Gholap Sourabh Somnath

assignment-5

February 20, 2023

[108]:
```
import numpy as np
import scipy.stats as stats
from scipy.stats import norm
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.mixture import GaussianMixture
```

Question 1

[4]:
```
dens = np.array([2.12, 2.71, 3.44, 2.76, 2.72, 0.96, 2.00, 3.26, 2.50, 1.20, 1.
↪62, 1.30, 1.96, 2.60, 1.30, 2.67, 4.40, 1.80, 4.90, 2.39, 1.62, 1.47, 0.89,␣
↪2.52, 1.21, 0.90, 0.80])

deviation = np.array([0.04,0.11,0.12,1.2,0.12,0.3,0.6,0.6,0.3,0.4,0.3,0, 0.34,0.
↪5,0.2,0.03,2.1,0.8,3.9,0.9,1.05,0.95,0.13,0.3,0.25,0.1,0.15])
```

Shapiro-Wilk test

[5]:
```
stat = stats.shapiro(dens)
stats_density = stat[0]
stats_p_value = stat[1]
print('Density using Shapiro-Wilk test is:',stats_density)
print('Its p value is : ', stats_p_value)
```

```
Density using Shapiro-Wilk test is: 0.9246721863746643
Its p value is :  0.051220282912254333
```

Statistics for log density

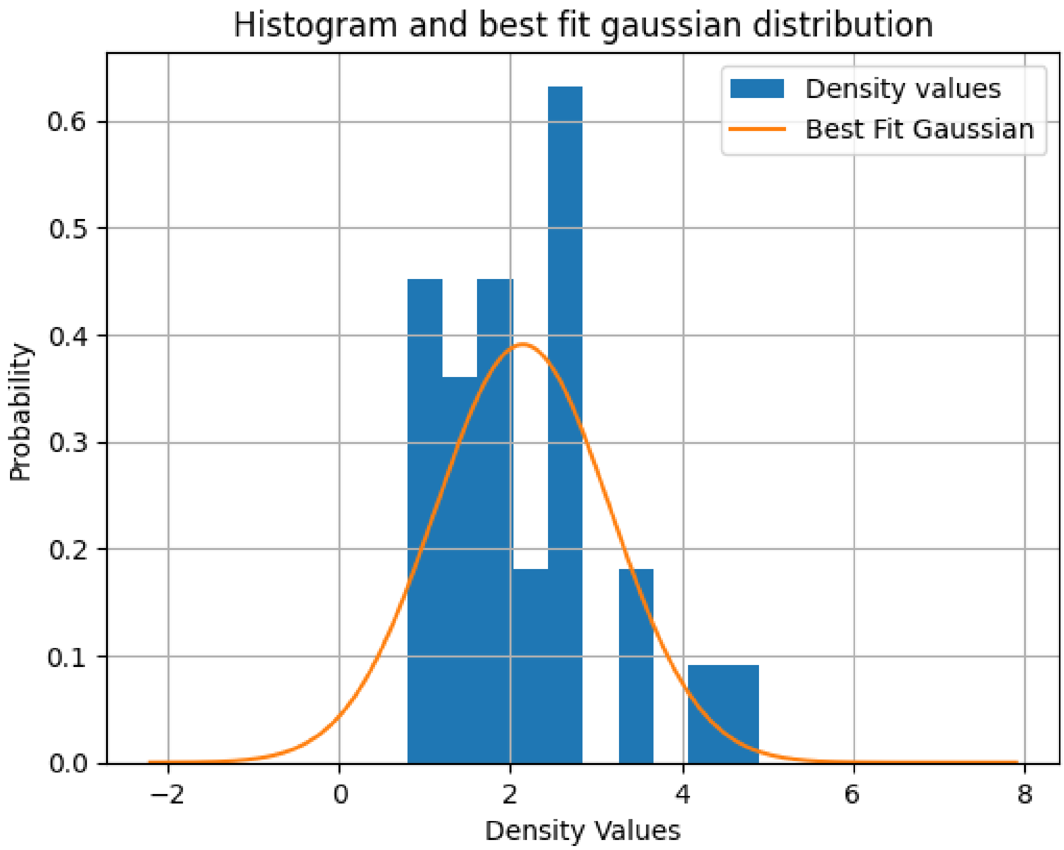
[6]:
```
stat_log = stats.shapiro(np.log(dens))
stats_density_log = stat_log[0]
stats_p_value_log = stat_log[1]
print('log Density using Shapiro-Wilk test is:',stats_density_log)
print('Its p value is : ', stats_p_value_log)
```

```
log Density using Shapiro-Wilk test is: 0.9686306715011597
Its p value is :  0.5660613775253296
```

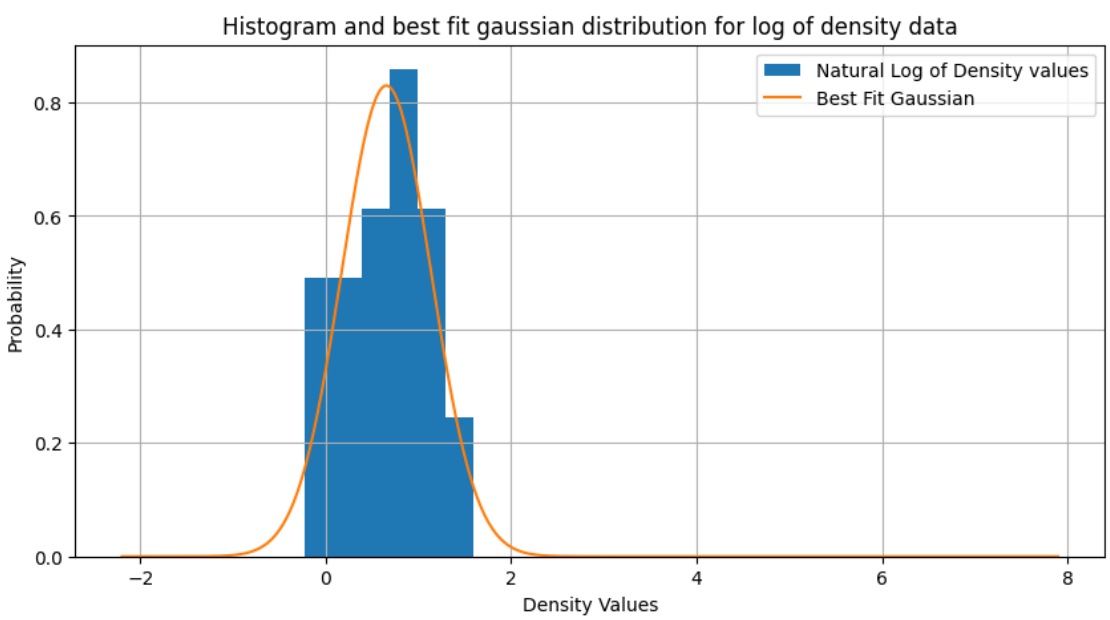
By comparing P values we come to know that log density more close to Gaussian distribution

[23]:
```
x_data = np.linspace(np.min(dens)-3,np.max(dens)+3,1000)

mean1,std1 = stats.norm.fit(dens)
gaussian_fit1 = stats.norm.pdf(x_data,mean1,std1)

mean2,std2 = stats.norm.fit(np.log(dens))
gaussian_fit2 = stats.norm.pdf(x_data,mean2,std2)
```

[24]:
```
plt.title("Histogram and best fit gaussian distribution")
plt.hist(dens, density = True, label = 'Density values')
plt.plot(x_data, gaussian_fit1, label = 'Best Fit Gaussian')
plt.xlabel("Density Values")
plt.ylabel("Probability")
plt.legend()
plt.grid()

fig = plt.figure(figsize = (10, 5))
plt.title("Histogram and best fit gaussian distribution for log of density␣
 ↪data")
plt.hist(np.log(dens), density = True, label = 'Natural Log of Density␣
 ↪values',bins = 'auto')
plt.plot(x_data, gaussian_fit2, label = 'Best Fit Gaussian')
plt.xlabel("Density Values")
plt.ylabel("Probability")
plt.legend()
plt.grid()
plt.show()
```



Histogram and best fit gaussian distribution
Density values
Best Fit Gaussian
Probability
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−2
0
2
4
6
8
Density Values



Histogram and best fit gaussian distribution for log of density data
Natural Log of Density values
Best Fit Gaussian
Probability
0.0
0.2
0.4
0.6
0.8
−2
0
2
4
6
8
Density Values

Here we can see that second histogram is much closer to gaussian distribution.

0.2 Question 2

[67]:
```
data2 = pd.read_csv('https://people.iith.ac.in/shantanud/HIP_star.dat',sep= ' ')
data2=pd.DataFrame(data2)
data2.head()
```

[67]:

	HIP	Vmag	RA	DE	Plx	pmRA	pmDE	e_Plx	B-V
0	2	9.27	0.003797	-19.498837	21.90	181.21	-0.93	3.10	0.999
1	38	8.65	0.111047	-79.061831	23.84	162.30	-62.40	0.78	0.778
2	47	10.78	0.135192	-56.835248	24.45	-44.21	-145.90	1.97	1.150
3	54	10.57	0.151656	17.968956	20.97	367.14	-19.49	1.71	1.030
4	74	9.93	0.221873	35.752722	24.22	157.73	-40.31	1.36	1.068

[68]:
```
hyades = []
non_hyades = []

for i in range(1,len(data)):
    RA = data2['RA'][i]
    DE = data2['DE'][i]
    pmRA = data2['pmRA'][i]
    pmDE = data2['pmDE'][i]
    B_V = data2['B-V'][i]

    if RA>=50 and RA<=100 and DE>=0 and DE<=25 and pmRA>=90 and pmRA<=130 and␣
 ↪pmDE>=-60 and pmDE<=-10:
        hyades.append(B_V)
    else:
        non_hyades.append(B_V)
```

