DSA0402 - Fundamentals of Data Science - Lab Questions

1. **Scenario:** You are working on a project that involves analyzing student performance data for a class of 32 students. The data is stored in a NumPy array named `student_scores`, where each row represents a student and each column represents a different subject. The subjects are arranged in the following order: Math, Science, English, and History. Your task is to calculate the average score for each subject and identify the subject with the highest average score.

Question: How would you use NumPy arrays to calculate the average score for each subject and determine the subject with the highest average score? Assume 4x4 matrix that stores marks of each student in given order.

```
import numpy as np
import csv

data = []
with open("C:/Users/bhara/Downloads/studies/work/New
folder/FODS/students_score.csv", 'r') as csvfile:
    csvreader = csv.reader(csvfile)
    for row in csvreader:
        data.append(row)
data = np.array(data[1:], dtype=float)  # Assuming the data is numerical


avg = np.mean(data, axis=0)


formatted_avg = ["{:.2f}".format(score) for score in avg]

# Print average scores
print("Average score:", formatted_avg)

# Find the highest average score
high = np.max(avg)
print("Highest average score:", high)

'''
sample output
Average score: ['96.38', '79.80', '65.70', '60.33']
Highest average score: 96.375
'''
```

2. **Scenario:** You are a data analyst working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a NumPy array.

Question: How would you find the average price of all the products sold in the past month? Assume 3x3 matrix with each row representing the sales for a different product

```
import numpy as np
import csv


data = []
with open("C:/Users/bhara/Downloads/studies/work/New folder/FODS/sales_data.csv",
'r') as csvfile:
    csvreader = csv.reader(csvfile)
    for row in csvreader:
        data.append(row)

sales_data = np.array(data[1:], dtype=float)

avg = np.mean(sales_data)

print("The average sold in the past month: {:.2f}".format(avg))


'''
sample output
The average sold in the past month: 56527.34
'''
```

3. **Scenario:** You are working on a project that involves analyzing a dataset containing information about houses in a neighborhood. The dataset is stored in a CSV file, and you have imported it into a NumPy array named `house_data`. Each row of the array represents a house, and the columns contain various features such as the number of bedrooms, square footage, and sale price.

Question: Using NumPy arrays and operations, how would you find the average sale price of houses with more than four bedrooms in the neighborhood?

```
import numpy as np
data = np.array([
    [3, 1200, 250000],
    [4, 1500, 300000],
    [5, 1800, 350000],
    [4, 1600, 280000],
    [5, 2000, 400000],
    [6, 2200, 420000],
    [3, 1400, 260000],
    [4, 1700, 310000],
    [5, 1900, 370000],
    [4, 1800, 320000]
])
column = data[:, 0]
houses = data[column > 4]
ave = np.mean(houses[:, 2])
```

```
print("Average sales with four bedrooms:", ave)


'''
sample output
Average sales with four bedrooms: 385000.0
'''
```

4. **Scenario:** You are working on a project that involves analyzing the sales performance of a company over the past four quarters. The quarterly sales data is stored in a NumPy array named `sales_data`, where each element represents the sales amount for a specific quarter. Your task is to calculate the total sales for the year and determine the percentage increase in sales from the first quarter to the fourth quarter.

Question: Using NumPy arrays and arithmetic operations calculate the total sales for the year and determine the percentage increase in sales from the first quarter to the fourth quarter?

```
import numpy as np


sales_data = np.array([250000, 300000, 350000, 400000])


total_sales = np.sum(sales_data)


percentage_increase = ((sales_data[3] - sales_data[0]) / sales_data[0]) * 100

print("Total sales for the year:", total_sales)
print("Percentage increase in sales from Q1 to Q4:", percentage_increase, "%")


'''
sample output
Total sales for the year: 1300000
Percentage increase in sales from Q1 to Q4: 60.0 %
'''
```

5. **Scenario:** You are a data analyst working for a car manufacturing company. As part of your analysis, you have a dataset containing information about the fuel efficiency of different car models. The dataset is stored in a NumPy array named `fuel_efficiency`, where each element represents the fuel efficiency (in miles per gallon) of a specific car model. Your task is to calculate the average fuel efficiency and determine the percentage improvement in fuel efficiency between two car models.

Question: How would you use NumPy arrays and arithmetic operations to calculate the average fuel efficiency and determine the percentage improvement in fuel efficiency between two car models?

```
import numpy as np

fuel_efficiency = np.array([25.5, 30.2, 22.8, 28.6, 33.1, 19.7, 24.8, 27.4, 31.5,
26.9])

average_fuel_efficiency = np.mean(fuel_efficiency)
print(f"Average Fuel Efficiency: {average_fuel_efficiency:.2f} ")

model_a_efficiency = 22.8  # Example fuel efficiency for car model A
model_b_efficiency = 28.6  # Example fuel efficiency for car model B

percentage_improvement = ((model_b_efficiency - model_a_efficiency) /
model_a_efficiency) * 100
print(f"Percentage Improvement: {percentage_improvement:.2f}%")


'''
sample output
Average Fuel Efficiency: 27.05 mpg
Percentage Improvement: 25.44%
'''
```

6. **Scenario:** You are a cashier at a grocery store and need to calculate the total cost of a customer's purchase, including applicable discounts and taxes. You have the item prices and quantities in separate lists, and the discount and tax rates are given as percentages. Your task is to calculate the total cost for the customer.

Question: Use arithmetic operations to calculate the total cost of a customer's purchase, including discounts and taxes, given the item prices, quantities, discount rate, and tax rate?

```
prices = [2.5, 3.0, 1.75, 4.5, 2.0]
quantities = [5, 3, 2, 4, 6]
discount = 10
tax = 8

before_discount = sum(price * quantity for price, quantity in zip(prices,
quantities))

discount_amount = (discount / 100) * before_discount
```

```
after_discount = before_discount - discount_amount

tax_amount = (tax/ 100) * after_discount

final = after_discount + tax_amount

print(f"Total cost before discounts: {before_discount:.2f}")
print(f"Discount amount: ${discount_amount:.2f}")
print(f"Total cost after discounts: {after_discount:.2f}")
print(f"Tax amount: ${tax_amount:.2f}")
print(f"Final total cost: ${final:.2f}")


'''
sample output
Total cost before discounts: 55.00
Discount amount: $5.50
Total cost after discounts: 49.50
Tax amount: $3.96
Final total cost: $53.46
'''
```

7. **Scenario:** You are working as a data analyst for an e-commerce company. You have been given a dataset containing information about customer orders, stored in a Pandas DataFrame named `order_data`. The DataFrame has columns for customer ID, order date, product name, and order quantity. Your task is to analyze the data and answer specific questions about the orders.

Question: Using Pandas DataFrame operations, how would you find the following information from the `order_data` DataFrame:

1. The total number of orders made by each customer.
2. The average order quantity for each product.
3. The earliest and latest order dates in the dataset.

```
import pandas as pd
data = {
    'customer ID': [101, 102, 101, 103, 104, 102, 101],
    'order date': ['2023-07-01', '2023-07-02', '2023-07-03', '2023-07-03', '2023-
07-04', '2023-07-05', '2023-07-06'],
    'product name': ['A', 'B', 'A', 'C', 'B', 'A', 'C'],
    'order quantity': [3, 5, 2, 1, 2, 4, 3]
}
order = pd.DataFrame(data)
order['order date'] = pd.to_datetime(order['order date'])
total = order['customer ID'].value_counts()
```

```
ave = order.groupby('product name')['order quantity'].mean()
early = order['order date'].min()
latest = order['order date'].max()

print("Total number of orders made by each customer:")
print(total)
print("\nAverage order quantity for each product:")
print(ave)
print("\nEarliest order date:", early)
print("Latest order date:", latest)


'''
sample output
Total number of orders made by each customer:
customer ID
101    3
102    2
103    1
104    1
Name: count, dtype: int64

Average order quantity for each product:
product name
A    3.0
B    3.5
C    2.0
Name: order quantity, dtype: float64

Earliest order date: 2023-07-01 00:00:00
Latest order date: 2023-07-06 00:00:00
'''
```

8. **Scenario:** You are a data scientist working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a Pandas data frame.

Question: How would you find the top 5 products that have been sold the most in the past month?

```
import pandas as pd

data = {
    'product_name': ['Product A', 'Product B', 'Product A', 'Product C', 'Product
B', 'Product A', 'Product C'],
    'quantity': [100, 80, 70, 50, 120, 90, 60]
}
data1 = pd.DataFrame(data)
sales = data1.groupby('product_name')['quantity'].sum()
```

```
sorted = sales.sort_values(ascending=False)
top5 = sorted.head(5)
print("Top 5 products in the past month:")
print(top5)


'''
sample output
Top 5 products in the past month:
product_name
Product A    260
Product B    200
Product C    110
Name: quantity, dtype: int64
'''
```

9. **Scenario:** You work for a real estate agency and have been given a dataset containing information about properties for sale. The dataset is stored in a Pandas DataFrame named `property_data`. The DataFrame has columns for property ID, location, number of bedrooms, area in square feet, and listing price. Your task is to analyze the data and answer specific questions about the properties.

Question: Using Pandas DataFrame operations, how would you find the following information from the `property_data` DataFrame:

1. The average listing price of properties in each location.
2. The number of properties with more than four bedrooms.
3. The property with the largest area.

```
import pandas as pd
data = {
    'property ID': [1, 2, 3, 4, 5, 6],
    'location': ['City A', 'City B', 'City A', 'City C', 'City B', 'City C'],
    'number of bedrooms': [3, 4, 5, 5, 2, 4],
    'area in square feet': [1500, 1800, 1600, 2200, 1200, 2000],
    'listing price': [250000, 320000, 280000, 420000, 180000, 380000]
}
property = pd.DataFrame(data)
average = property.groupby('location')['listing price'].mean()
four_bedrooms = property[property['number of bedrooms'] > 4]
total4bed = len(four_bedrooms)
largest_area = property[property['area in square feet'] == property['area in
   square feet'].max()]
print("Average listing price")
print(average)
print("\nproperties with more than four bedrooms:", total4bed)
print("\nProperty with the largest area:")
print(largest_area)

```

```
'''
sample output
Average listing price
location
City A    265000.0
City B    250000.0
City C    400000.0
Name: listing price, dtype: float64

properties with more than four bedrooms: 2

Property with the largest area:
   property ID location  number of bedrooms  area in square feet  listing price
3            4   City C                   5                 2200         420000
'''
```

10. **Scenario:** You are working on a data visualization project and need to create basic plots using Matplotlib. You have a dataset containing the monthly sales data for a company, including the month and corresponding sales values. Your task is to develop a Python program that generates line plots and bar plots to visualize the sales data.

Question:

1. How would you develop a Python program to create a line plot of the monthly sales data?
2: How would you develop a Python program to create a bar plot of the monthly sales data?

```python
import matplotlib.pyplot as plt
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
sales = [1000, 1200, 1500, 900, 1800, 2000]

plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
plt.plot(months, sales, marker='o', linestyle='-', color='b')
plt.xlabel('Months')
plt.ylabel('Sales')
plt.title('Monthly Sales Data - Line Plot')
plt.grid(True)

plt.subplot(1, 2, 2)
plt.bar(months, sales, color='b')
plt.xlabel('Months')
plt.ylabel('Sales')
plt.title('Monthly Sales Data - Bar Plot')
plt.grid(True)
```

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


'''
sample output

'''
```

11. **Scenario:** You are a data scientist working for a company that sells products online. You have been tasked with creating a simple plot to show the sales of a product over time.

Question:

1. Write code to create a simple line plot in Python using Matplotlib to predict sales happened in a month?
2. Write code to create a scatter plot in Python using Matplotlib to predict sales happened in a month?
3. Develop a Python program to create a bar plot of the monthly sales data.

```
import matplotlib.pyplot as plt
months = [1, 2, 3, 4, 5, 6]
sales = [1000, 1200, 900, 1500, 1800, 1300]
plt.plot(months, sales, marker='o', linestyle='-')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales of Product X Over Time')
plt.grid(True)
plt.show()

plt.scatter(months, sales)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Sales of Product X Over Time')
plt.grid(True)
plt.show()

plt.bar(months, sales)
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Monthly Sales Data')
plt.grid(True)
plt.show()
```

12. **Scenario:** You are working on a data analysis project that involves analyzing the monthly temperature and rainfall data for a city. You have a dataset containing the monthly temperature and

rainfall values for each month of a year. Your task is to develop a Python program that generates line plots and scatter plots to visualize the temperature and rainfall data.

Question:

1. Develop a Python program to create a line plot of the monthly temperature data.
2: Develop a Python program to create a scatter plot of the monthly rainfall data.

```
import matplotlib.pyplot as plt
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct',
'Nov', 'Dec']
temperature = [10, 12, 15, 20, 25, 28, 30, 29, 26, 20, 15, 12]
rainfall = [50, 40, 60, 80, 100, 120, 130, 120, 100, 80, 60, 40]

plt.figure(figsize=(10, 5))
plt.plot(months, temperature, marker='o')
plt.title("Monthly Temperature Data")
plt.xlabel("Month")
plt.ylabel("Temperature (°C)")
plt.grid(True)
plt.show()

plt.figure(figsize=(10, 5))
plt.scatter(months, rainfall, color='blue', marker='o')
plt.title("Monthly Rainfall Data")
plt.xlabel("Month")
plt.ylabel("Rainfall (mm)")
plt.grid(True)
plt.show()
```

13. **Scenario:** You are working on a text analysis project and need to determine the frequency distribution of words in a given text document. You have a text document named "sample_text.txt" containing a paragraph of text. Your task is to develop a Python program that reads the text document, processes the text, and generates a frequency distribution of the words.

Question: How would you develop a Python program to calculate the frequency distribution of words in a text document?

```
from collections import Counter

text = """
This is a sample text. It contains some words , and some words are repeated.
The goal is to calculate the frequency distribution of words in this text.
"""

text = text.lower()

words = text.split()
```

```
freq = Counter(words)

print("Word Frequency Distribution:")
for word, frequency in freq.items():
    print(f"{word}: {frequency}")


'''
sample output
Word Frequency Distribution:
this: 2 is: 2
a: 1    sample: 1
text.: 2    it: 1
contains: 1 some: 2
words: 3    ,: 1
and: 1  are: 1
repeated.: 1    the: 2
goal: 1 to: 1
calculate: 1    frequency: 1
distribution: 1 of: 1
in: 1
'''
```

14. **Scenario**: You are a data analyst working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a Pandas data frame.

Question: Develop a code in python to find the frequency distribution of the ages of the customers who have made a purchase in the past month.

```
import pandas as pd

data = {'customer_id': [1, 2, 3, 4, 5, 6, 7],
        'age': [25, 30, 22, 40, 30, 30, 25]}

df = pd.DataFrame(data)
freq = df['age'].value_counts().sort_index()

print("Age Frequency Distribution:")
print(freq)


'''
sample output
Age Frequency Distribution:
age
22    1
25    2
```

```
30    3
40    1
Name: count, dtype: int64
'''
```

15. **Scenario:** You are a data analyst working for a social media platform. As part of your analysis, you have a dataset containing user interaction data, including the number of likes received by each post. Your task is to develop a Python program that calculates the frequency distribution of likes among the posts.

Question: Develop a Python program to calculate the frequency distribution of likes among the posts?

```
import pandas as pd

data = {'post_id': [1, 2, 3, 4, 5, 6, ],
        'likes': [100, 150, 50, 50, 120,150]}

df = pd.DataFrame(data)
freq= df['likes'].value_counts().sort_index()
print("Likes Frequency Distribution:")
print(freq)


'''
sample output
Likes Frequency Distribution:
likes
50     2
100    1
120    1
150    2
Name: count, dtype: int64
'''
```

16. **Scenario:** You are working on a project that involves analyzing customer reviews for a product. You have a dataset containing customer reviews, and your task is to develop a Python program that calculates the frequency distribution of words in the reviews.

Question: Develop a Python program to calculate the frequency distribution of words in the customer reviews dataset?

```
from collections import Counter

reviews = [
    "The product is amazing! I love it.",
    "Not satisfied with the quality.",
```

```
    "This is the best purchase I've made.",
    "The customer service was terrible.",
    "I would recommend this product to others."
]

all_text = ' '.join(reviews)

words = all_text.lower().split()

word = Counter(words)

print("Word Frequency Distribution:")
for word, frequency in word.items():
    print(f"{word}: {frequency}")



'''
sample output
Word Frequency Distribution:
the: 4  product: 2
is: 2 amazing!: 1
i: 2    love: 1
it.: 1  not: 1
satisfied: 1 with: 1
quality.: 1 this: 2
best: 1 purchase: 1
i've: 1 made.: 1
customer: 1 service: 1
was: 1  terrible.: 1
would: 1    recommend: 1
to: 1   others.: 1
'''
```

17. **Scenario:** You are a data analyst working for a marketing research company. Your team has collected a large dataset containing customer feedback from various social media platforms. The dataset consists of thousands of text entries, and your task is to develop a Python program to analyze the frequency distribution of words in this dataset. Your program should be able to perform the following tasks:

- Load the dataset from a CSV file (data.csv) containing a single column named "feedback" with each row representing a customer comment.
- Preprocess the text data by removing punctuation, converting all text to lowercase, and eliminating any stop words (common words like "the," "and," "is," etc. that don't carry significant meaning).
- Calculate the frequency distribution of words in the preprocessed dataset.
- Display the top N most frequent words and their corresponding frequencies, where N is provided as user input.

- Plot a bar graph to visualize the top N most frequent words and their frequencies.

Question: Create a Python program that fulfills these requirements and helps your team gain insights from the customer feedback data.

```
import pandas as pd
import string
from collections import Counter
import matplotlib.pyplot as plt

# Define a list of common English stopwords
stop_words = set([
    "i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your",
"yours", "yourself", "yourselves",
    "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its",
"itself", "they", "them", "their",
    "theirs", "themselves", "what", "which", "who", "whom", "this", "that",
"these", "those", "am", "is", "are", "was",
    "were", "be", "been", "being", "have", "has", "had", "having", "do", "does",
"did", "doing", "a", "an", "the", "and",
    "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for",
"with", "about", "against", "between",
    "into", "through", "during", "before", "after", "above", "below", "to", "from",
"up", "down", "in", "out", "on", "off",
    "over", "under", "again", "further", "then", "once", "here", "there", "when",
"where", "why", "how", "all", "any",
    "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor",
"not", "only", "own", "same", "so",
    "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now"
])

df = pd.read_csv("C:/Users/vishwa2003/Downloads/data1.csv")

def preprocess_text(text):
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return words

all_words = df['feedback'].apply(preprocess_text).explode()

word_frequency = all_words.value_counts()

top_n = int(input("Enter the value of N for top N most frequent words: "))

print(f"Top {top_n} Most Frequent Words:")
print(word_frequency.head(top_n))
```

```
word_frequency.head(top_n).plot(kind='bar', figsize=(10, 6))
plt.title(f"Top {top_n} Most Frequent Words")
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

18. Suppose a hospital tested the age and body fat data for 18 randomly selected adults with the following result.

age	23	23	27	27	39	41	47	49	50
%fat	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
age	52	54	54	56	57	58	58	60	61
%fat	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7

Question:

- Calculate the mean, median and standard deviation of age and %fat using Pandas.
- Draw the boxplots for age and %fat.
- Draw a scatter plot and a q-q plot based on these two variables

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

# Sample data
data = {'age': [32, 45, 23, 52, 26, 48, 34, 29, 41, 36, 28, 50, 31, 40, 33,
47, 25, 38],
        'fat': [22.4, 34.2, 18.5, 30.3, 25.9, 31.4, 27.2, 26.7, 29.9, 30.5,
28.5, 32.4, 25.1, 29.7, 23.7, 29.5, 25.8, 27.5]}

df = pd.DataFrame(data)

age_mean, age_median, age_std = df['age'].mean(), df['age'].median(),
df['age'].std()
fat_mean, fat_median, fat_std = df['fat'].mean(), df['fat'].median(),
df['fat'].std()

print("Age:")
print(f"Mean: {age_mean:.2f}, Median: {age_median:.2f}, Std Dev:
{age_std:.2f}")
print("\nfat:")
```

```
print(f"Mean: {fat_mean:.2f}, Median: {fat_median:.2f}, Std Dev:
{fat_std:.2f}")

plt.figure(figsize=(10, 6))
sns.boxplot(data=df)
plt.title('Boxplots for Age and fat')
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='age', y='fat')
plt.title('Scatter Plot: Age vs. fat')
plt.show()

plt.figure(figsize=(10, 6))
stats.probplot(df['fat'], dist='norm', plot=plt)
plt.title('Q-Q Plot: fat')
plt.show()

'''
sample output

'''
```

19. **Scenario:**

You are a medical researcher investigating the effectiveness of a new drug in reducing blood pressure. You conduct a clinical trial with a sample of 50 patients who were randomly assigned to receive either the new drug or a placebo. After measuring their blood pressure levels at the end of the trial, you obtain the data for both groups. Now, you want to determine the confidence intervals for the mean reduction in blood pressure for both the drug and placebo groups.

Question:

What is the 95% confidence interval for the mean reduction in blood pressure for patients who received the new drug? Also, what is the 95% confidence interval for the mean reduction in blood pressure for patients who received the placebo?

```
import numpy as np
import scipy.stats as stats
dgroup = np.array([5.1, 4.8, 6.2, 5.7, 4.9, 5.5, 6.8, 5.3, 5.9, 6.5, 4.4, 5.1, 6.0,
5.7, 6.3, 5.6, 5.8, 4.7, 6.2, 5.9,
                   5.2, 6.1, 5.5, 5.0, 6.4, 5.6, 6.2, 5.4, 6.7, 4.9, 5.6, 6.3,
5.8, 6.0, 4.8, 5.9, 5.3, 6.1, 5.4, 4.7,
                   6.5, 5.2, 6.7, 5.8, 5.5, 5.0, 6.3, 5.6])
pgroup = np.array([2.3, 2.6, 2.8, 2.4, 2.1, 2.7, 2.5, 2.2, 2.9, 2.4, 2.6, 2.3, 2.7,
2.5, 2.8, 2.6, 2.4, 2.2, 2.1, 2.7,
                    2.3, 2.4, 2.5, 2.6, 2.2, 2.8, 2.4, 2.5, 2.3, 2.6, 2.7,
2.5, 2.4, 2.9, 2.2, 2.3, 2.7, 2.5, 2.8,
                    2.1, 2.6, 2.4, 2.5, 2.3, 2.7, 2.2, 2.9])
```

```
con_lvl = 0.95
drug_mean = np.mean(dgroup)
drug_std = np.std(dgroup, ddof=1)
drug_n = len(dgroup)
drug_mof = stats.t.ppf((1 + con_lvl) / 2, df=drug_n - 1) * (drug_std /
np.sqrt(drug_n))
drug_ci = (drug_mean - drug_mof, drug_mean + drug_mof)

pmean = np.mean(pgroup)
pstd = np.std(pgroup, ddof=1)
placebo_n = len(pgroup)
p_mof = stats.t.ppf((1 + con_lvl) / 2, df=placebo_n - 1) * (pstd /
np.sqrt(placebo_n))
placebo_ci = (pmean - p_mof, pmean + p_mof)

print(f"95% Confidence Interval for Drug Group: ({drug_ci[0]:.2f},
{drug_ci[1]:.2f})")
print(f"95% Confidence Interval for Placebo Group: ({placebo_ci[0]:.2f},
{placebo_ci[1]:.2f})")


'''
sample output
95% Confidence Interval for Drug Group: (5.49, 5.84)
95% Confidence Interval for Placebo Group: (2.42, 2.56)
'''
```

20. **Scenario:**

You are a data scientist working for an e-commerce company. The marketing team has conducted an A/B test to evaluate the effectiveness of two different website designs (A and B) in terms of conversion rate. They randomly divided the website visitors into two groups, with one group experiencing design A and the other experiencing design B. After a week of data collection, you now have the conversion rate data for both groups. You want to determine whether there is a statistically significant difference in the mean conversion rates between the two website designs.

Question:

"Based on the data collected from the A/B test, is there a statistically significant difference in the mean conversion rates between website design A and website design B?"

```
import scipy.stats as stats

conversion_rate_a = [0.12, 0.10, 0.14, 0.11, 0.09, 0.13, 0.08, 0.10, 0.12, 0.11]
conversion_rate_b = [0.15, 0.16, 0.14, 0.17, 0.13, 0.18, 0.14, 0.16, 0.15, 0.14]

t_statistic, p_value = stats.ttest_ind(conversion_rate_a, conversion_rate_b)

alpha = 0.05
```

```
if p_value < alpha:
    result = "statistically significant"
else:
    result = "not statistically significant"

print(f"The t-statistic is: {t_statistic:.4f}")
print(f"The p-value is: {p_value:.4f}")
print(f"Conclusion: There is a {result} difference in mean conversion rates between designs A and B.")


'''
sample output
The t-statistic is: -5.5468
The p-value is: 0.0000
Conclusion: There is a statistically significant difference in mean conversion rates between designs A and B.
'''
```

21.**Scenario:**

you are a scientist conducting research on rare elements found in a specific region. Your goal is to estimate the average concentration of a rare element in the region using a random sample of measurements. You will use the NumPy library to perform point estimation and calculate confidence intervals for the population mean.The rare element concentration data is stored in a CSV file named "rare_elements.csv," where each row contains a single measurement of the concentration.

Question:

write a Python program that allows the user to input the sample size, confidence level, and desired level of precision.

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

# Sample data
data = {'concentration': [4.6, 3.8, 4.2, 4.0, 4.3, 3.9, 4.1, 4.4, 3.7, 4.5]}

df = pd.DataFrame(data)

size = 5
c_lvl = 0.95
desired_precision = 0.1

mean = df['concentration'][:size].mean()
std = df['concentration'][:size].std(ddof=1)

error = std / np.sqrt(size)
```

```
t_score = stats.t.ppf(1 - (1 - c_lvl) / 2, df=size - 1)

mof = t_score * error

lower_bound = mean - mof
upper_bound = mean + mof

requiredsize = ((t_score * std) / desired_precision) ** 2

print("\nPoint Estimation:")
print(f"Sample Mean: {mean:.4f}")
print(f"Sample Standard Deviation: {std:.4f}")

print("\nConfidence Interval:")
print(f"Lower Bound: {lower_bound:.4f}")
print(f"Upper Bound: {upper_bound:.4f}")
print(f"Confidence Level: {c_lvl * 100:.2f}%")

print("\nRequired Sample Size for Desired Precision:")
print(f"Required Sample Size: {int(np.ceil(requiredsize))}")


'''
sample output
Point Estimation:
Sample Mean: 4.1800
Sample Standard Deviation: 0.3033

Confidence Interval:
Lower Bound: 3.8034
Upper Bound: 4.5566
Confidence Level: 95.00%

Required Sample Size for Desired Precision:
Required Sample Size: 71
'''
```

22.**Scenario:**

Imagine you are an analyst for a popular online shopping website. Your task is to analyze customer reviews and provide insights on the average rating and customer satisfaction level for a specific product category.

Question:

You will use the pandas library to calculate confidence intervals to estimate the true population mean rating.

You have been provided with a CSV file named "customer_reviews.csv," which contains customer ratings for products in the chosen category.

```
import pandas as pd
import scipy.stats as stats

data = {'rating': [4, 5, 3, 4, 5, 4, 3, 5, 4, 4, 5, 3, 4, 5, 4, 4, 3, 4, 5, 3]}
df = pd.DataFrame(data)
ratings = df['rating']
mean = ratings.mean()
std = ratings.std()
size = len(ratings)
con_lvl = 0.95

# Calculate the t-score for the desired confidence level
t_score = stats.t.ppf((1 + con_lvl) / 2, df=size - 1)

# Calculate standard error of the mean (SEM)
sem = std / (size ** 0.5)

# Calculate the margin of error
mof = t_score * sem

# Calculate the confidence interval
con_interval = (mean - mof, mean + mof)
print(f"Sample Mean Rating: {mean:.2f}")
print(f"Confidence Interval ({con_lvl*100:.0f}%): "
      f"({con_interval[0]:.2f}, {con_interval[1]:.2f})")

'''sample output
Sample Mean Rating: 4.05
Confidence Interval (95%): (3.69, 4.41)
'''
```

23.**Scenario:**

You are a researcher working in a medical lab, investigating the effectiveness of a new treatment for a specific disease. You have collected data from a clinical trial with two groups: a control group receiving a placebo, and a treatment group receiving the new drug.Your goal is to analyze the data using hypothesis testing and calculate the p-value to determine if the new treatment has a statistically significant effect compared to the placebo. You will use the matplotlib library to visualize the data and the p-value.

```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
```

```
placebo_group = np.array([68, 72, 65, 70, 71, 67, 73, 69, 70, 66])
treatment_group = np.array([60, 64, 55, 62, 58, 61, 59, 57, 63, 56])
# Perform t-test (two-sample independent t-test)
t_statistic, p_value = stats.ttest_ind(placebo_group, treatment_group)

print(f"t-statistic: {t_statistic:.2f}")
print(f"p-value: {p_value:.10f}")

# Determine if the p-value is statistically significant (common threshold is 0.05)
if p_value < 0.05:
    print("has a statistically significant effect.")
else:
    print("has no statistically significant effect.")

plt.figure(figsize=(8, 6))
plt.boxplot([placebo_group, treatment_group], labels=['Placebo', 'Treatment'])
plt.title('Boxplot of Placebo vs. Treatment')
plt.ylabel('Response')
plt.show()

'''sample output
t-statistic: 7.61
p-value: 0.0000004999
has a statistically significant effect.
'''
```

24.**Question:** K-Nearest Neighbors (KNN) Classifier

You are working on a classification problem to predict whether a patient has a certain medical condition or not based on their symptoms. You have collected a dataset of patients with labeled data (0 for no condition, 1 for the condition) and various symptom features.

Write a Python program that allows the user to input the features of a new patient and the value of k (number of neighbors). The program should use the KNN classifier from the scikit-learn library to predict whether the patient has the medical condition or not based on the input features.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

# Larger sample dataset (features and labels)
X = np.random.rand(100, 5)  # 100 samples with 5 features each
y = np.random.choice([0, 1], size=100)  # Random labels (0 or 1)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Create a KNN classifier
k = int(input("Enter the value of k (number of neighbors): "))
knn = KNeighborsClassifier(n_neighbors=k)

# Fit the classifier on the training data
knn.fit(X_train, y_train)

# Input features for the new patient
features = np.random.rand(1, 5)  # Generate random features for a new patient

# Predict whether the patient has the medical condition or not
predict = knn.predict(features)

if predict[0] == 0:
    print("The patient does not have the medical condition.")
else:
    print("The patient has the medical condition.")

'''sample output
Enter the value of k (number of neighbors): 30
The patient does not have the medical condition.
'''
```

25.**Question 2:** Decision Tree for Iris Flower Classification

You are analyzing the famous Iris flower dataset to classify iris flowers into three species based on their sepal and petal dimensions. You want to use a Decision Tree classifier to accomplish this task.

Write a Python program that loads the Iris dataset from scikit-learn, and allows the user to input the sepal length, sepal width, petal length, and petal width of a new flower. The program should then use the Decision Tree classifier to predict the species of the new flower.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Load the Iris dataset
iris = load_iris()
x = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)
```

```
dt= DecisionTreeClassifier()

# Fit the classifier on the training data
dt.fit(X_train, y_train)

# Input features for the new flower
sepal_length = float(input("Enter sepal length: "))
sepal_width = float(input("Enter sepal width: "))
petal_length = float(input("Enter petal length: "))
petal_width = float(input("Enter petal width: "))

features = np.array([[sepal_length, sepal_width,
                                  petal_length, petal_width]])

# Predict the species of the new flower
predicted_species = dt.predict(features)

species_names = ['Setosa', 'Versicolor', 'Virginica']
predicted_species_name = species_names[predicted_species[0]]

print(f"The predicted species of the new flower is: {predicted_species_name}")


'''
sample output
Enter sepal length: 2
Enter sepal width: 2
Enter petal length: 2
Enter petal width: 2
The predicted species of the new flower is: Setosa
'''
```

26.**Question** : Linear Regression for Housing Price Prediction

You are a real estate analyst trying to predict housing prices based on various features of the houses, such as area, number of bedrooms, and location. You have collected a dataset of houses with their respective prices.

Write a Python program that allows the user to input the features (area, number of bedrooms, etc.) of a new house. The program should use linear regression from scikit-learn to predict the price of the new house based on the input features.

```
import numpy as np
from sklearn.linear_model import LinearRegression

# Sample dataset (features: area, bedrooms; target: price)
X = np.array([[1400, 3], [1600, 4], [1800, 3], [2000, 4], [2200, 5]])
```

```
y = np.array([200000, 250000, 280000, 300000, 330000])

# Create a Linear Regression model
lin_reg = LinearRegression()

# Fit the model on the dataset
lin_reg.fit(X, y)

# Input features for the new house
house_area = float(input("Enter area: "))
house_bedrooms = int(input("Enter no of bedrooms: "))

house_features = np.array([[house_area, house_bedrooms]])

# Predict the price of the new house
predict_price = lin_reg.predict(house_features)

print(f"The predicted price of the new house is: {predict_price[0]:,.2f}")


'''
sample output
Enter area: 200
Enter no of bedrooms: 1
The predicted price of the new house is: 23,333.33
'''
```

27.**Question:** Logistic Regression for Customer Churn Prediction

You are working for a telecommunications company, and you want to predict whether a customer will churn (leave the company) based on their usage patterns and demographic data. You have collected a dataset of past customers with their churn status (0 for not churned, 1 for churned) and various features.

Write a Python program that allows the user to input the features (e.g., usage minutes, contract duration) of a new customer. The program should use logistic regression from scikit-learn to predict whether the new customer will churn or not based on the input features.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Sample dataset (features and labels)
x = np.array([[100, 24], [200, 12], [50, 6], [300, 36], [150, 18], [80, 9]])
y = np.array([0, 1, 0, 1, 0, 1])  # 0: Not Churned, 1: Churned

# Split the dataset into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)

# Create a Logistic Regression classifier
logreg = LogisticRegression()

# Fit the classifier on the training data
logreg.fit(X_train, y_train)

# Input features for the new customer
min = int(input("Enter usage minutes: "))
duration = int(input("Enter contract duration: "))

features = np.array([[min, duration]])

# Predict whether the new customer will churn or not
predict = logreg.predict(features)

if predict[0] == 0:
    print("The new customer is not likely to churn.")
else:
    print("The new customer is likely to churn.")


'''
sample output
Enter usage minutes: 180
Enter contract duration: 20
The new customer is likely to churn.
'''
```

28.**Question:** K-Means Clustering for Customer Segmentation

You are working for an e-commerce company and want to segment your customers into distinct groups based on their purchasing behavior. You have collected a dataset of customer data with various shopping-related features.

Write a Python program that allows the user to input the shopping-related features of a new customer. The program should use K-Means clustering from scikit-learn to assign the new customer to one of the existing segments based on the input features.

```
import numpy as np
from sklearn.cluster import KMeans

# Sample customer data (features)
customer_data = np.array([[5.1, 3.5, 1.4, 0.2],   # Feature 1: Sepal length, Sepal
width, Petal length, Petal width
```

```
                          [4.9, 3.0, 1.4, 0.2],
                          [5.8, 2.6, 4.0, 1.2],
                          [6.6, 3.0, 4.4, 1.4],
                          [7.3, 2.9, 6.3, 1.8]])

# Create a K-Means clustering model with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)

# Fit the model on the customer data
kmeans.fit(customer_data)

# Input features for the new customer
new_customer_features = np.array([[6.2, 3.1, 5.2, 2.3]])

# Predict the segment for the new customer
predicted_segment = kmeans.predict(new_customer_features)

print(f"The predicted segment for the new customer is: {predicted_segment[0]}")


'''
sample output
The predicted segment for the new customer is: 2
'''
```

29.**Question:** Evaluation Metrics for Model Performance

You have trained a machine learning model on a dataset, and now you want to evaluate its performance using various metrics.

Write a Python program that loads a dataset and trained model from scikit-learn. The program should ask the user to input the names of the features and the target variable they want to use for evaluation. The program should then calculate and display common evaluation metrics such as accuracy, precision, recall, and F1-score for the model's predictions on the test data.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Example feature values
X = [
    [5.1, 3.5, 1.4, 0.2],
    [4.9, 3.0, 1.4, 0.2],
    [5.8, 2.6, 4.0, 1.2],
    [6.6, 3.0, 4.4, 1.4],
    [7.3, 2.9, 6.3, 1.8]]
# Example target labels
```

```
y = [0, 0, 1, 1, 2]

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

predict = model.predict(X_test)

print(f"Evaluation metrics for the model:")
print(f"Accuracy: {accuracy_score(y_test, predict):.4f}")
print(f"Precision: {precision_score(y_test, predict, average='weighted'):.4f}")
print(f"Recall: {recall_score(y_test, predict, average='weighted'):.4f}")
print(f"F1-Score: {f1_score(y_test, predict, average='weighted'):.4f}")


'''
sample output
Evaluation metrics for the model:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1-Score: 1.0000
'''
```

30.**Question**: Classification and Regression Trees (CART) for Car Price Prediction

You are working for a car dealership, and you want to predict the price of used cars based on various features such as the car's mileage, age, brand, and engine type. You have collected a dataset of used cars with their respective prices.

Write a Python program that loads the car dataset and allows the user to input the features of a new car they want to sell. The program should use the Classification and Regression Trees (CART) algorithm from scikit-learn to predict the price of the new car based on the input features.

The CART algorithm will create a tree-based model that will split the data into subsets based on the chosen features and their values, leading to a decision path that eventually predicts the price of the car. The program should output the predicted price and display the decision path (the sequence of conditions leading to the prediction) for the new car.

```
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import export_text
X = np.array([
    [50000, 3, 0],  # Mileage: 50000, Age: 3, Brand: 0 (Toyota)
    [80000, 5, 1],  # Mileage: 80000, Age: 5, Brand: 1 (Honda)
    [30000, 2, 0],  # Mileage: 30000, Age: 2, Brand: 0 (Toyota)
```

```
    [60000, 4, 2]   # Mileage: 60000, Age: 4, Brand: 2 (Ford)
])
# target prices
y = np.array([15000, 12000, 18000, 10000])

model = DecisionTreeRegressor()
model.fit(X, y)
# Input feature values for the new car
features = np.array([33000, 2, 0])  # Mileage: 33000, Age: 2, Brand: 0 (Honda)
predicted_price = model.predict([features])[0]
print(f"Predicted price for the new car: ${predicted_price:.2f}")

decision_path = model.decision_path([features])
print("\nDecision path:")
print(export_text(model, feature_names=['Mileage', 'Age', 'Brand']))


'''
sample output
Predicted price for the new car: $18000.00

Decision path:
|--- Brand <= 0.50
|   |--- Age <= 2.50
|   |   |--- value: [18000.00]
|   |--- Age >  2.50
|   |   |--- value: [15000.00]
|--- Brand >  0.50
|   |--- Mileage <= 70000.00
|   |   |--- value: [10000.00]
|   |--- Mileage >  70000.00
|   |   |--- value: [12000.00]
'''
```

31**. Scenario**: You work as a data scientist for an e-commerce company that sells a wide range of products online. The company collects vast amounts of data about its customers, including their purchase history, browsing behavior, demographics, and more. The marketing team wants to understand their customer base better and improve their targeted marketing strategies. They have asked you to perform customer segmentation using clustering techniques to identify distinct groups of customers with similar characteristics.

Question: Your task is to use Python and clustering algorithms to segment the customers into different groups based on their behavior and characteristics. The marketing team will use these segments to tailor their marketing campaigns and promotions effectively.

```
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
# Generate synthetic customer data
np.random.seed(0)
num_samples = 300
num_features = 2

data = np.random.rand(num_samples, num_features) * 10

num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(data)
cluster_labels = kmeans.labels_

plt.figure(figsize=(8, 6))
for cluster_id in range(num_clusters):
    plt.scatter(
        data[cluster_labels == cluster_id, 0],
        data[cluster_labels == cluster_id, 1],
        label=f'Cluster {cluster_id + 1}'
    )

plt.scatter(
    kmeans.cluster_centers_[:, 0],
    kmeans.cluster_centers_[:, 1],
    s=200,
    marker='X',
    c='black',
    label='Centroids')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Customer Segmentation using K-Means Clustering')
plt.legend()
plt.show()
```

32. **Scenario:** You work as a data scientist for a real estate company. The company has collected data on various houses, including features such as the size of the house, number of bedrooms, location, and other relevant attributes. The marketing team wants to build a predictive model to estimate the price of houses based on their features. They believe that linear regression modeling can be an effective approach for this task.

Question: Your task is write a Python program to perform bivariate analysis and build a linear regression model to predict house prices based on a selected feature (e.g., house size) from the dataset. Additionally, you need to evaluate the model's performance to ensure its accuracy and reliability.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
import matplotlib.pyplot as plt

# Generate synthetic housing data
np.random.seed(0)
num_samples = 100
house_size = np.random.rand(num_samples) * 200 + 800
house_price = 300000 + 1500 * house_size + np.random.randn(num_samples) * 50000

# Create a DataFrame
data = pd.DataFrame({'Size': house_size, 'Price': house_price})

# Split the data into training and testing sets
X = data[['Size']]
y = data['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict house prices using the model
y_pred = model.predict(X_test)

# Display model accuracy and reliability
accuracy = model.score(X_test, y_test)
reliability = model.score(X_train, y_train)

print(f"Model Accuracy: {accuracy:.2f}")
print(f"Model Reliability: {reliability:.2f}")

'''
# Visualize the regression line
plt.scatter(X_test, y_test, label='Test Data')
plt.plot(X_test, y_pred, color='red', label='Regression Line')
plt.xlabel('House Size')
plt.ylabel('House Price')
plt.title('Linear Regression Model: House Size vs. Price')
plt.legend()
plt.show()
'''

'''
sample output
Model Accuracy: 0.65
Model Reliability: 0.76
'''
```

33. **Scenario:** You work as a data scientist for an automobile company that sells various car models. The company has collected data on different car attributes, such as engine size, horsepower, fuel efficiency, and more, along with their corresponding prices. The marketing team wants to build a predictive model to estimate the price of cars based on their features.

Question: Your task is write a Python program that perform linear regression modeling to predict car prices based on a selected set of features from the dataset. Additionally, you need to evaluate the model's performance and provide insights to the marketing team to understand the most influential factors affecting car prices.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Generate synthetic car data
np.random.seed(0)
num_samples = 100
engine_size = np.random.rand(num_samples) * 2 + 1
horsepower = np.random.randint(100, 400, size=num_samples)
car_price = 20000 + 10000 * engine_size + 50 * horsepower +
np.random.randn(num_samples) * 5000

# Create a DataFrame
data = pd.DataFrame({
    'EngineSize': engine_size,
    'Horsepower': horsepower,
    'Price': car_price
})
# Select features and target variable
features = ['EngineSize', 'Horsepower']
target = 'Price'
X = data[features]
y = data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict car prices using the model
y_pred = model.predict(X_test)

# Calculate model accuracy
accuracy = model.score(X_test, y_test)
print(f"Model Accuracy: {accuracy:.2f}")
```

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs. Predicted Car Prices')
plt.show()


'''
sample output
Model Accuracy: 0.44
'''
```

34. **Scenario:** Suppose you are working as a data scientist for a medical research organization. Your team has collected data on patients with a certain medical condition and their treatment outcomes. The dataset includes various features such as age, gender, blood pressure, cholesterol levels, and whether the patient responded positively ("Good") or negatively ("Bad") to the treatment. The organization wants to use this model to identify potential candidates who are likely to respond positively to the treatment and improve their medical approach.

Question: Your task is to build a classification model using the KNN algorithm to predict the treatment outcome ("Good" or "Bad") for new patients based on their features. Evaluate the model's performance using accuracy, precision, recall, and F1-score.Make predictions on the test set and display the results.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report

# Generate synthetic medical data
np.random.seed(0)
num_samples = 200
age = np.random.randint(20, 80, size=num_samples)
gender = np.random.choice(['Male', 'Female'], size=num_samples)
blood_pressure = np.random.randint(80, 160, size=num_samples)
cholesterol = np.random.randint(120, 300, size=num_samples)
treatment_outcome = np.random.choice(['Good', 'Bad'], size=num_samples)

data = pd.DataFrame({
    'Age': age,
    'Gender': gender,
    'BloodPressure': blood_pressure,
    'Cholesterol': cholesterol,
    'Outcome': treatment_outcome
})
# Convert categorical features to numerical using one-hot encoding
data = pd.get_dummies(data, columns=['Gender'], drop_first=True)
```

```
features = ['Age', 'BloodPressure', 'Cholesterol', 'Gender_Male']
target = 'Outcome'

X = data[features]
y = data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train a KNN classifier
k = 5  # Number of neighbors
model = KNeighborsClassifier(n_neighbors=k)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print(f"Precision: {precision_score(y_test, y_pred, pos_label='Good'):.2f}")
print(f"Recall: {recall_score(y_test, y_pred, pos_label='Good'):.2f}")
print(f"F1-Score: {f1_score(y_test, y_pred, pos_label='Good'):.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))

'''
sample output
Accuracy: 0.55
Precision: 0.56
Recall: 0.71
F1-Score: 0.63
Classification Report:
              precision    recall  f1-score   support

         Bad       0.54      0.37      0.44        19
        Good       0.56      0.71      0.63        21

    accuracy                           0.55        40
   macro avg       0.55      0.54      0.53        40
weighted avg       0.55      0.55      0.54        40
'''
```

35. **Scenario:** You work as a data scientist for a retail company that operates multiple stores. The company is interested in segmenting its customers based on their purchasing behavior to better understand their preferences and tailor marketing strategies accordingly. To achieve this, your team has collected transaction data from different stores, which includes customer IDs, the total amount spent in each transaction, and the frequency of visits.

Question: Your task is to build a clustering model using the K-Means algorithm to group customers into distinct segments based on their spending patterns.

```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
np.random.seed(0)
num_customers = 200
total_amount_spent = np.random.randint(50, 1000, size=num_customers)
visit_frequency = np.random.randint(1, 30, size=num_customers)
data = pd.DataFrame({
    'TotalAmountSpent': total_amount_spent,
    'VisitFrequency': visit_frequency
})

kmeans = KMeans(n_clusters=3, n_init=10)
kmeans.fit(data)
cluster_labels = kmeans.labels_

# Add cluster labels to the DataFrame
data['Cluster'] = cluster_labels

plt.figure(figsize=(8, 6))
for cluster_id in range(3):
    plt.scatter(
        data[data['Cluster'] == cluster_id]['TotalAmountSpent'],
        data[data['Cluster'] == cluster_id]['VisitFrequency'],
        label=f'Cluster {cluster_id}'
    )
plt.xlabel('Total Amount Spent')
plt.ylabel('Visit Frequency')
plt.title('Customer Segmentation using K-Means Clustering')
plt.legend()
plt.show()
```

36. **Scenario:** You are a data analyst working for a finance company. Your team is interested in analyzing the variability of stock prices for a particular company over a certain period. The company's stock data includes the closing prices for each trading day of the specified period.

Question: Your task is to build a Python program that reads the stock data from a CSV file, calculates the variability of stock prices, and provides insights into the stock's price movements.

```
import pandas as pd

# Sample stock price data
data = {
    'Date': ['2023-08-01', '2023-08-02', '2023-08-03', '2023-08-04', '2023-08-05'],
    'ClosingPrice': [100, 105, 102, 98, 110]
}
```

```
stock_data = pd.DataFrame(data)
stock_data['PriceChange'] = stock_data['ClosingPrice'].diff()

mean_change = stock_data['PriceChange'].mean()
std = stock_data['PriceChange'].std()

print("Stock Price Variability Analysis")
print(f"Mean Daily Price Change: {mean_change:.2f}")
print(f"Standard Deviation of Daily Price Changes: {std:.2f}")

positive_changes = stock_data[stock_data['PriceChange'] > 0]['PriceChange'].count()
negative_changes = stock_data[stock_data['PriceChange'] < 0]['PriceChange'].count()

print("\nStock Movement Direction")
print(f"Days with Positive Price Change: {positive_changes}")
print(f"Days with Negative Price Change: {negative_changes}")


'''
sample output
Stock Price Variability Analysis
Mean Daily Price Change: 2.50
Standard Deviation of Daily Price Changes: 7.51

Stock Movement Direction
Days with Positive Price Change: 2
Days with Negative Price Change: 2
'''
```

37. **Scenario:** You are a data scientist working for an educational institution, and you want to explore the correlation between students' study time and their exam scores. You have collected data from a group of students, noting their study time in hours and their corresponding scores in an exam.

Question: Identify any potential correlation between study time and exam scores and explore various plotting functions to visualize this relationship effectively.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

np.random.seed(0)
num_students = 50
study_time = np.random.uniform(1, 10, size=num_students)
exam_scores = 50 + 10 * study_time + np.random.randn(num_students) * 5

data = pd.DataFrame({'StudyTime': study_time, 'ExamScore': exam_scores})
```

```
correlation = data['StudyTime'].corr(data['ExamScore'])
# Scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(data['StudyTime'], data['ExamScore'])
plt.xlabel('Study Time (hours)')
plt.ylabel('Exam Score')
plt.title('Scatter Plot of Study Time vs Exam Score')
plt.show()

# Correlation heatmap
correlation_matrix = data.corr()
plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap')
plt.show()

print(f"Correlation coefficient: {correlation:.2f}")


'''
sample output
Correlation coefficient: 0.98
'''
```

38**. Scenario:** You work for a weather data analysis company, and your team is responsible for developing a program to calculate and analyze variability in temperature data for different cities.

Question: Write a python program will take in a dataset containing daily temperature readings for each city over a year and perform the following tasks:

1. Calculate the mean temperature for each city.
2. Calculate the standard deviation of temperature for each city.
3. Determine the city with the highest temperature range (difference between the highest and lowest temperatures).
4. Find the city with the most consistent temperature (the lowest standard deviation).

```
import pandas as pd
import numpy as np
data = {
    'City': ['City A', 'City A', 'City A', 'City B', 'City B', 'City B', 'City C', 'City C', 'City C'],
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-01', '2023-01-02', '2023-01-03', '2023-01-01', '2023-01-02', '2023-01-03'],
    'Temperature': [25, 28, 30, 20, 22, 23, 18, 20, 19]}
df = pd.DataFrame(data)
# Convert 'Date' to datetime and set as index
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
#mean temperature for each city
```

```
mean = df.groupby('City')['Temperature'].mean()
#standard deviation of temperature for each city
std= df.groupby('City')['Temperature'].std()
#temperature range for each city
tempranges = df.groupby('City')['Temperature'].apply(lambda x: x.max() -
   x.min())
print("Mean Temperature for Each City:")
print(mean)
print("\nStandard Deviation of Temperature for Each City:")
print(std)
print(f"\nCity with the Highest Temperature Range: {tempranges.idxmax()}")
print(f"City with the Most Consistent Temperature: {std.idxmin()}")

'''
sample output
Mean Temperature for Each City:
City
City A    27.666667
City B    21.666667
City C    19.000000
Name: Temperature, dtype: float64

Standard Deviation of Temperature for Each City:
City
City A    2.516611
City B    1.527525
City C    1.000000
Name: Temperature, dtype: float64

City with the Highest Temperature Range: City A
City with the Most Consistent Temperature: City C
'''
```

39. **Scenario:** You work as a data scientist for a marketing agency, and one of your clients is a large e-commerce company. The company wants to understand the purchasing behavior of its customers and segment them into different groups based on their buying patterns. The e-commerce company has provided you with transaction data, including customer IDs, the total amount spent in each transaction, and the number of items purchased.

Question: Build a clustering model using the K-Means algorithm to group customers based on their spending and purchase behavior and visualize the clusters using scatter plots or other appropriate visualizations to gain insights into customer distribution and distinguish different segments.

```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

```
np.random.seed(0)
num_customers = 200
total_amount_spent = np.random.randint(50, 1000, size=num_customers)
num_items_purchased = np.random.randint(1, 30, size=num_customers)
data = pd.DataFrame({
    'TotalAmountSpent': total_amount_spent,
    'NumItemsPurchased': num_items_purchased})
num_clusters = 3  # Number of clusters
kmeans = KMeans(n_clusters=num_clusters, random_state=0)
cluster_labels = kmeans.fit_predict(data)
# Add cluster labels to the DataFrame
data['Cluster'] = cluster_labels

plt.figure(figsize=(8, 6))
for cluster_id in range(num_clusters):
    plt.scatter(
        data[data['Cluster'] == cluster_id]['TotalAmountSpent'],
        data[data['Cluster'] == cluster_id]['NumItemsPurchased'],
        label=f'Cluster {cluster_id}'
    )
plt.xlabel('Total Amount Spent')
plt.ylabel('Number of Items Purchased')
plt.title('Customer Segmentation using K-Means Clustering')
plt.legend()
plt.show()
```

40. **Scenario:** You are a data analyst working for a sports analytics company. The company has collected data on various soccer players, including their names, ages, positions, number of goals scored, and weekly salaries. Create dataset on your own and store in a CSV file.

Question: Develop a Python program to read the data from the CSV file into a pandas data frame, to find the top 5 players with the highest number of goals scored and the top 5 players with the highest salaries. Also calculate the average age of players and display the names of players who are above the average age and visualize the distribution of players based on their positions using a bar chart.

```
import pandas as pd
import matplotlib.pyplot as plt

# Sample data
data = {
    'Name': ['Lionel Messi', 'Cristiano Ronaldo', 'Neymar Jr', 'Kylian Mbappé',
'Robert Lewandowski',
             'Luka Modric', 'Sergio Ramos'],
    'Age': [34, 36, 29, 22, 33, 35, 35],
    'Position': ['Forward', 'Forward', 'Forward', 'Midfielder', 'Forward',
'Midfielder', 'Defender'],
    'Goals': [45, 40, 30, 35, 38, 12, 5],
```

```
    'Salary': [1000000, 900000, 800000, 750000, 700000, 500000, 400000]
}

data = pd.DataFrame(data)

average_age = data['Age'].mean()
above_average_age_players = data[data['Age'] > average_age]['Name']

position_counts = data['Position'].value_counts()
plt.bar(position_counts.index, position_counts.values)
plt.xlabel('Position')
plt.ylabel('Number of Players')
plt.title('Distribution of Players Based on Positions')
plt.xticks(rotation=45)
plt.show()

print("Top 5 Goal Scorers:")
print(data.nlargest(5, 'Goals'))
print("\nTop 5 Highest Salaries:")
print(data.nlargest(5, 'Salary'))
print(f"\nAverage Age of Players: {average_age:.2f}")
print("\nPlayers Above Average Age:")
print(above_average_age_players)


'''
sample output
Top 5 Goal Scorers:
                Name  Age    Position  Goals   Salary
0       Lionel Messi   34     Forward     45  1000000
1  Cristiano Ronaldo   36     Forward     40   900000
4  Robert Lewandowski   33     Forward     38   700000
3      Kylian Mbappé   22  Midfielder     35   750000
2          Neymar Jr   29     Forward     30   800000

Top 5 Highest Salaries:
                Name  Age    Position  Goals   Salary
0       Lionel Messi   34     Forward     45  1000000
1  Cristiano Ronaldo   36     Forward     40   900000
2          Neymar Jr   29     Forward     30   800000
3      Kylian Mbappé   22  Midfielder     35   750000
4  Robert Lewandowski   33     Forward     38   700000

Average Age of Players: 32.00

Players Above Average Age:
0         Lionel Messi
1    Cristiano Ronaldo
```

```
4    Robert Lewandowski
5            Luka Modric
6          Sergio Ramos
Name: Name, dtype: object
'''
```