P(T>9)/P(T>3)

'''
```
Out[88]:
```
'\nP(T>9|T>3) = P(T>9 intersection T>3)/p(T>3)\n           = P(T>9)/P(T>3)\n    \n'
```
In [89]:
```
(1-expon.cdf(x = 9,scale = 5))/(1-expon.cdf(x = 3,scale = 5))
```
Out[89]:
```
0.3011942119122021
```

Paired TTest

Paired Ttest is often used when we are interested in the difference between 2 variables.

'Before vs After'

In [125…
```
'''
The Zumba trainer claims that the new dance routine helps to reduce more weight of the cus
The weights of 8 people are recorded for before the Zumba training and after the Zumba tra

Test the trainer's claim with 90% confidence.
'''
```
Out[125…
```
"\nThe Zumba trainer claims that the new dance routine helps to reduce more weight of the
customers. \nThe weights of 8 people are recorded for before the Zumba training and after
the Zumba training for a month.\n\nTest the trainer's claim with 90% confidence.\n"
```
In [126…
```
from scipy.stats import ttest_rel
```
In [130…
```
Ho = 'same'
Ha = 'reduces'

wt_before=[85, 74, 63.5, 69.4, 71.6, 65,90,78]
wt_after=[82, 71, 64, 65.2, 67.8, 64.7,95,77]
```

```
tstat,pval = ttest_rel(wt_before,wt_after)
print(pval)
print(tstat)
if pval<0.10:
    print('Reject Ho')
else:
    print('Failed to reject ho')
```

```
0.29093617002652783
1.142185379355503
Failed to reject ho
```

In [131…

```
'''
You are appointed as a Data Analyst for a training program deployed by the Government of I
The participants’ skills were tested before and after the training using some metrics on a
'''
```

Out[131…

```
'\nYou are appointed as a Data Analyst for a training program deployed by the Government o
f India.\nThe participants’ skills were tested before and after the training using some me
trics on a scale of 10.\n'
```

In [140…

```
Ho = 'No Effect'
Ha = 'There is effect'
before = [2.45, 0.69, 1.80, 2.80, 0.07, 1.67, 2.93, 0.47, 1.45, 1.34]

after = [7.71, 2.17, 5.65, 8.79, 0.23, 5.23, 9.19, 1.49, 4.56, 4.20]
tstat,pval = ttest_rel(before,after,alternative = 'less')
print(tstat)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
-5.111096450191606
0.00031778119819482275
Reject Ho
```

Feature Engineering

- First I will determine whether the feature is dependent on target or not.
- Only if dependent than will i impute null values

Simple Imputer

In [210…

```
from sklearn.impute import SimpleImputer
```

In [211…

```
a = pd.DataFrame([10,10,20,10,30,10,np.nan,50])
a
```

Out[211…

	0
0	10.0
1	10.0
2	20.0
3	10.0

	0
4	30.0
5	10.0
6	NaN
7	50.0

In [212…

```
SimpleImputer(strategy = 'median').fit_transform(a)
```

Out[212…

```
array([[10.],
       [10.],
       [20.],
       [10.],
       [30.],
       [10.],
       [10.],
       [50.]])
```

In [213…

```
df = pd.read_csv('assignment1.txt')
df.head()
```

Out[213…

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanA
0	LP001002	Male	No	0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	

In [214…

```
df.isna().sum()
```

Out[214…

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [216…

```
col = ['LoanAmount','Loan_Amount_Term']
median_imputer = SimpleImputer(strategy = 'median')
for i in col:
    df[i] = pd.DataFrame(median_imputer.fit_transform(pd.DataFrame(df[i])))
```

In [217…

```
df.isna().sum()
```

Out[217…
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [218…
```
cat_missing = ['Gender','Married','Dependents']
freq_imputer = SimpleImputer(strategy = 'most_frequent')
for col in cat_missing:
    df[col] = pd.DataFrame(freq_imputer.fit_transform(pd.DataFrame(df[col])))
```

In [219…
```
df.isna().sum()
```

Out[219…
```
Loan_ID               0
Gender                0
Married               0
Dependents            0
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Label Encoder

For 2 categories

In [220…
```
from sklearn.preprocessing import LabelEncoder
```

In [222…
```
label_encoder = LabelEncoder()
df['Loan_Status'] = label_encoder.fit_transform(df['Loan_Status'])
```

In [223…
```
df['Loan_Status'].value_counts()
```

Out[223…
```
1    422
0    192
Name: Loan_Status, dtype: int64
```

Target_Encoder

In [224…
```
from category_encoders import TargetEncoder
```

In [226…

```
pd.crosstab(df['Property_Area'],df['Loan_Status'])
```

Out[226…

Loan_Status	**0**	**1**
Property_Area		
Rural	69	110
Semiurban	54	179
Urban	69	133

In [229…

```
te = TargetEncoder()
df['Property_Area'] = te.fit_transform(df['Property_Area'],df['Loan_Status'])
df['Property_Area'].value_counts()
```

Out[229…

```
0.768240    233
0.658416    202
0.614525    179
Name: Property_Area, dtype: int64
```

In [90]:

```
'''
        Ho True                              Ho False
Reject  False Poisitive / Type 1 error      True Negative
Accept  True Positive                        False Negative / Type 2 error
'''
```

Out[90]:

```
'\n        Ho True                              Ho False\nReject  False Poisitive / Type 1 er
ror     True Negative\nAccept  True Positive                        False Negative / Type 2 er
ror \n'
```

In [ ]:

```
Ho  = 'defendant is innocent'
Type 2 error = 'False Negative'
False Negative = Ho False, and we are accepting Ho
False positive  = Ho True , and we are rejecting Ho
```

In [91]:

```
from scipy.stats import chisquare,chi2_contingency
```

In [92]:

```
Ho = 'Independent'
Ha = 'Dependent'
a = [[67,213,74],[411,633,129],[85,51,7],[27,60,15]]
chi_stat,pval,df,ex = chi2_contingency(a)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
3.925170647869838e-18
Reject Ho
```

In [93]:

```
Ho = 'Independent'
Ha = 'Dependent'
a = [[33,218],[25,389],[20,393],[17,178]]
chistat,pval,df,ex = chi2_contingency(a)
print(pval)
if pval<0.01:
    print('reject Ho',Ha)
```

```
else:
    pritn('Failed to reject Ho',Ho)
```

```
0.000554511571355531
reject Ho Dependent
```

In [103…

```
act = [77.4,36.1,15.5]
obs = [73,38,18]
Ho = 'Consistent'
Ha = 'Not consistent'
chitest,pval = chisquare(obs,act)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.6861373156447124
Failed to reject Ho
```

In [104…

```
Ho = 'The coin is Fair'
Ha = 'The coin is not Fair'
exp = [50,50]
obs = [48,52]
ctest,pval = chisquare(obs,exp)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.6891565167793516
Failed to reject Ho
```

In [107…

```
0.4*200
```

Out[107…

```
80.0
```

In [109…

```
exp = [60,80,60]
obs = [70,80,50]
Ho = 'Matches'
Ha = 'Doesnot Matches'
chistat,pval = chisquare(obs,exp)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.1888756028375618
Failed to reject Ho
```

In [99]:

```
0.28*129
```

Out[99]:

```
36.120000000000005
```

In [101…

```
77.4+36.1+15.5
```

Out[101…

```
129.0
```

In [110…
```
from scipy.stats import ttest_1samp,ttest_ind
Ho = 'Same'
Ha = 'Different'
men = [13.3, 6.0, 20.0, 8.0, 14.0, 19.0, 18.0, 25.0, 16.0, 24.0, 15.0, 1.0, 15.0]
women = [22.0, 16.0, 21.7, 21.0, 30.0, 26.0, 12.0, 23.2, 28.0, 23.0]
tstat,pval = ttest_ind(men,women)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.010730607904197957
Reject Ho
```

In [111…
```
Ho = 'yield is same'
Ha = 'yield is higher'
x = [21.5, 24.5, 18.5, 17.2, 14.5, 23.2, 22.1, 20.5, 19.4, 18.1, 24.1, 18.5]
mu = 20
tstat,pval = ttest_1samp(x,20,alternative = 'greater')
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.4223145946526807
Failed to reject Ho
```

In [112…
```
Ho = 'Height is same as average'
Ha = 'Height is shorther than average'
mu = 65
std = 2.5
obs = 64.5
n = 20
std_error = std/np.sqrt(n)
z = (obs-mu)/std_error
pval = norm.cdf(z)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.18554668476134878
Failed to reject Ho
```

In [113…
```
std_error = 8.5/np.sqrt(10)
z = (155-150)/std_error
pval = 1-norm.cdf(z)
print(pval)
if pval<0.05:
    print('Reject Ho')
else:
    print('Failed to reject ho')
```

```
0.031431210741779014
Reject Ho
```

In [119…
```
Ho = 'average = 500'
Ha = 'average > 530'
std_error = 125/np.sqrt(70)
```

```
z = (530-500)/std_error
pval = norm.cdf(z)
print(pval,z)
```

```
0.9776775074187065 2.007984063681781
```

In [118…

```
norm.ppf(0.95)
```

Out[118…

```
1.6448536269514722
```

In [120…

```
norm.cdf(1.64)
```

Out[120…

```
0.9494974165258963
```

In [121…

```
std_error = 0.5/np.sqrt(30)
z = (1.85-1.7)/std_error
print(z)
```

```
1.6431676725155
```

In [123…

```
if z<1.96:
    print('Failed to reject Ho')
```

```
Failed to reject Ho
```

In [124…

```
std_error = 0.7/np.sqrt(45)
o=(1.28*std_error)+3.5
o
```

Out[124…

```
3.6335677938559874
```

In [142…

```
from scipy.stats import expon,poisson
```

In [154…

```
pval = expon.ppf(q = 0.75,scale = 4)
print(pval)
```

```
5.545177444479562
```

In [155…

```
expon.cdf(x = 5,scale = 5)-expon.cdf(x = 4,scale = 5)
```

Out[155…

```
0.08144952294577923
```

In [157…

```
scale = 1000
1-expon.cdf(scale = 1000,x = 600)
```

Out[157…

```
0.5488116360940265
```

In [158…

```
(1-expon.cdf(x = 9,scale = 6))/(1-expon.cdf(x = 5,scale = 6))
```

Out[158…

```
0.513417119032592
```

In [159…

```
1 - expon.cdf(x = 6,scale =2 )
```

Out[159... 0.04978706836786395

In [160...
```
expon.cdf(x = 6,scale = 2)
```
Out[160... 0.950212931632136

In [163...
```
poisson.cdf(k = 1,mu = 3/20)
```
Out[163... 0.9898141728888165

In [165...
```
rate = (240/3600)*30
poisson.pmf(k = 1,mu = rate)
```
Out[165... 0.2706705664732254

In [168...
```
rate = 74*(1/4)
poisson.cdf(k = 15,mu = rate)
```
Out[168... 0.24902769151284776

In [171...
```
1-poisson.cdf(k = 6,mu = rate)
```
Out[171... 0.9992622541111789

In [173...
```
poisson.pmf(k = 1,mu = (1/365)*499)
```
Out[173... 0.34839633781319934

In [189...
```
mu = 0.0002*15000
poisson.pmf(k = 0,mu= 0.0002*15000)
```
Out[189... 0.049787068367863944

In [180...
```
0.02*15000/12
```
Out[180... 25.0

In [190...
```
0.0002*15000
```
Out[190... 3.0

In [193...
```
73/3.5
```
Out[193... 20.857142857142858

In [194...
```
20.85**2
```
Out[194... 434.7225000000001

In [195…
```
from scipy.stats import f_oneway
```

In [198…
```
Ho = 'same'
Ha = 'different'
a = [13, 8, 11, 12, 11]
b = [15, 10, 16, 11, 13, 10]
c = [5, 11, 9, 5]
d = [8, 10, 6, 5, 7]
atest,pval = f_oneway(a,b,c,d)
print(pval)
if pval<0.05:
    print('reject Ho')
else:
    print('failed to reject Ho')
```

```
0.004930291920562857б
reject Ho
```

In [199…
```
one_star = [382, 391, 335, 368, 400, 372]
two_star = [560, 343, 512, 329, 391, 367]
three_star = [384, 458, 409, 309, 374, 459]
four_star = [325, 390, 304, 240, 306, 169]
five_star = [360, 298, 272, 368, 320, 326]
f_test,pval = f_oneway(one_star,two_star,three_star,four_star,five_star)
print(pval)
if pval<0.01:
    print('Reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.009362001936328837
Reject Ho
```

In [200…
```
45+50+33
```

Out[200…
```
128
```

In [201…
```
0.30*150
```

Out[201…
```
45.0
```

In [202…
```
0.40*150
```

Out[202…
```
60.0
```

In [204…
```
ex = [45,45]
g = [60,50]
a = [45,55]
ori = [45,50,55]
act = [45,60,45]
ftest,pval = f_oneway(ex,g,a)
print(pval)
if pval<0.05:
    print('reject Ho')
else:
    print('Failed to reject Ho')
```

```
0.3535533905932738
```

```
Failed to reject Ho
```

In [206…

```
df = pd.read_csv('assignment1.txt')
df
```

Out[206…

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loar
0	LP001002	Male	No	0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	
...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	

614 rows × 13 columns

In [207…

```
men = df.loc[(df['Gender']=='Male')&(df['Married']=='No'),['ApplicantIncome']]
women = df.loc[(df['Gender']=='Female'),['ApplicantIncome']]
```

In [208…

```
tstat,pval = ttest_ind(men,women)
print(pval)
if pval<0.05:
    print('reject Ho')
else:
    print('Failed to reject Ho')
```

```
[0.2552975]
Failed to reject Ho
```

Feature Engineering

- Feature engineering is a critical step in building accurate and effective machine learning models.
- One key aspect of feature engineering is scaling, normalization and standardization, which involves transforming the data to make it more suitable for modeling.

Feature Scaling

Feature scaling is a data preprocessing rechnique used to transform the values of features or variables in a dataset to a similar scale. The purpose is to ensure that all features contribute equally to the model and to avoid the domination of feature with larger values.

Feature scaling becomes necessary when dealing with datasets containing features that have different ranges, units of measurement, or orderss of magnitude. In such cases variation in feature values can lead to biased model performance or difficulties during the learning process.

Why should we use feature scaling

- Machine learning algos that use gradient descent as an optimization technique require data to be scaled. The difference in ranges of features will cause different step sizes for each feature.
- Distance algos like KNN, K-means clustering and SVM are most affected by the range of features. This is becuase, behind the scenes they are using distances between data points to determine their similarity.
- Tree based algos on the other hand are insensitive to the scale of features. A decision tree only splits a node based on a single feature. The decision tree splits a node on a feature that increases the homogenity of the node. Other features do not influence this split on a feature.

What is Normalization

Normalization is a data preprocessing technique used to adjust the values of features in a dataset to a common scale. This is done to facilitate data analysis and modeling, and to reduce the impact of different scales on the accuracy of machine learning models.

Normalization is a scaling technique in which values are shifted and resclaed so that they end up ranging between 0 and 1. It is also known as Min-Max scaling

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Xmax and Xmin are maximum and minimum values of the feature

What is Standardization

Standardization is another scaling method where the values are centerd around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero, and the resultant distribution has a unit standard deviation.

$$X' = \frac{X - \mu}{\sigma}$$

- mu is the mean
- sigma is the standard deviation fo the feature values

When to use Normalization or Standardization?

Normalization

- Rescales values to a range between 0 and 1
- Useful when the distribution of the data is unknown or not Gaussian
- Sensitive to outliers
- Retains the shaps of the original distribution
- May not preserve the relationship between the data points

Standardization

- Center data around the mean and scales to a standard deviation of 1.

- Useful when the distribution of the data is Gaussian or Unknown
- Less sensitive to outliers
- Changes the shape of the original distribution

- # Preserves the relationships between the data points

In [ ]: