FIFA Case Study Report

Great lakes institute of management

PGPDSE-FT Online April 23, Group #3

Submitted by:

Pranit Jadhav	
Richa Gupta	
Raghavendra Abhishek	Z60SKJ8ADG
Harsh Chaudhary	AHZUJOTP1Y

Table of content:

What problem are you solving?

BMI of young adults:

In the given case study, we are solving a problem related to Body Mass Index of 50 young adults. We have converted the given data into a list and performed various operations on the data to retrieve specific details.

The tasks involved are:

Q1. Compute the mean, median and the mode of the data
Q2. Compute the range, variance and standard deviation of BMI
Q3. Find the mean deviation for the data.
Q4. Calculate the Pearson coefficient of skewness and comment on the skewness of the data.
Q5. Count the number of data values that fall within one standard deviation of
the mean. Compare this with the answer from Chebyshev's Theorem.
Q6. Find the three quartiles and the interquartile range (IQR).
Q7. Are there any outliers in the data set?
Q8. Draw a boxplot of the dataset to confirm.
Q9. Find the percentile rank of the datapoint 25.0.
Q10. What is the probability that a young adult has a BMI above 25.0?
Q11. Create a frequency distribution for the data and visualize it appropriately
Q12. Create a probability distribution of the data and visualize it appropriately.
Q13. What is the shape of the distribution of this dataset?
Q14. Treat this dataset as a binomial distribution where *p* is the probability that a young adult has a BMI above 25.0. What is the probability that out of a random sample of 10 young adults exactly 6 are having BMI greater than 25.0?
Q15. A study claims that 40% of all young adults have BMI greater than 25.0. Using the Normal approximation of a Binomial distribution, find the probability that in a random sample of 100 young adults exactly 50 of them will have a BMI is greater than 25.0.
Q16. Compute a 95% Confidence Interval for the true BMI of the population of young adults using appropriate distribution. (State reasons as to why did you use a *z* or *t* distribution)
Q17. A data scientist wants to estimate with 95% confidence the proportion of young adults having BMI greater than 25.0. A recent study showed that 40% of all young adults have BMI greater than 25.0. The data scientist wants to be accurate within 2% of the true proportion. Find the minimum sample size necessary.

Q18. The same data scientist wants to estimate the true proportion of young adults having BMI greater than 25.0. She wants to be 90% confident and accurate within 5% of true proportion. Find the minimum sample size necessary.
Q19. A researcher claims that currently 55% of all young adults have BMI greater than 25. Test his claim with an alpha =0.05 if out of a random sample
of 30 CEOs only 20 are having BMI above 25
Q 20. A data scientist is researching the hypothesis that there is no difference between BMI of public vs private school students. So, he collects data from the two schools and finds that the proportion of public-school students whose BMI is above
25.0 is 31.8 % vs Private school students is 38.7 %. Suppose the data scientist got these values after interviewing 500 students at each school.
a. What hypothesis would he use to compare the proportions of students having BMI greater than 25.0 among both the schools.
b. What are critical values to be used?
c. What statistical test will be used to compare these proportions?
d. Complete the test and obtain the P-value.
e. Summarize his conclusion based on the P-value.

FIFA Case Study:

In the given case study, we are solving problem related to a new football club named 'Brussels United FC' that has just been inaugurated, which does not have a team yet. The team is looking to hire players for their roster. Management wants to make such decisions using a data-based approach. The data contains details for over 25490 players playing in various football clubs in Europe. We performed various operations on the data to retrieve specific details.

The tasks involved are:

1. Import the necessary libraries and read the data.
2. Drop any columns that you deem unnecessary for analysis.

4. Check the duplicate records and do appropriate treatments.
5. Check the variation of the features.

If you are performing variance and standard deviation. Kindly explain why the variances of the variables are higher than the standard deviation.
Also, explain which one tells the exact variation of the features.
Based on this analysis decide which feature is not needed.

6. Check for missing values and do imputations where necessary.
Note: Do the appropriate
imputation based on the distribution.

1. Test statistically whether the Left-hand player’s overall rating is higher than the Right-hand overall score. Alpha = 0.05
Before checking the test, kindly make sure data is normally distributed.
2. Does the age factor affect the player’s potential? Check the claim that the players who are greater than 35, their potential will be lesser than those whose age is less than 35. Alpha = 0.05
3. Use the statistical test to check the relationship between the Preferred Foot and Position with the 99% confident interval.
4. Does the International Reputation cause a significant effect on players' Wages? Check the claim with a 0.04 significance level. Check the Normality of data before the actual test.
5. Check the claim that the median wages of under top 20 players are lesser than or equal to 25000. Test the claim with a 0.05 % significance level. Check the data is normally distributed or not before the testing the claim statistically.

How did you solve the problem?

To solve the problems in the given case studies, a systematic approach was followed that involves several key steps:

Data Collection:
Imported data into python using libraries like ‘Pandas’ to create data frames.

Data Inspection:

Checked the first few rows of the data to get an initial sense of its structure.

Data Cleaning:

Handled missing values by using methods like ‘dropna()’,’fillna()’ or imputation.

Removed duplicates using ‘drop duplicates()’.

Data Visualization:

Visualised the data to gain insights using libraries like ‘matplotlib', 'Seaborn’.

Data Analysis:

Performed summary statistics and aggregation to understand the data better.

Hypothesis Testing:

Conducted statistical tests to validate hypothesis.

Correlation Analysis:

Used correlation plots to identify relationships between variables.

What were the steps you took to solve?

Questions with code and relevant outputs

Q1. Compute the mean, median and the mode of the data

```
#Ques: 1
import statistics as stats

bmiData= [17.5, 18.0, 36.8, 31.7, 31.7, 17.3, 24.3, 47.7, 38.5, 17.0, 23.7, 16.5, 25.1, 17.4, 18.0, 37.6, 19.7,
          21.4, 28.6, 21.6, 19.3, 20.0, 16.9, 25.2, 19.8, 25.0, 17.2, 20.4, 20.1, 29.1, 19.1, 25.2, 23.2, 25.9,
          24.0, 41.7, 24.0, 16.8, 26.8, 31.4, 16.9, 17.2, 24.1, 35.2, 19.1, 22.9, 18.2, 25.4, 35.4, 25.5]

mean_value = stats.mean(bmiData);
median_value = stats.median(bmiData);
mode_value = stats.mode(bmiData);
print("Mean is :", mean_value)
print("Median is :", median_value)
print("Mode is :", mode_value)
```

```
Mean is : 24.422
Median is : 23.45
Mode is : 18.0
```

- We have been given a data about ‘BMI’ scores. We nedd to find their mean, median and mode.
- We import statistics library and name it as stats.
- We create variables mean value, median value and mode value.
- And assign them value by deriving it using ‘stats.mean()’, ‘stats.median()’ and stats.mode() respectively.
- Finally, we display the values using print function.

Q2. Compute the range , variance and standard deviation of BMI ¶

```
#Ques: 2
import numpy as np

bmi_range = max(bmiData) - min(bmiData)
bmi_variance = np.var(bmiData)
bmi_stddev = np.std(bmiData)
print("Range of BMI:", bmi_range)
print("Variance of BMI:", bmi_variance)
print("Standard Deviation of BMI:", bmi_stddev)
```

```
Range of BMI: 31.200000000000003
Variance of BMI: 53.469716000000005
Standard Deviation of BMI: 7.312298954501245
```

- Here we need to find the range, variance and standard deviation of the BMI data.
- We import the NumPy library and give it an alias np.
- We create following variables ‘bmi_range’, ‘bmi_variance’ and ‘bmi_stddev’.
- For ‘bmi_range’ we got the range by subtracting the maximum value of the data with the minimum value of the data. (Used the max and min function.)

- For the ‘bmi_variance’ and ‘bmi_stddev’ we used the np.var() and np.std() functions respectively.
- Finally, we displayed the answers using the print function.

Q3. Find the mean deviation for the data

```
#Ques: 3
mean_deviation = sum(abs(x - mean_value) for x in bmiData) / len(bmiData)
print("Mean Deviation:", mean_deviation)
```

```
Mean Deviation: 5.6423999999999985
```

- Here we need to find the average of standard deviation.
- We used generator expression to find the variation for mean value.
- After sum we divide it with the length of the data to obtain the average.
- Finally, we display the average using print function.

Q4. Calculate the Pearson coefficient of skewness and comment on the skewness of the data

```
#Ques: 4
skewness = (3 * (mean_value - median_value)) / bmi_stddev
print("Pearson Coefficient of Skewness:", skewness)
```

```
Pearson Coefficient of Skewness: 0.3987801945932471
```

- Here we find the Pearson coefficient of skewness.
- Here we use already found mean, median and standard deviation of the data.
- Used the formula and displayed the answer using print function.

Q5. Count the number of data values that fall within one standard deviation of the mean.

Compare this with the answer from Chebyshev’s Theorem.

```
#Ques: 5
lower_bound = mean_value - bmi_stddev
upper_bound = mean_value + bmi_stddev
count_within_stddev = sum(lower_bound <= x <= upper_bound for x in bmiData)
print("Number of data values within one standard deviation:", count_within_stddev)
```

```
Number of data values within one standard deviation: 38
```

- Here we found the count of values that lie within one standard deviation of mean.
- We used the generator expression for x in bmi_data within the sum function.
- Finally, we display the answer.

Q6. Find the three quartiles and the interquartile range (IQR).

```
#Ques: 6
sorted_data = np.sort(bmiData)
Q1 = np.percentile(sorted_data, 25)
Q2 = np.median(sorted_data)
Q3 = np.percentile(sorted_data, 75)
IQR = Q3 - Q1
print("Q1 (25th percentile):", Q1)
print("Q2 (median, 50th percentile):", Q2)
print("Q3 (75th percentile):", Q3)
print("Interquartile Range (IQR):", IQR)
```

```
Q1 (25th percentile): 18.425
Q2 (median, 50th percentile): 23.45
Q3 (75th percentile): 26.575
Interquartile Range (IQR): 8.149999999999999
```

- Here we have used percentile function and the median function from NumPy Library.
- We have found the required quartiles and found the IQR by subtracting them.
- Finally, we display our answer.

Q7. Are there any outliers in the data set ?

```
#Ques: 7
lower_bound_outliers = Q1 - 1.5 * IQR
upper_bound_outliers = Q3 + 1.5 * IQR
outliers = [x for x in bmiData if x < lower_bound_outliers or x > upper_bound_outliers]
print("Potential outliers:", outliers)
```

```
Potential outliers: [47.7, 41.7]
```

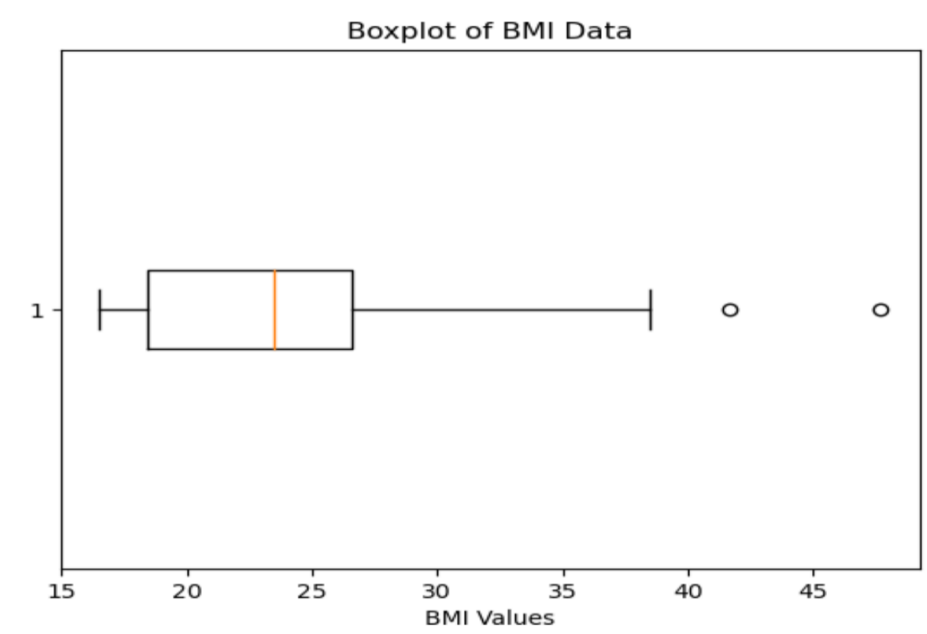
- Here we are tasked with finding the outliers the data.
- We define the upper and lower limits for outliers.
- Using list comprehension, we create a list of outliers which lie beyond our lower and upper limits.
- Finally, we display our answer.

Q8. Draw a boxplot of the dataset to confirm

```
#Ques: 8
import matplotlib.pyplot as plt

plt.boxplot(bmiData, vert=False)
plt.title('Boxplot of BMI Data')
plt.xlabel('BMI Values')
plt.show()
```

- Here we use the ‘Matplotlib.pyplot’ library and gave it alias ‘plt’.
- We use the boxplot function.
- And the outliers are visible.



Q9. Find the percentile rank of the datapoint 25.0.

```
#Ques: 9
sorted_data1 = sorted(bmiData)
index_of_25 = sorted_data1.index(25.0)
percentile_rank = (index_of_25 + 1) / len(sorted_data1) * 100
print("Percentile Rank of 25.0:", percentile_rank)
```

```
Percentile Rank of 25.0: 62.0
```

- Here we sort the data.
- Obtain its index number
- Using the percentile rank formula. We find our answer.
- The rank is then displayed.

Q10. What is the probability that a young adult has a BMI above 25.0?

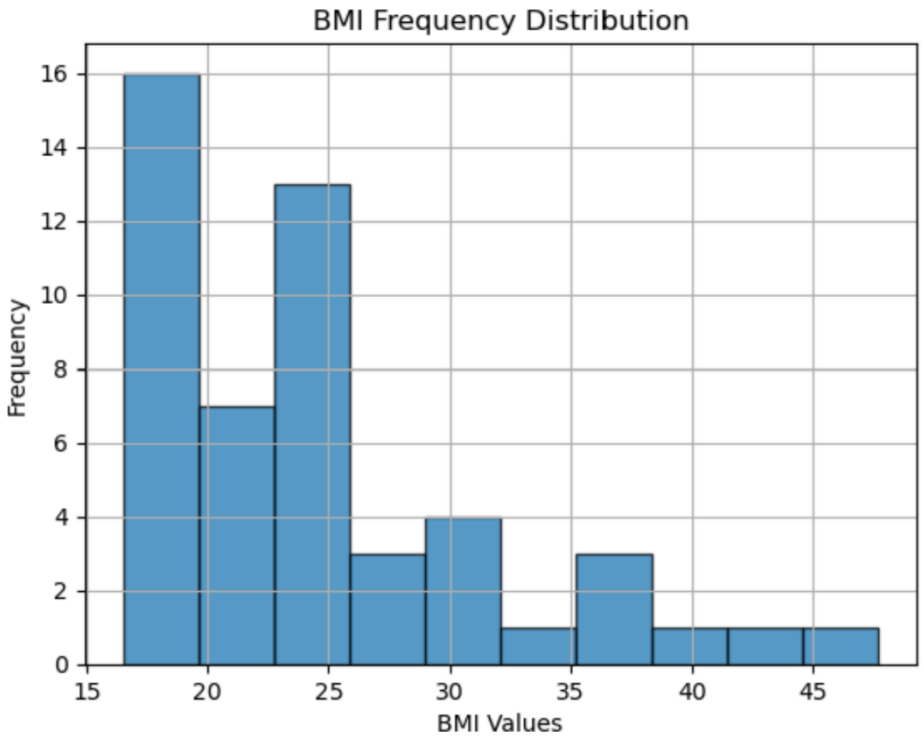
```
#Ques: 10
count_above_25 = sum(x > 25.0 for x in bmiData)
probability_above_25 = count_above_25 / len(bmiData)
print("Probability of BMI above 25.0:", probability_above_25)
```

```
Probability of BMI above 25.0: 0.38
```

- Here we generator expression within count function to acquire the count of such instances.
- In dividing it with the total length of observations we get the required probability.
- Finally, we display the answer.

Q11. Create a frequency distribution for the data and visualize it appropriately

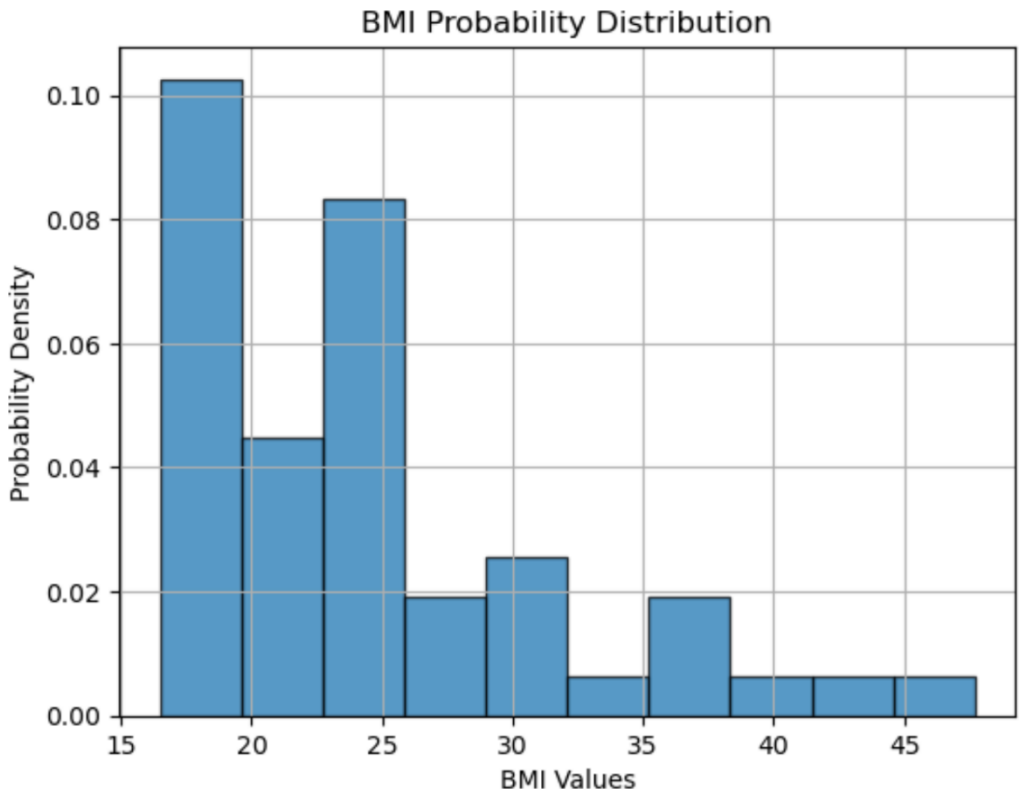
```
#Ques: 11
plt.hist(bmiData, bins=10, edgecolor='k', alpha=0.75)
plt.title('BMI Frequency Distribution')
plt.xlabel('BMI Values')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



- Here we used hist function to obtain the distribution.
- As we can see the data is not normal. It is skewed.

Q12. Create a probability distribution of the data and visualize it appropriately

```
: #Ques: 12
plt.hist(bmiData, bins=10, density=True, edgecolor='k', alpha=0.75)
plt.title('BMI Probability Distribution')
plt.xlabel('BMI Values')
plt.ylabel('Probability Density')
plt.grid(True)
plt.show()
```



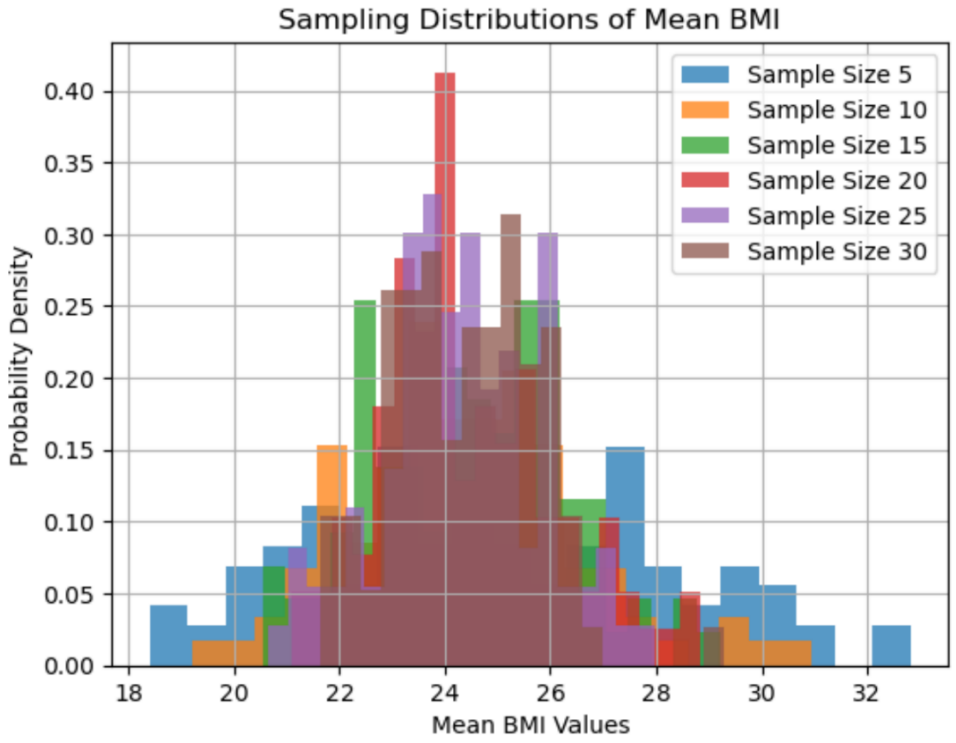
- Here we can observe that the data is not normally distributed.

Q13. What is the shape of the distribution of this dataset?

```
#Ques: 13
plt.hist(bmiData, bins=20, density=True, edgecolor='k', alpha=0.75)
plt.title('Original BMI Distribution')
plt.xlabel('BMI Values')
plt.ylabel('Probability Density')
plt.grid(True)
plt.show()
np.random.seed(42)
sample_sizes = [5, 10, 15, 20, 25, 30]
for sample_size in sample_sizes:
    sampling_distribution = []
    for _ in range(100):
        sample = np.random.choice(bmiData, size=sample_size, replace=True)
        sample_mean = np.mean(sample)
        sampling_distribution.append(sample_mean)

    plt.hist(sampling_distribution, bins=20, density=True, alpha=0.75, label=f'Sample Size {sample_size}')

plt.title('Sampling Distributions of Mean BMI')
plt.xlabel('Mean BMI Values')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
```



- Here we can observe that as the data size increases the we get more and more naturally distributed samples.
- Clearly showing the Central Limit Theorem.

Q14. Treat this dataset as a binomial distribution where p is the probability that a young adult has a BMI above 25.0. What is the probability that out of a random sample of 10 young adults exactly 6 are having BMI greater than 25.0?

```
#Ques: 14
import math

p = sum(x > 25.0 for x in bmiData) / len(bmiData)
n = 10
k = 6
binomial_prob = math.comb(n, k) * (p**k) * ((1 - p)**(n - k))
print("Probability that exactly 6 out of 10 young adults have BMI > 25.0:", binomial_prob)
```

```
Probability that exactly 6 out of 10 young adults have BMI > 25.0: 0.09343027613767894
```

- Here we used math library comb function to find the combination for the values of p, n and k.
- The required probability is approx. 9%

Q15. A study claims that 40% of all young adults have BMI greater than 25.0. Using the Normal approximation of a Binomial distribution, find the probability that in a random sample of 100 young adults exactly 50 of them will have will have a BMI is greater than 25.0.

```
#Ques: 15
n = 100
p = 0.40
x = 50
mu = n * p
sigma = math.sqrt(n * p * (1 - p))
x_lower = x - 0.5
x_upper = x + 0.5
z_lower = (x_lower - mu) / sigma
z_upper = (x_upper - mu) / sigma
cumulative_prob = (math.erf(z_upper / math.sqrt(2)) - math.erf(z_lower / math.sqrt(2))) / 2
print("Probability that exactly 50 out of 100 young adults have BMI > 25.0:", cumulative_prob)
```

```
Probability that exactly 50 out of 100 young adults have BMI > 25.0: 0.010195382790507634
```

- Here we used normal approximation of binomial distribution.
- And found the probability.

Q16. Compute a 95% Confidence Interval for the true BMI of the population of young adults using appropriate distribution.

```
#Ques: 16
import scipy.stats as stats

sample_mean = np.mean(bmiData)
sample_std = np.std(bmiData, ddof=1)

n = len(bmiData)
alpha = 0.05
t_critical = stats.t.ppf(1 - alpha / 2, df=n - 1)
standard_error = sample_std / np.sqrt(n)
margin_of_error = t_critical * standard_error
lower_bound = sample_mean - margin_of_error
upper_bound = sample_mean + margin_of_error

print(f"95% Confidence Interval for BMI: ({lower_bound:.2f}, {upper_bound:.2f})")
```

```
95% Confidence Interval for BMI: (22.32, 26.52)
```

- Here to find the confidence interval we find the critical value.
- We find the margin of error using critical values.
- Adding and subtracting from the mean we found the confidence interval.

Q17. A data scientist wants to estimate with 95% confidence the proportion of young adults having BMI greater than 25.0. A recent study showed that 40% of all young adults have BMI greater than 25.0. The data scientist wants to be accurate within 2% of the true proportion. Find the minimum sample size necessary

```
#Ques: 17
from scipy.stats import norm

confidence_level = 0.95
estimated_proportion = 0.40
margin_of_error = 0.02
z = norm.ppf(1 - (1 - confidence_level) / 2)
minimum_sample_size = (z**2 * estimated_proportion * (1 - estimated_proportion)) / (margin_of_error**2)
minimum_sample_size = np.ceil(minimum_sample_size)
print("Minimum Sample Size:", minimum_sample_size)
```

```
Minimum Sample Size: 2305.0
```

Q18. The same data scientist wants to estimate the true proportion of young adults having BMI greater than 25.0. She wants to be 90% confident and accurate within 5% of true proportion. Find the minimum sample size necessary.

```
#Ques: 18
confidence_level = 0.90
margin_of_error = 0.05
z = norm.ppf(1 - (1 - confidence_level) / 2)
estimated_proportion = 0.5
minimum_sample_size = (z**2 * estimated_proportion * (1 - estimated_proportion)) / (margin_of_error**2)
minimum_sample_size = np.ceil(minimum_sample_size)
print("Minimum Sample Size:", minimum_sample_size)
```

```
Minimum Sample Size: 271.0
```

Q19. A researcher claims that currently 55% of all young adults have BMI greater than 25 . Test his claim with an alpha =0.05 if out of a random sample ¶ of 30 CEOs only 20 are having BMI above 25

```
#Ques: 19
from statsmodels.stats.proportion import proportions_ztest
n = 30
x = 20
p_null = 0.55
z_stat, p_value = proportions_ztest(x, n, p_null, alternative='two-sided')
alpha = 0.05

if p_value < alpha:
    print(f"Reject the null hypothesis. There is evidence to suggest that the true proportion is not 55%.")
else:
    print("Fail to reject the null hypothesis. There is no strong evidence to suggest a difference in the proportion.")
```

```
Fail to reject the null hypothesis. There is no strong evidence to suggest a difference in the proportion.
```

```
Q 20. A data scientist is researching the hypothesis that there is no difference
between BMI of public vs private schools students. So he collects data from the two
schools and finds that the proportion of public school students whose BMI is above
25.0 is 31.8 % vs Private school students is 38.7 %. Suppose the data scientist got
these values after interviewing 500 students of each school.
a. What hypothesis would he use to compare the proportions of students having
BMI greater than 25.0 among both the schools.
b. What are critical values to be used?
c. What statistical test will be used to compare these proportions ?
4
d. Complete the test and obtain the P-value.
e. Summarize his conclusion based on the P-value.
```

```
a.null hypothesis the bmi of public school and private school is same.
  alternate hypothesis the bmi of public school and private school is  not same.
  prop of public school students with bmi above 25 31.8% as p1
  prop of private school students with bmi above 25 38.7% p2
  H0: p1=p2
  Ha: p1!=p2
b.p1=31.8, p2=38.7, we use sample sizes for both as 100 or we can use,
  p1=(31.8/100)*500 and p1=0.387*500 with sample size 500 for each.
c.we will use 2 proportion z test
  *   Binomally distributed population - Yes, a product is either defective or non-defective.
  *   Random sampling from the population - Yes, we are informed that the collected sample is a simple       random sample.
```

```
# d
# import the required fuction
from statsmodels.stats.proportion import proportions_ztest
import numpy as np
# set the proportions.
proportion_count = np.array([31.8, 38.7])

# set the sample sizes
nobs = np.array([100, 100])

# find the p-value
test_stat, p_value = proportions_ztest(proportion_count, nobs)
print('The p-value is ' + str(p_value))


#e. since the p value is greater than 0.5 we fail to reject the null hypothesis.
# the bmi for both school is same.
```

```
The p-value is 0.3071331392350505
```

- Here we are doing a two-proportion z test.
- Since the p value is greater than 0.05 we fail to reject the null hypothesis.
- Hence the BMI of a public and a private school is same.

STATS

1. Test statistically whether the Left-hand player's overall rating is higher than the Right-hand overall score. Alpha = 0.05 Before checking the test, Kindly make sure data is normally distributed.

```
df['Preferred Foot'].value_counts()
```

```
Right    10483
Left      3174
Name: Preferred Foot, dtype: int64
```

```
df_r = df[(df['Preferred Foot']=='Right')]['Overall']
df_l = df[(df['Preferred Foot']=='Left')]['Overall']
```

```
#step1 :
#H0 : mu1-mu2 <= 0        #mu1 stands for df_1 (left foot player)
#H1 : mu1 -mu2 > 0        #mu2 stands for df_2 (right foot player)
```

```
# step2:
stats.shapiro(df['Overall'])
```

```
ShapiroResult(statistic=0.9966169595718384, pvalue=3.6861401483016806e-17)
```

```
#since the p_value is less than the alpha so the data is not normal so we go for mannwhitneyu test.
```

```
#step3
#test_statistic
alpha = 0.05
stat,p_value = stats.mannwhitneyu(x = df_l,y = df_r,alternative = 'greater')
print('The P_value is ',p_value)
```

```
The P_value is  8.985563171719411e-07
```

```
#step 4
'We Reject the Null Hypothesis H0' if p_value < alpha else 'We fail to Reject the Null Hypothesis H0'
```

```
'We Reject the Null Hypothesis H0'
```

Therefore left footed players overall rating is higher than the right footed player overall rating

-

- We performed a Shapiro test to find if the data is normally distributed.
- Since it s not normally distributed we performed manwhitneyu test.
- Thus we can say that left foot players have a higher rating.

2. Does the age factor affect the player's potential? Check the claim that the players who are greater than 35, their potential will be lesser than those whose age is less than 35. Alpha = 0.05

```
df_a1 = df[(df['Age']>=35)]['Potential']
df_a2 = df[(df['Age']<35)]['Potential']

n1 = len(df_a1)                #df_a1 players potential greater than 35
n2 = len(df_a2)                #df_a2 stands fro players potentail less than 35

print('No of players potential greater than eqaul to age 35',n1)
print('No of players potential less than age 35',n2)
```

```
No of players potential greater than eqaul to age 35 346
No of players potential less than age 35 13311
```

```
##step1 :
#H0 : mu1-mu2 >= 0
#H1 : mu1 -mu2 < 0
```

```
## step 2
#checking for normality
stats.shapiro(df['Potential'])
```

```
ShapiroResult(statistic=0.9933071136474609, pvalue=1.190773703517539e-24)
```

```
#p_value is lesseer than the alpha  so the data is not nornal
```

```
alpha = 0.05
stat,p_value = stats.mannwhitneyu(x = df_a1,y = df_a2,alternative = 'less')
print('The P_value is ',p_value)
```

```
The P_value is  1.115069496276631e-20
```

```
#step 4
'We Reject the Null Hypothesis H0' if p_value < alpha else 'We fail to Reject the Null Hypothesis H0'
```

```
'We Reject the Null Hypothesis H0'
```

Therefore, players who are greater than 35, their potential will be lesser than those whose age is less than 35

- Here we try to find if the potential of a player is dependent on it's age.
- We get two data sets to compare .
- We perform Shapiro test to test normality.

- Since it is not normally distributed, we perform Manwhitneyu test .
- We can say that the players potential is dependent on his age.

3. Use the statistical test to check the relationship between the Preferred Foot and Position with the 99% confident interval.

```
a = pd.crosstab(df['Preferred Foot'],df['Position'])
a
```

Position	CAM	CB	CDM	CF	CM	GK	LAM	LB	LCB	LCM	...	RB	RCB	RCM	RDM	RF	RM	RS	RW	RWB	ST
Preferred Foot																					
Left	202	253	99	11	195	159	7	863	198	75	...	10	26	30	19	3	189	29	74	2	210
Right	540	1074	622	43	874	1372	7	115	286	216	...	963	452	269	164	9	650	137	200	63	1412

2 rows × 27 columns

```
#step1 :
#H0 : There is no relation between preferred foot and postion
#H1 : There is a relation between preferred foot and position
```

```
#step2 :
#performing chi-square test.
chi2, p, dof, expected = stats.chi2_contingency(a)
```

```
#step3
alpha = 0.01
critical_value = stats.chi2.ppf(1-alpha, dof)

# Print the results
print("Chi-squared test statistic:", chi2)
print("P-value:", p)
print("Degrees of freedom:", dof)
print("Expected frequencies:", expected)
print("Critical value:", critical_value)
```

```
Chi-squared test statistic: 3394.5278485109557
P-value: 0.0
Degrees of freedom: 26
Expected frequencies: [[ 172.44695028  308.40579922  167.56637622   12.55004759  248.44446072
   355.81709014    3.25371604  227.29530644  112.48561177   67.63081204
    43.92516658    3.02130775  190.8072051    36.02328476   65.07432086
    13.94449733    3.95094091  226.13326499  111.09116204   69.49007835
    42.53071685    2.78889947  194.99055429   38.57977594   63.67987113
    15.10653877  376.96624442]
 [ 569.55304972 1018.59420078  553.43362378   41.44995241  820.55553928
  1175.18290986   10.74628396  750.70469356  371.51438823  223.36918796
   145.07483342    9.97869225  630.1927949   118.97671524  214.92567914
    46.05550267   13.04905909  746.86673501  366.90883796  229.50992165
   140.46928315    9.21110053  644.00944571  127.42022406  210.32012887
    49.89346123 1245.03375558]]
Critical value: 45.64168266628317
```

```
# step4:
if chi2 > critical_value:
    print("Reject the null hypothesis, there is a significant relationship between Preferred Foot and Position")
else:
    print("Fail to reject the null hypothesis, there is no significant relationship between Preferred Foot and Position")
```

```
Reject the null hypothesis, there is a significant relationship between Preferred Foot and Position
```

- Here we are given two variables we are asked is they are dependent.
- We performed chi square test and conclude that there is a correlation between these two variables.

4. Does the International Reputation cause a significant effect on players' Wages? Check the claim with a 0.04 significance level. Check the Normality of data before the actual test.

```
df['International Reputation'].value_counts()
```

```
1.0    12464
2.0      928
3.0      224
4.0       37
5.0        4
Name: International Reputation, dtype: int64
```

```
#step 1
stats.shapiro(df['Wage'])              #p_value is greater less than alpha the data is not normal
```

```
ShapiroResult(statistic=0.38748836517333984, pvalue=0.0)
```

```
#step2
#H0 : there is a significant effect on players Wage cause by Ineternational Reputation
#H1 : there is no significant effect on players Wage cause by Ineternational Reputation
```

```
a= df[(df['International Reputation']==1.0)]['Wage']
b= df[(df['International Reputation']==2.0)]['Wage']
c= df[(df['International Reputation']==3.0)]['Wage']
d= df[(df['International Reputation']==4.0)]['Wage']
e= df[(df['International Reputation']==5.0)]['Wage']
```

```
#step3
#test statistic

alpha = 0.04
stat,p_value = stats.f_oneway(a,b,c,d,e)
print('The test_statistic is',stat)
print('The p_value is',p_value)
```

```
The test_statistic is 3485.1081736616075
The p_value is 0.0
```

```
#step 4
'We Reject the Null Hypothesis H0' if p_value < alpha else 'We fail to Reject the Null Hypothesis H0'
```

```
'We fail to Reject the Null Hypothesis H0'
```

Therefore "there is a significant effect on players Wage cause by Ineternational Reputation"

- Here we are performing the one way
- We are trying to find if there is any statistically significant difference in the wages for players with different international rating.

5. Check the claim that the median wages of under top 20 players are lesser than or equal to 25000. Test the claim with a 0.05 % significance level. Check the data is normally distributed or not before the testing the claim statistically.

```
#step1
#H0 mu <= 25000
#H1 mu>25000

alpha = 0.05
mu = 25000
n = len(df_top20['Wage'])
```

```
#step2 :
#checking the normality of the data

stats.shapiro(df_top20['Wage'])
```

```
ShapiroResult(statistic=0.9450737833976746, pvalue=0.29842087626457214)
```

```
#the p_value is greater than the alpha so the data is normal.
```

```
#step 3:
stat,p_value = stats.ttest_1samp(a = df_top20['Wage'],popmean = 25000,alternative = 'greater')
print('The test_statistic is ',stat)
print('The p_value is ',p_value)
```

```
The test_statistic is  -1037.3394335814062
The p_value is  1.0
```

- Here we found that the top 20 players have a wage around 25000.
- We have performed 1 sample t-test.

What did you learn from solving and how do you plan on using it in the future?

By solving the questions, we have gained knowledge regarding the following concepts:

- Data Distribution
- Central Tendency
- Data Spread
- Data Relationships
- Outliers
- Data Patterns
- Missing Data
- Data Visualizations
- Data Quality
- Feature Selection
- Hypothesis generation
- Data Preprocessing
- Decision making

Exploratory data analysis and Statistics are the two crucial steps that helps us to gain a deep understanding of data, allowing us to make informed choices

throughout the data analysis process and ultimately draw meaningful insights and conclusions.