KRITIK #7

February 11, 2024

[2]:
```
#1a)
import numpy as np
from scipy.special import gamma

def t_distribution_pdf(x,nu): #using code given in assignment and just adding␣
 ↪random values for x and nu
    x=2.5
    nu=4
    coeff=gamma((nu + 1) / 2) / (np.sqrt(nu * np.pi) * gamma(nu / 2))
    density = coeff * (1 + x**2 / nu) ** (-0.5 * (nu + 1))
    return density


print (t_distribution_pdf(2.5,4))
```

```
0.03567562436955666
```

[3]:
```
#2a)
test_scores= [92.64,79.00,84.79,97.41,93.68,65.23,84.50,73.49,73.97,79.11]␣
 ↪#initializing the list
mean = sum(test_scores)/len(test_scores) #finding average by summing the␣
 ↪numbers in list and dividing by amount of terms
def standard_deviation(test_scores): #defining a function to calculate standard␣
 ↪deviation
       subtraction_squared = ((x-mean)**2 for x in test_scores) #represents␣
 ↪(xi-mean)^2 portion of standard deviation formula
       summation = sum(subtraction_squared) #creates a summation of␣
 ↪subtraction squared as per the formula of standard dev
       deviation = np.sqrt((1/9)*summation) #the formula given in assignment␣
 ↪for standard dev
       return deviation

print ("The mean value is:", mean) #returns mean value
print ("The standard deviation value is:",standard_deviation(test_scores))␣
 ↪#returns standard deviation
```

```
The mean value is: 82.382
```

```
The standard deviation value is: 10.193467189005581
```

[4]:
```
#2b)
t0= (mean-75)/((standard_deviation(test_scores)/np.sqrt(len(test_scores))))␣
 ↪#formula for t0 given in assignment
print ("t0 is:",t0)
```

```
t0 is: 2.290087686017293
```

[5]:
```
#2c)
prob=0.95 #setting probability to 0.95 since we want a certainty of 95%
nu= len(test_scores)-1 #len(test_scores) represents n, and nu= n-1
x_start=0 #initializing the x start, x end and numpoints values given in␣
 ↪assignment
x_end=20
num_points=10000

def t_distribution_pdf(x,nu): #redefining t_distribution
    coeff=gamma((nu + 1) / 2) / (np.sqrt(nu * np.pi) * gamma(nu / 2))
    density = coeff * (1 + x**2 / nu) ** (-0.5 * (nu + 1))
    return density

def find_t_star(prob, nu, x_start, x_end, num_points): #importing Greg's code␣
 ↪from assignment
    # Define the x values
    x = np.linspace(x_start, x_end, num_points)
    # Apply the density function to the x values
    y = t_distribution_pdf(x, nu)
    # This next line is the integration (exercise: why does this work?)
    cdf = np.cumsum(y) * (x[1] - x[0])
    target_half_prob = prob / 2
    index = np.where(cdf >= target_half_prob)[0][0]
    return x[index]
print ("t* is:",find_t_star(prob, nu, x_start, x_end, num_points))
```

```
t* is: 2.2522252225222523
```

[8]:
```
#2d and #2e
#Finding if t0 is in the range of -t* to t* to determine whether u can equal 75
if -(find_t_star(prob, nu, x_start, x_end, num_points))<=t0<=(find_t_star(prob,␣
 ↪nu, x_start, x_end, num_points)):
    print ("True : We accept the null hypothesis, and u=75")
else:
    print ("False : We reject the null hypothesis, and u is not equal to 75 ")
```

```
False : We reject the null hypothesis, and u is not equal to 75
```

[ ]:
```

```