In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.metrics import accuracy_score
import copy
```

Loading Dataset

In [2]:

```
input_data=pd.read_csv("data/BitcoinHeistData.csv")
```

In [3]:

```
print(input_data.head())
```

```
                              address  year  day  length    weight  count  \
0   111K8kZAEnJg245r2cM6y9zgJGHZtJPy6  2017   11      18  0.008333      1
1  1123pJv8jzeFQaCV4w644pzQJzVWay2zcA  2016  132      44  0.000244      1
2  112536im7hy6wtKbpH1qYDWtTyMRAcA2p7  2016  246       0  1.000000      1
3  1126eDRw2wqSkWosjTCre8cjjQW8sSeWH7  2016  322      72  0.003906      1
4  1129TSjKtx65E35GiUo4AYVeyo48twbrGX  2016  238     144  0.072848    456

   looped  neighbors       income            label
0       0          2  100050000.0  princetonCerber
1       0          1  100000000.0   princetonLocky
2       0          2  200000000.0  princetonCerber
3       0          2   71200000.0  princetonCerber
4       0          1  200000000.0   princetonLocky
```

Preprocessing

Checking for Duplicates

In [4]:

```
print('No of duplicates in the Input Data:',sum(input_data.duplicated()))
```

```
No of duplicates in the Input Data: 0
```

Checking for NaN/null values

In [5]:

```
print('No of NaN/Null values in Input Data:',input_data.isnull().values.sum())
```

```
No of NaN/Null values in Input Data: 0
```

Data Prepration

In [6]:

```
X = input_data.drop(['label'], axis = 1)
Y = input_data['label']

print(X.head())
```

```
                              address  year  day  length    weight  count  \
0   111K8kZAEnJg245r2cM6y9zgJGHZtJPy6  2017   11      18  0.008333      1
1  1123pJv8jzeFQaCV4w644pzQJzVWay2zcA  2016  132      44  0.000244      1
2  112536im7hy6wtKbpH1qYDWtTyMRAcA2p7  2016  246       0  1.000000      1
3  1126eDRw2wqSkWosjTCre8cjjQW8sSeWH7  2016  322      72  0.003906      1
4  1129TSjKtx65E35GiUo4AYVeyo48twbrGX  2016  238     144  0.072848    456

   looped  neighbors       income
0       0          2  100050000.0
1       0          1  100000000.0
2       0          2  200000000.0
3       0          2   71200000.0
4       0          1  200000000.0
```

Feature Subset Selection

In [7]:

```
e type of Ransomware and including it in X will cause overfitting of the model. Therefore, w
)
```

```
   year  day  length    weight  count  looped  neighbors       income
0  2017   11      18  0.008333      1       0          2  100050000.0
1  2016  132      44  0.000244      1       0          1  100000000.0
2  2016  246       0  1.000000      1       0          2  200000000.0
3  2016  322      72  0.003906      1       0          2   71200000.0
4  2016  238     144  0.072848    456       0          1  200000000.0
```

Label Encoding

In [8]:

```
# Transforming non-numerical value in Y to numerical value using label Encoder
le = preprocessing.LabelEncoder()
le.fit(Y)
Y = le.transform(Y)
print(le.classes_)
```

```
['montrealAPT' 'montrealComradeCircle' 'montrealCryptConsole'
 'montrealCryptXXX' 'montrealCryptoLocker' 'montrealCryptoTorLocker2015'
 'montrealDMALocker' 'montrealDMALockerv3' 'montrealEDA2' 'montrealFlyper'
 'montrealGlobe' 'montrealGlobeImposter' 'montrealGlobev3'
 'montrealJigSaw' 'montrealNoobCrypt' 'montrealRazy' 'montrealSam'
 'montrealSamSam' 'montrealVenusLocker' 'montrealWannaCry'
 'montrealXLocker' 'montrealXLockerv5.0' 'montrealXTPLocker'
 'paduaCryptoWall' 'paduaJigsaw' 'paduaKeRanger' 'princetonCerber'
 'princetonLocky' 'white']
```

Normalising the data

In [9]:

```
X_n = preprocessing.normalize(X)
```

Feature Scaling

In [10]:

```
# MinMaxScalar
from sklearn.preprocessing import MinMaxScaler
scaler1 = MinMaxScaler().fit(X_n)
X_mm = scaler1.transform(X_n)

# Standard Scaler
# from sklearn.preprocessing import StandardScaler
# scaler2 = StandardScaler().fit(X_n)
# X_st = scaler2.transform(X_n)
```

Training the model

Decision Tree

In [11]:

```
k=5
n=len(X_mm)//k
train_accuracy_scores=[]
test_accuracy_scores=[]
# Using K-fold cross validation
for i in range(k):
    X_dummy=copy.deepcopy(X_mm)
    Y_dummy=copy.deepcopy(Y)
    #Train-test split

    X_test=X_dummy[n*i:n*(i+1)]
    Y_test=Y_dummy[n*i:n*(i+1)]
    X_train=[]
    Y_train=[]
    if i==0:
        X_train=X_dummy[n:]
        Y_train=Y_dummy[n:]
    else:
        X_t=X_dummy[0:n*i]
        Y_t=Y_dummy[0:n*i]
        X_tt=X_dummy[n*(i+1):]
        Y_tt=Y_dummy[n*(i+1):]
        X_train=np.concatenate((X_t,X_tt))
        Y_train=np.concatenate((Y_t,Y_tt))

    # model training
    clf_D = tree.DecisionTreeClassifier()
    clf_D = clf_D.fit(X_train, Y_train)

    #Accuracy calculation
    Y_train_pred = clf_D.predict(X_train)
    Y_test_pred = clf_D.predict(X_test)
    train_accuracy_scores.append(accuracy_score(Y_train, Y_train_pred))
    test_accuracy_scores.append(accuracy_score(Y_test, Y_test_pred))
print('-------------------------------------------------------------------------------')
print('Accuracy Score on Training Data:',np.mean(train_accuracy_scores))
print('\n\n-------------------------------------------------------------------------------')
print('Accuracy Score on Test Data:',np.mean(test_accuracy_scores))
print('\n-------------------------------------------------------------------------------')
```

```
-------------------------------------------------------------------------------
------------------------

Accuracy Score on Training Data: 0.9996392323852575


-------------------------------------------------------------------------------
--------------------

Accuracy Score on Test Data: 0.9353830277077309

-------------------------------------------------------------------------------
--------------------
```

Random Forest

In [12]:

```
from sklearn.ensemble import RandomForestClassifier
k=5
n=len(X_mm)//k
max_depths=[2,3,4,5]
random_states=[0,1]
for depth in max_depths:
    for state in random_states:
        print('\n\n\n----------------------------------------------------------------------------')
        print('----------------------------------------------------------------------------')
        print('----------When max_depth =',depth,' and random state =',state,' ---------')
        train_accuracy_scores=[]
        test_accuracy_scores=[]
        # Using K-fold cross validation
        for i in range(k):
            X_dummy=copy.deepcopy(X_mm)
            Y_dummy=copy.deepcopy(Y)
            #Train-test split

            X_test=X_dummy[n*i:n*(i+1)]
            Y_test=Y_dummy[n*i:n*(i+1)]
            X_train
            Y_train
            if i==0:
                X_train=X_dummy[n:]
                Y_train=Y_dummy[n:]
            else:
                X_t=X_dummy[0:n*i]
                Y_t=Y_dummy[0:n*i]
                X_tt=X_dummy[n*(i+1):]
                Y_tt=Y_dummy[n*(i+1):]
                X_train=np.concatenate((X_t,X_tt))
                Y_train=np.concatenate((Y_t,Y_tt))

            # model training
            clf_R = RandomForestClassifier(max_depth=depth, random_state=state)
            clf_R = clf_R.fit(X_train, Y_train)

            #Accuracy calculation
            Y_train_pred = clf_R.predict(X_train)
            Y_test_pred = clf_R.predict(X_test)
            train_accuracy_scores.append(accuracy_score(Y_train, Y_train_pred))
            test_accuracy_scores.append(accuracy_score(Y_test, Y_test_pred))
        print('Accuracy Score on Training Data:',np.mean(train_accuracy_scores))
        print('Accuracy Score on Test Data:',np.mean(test_accuracy_scores))
```

```
----------------------------------------------------------------------------
------------------------


----------------------------------------------------------------------------
------------------------


----------When max_depth = 2  and random state = 0  ---------
Accuracy Score on Training Data: 0.9796468402483162
```

```
Accuracy Score on Test Data: 0.9730125886557665



------------------------------------------------------------------------
------------------------

------------------------------------------------------------------------
------------------------


----------When max_depth = 2  and random state = 1  ---------
Accuracy Score on Training Data: 0.9801899836023971
Accuracy Score on Test Data: 0.9772597068038396



------------------------------------------------------------------------
------------------------

------------------------------------------------------------------------
------------------------


----------When max_depth = 3  and random state = 0  ---------
Accuracy Score on Training Data: 0.9824401339759801
Accuracy Score on Test Data: 0.9804234950158263



------------------------------------------------------------------------
------------------------

------------------------------------------------------------------------
------------------------


----------When max_depth = 3  and random state = 1  ---------
Accuracy Score on Training Data: 0.9851944298786197
Accuracy Score on Test Data: 0.9853235242918484



------------------------------------------------------------------------
------------------------

------------------------------------------------------------------------
------------------------


----------When max_depth = 4  and random state = 0  ---------
Accuracy Score on Training Data: 0.9879082824229665
Accuracy Score on Test Data: 0.9811546719531687



------------------------------------------------------------------------
------------------------

------------------------------------------------------------------------
------------------------
```

```
----------When max_depth = 4  and random state = 1  ---------
Accuracy Score on Training Data: 0.989902151745714
Accuracy Score on Test Data: 0.9785577074769386



------------------------------------------------------------------------------
------------------------

------------------------------------------------------------------------------
------------------------


