In [4]:
```
#Created by Rami ALmehdawi

# Import the necessary Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report, roc_
from imblearn.over_sampling import RandomOverSampler, SMOTE
```

In [5]:
```
# Read in the Data
MW_Data = pd.read_csv("Malware-staDyn-data.csv")
```

In [7]:
```
# Drops all Missing values in Label and selects the Feature Column
x = MW_Data.drop('label', axis = 1)
y = MW_Data["label"]

# Split into Training and Test Sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=42)
```

In [13]:
```
# Creats the kNN Model and fits it.
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
```

Out[13]:
```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [14]:
```
y_pred = knn.predict(x_test)

y_pred_proba = knn.predict_proba(x_test)[:, 1]
```

In [15]:
```
# Prints out Relevant Metrics
print("Accuracy: ", accuracy_score(y_test, y_pred))

print("AUC-ROC:", roc_auc_score(y_test, y_pred_proba))

print("F1 Score: ", f1_score(y_test, y_pred))
```

```
print("Precision:", precision_score(y_test, y_pred))

print("Recall: ", recall_score(y_test, y_pred))
```

```
Accuracy:  0.9712
AUC-ROC: 0.9696495930117133
F1 Score:  0.9837251356238698
Precision: 0.9819494584837545
Recall:  0.9855072463768116
```

In [17]:
```
# Balances the training data
smote = SMOTE(random_state=42)
x_train_resampled, y_train_resampled = smote.fit_resample(x_train, y_train)
```

In [18]:
```
# Creats the kNN Model and fits it.
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(x_train, y_train)
```

Out[18]:
```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [19]:
```
y_pred = knn.predict(x_test)

y_pred_proba = knn.predict_proba(x_test)[:, 1]
```

In [20]:
```
# Prints out Relevant Metrics
print("Accuracy: ", accuracy_score(y_test, y_pred))

print("AUC-ROC:", roc_auc_score(y_test, y_pred_proba))

print("F1 Score: ", f1_score(y_test, y_pred))

print("Precision:", precision_score(y_test, y_pred))

print("Recall: ", recall_score(y_test, y_pred))
```

```
Accuracy:  0.9712
AUC-ROC: 0.9696495930117133
F1 Score:  0.9837251356238698
Precision: 0.9819494584837545
Recall:  0.9855072463768116
```

In [22]:
```
# Creats the kNN Model and fits it.
knn = KNeighborsClassifier(n_neighbors = 1)
knn.fit(x_train_resampled, y_train_resampled)
```

Out[22]:
```
▾        KNeighborsClassifier

KNeighborsClassifier(n_neighbors=1)
```

In [23]:
```
y_pred = knn.predict(x_test)

y_pred_proba = knn.predict_proba(x_test)[:, 1]
```

In [24]:
```
# Prints out Relevant Metrics
print("Accuracy: ", accuracy_score(y_test, y_pred))

print("AUC-ROC:", roc_auc_score(y_test, y_pred_proba))

print("F1 Score: ", f1_score(y_test, y_pred))

print("Precision:", precision_score(y_test, y_pred))

print("Recall: ", recall_score(y_test, y_pred))
```

```
Accuracy:  0.9888
AUC-ROC: 0.9639418304546357
F1 Score:  0.993676603432701
Precision: 0.990990990990991
Recall:  0.9963768115942029
```

In [25]:
```
# Creats the kNN Model and fits it.
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train_resampled, y_train_resampled)
```

Out[25]:
```
▾        KNeighborsClassifier

KNeighborsClassifier(n_neighbors=3)
```

In [26]:
```
y_pred = knn.predict(x_test)

y_pred_proba = knn.predict_proba(x_test)[:, 1]
```

In [27]:
```
# Prints out Relevant Metrics
print("Accuracy: ", accuracy_score(y_test, y_pred))

print("AUC-ROC:", roc_auc_score(y_test, y_pred_proba))

print("F1 Score: ", f1_score(y_test, y_pred))

print("Precision:", precision_score(y_test, y_pred))
```

```
print("Recall: ", recall_score(y_test, y_pred))
```

```
Accuracy:  0.984
AUC-ROC: 0.9836212030970816
F1 Score:  0.9909255898366606
Precision: 0.9927272727272727
Recall:  0.9891304347826086
```

In [28]:
```
# Creats the kNN Model and fits it.
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(x_train_resampled, y_train_resampled)
```

Out[28]:
```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [29]:
```
y_pred = knn.predict(x_test)

y_pred_proba = knn.predict_proba(x_test)[:, 1]
print("AUC-ROC:", roc_auc_score(y_test, y_pred_proba))
```

```
AUC-ROC: 0.9839934484812388
```

In [30]:
```
# Prints out Relevant Metrics
print("Accuracy: ", accuracy_score(y_test, y_pred))

print("AUC-ROC:", roc_auc_score(y_test, y_pred_proba))

print("F1 Score: ", f1_score(y_test, y_pred))

print("Precision:", precision_score(y_test, y_pred))

print("Recall: ", recall_score(y_test, y_pred))
```

```
Accuracy:  0.9744
AUC-ROC: 0.9839934484812388
F1 Score:  0.9854014598540147
Precision: 0.9926470588235294
Recall:  0.9782608695652174
```

In [31]:
```
# Creats the kNN Model and fits it.
knn = KNeighborsClassifier(n_neighbors = 10)
knn.fit(x_train, y_train)
```

Out[31]:
```
▾        KNeighborsClassifier

KNeighborsClassifier(n_neighbors=10)
```

In [32]:
```
y_pred = knn.predict(x_test)

y_pred_proba = knn.predict_proba(x_test)[:, 1]
```

In [33]:
```
# Prints out Relevant Metrics
print("Accuracy: ", accuracy_score(y_test, y_pred))

print("AUC-ROC:", roc_auc_score(y_test, y_pred_proba))

print("F1 Score: ", f1_score(y_test, y_pred))

print("Precision:", precision_score(y_test, y_pred))

print("Recall: ", recall_score(y_test, y_pred))
```

```
Accuracy:  0.9728
AUC-ROC: 0.9744267421083979
F1 Score:  0.9846153846153847
Precision: 0.9837251356238698
Recall:  0.9855072463768116
```

In [34]:
```
# Creats the kNN Model and fits it.
knn = KNeighborsClassifier(n_neighbors = 20)
knn.fit(x_train, y_train)
```

Out[34]:
```
▾        KNeighborsClassifier

KNeighborsClassifier(n_neighbors=20)
```

In [35]:
```
y_pred = knn.predict(x_test)

y_pred_proba = knn.predict_proba(x_test)[:, 1]
```

In [36]:
```
# Prints out Relevant Metrics
print("Accuracy: ", accuracy_score(y_test, y_pred))

print("AUC-ROC:", roc_auc_score(y_test, y_pred_proba))

print("F1 Score: ", f1_score(y_test, y_pred))

print("Precision:", precision_score(y_test, y_pred))
```

```
print("Recall: ", recall_score(y_test, y_pred))
```

```
Accuracy:  0.9712
AUC-ROC: 0.9879888822711931
F1 Score:  0.9837545126353789
Precision: 0.9802158273381295
Recall:  0.9873188405797102
```

The Kernel crashed while executing code in the the current cell or a previous cell. Please review the code in the cell(s) to identify a possible cause of the failure. Click here for more info. View Jupyter log for further details.