Logisticregression on UCI repository malware dataset.

Prakhar Khanduri (19BCE0486) Malware Detection

Import Libraries

In [25]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset(Malware Dataset)

In [26]:
```
dataset = pd.read_csv('/content/Malware dataset.csv')
dataset2 = dataset.copy()
dataset2 = dataset.drop(['classification'], axis=1)

X = dataset2.iloc[:,1:].values
y = dataset.iloc[:, 2].values
print (X)
print(y)
```

```
[[   0    0    0 ...    0    0    0]
 [   1    0    0 ...    0    0    0]
 [   2    0    0 ...    0    0    0]
 ...
 [ 997 4096    0 ...    0    0    0]
 [ 998 4096    0 ...    0    0    0]
 [ 999 4096    0 ...    0    0    0]]
['malware' 'malware' 'malware' ... 'malware' 'malware' 'malware']
```

In [26]:
```

```

New Section

Label Encoding(Converting Categorical Data to Numbers So that can work on them. Here converting Malware and Bening)

In [27]:
```
dataset.head()
```

Out[27]:

	hash	millisecond	classification	state	usage_counter	prio	static_prio	normal_prio	poli
0	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	0	malware	0	0	3069378560	14274	0	
1	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	1	malware	0	0	3069378560	14274	0	
2	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	2	malware	0	0	3069378560	14274	0	
3	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	3	malware	0	0	3069378560	14274	0	
4	42fb5e2ec009a05ff5143227297074f1e9c6c3ebb9c914...	4	malware	0	0	3069378560	14274	0	

5 rows × 35 columns

In [28]:
```
dataset.describe()
```

Out[28]:

	millisecond	state	usage_counter	prio	static_prio	normal_prio	policy	vm_pgoff	vm_truncate_count	task_si
count	100000.000000	1.000000e+05	100000.0	1.000000e+05	100000.000000	100000.0	100000.0	100000.0	100000.000000	100000
mean	499.500000	1.577683e+05	0.0	3.069706e+09	18183.900070	0.0	0.0	0.0	15312.739510	0
std	288.676434	9.361726e+05	0.0	2.963061e+05	4609.792765	0.0	0.0	0.0	3256.475008	0
min	0.000000	0.000000e+00	0.0	3.069190e+09	13988.000000	0.0	0.0	0.0	9695.000000	0
25%	249.750000	0.000000e+00	0.0	3.069446e+09	14352.000000	0.0	0.0	0.0	12648.000000	0
50%	499.500000	0.000000e+00	0.0	3.069698e+09	16159.000000	0.0	0.0	0.0	15245.000000	0
75%	749.250000	4.096000e+03	0.0	3.069957e+09	22182.000000	0.0	0.0	0.0	17663.000000	0
max	999.000000	4.326605e+07	0.0	3.070222e+09	31855.000000	0.0	0.0	0.0	27157.000000	0

8 rows × 33 columns

In [29]:
```
dataset.isna().sum()
```

Out[29]:
```
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
```

Label Encoding(Converting Categorical Data to Numbers So that can work on them. Here converting Malware and Bening)

In [30]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print (y)
```

```
[1 1 1 ... 1 1 1]
```

Splitting the dataset into the Training set and Test set. (75% data is for traing and 25% Testing data)

In [31]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

In [32]:
```
print(X_train)
```

```
[[  606 12288     0 ...    10     0     0]
 [  228 28672     0 ...     0     0     0]
 [  382  4096     0 ...     1     0     0]
 ...
 [  613 12288     0 ...    11     0     0]
 [  567  4096     0 ...     2     0     0]
 [  268     0     0 ...     0     0     0]]
```

In [33]:
```
print(y_train)
```

```
[0 1 0 ... 0 0 1]
```

In [34]:
```
print(X_test)
```

```
[[   582       0       0 ...       8       0       0]
 [   498       0       0 ...       0       0       0]
 [   227 1028096       0 ...       4       0       0]
 ...
 [   585    4096       0 ...       0       0       0]
 [   519       0       0 ...       7       0       0]
 [   831       0       0 ...       0       0       0]]
```

In [35]:
```
print(y_test)
```

```
[1 0 0 ... 1 0 0]
```

Feature Scaling (Scaling down the data using funtcion StandardScaler()) 19BCE0486

In [36]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

X_train after scaling down

In [37]:
```
print(X_train)
```

```
[[ 0.36731703 -0.15969785  0.          ...  2.5559669   0.
   0.        ]
 [-0.941068   -0.14179605  0.          ... -0.50896701  0.
   0.        ]
 [-0.40802225 -0.16864875  0.          ... -0.20247362  0.
   0.        ]
 ...
 [ 0.39154638 -0.15969785  0.          ...  2.8624603   0.
   0.        ]
 [ 0.23323492 -0.16864875  0.          ...  0.11401977  0.
   0.        ]
 [-0.80261455 -0.1731242   0.          ... -0.50896701  0.
   0.        ]]
```

X_test after scaling

In [38]:
```
print(X_test)
```

```
[[ 0.28424496 -0.1731242   0.          ...  1.94298012  0.
   0.        ]
 [-0.00650727 -0.1731242   0.          ... -0.50896701  0.
   0.        ]
 [-0.94452933  0.95021383  0.          ...  0.71700655  0.
   0.        ]
 ...
 [ 0.29462897 -0.16864875  0.          ... -0.50896701  0.
   0.        ]
 [ 0.04618079 -0.1731242   0.          ...  1.63648673  0.
   0.        ]
 [ 1.14611763 -0.1731242   0.          ... -0.50896701  0.
   0.        ]]
```

Training Model on the Training set(That is traing the Model on 75% of the data set malware detection)

Using Logisticregression Traing my model.

In [39]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Out[39]:
```
LogisticRegression(random_state=0)
```

Predicting the Test set results(Prediction of the trained model on the 25% test data)

In [40]:
```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[1 1]
 [0 0]
 [0 0]
 ...
 [1 1]
 [0 0]
 [0 0]]
```

Confusion Matrix and Accuracy(Finding out the accuracy of the model using confusion matrix)

In [41]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix\n",cm)
```

```
Confusion Matrix
 [[11590   938]
 [  560 11912]]
```

Final Result (Logisticregression) Accuracy , Precession, Recal

In [47]:
```
from sklearn import metrics
print("Final Results")
print("Logisticregression Malware Data set")
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
print("Precision",metrics.precision_score(y_test,y_pred))
print("Recall:",metrics.recall_score(y_test,y_pred))
```

```
Final Results
Logisticregression Malware Data set
Accuracy: 0.94008
Precision 0.9270038910505837
Recall: 0.9550394227068633
```

ROC Curve

In [46]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test,y_pred)
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```