1**. Write a python program to accept input strings from the user and display the number of**

**vowels and consonants in the string.** .

def count\_vowels\_and\_consonants(input\_string):

vowels = "aeiouAEIOU"

vowel\_count = 0

consonant\_count = 0

for char in input\_string:

if char.isalpha():

if char in vowels:

vowel\_count += 1

else:

consonant\_count += 1

return vowel\_count, consonant\_count

# Get input from the user

user\_input = input("Enter a string: ")

# Calculate the counts

vowel\_count, consonant\_count = count\_vowels\_and\_consonants(user\_input)

# Display the results

print(f"Number of vowels: {vowel\_count}")

print(f"Number of consonants: {consonant\_count}")

**2. Implement Data Visualization Kindly refers to your own data.**

**1. Draw a pie chart.**

**2. Draw a Bar graph.**

import seaborn as sns

import matplotlib.pyplot as plt

# Load Iris dataset

iris = sns.load\_dataset('iris')

# 1. Draw a pie chart for the distribution of species

species\_counts = iris['species'].value\_counts()

plt.pie(species\_counts, labels=species\_counts.index, autopct='%1.1f%%', startangle=140)

plt.title('Distribution of Iris Species')

plt.show()

# 2. Draw a bar graph for the average sepal length for each species

average\_sepal\_length = iris.groupby('species')['sepal\_length'].mean()

average\_sepal\_length.plot(kind='bar', color='skyblue')

plt.title('Average Sepal Length for Each Iris Species')

plt.xlabel('Species')

plt.ylabel('Average Sepal Length')

plt.show()

**3. Implement Data Visualization Kindly refers to the “Salary\_Data.csv” dataset.**

1. **Draw a Scatter plot for Age and Salary.**
2. **Draw Histogram for Salary. .**

import pandas as pd

import matplotlib.pyplot as plt

# Load the Salary\_Data.csv dataset

iris = pd.read\_csv(‘Iris.csv')

# 1. Draw a Scatter plot for Age and Salary

plt.figure(figsize=(8, 6))

plt.scatter(data['Age'], data['Salary'], color='blue', alpha=0.7)

plt.title('Scatter Plot for Age and Salary')

plt.xlabel('Age')

plt.ylabel('Salary')

plt.grid(True)

plt.show()

# 2. Draw Histogram for Salary

plt.figure(figsize=(8, 6))

plt.hist(data['Salary'], bins=20, color='green', edgecolor='black')

plt.title('Histogram for Salary')

plt.xlabel('Salary')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

**4. Implement clustering algorithm on given “income.csv” dataset and display it using scatter**

**plot.**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import StandardScaler

# Load the "income.csv" dataset

data = pd.read\_csv('income.csv')

# Drop rows with missing values for simplicity

data = data.dropna()

# Encode categorical variables using Label Encoding

le = LabelEncoder()

data['SEX'] = le.fit\_transform(data['SEX'])

data['MARITAL.STATUS'] = le.fit\_transform(data['MARITAL.STATUS'])

data['EDUCATION'] = le.fit\_transform(data['EDUCATION'])

data['OCCUPATION'] = le.fit\_transform(data['OCCUPATION'])

data['AREA'] = le.fit\_transform(data['AREA'])

data['DUAL.INCOMES'] = le.fit\_transform(data['DUAL.INCOMES'])

data['HOME.TYPE'] = le.fit\_transform(data['HOME.TYPE'])

data['ETHNIC.CLASS'] = le.fit\_transform(data['ETHNIC.CLASS'])

data['LANGUAGE'] = le.fit\_transform(data['LANGUAGE'])

# Select relevant features for clustering

features = ['AGE', 'INCOME']

X = data[features]

# Standardize the data

scaler = StandardScaler()

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

# Apply k-means clustering

kmeans = KMeans(n\_clusters=3, random\_state=42)

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

# Display the scatter plot

plt.figure(figsize=(8, 6))

for cluster in data['Cluster'].unique():

cluster\_data = data[data['Cluster'] == cluster]

plt.scatter(cluster\_data['AGE'], cluster\_data['INCOME'], label=f'Cluster {cluster}')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=200, c='red', marker='X', label='Centroids')

plt.title('Clustering of Age and Income')

plt.xlabel('Age')

plt.ylabel('Income')

plt.legend()

plt.show()

**5. Write a program to check whether a number is Harshad number or Not.**

def is\_harshad\_number(number):

# Convert the number to a list of digits

digits = [int(digit) for digit in str(number)]

# Calculate the sum of the digits

digit\_sum = sum(digits)

# Check if the number is divisible by the sum of its digits

return number % digit\_sum == 0

# Get input from the user

user\_input = int(input("Enter a number: "))

# Check if the number is a Harshad number

if is\_harshad\_number(user\_input):

print(f"{user\_input} is a Harshad number.")

else:

print(f"{user\_input} is not a Harshad number.")

**6. Implement Data Visualization Kindly refers to your own data.**

1. **Draw scatter plot diagram**
2. **Draw Histogram**.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the Iris dataset

iris = sns.load\_dataset('iris')

# 1. Draw a Scatter plot diagram

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

sns.scatterplot(x='sepal\_length', y='sepal\_width', hue='species', data=iris)

plt.title('Scatter Plot of Sepal Length vs Sepal Width')

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Sepal Width (cm)')

plt.legend()

plt.show()

# 2. Draw Histogram

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

sns.histplot(data=iris, x='petal\_length', kde=True, hue='species', multiple='stack', bins=20)

plt.title('Histogram of Petal Length by Species')

plt.xlabel('Petal Length (cm)')

plt.ylabel('Frequency')

plt.legend()

plt.show()

**7. Write a python program to accept input from users and check whether the number is**

**Armstrong or not.**

def is\_armstrong\_number(number):

num\_str = str(number)

num\_digits = len(num\_str)

armstrong\_sum = sum(int(digit) \*\* num\_digits for digit in num\_str)

return armstrong\_sum == number

# Get input from the user

user\_input = int(input("Enter a number: "))

# Check if the number is an Armstrong number

if is\_armstrong\_number(user\_input):

print(f"{user\_input} is an Armstrong number.")

else:

print(f"{user\_input} is not an Armstrong number.")

**8. Implement Classification algorithm KNN classifier Data Analysis on given iris dataset.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

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

# Preparing data with scaling

feature\_names = ['sepal length, 'sepal width', 'petal length', 'petal width']

X = iris\_df[feature\_names]

y = iris\_df['variety']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

scaler = MinMaxScaler()

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

X\_test = scaler.transform(X\_test)

# K-Nearest Neighbors (KNN) classifier

knn\_classifier = KNeighborsClassifier(n\_neighbors=3)

knn\_classifier.fit(X\_train, y\_train)

#print score of train data

print('Accuracy of knn classifier on training set:{:.2f}'. format(knn\_classifier.score(x\_train, y\_train)))

#print score of test data

print('Accuracy of knn classifier on test set:{:.2f}'. format(knn\_classifier.score(x\_test,

y\_test)))

**9. Implement Classification algorithm Decision tree Data Analysis on given iris dataset.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

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

# Preparing data with scaling

feature\_names = ['sepal length, 'sepal width', 'petal length', 'petal width']

X = iris\_df[feature\_names]

y = iris\_df['variety']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

scaler = MinMaxScaler()

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

X\_test = scaler.transform(X\_test)

# Decision Tree classifier

DTC = DecisionTreeClassifier(random\_state=0)

DTC.fit(X\_train, y\_train)

#print score of train data

print('Accuracy of decision tree classifier on training set:{:.2f}'. format(DTC.score(x\_train, y\_train)))

#print score of test data

print('Accuracy of decision tree classifier on test set:{:.2f}'. format(DTC.score(x\_test,

y\_test)))

**10.Write a Python program to print factorials of numbers using Recursion.**

def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n - 1)

# Get input from the user

num = int(input("Enter a number: "))

# Check if the input is non-negative

if num < 0:

print("Factorial is not defined for negative numbers.")

else:

result = factorial(num)

print(f"The factorial of {num} is: {result}")

**11. Implement Data Visualization Kindly refers to the “income.csv” dataset.**

1. **Draw a scatter plot for Age and Income.**
2. **Draw a bar graph for Age and Income.**

import pandas as pd

import matplotlib.pyplot as plt

# Load the "income.csv" dataset

data = pd.read\_csv('income.csv')

# 1. Draw a scatter plot for Age and Income

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

plt.scatter(data['AGE'], data['INCOME'], color='blue', alpha=0.7)

plt.title('Scatter Plot for Age and Income')

plt.xlabel('Age')

plt.ylabel('Income')

plt.grid(True)

plt.show()

# 2. Draw a bar graph for Age and Income

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

plt.bar(data['AGE'], data['INCOME'], color='green', alpha=0.7)

plt.title('Bar Graph for Age and Income')

plt.xlabel('Age')

plt.ylabel('Income')

plt.grid(axis='y')

plt.show()

**12. Implement clustering algorithm on given “income.csv” dataset and display it using scatter**

**plot on given iris dataset.**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

data = pd.read\_csv('income.csv')

# Select relevant features for clustering

features = ['AGE', 'INCOME']

X = data[features]

# Standardize the data

scaler = StandardScaler()

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

# Apply k-means clustering

kmeans = KMeans(n\_clusters=3, random\_state=42)

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

# Display the scatter plot

plt.figure(figsize=(8, 6))

for cluster in data['Cluster'].unique():

cluster\_data = data[data['Cluster'] == cluster]

plt.scatter(cluster\_data['AGE'], cluster\_data['INCOME'], label=f'Cluster {cluster}')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=200, c='red', marker='X', label='Centroids')

plt.title('Clustering of Age and Income')

plt.xlabel('Age')

plt.ylabel('Income')

plt.legend()

plt.show()

**13. Write a program to print the length of String using Recursion.**

def string\_length(s):

if s == "":

return 0

else:

return 1 + string\_length(s[1:])

# Get input from the user

user\_input = input("Enter a string: ")

# Print the length of the string using recursion

print(f"The length of the string is: {string\_length(user\_input)}")

**14.Write a python program to count the occurrence of each word in a given sentence**

def count\_word\_occurrences(sentence):

words = sentence.split()

word\_counts = {}

for word in words:

word\_counts[word] = word\_counts.get(word, 0) + 1

return word\_counts

# Get input from the user

user\_sentence = input("Enter a sentence: ")

# Count the occurrence of each word in the sentence

word\_occurrences = count\_word\_occurrences(user\_sentence)

# Print the results

print("Word occurrences:")

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

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

**15. Implement Classification algorithm SVM Classifier Data Analysis on given iris dataset.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.svm import SVC

# Load the "iris.csv" dataset

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

# Preparing data with scaling

feature\_names = ['sepal length', 'sepal width', 'petal length', 'petal width']

X = df[feature\_names]

y = df['variety']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

scaler = MinMaxScaler()

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

X\_test = scaler.transform(X\_test)

# SVM classifier

svm\_classifier = SVC(random\_state=0)

svm\_classifier.fit(X\_train, y\_train)

#print score of train data

print('Accuracy of SVM classifier on training set:{:.2f}'. format(svm\_classifier.score(x\_train, y\_train)))

#print score of test data

print('Accuracy of SVM classifier on test set:{:.2f}'. format(svm\_classifier.score(x\_test,

y\_test)))

**16. Write a python program to print prime numbers between 1 to 100.**

def is\_prime(num):

if num < 2:

return False

for i in range(2, int(num\*\*0.5) + 1):

if num % i == 0:

return False

return True

# Print prime numbers between 1 to 100

print("Prime numbers between 1 to 100:")

for number in range(1, 101):

if is\_prime(number):

print(number, end=" ")

**17.Write a python program to find the quadrants in which coordinates lie, get the value of x and**

**y coordinates as input from the user and check in which quadrants the point lies and print it.**

# Get input from the user

x = float(input("Enter the x-coordinate: "))

y = float(input("Enter the y-coordinate: "))

# Check in which quadrant the point lies

if x > 0 and y > 0:

print("The point lies in Quadrant I.")

elif x < 0 and y > 0:

print("The point lies in Quadrant II.")

elif x < 0 and y < 0:

print("The point lies in Quadrant III.")

elif x > 0 and y < 0:

print("The point lies in Quadrant IV.")

elif x == 0 and y != 0:

print("The point lies on the y-axis.")

elif x != 0 and y == 0:

print("The point lies on the x-axis.")

else:

print("The point lies at the origin (0, 0).")

**18.Implement Classification algorithm Decision Tree Classifier Data Analysis on given iris**

**dataset.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

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

# Preparing data with scaling

feature\_names = ['sepal length, 'sepal width', 'petal length', 'petal width']

X = iris\_df[feature\_names]

y = iris\_df['variety']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

scaler = MinMaxScaler()

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

X\_test = scaler.transform(X\_test)

# Decision Tree classifier

DTC = DecisionTreeClassifier(random\_state=0)

DTC.fit(X\_train, y\_train)

#print score of train data

print('Accuracy of decision tree classifier on training set:{:.2f}'. format(DTC.score(x\_train, y\_train)))

#print score of test data

print('Accuracy of decision tree classifier on test set:{:.2f}'. format(DTC.score(x\_test,

y\_test)))

**19.Write a python program to Implement Data Visualization Kindly refer to your own data.**

1. **Draw a horizontal bar graph.**
2. **Draw scatter plot diagram.**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the Iris dataset

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

# 1. Draw a horizontal bar graph

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

sns.countplot(y='variety', data=df, palette='viridis')

plt.title('Count of Iris varieties')

plt.xlabel('Count')

plt.ylabel('Variety')

plt.show()

# 2. Draw scatter plot diagram

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

sns.scatterplot(x='sepal length', y='sepal width', hue='variety', data=df, palette='viridis')

plt.title('Scatter Plot of Sepal Length vs Sepal Width')

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Sepal Width (cm)')

plt.show()

**20.Write a python program to accept numbers and check whether numbers are Automorphic or**

**not.**

def is\_automorphic(number):

square = number \*\* 2

num\_digits = len(str(number))

# Extract the last 'num\_digits' digits from the square

square\_digits = square % (10 \*\* num\_digits)

return square\_digits == number

# Get input from the user

user\_number = int(input("Enter a number: "))

# Check if the number is Automorphic

if is\_automorphic(user\_number):

print(f"{user\_number} is an Automorphic number.")

else:

print(f"{user\_number} is not an Automorphic number.")