[69]:
```
t, p = stats.ttest_ind(hyades, non_hyades, equal_var=True)
hyades_variance = np.var(hyades)
non_hyades_variance = np.var(non_hyades)
print(f"The t value is {t} and P value is {p}")
print(f"The variance of the hydes is {hyades_variance} \nVariance of non hyades␣
 ↪is {non_hyades_variance}")
```

```
The t value is -3.857407805640729 and P value is 0.00011725955837332216
The variance of the hydes is 0.10580084865302346
Variance of non hyades is 0.10778723438004537
```

0.3 Question 3

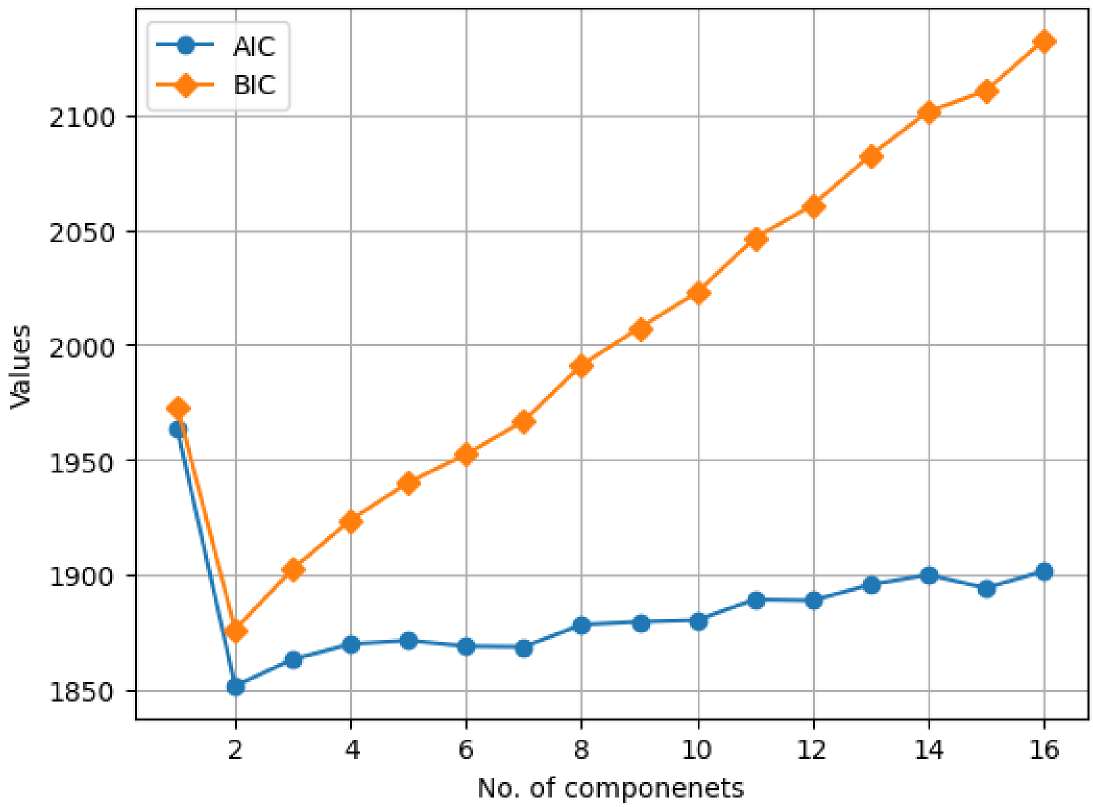
[109]:
```
data3 = pd.read_csv('A5_input.txt',sep= ' ')
data3 =pd.DataFrame(data3)
data3.head()
```

[109]:

	X
0	3.0
1	11.0
2	14.0
3	32.0
4	6.0

[120]:
```
x_data = data3['X']
x_data  = np.array(x_data)
x_log = np.log10(x_data)
x_log = x_log.reshape(-1,1)

AIC =[]
BIC = []

for i in range(1,17):
    model_before_fit  = GaussianMixture(n_components = i, covariance_type =
 ↪'full', max_iter = 10000)
    fitted_model = model_before_fit.fit(x_log)
    AIC.append(fitted_model.aic(x_log))
    BIC.append(fitted_model.bic(x_log))
```

[145]:
```
x = np.linspace(1,16,16).astype(int)
plt.plot(x,AIC,marker ='o')
plt.plot(x,BIC,marker = 'D')
plt.xlabel("No. of componenets")
plt.ylabel("Values")
plt.legend(['AIC', 'BIC'])
plt.grid()
plt.show()
```



The optimum number of components using AIC and BIC by plotting BIC as a function of number of componts are 2.