```
#1a Candidate Elimination Algorithm

#pip install pyarrow
#pip install numpy
#pip install pandas
import numpy as np
import pandas as pd

data = pd.DataFrame(data=pd.read_csv("enjoysport.csv"))
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target = np.array(data.iloc[:,-1])
print(target)

def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h and general_h")
    print(specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in range(len(specific_
    print(general_h)
    for i, h in enumerate(concepts):
        if target[i] == "yes":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'
            print(specific_h)
        print(specific_h)
        if target[i] == "no":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'
        print(" steps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)

    indices = [i for i, val in enumerate(general_h) if val == ['?', '?', '?', '?',
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h

s_final, g_final = learn(concepts, target)
print("Final Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

```
[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
 ['sunny' 'warm' 'high' 'strong' 'warm' 'same']
 ['rainy' 'cold' 'high' 'strong' 'warm' 'change']
 ['sunny' 'warm' 'high' 'strong' 'cool' 'change']]
['yes' 'yes' 'no' 'yes']
initialization of specific_h and general_h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
 steps of Candidate Elimination Algorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
 steps of Candidate Elimination Algorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?
['sunny' 'warm' '?' 'strong' 'warm' 'same']
 steps of Candidate Elimination Algorithm 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?',
['sunny' 'warm' '?' 'strong' '?' '?']
['sunny' 'warm' '?' 'strong' '?' '?']
 steps of Candidate Elimination Algorithm 4
['sunny' 'warm' '?' 'strong' '?' '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?',
```

```
Final Specific_h:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
```

```
#1b Interchanging Numbers

def swap_first_last_3(lst):
    if len(lst)>=2:
        lst=lst[-1:]+lst[1:-1]+lst[:1]
    return lst
inp=[12,35,9,56,24]
print("The Original input is:",inp)
result=swap_first_last_3(inp)
print("The Output after swapping is:",result)
```

```
The Original input is: [12, 35, 9, 56, 24]
The Output after swapping is: [24, 35, 9, 56, 12]
```

```
!pip install pgmpy
```

```
Collecting pgmpy
  Downloading pgmpy-0.1.25-py3-none-any.whl (2.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 21.6 MB/s eta 0:00
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/di
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dis
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.10/dist
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/pytho
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/di
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/p
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-pac
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch->pgmpy)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.w
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch->pgmpy)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch->pgmpy)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.w
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch->pgmpy)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (7
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch->pgmpy)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch->pgmpy)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (
Collecting nvidia-curand-cu12==10.3.2.106 (from torch->pgmpy)
  Using cached nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch->pgmpy)
  Using cached nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.w
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch->pgmpy)
  Using cached nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.w
Collecting nvidia-nccl-cu12==2.20.5 (from torch->pgmpy)
  Using cached nvidia_nccl_cu12-2.20.5-py3-none-manylinux2014_x86_64.whl (1
Collecting nvidia-nvtx-cu12==12.1.105 (from torch->pgmpy)
  Using cached nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99
Requirement already satisfied: triton==2.3.0 in /usr/local/lib/python3.10/d
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->to
  Downloading nvidia_nvjitlink_cu12-12.5.40-py3-none-manylinux2014_x86_64.w
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.3/21.3 MB 62.0 MB/s eta 0:
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/di
```

```
#1a Candidate Elimination Algorithm

#pip install pyarrow
#pip install numpy
#pip install pandas
import numpy as np
import pandas as pd

data = pd.DataFrame(data=pd.read_csv("enjoysport
concepts = np.array(data.iloc[:,0:-1])
print(concepts)
target = np.array(data.iloc[:,-1])
print(target)

def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("initialization of specific_h and gener
    print(specific_h)
    general_h = [["?" for i in range(len(specifi
    print(general_h)
    for i, h in enumerate(concepts):
        if target[i] == "yes":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'
            print(specific_h)
        print(specific_h)
        if target[i] == "no":
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    general_h[x][x] = specific_h
                else:
                    general_h[x][x] = '?'
        print(" steps of Candidate Elimination A
        print(specific_h)
        print(general_h)

    indices = [i for i, val in enumerate(general_
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?
    return specific_h, general_h

s_final, g_final = learn(concepts, target)
print("Final Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

```
[['sunny' 'warm' 'normal' 'strong' 'warm' 'same
 ['sunny' 'warm' 'high' 'strong' 'warm' 'same']
 ['rainy' 'cold' 'high' 'strong' 'warm' 'change
 ['sunny' 'warm' 'high' 'strong' 'cool' 'change
['yes' 'yes' 'no' 'yes']
initialization of specific_h and general_h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
 steps of Candidate Elimination Algorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
 steps of Candidate Elimination Algorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
['sunny' 'warm' '?' 'strong' 'warm' 'same']
 steps of Candidate Elimination Algorithm 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm
['sunny' 'warm' '?' 'strong' '?' '?']
['sunny' 'warm' '?' 'strong' '?' '?']
 steps of Candidate Elimination Algorithm 4
['sunny' 'warm' '?' 'strong' '?' '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm
Final Specific_h:
```

```
Installing collected packages: nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvi
Successfully installed nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-1
```

```
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm
```

```
#2a CORONA infection

#pip install pgmpy
import pandas as pd
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination

data = pd.DataFrame({
    'Fever': ['Yes', 'No', 'Yes', 'No', 'Yes'],
    'Cough': ['Yes', 'Yes', 'No', 'Yes', 'No'],
    'SoreThroat': ['No', 'Yes', 'No', 'No', 'Yes'],
    'RunningNose': ['No', 'Yes', 'No', 'Yes', 'No'],
    'Fatigue': ['Yes', 'Yes', 'Yes', 'No', 'No'],
    'Asthma': ['No', 'No', 'Yes', 'No', 'Yes'],
    'ChronicLungDisease': ['No', 'No', 'No', 'No', 'Yes'],
    'Headache': ['Yes', 'No', 'No', 'Yes', 'Yes'],
    'HeartDisease': ['No', 'Yes', 'No', 'No', 'No'],
    'Diabetes': ['No', 'Yes', 'Yes', 'No', 'No'],
    'Hypertension': ['No', 'Yes', 'Yes', 'Yes', 'No'],
    'COVID': ['Yes', 'Yes', 'No', 'No', 'Yes'],
    'Breathlessness': ['Yes', 'No', 'No', 'No', 'Yes']
})


model = BayesianNetwork([
    ('Fever', 'COVID'), ('Cough', 'COVID'), ('SoreThroat', 'COVID'),
    ('RunningNose', 'COVID'), ('Fatigue', 'COVID'), ('Asthma', 'COVID'),
    ('ChronicLungDisease', 'COVID'), ('Headache', 'COVID'), ('HeartDisease', 'COVI
    ('Diabetes', 'COVID'), ('Hypertension', 'COVID'), ('COVID', 'Breathlessness')
])


model.fit(data)

infer = VariableElimination(model)


print("Please provide the following symptoms (Yes/No):")
evidence = {}
for variable in data.columns[:-2]:
    evidence[variable] = input(f"{variable}: ")


query = infer.query(variables=['COVID'], evidence=evidence)
print(query)
```

```
Please provide the following symptoms (Yes/No):
Fever: Yes
Cough: Yes
SoreThroat: Yes
RunningNose: No
Fatigue: Yes
Asthma: No
ChronicLungDisease: Yes
Headache: No
HeartDisease: Yes
Diabetes: No
Hypertension: Yes
+------------+--------------+
| COVID      |   phi(COVID) |
+============+==============+
| COVID(No)  |       0.5000 |
+------------+--------------+
| COVID(Yes) |       0.5000 |
+------------+--------------+
```

```
#2b PrimeNumber less than 256

def is_prime(n):
    if n <= 1:
        return False
    if n <= 2:
        return True
    if n % 2 == 0:
        return False
    i=3
    while i*i <= n:
        if n % i == 0:
            return False
        i += 2
    return True
print("Prime numbers less than 256:")
for num in range(2, 256):
    if is_prime(num):
        print(num,end=" ")
```

```
Prime numbers less than 256:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103
```

```
#4a Probability

pAF=0.03
print("The probability that it is Friday and that a student is absent :",pAF)
# The probability that it is Friday is 20%
pF=0.2
print("The probability that it is Friday : ",pF)
# The probability that a student is absent given that today is Friday
pResult=(pAF/pF)

print("The probability that a student is absent given that today is Friday : ",pRe
```

```
The probability that it is Friday and that a student is absent : 0.03
The probability that it is Friday :  0.2
The probability that a student is absent given that today is Friday :  15.0 %
```

```
#4b largest number

def find_largest(num1, num2, num3):
    if num1 >= num2 and num1 >= num3:
        largest = num1
    elif num2 >= num1 and num2 >= num3:
        largest = num2
    else:
        largest = num3
    return largest

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
num3 = float(input("Enter third number: "))


largest_number = find_largest(num1, num2, num3)
print(f"The largest number is: {largest_number}")
```

```
Enter first number: 10
Enter second number: 20
Enter third number: 30
The largest number is: 30.0
```

```
#5a Finite word Backpropagation

#pip install pandas
#pip install scikit-learn
#pip install scikit-neuralnetwork
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, rec

msglbl_data = pd.read_csv('Statements-2.csv', names=['Message', 'Label'])
print("The Total instances in the Dataset: ", msglbl_data.shape[0])
msglbl_data['labelnum'] = msglbl_data.Label.map({'pos': 1, 'neg': 0})

X = msglbl_data["Message"]
Y = msglbl_data.labelnum

Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, Y)

count_vect = CountVectorizer()
Xtrain_dims = count_vect.fit_transform(Xtrain)
Xtest_dims = count_vect.transform(Xtest)
df = pd.DataFrame(Xtrain_dims.toarray(),columns=count_vect.get_feature_names_out()
clf = MLPClassifier(solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2), random_s

clf.fit(Xtrain_dims, Ytrain)

prediction = clf.predict(Xtest_dims)
print('******** Accuracy Metrics *********')
print('Accuracy : ', accuracy_score(Ytest, prediction))
print('Recall : ', recall_score(Ytest, prediction))
print('Precision : ',precision_score(Ytest, prediction))
print('Confusion Matrix : \n', confusion_matrix(Ytest, prediction))
print(10*"-")

test_stmt = [input("Enter any statement to predict :")]
test_dims = count_vect.transform(test_stmt)
pred = clf.predict(test_dims)
for stmt,lbl in zip(test_stmt,pred):
    if lbl == 1:
        print("Statement is Positive")
    else:
        print("Statement is Negative")
```

```
The Total instances in the Dataset:  18
******** Accuracy Metrics *********
Accuracy :  0.4
Recall :  0.3333333333333333
Precision :  0.5
Confusion Matrix :
 [[1 1]
 [2 1]]
----------
Enter any statement to predict :I love biriyani
Statement is Positive
```

```
#5b Celsius to Fahrenheit

def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32
def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

def main():
    print("Options:")
    print("1. Celsius to Fahrenheit")
    print("2. Fahrenheit to Celsius")

    option=input("Choose an option(1 or 2):")
    if option == '1':
        celsius = float(input("Enter temperature in Celsius: "))
        print("Temperature in Fahrenheit:", celsius_to_fahrenheit(celsius))
    elif option == '2':
        fahrenheit = float(input("Enter temperature in Fahrenheit: "))
        print("Temperature in Celsius:", fahrenheit_to_celsius(fahrenheit))
    else:
        print("Invalid option")
if __name__ == "__main__":
    main()
#Output 1
```

```
Options:
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
Choose an option(1 or 2):1
Enter temperature in Celsius: 36
Temperature in Fahrenheit: 96.8
```

```
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32
def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

def main():
    print("Options:")
    print("1. Celsius to Fahrenheit")
    print("2. Fahrenheit to Celsius")

    option=input("Choose an option(1 or 2):")
    if option == '1':
        celsius = float(input("Enter temperature in Celsius: "))
        print("Temperature in Fahrenheit:", celsius_to_fahrenheit(celsius))
    elif option == '2':
        fahrenheit = float(input("Enter temperature in Fahrenheit: "))
        print("Temperature in Celsius:", fahrenheit_to_celsius(fahrenheit))
    else:
        print("Invalid option")
if __name__ == "__main__":
    main()
#Output 2
```

```
Options:
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
Choose an option(1 or 2):2
Enter temperature in Fahrenheit: 96
Temperature in Celsius: 35.55555555555556
```

```
#6a k-nearest neighbour

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import random

data_iris = load_iris()

label_target = data_iris.target_names
print()
print("Sample Data from Iris Dataset")
print("*"*30)

for i in range(10):
    rn = random.randint(0,120)
    print(data_iris.data[rn],"===>",label_target[data_iris.target[rn]])

X = data_iris.data
y = data_iris.target


X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size = 0.3, random_state=1)

print("The Training dataset length: ",len(X_train))
print("The Testing dataset length: ",len(X_test))
try:
    nn = int(input("Enter number of neighbors :"))
    knn = KNeighborsClassifier(nn)

    knn.fit(X_train, y_train)

    print("The Score is :",knn.score(X_test, y_test))

    test_data = input("Enter Test Data :").split(",")
    for i in range(len(test_data)):
        test_data[i] = float(test_data[i])
    print()

    v = knn.predict([test_data])
    print("Predicted output is :",label_target[v])
except:
    print("Please supply valid input......")
 #output 1
```

```
Sample Data from Iris Dataset
******************************
[4.7 3.2 1.6 0.2] ===> setosa
[5.  3.2 1.2 0.2] ===> setosa
[5.  3.4 1.6 0.4] ===> setosa
[5.1 3.8 1.5 0.3] ===> setosa
[5.8 2.7 5.1 1.9] ===> virginica
[6.1 2.9 4.7 1.4] ===> versicolor
[5.7 2.5 5.  2. ] ===> virginica
[6.  2.7 5.1 1.6] ===> versicolor
[4.9 3.1 1.5 0.2] ===> setosa
[5.6 2.9 3.6 1.3] ===> versicolor
The Training dataset length:  105
The Testing dataset length:  45
Enter number of neighbors :10
The Score is : 0.9777777777777777
Enter Test Data :6.2,2.6,3.4,0.6

Predicted output is : ['versicolor']
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import random

data_iris = load_iris()

label_target = data_iris.target_names
print()
print("Sample Data from Iris Dataset")
print("*"*30)

for i in range(10):
    rn = random.randint(0,120)
    print(data_iris.data[rn],"===>",label_target[data_iris.target[rn]])

X = data_iris.data
y = data_iris.target

X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size = 0.3, random_state=1)

print("The Training dataset length: ",len(X_train))
print("The Testing dataset length: ",len(X_test))
try:
    nn = int(input("Enter number of neighbors :"))
    knn = KNeighborsClassifier(nn)

    knn.fit(X_train, y_train)

    print("The Score is :",knn.score(X_test, y_test))

    test_data = input("Enter Test Data :").split(",")
    for i in range(len(test_data)):
        test_data[i] = float(test_data[i])
    print()

    v = knn.predict([test_data])
    print("Predicted output is :",label_target[v])
except:
    print("Please supply valid input......")
 #output 2
```

```
 Sample Data from Iris Dataset
 ******************************
 [4.6 3.6 1.  0.2] ===> setosa
 [6.5 3.  5.8 2.2] ===> virginica
 [6.4 3.2 5.3 2.3] ===> virginica
 [7.6 3.  6.6 2.1] ===> virginica
 [5.1 3.5 1.4 0.2] ===> setosa
 [5.1 3.5 1.4 0.2] ===> setosa
 [5.5 2.4 3.8 1.1] ===> versicolor
 [6.7 2.5 5.8 1.8] ===> virginica
 [5.1 3.5 1.4 0.3] ===> setosa
 [6.1 2.9 4.7 1.4] ===> versicolor
 The Training dataset length:  105
 The Testing dataset length:  45
 Enter number of neighbors :10
 The Score is : 0.9777777777777777
 Enter Test Data :5.0,3.3,1.4,0.3

 Predicted output is : ['setosa']
```

```
#6b Exchange the values of two variables

x = 10
y = 50
temp = x
x = y
y = temp

print("Value of x:", x)
print("Value of y:", y)
```

```
Value of x: 50
Value of y: 10
```

```
#7a Decision Tree Algorithm
import pandas as pd
import numpy as np

train_df = pd.read_csv("id3.csv")
features_train = [col for col in train_df.columns if col != "PlayTennis"]
X_train = train_df[features_train]
y_train = train_df["PlayTennis"]

class Node:
    def __init__(self, value=None, children=None, isLeaf=False, pred=None):
        self.value = value
        self.children = children if children is not None else []
        self.isLeaf = isLeaf
        self.pred = pred

def entropy(y):
    _, counts = np.unique(y, return_counts=True)
    probabilities = counts / len(y)
    return -np.sum(probabilities * np.log2(probabilities))

def information_gain(X, y, feature):
    values, counts = np.unique(X[feature], return_counts=True)
    weighted_entropy = np.sum([(counts[i] / np.sum(counts)) * entropy(y[X[feature
    return entropy(y) - weighted_entropy

def ID3(X, y, features):
    if len(np.unique(y)) == 1:
        return Node(isLeaf=True, pred=np.unique(y)[0])
    if len(features) == 0:
        return Node(isLeaf=True, pred=np.unique(y)[np.argmax(np.bincount(y))])

    best_feature = max(features, key=lambda x: information_gain(X, y, x))
    root = Node(value=best_feature)

    for value in np.unique(X[best_feature]):
        subset_indices = X[best_feature] == value
        subset_X, subset_y = X[subset_indices].drop(columns=[best_feature]), y[su
        if len(subset_y) == 0:
            root.children.append(Node(isLeaf=True, pred=np.unique(y)[np.argmax(np
        else:
            child_node = ID3(subset_X, subset_y, [f for f in features if f != bes
            child_node.value = value
            root.children.append(child_node)

    return root

def printTree(root: Node, depth=0):
    if root is None:
        return
    for i in range(depth):
        print("\t", end="")
    if root.value is not None:
        print(root.value, end="")
    if root.isLeaf:
        print(" -> ", root.pred)
    else:
        print()
        for child in root.children:
            printTree(child, depth + 1)

def classify(root: Node, new):
    if root.isLeaf:
        return root.pred
    value = new.get(root.value)
    if value is None:
        return 'yes'
    for child in root.children:
        if child.value == value:
            return classify(child, new)

    return classify(root.children[0], new)
```

```
root = ID3(X_train, y_train, features_train)
```

```
#7b Distance btw two points
import math

def distance_between_points(x1, y1, x2, y2):
    return math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
def circulate_values(*args):
            return args[1:] + (args[0],)

x1, y1 = 1, 2
x2, y2 = 4, 6

distance = distance_between_points(x1, y1, x2, y2)
print("Distance between points:", distance)

n_values = 4

values = (1, 2, 3, 4)
circulated_values = circulate_values(*values)
print("Circulated values:", circulated_values)
```

```
Distance between points: 5.0
Circulated values: (2, 3, 4, 1)
```

```
#9a Linear Regression

#pip install pandas
#pip install matplotlib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dataFrame = pd.read_csv('Age_Income.csv')

age = dataFrame['age']
income = dataFrame['income']
```