Quiz_5

November 9, 2022

1 Rules

- Internet and Books are ALLOWED
- Name your file as following: StudentID_ChineseName/EnglishName_quiz#
- Extension of your file or your file type should be .py .
- If MOSS (Measure of Software Similarity) detects that any two files have more than 50% similarity, both students will get 0 for this quiz (We give it a base code that everyone has, so no need to worry about that)
- If you submit the code within an hour you will get the full score. Otherwise, you will have 24 hours to finish it and you will get 80% of your final score.

#Problem 1

#Instructions - Do a few probability and density calculations for a normal distribution.
- Calculate and plot the likelihood of a sample of just 3 observations. - Determine the Maximum Likelihood Estimates.

Let

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

Determine

$$P(X \geq t)$$

(a) Let $X \sim \mathcal{N}(500, 75^2)$. Determine $P(X \geq 600)$

Use norm.cdf from scipy.stats to find $P(X \geq 600)$. **Print it out**

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

[ ]:
```
from scipy.stats import norm
```

[ ]:
```
prob = 1-norm.cdf(___,___,___)
prob
```

(b) **Plot** the normal distribution of $X \sim N(500, 75^2)$

[ ]:
```
# define parameters
mu = ___
sigma = ___

# the 'dummy' x for plotting
x = np.arange(200,800)

# calculate the normal distribution at each value of x
prob = norm.pdf(___,mu,sigma)

# plot it
plt.plot(___,___);
plt.title(r'$\mathrm{N(\mu=500, \sigma^2=75^2)}$')
plt.ylim((0,0.006))
plt.show()
```

(c) Calculating simple likelihoods

[ ]:
```
### edTest(test_likelihood) ###
# define the data set
x = [3,5,10]

# sigma is known to be 2, an estimate for mu
# is what we need to determine. Consider
#the values (4, 4.01, 4.02, ..., 7.99).
sigma = 2
mu = np.arange(___,___,0.01)

# calculate the likelihood
like = norm.pdf(x[0],mu,sigma)*___*___

#plot it
plt.plot(mu,like,color="darkred");
plt.title('Likelihood Function')
plt.xlabel(r'$\mu$')
plt.show()
```

(d) Determine the maximum likelihood estimate for μ. Print it out.

[ ]:
```
### edTest(test_mle) ###
# determine which value of mu aligns with where
# the maximum of the likelihood function is
mle = ___[np.argmax(__)]
mle
```

(e) Find the CDF from the PDF figure that you obtained. Use **prob** in (b) to find $P(X \geq 600)$

Hint: CDF is the **sum** of the PDF up to certain value.

In this case, your PDF is the variable **prob** in (b). Your answer should be similar to (a).

[ ]:
```
# Your code here
```

(f) Plot your CDF

[ ]:
```
# Your code here
```

Question: How would you numerically maximize this function if both the mean and variance were unknown? How would you visualize the likelihood function?

#Problem 2

2 Description

- Define a function bootstrap that takes a dataframe as the input. Use NumPy's random.randint() function to generate random integers in the range of the length of the dataset. These integers will be used as the indices to access the rows of the dataset.
- Compute the β_0 and β_1 values for each instance of the dataframe.
- Plot the β_0, β_1 histograms.

3 Hints:

To compute the beta values use the following equations:

$$\beta_0 = \bar{y} - (b_1 * \bar{x})$$

$$\beta_1 = \frac{\Sigma(x - \bar{x}) * (y - \bar{y})}{\Sigma(x - \bar{x})^2}$$

where $\bar{x}$ is the mean of x and $\bar{y}$ is the mean of y

np.random.randint() : Returns list of integers as per mentioned size

np.dot() : Computes the dot product of two arrays

[ ]:
```
# Read the file "Advertising_csv"
df = pd.read_csv('Advertising_adj.csv')
```

[ ]:
```
# Define a bootstrap function, which inputs a dataframe & outputs a␣
 ↪bootstrapped dataframe
def bootstrap(df):
    selectionIndex = np.random.randint(___, size = ___)
    new_df = df.iloc[___]
    return new_df
```

[ ]:
```
# Create two empty lists to store beta values
beta0_list, beta1_list = [],[]


# For each instance of the for loop, call your bootstrap function, calculate the beta values
# Store the beta values in the appropriate list

#Choose the number of "parallel" Universes to generate the new dataset
number_of_bootstraps = 1000

for i in range(number_of_bootstraps):
    df_new = bootstrap(df)

# x is the predictor variable given by 'tv' values
# y is the response variable given by 'sales' values
    x = ___
    y = ___

#Find the mean of x
    xmean = x.___

#Find the mean of y
    ymean = y.___

# Using equations given and discussed in lecture compute the beta0 and beta1 values
# Hint: use np.dot to perform the multiplication operation
    beta1 = ___
    beta0 = ___

# Append the calculated values of beta1 and beta0
    beta0_list.append(___)
    beta1_list.append(___)
```

[ ]:
```
### edTest(test_beta) ###

#Compute the mean of the beta0 and beta1 lists
beta0_mean = np.mean(___)
beta1_mean = np.mean(___)
```

[ ]:
```
# plot histogram of beta0 and beta1
fig, ax = plt.subplots(1,2, figsize=(18,8))
ax[0].___
ax[1].___
ax[0].set_xlabel(___)
ax[1].set_xlabel(___)
```

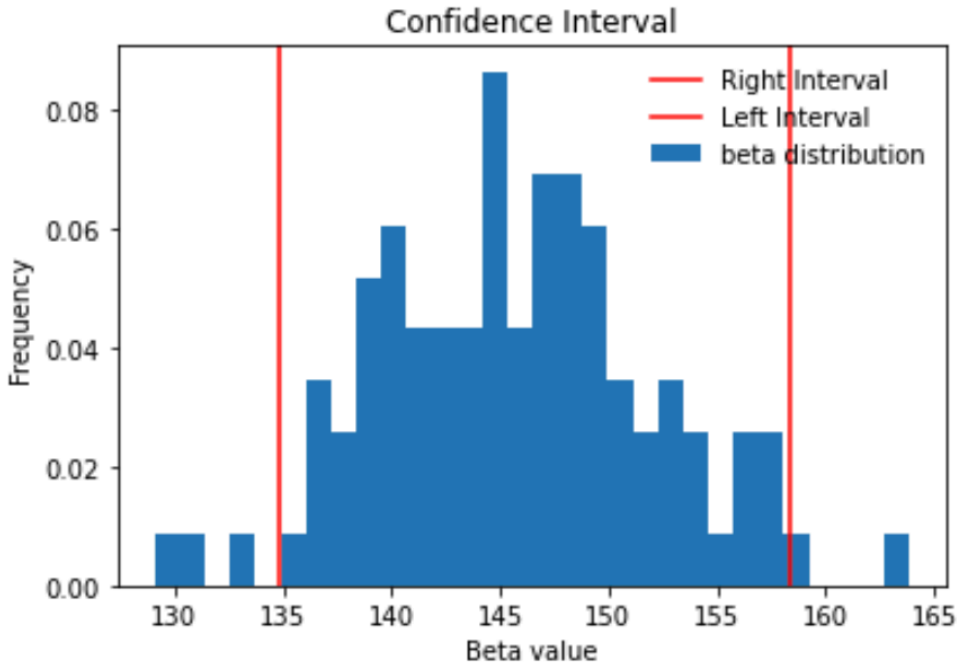
```
ax[0].set_ylabel('Frequency');
```

4 Problem 3

5 Description

The goal of this exercise is to create a plot like the one given below for β_0 and β_1 with its confidence interval

Note: It is going to look SIMILAR, but probably not the same



#Instructions - Follow the steps from the previous exercise to get the lists of beta values. - Sort the list of beta values (from low to high).
- To compute the 95% confidence interval, find the 2.5 percentile and the 97.5 percentile using np.percentile() - Use the helper code plot_simulation() to visualise the values along with its confidence interval

6 Hints:

np.random.randint() : Returns list of integers as per mentioned size

df.iloc[] : Purely integer-location based indexing for selection by position

plt.hist() : Plots a histogram

ndarray.sort() :Returns the sorted ndarray.

np.percentile(list, q) : Returns the q-th percentile value based on the provided ascending list of values

[ ]:
```
# Read the 'Advertising_adj.csv' file
df = pd.read_csv('Advertising_adj.csv')
```

[ ]:
```
# Use your bootstrap function from the previous exercise
```

[ ]:
```
# Like last time, create a list of beta values using 100 bootstraps of your original data
beta0_list, beta1_list = [],[]

numberOfBootstraps = 100

for i in range(numberOfBootstraps):
    df_new = bootstrap(df)

    xmean = df_new.tv.mean()
    ymean = df_new.sales.mean()

    beta1 = np.dot((df_new.tv-xmean) , (df_new.sales-ymean))/((df_new.tv-xmean)**2).sum()
    beta0 = ymean - beta1*xmean

    beta0_list.append(beta0)
    beta1_list.append(beta1)
```

[ ]:
```
### edTest(test_sort) ###

# Sort the two lists of beta values  from lowest value to highest
beta0_list.___;
beta1_list.___;
```

[ ]:
```
### edTest(test_beta) ###

# Now we find the confidence interval
# Find the 95% percent confidence interval using the percentile function
beta0_CI = (np.___,np.___)

beta1_CI = (np.___,np.___)
```

[ ]:
```
#Print the confidence interval of beta0 upto 3 decimal points
print(f'The beta0 confidence interval is {___}')
```

[ ]:
```
#Print the confidence interval of beta1 upto 3 decimal points
print(f'The beta1 confidence interval is {___}')
```

[ ]:
```
# Use this helper function to plot the histogram of beta values along with the␣
 ↪95% confidence interval
def plot_simulation(simulation,confidence):
    plt.hist(simulation, bins = 30, label = 'beta distribution', align =␣
 ↪'left', density = True)
    plt.axvline(confidence[1], 0, 1, color = 'r', label = 'Right Interval')
    plt.axvline(confidence[0], 0, 1, color = 'red', label = 'Left Interval')
    plt.xlabel('Beta value')
    plt.ylabel('Frequency')
    plt.title('Confidence Interval')
    plt.legend(frameon = False, loc = 'upper right')
```

[ ]:
```
# Plot for beta 0
plot_simulation(___, ___)
```

[ ]:
```
#Plot for beta 1
plot_simulation(___, ___)
```