import dataset

```
import pandas as pd
```

```
import nltk
import re
import string
string.punctuation
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords

def tokenization(text):
    tokens = text.split(' ')
    return tokens

def remove_punctuation(text):
    punctuationfree=[]
    for t in text:
        punctuationfree.append(''.join(i for i in t if i not in string.punctuation))
    return punctuationfree


nltk.download('stopwords')
stopwords=stopwords.words('english')

def remove_stopwords(text):
    output= [i for i in text if i not in stopwords]
    return output


porter_stemmer = PorterStemmer()

def stemming(text):
    stem_text = [porter_stemmer.stem(word) for word in text]
    return stem_text
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
import pandas as pd

df = pd.read_excel(r'Data.xlsx')
```

```
df['Processed Text']=df['Summary Text'].str.replace('\d+', '')
df['Processed Text']= df['Processed Text'].apply(lambda x: x.lower())
df['Processed Text']=df['Processed Text'].apply(lambda x: tokenization(x))
df['Processed Text']=df['Processed Text'].apply(lambda x: remove_punctuation(x))
df['Processed Text']=df['Processed Text'].apply(lambda x: remove_stopwords(x))

df['Processed Text']=df['Processed Text'].apply(lambda x: ' '.join(x))
```

```
df.head(5)
```

```
df.shape
```

```
df.sample(10)
```

```
data=df
```

```

```

F

```
from sklearn.feature_extraction.text import CountVectorizer
c_vec = CountVectorizer(ngram_range=(2,2))
```

```
def f(n,features):
  sums = n.sum(axis = 0)
  data1 = []
  for col, term in enumerate(features):
      data1.append( (term, sums[0, col] ))
  ranking = pd.DataFrame(data1, columns = ['term', 'rank'])
  words = (ranking.sort_values('rank', ascending = False))
  print ("\n\nWords : \n", words.head(20))
  return list(words.term)[:20]
```

Code Execute

```
n1 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Code Execute')])
features = (c_vec.get_feature_names_out())
words=f(n1,features)
```

```
Words :
                          term   rank
70206         execute arbitrary  31309
49700                  cve cve  23111
13174            arbitrary code  21596
16936         attackers execute  21267
178874         remote attackers  21086
9374              allows remote  18786
35041                   code via  12089
34280            code execution   9643
37166              commands via   7865
54026            denial service   6907
24019           buffer overflow   6494
230610              via crafted   6342
27536              cause denial   6330
201722            sql injection   6325
13362             arbitrary sql   6226
201702             sql commands   6169
127535        memory corruption   5729
104042  injection vulnerability   5499
178889              remote code   5410
8948               allow remote   4633
```

```
indices=[0,2,3,4,5,6,7,8,10,16,17,18,19]
f1=[words[i] for i in indices]
print(f1)
```

```
['execute arbitrary', 'arbitrary code', 'attackers execute', 'remote attackers', 'allows remote', 'code via', 'code execution', 'co
```

Denial of Service

```
n2 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Denial of Service')])
features = (c_vec.get_feature_names_out())
words=f(n2,features)
```

```
Words :
                          term   rank
37525            denial service  28628
17624              cause denial  22672
129129         remote attackers  14371
5037              allows remote  13725
9989            attackers cause  13566
34264                  cve cve  11275
168063              via crafted   7992
49804         execute arbitrary   6556
7426             arbitrary code   6460
32853                 crash via   5322
94043         memory corruption   4533
140334           service memory   4147
6857          application crash   3966
10015         attackers execute   3552
23018               code cause   3430
14983           buffer overflow   3051
163477              users cause   2635
23343                  code via   2264
39683   different vulnerability   2210
139751            service crash   2173
```

```
indices=[0,1,2,3,7,8,9,10,11,12,13,14,15,19]
f2=[words[i] for i in indices]
print(f2)
```

```
['denial service', 'cause denial', 'remote attackers', 'allows remote', 'execute arbitrary', 'arbitrary code', 'crash via', 'memory
```

SQL Injectione

```
n3 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('SQL Injection')])
features = (c_vec.get_feature_names_out())
words=f(n3,features)
```

```
Words :
                           term   rank
46331              sql injection  10167
3789               arbitrary sql   6414
15856          execute arbitrary   6412
46296               sql commands   6239
9017                commands via   6061
23726    injection vulnerability   5933
41459           remote attackers   5880
4559           attackers execute   5659
2920               allows remote   4984
2799                allow remote   1566
23724  injection vulnerabilities   1556
29883               multiple sql   1507
21312               id parameter   1362
53341                     via id   1200
23719              injection via    843
14068             earlier allows    706
41460       remote authenticated    652
58276     vulnerability indexphp    599
54879            via unspecified    582
49965        unspecified vectors    568
```

```
indices=[0,1,3,5,7,10,11,14,17]
f3=[words[i] for i in indices]
print(f3)
```

```
['sql injection', 'arbitrary sql', 'sql commands', 'injection vulnerability', 'attackers execute', 'injection vulnerabilities', 'mu
```

gain priviledge

```
n4 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('gain priviledge')])
features = (c_vec.get_feature_names_out())
words=f(n4,features)
```

```
Words :
                       term  rank
17774      gain privileges  3973
24551          local users  2968
45898           users gain  2814
34065       privileges via  2788
2241          allows local  2628
41091           sp windows  1160
47353          via crafted   997
4011        attackers gain   784
49681       windows server   778
2261         allows remote   733
9723   crafted application   677
41075                sp sp   623
43979         trojan horse   619
36355     remote attackers   577
47654           via trojan   553
38470          search path   536
39201            server sp   521
2130           allow local   511
37421      root privileges   510
11437       denial service   485
```

```
indices=[0,2,3,7,18]
f4=[words[i] for i in indices]
print(f4)
```

```
['gain privileges', 'users gain', 'privileges via', 'attackers gain', 'root privileges']
```

Http Response Splitting

```
n5 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Http Response Splitting')])
features = (c_vec.get_feature_names_out())
words=f(n5,features)
```

```
Words :
                          term  rank
711             http response   148
1261       response splitting   148
1450        splitting attacks   107
706              http headers   106
324              conduct http    97
100            arbitrary http    97
778          inject arbitrary    97
393            crlf injection    95
68              allows remote    94
1227         remote attackers    94
163               attacks via    93
673           headers conduct    90
793   injection vulnerability    86
141          attackers inject    85
1711          via unspecified    33
1560      unspecified vectors    30
396       crosssite scripting    23
1682              via crafted    22
1824          vulnerable http    16
212           cache poisoning    14
```

```
indices=[0,1,2,3,4,5,7,11,18,19]
f5=[words[i] for i in indices]
print(f5)
```

```
['http response', 'response splitting', 'splitting attacks', 'http headers', 'conduct http', 'arbitrary http', 'crlf injection', 'h
```

Memory Corruption

```
n6 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Memory Corruption')])
features = (c_vec.get_feature_names_out())
words=f(n6,features)
```

```
Words :
                            term   rank
5512                   cve cve  13988
13907         memory corruption   7441
1274             arbitrary code   4617
7692          execute arbitrary   4046
5989             denial service   3939
2765               cause denial   3844
20404            service memory   3366
1668          attackers execute   3245
18977          remote attackers   2957
3493                 code cause   2594
870               allows remote   2558
24092               via crafted   2462
4721             corruption via   2076
4725   corruption vulnerability   1994
6280    different vulnerability   1860
24561         vulnerability cve   1721
11422         internet explorer   1445
5224                crafted web   1339
25003                  web site   1216
1166          application crash   1149
```

```
indices=[1,2,3,4,5,6,7,8,9,10,12,13,17,19]
f6=[words[i] for i in indices]
print(f6)
```

```
['memory corruption', 'arbitrary code', 'execute arbitrary', 'denial service', 'cause denial', 'service memory', 'attackers execute
```

Gain Information

```
n7 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Gain Information')])
features = (c_vec.get_feature_names_out())
words=f(n7,features)
```

```
Words :
                          term  rank
82070     sensitive information  7152
62433          obtain sensitive  6401
76052          remote attackers  3799
5370              allows remote  3750
46294           information via  3618
8804            attackers obtain  3274
98993                via crafted  2619
45630    information disclosure  1778
55789  maninthemiddle attackers  1585
5345      allows maninthemiddle  1558
8838             attackers spoof  1522
83081             servers obtain  1521
22168       crafted certificate  1498
86760              spoof servers  1490
83041            servers allows  1489
98158       verify certificates  1483
87058               ssl servers  1475
14402          certificates ssl  1471
6825        application android  1368
5927              android verify  1350
```

```
indices=[0,1,4,5,7,8,9,10,11,12,13,16,17,18,19]
f7=[words[i] for i in indices]
print(f7)
```

```
['sensitive information', 'obtain sensitive', 'information via', 'attackers obtain', 'information disclosure', 'maninthemiddle atta
```

Cross Site Scripting (XSS)

```
n8 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type']=='Cross Site Scripting (XSS)'])
features = (c_vec.get_feature_names_out())
words=f(n8,features)
```

```
Words :
                          term   rank
25754       crosssite scripting  15841
99411             scripting xss  13546
54890          inject arbitrary  10576
8595              arbitrary web  10339
135186               web script  10082
93809          remote attackers   9985
98988               script html   9920
50860                  html via   9805
140230        xss vulnerability   9268
10307          attackers inject   9125
6343              allows remote   8184
6039               allow remote   3157
140228      xss vulnerabilities   3058
70449        multiple crosssite   2959
125587          via unspecified   2079
115953      unspecified vectors   1875
99402   scripting vulnerability   1744
140209                  xss via   1571
93811      remote authenticated   1355
25708                cross site   1314
```

```
indices=[0,2,3,4,5,6,7,8,10,16,17,18,19]
f8=[words[i] for i in indices]
print(f8)
```

```
['crosssite scripting', 'inject arbitrary', 'arbitrary web', 'web script', 'remote attackers', 'script html', 'html via', 'xss vuln
```

CSRF

```
n9 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('CSRF')])
features = (c_vec.get_feature_names_out())
words=f(n9,features)
```

```
Words :
                               term  rank
22351               request forgery  2361
7197              crosssite request  2196
11701                  forgery csrf  1948
22042              remote attackers  1347
7581             csrf vulnerability  1318
12955         hijack authentication  1106
2057                  allows remote  1052
3263               attackers hijack  1023
3535   authentication administrators   534
1321        administrators requests   434
6816                    could allow   402
1923                   allow remote   402
7580           csrf vulnerabilities   377
17505            multiple crosssite   361
31979              wordpress plugin   344
20203              plugin wordpress   325
2752                arbitrary users   286
28343                users requests   261
3538       authentication arbitrary   261
7313                    csrf attack   248
```

```
indices=[0,1,2,4,5,6,7,8,9,12,13,19]
f9=[words[i] for i in indices]
print(f9)
```

```
['request forgery', 'crosssite request', 'forgery csrf', 'csrf vulnerability', 'hijack authentication', 'allows remote', 'attackers
```

Security Vulnerabilities

```
n10 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Security Vulnerabilities')])
features = (c_vec.get_feature_names_out())
words=f(n10,features)
```

```
Words :
                          term  rank
17953        remote attackers  3094
1311            allows remote  2546
7207         execute arbitrary  1865
1896         attackers execute  1496
3994              commands via  1092
20223             sql commands  1040
1627            arbitrary sql  1040
20227           sql injection  1034
10271  injection vulnerability   836
5139       crosssite scripting   698
18863            scripting xss   697
1303              allow remote   693
10261         inject arbitrary   688
1635             arbitrary web   685
25892               web script   675
18840             script html   675
9400                 html via   674
1903          attackers inject   641
3721                 code via   510
1584           arbitrary code   477
```

```
indices=[i for i in range(20)]
f10=[words[i] for i in indices]
print(f10)
```

```
['remote attackers', 'allows remote', 'execute arbitrary', 'attackers execute', 'commands via', 'sql commands', 'arbitrary sql', 's
```

Directory Traversal

```
n11 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Directory Traversal')])
features = (c_vec.get_feature_names_out())
words=f(n11,features)
```

```
Words :
                          term  rank
10772        directory traversal  4258
40527    traversal vulnerability  3138
1901               allows remote  2739
32987           remote attackers  2674
15870                  files via  2544
2804             arbitrary files  2478
11560                    dot dot  1882
43732                    via dot  1666
32395             read arbitrary  1476
3686              attackers read  1335
29221             path traversal   923
13838          execute arbitrary   772
23111                local files   578
2825             arbitrary local   569
3662           attackers include   551
19549            include execute   520
1712                allow remote   463
12421              earlier allows   427
32989       remote authenticated   398
40526  traversal vulnerabilities   372
```

```
indices=[0,1,4,5,8,9,10,12,13,14,19]
f11=[words[i] for i in indices]
print(f11)
```

```
['directory traversal', 'traversal vulnerability', 'files via', 'arbitrary files', 'read arbitrary', 'attackers read', 'path traver
```

Bypass

```
n12 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Bypass')])
features = (c_vec.get_feature_names_out())
words=f(n12,features)
```

```
Words :
                        term  rank
7371        attackers bypass  3518
4000           allows remote  3196
65953       remote attackers  3150
11231        bypass intended  1598
10897  bypass authentication  1243
21524                cve cve  1134
638      access restrictions  1020
84033           users bypass   929
86867             via crafted   885
41302        intended access   865
11785   bypass vulnerability   728
19987             could allow   695
87417        via unspecified   676
7122         attacker bypass   653
68132       restrictions via   643
81926    unspecified vectors   641
65955   remote authenticated   567
46520             local users   558
63212   protection mechanism   532
7980   authentication bypass   531
```

```
indices=[0,3,4,6,7,10,13,14,18,19]
f12=[words[i] for i in indices]
print(f12)
```

```
['attackers bypass', 'bypass intended', 'bypass authentication', 'access restrictions', 'users bypass', 'bypass vulnerability', 'at
```

File Inclusion

```
n13 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('File Inclusion')])
features = (c_vec.get_feature_names_out())
words=f(n13,features)
```

```
Words :
                           term  rank
4656              file inclusion  2098
11599           remote attackers  1900
11606                remote file  1882
4388           execute arbitrary  1878
10442                 php remote  1839
1237           attackers execute  1817
10388                  php code  1800
1058               arbitrary php  1799
2379                    code via  1782
15490                    via url  1647
808                allows remote  1380
6398     inclusion vulnerability  1368
781                 allow remote   549
6397   inclusion vulnerabilities   538
8639                multiple php   534
3988              earlier allows   310
9897              parameter note   209
11532    registerglobals enabled   208
13575                third party   172
7621                  local file   161
```

```
indices=[0,2,4,6,7,8,10,11,13,14,16,17,19]
f13=[words[i] for i in indices]
print(f13)
```

```
['file inclusion', 'remote file', 'php remote', 'php code', 'arbitrary php', 'code via', 'allows remote', 'inclusion vulnerability'
```

Overflow

```
n14 = c_vec.fit_transform(data['Processed Text'][data['Vulnerability Type'].str.contains('Overflow')])
features = (c_vec.get_feature_names_out())
words=f(n14,features)
```

```
Words :
                       term   rank
25095              cve cve  13150
11289     buffer overflow  11651
5440       arbitrary code  10583
35771   execute arbitrary  10066
99027    remote attackers   8938
27498      denial service   8461
13451        cause denial   7783
3719         allows remote   7729
7585    attackers execute   6955
17424            code via   5769
125900        via crafted   5746
68773   memory corruption   4589
7556      attackers cause   3972
103788              sd sd   2852
111644  stackbased buffer   2698
106301     service memory   2674
17081           code cause   2582
55639    integer overflow   2571
126321           via long   2542
49586    heapbased buffer   2532
```

```
indices=[1,2,3,5,6,8,10,11,12,14,15,17,18,19]
f14=[words[i] for i in indices]
print(f14)
```

```
['buffer overflow', 'arbitrary code', 'execute arbitrary', 'denial service', 'cause denial', 'attackers execute', 'via crafted', 'm
```

All_F

```
feat=[]
for i in range(1,15):
```

```
  feat+=globals()["f"+str(i)]
print(feat)
```

```
['execute arbitrary', 'arbitrary code', 'attackers execute', 'remote attackers', 'allows remote', 'code via', 'code execution', 'co
```

```
print(len(feat))
print(len(set(feat)))
```

```
173
122
```

```
feat=list(set(feat))
print(feat)
```

```
['bypass vulnerability', 'code via', 'local files', 'remote file', 'parameter note', 'scripting xss', 'maninthemiddle attackers', '
```

```
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
big = CountVectorizer(ngram_range=(1,2))
big.fit(feat)
#ngrams = c_vec.fit_transform(data['Processed Text'])
```

```
CountVectorizer
CountVectorizer(ngram_range=(1, 2))
```

```
!cat /proc/meminfo
```

```
MemTotal:       13297192 kB
MemFree:        10563204 kB
MemAvailable:   11111640 kB
Buffers:           82972 kB
Cached:           686896 kB
SwapCached:            0 kB
Active:           453692 kB
Inactive:        2089992 kB
Active(anon):       3676 kB
Inactive(anon):  1787448 kB
Active(file):     450016 kB
Inactive(file):   302544 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:              2080 kB
Writeback:             0 kB
AnonPages:       1773876 kB
Mapped:           231764 kB
Shmem:             21732 kB
KReclaimable:      74908 kB
Slab:             106268 kB
SReclaimable:      74908 kB
SUnreclaim:        31360 kB
KernelStack:        4080 kB
PageTables:        27032 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     6648596 kB
Committed_AS:    3245168 kB
VmallocTotal:   34359738367 kB
VmallocUsed:        8948 kB
VmallocChunk:          0 kB
Percpu:             1336 kB
HardwareCorrupted:     0 kB
AnonHugePages:     28672 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:       82752 kB
```

```
DirectMap2M:     3059712 kB
DirectMap1G:    12582912 kB
```

```
X = big.fit_transform(data['Processed Text'][:in0t(0.7*len(data))]).toarray()
```

```
y=df.iloc[:, 1]
```

Feature Importances

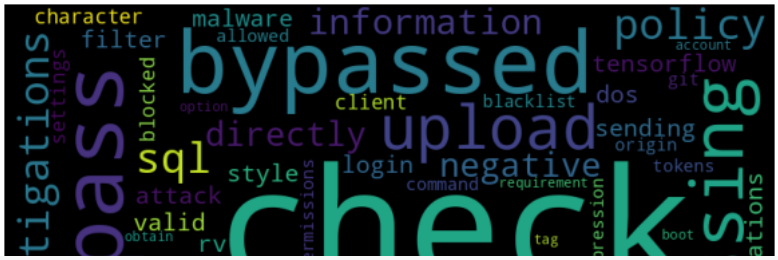
```
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
v = CountVectorizer(ngram_range=(2,2))
x = v.fit_transform(data['Processed Text'])
model=LogisticRegression()
#forest = RandomForestClassifier(n_estimators = 300)

model.fit(x, data['Vulnerability Type'])
```

```
points=feat
```

```
comment_words = ''
stopwords = set(STOPWORDS)

# iterate through the csv file
for val in points:

    # typecaste each val to string
    val = str(val)
    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    comment_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)
```

```
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
df_sorted= feature_importance.sort_values('feature_importance',ascending=False)[:20]
```

```
plt.figure(figsize=(25,8))
# bar plot with matplotlib
plt.bar('feature', 'feature_importance',data=df_sorted)
plt.xlabel("features", size=15)
plt.ylabel("importances of features", size=15)
plt.title("Bar plot in Descending Order", size=18)
#plt.savefig("bar_plot_matplotlib_descending_order_Python.png")
```

```
Text(0.5, 1.0, 'Bar plot in Descending Order')
```