In [ ]:

```
##MALWARE CLASSIFICATION USING NAIVE BAYES CLASSIFIER
```

In [ ]:

```
#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 [ ]:

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

In [ ]:

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

In [ ]:

```
data.head()
```

Out[ ]:

	hash	millisecond	classification	state	usage
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	malware	0	
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	malware	0	
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	malware	0	
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	malware	0	
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	malware	0	

In [ ]:

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

Out[ ]:

```
(100000, 35)
```

In [ ]:

```
#1.2  Drop unused columns
```

In [ ]:

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

Out[ ]:

```
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 [ ]:

```
data.columns
```

Out[ ]:

```
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_ca
che',
       '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 [ ]:

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

Out[ ]:

	hash	millisecond	classification	state	usage
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	malware	0	
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	malware	0	
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	malware	0	
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	malware	0	
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	malware	0	

In [ ]:

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

In [ ]:

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

Out[ ]:

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

In [ ]:

```
data1.head()
```

Out[ ]:

	hash	millisecond	classification	state	usage
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	1	0	
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	1	0	
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	1	0	
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	1	0	
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	1	0	

In [ ]:

```
data1.tail()
```

Out[ ]:

	hash	millisecond	classification	state
99995	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	995	1	4096
99996	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	996	1	4096
99997	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	997	1	4096
99998	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	998	1	4096
99999	025c63d266e05d9e3bd57dd9ebd0abe904616f569fe4e2...	999	1	4096

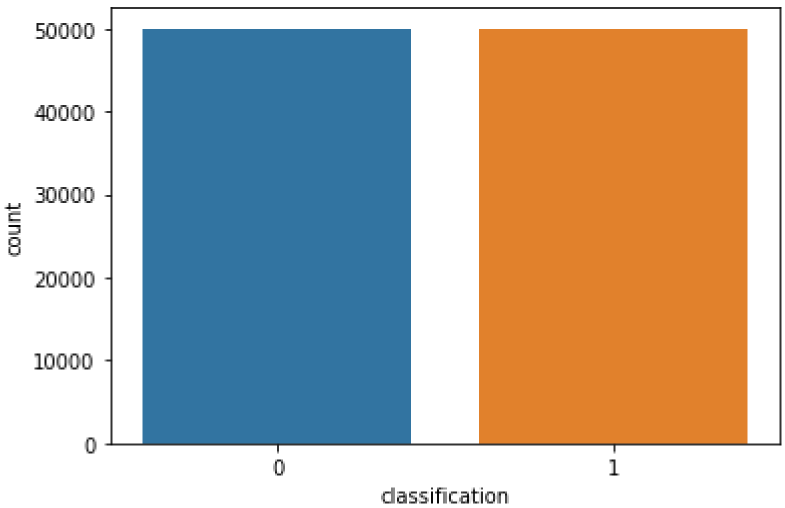
In [ ]:

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

In [ ]:

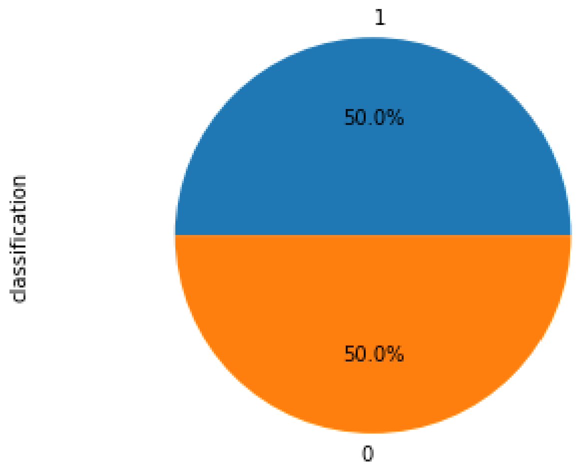
```
sns.countplot(data1["classification"])
plt.show()
```

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



In [ ]:

```
# Plot -using a Pie chart
data1["classification"].value_counts().plot(kind="pie",autopct="%1.1f%%")
plt.axis("equal")
plt.show()
```



In [ ]:

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

Out[ ]:

```
1000    benign
1001    benign
1002    benign
1003    benign
1004    benign
Name: classification, dtype: object
```

In [ ]:

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

Out[ ]:

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

In [ ]:

```
# find the pairwise correlation of all columns in the dataframe
corr=data1.corr()
# Return the first 35 rows ordered by columns in descending order.
corr.nlargest(35,'classification')["classification"]
```

Out[ ]:

```
classification       1.000000e+00
prio                 1.100359e-01
last_interval        6.952036e-03
min_flt              3.069595e-03
millisecond          5.482134e-15
gtime               -1.441608e-02
stime               -4.203713e-02
free_area_cache     -5.123678e-02
total_vm            -5.929110e-02
state               -6.470178e-02
mm_users            -9.364091e-02
reserved_vm         -1.186078e-01
fs_excl_counter     -1.378830e-01
nivcsw              -1.437912e-01
exec_vm             -2.551234e-01
map_count           -2.712274e-01
static_prio         -3.179406e-01
end_data            -3.249535e-01
maj_flt             -3.249535e-01
shared_vm           -3.249535e-01
vm_truncate_count   -3.548607e-01
utime               -3.699309e-01
nvcsw               -3.868893e-01
Name: classification, dtype: float64
```

In [ ]:

```

```

In [ ]:

```
# 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[ ]:

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

In [ ]:

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

Out[ ]:

```
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 [ ]:

```
#scikit-learn is a library for machine learning algorithms
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
```

In [ ]:

```
#  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 [ ]:

```
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
model.fit(x_train,y_train)
```

Out[ ]:

```
GaussianNB()
```

In [ ]:

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

Out[ ]:

```
array([1, 1, 1, ..., 1, 0, 1])
```

In [ ]:

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

Out[ ]:

```
0.6274
```

In [ ]:

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

Out[ ]:

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

30000 rows × 2 columns

In [ ]: