In [ ]:

```
##MALWARE CLASSIFICATION USING NEURAL NETWORK--MULTI-LAYER PRECEPTRON
```

In [1]:

```
#Pandas is used for data processing
#Seaborn is used for data visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
data=pd.read_csv("/content/Malware dataset.csv.zip")
```

In [3]:

```
#1.Data processing-
#1.1 Analyse the features of data.
data.head()
```

Out[3]:

	hash	millisecond	classification	state	usage_counter	prio	static_pr
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	malware	0	0	3069378560	142
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	malware	0	0	3069378560	142
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	malware	0	0	3069378560	142
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	malware	0	0	3069378560	142
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	malware	0	0	3069378560	142

5 rows × 35 columns

In [4]:

```
#The first one is the number of rows and
# the other one is the number of columns.
data.shape
```

Out[4]:

```
(100000, 35)
```

In [5]:

```
#1.2  Drop unused columns
# returns the number of missing values in the data set.
data.isnull().sum()
```

Out[5]:

```
hash                     0
millisecond              0
classification           0
state                    0
usage_counter            0
prio                     0
static_prio              0
normal_prio              0
policy                   0
vm_pgoff                 0
vm_truncate_count        0
task_size                0
```

```
cached_hole_size      0
free_area_cache       0
mm_users              0
map_count             0
hiwater_rss           0
total_vm              0
shared_vm             0
exec_vm               0
reserved_vm           0
nr_ptes               0
end_data              0
last_interval         0
nvcsw                 0
nivcsw                0
min_flt               0
maj_flt               0
fs_excl_counter       0
lock                  0
utime                 0
stime                 0
gtime                 0
cgtime                0
signal_nvcsw          0
dtype: int64
```

In [ ]:

```

```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['hash', 'millisecond', 'classification', 'state', 'usage_counter',
       'prio', 'static_prio', 'normal_prio', 'policy', 'vm_pgoff',
       'vm_truncate_count', 'task_size', 'cached_hole_size', 'free_area_cache',
       'mm_users', 'map_count', 'hiwater_rss', 'total_vm', 'shared_vm',
       'exec_vm', 'reserved_vm', 'nr_ptes', 'end_data', 'last_interval',
       'nvcsw', 'nivcsw', 'min_flt', 'maj_flt', 'fs_excl_counter', 'lock',
       'utime', 'stime', 'gtime', 'cgtime', 'signal_nvcsw'],
      dtype='object')
```

In [7]:

```
# Drop the rows where all of the elements are nan
data1=data.dropna(how="any",axis=0)
data1.head()
```

Out[7]:

	hash	millisecond	classification	state	usage_counter	prio	static_pr
0	**42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...**	**0**	**malware**	**0**	**0**	**3069378560**	**142**
1	**42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...**	**1**	**malware**	**0**	**0**	**3069378560**	**142**
2	**42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...**	**2**	**malware**	**0**	**0**	**3069378560**	**142**
3	**42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...**	**3**	**malware**	**0**	**0**	**3069378560**	**142**
4	**42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...**	**4**	**malware**	**0**	**0**	**3069378560**	**142**

5 rows × 35 columns

In [8]:

```
##convert strings  to integers (0, 1) using map()
data1['classification'] = data1.classification.map({'benign':0, 'malware':1})
```

In [9]:

```
#In this dataset we will work on the classification column, it will count number of times
a particular class has occurred.
data1["classification"].value_counts()
```

Out[9]:

```
1    50000
0    50000
Name: classification, dtype: int64
```

In [10]:

```
data1.head()
```

Out[10]:

	hash	millisecond	classification	state	usage_counter	prio	static_pr
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	1	0	0	3069378560	142
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	1	0	0	3069378560	142
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	1	0	0	3069378560	142
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	1	0	0	3069378560	142
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	1	0	0	3069378560	142

5 rows × 35 columns

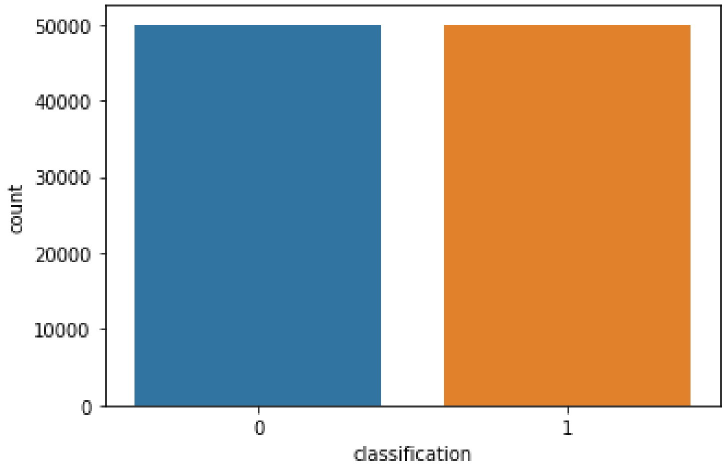
In [ ]:

```
#1.3 plot: number of benign[0] and malware[1] in the dataset.
```

In [11]:

```
sns.countplot(data1["classification"])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argu
ment will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  FutureWarning
```



In [12]:

```
benign1=data.loc[data['classification']=='benign']
benign1["classification"].head()
```

Out[12]:

```
1000    benign
1001    benign
1002    benign
1003    benign
```

```
1003    benign
1004    benign
Name: classification, dtype: object
```

In [ ]:

```

```

In [13]:

```
malware1=data.loc[data['classification']=='malware']
malware1["classification"].head()
```

Out[13]:

```
0    malware
1    malware
2    malware
3    malware
4    malware
Name: classification, dtype: object
```

In [14]:

```
# Define features and labels for model
x=data1.drop(["hash","classification",'vm_truncate_count','shared_vm','exec_vm','nvcsw',
'maj_flt','utime'],axis=1)
x.head()
```

Out[14]:

	millisecond	state	usage_counter	prio	static_prio	normal_prio	policy	vm_pgoff	task_size	cached_hole_size	...
0	0	0	0	3069378560	14274	0	0	0	0	0	...
1	1	0	0	3069378560	14274	0	0	0	0	0	...
2	2	0	0	3069378560	14274	0	0	0	0	0	...
3	3	0	0	3069378560	14274	0	0	0	0	0	...
4	4	0	0	3069378560	14274	0	0	0	0	0	...

5 rows × 27 columns

In [15]:

```
y=data1["classification"]
y
```

Out[15]:

```
0        1
1        1
2        1
3        1
4        1
        ..
99995    1
99996    1
99997    1
99998    1
99999    1
Name: classification, Length: 100000, dtype: int64
```

In [16]:

```
#scikit-learn is a library for machine learning algorithms
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
```

In [17]:

In [17]:

```
#  Split dataset into training (70%) and test (30%) set
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
```

In [18]:

```
model = MLPClassifier()
model.fit(x_train,y_train)
```

Out[18]:

```
MLPClassifier()
```

In [19]:

```
pred=model.predict(x_test)
pred
```

Out[19]:

```
array([0, 0, 0, ..., 0, 0, 0])
```

In [20]:

```
model.score(x_test,y_test)
```

Out[20]:

```
0.5016666666666667
```

In [21]:

```
result=pd.DataFrame({
    "Actual_Value":y_test,
    "Predict_Value":pred
})
result
```

Out[21]:

	Actual_Value	**Predict_Value**
43660	0	0
87278	1	0
14317	0	0
81932	1	0
95321	1	0
...	...	...
994	1	0
42287	0	0
4967	0	0
47725	0	0
42348	0	0

30000 rows × 2 columns

In [ ]: