```
### Thong Ke Nhieu Chieu - BT ca nhan chuong 4
# Nguyen Duc Vu Duy - 18110004
#Import thu vien
import numpy as np

print("Exercise 4.26: ")
print("a) ")
# x=np.array([[1,18.95],[2,19],[3,17.95],[3,15.54],[4,14],[5,12.95],[6,8.94],[8,7.49],[9,6],[11,3.99]])
# x=pd.DataFrame([[.48,40.53,2.19,.55,.74,.66,.93,.37,.22],[12.57,73.68,11.13,20.03,20.29,.78,4.64,.43,1.08]])
x=np.array([[.48,12.57],[2.19,11.13],[.55,20.03],[.74,20.29],[.66,.78],[.93,4.64],[.37,.43],[.22,1.08]])
x_1,x_2=np.array(x[:,0]),np.array(x[:,1])

x_1_mean=np.sum(x_1)/8
x_2_mean=np.sum(x_2)/8

x_mean=np.array([[x_1_mean],[x_2_mean]])

S=np.cov(x_1,x_2)
print('The Sample variance - covariance matrix S: ')
print(S)
print()

sq_distance=[]
for i in range(len(x)):
    x_0=x[i].reshape(2,1)
    x_left=np.dot((x_0-x_mean).T,np.linalg.pinv(S))
    x_result=np.dot(x_left,(x_0-x_mean))
    sq_distance.append(x_result)
    print('The squared statistical distances of x_'+str(i+1), ': ',x_result)
    print()
```

```
Exercise 4.26:
a)
The Sample variance - covariance matrix S:
[[ 0.37856429  1.03028214]
 [ 1.03028214 69.85975536]]

The squared statistical distances of x_1 :  [[0.51814443]]

The squared statistical distances of x_2 :  [[5.3838751]]

The squared statistical distances of x_3 :  [[2.18499937]]

The squared statistical distances of x_4 :  [[1.97289605]]

The squared statistical distances of x_5 :  [[0.936943]]

The squared statistical distances of x_6 :  [[0.39512842]]

The squared statistical distances of x_7 :  [[1.22453865]]
```

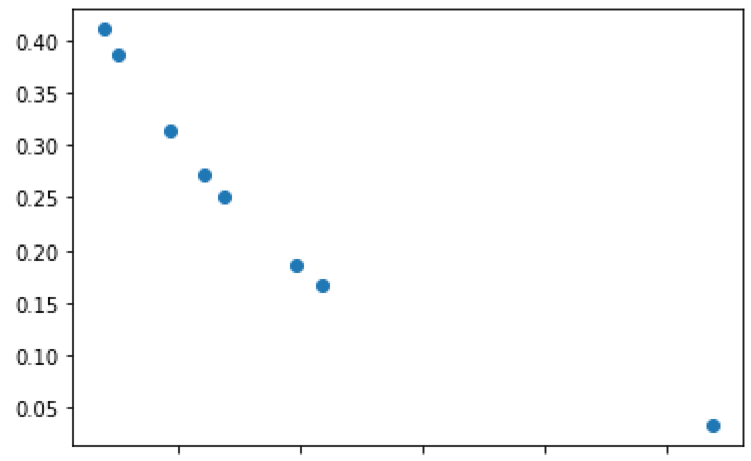
```
The squared statistical distances of x_8 :  [[1.38347497]]
```

```
print('b) ')
print('Squared Statistical distances of x_j in part a) are distributed as a chi square distribution')
print()
print('The observations falling within the estimated 50% probability contour of a bivariate normal distribution would have squared statisti
print()
print('the 50% percentile of chi square with 2 degree of freedom is 1.3863')
print()

for i in range(len(x)):
    if sq_distance[i]<1.3863:
        print('Observation x'+str(i+1)+' fall within the estimated 50%.')

print()
print('We have 5 among 10 observations. Thus, 50% of observations fall within the estimated 50%')
```

```
b)
Squared Statistical distances of x_j in part a) are distributed as a chi square distribution

The observations falling within the estimated 50% probability contour of a bivariate normal distribution would have squared statistical distance which is lower than the 50% perc

the 50% percentile of chi square with 2 degree of freedom is 1.3863

Observation x1 fall within the estimated 50%.
Observation x5 fall within the estimated 50%.
Observation x6 fall within the estimated 50%.
Observation x7 fall within the estimated 50%.
Observation x8 fall within the estimated 50%.

We have 5 among 10 observations. Thus, 50% of observations fall within the estimated 50%
```

```
print('c) ')
print('The ordered distances in part a) ')
sq_distance_copy=sq_distance.copy()
sq_distance_order=sorted(sq_distance_copy)
print(sq_distance_order)
print()

#Import thu vien
from scipy.stats import chi2
import matplotlib.pyplot as plt

print('The chi square plot')
sq_distance_order_array=np.array(sq_distance_order).reshape(-1)
plt.plot(sq_distance_order_array, chi2.pdf(sq_distance_order_array, 2),'o')
plt.show()

```

```
c)
The ordered distances in part a)
[array([[0.39512842]]), array([[0.51814443]]), array([[0.936943]]), array([[1.22453865]]), array([[1.38347497]]), array([[1.97289605]]), array([[2.18499937]]), array([[5.383875]

The chi square plot
```




```
print('d) ')
print(' From part b, 50% of the observations fall within the estimated 50% probability contour of a bivariate normal distribution')
print()
print('Also, from part c, the line is nearly linear')
print()
print('Therefore, the data comes from a bivariate normal distribution')
```

```
d)
 From part b, 50% of the observations fall within the estimated 50% probability contour of a bivariate normal distribution

Also, from part c, the line is nearly linear

Therefore, the data comes from a bivariate normal distribution
```

```
print('Exercise 4.29')
data=np.array([[8,98,7,2,12,8,2],[7,107,4,3,9,5,3],[7,103,4,3,5,6,3],[10,88,5,2,8,15,4],[6,91,4,2,8,10,3],[8,90,5,2,12,12,4],[9,84,7,4,12,1

x_5,x_6=np.array(data[:,4]),np.array(data[:,5])

x_5_mean=np.sum(x_5)/10
x_6_mean=np.sum(x_6)/10

x_mean_56=np.array([[x_5_mean],[x_6_mean]])

S_56=np.cov(x_5,x_6)
print('The Sample variance - covariance matrix S: ')
print(S_56)
print()

sq_distance_56=[]
for i in range(len(data)):
    data_0=np.array([[data[i,4]],[data[i,5]]]).reshape(2,1)
    data_left=np.dot((data_0-x_mean_56).T,np.linalg.pinv(S_56))
    data_result=np.dot(data_left,(data_0-x_mean_56))
```

```
    sq_distance_56.append(data_result)
    print('The squared statistical distances of x_'+str(i+1), ': ',data_result)
```

```
Exercise 4.29
The Sample variance - covariance matrix S:
[[11.36353078  3.12659698]
 [ 3.12659698 30.97851336]]

The squared statistical distances of x_1 :  [[98.11679939]]
The squared statistical distances of x_2 :  [[118.36034603]]
The squared statistical distances of x_3 :  [[139.74972261]]
The squared statistical distances of x_4 :  [[110.48985158]]
The squared statistical distances of x_5 :  [[116.33019929]]
The squared statistical distances of x_6 :  [[92.48813839]]
The squared statistical distances of x_7 :  [[88.96389426]]
The squared statistical distances of x_8 :  [[52.39345208]]
The squared statistical distances of x_9 :  [[98.83255908]]
The squared statistical distances of x_10 :  [[91.79073439]]
The squared statistical distances of x_11 :  [[116.60915696]]
The squared statistical distances of x_12 :  [[99.68997693]]
The squared statistical distances of x_13 :  [[68.85971073]]
The squared statistical distances of x_14 :  [[104.65375632]]
The squared statistical distances of x_15 :  [[116.33019929]]
The squared statistical distances of x_16 :  [[110.76852113]]
The squared statistical distances of x_17 :  [[126.31916125]]
The squared statistical distances of x_18 :  [[86.98235906]]
The squared statistical distances of x_19 :  [[103.86054199]]
The squared statistical distances of x_20 :  [[118.36034603]]
The squared statistical distances of x_21 :  [[95.53309159]]
The squared statistical distances of x_22 :  [[127.86738944]]
The squared statistical distances of x_23 :  [[88.93986219]]
The squared statistical distances of x_24 :  [[146.46051769]]
The squared statistical distances of x_25 :  [[93.18098112]]
The squared statistical distances of x_26 :  [[127.86738944]]
The squared statistical distances of x_27 :  [[98.83255908]]
The squared statistical distances of x_28 :  [[122.07290619]]
The squared statistical distances of x_29 :  [[113.60603634]]
The squared statistical distances of x_30 :  [[134.72435136]]
The squared statistical distances of x_31 :  [[109.79856446]]
The squared statistical distances of x_32 :  [[98.11679939]]
The squared statistical distances of x_33 :  [[125.5299788]]
The squared statistical distances of x_34 :  [[92.70682395]]
The squared statistical distances of x_35 :  [[129.32738441]]
The squared statistical distances of x_36 :  [[110.76852113]]
The squared statistical distances of x_37 :  [[125.5343714]]
The squared statistical distances of x_38 :  [[81.05356433]]
The squared statistical distances of x_39 :  [[97.95378394]]
The squared statistical distances of x_40 :  [[122.19776596]]
The squared statistical distances of x_41 :  [[95.16391276]]
The squared statistical distances of x_42 :  [[135.31440391]]
```

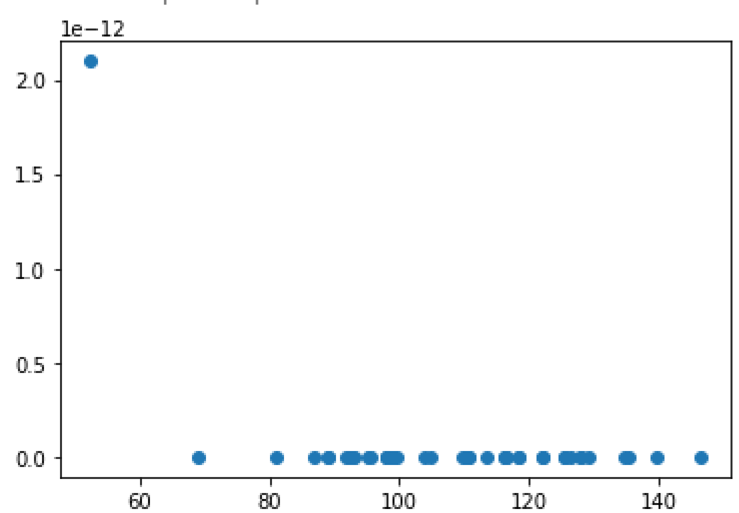
```
print('b) ')
print('Squared Statistical distances of x_j in part a) are distributed as a chi square distribution')
print()
print('The observations falling within the estimated 50% probability contour of a bivariate normal distribution would have squared statisti
print()
```

```
print('the 50% percentile of chi square with 7 degree of freedom is 6.346')
print()

for i in range(len(data)):
    if sq_distance_56[i]<6.346:
        print('Observation x'+str(i+1)+' fall within the estimated 50%.')

print()
print('We have 0 among 42 observations. Thus, 0% of observations fall within the estimated 50%')
```

```
b)
Squared Statistical distances of x_j in part a) are distributed as a chi square distribution

The observations falling within the estimated 50% probability contour of a bivariate normal distribution would have squared statistical distance which is lower than the 50% perc

the 50% percentile of chi square with 7 degree of freedom is 6.346


We have 0 among 42 observations. Thus, 0% of observations fall within the estimated 50%
```

```
print('c) ')
print('The ordered distances in part a) ')
sq_distance_copy_56=sq_distance_56.copy()
sq_distance_order_56=sorted(sq_distance_copy_56)
print(sq_distance_order_56)
print('The chi square plot')
sq_distance_order_array_56=np.array(sq_distance_order_56).reshape(-1)
plt.plot(sq_distance_order_array_56, chi2.pdf(sq_distance_order_array_56, 2),'o')
plt.show()
```

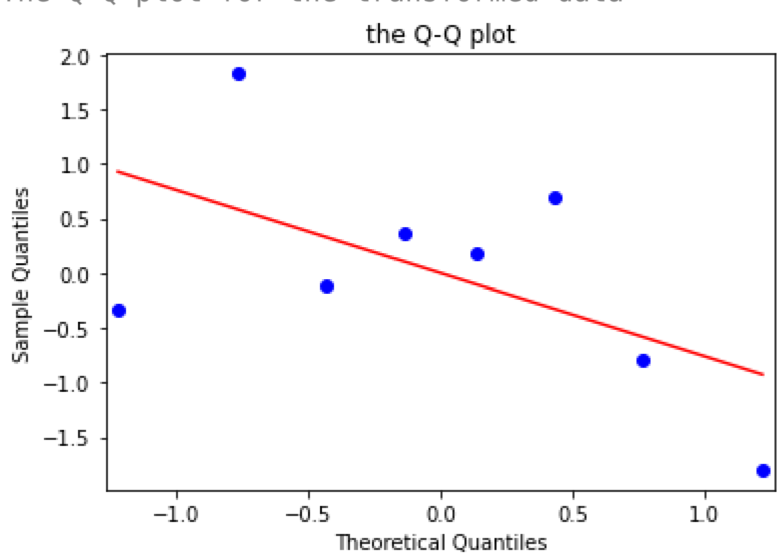
```
c)
The ordered distances in part a)
[array([[52.39345208]]), array([[68.85971073]]), array([[81.05356433]]), array([[86.98235906]]), array([[88.93986219]]), array([[88.96389426]]), array([[91.79073439]]), array([[
The chi square plot
```




```
print('Exercise 4.30')
```

```
print('a) ')

from sklearn.preprocessing import PowerTransformer
pt_1=PowerTransformer('box-cox')
pt_1.fit(x_1.reshape(-1,1))
print('The power transformation lambda_1 that makes the x_1 values approximately normal: ')
print(pt_1.lambdas_)
print()

transform_x_1=pt_1.transform(x_1.reshape(-1,1))

import statsmodels.api as sm
print('The Q-Q plot for the transformed data')
sm.qqplot(transform_x_1,line ='r')
plt.title('the Q-Q plot')
plt.show()
```

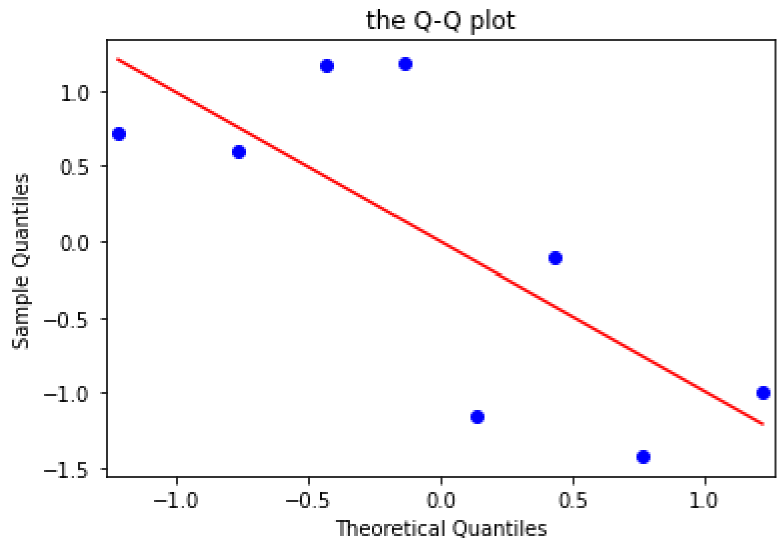
```
Exercise 4.30
a)
The power transformation lambda_1 that makes the x_1 values approximately normal:
[-0.23626729]

/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing
  import pandas.util.testing as tm
The Q-Q plot for the transformed data
```




```
print('b) ')
pt_2=PowerTransformer('box-cox')
pt_2.fit(x_2.reshape(-1,1))
print('The power transformation lambda_2 that makes the x_2 values approximately normal: ')
print(pt_2.lambdas_)
print()

transform_x_2=pt_2.transform(x_2.reshape(-1,1))

print('The Q-Q plot for the transformed data')
```

```
sm.qqplot(transform_x_2,line='r')
plt.title('the Q-Q plot')
plt.show()
```

```
b)
The power transformation lambda_2 that makes the x_2 values approximately normal:
[0.23805971]

The Q-Q plot for the transformed data
```




```
print('c) ')
pt=PowerTransformer('box-cox')
pt.fit(x)
print('The power transformation lambda that makes the x_1, x_2 values jointly normal: ')
print(pt.lambdas_)
print()

print('The results are the same as those obtained in part a and b')

```

```
c)
The power transformation lambda that makes the x_1, x_2 values jointly normal:
[-0.23626729  0.23805971]

The results are the same as those obtained in part a and b
```

Nhập dữ liệu

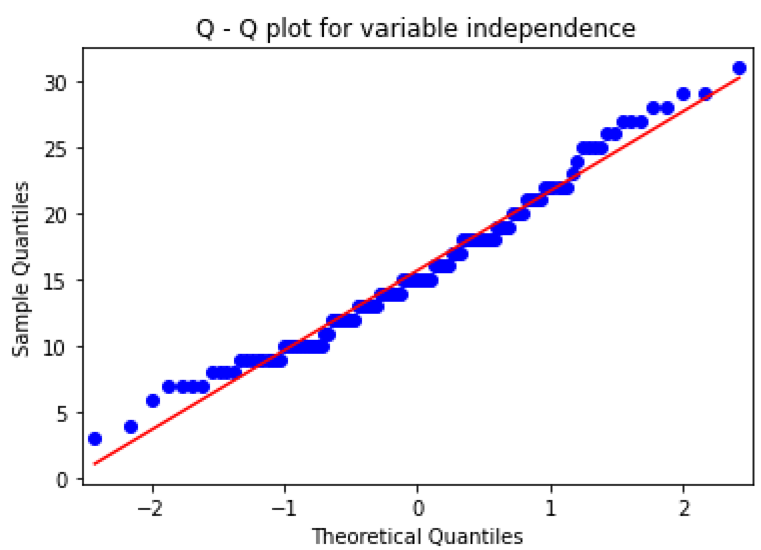
[ ] ↳ 1 cell hidden

Bài 4.39

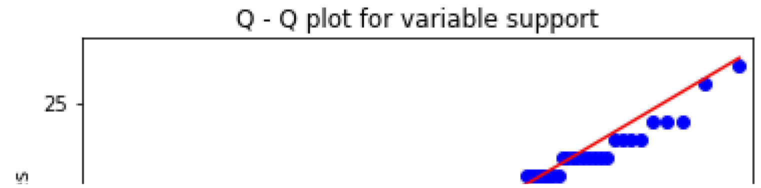
```
print('Exercise 4.39')
print('a) ')
```

```
dt_new=dt[:,0:5]
v_0,v_1,v_2,v_3,v_4=np.array([dt_new[:,0]]),np.array([dt_new[:,1]]),np.array([dt_new[:,2]]),np.array([dt_new[:,3]]),np.array([dt_new[:,4]])

v_0_new=v_0.reshape(130,1)
v_1_new=v_1.reshape(130,1)
v_2_new=v_2.reshape(130,1)
v_3_new=v_3.reshape(130,1)
v_4_new=v_4.reshape(130,1)

v_0_sort=sorted(v_0_new.copy())
v_1_sort=sorted(v_1_new.copy())
v_2_sort=sorted(v_2_new.copy())
v_3_sort=sorted(v_3_new.copy())
v_4_sort=sorted(v_4_new.copy())
sm.qqplot(np.array([v_0_sort]).reshape(-1,1),line='r')
plt.title('Q - Q plot for variable independence')
plt.show()

```

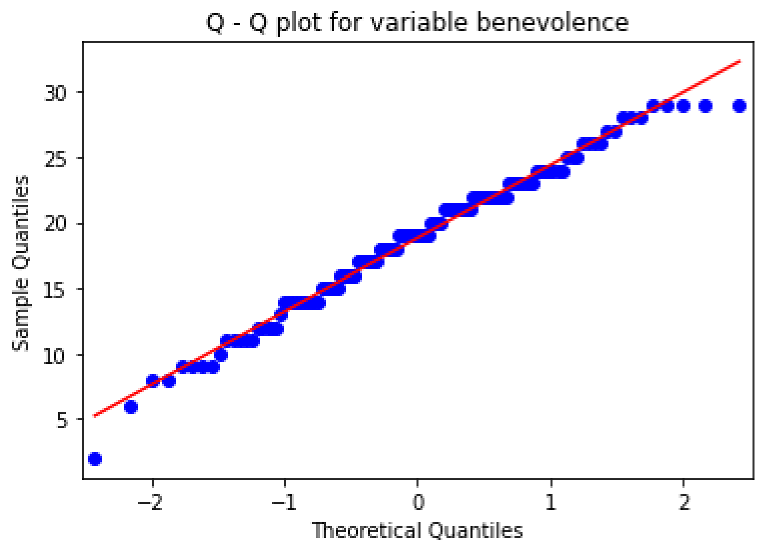
```
Exercise 4.39
a)
```



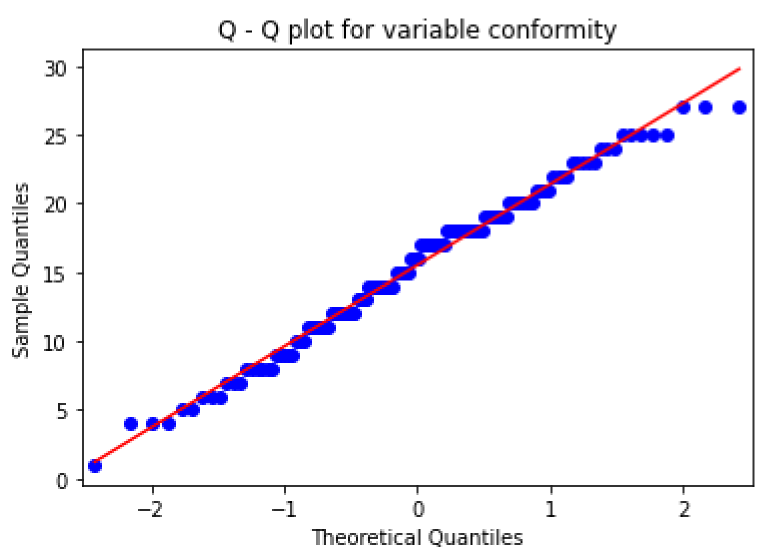

```
sm.qqplot(np.array([v_1_sort]).reshape(-1,1),line='r')
plt.title('Q - Q plot for variable support')
plt.show()
```



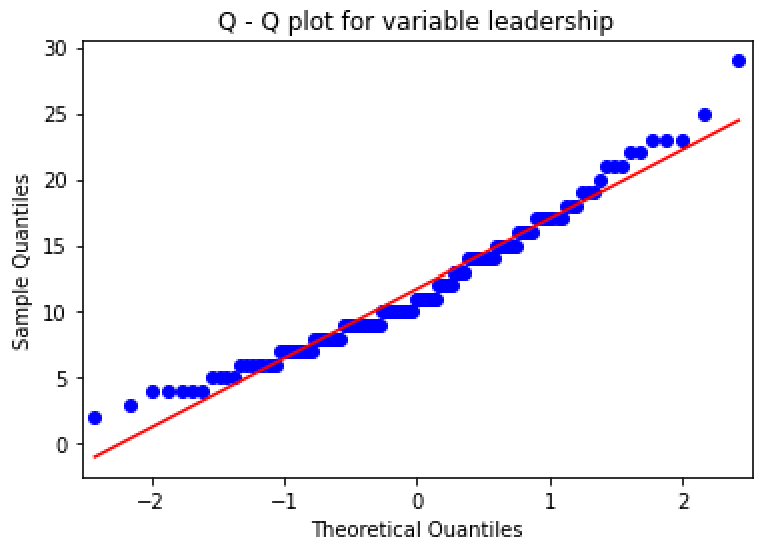

```
sm.qqplot(np.array([v_2_sort]).reshape(-1,1),line='r')
plt.title('Q - Q plot for variable benevolence')
plt.show()
```




```
sm.qqplot(np.array([v_3_sort]).reshape(-1,1),line='r')
plt.title('Q - Q plot for variable conformity')
plt.show()
```




```
sm.qqplot(np.array([v_4_sort]).reshape(-1,1),line='r')
plt.title('Q - Q plot for variable leadership')
plt.show()
```



Independence, support, leadership are the variables that are nonnormal

```
print('b) ')

v_0_mean=np.sum(v_0)/130
v_1_mean=np.sum(v_1)/130
v_2_mean=np.sum(v_2)/130
v_3_mean=np.sum(v_3)/130
v_4_mean=np.sum(v_4)/130

v_mean=np.array([[v_0_mean],[v_1_mean],[v_2_mean],[v_3_mean],[v_4_mean]])
data_cov = np.vstack([v_0,v_1,v_2,v_3,v_4])
S_new = np.cov(data_cov)
print('The Sample variance - covariance matrix S: ')
print(S_new)

sq_distance_new=[]
for i in range(len(dt)):
    x_0=dt[i,0:5].reshape(5,1)
    x_left=np.dot((x_0-v_mean).T,np.linalg.pinv(S_new))
    x_result=np.dot(x_left,(x_0-v_mean))
    sq_distance_new.append(x_result)

print()
print('the 50% percentile of chi square with 5 degree of freedom is 4.3515')
print()

count=0
for i in range(len(dt)):
    if sq_distance_new[i]<4.3515:
        count+=1

print('There are '+str(count/130)+ ' percentage of observations whose squared distances less than 4.3515')
print('Therefore, all five variables do not have multivariate normality')
```

```
b)
The Sample variance - covariance matrix S:
[[ 34.75020871  -4.27668456 -18.07179487 -15.97286822   5.71645796]
 [ -4.27668456  17.51341682   0.41979726  -7.86821705  -8.72331544]
 [-18.07179487   0.41979726  29.84472272   9.34883721 -13.94215862]
 [-15.97286822  -7.86821705   9.34883721  33.04263566  -9.94186047]
 [  5.71645796  -8.72331544 -13.94215862  -9.94186047  26.95796064]]

the 50% percentile of chi square with 5 degree of freedom is 4.3515

There are 0.49230769230769234 percentage of observations whose squared distances less than 4.3515
Therefore, all five variables do not have multivariate normality
```

```
print('c) ')
pt_new_0=PowerTransformer('box-cox')
pt_new_0.fit(v_0.reshape(-1,1))
transform_v_0=pt_new_0.transform(v_0.reshape(-1,1))
print('The transformation to make independence variable normal: ',pt_new_0.lambdas_)
print()

pt_new_1=PowerTransformer('box-cox')
pt_new_1.fit(v_1.reshape(-1,1))
transform_v_1=pt_new_1.transform(v_1.reshape(-1,1))
print('The transformation to make support variable normal: ',pt_new_1.lambdas_)
print()

pt_new_4=PowerTransformer('box-cox')
pt_new_4.fit(v_4.reshape(-1,1))
transform_v_4=pt_new_4.transform(v_4.reshape(-1,1))
print('The transformation to make leadership variable normal: ',pt_new_4.lambdas_)
print()
```

```
c)
The transformation to make independence variable normal:  [0.52377241]

The transformation to make support variable normal:  [1.39626145]

The transformation to make leadership variable normal:  [0.38154699]
```

✓ 0s completed at 11:44 PM