Machine Learning Lab

Submitted to: Er. Kamaldeep Kaur

Submitted by: Guneet Kohli

D3 CSE A1

URN:1805172

CRN:1815017

TABLE OF CONTENTS:

EXPERIMENT 1

Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

Code:

```
from google.colab import files
uploaded = files.upload()
'''Uploaded CSV File of Enjoy sport'''

import random
import csv
attributes = [['Sunny','Rainy'],
              ['Warm','Cold'],
              ['Normal','High'],
              ['Strong','Weak'],
              ['Warm','Cool'],
              ['Same','Change']]
num_attributes = len(attributes)
print (" \n The most general hypothesis : ['?','?','?','?','?','?']\n")
print ("\n The most specific hypothesis : ['0','0','0','0','0','0']\n")
a = []
print("\n The Given Training Data Set \n")
with open('ws.csv', 'r') as csvFile:
    reader = csv.reader(csvFile)
    for row in reader:
        a.append (row)
        print(row)
print("\n The initial value of hypothesis: ")
hypothesis = ['0'] * num_attributes
print(hypothesis)
# Comparing with First Training Example
for j in range(0,num_attributes):
        hypothesis[j] = a[0][j];
# Comparing with Remaining Training Examples of Given Data Set
print("\n Find S: Finding a Maximally Specific Hypothesis\n")
for i in range(0,len(a)):
    if a[i][num_attributes]=='Yes':
            for j in range(0,num_attributes):
                if a[i][j]!=hypothesis[j]:
                    hypothesis[j]='?'
                else :
                    hypothesis[j]= a[i][j]
    print(" For Training Example No :{0} the hypothesis is ".format(i),hypothesis)

print("\n The Maximally Specific Hypothesis for a given Training Examples :\n")
print(hypothesis)
```

```
['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'Yes']
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'Yes']
['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'No']
['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'Yes']
```

Dataset used:-

	A	B	C	D	E	F	G
1	morning	sunny	warm	yes	mild	strong	yes
2	evening	rainy	cold	no	mild	normal	no
3	morning	sunny	moderate	yes	normal	normal	yes
4	evening	sunny	cold	yes	high	strong	yes

Enjoy Sport

OUTPUT :-

The most general hypothesis : ['?','?','?','?','?','?']

The most specific hypothesis : ['0','0','0','0','0','0']

The Given Training Data Set

['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes'] ['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes'] ['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No'] ['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes']

The initial value of hypothesis: ['0', '0', '0', '0', '0', '0']

Find S: Finding a Maximally Specific Hypothesis

For Training Example No :0 the hypothesis is ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same'] For Training Example No :1 the hypothesis is ['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same'] For Training Example No :2 the hypothesis is ['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same'] For Training Example No :3 the hypothesis is ['Sunny', 'Warm', '?', 'Strong', '?', '?']

The Maximally Specific Hypothesis for a given Training Examples :

['Sunny', 'Warm', '?', 'Strong', '?', '?'] ['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'Yes']

EXPERIMENT 2

Write a program for Candidate Elimination algorithm for finding the consistent version space based on a given set of training data samples. The training data is read from a .CSV file.

Code:

```
import numpy as np
import pandas as pd
data = pd.read_csv("enjoySport.csv")

concepts = np.array(data)[:,:-1]
print("\nInstances are:\n",concepts)
target = np.array(data)[:,-1]
print("\nTarget Values are: ",target)

def learn(concepts, target):
    specific_h = concepts[0].copy()
    print("\nInitialization of specific_h and genearal_h")
    print("\nSpecific Boundary: ", specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in
range(len(specific_h))]
    print("\nGeneric Boundary: ",general_h)
    for i, h in enumerate(concepts):
        print("\nInstance", i+1 , "is ", h)
        if target[i] == "yes":
            print("Instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'
        if target[i] == "no":
            print("Instance is Negative ")
            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("Specific Bundary after ", i+1, "Instance is ", specific_h)
```

```
        print("Generic Boundary after ", i+1, "Instance is ", general_h)
        print("\n")
    indices = [i for i, val in enumerate(general_h) if val == ['?', '?',
'?', '?', '?', '?']]
    for i in indices:
        general_h.remove(['?', '?', '?', '?', '?', '?'])
    return specific_h, general_h
```

Dataset used:-

	A	B	C	D	E	F	G
1	morning	sunny	warm	yes	mild	strong	yes
2	evening	rainy	cold	no	mild	normal	no
3	morning	sunny	moderate	yes	normal	normal	yes
4	evening	sunny	cold	yes	high	strong	yes

Enjoy Sport

Output:-

```
Instances are:
 [['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
 ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
 ['Morning' 'Sunny' 'Moderate' 'Yes' 'Normal' 'Normal']
 ['Evening' 'Sunny' 'Cold' 'Yes' 'High' 'Strong']]

Target Values are:  ['Yes' 'No' 'Yes' 'Yes']

Initialization of specific_h and genearal_h

Specific Boundary:  ['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']

Generic Boundary:  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 1 is  ['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
Specific Bundary after  1 Instance is  ['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']

Generic Boundary after  1 Instance is  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]


Instance 2 is  ['Evening' 'Rainy' 'Cold' 'No' 'Mild' 'Normal']
Specific Bundary after  2 Instance is  ['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']

Generic Boundary after  2 Instance is  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]


Instance 3 is  ['Morning' 'Sunny' 'Moderate' 'Yes' 'Normal' 'Normal']
Specific Bundary after  3 Instance is  ['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']

Generic Boundary after  3 Instance is  [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
```

```
Instance 4 is  ['Evening' 'Sunny' 'Cold' 'Yes' 'High' 'Strong']
Specific Bundary after  4 Instance is  ['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']

Generic Boundary after  4 Instance is  [['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
```

Final Specific Hypothesis:

```
Final Specific_h:
['Morning' 'Sunny' 'Warm' 'Yes' 'Mild' 'Strong']
Final General_h:
[]
```

EXPERIMENT 3

Write A Program to implement K- Nearest Neighbor Classification model.

Code:

```
from io import StringIO,BytesIO
import pandas as pd
import numpy as np
import math

a=('Name,Age,Gender,Sport\n'
  'Ajay,32,M,Football\n'
  'Mark,40,M,Neither\n'
  'Sara,16,F,Cricket\n'
  'Zara,34,F,Cricket\n'
  'Sachin,55,M,Neither\n'
  'Rahul,40,M,Cricket\n'
  'Pooja,20,F,Neither\n'
  'Smith,15,M,Cricket\n'
  'Laxmi,55,F,Football\n'
  'Michael,15,M,Football\n')


data=pd.read_csv(StringIO(a))
data.to_csv('data.csv')
data
```

Out[3]:

	Name	Age	Gender	Sport
0	Ajay	32	M	Football
1	Mark	40	M	Neither
2	Sara	16	F	Cricket
3	Zara	34	F	Cricket
4	Sachin	55	M	Neither
5	Rahul	40	M	Cricket
6	Pooja	20	F	Neither
7	Smith	15	M	Cricket
8	Laxmi	55	F	Football
9	Michael	15	M	Football

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Gender']= le.fit_transform(data['Gender'])
```

Out[5]:

	Name	Age	Gender	Sport
0	Ajay	32	1	Football
1	Mark	40	1	Neither
2	Sara	16	0	Cricket
3	Zara	34	0	Cricket
4	Sachin	55	1	Neither
5	Rahul	40	1	Cricket
6	Pooja	20	0	Neither
7	Smith	15	1	Cricket
8	Laxmi	55	0	Football
9	Michael	15	1	Football

```
x_data = data
y_data = data['Sport']


x_test = {'Age': 5, 'Gender': 1}


  distance = []
for i in range(len(x_data)):
  temp = math.sqrt((x_test['Age'] - data['Age'][i])**2 + (x_test['Gender'] - data['Ge
nder'][i])**2)
  distance.append(round(temp,2))


distance
```

```
x_data['distance'] = distance
```

```
x_data
```

Out[11]:

	Name	Age	Gender	Sport	distance
0	Ajay	32	1	Football	27.00
1	Mark	40	1	Neither	35.00
2	Sara	16	0	Cricket	11.05
3	Zara	34	0	Cricket	29.02
4	Sachin	55	1	Neither	50.00
5	Rahul	40	1	Cricket	35.00
6	Pooja	20	0	Neither	15.03
7	Smith	15	1	Cricket	10.00
8	Laxmi	55	0	Football	50.01
9	Michael	15	1	Football	10.00

```
k = 3
from heapq import nsmallest
nearest_neighbors = nsmallest(k, distance)
```

```
nearest_neighbors
```

In [13]:
```
nearest_neighbors
```

Out[13]:
```
[10.0, 10.0, 11.05]
```

In [14]:
```
sorted = x_data.sort_values(by=['distance'])
sorted
```

Out[14]:

	Name	Age	Gender	Sport	distance
7	Smith	15	1	Cricket	10.00
9	Michael	15	1	Football	10.00
2	Sara	16	0	Cricket	11.05
6	Pooja	20	0	Neither	15.03
0	Ajay	32	1	Football	27.00
3	Zara	34	0	Cricket	29.02
1	Mark	40	1	Neither	35.00
5	Rahul	40	1	Cricket	35.00
4	Sachin	55	1	Neither	50.00
8	Laxmi	55	0	Football	50.01

Dataset Used:-

	Name	Age	Gender	Sport
0	Ajay	32	M	Football
1	Mark	40	M	Neither
2	Sara	16	F	Cricket
3	Zara	34	F	Cricket
4	Sachin	55	M	Neither
5	Rahul	40	M	Cricket
6	Pooja	20	F	Neither
7	Smith	15	M	Cricket
8	Laxmi	55	F	Football
9	Michael	15	M	Football

Output:

```
In [15]: sports = ['Cricket', 'Football', 'Neither']
         x = sorted.head(3)['Sport'].value_counts()
         x.sort_values(ascending = False)
         x.index[0]

Out[15]: 'Cricket'
```

EXPERIMENT 4

Write A Program to implement K- Means Clustering model.

Code:

```
import pandas as pd
import numpy as np
import math
df = pd.read_csv('data4.csv')
df
```

	x	y
0	2	4
1	2	6
2	5	6
3	4	7
4	8	3
5	6	6
6	5	2
7	5	7
8	6	3
9	4	4

```
def dis(x,y,center):
    return float(math.sqrt((float(x)-float(center[0]))**2+(float(y)-
float(center[1]))**2))


k = int(input("Enter Number of clusters Required : "))
arr = []
for i in range(k):
    print("For centeroid C{} :".format(i))
    x = input("x = ")
    y = input("y = ")
    arr.append((x,y))
```

```
#3 1 5 4 1 8 4 sample TC
```

```
Enter Number of clusters Required : 3
For centeroid C0 :
x = 1
y = 5
For centeroid C1 :
x = 4
y = 1
For centeroid C2 :
x = 8
y = 4
```

```
# initial filling of c1,c2,c3

for i in range(k):

    df[i] = [dis(float(df['x'][j]),float(df['y'][j]),arr[i]) for j in
range(df.shape[0])]

df["result"] = df[[0,1,2]].idxmin(axis = 1)

df
```

	x	y	0	1	2	result
0	2	4	1.414214	3.605551	6.000000	0
1	2	6	1.414214	5.385165	6.324555	0
2	5	6	4.123106	5.099020	3.605551	2
3	4	7	3.605551	6.000000	5.000000	0
4	8	3	7.280110	4.472136	1.000000	2
5	6	6	5.099020	5.385165	2.828427	2
6	5	2	5.000000	1.414214	3.605551	1
7	5	7	4.472136	6.082763	4.242641	2
8	6	3	5.385165	2.828427	2.236068	2
9	4	4	3.162278	3.000000	4.000000	1

```
new_arr = []

dfx = df.groupby('result').mean()

for x in range(4):

    new_arr = []
```

```
    for i in range(dfx.shape[0]):
        new_arr.append((float(dfx['x'][i]),float(dfx['y'][i])))
    for j in range(k):
        df[j] = [dis(float(df['x'][k]),float(df['y'][k]),new_arr[j]) for k in
range(df.shape[0])]
    df["result"] = df[[0,1,2]].idxmin(axis = 1)
    dfx = df.groupby('result').mean()
print(new_arr)
```

DATA SET USED:

	x	y
0	2	4
1	2	6
2	5	6
3	4	7
4	8	3
5	6	6
6	5	2
7	5	7
8	6	3
9	4	4

OUTPUT:

In [111]:
```
new_arr = []
dfx = df.groupby('result').mean()
for x in range(4):
    new_arr = []
    for i in range(dfx.shape[0]):
        new_arr.append((float(dfx['x'][i]),float(dfx['y'][i])))
    for j in range(k):
        df[j] = [dis(float(df['x'][k]),float(df['y'][k]),new_arr[j]) for k in ra
    df["result"] = df[[0,1,2]].idxmin(axis = 1)
    dfx = df.groupby('result').mean()
print(new_arr)
```

```
[(2.0, 5.0), (5.75, 3.0), (5.0, 6.5)]
```

EXPERIMENT 5

Write A Program to implement Linear SVM.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets.samples_generator import make_blobs
#Create 40 separable points
x,y=make_blobs(n_samples=40,centers=2,random_state=20)
#fit SVM model
clf=svm.SVC(kernel='linear',C=1)
clf.fit(x,y)
#using to predict unknown data
new_data=[[3,4],[5,6]]
print(clf.predict(new_data))
#display the data in graph
plt.scatter(x[:,0],x[:,1],c=y,s=30,cmap=plt.cm.Paired)
plt.show()
clf=svm.SVC(kernel='linear',C=1) #classifier
clf.fit(x,y) #fitting the data
plt.scatter(x[:,0],x[:,1],c=y,s=30,cmap=plt.cm.Paired)
ax=plt.gca() #getting current axis
xlim=ax.get_xlim() #Get the x-axis range [left, right]
ylim=ax.get_ylim() #Get the y-axis range [left, right]
xx=np.linspace(xlim[0],xlim[1],30)
#xx is a ndarray with equally spaced intervals between the start and stop with 30
spaces between 2 intervals
yy=np.linspace(ylim[0],xlim[1],30)
YY,XX=np.meshgrid(yy,xx)
#returns two 2-Dimensional arrays representing the X and Y coordinates of all the
points.
xy=np.vstack([XX.ravel(),YY.ravel()]).T

Z=clf.decision_function(xy).reshape(XX.shape)
#EVALUATES DECISION FUNCTION AND RESHAPES THE MATRIX
ax.contour(XX,YY,Z,colors='k',levels=[-1,0,1],
 alpha=0.5,
```

```
 linestyles=['--', '-', '--'])
ax.scatter(clf.support_vectors_[:,0],
 clf.support_vectors_[:,1],s=100,
 linewidth=1,facecolors='none')
plt.show()
```