```
from rdkit import Chem
import pandas as pd

##Patrick Chirdon 2022
##This program takes a smarts for skin sensitizers and loops through a test set for
matches.  If a SMILES contains
##a substructure match for a known skin sensitization alert, the program grabs the
index of the compound from the test
#set, the corresponding label for positive or negative for sensitization, and the
associated SMILES strings.
##Each time the function skinsensitization is run in a loop, it removes the
molecules containing the SMARTS pattern
##from the previous run so that they are exluded from the count.  It prints out the
number of positives for skin
##sensitization and the number of hits that contain the SMARTS pattern.

def skinsensitization(smartspattern, list_of_num):
##the function accepts a smarts pattern and a list of numbers to be removed from
the list
##it accepts a list of smarts from Skinsensitization.csv and reads it to a
dataframe
    mydf=pd.read_csv('Skinsensitization-2.csv', sep='\t')
#next, it reads a testset into a pandas dataframe.  The test set contains skin
sensitizers labeled 1 and known non
#sensitizers labeled 0
    test=pd.read_csv('testset.csv', sep='\t')
    test2=test['SMILES']
    test3=test['label']


    mydflist=test2.values.tolist()
    molist=[Chem.MolFromSmiles(x) for x in mydflist]
#the testset of molecules is converted to a list

    testlist=test3.values.tolist()


    #molist is the list of test smiles

    j=0
    myindex=[]
    mylabel=[]
    mymols=[]
    p=0
#j is an index variable used for the labels of the sensitizers
#p is the index variable used for the molecules
##the index is a list of the indices that contain the SMARTs matches
#mylabel is the variable that will contain the label associated with the index
##mymol contains the list of the molecules that contain the SMARTS matches

#convert each molecule in molist to smiles
    for v in molist:
        myindex=[]
        molecule=molist[p]
        try:
            mymol=Chem.MolToSmiles(molecule)
        except:
            h=1
```

```
#convert each smartspattern to a molecule.  if a molecule has a substructure match,
append its index to myindex
#as well as its label to mylabel and the molecule associated to mymols
        j=0
        for i in mydf['SMARTS']:
            try:
                k=smartspattern
                k='c1([F,Cl,Br,I,$(N(=O)~O)])c([F,Cl,Br,I,$(N(=O)~O),$(C#N),$(C=O),
$(C(F)(F)F),$(S=O)])cc([F,Cl,Br,I,$(N(=O)~O),$(C#N),$(C=O),$(C(F)(F)F),$(S=O)])cc1'



            except:
                h=1
            try:

                n=Chem.MolFromSmarts(k)

            #n is the smarts
            except:
                h=1
            try:
                a=molecule.HasSubstructMatch(n)

                if(a==True):

                    myindex.append(j)
                    mylabel.append(p)
                    mymols.append(mymol)



            except:
                h=1

            j=j+1
        #print(p, mymol, myindex)

        p=p+1



    #print(mysum)
    #print('/')
    #print(num)
    #print(mymols)


    myindices=[]



     #print(mylabel)
    #from the list mylabel, take the list of indices of labels in teslist and grab
the associated label.  append
    #the labels to mylabels
    mylabels=[]
    for i in mylabel:

        mi=testlist[i]
        mylabels.append(mi)

    #list_of_num is the list of indices for compounds in the test set that need to
be removed.
    #compounds need to be removed from the list because they are matches from the
previous round
    #of smarts matching.  This is necessary because some compounds may contain
multiple smarts matches
    #this way, we have no overlap.
    for i in list_of_num:
        try:
            theindex=mylabel.index(i)
            mylabel.remove(i)
            myindices.append(theindex)
        except:
            h=1
     #remove the label and the index from mylabel and myindices
    myremoved=[]
    #create a list called myremoved that contains the list of molecules that were
removed.
    for i in myindices:
        try:
            mylabels.pop(i)
            mymols.pop(i)
            myremoved.append(mymols[i])
        except:
            h=1
   # print(mylabels)
    #we want to take the sum of the positive hits so that we can list positives /
total hits
    mysum=0
    for h in mylabels:
        mysum=mysum +h

    num=len(mylabels)
    #create a dataframe that contains the list of molecules and their associated
labels and save it to an excel file

    newdf=pd.DataFrame(zip(mymols, mylabels))
    newdf.to_excel('cnitrosocompounds.xls')








    #return the index of the labels, the labels, the molecules, the number of hits,
the number of positives and
    #the removed compounds from the test set






    return mylabel, mylabels, mymols, num, mysum, myremoved
#        try:
#           a=molecule.HasSubstructMatch(n)
#            if(a==True):
#                myindex.append(j)
#        except:
#            print('fail')

#        j=j+1
#       print(j)

#loop through the list of smarts and create a list called excludedlist that
contains the indices of smarts matches
mydf3=pd.read_csv('Skinsensitization-2.csv', sep='\t')
test3=mydf3['SMARTS']
smartslist=test3.values.tolist()
print(len(smartslist))
#smartslist=['[CH2]N([CH3])[CH3]']
excludedlist=[]
for i in smartslist:
    index, thelabel, molecules, hits, positives, removed=skinsensitization(i,
excludedlist)
    excludedlist=excludedlist + index
    print(positives)
    print('/')
    print(hits)
    print(index)
#[CH2][NH2]
##26/44
##[CH2]N([CH3])[CH3]
##22/62
##[CH2][NH2], [CH2]N([CH3])[CH3]
##26/44, 20/60
##overlap:
#[2248, 2255]
#CN(C)CCCN
#CN(C)CCCNCCCN
#mydf=pd.read_csv('Skinsensitization.csv', sep=",")
#next, it reads a testset into a pandas dataframe.  The test set contains skin
sensitizers labeled 1 and known non
#sensitizers labeled 0
#test=pd.read_csv('testset.csv')
#test2=test['SMILES']
#test3=test['label']


#mydflist=test2.values.tolist()
#molist=[Chem.MolFromSmiles(x) for x in mydflist]
#the testset of molecules is converted to a list


#print(mydflist[2248])
#print(mydflist[2255])
```