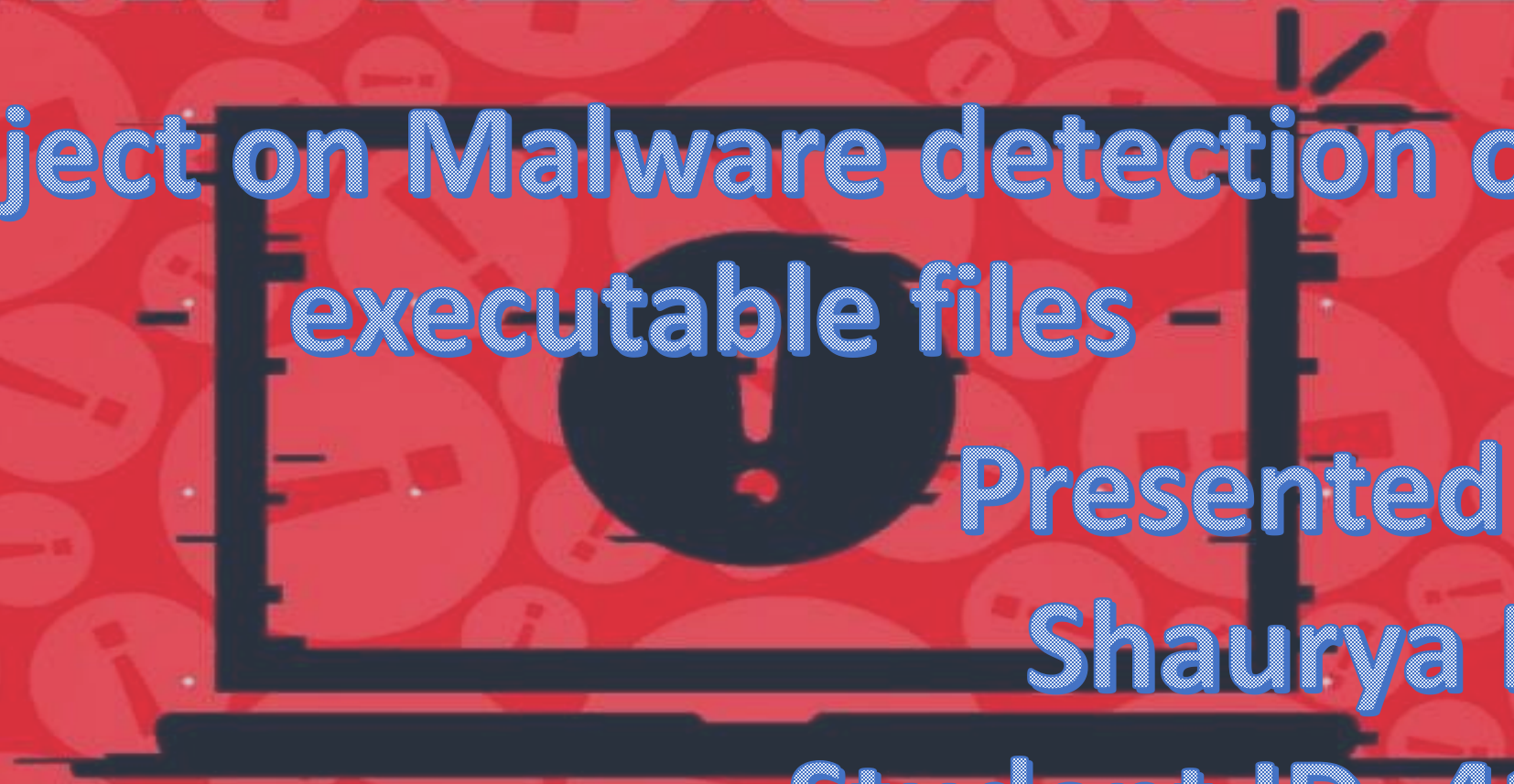
MALWARE

Project on Malware detection on executable files

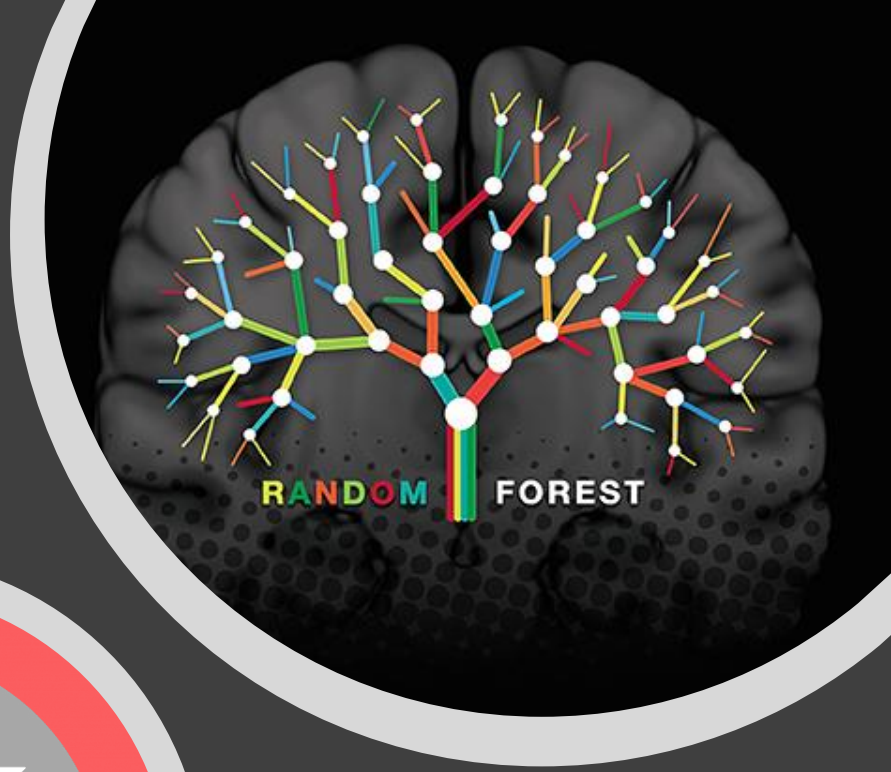


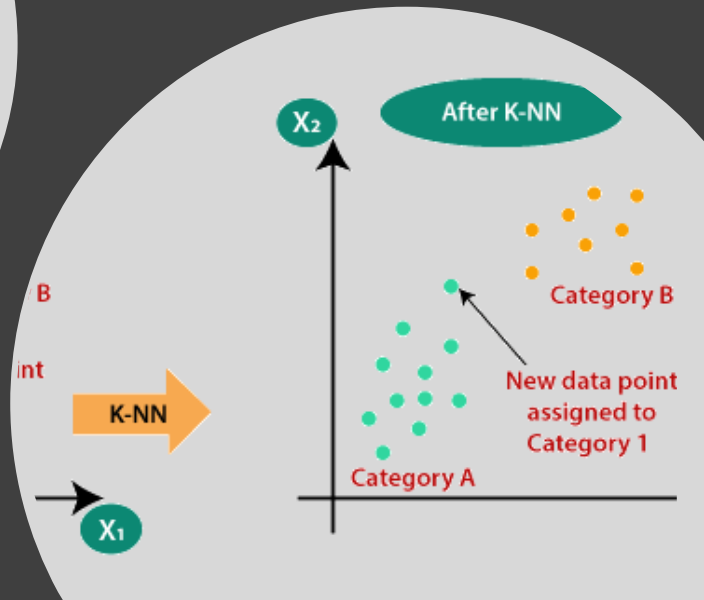
Presented by :
Shaurya Pal
Student ID: 4128742

DETECTION ENGINE

- AIM : The objective of this project was to find whether the executable files were malicious or not, before even installing it with the help of statistical analysis and machine learning algorithms. I decided to work with supervised learning model as random forest being my default model and KNN being the second as they are both non-para-metric model.

