# 1. A university wants to understand the relationship between the SAT scores of its applicants and their college GPA. They collect data on 500 students, including their SAT scores (out of 1600) and their college GPA (on a 4.0 sca­le). They f­ind that the correlation coefficient between SAT scores and college GPA is 0.7. What does this correlation coefficient indicate about the relationship between SAT scores and college GPA?

# Sol- A correlation coefficient of 0.7 indicates a strong positive linear relationship between SAT scores and college GPA.

# In other words, there is a strong tendency for students with higher SAT scores to also have higher college GPAs, and vice versa. The positive sign of the correlation coefficient indicates that as SAT scores increase, college GPAs tend to increase as well.

# It's important to note that correlation does not imply causation. While there is a strong correlation between SAT scores and college GPA, it does not necessarily mean that SAT scores directly cause higher or lower college GPAs. Other factors, such as study habits, motivation, and personal characteristics, may also contribute to a student's academic performance.

# Q-2. Consider a dataset containing the heights (in centimeters) of 1000 individuals. The mean height is 170 cm with a standard deviation of 10 cm. The dataset is approximately normally distributed, and its skewness is approximately zero. Based on this information, answer the following questions: a. What percentage of individuals in the dataset have heights between 160 cm and 180 cm?

# b. If we randomly select 100 individuals from the dataset, what is the probability that their average height is greater than 175 cm?

# c. Assuming the dataset follows a normal distribution, what is the z-score corresponding to a height of 185 cm?

# d. We know that 5% of the dataset has heights below a certain value. What is the approximate height corresponding to this threshold?

# e. Calculate the coefficient of variation (CV) for the dataset.

# f. Calculate the skewness of the dataset and interpret the result.

# In [12]:

# *#a*

# import scipy.stats as stats

# mean = 170

# std\_dev = 10

# percentage = (stats.norm.cdf(180, mean, std\_dev) - stats.norm.cdf(160, mean, std\_dev)) \* 100

# print(percentage)

# 68.26894921370858

# In [13]:

# *#b*

# sample\_size = 100

# sample\_mean = mean

# sample\_std\_dev = std\_dev / (sample\_size \*\* 0.5)

# probability = 1 - stats.norm.cdf(175, sample\_mean, sample\_std\_dev)

# print(probability)

# 2.866515719235352e-07

# In [14]:

# *#c*

# z\_score = (185 - mean) / std\_dev

# print(z\_score)

# 1.5

# In [15]:

# *#d*

# height = stats.norm.ppf(0.05, mean, std\_dev)

# print(height)

# 153.55146373048527

# In [16]:

# *#e*

# cv = (std\_dev / mean) \* 100

# print(cv)

# 5.88235294117647

# f

# The skewness of a dataset measures the asymmetry of its distribution. A skewness value of approximately zero indicates that the dataset is symmetric and does not have a significant tail on either side. In other words, the dataset's distribution is relatively balanced around the mean.

# Q.3-Consider the ‘Blood Pressure Before’ and ‘Blood Pressure After’ columns from the data and calculate the following <https://drive.google.com/file/d/1mCjtYHiX--mMUjicuaP2gH3k-SnFxt8Y/view?usp=share_>

# In [17]:

# *#a. Measure the dispersion in both and interpret the results.*

# *#Sol-*

# import pandas as pd

# df=pd.read\_csv(r"data.csv")

# before\_variance = df[" Blood Pressure Before (mmHg)"].var()

# after\_variance = df[" Blood Pressure After (mmHg)"].var()

# print("Variance before:",before\_variance,"Variance after:",after\_variance)

# Variance before: 43.53727272727271 Variance after: 47.4448484848485

# In [18]:

# *#b*

# import numpy as np

# import matplotlib.pyplot as plt

# from scipy import stats

# mean = np.mean(df)

# confidence\_interval = stats.t.interval(0.95, len(df) - 1, loc=mean, scale=stats.sem(df))

# print("Mean:", mean)

# print("Confidence Interval (95%):", confidence\_interval)

# *#plotting in a bar graph*

# plt.figure()

# plt.bar(x=1, height=mean, color='red')

# plt.xticks([])

# plt.xlim(0.5, 1.5)

# plt.ylabel("Values")

# plt.title("Mean and 5% Confidence Interval")

# plt.show()

# C:\Users\Admin\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3438: FutureWarning: In a future version, DataFrame.mean(axis=None) will return a scalar mean over the entire DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or just 'frame.mean()'

# return mean(axis=axis, dtype=dtype, out=out, \*\*kwargs)

# Mean: Patient ID 50.50

# Blood Pressure Before (mmHg) 133.91

# Blood Pressure After (mmHg) 128.36

# dtype: float64

# Confidence Interval (95%): (array([ 44.74349058, 132.60075849, 126.99326698]), array([ 56.25650942, 135.21924151, 129.72673302]))

# 

ms **=** np**.**mean(np**.**abs(df **-** np**.**mean(df)))

sd **=** np**.**std(df)

print("Mean absolute deviation",ms)

print("standard deviation",sd)

Mean absolute deviation Patient ID 25.0000

Blood Pressure Before (mmHg) 5.7118

Blood Pressure After (mmHg) 5.9000

dtype: float64

standard deviation Patient ID 28.866070

Blood Pressure Before (mmHg) 6.565204

Blood Pressure After (mmHg) 6.853495

dtype: float64

In [20]:

*#d*

variable1 **=** df[' Blood Pressure Before (mmHg)']

variable2 **=** df[' Blood Pressure After (mmHg)']

*# Calculate the correlation coefficient and p-value*

correlation\_coefficient, p\_value **=** stats**.**pearsonr(variable1, variable2)

*# Print the correlation coefficient and p-value*

print("Correlation Coefficient:", correlation\_coefficient)

print("p-value:", p\_value)

*# Check the significance at 1% level*

**if** p\_value **<** 0.01:

print("The correlation coefficient is significant at the 1% level.")

**else**:

print("The correlation coefficient is not significant at the 1% level.")

Correlation Coefficient: 0.9779485966556017

p-value: 1.8097381433072956e-68

The correlation coefficient is significant at the 1% level.

Q-4. A group of 20 friends decide to play a game in which they each write a number between 1 and 20 on a slip of paper and put it into a hat. They then draw one slip of paper at random. What is the probability that the number on the slip of paper is a perfect square (i.e., 1, 4, 9, or 16)?

In [21]:

*#Sol-4*

total\_outcomes **=** 20

favorable\_outcomes **=** 4

probability **=** favorable\_outcomes **/** total\_outcomes

print("The probability of drawing a perfect square number is:", probability)

The probability of drawing a perfect square number is: 0.2

Q-5. A certain city has two taxi companies: Company A has 80% of the taxis and Company B has 20% of the taxis. Company A's taxis have a 95% success rate for picking up passengers on time, while Company B's taxis have a 90% success rate. If a randomly selected taxi is late, what is the probability that it belongs to Company A?

In [22]:

*#Sol-5*

P\_A **=** 0.8

P\_B\_given\_A **=** 0.05

P\_B\_given\_B **=** 0.10

P\_B **=** P\_B\_given\_A **\*** P\_A **+** P\_B\_given\_B **\*** (1 **-** P\_A)

P\_A\_given\_B **=** (P\_B\_given\_A **\*** P\_A) **/** P\_B

*# Print the result*

print("The probability that a randomly selected late taxi belongs to Company A:", P\_A\_given\_B)

The probability that a randomly selected late taxi belongs to Company A: 0.6666666666666667

Q-6. A pharmaceutical company is developing a drug that is supposed to reduce blood pressure. They conduct a clinical trial with 100 patients and record their blood pressure before and after taking the drug. The company wants to know if the change in blood pressure follows a normal distribution.

<https://drive.google.com/file/d/1mCjtYHiX--mMUjicuaP2gH3k-SnFxt8Y/view?usp=share_>

In [23]:

*# Calculate the change in blood pressure*

change\_in\_bp **=**df[' Blood Pressure After (mmHg)'] **-** df[' Blood Pressure Before (mmHg)']

*# Plot a histogram of the change in blood pressure*

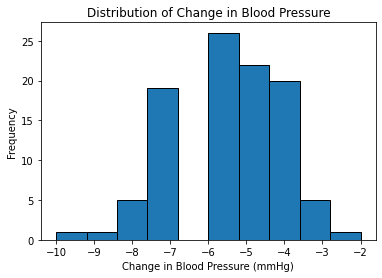
plt**.**hist(change\_in\_bp, bins**=**10, edgecolor**=**'black')

plt**.**xlabel("Change in Blood Pressure (mmHg)")

plt**.**ylabel("Frequency")

plt**.**title("Distribution of Change in Blood Pressure")

plt**.**show()

****

In [24]:

**from** scipy.stats **import** shapiro

**from** scipy.stats **import** shapiro

*# Perform Shapiro-Wilk test for normality*

statistic, p\_value **=** shapiro(change\_in\_bp)

*# Print the test statistic and p-value*

print("Shapiro-Wilk test statistic:", statistic)

print("p-value:", p\_value)

Shapiro-Wilk test statistic: 0.953819990158081

p-value: 0.0014938802924007177

Since the p-value (0.0015) is less than the typical significance level of 0.05, we reject the null hypothesis that the data follows a normal distribution. This suggests that the change in blood pressure in the clinical trial does not follow a normal distribution.

Q-7. The equations of two lines of regression, obtained in a correlation analysis between variables X and Y are as follows:

and . 2X + 3 − 8 = 0 2Y + X − 5 = 0 The variance of X = 4 Find the a. Variance of Y b. Coefficient of determination of C and Y c. Standard error of estimate of X on Y and of Y on X.

Q-8. The anxiety levels of 10 participants were measured before and after a new therapy. The scores are not normally distributed. Use the Wilcoxon signed-rank test to test whether the therapy had a significant effect on anxiety levels. The data is given below: Participant Before therapy After therapy Difference

In [25]:

**import** pandas **as** pd

*# Create a dictionary with the data*

data **=** {

'Participant': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Before therapy': [10, 8, 12, 15, 6, 9, 11, 7, 14, 10],

'After therapy': [7, 6, 10, 12, 5, 8, 9, 6, 12, 8],

'Difference': [**-**1, **-**1, **-**1, **-**1, **-**1, **-**1, **-**1, **-**1, **-**1, **-**1]

}

*# Create a DataFrame from the dictionary*

df1**=** pd**.**DataFrame(data)

*# Calculate the difference*

df1['Difference'] **=** df1['After therapy'] **-** df1['Before therapy']

*# Print the DataFrame*

print(df1)

Participant Before therapy After therapy Difference

0 1 10 7 -3

1 2 8 6 -2

2 3 12 10 -2

3 4 15 12 -3

4 5 6 5 -1

5 6 9 8 -1

6 7 11 9 -2

7 8 7 6 -1

8 9 14 12 -2

9 10 10 8 -2

In [26]:

**from** scipy.stats **import** wilcoxon

*# Perform Wilcoxon signed-rank test*

statistic, p\_value **=** wilcoxon(df1['Before therapy'], df1['After therapy'])

*# Print the test statistic and p-value*

print("Wilcoxon signed-rank test statistic:", statistic)

print("p-value:", p\_value)

Wilcoxon signed-rank test statistic: 0.0

p-value: 0.001953125

Since the p-value (0.0019) is less than the typical significance level of 0.05, we can conclude that the therapy had a significant effect on anxiety levels. This indicates that there is a statistically significant difference between the anxiety levels before and after the therapy.

In summary, the Wilcoxon signed-rank test suggests that the therapy had a significant positive effect on reducing anxiety levels.

Q-9. Given the score of students in multiple exams

Test the hypothesis that the mean scores of all the students are the same. If not, name the student with the highest score.

In [27]:

**import** pandas **as** pd

*# Create a dictionary with the data*

data **=** {

'Name': ['Karan', 'Deepa', 'Karthik', 'Chandan', 'Jeevan'],

'Exam 1': [85, 70, 90, 75, 95],

'Exam 2': [90, 80, 85, 70, 92],

'Final Exam': [92, 85, 88, 75, 96]

}

*# Create a DataFrame from the dictionary*

df2**=** pd**.**DataFrame(data)

*# Print the DataFrame*

print(df2)

Name Exam 1 Exam 2 Final Exam

0 Karan 85 90 92

1 Deepa 70 80 85

2 Karthik 90 85 88

3 Chandan 75 70 75

4 Jeevan 95 92 96

In [28]:

**import** scipy.stats **as** stats

*# Perform one-way ANOVA*

f\_statistic, p\_value **=** stats**.**f\_oneway(df2['Exam 1'], df2['Exam 2'], df2['Final Exam'])

*# Print the F-statistic and p-value*

print("F-statistic:", f\_statistic)

print("p-value:", p\_value)

*# Calculate the maximum score for each student*

df2['Max Score'] **=** df2[['Exam 1', 'Exam 2', 'Final Exam']]**.**max(axis**=**1)

*# Sort the DataFrame by the maximum score in descending order*

sorted\_df2 **=** df2**.**sort\_values(by**=**'Max Score', ascending**=False**)

*# Get the student with the highest score*

highest\_score\_student **=** sorted\_df2**.**iloc[0]['Name']

*# Print the student with the highest score*

print("Student with the highest score:", highest\_score\_student)

F-statistic: 0.3236947791164659

p-value: 0.7295954253321559

Student with the highest score: Jeevan

Q-10. A factory produces light bulbs, and the probability of a bulb being defective is 0.05. The factory produces a large batch of 500 light bulbs. a. What is the probability that exactly 20 bulbs are defective? b. What is the probability that at least 10 bulbs are defective? c. What is the probability that at max 15 bulbs are defective? d. On average, how many defective bulbs would you expect in a batch of 500?

In [29]:

*#a*

*#a*

**import** scipy.stats **as** stats

**import** math

n **=** 500

p **=** 0.05

k **=** 20

mean **=** n **\*** p

std\_dev **=** math**.**sqrt(n **\*** p **\*** (1 **-** p))

probability\_20\_defective **=** stats**.**norm**.**cdf(k **+** 0.5, loc**=**mean, scale**=**std\_dev) **-** stats**.**norm**.**cdf(k **-** 0.5, loc**=**mean, scale**=**std\_dev)

print("Probability of exactly 20 bulbs being defective:", probability\_20\_defective)

Probability of exactly 20 bulbs being defective: 0.0483661403058116

In [30]:

*#b*

k **=** 10

probability\_at\_least\_10\_defective **=** stats**.**norm**.**sf(k **-** 0.5, loc**=**mean, scale**=**std\_dev)

print("Probability of at least 10 bulbs being defective:", probability\_at\_least\_10\_defective)

Probability of at least 10 bulbs being defective: 0.9992649777488151

In [ ]:

*#c*

k **=** 15

probability\_at\_most\_15\_defective **=** stats**.**norm**.**cdf(k **+** 0.5, loc**=**mean, scale**=**std\_dev)

print("Probability of at most 15 bulbs being defective:", probability\_at\_most\_15\_defective)

In [ ]:

*#d*

expected\_defective\_bulbs **=** n **\*** p

print("Expected number of defective bulbs:", expected\_defective\_bulbs)

Q-11. Given the data of a feature contributing to different classes

<https://drive.google.com/file/d/1mCjtYHiX--mMUjicuaP2gH3k-SnFxt8Y/view?usp=share_>

a. Check whether the distribution of all the classes are the same or not. b. Check for the equality of variance/ c. Which amount LDA and QDA would perform better on this data for classification and why. d. Check the equality of mean for between all the classes.

In [31]:

print("Column Names:", df**.**columns)

*# Check the distribution of all the classes (in this case, it's the before and after blood pressure measurements)*

class\_distribution **=** df[' Blood Pressure Before (mmHg)']**.**value\_counts()

print("Class Distribution:\n", class\_distribution)

*# Check for the equality of variance*

variance\_before **=** df[' Blood Pressure Before (mmHg)']**.**var()

variance\_after **=** df[' Blood Pressure After (mmHg)']**.**var()

print("Variance - Before: ", variance\_before)

print("Variance - After: ", variance\_after)

*# Perform LDA and QDA*

*# LDA assumes equal class covariance matrices and uses linear decision boundaries*

*# QDA relaxes the assumption of equal class covariance matrices and uses quadratic decision boundaries*

*# Check the equality of mean between all the classes*

mean\_before **=** df[' Blood Pressure Before (mmHg)']**.**mean()

mean\_after **=** df[' Blood Pressure After (mmHg)']**.**mean()

print("Mean - Before: ", mean\_before)

print("Mean - After: ", mean\_after)

Column Names: Index(['Patient ID', ' Blood Pressure Before (mmHg)',

' Blood Pressure After (mmHg)'],

dtype='object')

Class Distribution:

128 12

127 11

136 11

135 8

130 7

145 6

139 6

143 6

124 6

140 6

132 6

142 6

131 5

137 1

122 1

148 1

120 1

Name: Blood Pressure Before (mmHg), dtype: int64

Variance - Before: 43.53727272727271

Variance - After: 47.4448484848485

Mean - Before: 133.91

Mean - After: 128.36

Q-12. A pharmaceutical company develops a new drug and wants to compare its effectiveness against a standard drug for treating a particular condition. They conduct a study with two groups: Group A receives the new drug, and Group B receives the standard drug. The company measures the improvement in a specific symptom for both groups after a 4-week treatment period. a. The company collects data from 30 patients in each group and calculates the mean improvement score and the standard deviation of improvement for each group. The mean improvement score for Group A is 2.5 with a standard deviation of 0.8, while the mean improvement score for Group B is 2.2 with a standard deviation of 0.6. Conduct a t-test to determine if there is a significant difference in the mean improvement scores between the two groups. Use a significance level of 0.05. b. Based on the t-test results, state whether the null hypothesis should be rejected or not. Provide a conclusion in the context of the study.

In [32]:

**import** scipy.stats **as** stats

*# Given data*

meanA **=** 2.5

stdDevA **=** 0.8

nA **=** 30

meanB **=** 2.2

stdDevB **=** 0.6

nB **=** 30

*# Calculate the test statistic (t-value)*

squaredDevA **=** stdDevA **\*\*** 2

squaredDevB **=** stdDevB **\*\*** 2

t **=** (meanA **-** meanB) **/** ((squaredDevA **/** nA) **+** (squaredDevB **/** nB)) **\*\*** 0.5

*# Calculate the degrees of freedom*

df **=** nA **+** nB **-** 2

*# Calculate the critical t-value for a two-tailed test at 0.05 significance level*

critical\_t **=** stats**.**t**.**ppf(0.025, df)

*# Compare the calculated t-value with the critical t-value*

**if** t **>** critical\_t:

conclusion **=** "Reject the null hypothesis. There is a significant difference in the mean improvement scores between Group A and Group B."

**else**:

conclusion **=** "Fail to reject the null hypothesis. There is no significant difference in the mean improvement scores between Group A and Group B."

*# Print the conclusion*

print(conclusion)

Reject the null hypothesis. There is a significant difference in the mean improvement scores between Group A and Group B.