Data Science Analysis - Assignment 5

Varenya Upadhyaya

February 18, 2023

1. The following code plots Fig. (1) containing the histograms for the densities and log of densities of asteroids from the Penn State asteroid dataset. We perform the Shapiro-Wilk test on both the datasets to find that the logarithm dataset gives a distribution closer to the gaussian curve than the raw data. This is apparent from the outputted p-values from the test. The best fit gaussian curves are plotted along with the histograms

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import pandas as pd

df=pd.read_csv('asteroids.csv')
dens = df['Dens'].to_numpy()
logdens = np.log(dens)

#shapiro-wilk test
shap_dens = stats.shapiro(dens)
shap_logdens = stats.shapiro(logdens)

print('p-value for densities =',shap_dens.pvalue)
print('p-value for log of densities =',shap_logdens.pvalue)

#finding best fit gaussians
mu, std = stats.norm.fit(dens)
mu_log, std_log = stats.norm.fit(logdens)

#plotting everything
fig, ax = plt.subplots(1,2)
fig.set_figwidth(12)
fig.set_figheight(7)
fig.suptitle('Asteroid Densities')
x=np.linspace(-2, 6,10000)
ax[0].hist(dens, bins=7, density=True, label='Asteroid Densities',color='#698269')
ax[0].plot(x,stats.norm.pdf(x,mu,std),label= 'Best-fit gaussian',color='#B99B6B')
ax[0].legend()
ax[1].hist(logdens, bins=4, density=True, label='Log of Asteroid Densities',color
    ='#698269')
ax[1].plot(x,stats.norm.pdf(x,mu_log,std_log),label= 'Best-fit gaussian',color='#B99B6B
    ')
ax[1].legend()
plt.savefig('1.png')
```

The output of the code :

```
p-value for densities = 0.051220282912254333
p-value for log of densities = 0.5660613775253296
```

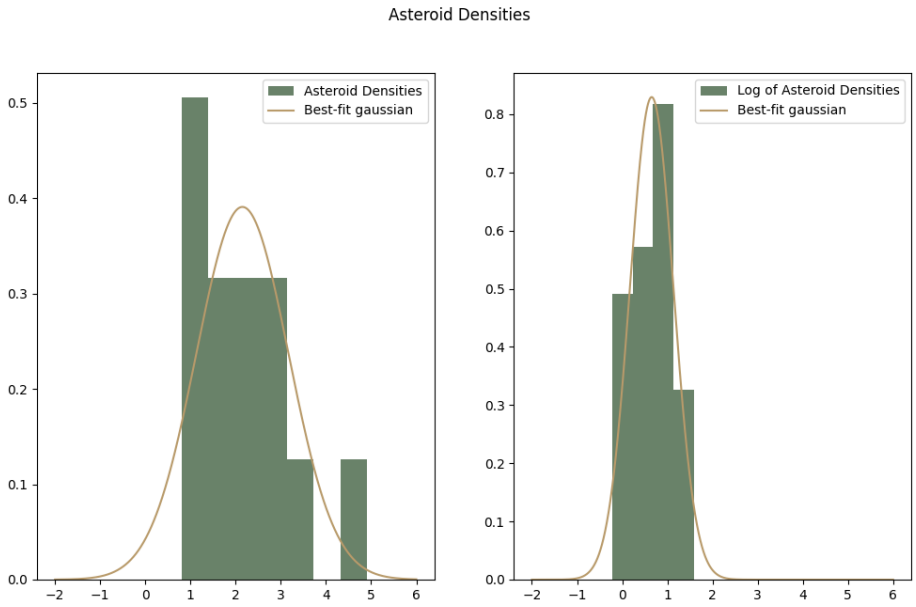

Figure 1: Histogram of the densities plotted along with the best fit gaussian curve

2. The p-value is calculated from the t-test performed on the colors of the Hyades and Non-Hyades stars in the following code. From the t-test we find a low p-value which supports the null hypothesis. The colors of the two types of stars are thus different.

```
import numpy as np
from scipy import stats

HIP, Vmag, RA, DE, Plx, pmRA, pmDE, e_Plx, BV=np.loadtxt('HIP_star.dat',skiprows=1,
    unpack=True)
BV_hyades,BV_nonhyades=[],[]

for i in range(len(HIP)):
    if(50<RA[i]<100 and 0<DE[i]<25 and 90<pmRA[i]<130 and -60<pmDE[i]<-10):
        BV_hyades.append(BV[i])
        continue
    BV_nonhyades.append(BV[i])
t = stats.ttest_ind(BV_hyades, BV_nonhyades)
print('p-value =',t.pvalue)
```

Code output:

```
p-value = 0.00011582222192442334
```

3. The following code performs the GMM on the logarithm of the T90 dataset.

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import stats
from sklearn.mixture import GaussianMixture

n_comp = np.arange(1,11)
t90 = np.loadtxt('beppoSax.txt',unpack=True)
data =np.log10(t90)
aic,bic,models=([] for _ in range(3))
for n in n_comp:
    gmm = GaussianMixture(n_components = n)
    X=np.expand_dims(data,1)
    gmm = gmm.fit(X)
    models.append(gmm)
```

```
    aic.append(gmm.aic(X))
    bic.append(gmm.bic(X))
print('Number of components =',n_comp)
print("AIC =",[ round(elem, 2) for elem in aic ])
print("BIC =",[ round(elem, 2) for elem in bic ])
print('opitimum n =',n_comp[np.argmin(aic)])

i_plot=[1,4,8]
for i in i_plot:
    x = np.linspace(-2,4,1000)
    y = np.exp(models[i].score_samples(x.reshape(-1,1)))
    plt.figure(figsize=(7,10))
    plt.plot(x, y, color="#B99B6B", lw=3, label='GMM')
    plt.hist(data, bins=20, density=True,color='#698269')
    plt.title('GMM for {} components'.format(n_comp[i]))
    plt.ylabel('N')
    plt.xlabel('log$_{10}$(T90)')
    plt.legend()
    plt.savefig('3gmm_{}'.format(str(i+1))+'.png')

#plotting the aic and bic values
i=np.argmin(aic)
plt.figure(figsize=(9,7))
plt.plot(n_comp,aic,label='AIC',ls='--',color='#B99B6B',lw=3)
plt.plot(n_comp,bic,label='BIC',color='#B99B6B',lw=3)
plt.scatter(i+1,bic[i],c='#B99B6B',s=50,)
plt.scatter(i+1,aic[i],c='#B99B6B',s=50)
plt.ylim(1840,2030)
plt.xlabel('number of components')
plt.ylabel('AIC/BIC values')
plt.title('AIC and BIC values for different numbers of components in the GMM')
plt.legend()
plt.savefig('3_aic.png')
```

Code output:

```
Number of components = [1  2  3  4  5  6  7  8  9 10]
AIC = [1963.18, 1851.71, 1863.22, 1867.27, 1871.14, 1868.89, 1880.13, 1877.91, 1883.78, 1890.11]
BIC = [1973.0, 1876.27, 1902.51, 1921.29, 1939.89, 1952.37, 1978.35, 1990.86, 2011.46, 2032.52]
opitimum n = 2
```

Figs. (2, 3, 4) show the best fit gaussian mixtures along with the real data. The minimum AIC and

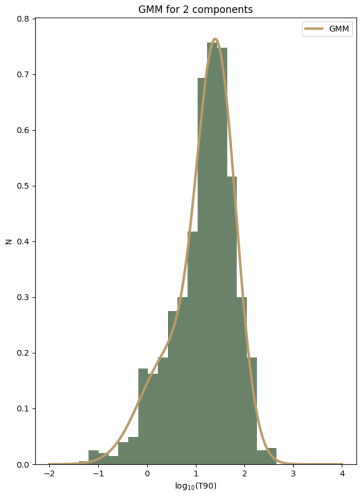

Figure 2: n=2

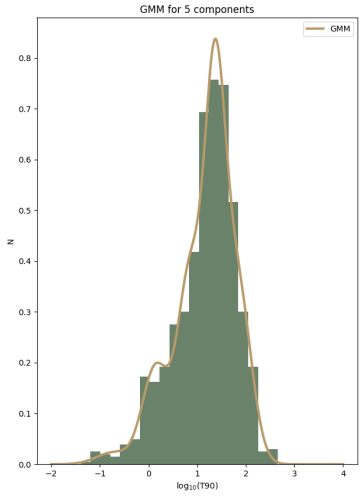

Figure 3: n=5

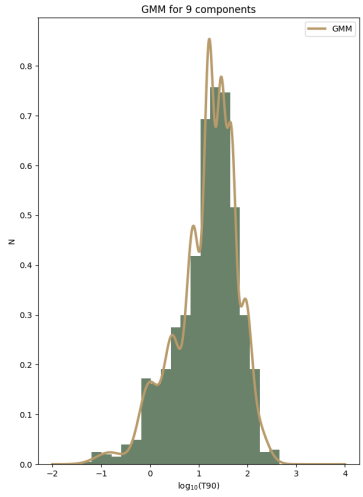

Figure 4: n=9

BIC values are attained at $n = 2$ as shown in Fig 5. Thus a mixture of 2 gaussians offers the best fit.

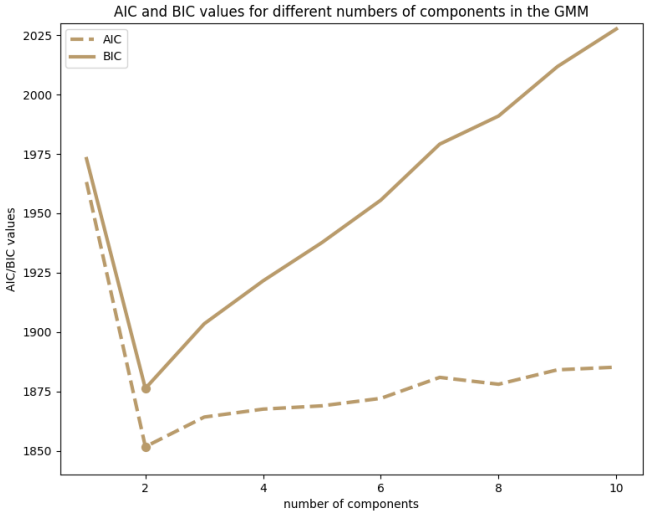

Figure 5: AIC and BIC values

All the codes and figures used in this assignment can be found in this repository