
MACHINE LEARNING LAB

Program 1:

```
import csv
with open('pgm1.csv','r') as f:
    reader=csv.reader(f)
    your_list=list(reader)
h=[['0','0','0','0','0','0']]
for i in your_list:
    print(i)
    if i[-1]=="Yes":
        j=0
        for x in i:
            if x!="Yes":
                if x!=h[0][j]and h[0][j]=='0':
                    h[0][j]=x
                elif x!=h[0][j]and h[0][j]!='0':
                    h[0][j]='?'
                else:
                    pass
            j=j+1
print("the most specific hypethesis is")
print(h)
```

-------------------------dataset pgm1.csv---------------------

```
'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
```

------------------------------output--------------------------

```
["'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 most specific hypethesis is
[["'Sunny'", "'Warm'", '?', "'Strong'", '?', '?']]
```

Program 2:

```
import numpy as np
import pandas as pd
data = pd.DataFrame(data=pd.read_csv('pgm2.csv'))
concepts = np.array(data.iloc[:,0:-1])
target = np.array(data.iloc[:,-1])
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_h))]
    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] = '?'
        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 ",i+1,"\n ")
        print(specific_h)
        print("general_h ", i+1, "\n ")
        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")
```

------------------------dataset pgm2.csv----------------------

```
'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
```

------------------------------output--------------------------

```
initialization of specific_h and general_h
["'Sunny'" "'Warm'" "'High'" "'Strong'" "'Warm'" "'Same'"]
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?']]
 steps of Candidate Elimination Algorithm 1
Specific_h  1

["'Sunny'" "'Warm'" "'High'" "'Strong'" "'Warm'" "'Same'"]
general_h  1
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?']]
 steps of Candidate Elimination Algorithm 2

Specific_h  2
["'Sunny'" "'Warm'" "'High'" "'Strong'" "'Warm'" "'Same'"]
general_h  2
[["'Sunny'", '?', '?', '?', '?', '?'], ['?', "'Warm'", '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?',
'?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', "'Same'"]]
 steps of Candidate Elimination Algorithm 3

Specific_h  3
["'Sunny'" "'Warm'" "'High'" "'Strong'" '?' '?']
general_h  3
[["'Sunny'", '?', '?', '?', '?', '?'], ['?', "'Warm'", '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?',
'?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?']]
```

```
Final Specific_h:
["'Sunny'" "'Warm'" "'High'" "'Strong'" '?' '?']
Final General_h:
[["'Sunny'", '?', '?', '?', '?', '?'], ['?', "'Warm'", '?',
'?', '?', '?']]
```

Program 3:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
from sklearn import tree
import pydotplus
iris=datasets.load_iris()
x=iris.data
y=iris.target
print(iris.target_names)
print(iris.feature_names)
clf=DecisionTreeClassifier(criterion="entropy")
model=clf.fit(x,y)
dot_data=tree.export_graphviz(clf,out_file=None,class_names=ir
is.target_names)
graph=pydotplus.graph_from_dot_data(dot_data)
graph.write_png("test.jpg")
```

-----------------------------**output**---------------------------

```
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
'petal width (cm)']
True
(view test.jpg)
```

Program 4:

```
import numpy as np
import matplotlib as m
X=np.array(([2,9],[1,5],[3,6]),dtype=float)
y=np.array(([92],[86],[89]),dtype=float)
X=X/np.amax(X,axis=0)
y=y/100
```

```
def sigmoid(x):
    return 1/(1+np.exp(-x))
def derivatives_sigmoid(x):
    return x*(1-x)
epoch=7000
lr=0.1
inputlayer_neurons=2
hiddenlayer_neurons=3
output_neurons=1
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neur
ons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neuron
s))
bout=np.random.uniform(size=(1,output_neurons))
for i in range(epoch):
    hinp1=np.dot(X,wh)
    hinp=hinp1+bh
    hlayer_act=sigmoid(hinp)
    outinp1=np.dot(hlayer_act,wout)
    outinp=outinp1+bout
    output=sigmoid(outinp)
    EO=y-output
    outgrad=derivatives_sigmoid(output)
    d_output=EO*outgrad
    EH=d_output.dot(wout.T)
    hiddengrad=derivatives_sigmoid(hlayer_act)
    d_hiddenlayer=EH*hiddengrad
    wout+=hlayer_act.T.dot(d_output)*lr
    wh+=X.T.dot(d_hiddenlayer)*lr
print("Input:\n"+str(X))
print("Actual Output:\n"+str(y))
print("Predicted Output:\n",output)
```
-----------------------------output---------------------------

```
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
```

```
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.89550443]
 [0.88002288]
 [0.89414655]]
```

Program 5:

```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
dataset=datasets.load_diabetes()
model=GaussianNB()
model.fit(dataset.data,dataset.target)
expected=dataset.target
predicted=model.predict(dataset.data)
print(metrics.confusion_matrix(expected,predicted))
print(metrics.accuracy_score(expected,predicted))
```

-----------------------------output---------------------------

```
[[1 0 0 ... 0 0 0]
 [0 1 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 1 0 0]
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 0 1]]
0.45248868778280543
```

Program 6:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
```

```
msg=pd.read_csv('six.csv',header=None,names=['message','label'
])
print("The dimensions of the dataset",msg.shape)
msg['labelnum']=msg.label.map({'pos':1,'neg':0})
x=msg.message
y=msg.labelnum
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=1)
count_vect=CountVectorizer()
xtrain_dtm=count_vect.fit_transform(xtrain)
xtest_dtm=count_vect.transform(xtest)

clf=MultinomialNB().fit(xtrain_dtm,ytrain)
predicted=clf.predict(xtest_dtm)

print("Accuracy metrics:")
print("Accuracy of the classifier
is:",metrics.accuracy_score(ytest,predicted))

print("Confusion matrix:")
print(metrics.confusion_matrix(ytest,predicted))

print("Recall and precision:")
print(metrics.recall_score(ytest,predicted))
print(metrics.precision_score(ytest,predicted))
```
-----------------------dataset six.csv------------------------

```
i like apple,pos
i hate pizza,neg
i like mango,pos
i like banana,pos
i hate radish,neg
i like flowers,pos
i hate burger,neg
i like listening to songs,pos
i do not like carrot,neg
i like watching tv,pos
```
--------------------------------output------------------------

```
The dimensions of the dataset (10, 2)
Accuracy metrics:
Accuracy of the classifier is: 1.0
Confusion matrix:
```

```
[[1 0]
 [0 2]]
Recall and precision:
1.0
1.0
```

--

Program 8:

```
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("p8.csv")
x1=data['x'].values
x2=data['y'].values
print(data)
X=np.matrix(list(zip(x1,x2)))
plt.scatter(x1,x2)
plt.show()
markers=['s','o','v']
k=3
clusters=KMeans(n_clusters=k).fit(X)
for i,l in enumerate(clusters.labels_):
    plt.plot(x1[i],x2[i],marker=markers[l])
```

---------------------------save as p8.csv---------------------

```
x,y
10,23
1,34
6,26
7,28
21,12
30,12
32,34
31,30
29,34
27,33
28,30
1,34
2,30
```

```
3,25
12,22
11,20
10,19
20,15
21,16
22,17
23,19
```

--

Program 9:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import
confusion_matrix,classification_report
from sklearn import datasets
iris=datasets.load_iris()
iris_data=iris.data
iris_labels=iris.target
print(iris_data)
x_train,X_test,Y_train,Y_test=train_test_split(iris_data,iris_
labels,test_size=0.20)
classifier=KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,Y_train)
y_prd=classifier.predict(X_test)
print(confusion_matrix(Y_test,y_prd))
print(classification_report(Y_test,y_prd))
```

------------------------------output--------------------------

```
[[5.9 4.  5.3 2.8]
.
.
.
[5.9 3.  5.1 1.8]]

[[ 9  0  0]
 [ 0  5  1]
 [ 0  1 14]]
             precision    recall  f1-score   support
          0       1.00      1.00      1.00         9
          1       0.83      0.83      0.83         6
          2       0.93      0.93      0.93        15
avg / total       0.93      0.93      0.93        30
```

Program 10:

```
from numpy import *
import operator
from os import listdir
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy.linalg
from scipy.stats.stats import pearsonr
def kernel(point,xmat,k):
    m,n=shape(xmat)
    weights=mat(eye((m)))
    for j in range(m):
        diff=point-X[j]
        weights[j,j]=exp(diff*diff.T/(-2.0*k**2))
        return weights
def localWeight(point,xmat,ymat,k):
    wei=kernel(point,xmat,k)
    W=((X.T*wei*X)).I*(X.T*(wei*ymat.T))
    return W
def localWeightRegression(xmat,ymat,k):
    m,n=shape(xmat)
    ypred=zeros(m)
    for i in range(m):
        ypred[i]=xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred
data=pd.read_csv('tips.csv')
bill=array(data.bill)
tip=array(data.tip)
mbill=mat(bill)
mtip=mat(tip)
m=shape(mbill)[1]
one=mat(ones(m))
X=hstack((one.T,mbill.T))
```

```
ypred=localWeightRegression(X,mtip,0.5)
SortIndex=X[:,1].argsort(0)
xsort=X[SortIndex][:,0]
fig=plt.figure()
ax=fig.add_subplot(1,1,1)
ax.scatter(bill,tip,color='green')
ax.plot(xsort[:,1],ypred[SortIndex],color='red',linewidth=5)
plt.xlabel('totalbil')
plt.ylabel('tip')
plt.show
```

----------------------------input tips.csv--------------------

```
bill,tip
300,30
400,20
5000,40
8000,50
```

☺ THANK YOU ☺

-By @2598