Install the Necessary Libraries

Name : Pratik Yuvraj Yawalkar

Roll No. : AI23MTECH11006

Department : AI & ML

```
!pip install astroML numpy pandas scipy matplotlib seaborn
```

```
Collecting astroML
  Downloading astroML-1.0.2.post1-py3-none-any.whl (134 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/134.3 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╸━━━━━ 112.6/134.3 kB 3.2 MB/s eta
0:00:01 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.3/134.3 kB 2.9
MB/s eta 0:00:00
ent already satisfied: numpy in /usr/local/lib/python3.10/dist-
packages (1.25.2)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: scipy in
/usr/local/lib/python3.10/dist-packages (1.11.4)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: seaborn in
/usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: scikit-learn>=0.18 in
/usr/local/lib/python3.10/dist-packages (from astroML) (1.2.2)
Requirement already satisfied: astropy>=3.0 in
/usr/local/lib/python3.10/dist-packages (from astroML) (5.3.4)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.49.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
```

```
Requirement already satisfied: pyerfa>=2.0 in
/usr/local/lib/python3.10/dist-packages (from astropy>=3.0->astroML)
(2.0.1.1)
Requirement already satisfied: PyYAML>=3.13 in
/usr/local/lib/python3.10/dist-packages (from astropy>=3.0->astroML)
(6.0.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1-
>pandas) (1.16.0)
Requirement already satisfied: joblib>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18-
>astroML) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18-
>astroML) (3.3.0)
Installing collected packages: astroML
Successfully installed astroML-1.0.2.post1
```

1. Download the asteroid dataset from http://astrostatistics.psu.edu/datasets/asteroid_dens.dat. Apply the Shapiro-Wilk test to both the asteroid density values and the natural logarithm of the density values. From the p values, which of these is closer to a Gaussian distribution? Verify this by plotting histograms of both density and its logarithm and overlaying the best-fit normal distribution (Look up stats.norm.fit)

```python
import numpy as np
from scipy.stats import shapiro, norm
import matplotlib.pyplot as plt
import pandas as pd

# Load the data from the file
data = pd.read_csv('asteroid_dens.dat', delim_whitespace=True,
skiprows=1, header=None)
asteroid_number = data[0].values
```

```
asteroid_name = data[1].values
asteroid_density = data[2].values
asteroid_density_error = data[3].values

asteroid_density
```

```
array([2.12, 2.71, 3.44, 2.76, 2.72, 0.96, 2.  , 3.26, 2.5 , 1.2 ,
1.62,
       1.3 , 1.96, 2.6 , 1.3 , 2.67, 4.4 , 1.8 , 4.9 , 2.39, 1.62,
1.47,
       0.89, 2.52, 1.21, 0.9 , 0.8 ])
```

```
# Compute the natural logarithm of density values without the errors
log_density_values = np.log(asteroid_density)
log_density_values
```

```
array([ 0.75141609,  0.99694863,  1.23547147,  1.01523068,
1.00063188,
       -0.04082199,  0.69314718,  1.1817272 ,  0.91629073,
0.18232156,
        0.48242615,  0.26236426,  0.67294447,  0.95551145,
0.26236426,
        0.98207847,  1.48160454,  0.58778666,  1.58923521,
0.87129337,
        0.48242615,  0.3852624 , -0.11653382,  0.9242589 ,
0.19062036,
       -0.10536052, -0.22314355])
```

Define the test parameters:

1. Alpha = 5%
2. Null Hypothesis: Samples comes from a certain Normal Distribution
3. Alternate Hypothesis: Samples does not comes from a certain Normal Distribution

```
# Shapiro-Wilk test for density values with errors
shapiro_test_asteroid_density, shapiro_test_asteroid_density_p_value =
shapiro(asteroid_density)

print("Shapiro-Wilk test results for density values:")
print("Test Statistic:", shapiro_test_asteroid_density, "\np-value:",
shapiro_test_asteroid_density_p_value)

# Fail to reject the null Hypothesis as p_value obtained > alpha
```

```
Shapiro-Wilk test results for density values:
Test Statistic: 0.9246721863746643
p-value: 0.05122082912254333
```

```
# Shapiro-Wilk test for natural logarithm of density values with
errors
shapiro_test_asteroid_density_log,
```

```
shapiro_test_asteroid_density_log_p_value =
shapiro(log_density_values)

print("Shapiro-Wilk test results for log density values:")
print("Test Statistic:", shapiro_test_asteroid_density_log, "\np-
value:", shapiro_test_asteroid_density_log_p_value)

# Fail to reject the null Hypothesis as p_value obtained > alpha

Shapiro-Wilk test results for log density values:
Test Statistic: 0.9686306715011597
p-value: 0.5660613775253296
```

Question :

1. The Shapiro–Wilk test statistic (Calc W) is basically a measure of how well the ordered and standardized sample quantiles fit the standard normal quantiles. It takes a value between 0 and 1.

2. We know that large p-value indicates the data set is normally distributed and a low p-value indicates that it isn't normally distributed.

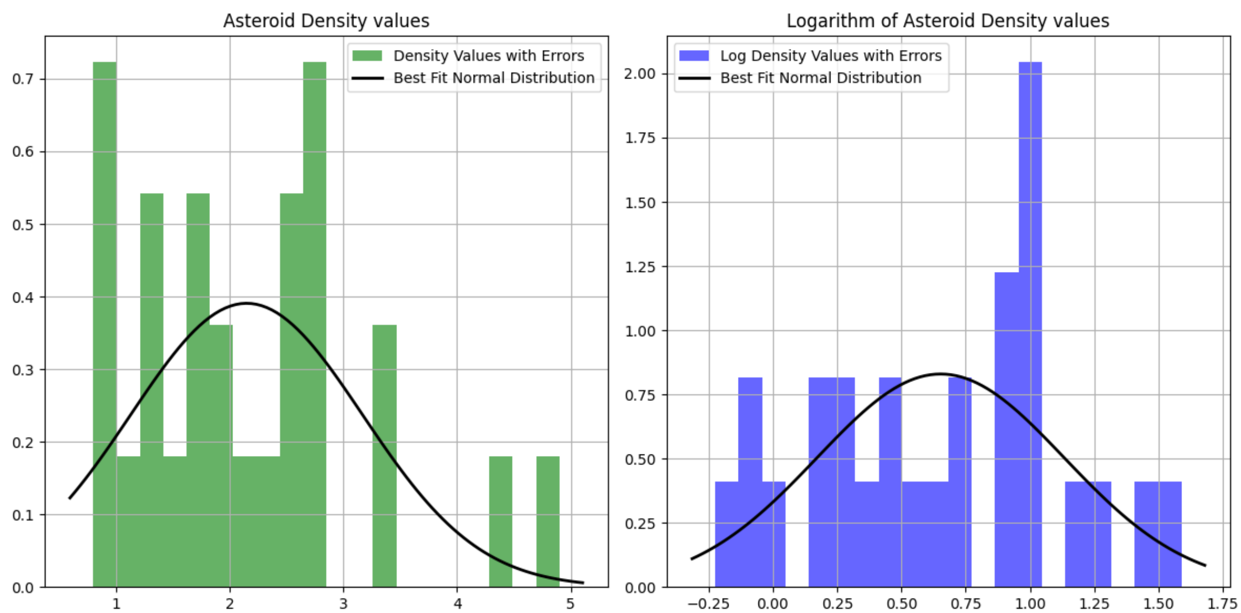
Solution :

1. The results shows that log of asteroid density values fits better to its normal curve that the original density values.

2. The results shows that log of asteroid density values have more p-value than its original density values which indicates a good fit of its values to a Normal Distribution

```
# Plot histograms and overlay best-fit normal distribution
plt.figure(figsize=(12, 6))

# Histogram and best-fit normal for density values
plt.subplot(1, 2, 1)
plt.hist(asteroid_density, bins=20, density=True, alpha=0.6,
color='g', label='Density Values with Errors')
mu, sigma = norm.fit(asteroid_density)
xmin, xmax = plt.xlim()
x_density = np.linspace(xmin, xmax, 100)
p_density = norm.pdf(x_density, mu, sigma)
plt.plot(x_density, p_density, 'k', linewidth=2, label='Best Fit
Normal Distribution')
plt.title('Asteroid Density values')
plt.grid(True)
plt.legend()

# Histogram and best-fit normal for log density values
plt.subplot(1, 2, 2)
```

```
plt.hist(log_density_values, bins=20, density=True, alpha=0.6,
color='b', label='Log Density Values with Errors')
mu_log, sigma_log = norm.fit(log_density_values)
xmin_log, xmax_log = plt.xlim()
x_log_density = np.linspace(xmin_log, xmax_log, 100)
p_log_density = norm.pdf(x_log_density, mu_log, sigma_log)
plt.plot(x_log_density, p_log_density, 'k', linewidth=2, label='Best
Fit Normal Distribution')
plt.title('Logarithm of Asteroid Density values')
plt.grid(True)
plt.legend()

plt.tight_layout()
plt.show()
```



2. Download the Hipparcos star catalog from http://iith.ac.in/~shantanud/HIP_star.dat. Detailed explanation of the columns in this dataset can be found in http://astrostatistics.psu.edu/datasets/HIP_star.html under “Dataset”. Calculate using two-sample t-test whether the color (B-V) of the Hyades stars differs from the non-Hyades ones. The Hyades stars have Right Ascension between 50° and 100°, declinations between 0 and 25°, proper motion in RA between 90 and 130 mas/year, proper motion in DEC between -60 and -10 mas/year. Any other star which does not satisfy any of the above conditions is considered a non-Hyades star.

```
from scipy.stats import ttest_ind
import pandas as pd
import numpy as np

# To give the appropriate column name to the columns of dataframe
columns_names = ['HIP', 'Vmag', 'RA', 'DE', 'Plx', 'pmRA', 'pmDE',
'e_Plx', 'B-V']

# Load the data from the file
data = pd.read_csv('HIP_star.dat', delim_whitespace=True, skiprows=1,
header=None)
data.columns = columns_names
data.head()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 2719,\n  \"fields\":
[\n    {\n      \"column\": \"HIP\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 35587,\n        \"min\": 2,\n
\"max\": 120003,\n        \"num_unique_values\": 2719,\n
\"samples\": [\n          21253,\n          14181,\n          103867\n
```

```
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Vmag\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 1.8847299978868373,\n        \"min\": 0.45,\n        \"max\": 12.74,\n        \"num_unique_values\": 773,\n        \"samples\": [\n          11.94,\n          3.54,\n          12.35\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"RA\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 107.5468491215425,\n        \"min\": 0.003797,\n        \"max\": 359.954685,\n        \"num_unique_values\": 2719,\n        \"samples\": [\n          68.392082,\n          45.71756,\n          315.695665\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"DE\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 38.93038701489685,\n        \"min\": -87.20273,\n        \"max\": 88.302681,\n        \"num_unique_values\": 2719,\n        \"samples\": [\n          -62.823631,\n          47.110697,\n          -64.294709\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Plx\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 1.4171932477723566,\n        \"min\": 20.0,\n        \"max\": 25.0,\n        \"num_unique_values\": 497,\n        \"samples\": [\n          24.91,\n          22.19,\n          20.77\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"pmRA\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 160.97985500671155,\n        \"min\": -868.01,\n        \"max\": 781.34,\n        \"num_unique_values\": 2668,\n        \"samples\": [\n          -15.1,\n          -250.63,\n          -221.86\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"pmDE\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 140.8904227640606,\n        \"min\": -1392.3,\n        \"max\": 481.19,\n        \"num_unique_values\": 2618,\n        \"samples\": [\n          114.03,\n          -69.35,\n          108.03\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"e_Plx\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 2.212866685496305,\n        \"min\": 0.45,\n        \"max\": 46.91,\n        \"num_unique_values\": 409,\n        \"samples\": [\n          0.5,\n          1.52,\n          1.3\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"B-V\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.31818761314250465,\n        \"min\": -0.158,\n        \"max\": 2.8,\n        \"num_unique_values\": 1031,\n        \"samples\": [\n          0.743,\n          0.814,\n          0.964\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"data"}
```

```
data.describe()
```


{"summary":"{\n \"name\": \"data\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"HIP\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 41492.17418906186,\n   \"min\": 2.0,\n   \"max\": 120003.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   56549.48289812431,\n   56413.0,\n   2719.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Vmag\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 958.8817995652839,\n   \"min\": 0.45,\n   \"max\": 2719.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   8.259385803604266,\n   8.28,\n   2719.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"RA\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 910.0952099895878,\n   \"min\": 0.003797,\n   \"max\": 2719.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   173.45299745972784,\n   173.369788,\n   2719.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"DE\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 960.6733192260226,\n   \"min\": -87.20273,\n   \"max\": 2719.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   -0.1397662603898495,\n   3.254234,\n   2719.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Plx\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 954.519659200025,\n   \"min\": 1.4171932477723566,\n   \"max\": 2719.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   22.19802133137183,\n   22.1,\n   2719.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"pmRA\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1056.0948679573617,\n   \"min\": -868.01,\n   \"max\": 2719.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   5.376134608311875,\n   10.55,\n   2719.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"pmDE\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1148.8948183527716,\n   \"min\": -1392.3,\n   \"max\": 2719.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   -63.94199337991909,\n   -49.48,\n   2719.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"e_Plx\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 958.6721614714997,\n   \"min\": 0.45,\n   \"max\": 2719.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   1.6267929385803603,\n   1.14,\n   2719.0\n   ],\n

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"B-V\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 946.5160953612981,\n
\"min\": -0.158,\n   \"max\": 2678.0,\n
\"num_unique_values\": 8,\n   \"samples\": [\n
0.7615298730395818,\n   0.7104999999999999,\n   2678.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe"}

```
# Replace all the nan values by the median of the rest of values if
there are some nan values
if(pd.isna(data).sum().sum()):
    for column in data.columns:
        median_value = data[column].median()
        data[column].fillna(median_value, inplace=True)

# Define criteria for Hyades stars and non-Hyades stars
hyades_criteria = (
    (data['RA'] >= 50) & (data['RA'] <= 100) &
    (data['DE'] >= 0) & (data['DE'] <= 25) &
    (data['pmRA'] >= 90) & (data['pmRA'] <= 130) &
    (data['pmDE'] >= -60) & (data['pmDE'] <= -10)
)

# Apply the criteria to filter Hyades and non-Hyades stars
hyades_stars = data[hyades_criteria]
non_hyades_stars = data[~hyades_criteria]

hyades_stars.head()
```

{"summary":"{\n \"name\": \"hyades_stars\",\n \"rows\": 93,\n
\"fields\": [\n   {\n   \"column\": \"HIP\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
704,\n   \"min\": 18735,\n   \"max\": 22607,\n
\"num_unique_values\": 93,\n   \"samples\": [\n   20648,\n
20284,\n   20885\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Vmag\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 2.015189775603542,\n   \"min\":
3.4,\n   \"max\": 11.66,\n   \"num_unique_values\": 88,\n
\"samples\": [\n   7.9,\n   5.89,\n   8.84\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"RA\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 2.3111062449178896,\n
\"min\": 60.202858,\n   \"max\": 72.958017,\n
\"num_unique_values\": 93,\n   \"samples\": [\n
66.372156,\n   65.219406,\n   67.143468\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"DE\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 3.004596687175422,\n

\"min\": 10.751794,\n   \"max\": 24.405512,\n
\"num_unique_values\": 93,\n   \"samples\": [\n
17.927989,\n   13.864471,\n   15.962217\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Plx\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 1.2016096419596107,\n
\"min\": 20.01,\n   \"max\": 24.98,\n
\"num_unique_values\": 84,\n   \"samples\": [\n   24.02,\n
21.99,\n   23.25\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"pmRA\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 7.773615701542547,\n   \"min\":
90.28,\n   \"max\": 129.49,\n   \"num_unique_values\": 90,\n
\"samples\": [\n   108.26,\n   105.29,\n
104.76\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"pmDE\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 10.049352038357341,\n   \"min\": -55.25,\n
\"max\": -10.52,\n   \"num_unique_values\": 91,\n
\"samples\": [\n   -32.47,\n   -17.97,\n   -
15.01\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"e_Plx\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.7820268615304802,\n   \"min\": 0.74,\n   \"max\":
6.94,\n   \"num_unique_values\": 64,\n   \"samples\": [\n
1.65,\n   0.76,\n   0.81\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"B-V\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.32703342012325415,\n
\"min\": 0.049,\n   \"max\": 1.408,\n
\"num_unique_values\": 89,\n   \"samples\": [\n   0.262,\n
1.014,\n   0.855\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"hyades_stars"}

```
non_hyades_stars.head()
```

{"summary":"{\n \"name\": \"non_hyades_stars\",\n \"rows\": 2626,\n
\"fields\": [\n   {\n   \"column\": \"HIP\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
35555,\n   \"min\": 2,\n   \"max\": 120003,\n
\"num_unique_values\": 2626,\n   \"samples\": [\n   7370,\
n   10535,\n   40732\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Vmag\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 1.8700732441590535,\n
\"min\": 0.45,\n   \"max\": 12.74,\n
\"num_unique_values\": 765,\n   \"samples\": [\n   8.01,\n
9.79,\n   4.24\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":

```
\"RA\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 107.5030545975371,\n        \"min\": 0.003797,\n
\"max\": 359.954685,\n        \"num_unique_values\": 2626,\n
\"samples\": [\n          23.742165,\n          33.928477,\n
124.697872\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"DE\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 39.47732105748982,\n        \"min\": -87.20273,\n
\"max\": 88.302681,\n        \"num_unique_values\": 2626,\n
\"samples\": [\n          33.11819,\n          25.043255,\n          -
3.361573\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Plx\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.4243389287432813,\n        \"min\": 20.0,\n        \"max\":
25.0,\n        \"num_unique_values\": 497,\n        \"samples\": [\n
24.58,\n          22.19,\n          20.77\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"pmRA\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 162.6059775659848,\n
\"min\": -868.01,\n        \"max\": 781.34,\n
\"num_unique_values\": 2580,\n        \"samples\": [\n          -
56.97,\n          154.47,\n          353.36\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"pmDE\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 143.19174789683603,\n
\"min\": -1392.3,\n        \"max\": 481.19,\n
\"num_unique_values\": 2533,\n        \"samples\": [\n          -
329.34,\n          107.51,\n          -41.17\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"e_Plx\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 2.2463019052896804,\
n        \"min\": 0.45,\n        \"max\": 46.91,\n
\"num_unique_values\": 407,\n        \"samples\": [\n          1.01,\n
0.55,\n          2.23\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"B-
V\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.3143612415151376,\n        \"min\": -0.158,\n
\"max\": 2.8,\n        \"num_unique_values\": 1012,\n
\"samples\": [\n          0.935,\n          0.813,\n          0.997\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"non_hyades_stars"}
```


```
# Extract the color (B-V) values for both groups
hyades_color = hyades_stars['B-V']
hyades_color = hyades_color.values

non_hyades_color = non_hyades_stars['B-V']
non_hyades_color = non_hyades_color.values
```

Define the test parameters:

1. Alpha = 5%
2. Null Hypothesis: There is no difference between color (B-V)
3. Alternate Hypothesis: There is difference between color (B-V)

```
# Perform two-sample t-test
t_stat, p_value = ttest_ind(hyades_color, non_hyades_color,
equal_var=False)

print("Two-sample t-test results:")
print("T-statistic:", t_stat)
print("P-value:", p_value)

# Pass to reject the null hypothesis as p_value obtained < alpha
```

```
Two-sample t-test results:
T-statistic: -4.202327212391445
P-value: 5.82998684016138e-05
```

Question :

1. The T-test is a test for the null hypothesis that 2 independent samples have identical average (expected) values.

2. The p-value quantifies the probability of observing as or more extreme values assuming the null hypothesis, that the samples are drawn from populations with the same population means, is true.

Solution :

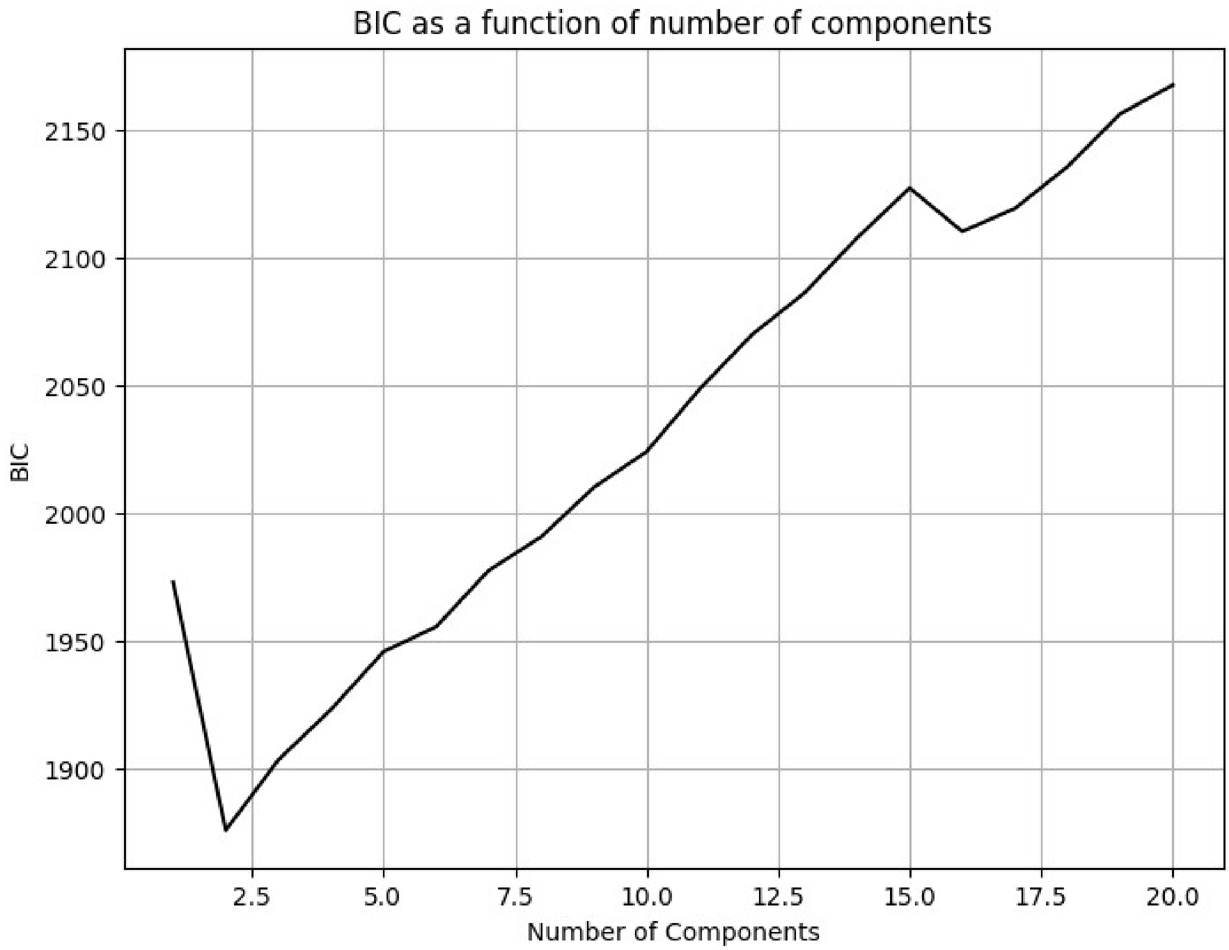
1. The results gives a significant difference bet'n mean of t-distribution and currrent t_stat value.

2. The results shows that there is a significant difference between both hyades color and non hyades color

3. The T90 distribution for Beppo-Sax T90 data can be found at http://www.iith.ac.in/~shantanud/beppoSax.txt. Apply GMM to log10 of T90 data and find the optimum number of components using AIC and BIC by plotting BIC as a function of number of compонts (20 points) (Hint: Look at the source code for astroML figure 6.6)

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


# Load the data from the file
data = np.loadtxt('beppoSax.dat')
data

array([  3.   ,  11.   ,  14.   , ..., 109.   ,   4.45,  36.   ])

# Take the log10 of T90
logT90 = np.log10(data)
logT90

array([0.47712125, 1.04139269, 1.14612804, ..., 2.0374265 ,
0.64836001,
       1.5563025 ])

# Define the range of components to consider
n_components = np.arange(1, 21)

# Initialize arrays to store AIC and BIC values
AIC = np.zeros(n_components.shape)
BIC = np.zeros(n_components.shape)

# Fit models and compute AIC and BIC
for i, n in enumerate(n_components):
    # Fit a Gaussian mixture model
    gmm = GaussianMixture(n_components=n,
random_state=0).fit(logT90.reshape(-1, 1))

    # Calculate AIC and BIC
```

```
    AIC[i] = gmm.aic(logT90.reshape(-1, 1))
    BIC[i] = gmm.bic(logT90.reshape(-1, 1))

# Plot BIC as a function of the number of components
plt.figure(figsize=(8, 6))
plt.plot(n_components, BIC, '-k')
plt.xlabel('Number of Components')
plt.ylabel('BIC')
plt.title('BIC as a function of number of components')
plt.grid(True)
plt.show()
```



Hence the number of components used are 2 for the GMM.