In [32]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [2]:

```
df = pd.read_csv("dynamic_api_call_sequence_per_malware_100_0_306.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	hash	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	...	t_91	t_92	t_93	t_94	t_95	t_96	t_97	t_98	t_99	malware
0	071e8c3f8922e186e57548cd4c703a5d	112	274	158	215	274	158	215	298	76	...	71	297	135	171	215	35	208	56	71	1
1	33f8e6d08a6aae939f25a8e0d63dd523	82	208	187	208	172	117	172	117	172	...	81	240	117	71	297	135	171	215	35	1
2	b68abd064e975e1c6d5f25e748663076	16	110	240	117	240	117	240	117	240	...	65	112	123	65	112	123	65	113	112	1
3	72049be7bd30ea61297ea624ae198067	82	208	187	208	172	117	172	117	172	...	208	302	208	302	187	208	302	228	302	1
4	c9b3700a77facf29172f32df6bc77f48	82	240	117	240	117	240	117	240	117	...	209	260	40	209	260	141	260	141	260	1

5 rows × 102 columns

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
hash       0
t_0        0
t_1        0
t_2        0
t_3        0
          ..
t_96       0
t_97       0
t_98       0
t_99       0
malware    0
Length: 102, dtype: int64
```

1. Drop column unwanted column from the data set (hash column).|

In [5]:

```
df = df.drop(['hash'], axis=1)
```

In [6]:

```
df.head()
```

Out[6]:

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	...	t_91	t_92	t_93	t_94	t_95	t_96	t_97	t_98	t_99	malware
0	112	274	158	215	274	158	215	298	76	208	...	71	297	135	171	215	35	208	56	71	1
1	82	208	187	208	172	117	172	117	172	117	...	81	240	117	71	297	135	171	215	35	1
2	16	110	240	117	240	117	240	117	240	117	...	65	112	123	65	112	123	65	113	112	1
3	82	208	187	208	172	117	172	117	172	117	...	208	302	208	302	187	208	302	228	302	1
4	82	240	117	240	117	240	117	240	117	172	...	209	260	40	209	260	141	260	141	260	1

5 rows × 101 columns

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43876 entries, 0 to 43875
Columns: 101 entries, t_0 to malware
dtypes: int64(101)
memory usage: 33.8 MB
```

2. Divide the given dataset into training and testing set, input (X) and output (Y) (malware column in the dataset) parameters.

In [8]:

```
X = df.drop(['malware'], axis=1)
y = df['malware']
```

In [9]:

```
X.head()
```

Out[9]:

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	...	t_90	t_91	t_92	t_93	t_94	t_95	t_96	t_97	t_98	t_99
0	112	274	158	215	274	158	215	298	76	208	...	117	71	297	135	171	215	35	208	56	71
1	82	208	187	208	172	117	172	117	172	117	...	60	81	240	117	71	297	135	171	215	35
2	16	110	240	117	240	117	240	117	240	117	...	123	65	112	123	65	112	123	65	113	112
3	82	208	187	208	172	117	172	117	172	117	...	215	208	302	208	302	187	208	302	228	302
4	82	240	117	240	117	240	117	240	117	172	...	40	209	260	40	209	260	141	260	141	260

5 rows × 100 columns

In [10]:

```
y.head()
```

Out[10]:

```
0    1
1    1
2    1
3    1
4    1
Name: malware, dtype: int64
```

In [11]:

```
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 = 13)
```

In [12]:

```
# check the shape of X_train and X_test
X_train.shape, X_test.shape
```

Out[12]:

```
((32907, 100), (10969, 100))
```

3. Apply NN algorithm over the given dataset (i.e. mlp = MLPClassifier(hidden_layer_sizes=(100,100,100),max_iter=1000, random_state=42))

In [13]:

```
mlp = MLPClassifier(hidden_layer_sizes=(100,100,100),max_iter=1000,random_state=42)
```

In [14]:

```
mlp.fit(X_train, y_train)
```

Out[14]:

```
MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=1000,
              random_state=42)
```

In [26]:

```
y_pred = mlp.predict(X_test)
```

4. Identify associated accuracy, precision, recall over the given dataset.

In [27]:

```
print("Training Accuracy = ", mlp.score(X_train, y_train))
```

```
Training Accuracy =  0.9975385176406236
```

In [28]:

```
print("Testing Accuracy = ", mlp.score(X_test, y_test))
```

```
Testing Accuracy =  0.9845017777372596
```

In [31]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.59      0.66       275
           1       0.99      0.99      0.99     10694

    accuracy                           0.98     10969
   macro avg       0.86      0.79      0.82     10969
weighted avg       0.98      0.98      0.98     10969
```

For parameters

In [41]:

```
error = []
# Calculating MAE error for K values between 1 and 39
for i in range(100, 105):
    mlp = MLPClassifier(hidden_layer_sizes=(i,i,i),max_iter=1000,random_state=42)
    mlp.fit(X_train, y_train)
    pred_i = mlp.predict(X_test)
    mae = mlp.score(X_test, pred_i)
    error.append(mae)
```

In [42]:

```
# import matplotlib.pyplot as plt
# plt.figure(figsize=(12, 6))

# plt.plot(range(100, 110), error, color='red',
# linestyle='dashed', marker='o',
# markerfacecolor='blue', markersize=100)

# plt.title('K Value MAE')
# plt.xlabel('K Value')
# plt.ylabel('Mean Absolute Error')

print(error)
```

```
[1.0, 1.0, 1.0, 1.0, 1.0]
```

In [40]:

```
mlp = MLPClassifier(hidden_layer_sizes=(50,100,100),max_iter=500,random_state=42)

mlp.fit(X_train, y_train)

y_pred = mlp.predict(X_test)

### 4. Identify associated accuracy, precision, recall over the given dataset.

print("Training Accuracy = ", mlp.score(X_train, y_train))

print("Testing Accuracy = ", mlp.score(X_test, y_test))

print(classification_report(y_test, y_pred))
```

```
Training Accuracy =  0.9935576017260765
Testing Accuracy =  0.9823137934178139
              precision    recall  f1-score   support

           0       0.66      0.60      0.63       275
           1       0.99      0.99      0.99     10694

    accuracy                           0.98     10969
   macro avg       0.83      0.80      0.81     10969
weighted avg       0.98      0.98      0.98     10969
```

In [ ]:

```

```