Sorel Data Preprocessing To get the Features

In [1]:
```
import tqdm
import lmdb
import json
import numpy as np
import pandas as pd
import sqlite3
import os
import msgpack
import zlib
import tqdm
import pandas as pd
import seaborn as sns
from tabulate import tabulate
```

Make sure 'data.mdb' and 'lock.mdb' files present in sorel_lmdb

Refer this https://github.com/sophos-ai/SOREL-20M to Download these files

'meta.db' 3.5GB file will be present in s3://sorel-20m/09-DEC-2020/processed-data

'data.mdb' and 'lock.mdb' will be present in s3://sorel-20m/09-DEC-2020/processed-data/ember_features with size approx ~72GB

In [2]:
```
sorel_dir = '../Dataset/sorel'
sorel_lmdb = '../Dataset/sorel/db'
sorel_db = '../Dataset/sorel/meta.db'
!ls '../Dataset/sorel/db'
```

```
data.mdb  lock.mdb
```

Reading malware_information from db and saving it to 'sorel_malware.csv'

If 'sorel_malware.csv' not exist run this below code

In [4]:
```
# con = sqlite3.connect(sorel_db)
# df = pd.read_sql_query("SELECT * from meta where is_malware=1", con)
# df.to_csv(os.path.join(sorel_dir,'sorel_malware.csv'))
```

Load the Data from 'sorel_malware.csv'

In [3]:
```
df = pd.read_csv(os.path.join(sorel_dir,'sorel_malware.csv'))
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Unnamed: 0', 'sha256', 'is_malware', 'rl_fs_t',
       'rl_ls_const_positives', 'adware', 'flooder', 'ransomware', 'dropper',
       'spyware', 'packed', 'crypto_miner', 'file_infector', 'installer',
       'worm', 'downloader'],
      dtype='object')
```

The current datset contains 11 different classes of malware

In [5]:
```
labels = ['adware', 'flooder', 'ransomware', 'dropper',
       'spyware', 'packed', 'crypto_miner', 'file_infector', 'installer',
       'worm', 'downloader']
```

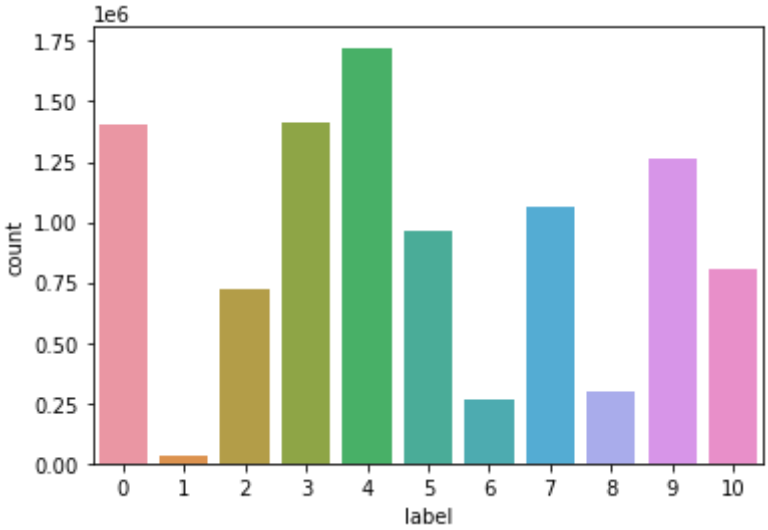
Added two extra columns sum and label. Divided the counts of each class with sum to get the probability of each class and label represents the class with max probability

In [6]:
```
df['sum'] =df.iloc[:,5:16 ].sum(axis=1)
```

```
for label in labels:
    df[label]= df[label]/df['sum']
```

In [7]:
```
df['label']= np.argmax(df.values[:,5:16],axis=1)
```

In [8]:
```
data = pd.DataFrame(data=df['label'], columns=["label"])
sns.countplot(x = 'label', data=data);
```



In [9]:
```
values,counts = np.unique(df['label'].values,return_counts=True)
table = []
for i,j in zip(labels,counts):
    table.append([i,str(j)])
print (tabulate(table,headers=['label', 'sample_count']))
```

```
label            sample_count
-------------  --------------
adware                1404601
flooder                 27896
ransomware             721508
dropper               1414233
spyware               1722713
packed                 964299
crypto_miner           268673
file_infector         1063928
installer              300247
worm                  1265735
downloader             808987
```

selected samples with detection sum > 15 and 7000 samples from each class

In [10]:
```
df2 = df[df['sum']>14].groupby('label').head(10000).reset_index(drop=True)
```

In [11]:
```
df2.iloc[:,5:18][:10]
```

Out[11]:

	adware	flooder	ransomware	dropper	spyware	packed	crypto_miner	file_infector	installer
0	0.000000	0.0	0.0	0.0	0.5	0.0	0.0	0.5	0.000000
1	0.388889	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.611111
2	0.388889	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.611111
3	0.388889	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.611111
4	0.368421	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.631579
5	0.533333	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.466667

	adware	flooder	ransomware	dropper	spyware	packed	crypto_miner	file_infector	installer
6	0.578947	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.421053
7	0.437500	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.562500
8	0.400000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.533333
9	0.368421	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.631579

In [12]:
```
df2.shape
```

Out[12]: (109481, 18)

Saving the selected samples with prob.scores and label to sorel_label.csv for future use

In [18]:
```
df2.to_csv(os.path.join(sorel_dir,'sorel_label.csv'))
```

Load sample data from sorel_label.csv

In [19]:
```
df2 = pd.read_csv(os.path.join(sorel_dir,'sorel_label.csv')).iloc[:,1:]
```

In [13]:
```
df2.shape
```

Out[13]: (109481, 18)

Get featutes Corressponds to each sample from lmdb database using sha256 Key

and save feature in data array and coresponding labels in y array

Some entry feature may not present in the db, igonre these samples

In [ ]:
```
env = lmdb.Environment(sorel_lmdb, readonly=True, map_size=1e13, max_readers=1024)

x =None
D = None
count = 0
with env.begin(write=False) as txn:
    for d in tqdm.tqdm(df2):
        x = txn.get(d[1].encode('ascii'))
        if x:
            count = count+1
            x = msgpack.loads(zlib.decompress(x),strict_map_key=False)
            data = np.append(data,np.array([x[0]]),axis=0)
            y = np.append(y,np.array([d[5:]]),axis=0)


        if count%1000==0:
            np.savez(os.path.join(sorel_dir,'sorel_data.npz'), data, y)
np.savez(os.path.join(sorel_dir,'sorel_data.npz'), data, y)
```

83%|████████▎ | 13229/16000 [3:16:04<51:54, 1.12s/it]

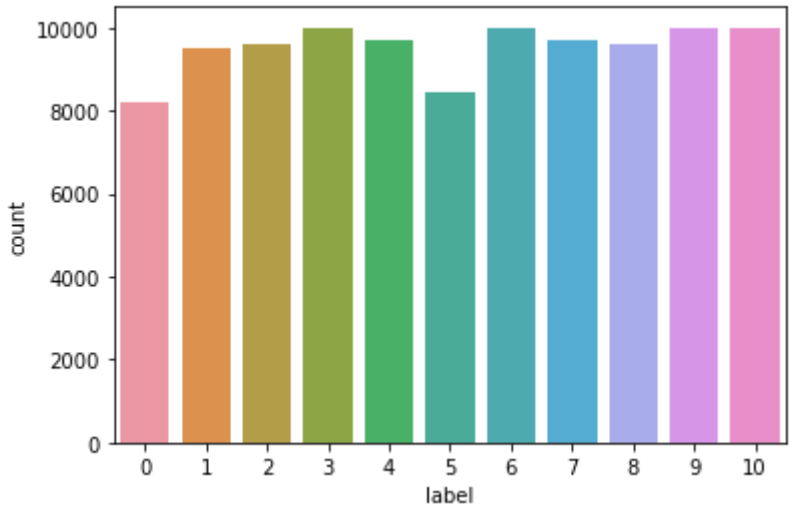
It is taking almost a day to finish the task

samples saved in 'sorel_data.npz' for future use

In [16]:
```
npzfile = np.load(os.path.join(sorel_dir,'sorel_data.npz'),allow_pickle=True)
data,y = npzfile['arr_0'],npzfile['arr_1']
data.shape
```

Out[16]: (104746, 2381)

In [18]:
```
data_labels = pd.DataFrame(data=y[:,-1], columns=["label"])
sns.countplot(x = 'label', data=data_labels);
```



In [23]:
```
values,counts = np.unique(y[:,-1],return_counts=True)
table = []
for i,j in zip(labels,counts):
    table.append([i,str(j)])
print (tabulate(table,headers=['label', 'sample_count']))
```

```
label            sample_count
-------------  --------------
adware                   8211
flooder                  9481
ransomware               9607
dropper                  9997
spyware                  9717
packed                   8465
crypto_miner             9988
file_infector            9709
installer                9595
worm                    10000
downloader               9976
```

In [19]:
```
npzfile = np.load(os.path.join(sorel_dir,'sorel_data.npz'),allow_pickle=True)
data,y = npzfile['arr_0'],npzfile['arr_1']
```

In [20]:
```
npzfile['arr_1'].shape
```

Out[20]: (104746, 13)

In [21]:
```
data,y = npzfile['arr_0'],npzfile['arr_1']
```

In [22]:
```
data.shape
```

Out[22]: (104746, 2381)