----------When max_depth = 5  and random state = 0  ---------
Accuracy Score on Training Data: 0.9907936810313196
Accuracy Score on Test Data: 0.9770535688778586



------------------------------------------------------------------------------
------------------------

------------------------------------------------------------------------------
------------------------


----------When max_depth = 5  and random state = 1  ---------
Accuracy Score on Training Data: 0.9914179092335152
Accuracy Score on Test Data: 0.9748013999402063
```

Naive Bayes Classifier

In [13]:

```
from sklearn.naive_bayes import MultinomialNB
k=5
n=len(X_mm)//k
train_accuracy_scores=[]
test_accuracy_scores=[]
# Using K-fold cross validation
for i in range(k):
    X_dummy=copy.deepcopy(X_mm)
    Y_dummy=copy.deepcopy(Y)
    #Train-test split

    X_test=X_dummy[n*i:n*(i+1)]
    Y_test=Y_dummy[n*i:n*(i+1)]
    X_train
    Y_train
    if i==0:
        X_train=X_dummy[n:]
        Y_train=Y_dummy[n:]
    else:
        X_t=X_dummy[0:n*i]
        Y_t=Y_dummy[0:n*i]
        X_tt=X_dummy[n*(i+1):]
        Y_tt=Y_dummy[n*(i+1):]
        X_train=np.concatenate((X_t,X_tt))
        Y_train=np.concatenate((Y_t,Y_tt))

    # model training
    clf_N = MultinomialNB()
    clf_N = clf_N.fit(X_train, Y_train)

    #Accuracy calculation
    Y_train_pred = clf_N.predict(X_train)
    Y_test_pred = clf_N.predict(X_test)
    train_accuracy_scores.append(accuracy_score(Y_train, Y_train_pred))
    test_accuracy_scores.append(accuracy_score(Y_test, Y_test_pred))
print('-------------------------------------------------------------------------------------')
print('Accuracy Score on Training Data:',np.mean(train_accuracy_scores))
print('\n\n-------------------------------------------------------------------------------------')
print('Accuracy Score on Test Data:',np.mean(test_accuracy_scores))
print('\n-------------------------------------------------------------------------------------')
```

```
-------------------------------------------------------------------------------
------------------------

Accuracy Score on Training Data: 0.9858014072422663


-------------------------------------------------------------------------------
--------------------

Accuracy Score on Test Data: 0.9858013950721622

-------------------------------------------------------------------------------
--------------------
```

KNN classifier

In [14]:

```
from sklearn.neighbors import KNeighborsClassifier
k=2
n=len(X_mm)//k
nearest_neighbours=[2,3,4]
for neighbour in nearest_neighbours:
    print('\n\n\n--------------------------------------------------------------------------------')
    print('--------------------------------------------------------------------------------')
    print('----------When we have ',neighbour,' nearest neighbours ---------')
    train_accuracy_scores=[]
    test_accuracy_scores=[]
    # Using K-fold cross validation
    for i in range(k):
        X_dummy=copy.deepcopy(X_mm)
        Y_dummy=copy.deepcopy(Y)
        #Train-test split

        X_test=X_dummy[n*i:n*(i+1)]
        Y_test=Y_dummy[n*i:n*(i+1)]
        X_train
        Y_train
        if i==0:
            X_train=X_dummy[n:]
            Y_train=Y_dummy[n:]
        else:
            X_t=X_dummy[0:n*i]
            Y_t=Y_dummy[0:n*i]
            X_tt=X_dummy[n*(i+1):]
            Y_tt=Y_dummy[n*(i+1):]
            X_train=np.concatenate((X_t,X_tt))
            Y_train=np.concatenate((Y_t,Y_tt))

        # model training
        clf_K = KNeighborsClassifier(n_neighbors=neighbour)
        clf_K = clf_K.fit(X_train, Y_train)

        #Accuracy calculation
        Y_train_pred = clf_K.predict(X_train)
        Y_test_pred = clf_K.predict(X_test)
        train_accuracy_scores.append(accuracy_score(Y_train, Y_train_pred))
        test_accuracy_scores.append(accuracy_score(Y_test, Y_test_pred))
    print('Accuracy Score on Training Data:',np.mean(train_accuracy_scores))
    print('Accuracy Score on Test Data:',np.mean(test_accuracy_scores))
```

```
--------------------------------------------------------------------------------
--------------------------


--------------------------------------------------------------------------------
--------------------------


----------When we have  2  nearest neighbours ---------
Accuracy Score on Training Data: 0.9632180389416014
Accuracy Score on Test Data: 0.9551857569571339
```

```
------------------------------------------------------------------------------
--------------------------

------------------------------------------------------------------------------
--------------------------


----------When we have  3  nearest neighbours ---------
Accuracy Score on Training Data: 0.9728231856804006
Accuracy Score on Test Data: 0.9615643830943675



------------------------------------------------------------------------------
--------------------------

------------------------------------------------------------------------------
--------------------------


----------When we have  4  nearest neighbours ---------
Accuracy Score on Training Data: 0.9733624540701200
Accuracy Score on Test Data: 0.9570979596664613
```