11. 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?

# CODE:

import numpy as np

fuel\_efficiency = np.array([25, 30, 28, 32, 26, 29, 31, 27, 33, 28])

average\_fuel\_efficiency = np.mean(fuel\_efficiency)

print("Average Fuel Efficiency:", average\_fuel\_efficiency)

model\_1 = 0

model\_2 = 3

fuel\_efficiency\_model\_1 = fuel\_efficiency[model\_1]

fuel\_efficiency\_model\_2 = fuel\_efficiency[model\_2]

percentage\_improvement = ((fuel\_efficiency\_model\_2 - fuel\_efficiency\_model\_1) / fuel\_efficiency\_model\_1) \* 100

print(f"Percentage Improvement from Model {model\_1} to Model {model\_2}: {percentage\_improvement:.2f}%")

### OUTPUT:

Average Fuel Efficiency: 28.9

Percentage Improvement from Model 0 to Model 3: 28.00%

12. 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?

# CODE:

import pandas as pd

sales\_data = pd.DataFrame({

'Product': ['A', 'B','C','D','E','F','G','H','I','K'],

'Quantity': [10, 5, 8, 12, 3, 15, 9, 7, 14, 6],

'Date': ['2023-10-05', '2023-10-10', '2023-10-15', '2023-10-20', '2023-10-25', '2023-10-05', '2023-10-20', '2023-10-10', '2023-10-15', '2023-10-25']

})

sales\_data['Date'] = pd.to\_datetime(sales\_data['Date'])

start\_date = pd.to\_datetime('2023-10-01')

end\_date = pd.to\_datetime('2023-10-31')

filtered\_data = sales\_data[(sales\_data['Date'] >= start\_date) & (sales\_data['Date'] <= end\_date)]

product\_sales = filtered\_data.groupby('Product')['Quantity'].sum().reset\_index()

sorted\_products = product\_sales.sort\_values(by='Quantity', ascending=False)

top\_5\_products = sorted\_products.head(5)

print(top\_5\_products)

### OUPUT:

Product Quantity

5 F 15

8 I 14

3 D 12

0 A 10

6 G 9

13. 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?

## CODE:

from collections import Counter

import re

# Sample customer reviews dataset

reviews = [

"The product is amazing. I love it!",

"Not satisfied with the quality. Disappointing.",

"Great value for the price. Highly recommended.",

"Terrible experience. Would not buy again."

]

# Combine all reviews into a single string

all\_reviews = ' '.join(reviews)

# Remove non-alphanumeric characters and convert to lowercase

cleaned\_reviews = re.sub(r'[^a-zA-Z\s]', '', all\_reviews).lower()

# Tokenize the text into words

words = cleaned\_reviews.split()

# Calculate the frequency distribution of words

word\_freq = Counter(words)

# Print the word frequency distribution

print("Word Frequency Distribution:")

for word, count in word\_freq.items():

print(f"{word}: {count}")

### OUTPUT:

Word Frequency Distribution:

the: 3

product: 1

is: 1

amazing: 1

i: 1

love: 1

it: 1

not: 2

satisfied: 1

with: 1

quality: 1

disappointing: 1

great: 1

value: 1

for: 1

price: 1

highly: 1

recommended: 1

terrible: 1

experience: 1

would: 1

buy: 1

again: 1

14. 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.

## CODE:

import pandas as pd

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from collections import Counter

import matplotlib.pyplot as plt

# Check if NLTK data is present, otherwise download it

try:

nltk.data.find('corpora/stopwords.zip')

except LookupError:

nltk.download('stopwords')

nltk.download('punkt')

def load\_dataset(file\_path):

# Load dataset from CSV file

df = pd.read\_csv(file\_path)

return df

def preprocess\_text(text):

# Convert to lowercase

text = text.lower()

# Remove punctuation

text = ''.join([char for char in text if char.isalnum() or char.isspace()])

# Remove stop words

stop\_words = set(stopwords.words('english'))

word\_tokens = word\_tokenize(text)

filtered\_text = [word for word in word\_tokens if word not in stop\_words]

return ' '.join(filtered\_text)

def calculate\_word\_frequency(text):

# Tokenize the text and calculate word frequencies

words = word\_tokenize(text)

word\_freq = Counter(words)

return word\_freq

def display\_top\_words(word\_freq, top\_n):

# Display the top N most frequent words

top\_words = word\_freq.most\_common(top\_n)

print(f"\nTop {top\_n} most frequent words and their frequencies:")

for word, freq in top\_words:

print(f"{word}: {freq}")

def plot\_word\_frequency(word\_freq, top\_n):

top\_words = dict(word\_freq.most\_common(top\_n))

plt.figure(figsize=(10, 6))

plt.bar(top\_words.keys(), top\_words.values(), color='skyblue')

plt.xlabel('Words')

plt.ylabel('Frequency')

plt.title(f'Top {top\_n} Most Frequent Words')

plt.xticks(rotation=45, ha='right')

plt.show()

# Main function

if \_\_name\_\_ == "\_\_main\_\_":

data = {'feedback': ["Great service! Will definitely come back.",

"The product was faulty. Very disappointed.",

"Amazing experience with friendly staff.",

"Poor customer service. Will not recommend.",

"Quick delivery and good quality product."]}

# Create a DataFrame

df = pd.DataFrame(data)

df.to\_csv('data.csv', index=False)

file\_path = 'data.csv'

df = load\_dataset(file\_path)

df['cleaned\_feedback'] = df['feedback'].apply(preprocess\_text)

all\_text = ' '.join(df['cleaned\_feedback'])

word\_freq = calculate\_word\_frequency(all\_text)

# Get user input for the top N most frequent words

top\_n = int(input("Enter the number of top words to display and plot: "))

display\_top\_words(word\_freq, top\_n)

plot\_word\_frequency(word\_freq, top\_n)

### OUTPUT:

Enter the number of top words to display and plot: 4

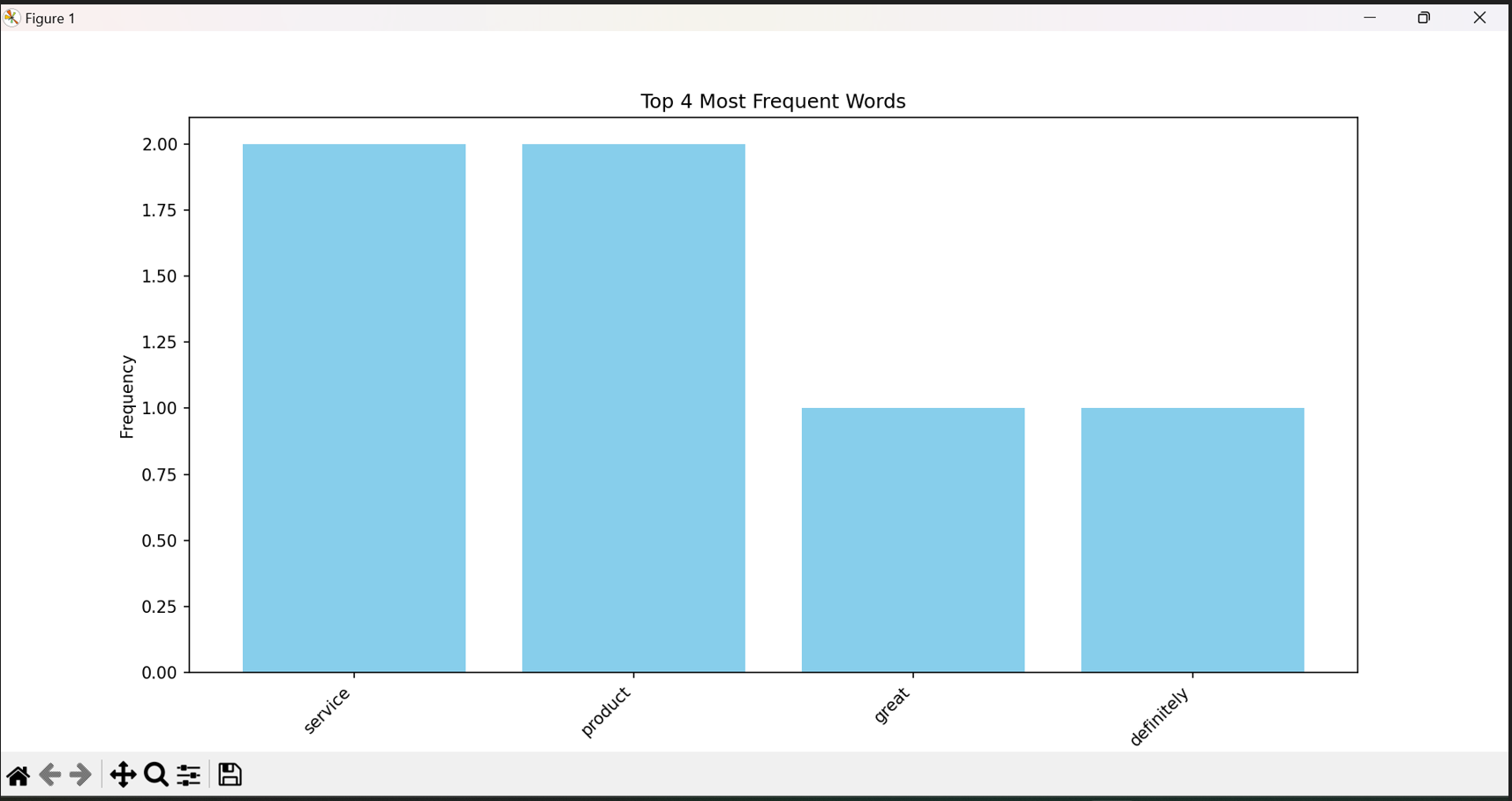
Top 4 most frequent words and their frequencies:

service: 2

product: 2

great: 1

definitely: 1



15. 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

## CODE:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

# Creating a sample DataFrame

np.random.seed(42)

control\_group = np.random.normal(loc=50, scale=10, size=50)

treatment\_group = np.random.normal(loc=55, scale=10, size=50)

data = pd.DataFrame({

'Control': control\_group,

'Treatment': treatment\_group

})

data.head()

# Visualizing the data using boxplots

data.boxplot(column=['Control', 'Treatment'])

plt.ylabel('Value')

plt.title('Boxplot of Control and Treatment Groups')

plt.show()

# Performing a two-sample t-test

t\_stat, p\_value = stats.ttest\_ind(data['Control'], data['Treatment'])

print(f"T-statistic: {t\_stat}")

print(f"P-value: {p\_value}")

# Visualizing the p-value

plt.bar(0, p\_value, align='center', alpha=0.5)

plt.xticks([0], ['P-value'])

plt.ylabel('P-value')

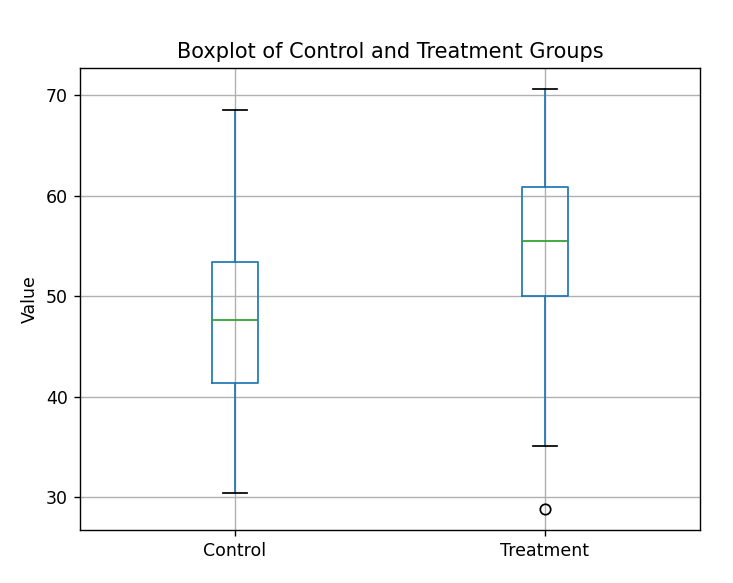
plt.title('P-value of the Two-Sample T-test')

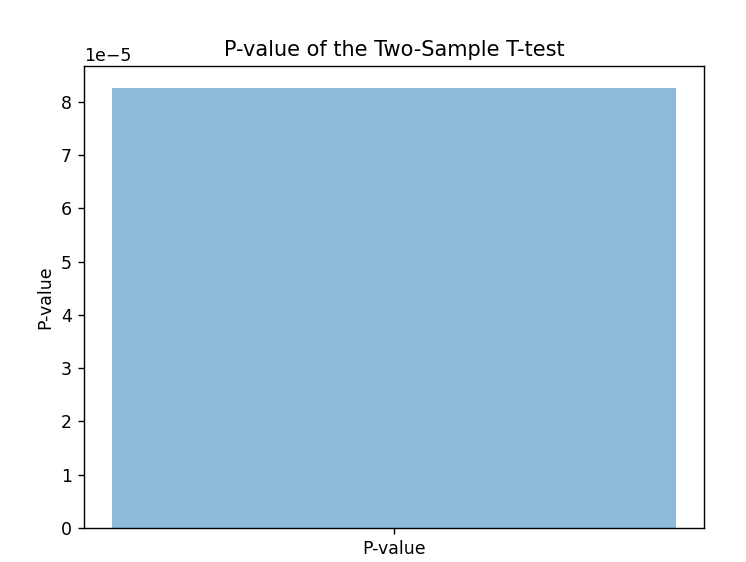
plt.show()

### OUTPUT:

T-statistic: -4.108723928204809

P-value: 8.261945608702611e-05





16. 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.

## CODE:

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))

### OUTPUT:

Accuracy: 0.5000

Precision: 0.5000

Recall: 1.0000

F1 Score: 0.6667

Confusion Matrix:

[[0 1]

 [0 1]]

17. 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.

## CODE:

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()

### OUTPUT:



18. 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).

## CODE:

import pandas as pd

data = {

'City': ['CityA', 'CityA', 'CityA', 'CityB', 'CityB', 'CityB', 'CityC', 'CityC', 'CityC'],

'Date': ['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-01', '2022-01-02', '2022-01-03', '2022-01-01', '2022-01-02', '2022-01-03'],

'Temperature': [20, 22, 18, 25, 23, 26, 30, 28, 32]

}

df = pd.DataFrame(data)

df['Date'] = pd.to\_datetime(df['Date'])

mean\_temperatures = df.groupby('City')['Temperature'].mean()

std\_dev\_temperatures = df.groupby('City')['Temperature'].std()

temperature\_range = df.groupby('City')['Temperature'].max() - df.groupby('City')['Temperature'].min()

city\_with\_highest\_range = temperature\_range.idxmax()

most\_consistent\_city = std\_dev\_temperatures.idxmin()

print("Mean Temperatures:")

print(mean\_temperatures)

print("\nStandard Deviations of Temperatures:")

print(std\_dev\_temperatures)

print(f"\nCity with the Highest Temperature Range: {city\_with\_highest\_range}")

print(f"City with the Most Consistent Temperature: {most\_consistent\_city}")

### OUTPUT:

Mean Temperatures:

City

CityA 20.000000

CityB 24.666667

CityC 30.000000

Name: Temperature, dtype: float64

Standard Deviations of Temperatures:

City

CityA 2.000000

CityB 1.527525

CityC 2.000000

Name: Temperature, dtype: float64

City with the Highest Temperature Range: CityA

City with the Most Consistent Temperature: CityB

19. 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

## CODE:

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

data = {

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

'TotalAmountSpent': [100, 300, 150, 500, 200, 600, 350, 800, 400, 700],

'NumberOfItems': [5, 10, 6, 15, 8, 18, 12, 20, 10, 16]

}

df = pd.DataFrame(data)

X = df[['TotalAmountSpent', 'NumberOfItems']]

# Standardize the features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Choose the number of clusters (you may need to tune this based on your data)

num\_clusters = 3

# Build the K-Means model

kmeans\_model = KMeans(n\_clusters=num\_clusters, random\_state=42)

df['Cluster'] = kmeans\_model.fit\_predict(X\_scaled)

# Visualize the clusters

plt.scatter(df['TotalAmountSpent'], df['NumberOfItems'], c=df['Cluster'], cmap='viridis')

plt.xlabel('Total Amount Spent')

plt.ylabel('Number of Items Purchased')

plt.title('Customer Segmentation with K-Means Clustering')

plt.show()

### OUTPUT:



20. 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.

## CODE:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('players.csv')

top\_goal\_scorers = df.nlargest(5, 'Goals')

print("Top 5 Players with Highest Goals Scored:")

print(top\_goal\_scorers[['Name', 'Goals']])

top\_salary\_earners = df.nlargest(5, 'WeeklySalary')

print("\nTop 5 Players with Highest Salaries:")

print(top\_salary\_earners[['Name', 'WeeklySalary']])

average\_age = df['Age'].mean()

print("\nAverage Age of Players:", average\_age)

above\_average\_age\_players = df[df['Age'] > average\_age]

print("\nPlayers Above Average Age:")

print(above\_average\_age\_players[['Name', 'Age']])

position\_distribution = df['Position'].value\_counts()

position\_distribution.plot(kind='bar', color='skyblue')

plt.title('Player Distribution by Position')

plt.xlabel('Position')

plt.ylabel('Number of Players')

plt.show()

### OUTPUT:

Top 5 Players with Highest Goals Scored:

Name Goals

9 Player10 30

3 Player4 25

6 Player7 22

0 Player1 20

4 Player5 18

Top 5 Players with Highest Salaries:

Name WeeklySalary

9 Player10 7000

3 Player4 6000

6 Player7 5800

4 Player5 5500

0 Player1 5000

Average Age of Players: 26.5

Players Above Average Age:

Name Age

1 Player2 28

3 Player4 30

6 Player7 27

7 Player8 29

9 Player10 31

