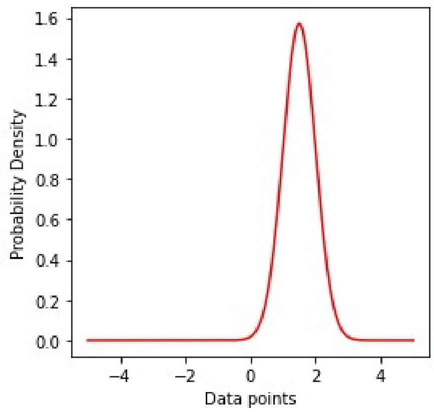
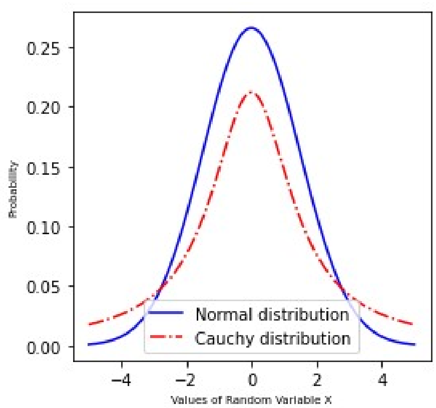
```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: from scipy import stats
   ...: import numpy as np
   ...: import pandas as pd
   ...: import matplotlib.pyplot as plt
   ...: import astroML
   ...: from numpy import random
   ...: import scipy
   ...: import statistics
   ...: from scipy.stats import skew
   ...: from scipy.stats import kurtosis

In [2]: x = stats.norm.rvs(loc=1.5,scale=0.5, size=1000)

In [3]: sample_mean=np.mean(x)

In [4]: sample_mean
Out[4]: 1.5008712066383845

In [5]: import statistics
   ...: variance=statistics.variance(x)
   ...: variance
Out[5]: 0.2554784340991375

In [6]: standard_deviation=statistics.stdev(x)
   ...: standard_deviation
Out[6]: 0.5054487452740758

In [7]: skewness=skew(x)
   ...: skewness
Out[7]: 0.07875666683918563

In [8]: kurtosis(x)
Out[8]: -0.2731067907833462

In [9]: median=np.median(x)
   ...: median
Out[9]: 1.4927138592154168

In [10]: MAD=scipy.stats.median_abs_deviation(x)

In [11]: MAD
Out[11]: 0.35842219572770695

In [12]: sd_G=1.482*MAD
    ...: print(sd_G)
0.5311816940684617

In [13]: x = np.linspace(-5,5,1000)
    ...:
    ...: #Creating a Function.
```

```
   ...: def normal_dist(x , mean , sd):
   ...:     prob_density = (np.pi*sd) * np.exp(-0.5*((x-mean)/sd)**2)
   ...:     return prob_density
   ...:
   ...: #Calculate mean and Standard deviation.
   ...: mean = np.mean(x)
   ...: sd = np.std(x)
   ...:
   ...: #Apply function to the data.
   ...: mean = 1.5
   ...: sd = 0.5
   ...: pdf = normal_dist(x,mean,sd)
   ...:
   ...: #Plotting the Results
   ...: plt.figure(figsize=(4,4))
   ...: plt.plot(x,pdf , color = 'red')
   ...: plt.xlabel('Data points')
   ...: plt.ylabel('Probability Density')
   ...:
   ...:
Out[13]: Text(0, 0.5, 'Probability Density')
```



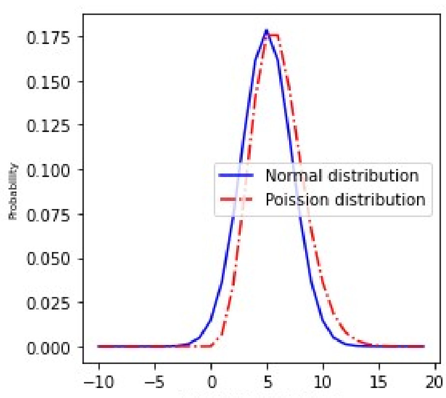

```
In [14]:
```

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: import numpy as np
   ...: import pandas as pd
   ...: from scipy import stats
   ...: import matplotlib.pyplot as plt
   ...: from scipy.stats import cauchy
   ...: from scipy.stats import norm

In [2]: mu,std = 0,1.5
   ...: x = np.linspace(-5,5,10000)
   ...: sndn = stats.norm(mu, std)
   ...:
   ...: # Defining mean and standard_deviation of the Cauchy distribution
   ...: muc,stdc = 0,1.5
   ...: sndc = cauchy(muc,stdc)
   ...:
   ...: #Plotting the figure
   ...: plt.figure(figsize=(4,4))
   ...: plt.plot(x, sndn.pdf(x),linestyle='solid',color='blue')
   ...: plt.plot(x, sndc.pdf(x),linestyle='-.',color='red')
   ...:
   ...: #labelling
   ...: plt.xlabel('Values of Random Variable X', fontsize='7')
   ...: plt.ylabel('Probability', fontsize='7')
   ...: plt.legend(['Normal distribution','Cauchy distribution'])
   ...: plt.show()
```




```
In [3]:
```

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: from scipy import stats
   ...: import numpy as np
   ...: import pandas as pd
   ...: import matplotlib.pyplot as plt
   ...: from scipy.stats import norm
   ...: from scipy.stats import poisson
   ...: import math

In [2]: mun,stdn = 5,math.sqrt(5)
   ...: x = np.arange(-10, 20, 1)
   ...: sndn = stats.norm(mun, stdn)
   ...:
   ...: # Defining mean of the Poission distribution
   ...: mup =5
   ...: sndp = stats.poisson(mup,1)
   ...:
   ...: #Plotting the fig
   ...: plt.figure(figsize=(4,4))
   ...: plt.plot(x, sndn.pdf(x),linestyle='solid',color='blue')
   ...: plt.plot(x, sndp.pmf(x),linestyle='-.',color='red')
   ...:
   ...: #Labelling
   ...: plt.xlabel('Values of Random Variable X', fontsize='7')
   ...: plt.ylabel('Probability', fontsize='7')
   ...: plt.legend(['Normal distribution','Poission distribution'])
   ...: plt.show()
```

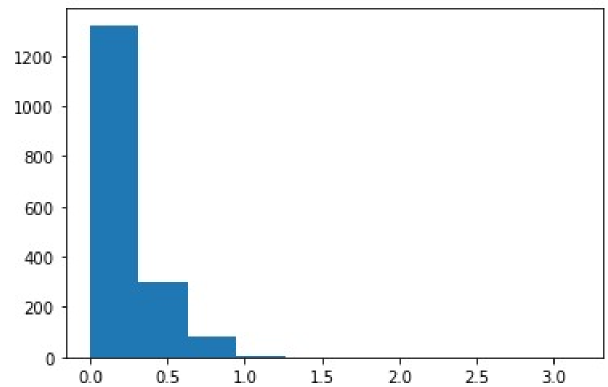



```
In [3]:
```

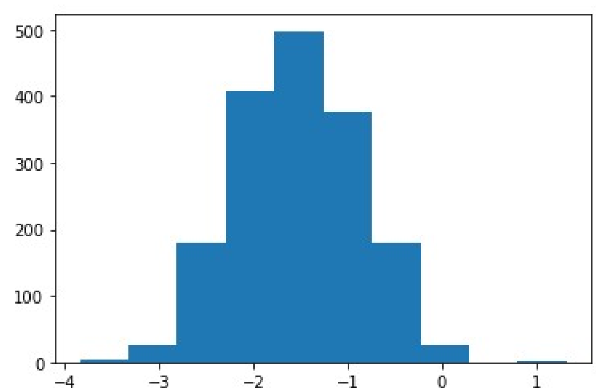
```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: from scipy import stats
   ...: import numpy as np
   ...: import pandas as pd
   ...: import matplotlib.pyplot as plt
   ...: import astroML

In [2]: d1=0.8920
   ...: sigma1=0.00044
   ...:
   ...: d2=0.881
   ...: sigma2=0.009
   ...:
   ...: d3=0.8913
   ...: sigma3=0.00032
   ...:
   ...: d4=0.9837
   ...: sigma4=0.00048
   ...:
   ...: d5=0.8958
   ...: sigma5=0.00045
   ...: x=[d1,d2,d3,d4,d5]
   ...: sigma=[sigma1,sigma2,sigma3,sigma4,sigma5]

In [3]: num=0
   ...: dek=0
   ...: for i in range(0,len(x)):
   ...:     num+=(x[i]/sigma[i]**2)
   ...:     dek+=(1/sigma[i]**2)
   ...:
   ...: mean=num/dek
   ...: print(mean)
0.9089185199574897

In [4]: uncertainity=((1/dek)**0.5)
   ...: print(uncertainity)
0.00020318737026848627

In [5]:
```

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: from scipy import stats
   ...: import numpy as np
   ...: import pandas as pd
   ...: import matplotlib.pyplot as plt
   ...: import astroML
   ...: from scipy.stats import boxcox
   ...: from scipy.special import boxcox
   ...: import seaborn as sns

In [2]: data=pd.read_csv("C:\\Users\\et22m\\Downloads\\exoplanet.eu_catalog.csv")

In [3]: data
Out[3]:
         # name  ...                            star_alternate_names
0      11 Com b  ...                                             NaN
1      11 Oph b  ...                            Oph 1622-2405, Oph 11A
2      11 UMi b  ...                                             NaN
3      14 And b  ...                                             NaN
4      14 Her b  ...                                             NaN
...         ...  ...                                             ...
5298  ups And c  ...                                             NaN
5299  ups And d  ...                                             NaN
5300  ups And e  ...                                             NaN
5301  ups Leo b  ...                                             NaN
5302  zet Del B  ...  HD 196180, HIP 101589, 2MASS J20351852+1440272

[5303 rows x 98 columns]

In [4]: data.shape
Out[4]: (5303, 98)

In [5]: print("Single column value using Dataframe[]")
   ...: print(data['eccentricity'])
Single column value using Dataframe[]
0       0.23100
1           NaN
2       0.08000
3       0.00000
4       0.36900
         ...
5298    0.24450
5299    0.31600
5300    0.00536
5301    0.32000
5302        NaN
Name: eccentricity, Length: 5303, dtype: float64

In [6]: e=data['eccentricity']

In [7]: e.shape
Out[7]: (5303,)

In [8]: e=e.dropna()
   ...: e.shape
Out[8]: (2145,)
```

```
In [9]: e_new=np.array(e)
   ...: e_new
   ...: e.shape
Out[9]: (2145,)

In [10]: e_new[e_new < 0] = 0
    ...: print("New Array :")
    ...: e_new
New Array :
Out[10]: array([0.231  , 0.08   , 0.     , ..., 0.316  , 0.00536, 0.32   ])

In [11]: e_new=e_new[e_new != 0]

In [12]: e_new
Out[12]: array([0.231  , 0.08   , 0.369  , ..., 0.316  , 0.00536, 0.32   ])

In [13]: e_new.shape
Out[13]: (1704,)

In [14]: print(e_new)
[0.231   0.08    0.369   ... 0.316   0.00536 0.32   ]

In [15]: plt.hist(e_new)
    ...: plt.show()
```

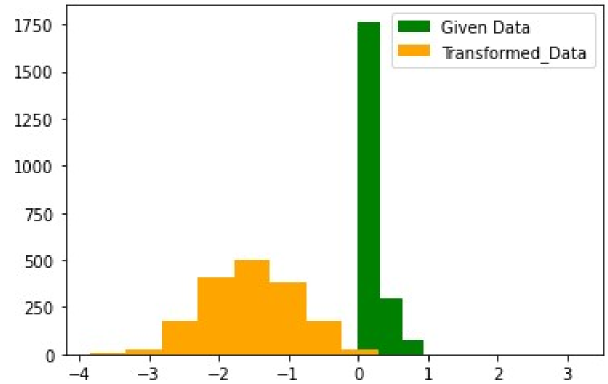



```
In [16]: fit_data,_ = stats.boxcox(e_new)

In [17]: plt.hist(fit_data)
    ...: plt.show()
```




```
In [18]: plt.hist(e,color='green')
    ...: plt.hist(fit_data,color='orange')
```

```
...: plt.legend(['Given Data','Transformed_Data'])
...: plt.show()
```




```
In [19]:
```