Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Exploring the malware dataset.

Malware Dataset: https://github.com/PacktPublishing/Mastering-Machine-Learning-for-Penetration-Testing/blob/master/Chapter03/MalwareData.csv.gz (https://github.com/PacktPublishing/Mastering-Machine-Learning-for-Penetration-Testing/blob/master/Chapter03/MalwareData.csv.gz)

```
-> 41,323 binaries(exe, dll) - legitimate
-> 96,724 malware files from virusshare.com
```

In [2]:

```
malData = pd.read_csv("E:/4th semester/Cybersecurity/Project File/\MalwareData.csv", sep =
```

In [3]:

```
malData.head()
```

Out[3]:

	Name	md5	Machine	SizeOfOptionalHeader	Characteristi
0	memtest.exe	631ea355665f28d4707448e442fbf5b8	332	224	2
1	ose.exe	9d10f99a6712e28f8acd5641e3a7ea6b	332	224	33
2	setup.exe	4d92f518527353c0db88a70fddcfd390	332	224	33
3	DW20.EXE	a41e524f8d45f0074fd07805ff0c9b12	332	224	2
4	dwtrig20.exe	c87e561258f2f8650cef999bf643a731	332	224	2

5 rows × 57 columns

In [4]:

```
malData.shape
```

Out[4]:

```
(138047, 57)
```

In [5]:

```
malData.describe()
```

Out[5]:

	Machine	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVer
count	138047.000000	138047.000000	138047.000000	138047.000000	138047.00(
mean	4259.069274	225.845632	4444.145994	8.619774	3.81(
std	10880.347245	5.121399	8186.782524	4.088757	11.86:
min	332.000000	224.000000	2.000000	0.000000	0.00(
25%	332.000000	224.000000	258.000000	8.000000	0.00(
50%	332.000000	224.000000	258.000000	9.000000	0.00(
75%	332.000000	224.000000	8226.000000	10.000000	0.00(
max	34404.000000	352.000000	49551.000000	255.000000	255.00(

8 rows × 55 columns

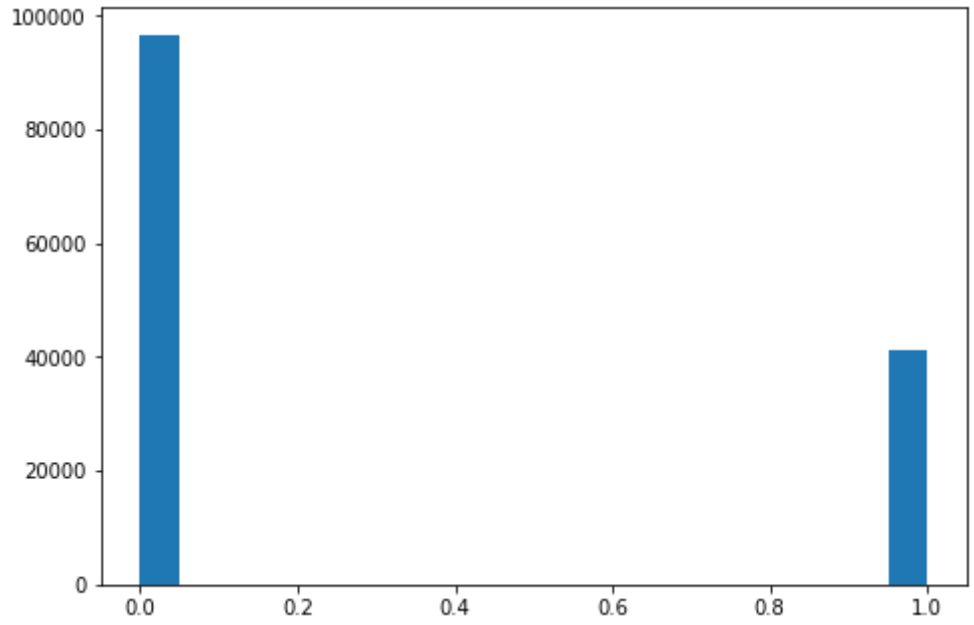
In [6]:

```
legit = malData[0:41323].drop(["legitimate"], axis=1)
mal = malData[41323::].drop(["legitimate"],axis=1)
print("The shape of the legit dataset is: %s sample, %s features"%(legit.shape[0],legit.sha
print("The shape of the malware dataset is: %s sample, %s features"%(mal.shape[0],mal.shape
```

```
The shape of the legit dataset is: 41323 sample, 56 features
The shape of the malware dataset is: 96724 sample, 56 features
```

In [7]:

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.hist(malData['legitimate'],20)
plt.show()
```



Data Cleaning

In [8]:

```
y=malData['legitimate']
malData = malData.drop(['legitimate'],axis=1)
```

In [9]:

```
malData = malData.drop(['Name'],axis=1)
malData = malData.drop(['md5'],axis=1)
print("The Name and md5 variables are reomved successfully")
```

```
The Name and md5 variables are reomved successfully
```

Spliting the dataset into test and train

In [10]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(malData,y,test_size=0.2, random_state=4
```

In [11]:

```
x_train.shape
```

Out[11]:

```
(110437, 54)
```

Model Building

1- Random Forest

In [12]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
clf = RandomForestClassifier(max_depth = 2, random_state=0)
randomModel = clf.fit(x_train, y_train)
```

Random forest Evaluation on test data

In [13]:

```
from sklearn.metrics import f1_score,accuracy_score,plot_confusion_matrix,auc,confusion_mat
```

In [14]:

```
# Accuracy on the train dataset
train_pred = randomModel.predict(x_train)
accuracy_score(y_train,train_pred)
```

Out[14]:

```
0.9828318407780001
```

In [15]:

```
# Accuracy on the test dataset
prediction = randomModel.predict(x_test)
accuracy_score(y_test,prediction)
```

Out[15]:

```
0.9838102136906918
```

In [16]:

```
f1_score(y_test,prediction)
```

Out[16]:

```
0.9730933606212002
```

In [54]:

```
confusion_matrix(y_test, prediction)
```

Out[54]:

```
array([[19080,   170],
       [  277,  8083]], dtype=int64)
```

2 - Logistic regresion

In [18]:

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=0)
logModel = clf.fit(x_train, y_train)
```

```
F:\apps\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:814: Co
nvergenceWarning: lbfgs failed to converge (status=2):
ABNORMAL_TERMINATION_IN_LNSRCH.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

Model Evaluation

In [19]:

```
# Accuracy on the train dataset
train_log = logModel.predict(x_train)
accuracy_score(y_train,train_log)
```

Out[19]:

```
0.7015221347917817
```

In [20]:

```
# Accuracy on the test dataset
pred=logModel.predict(x_test)
accuracy_score(y_test,pred)
```

Out[20]:

```
0.6972111553784861
```

In [21]:

```
f1_score(y_test, pred)
```

Out[21]:

```
0.0
```

In [53]:

```
confusion_matrix(y_test, pred)
```

Out[53]:

```
array([[19250,     0],
       [ 8360,     0]], dtype=int64)
```

3- AdaBoost

In [23]:

```
from sklearn.ensemble import AdaBoostClassifier
```

In [24]:

```
AdaModel = AdaBoostClassifier(n_estimators=100, learning_rate = 1)
```

Model Evaluation

In [25]:

```
adaBoostmodel = AdaModel.fit(x_train, y_train)
```

In [26]:

```
#Accuracy of train dataset
train_ada = adaBoostmodel.predict(x_train)
accuracy_score(y_train,train_ada)
```

Out[26]:

```
0.9887718789898313
```

In [27]:

```
# Accuracy on the test dataset
pred_ada=adaBoostmodel.predict(x_test)
accuracy_score(y_test,pred_ada)
```

Out[27]:

```
0.9892068091271279
```

In [28]:

```
f1_score(y_test, pred_ada)
```

Out[28]:

```
0.9821812963405883
```

In [52]:

```
confusion_matrix(y_test, pred_ada)
```

Out[52]:

```
array([[19099,   151],
       [  147,  8213]], dtype=int64)
```

4- SVM

In [30]:

```
from sklearn import svm
```

In [31]:

```
#Create a svm Classifier
xyz = svm.SVC()
```

Model Evaluation

In [32]:

```
svmModel=xyz.fit(x_train, y_train)
```

In [33]:

```
#Accuracy of train dataset
train_svm = svmModel.predict(x_train)
accuracy_score(y_train,train_svm)
```

Out[33]:

```
0.7015492996006774
```

In [34]:

```
# Accuracy on the test dataset
pred_svm=svmModel.predict(x_test)
accuracy_score(y_test,pred_svm)
```

Out[34]:

```
0.6972473741398044
```

In [35]:

```
f1_score(y_test, pred_svm)
```

Out[35]:

```
0.0002392058366224136
```

In [51]:

```
confusion_matrix(y_test, pred_svm)
```

Out[51]:

```
array([[19250,     0],
       [ 8359,     1]], dtype=int64)
```

5 - Neural Network

In [39]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [41]:

```
#Define model
NNmodel = Sequential()
NNmodel.add(Dense(16, input_dim=54, activation = "relu"))
NNmodel.add(Dense(8, activation= "relu"))
NNmodel.add(Dense(4, activation= "relu"))
NNmodel.add(Dense(1, activation= "sigmoid"))
NNmodel.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 16)                880

 dense_5 (Dense)             (None, 8)                 136

 dense_6 (Dense)             (None, 4)                 36

 dense_7 (Dense)             (None, 1)                 5

=================================================================
Total params: 1,057
Trainable params: 1,057
Non-trainable params: 0
_________________________________________________________________
```

In [43]:

```
# Compile model
NNmodel.compile(loss = "binary_crossentropy", optimizer = "rmsprop", metrics = ["accuracy"]
```

In [45]:

```
#fit Model
NNmodel.fit(x_train, y_train, epochs=5, batch_size=32)
```

```
Epoch 1/5
3452/3452 [==============================] - 4s 958us/step - loss: 16072966.
0000 - accuracy: 0.9309
Epoch 2/5
3452/3452 [==============================] - 3s 977us/step - loss: 12942463.
0000 - accuracy: 0.9479
Epoch 3/5
3452/3452 [==============================] - 3s 945us/step - loss: 10092774.
0000 - accuracy: 0.9477
Epoch 4/5
3452/3452 [==============================] - 3s 951us/step - loss: 6608472.0
000 - accuracy: 0.9478
Epoch 5/5
3452/3452 [==============================] - 3s 968us/step - loss: 6379676.0
000 - accuracy: 0.9467
```

Out[45]:

```
<keras.callbacks.History at 0x19297003100>
```

Model Evaluation

In [47]:

```
# Accuracy on the training dataset
trainPred = NNmodel.predict(x_train)
trainPred=[1 if y>=0.5 else 0  for y in trainPred]
accuracy_score(y_train, trainPred)
```

Out[47]:

```
0.9604027635665583
```

In [48]:

```
# Accuracy of the test dataset
y_prediction=NNmodel.predict(x_test)
y_prediction=[1 if y>=0.5 else 0  for y in y_prediction]
accuracy_score(y_test, y_prediction)
```

Out[48]:

```
0.9612459253893517
```

In [49]:

```
f1_score(y_test, y_prediction)
```

Out[49]:

```
0.9384986780089665
```

In [50]:

```
confusion_matrix(y_test, y_prediction)
```

Out[50]:

```
array([[18376,   874],
       [  196,  8164]], dtype=int64)
```

In [ ]:

```

```