**Machine Learning Assignment 1**

**- Sundari Selvarajan**

**Classification Algorithm**

**Location:**

/afs/cad.njit.edu/courses/ccs/f15/cs/675/001/ss2738/Assignment1/Classification\_algo

**File Name:**

class\_algo.py

**Argument:**

class\_labels

**Program:**

\_\_author\_\_ = 'Sundu'

import sys

##### Read Labels #############################################

train\_file = sys.argv[1]

with open(train\_file) as f:

data = f.read()

labelset = {}

v = []

for line in data.split("\n"):

if line:

#print line

labelset[int(line.split()[0])]=line.split()[1:]

print (labelset)

####### Calculate TP , FP , FN and TN ##########################

tp = 0

fp = 0

fn = 0

tn = 0

for key , val in labelset.items():

if ((val[0]=='1') and (val[1]=='1')):

tp+=1

elif ((val[0]=='1') and (val[1]=='0')):

fp+=1

elif ((val[0]=='0') and (val[1]=='1')):

fn+=1

elif ((val[0]=='0') and (val[1]=='0')):

tn+=1

print("TP:",tp,"FP:",fp,"FN:",fn,"TN",tn)

####### Calculate E##############################################

e = 0.0

tot = 0.0

fsum = 0.0

tot = float(tp+fp+fn+tn)

#print tot

fsum = float(fp+fn)

#print fsum

e = fsum/tot

print ("No. of misclassified points=",e)

###### Calculate BER###############################################

ber = 0.0

sum1 = float(fp+tn)

sum2 = float(fn+tp)

f1 = (float)(fp)/sum1

f2 = (float)(fn)/sum2

ber = 0.5 \* (f1+f2)

print ("Balanced Error Rate=",ber)

###### Calculate Precision #######################################

prec = 0.0

sum3 = float(tp+fp)

prec = (float)(tp)/sum3

print ("Precision=",prec)

##### Calculate Recall#############################################

rec = 0.0

sum4 = (float)(tp+fn)

rec= (float)(tp)/sum4

print ("Recall=",rec)

**Nearest Means Algorithm**

**Location:**

/afs/cad.njit.edu/courses/ccs/f15/cs/675/001/ss2738/Assignment1/Nearest\_mean

**File Name:**

nm.py

**Program:**

\_\_author\_\_ = 'Sundu'

import copy

import sys

################### Reading Data and Label Sets###################################

input\_file = sys.argv[1]

with open(input\_file) as f:

data = f.read()

dataset = []

for line in data.split("\n"):

if line:

dataset.append([float(x) for x in line.split()])

train\_file = sys.argv[2]

with open(train\_file) as f:

data = f.read()

labelset = {}

for line in data.split("\n"):

if line:

if int(line.split()[0]) in labelset:

#print labelset

################### Calculating Means ############################################

means = {}

for k,v in labelset.items():

for data\_item in v:

if k in means:

t =list( zip(means[k],dataset[data\_item]))

for col in range(len(dataset[data\_item])):

means[k][col]=sum(t[col])

else:

means[k] = copy.copy(dataset[data\_item])

for k,v in means.items():

count = len(labelset[k])

means[k] = [x/count for x in means[k]]

print (means)

#################### Nearest Means Algorithm Implementation #######################

name=sys.argv[1]+".predictions"

op = open(name,"w")

for data in dataset:

distance={}

for k in means.keys():

t=zip(data,means[k])

#print "T=",t

distance[k]=0

for items in t:

distance[k]+=(items[0]-items[1])\*\*2

min\_dis = min(distance.values())

cls = None

for k,v in distance.items():

if min\_dis == v:

cls = k

break

op.write(str(cls)+"\n")

print (data,cls)

**Naïve Bayes Algorithm**

**Location:**

/afs/cad.njit.edu/courses/ccs/f15/cs/675/001/ss2738/Assignment1/Naive\_bayes

**File Name:**

nb.py

**Program:**

\_\_author\_\_ = 'Sundu'

import copy

import sys

################ Reading Data and Label Sets ######################################

input\_file = sys.argv[1]

with open(input\_file) as f:

data = f.read()

dataset = []

for line in data.split("\n"):

if line:

dataset.append([float(x) for x in line.split()])

train\_file = sys.argv[2]

with open(train\_file) as f:

data = f.read()

labelset = {}

for line in data.split("\n"):

if line:

if int(line.split()[0]) in labelset:

labelset[int(line.split()[0])].append(int(line.split()[1]))

else:

labelset[int(line.split()[0])] = [int(line.split()[1])]

#print("dataset:", dataset)

#print("labelset:", labelset)

################# Calculating Means ##############################################

means = {}

for k, v in labelset.items():

for data\_item in v:

if k in means:

t =list(zip(means[k], dataset[data\_item]))

for col in range(len(dataset[data\_item])):

means[k][col] = sum(t[col])

else:

means[k] = copy.copy(dataset[data\_item])

for k, v in means.items():

count = len(labelset[k])

means[k] = [x / count for x in means[k]]

################## Calculating Variance ###########################################

variance = [0.0]\*len(means)

print(variance)

for k,v in labelset.items():

print("v:",v)

#for y in v:

#print "y:",y

for x in v:

if k in means.keys():

#print x

#print dataset[x]

#print means[k]

t = list(zip(means[k],dataset[x]))

#print "t:",t

variance[k]+= ((t[0][0] - t[0][1])\*\*2)+((t[1][0] - t[1][1])\*\*2)

#print variance

#print len(v)

variance[k] /= len(v)

#print ("var:",variance)

################# Naive Bayes Algorithm Implementation ############################

name=sys.argv[1]+".predictions"

op = open(name,"w")

#means = {}

for data in dataset:

distance={}

for k in means.keys():

t=zip(data,means[k])

#print "T=",t

distance[k]=0

for items in t:

distance[k]+=(items[0]-items[1])\*\*2

distance[k]/=variance[k]

#print ("distance:",distance)

min\_dis = min(distance.values())

cls = None

for k,v in distance.items():

if min\_dis == v:

cls = k

break

op.write(str(cls)+"\n")

print (data,cls)