1.

def import\_data(filename):

X = []

y = []

file = open(filename, 'r') # open the file in read mode

# loop in through the file content

for line in file:

new\_line = line.split(',') # split entry by comma

y.append(new\_line[-1]) # get the class

# get the attributes

for x in new\_line[0:len(new\_line)-1]:

if x=="?":

X.append(float("NaN"))

else:

X.append(x)

# return the arrays

return X,y

print(import\_data('arrhythmia.data'))

2.

(a)

def impute\_missing(X):

median = ''

if len(X) % 2 == 0:

median = X[len(X)/2]

else:

median = X[len(X)//2 + 1]

for x in X:

if x == "NaN":

X.append(float(median))

return X

(b) because there could be outliers, which makes the median more suitable to represent an unknown value

(c)

3.

4.

def import\_data\_nonNum(filename):

X=[]

y=[]

file = open(filename, 'r')

lines = file.readlines()

temp = []

for l in lines:

new\_line = []

for i in l.split("\n"):

new\_line.append(i)

temp.append( list )

temp\_1 = []

for row in range( len(wholeList) ):

temp\_1.append( [temp[row][0]] )

row\_1 = 0

for num in temp\_1[row]:

temp\_2=[]

counter=1

for j in num.split(","):

if row\_1 == 0:

if counter == 2:

y.append( str(j) )

else:

temp\_2.append( str(j) )

if counter == 2:

y.append( float(j) )

else:

if counter == 5:

if j == 'male':

temp\_2.append(1)

if j == 'female':

temp\_2.append(0)

if counter == 12:

if j == 'C':

temp\_2.append(0)

if j == 'Q':

temp\_2.append(1)

if j == 'S':

temp\_2.append(2)

temp\_2.append(j)

counter += 1

X.append( temp\_2 )

row\_1 += 1

return X,y

5.

(a)

def train\_test\_split( X, y, t\_f ):

X\_train = []

y\_train = []

X\_test = []

y\_test = []

Ran\_list = random.sample(range(0, len(X)), int((len(X)-1)\*t\_f))

for i in range(len(X)):

if i == 0:

X\_train.append( X[0] )

y\_train.append( y[0] )

X\_test.append( X[0] )

y\_test.append( y[0] )

else:

dec= True

for Ran\_in in Ran\_list:

if i == Ran\_in:

y\_train.append( y[i] )

X\_train.append( X[i] )

dec= False

if dec:

X\_test.append( X[i] )

y\_test.append( y[i] )

return X\_train, y\_train, X\_test, y\_test

(b)

def train\_test\_CV\_split( X, y, t\_f, cv\_f ):

X\_train = []

y\_train = []

X\_test = []

y\_test = []

X\_cv = []

y\_cv = []

cv\_f\_ran = random.sample( range( 0, len(X) ), (len(X)-1)\*cv\_f )

t\_f\_ran = random.sample( range( 0, len(X) ), (len(X)-1)\*t\_f )

for i in range( len(X) ):

if i == 0:

X\_train.append( X[0] )

y\_train.append( y[0] )

X\_test.append( X[0] )

y\_test.append( y[0] )

X\_cv.append( X[0] )

y\_cv.append( y[0] )

else:

dec = True

for t\_f\_ran\_in in t\_f\_ran:

if i == t\_f\_ran\_in:

X\_train.append( X[i] )

y\_train.append( y[i] )

dec = False

for cv\_f\_ran\_in in cv\_f\_ran:

if i == cv\_f\_ran\_in:

X\_cv.append( X[i] )

y\_cv.append( y[i] )

dec = False

if dec:

X\_test.append(X[i])

y\_test.append(y[i])

return X\_train, y\_train, X\_test, y\_test, X\_cv, y\_cv