DATA SET: 373 samples, 531 features





Importing the libraries

In [1]:
```
import numpy as np
import os
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

DATA PREPROCESSING

importing the data set

In [2]:
```
df=pd.read_csv('C:/Users/shaur/Downloads/uci_malware_detection.csv')
df.head()
```

Out[2]:

	Label	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	...	F_522	F_523	F_524	F_525	F_526	F_527	F_528	F_529	F_530	F_531
0	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
1	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
2	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
3	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0
4	non-malicious	1	0	1	0	1	0	1	0	1	...	0	0	0	0	0	0	0	0	0	0

5 rows × 532 columns

Select items to perform actions on them.

Upload New

In [3]:
```
df.describe()
```

Out[3]:

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_10	...	F_522	F_523	F_524	F_525
count	373.000000	373.0	373.000000	373.000000	373.000000	373.0	373.000000	373.0	373.000000	373.0	...	373.00000	373.000000	373.000000	373.000000
mean	0.997319	0.0	0.994638	0.002681	0.994638	0.0	0.994638	0.0	0.994638	0.0	...	0.10992	0.107239	0.088472	0.099196
std	0.051778	0.0	0.073127	0.051778	0.073127	0.0	0.073127	0.0	0.073127	0.0	...	0.31321	0.309832	0.284361	0.299326
min	0.000000	0.0	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.0	...	0.00000	0.000000	0.000000	0.000000
25%	1.000000	0.0	1.000000	0.000000	1.000000	0.0	1.000000	0.0	1.000000	0.0	...	0.00000	0.000000	0.000000	0.000000
50%	1.000000	0.0	1.000000	0.000000	1.000000	0.0	1.000000	0.0	1.000000	0.0	...	0.00000	0.000000	0.000000	0.000000
75%	1.000000	0.0	1.000000	0.000000	1.000000	0.0	1.000000	0.0	1.000000	0.0	...	0.00000	0.000000	0.000000	0.000000
max	1.000000	0.0	1.000000	1.000000	1.000000	0.0	1.000000	0.0	1.000000	0.0	...	1.00000	1.000000	1.000000	1.000000

8 rows × 531 columns

Taking care of the null values

In [4]:
```
df = df.drop_duplicates(keep=False)
print(df.shape)
```

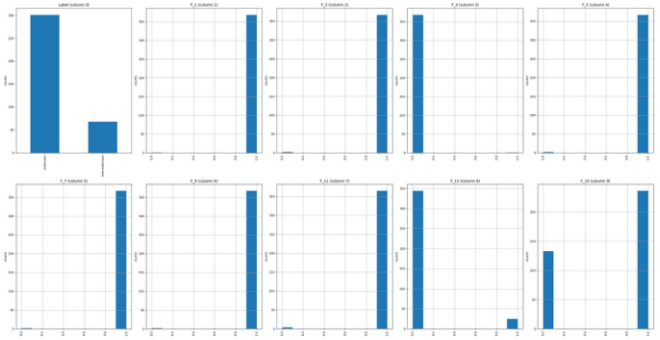
```
(369, 532)
```

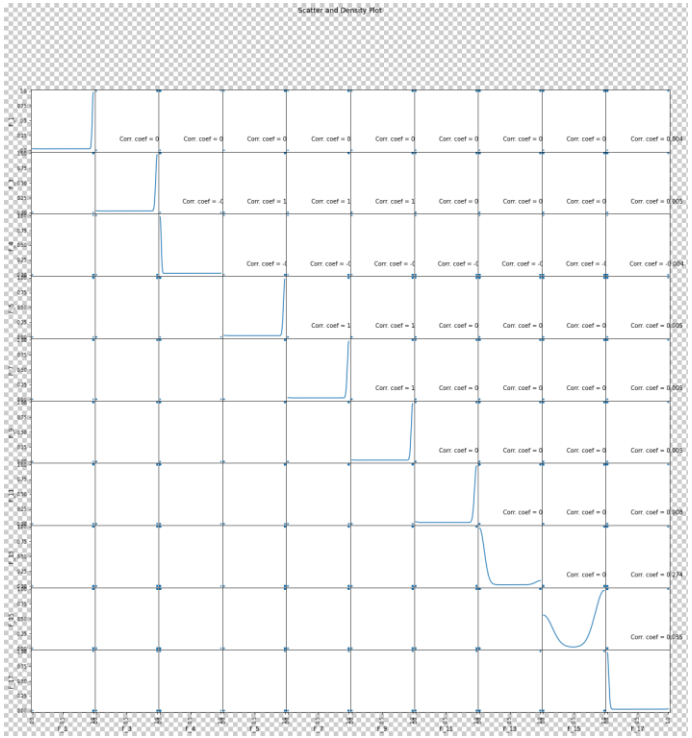
In [5]:
```
df.isnull().sum()
```

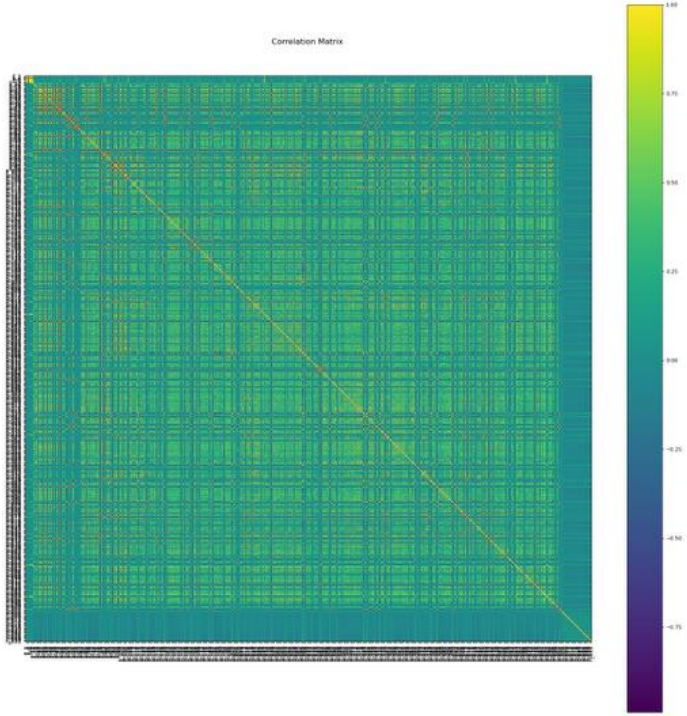
Out[5]:
```
Label      0
F_1        0
F_2        0
F_3        0
F_4        0
          ..
F_527      0
F_528      0
F_529      0
F_530      0
F_531      0
Length: 532, dtype: int64
```

Selecting the dependent and independent variable

In [6]:
```
X = df.drop(["Label"],axis=1)
y = df['Label'].values
```





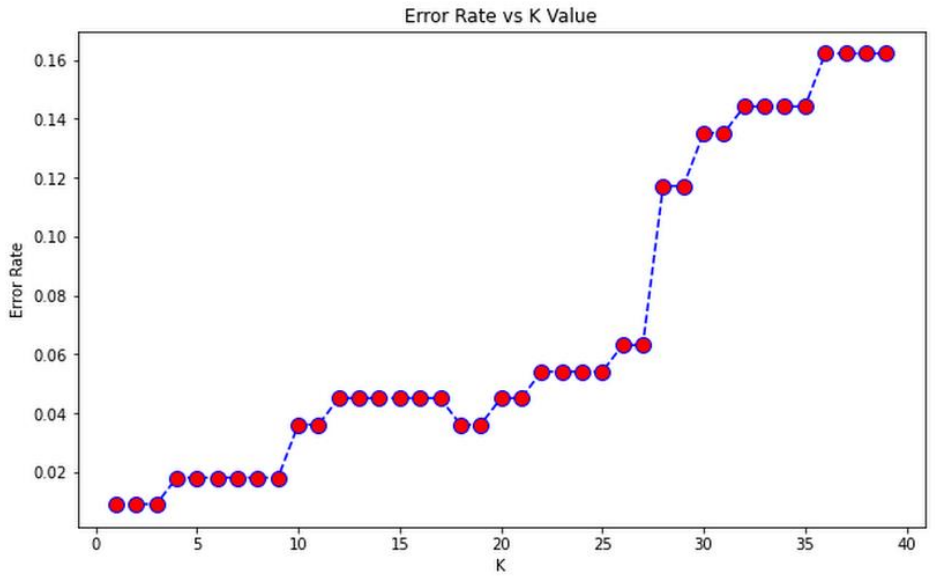


KNearest Neighbour Classfier

In [25]:
```
error_rate=[]
for i in range(1,40):
    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred=knn.predict(X_test)
    error_rate.append(np.mean(pred!=y_test))
```

In [26]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,linestyle='--',color='blue', marker='o',markerfacecolor='red',markersize=10)
plt.title('Error Rate vs K Value')
plt.xlabel('K Value')
plt.ylabel('Error Rate')
```

Out[26]: Text(0, 0.5, 'Error Rate')



Splitting the Data into Test and Train

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=10)
```

Random Forest Classifier

In [20]:
```
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

Out[20]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)

In [21]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

Out[21]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)

Results for KNN classifier

In [27]:
```
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH K=5')
print('\n')
print('\033[01m              Confusuion Matrix \033[0m')
print(confusion_matrix(y_test,pred))
print('\n')
print('\033[01m              KNearest Neighbour Classification_report \033[0m')
print(classification_report(y_test,pred))
print('\033[01m Accuracy\033[0m')
print(accuracy_score(y_test,pred))
cf_matrix = confusion_matrix(y_test, pred)
sns.heatmap(cf_matrix,cmap="Greens",annot = True)
plt.show()
```

```
WITH K=5


              Confusuion Matrix
[[89  1]
 [ 1 20]]


              KNearest Neighbour Classification_report
               precision    recall  f1-score   support

    malicious       0.99      0.99      0.99        90
non-malicious       0.95      0.95      0.95        21

    accuracy                            0.98       111
   macro avg       0.97      0.97      0.97       111
weighted avg       0.98      0.98      0.98       111

 Accuracy
0.9819819819819819
```

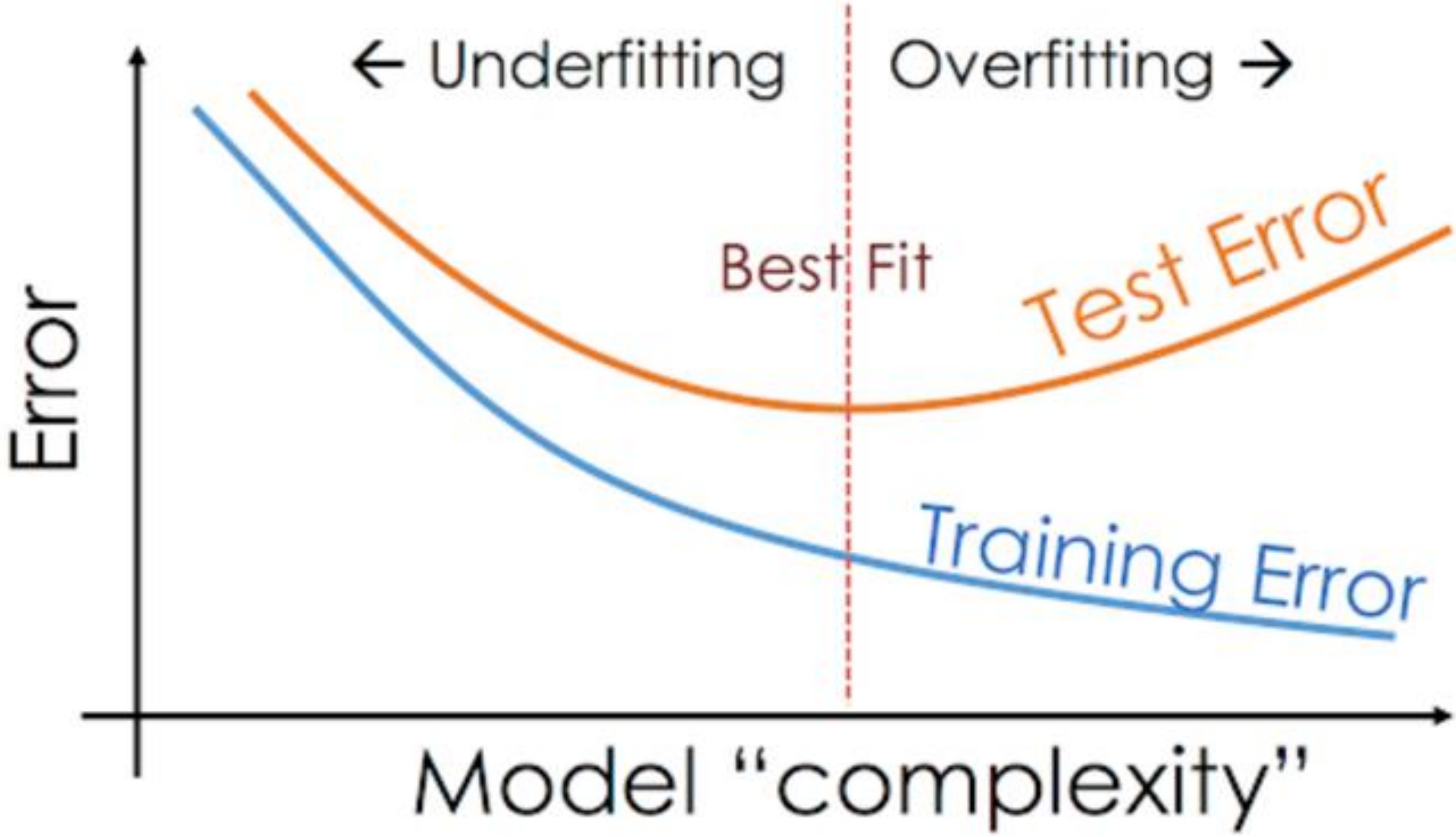
```
              Confusion Matrix
[[89  1]
 [ 0 21]]


              Random Forest  Classification_report
               precision    recall  f1-score   support

    malicious       1.00      0.99      0.99        90
non-malicious       0.95      1.00      0.98        21

    accuracy                            0.99       111
   macro avg       0.98      0.99      0.99       111
weighted avg       0.99      0.99      0.99       111

 Accuracy
0.990990990990991
```


← Underfitting
Overfitting →
Best Fit
Test Error
Training Error
Error
Model "complexity"


How to improve the performance of a machine learning (ML) model

WITH K=5

Confusuion Matrix

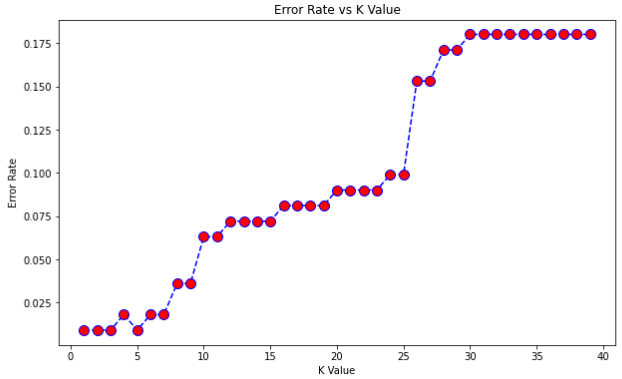
```
[[89  1]
 [ 0 21]]
```

KNearest Neighbour Classification_report

	precision	recall	f1-score	support
malicious	1.00	0.99	0.99	90
non-malicious	0.95	1.00	0.98	21
accuracy			0.99	111
macro avg	0.98	0.99	0.99	111
weighted avg	0.99	0.99	0.99	111

Accuracy

0.990990990990991



Feature Scaling(used for gradient descent and Distance driven modules like KNN and SVM)

In [20]:
```
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)
```

Thankyou