Lab1:

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

Program:

import pandas as pd

import numpy as np

data = pd.read\_csv('lab1.csv')

concepts=np.array(data)[:,:-1]

target=np.array(data)[:,-1]

def search(con,tar):

for i,val in enumerate(tar):

if val=="yes":

specifichyp=con[i].copy()

break

for i,val in enumerate(con):

if tar[i]=="yes":

for x in range(len(specifichyp)):

if val[x]!=specifichyp[x]:

specifichyp[x]="?"

else:

pass

return specifichyp

print(search(concepts, target))

Output:

['sunny', 'warm', '?', 'strong', '?', '?']

Lab2:

For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

Program:

import numpy as np

import pandas as pd

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

concepts=np.array(data)[0:,:-1]

target=np.array(data)[0:,-1]

def candidate\_elimination(con,tar):

s\_hyp=con[0].copy()

g\_hyp=[["?" for i in range(len(s\_hyp))] for i in range(len(s\_hyp))]

for i,val in enumerate(con):

if tar[i]=="yes":

for x in range(len(s\_hyp)):

if val[x]!=s\_hyp[x]:

s\_hyp[x]="?"

g\_hyp[x][x]="?"

if tar[i]=="no":

for x in range(len(s\_hyp)):

if val[x]!=s\_hyp[x]:

g\_hyp[x][x]=s\_hyp[x]

else:

g\_hyp[x][x]="?"

indices=[i for i,val in enumerate(g\_hyp) if val==["?","?","?","?","?","?"]]

for i in indices:

g\_hyp.remove(["?","?","?","?","?","?"])

return s\_hyp,g\_hyp

s\_final,g\_final=candidate\_elimination(concepts,target)

print(s\_final)

print(g\_final)

Output:

['sunny' 'warm' '?' 'strong' '?' '?']

[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]

Lab3:

Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Program:

import pandas as pd

import math

import numpy as np

data = pd.read\_csv("3-dataset.csv")

features = [feat for feat in data]

features.remove("answer")

class Node:

def \_\_init\_\_(self):

self.children = []

self.value = ""

self.isLeaf = False

self.pred = ""

def entropy(examples):

pos = 0.0

neg = 0.0

for \_, row in examples.iterrows():

if row["answer"] == "yes":

pos += 1

else:

neg += 1

if pos == 0.0 or neg == 0.0:

return 0.0

else:

p = pos / (pos + neg)

n = neg / (pos + neg)

return -(p \* math.log(p, 2) + n \* math.log(n, 2))

def info\_gain(examples, attr):

uniq = np.unique(examples[attr])

#print ("\n",uniq)

gain = entropy(examples)

#print ("\n",gain)

for u in uniq:

subdata = examples[examples[attr] == u]

#print ("\n",subdata)

sub\_e = entropy(subdata)

gain -= (float(len(subdata)) / float(len(examples))) \* sub\_e

#print ("\n",gain)

return gain

def ID3(examples, attrs):

root = Node()

max\_gain = 0

max\_feat = ""

for feature in attrs:

#print ("\n",examples)

gain = info\_gain(examples, feature)

if gain > max\_gain:

max\_gain = gain

max\_feat = feature

root.value = max\_feat

#print ("\nMax feature attr",max\_feat)

uniq = np.unique(examples[max\_feat])

#print ("\n",uniq)

for u in uniq:

#print ("\n",u)

subdata = examples[examples[max\_feat] == u]

#print ("\n",subdata)

if entropy(subdata) == 0.0:

newNode = Node()

newNode.isLeaf = True

newNode.value = u

newNode.pred = np.unique(subdata["answer"])

root.children.append(newNode)

else:

dummyNode = Node()

dummyNode.value = u

new\_attrs = attrs.copy()

new\_attrs.remove(max\_feat)

child = ID3(subdata, new\_attrs)

dummyNode.children.append(child)

root.children.append(dummyNode)

return root

def printTree(root: Node, depth=0):

for i in range(depth):

print("\t", end="")

print(root.value, end="")

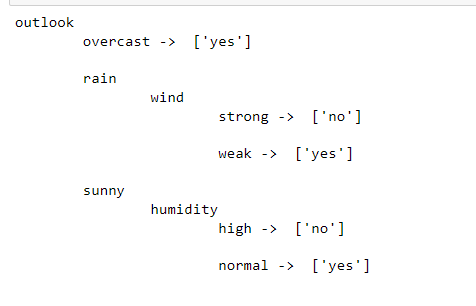
if root.isLeaf:

print(" -> ", root.pred)

print()

for child in root.children:

printTree(child, depth + 1)



Lab4:

Implement the Linear Regression algorithm in order to fit data points. Select

appropriate data set for your experiment and draw graphs.

Program:

import numpy as np

import matplotlib.pyplot as plt

def estimate\_coef(x, y):

# number of observations/points

n = np.size(x)

# mean of x and y vector

m\_x = np.mean(x)

m\_y = np.mean(y)

# calculating cross-deviation and deviation about x

SS\_xy = np.sum(y\*x) - n\*m\_y\*m\_x

SS\_xx = np.sum(x\*x) - n\*m\_x\*m\_x

# calculating regression coefficients

b\_1 = SS\_xy / SS\_xx

b\_0 = m\_y - b\_1\*m\_x

return (b\_0, b\_1)

def plot\_regression\_line(x, y, b):

# plotting the actual points as scatter plot

plt.scatter(x, y, color = "m",

marker = "o", s = 30)

# predicted response vector

y\_pred = b[0] + b[1]\*x

# plotting the regression line

plt.plot(x, y\_pred, color = "g")

# putting labels

plt.xlabel('x')

plt.ylabel('y')

# function to show plot

plt.show()

def main():

# observations / data

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12])

y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12,13,15,14])

# estimating coefficients

b = estimate\_coef(x, y)

print("Estimated coefficients:\nb\_0 = {} \

\nb\_1 = {}".format(b[0], b[1]))

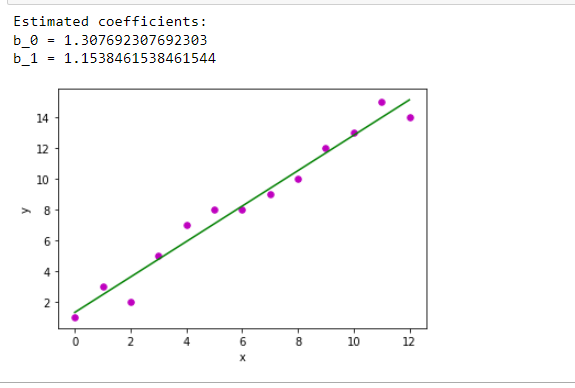
# plotting regression line

plot\_regression\_line(x, y, b)

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

main()

Output:



Lab5:

Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets

Program: