In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.metrics import f1_score, mean_squared_error
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df=pd.read_csv('C:\\Users\\HP\\Documents\\Data Science\\Python\\Projects\\drebin-215-dataset.csv')
```

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

Out[3]:

ATE	WRITE_EXTERNAL_STORAGE	ACCESS_FINE_LOCATION	SET_WALLPAPER_HINTS	SET_PREFERRED_APPLICATIONS	WRITE_SEC
0	1	0	0	0	
0	1	0	0	0	
0	0	0	0	0	
1	1	1	0	0	
1	0	1	0	0	

In [4]:
```
df.shape
```

Out[4]:
```
(15036, 216)
```

```
In [5]: df.columns
```

```
Out[5]: Index(['transact', 'onServiceConnected', 'bindService', 'attachInterface',
               'ServiceConnection', 'android.os.Binder', 'SEND_SMS',
               'Ljava.lang.Class.getCanonicalName', 'Ljava.lang.Class.getMethods',
               'Ljava.lang.Class.cast',
               ...
               'READ_CONTACTS', 'DEVICE_POWER', 'HARDWARE_TEST', 'ACCESS_WIFI_STATE',
               'WRITE_EXTERNAL_STORAGE', 'ACCESS_FINE_LOCATION', 'SET_WALLPAPER_HINTS',
               'SET_PREFERRED_APPLICATIONS', 'WRITE_SECURE_SETTINGS', 'class'],
              dtype='object', length=216)
```

```
In [6]: df.isnull().sum()
```

```
Out[6]: transact                      0
        onServiceConnected            0
        bindService                   0
        attachInterface               0
        ServiceConnection             0
                                     ..
        ACCESS_FINE_LOCATION          0
        SET_WALLPAPER_HINTS           0
        SET_PREFERRED_APPLICATIONS    0
        WRITE_SECURE_SETTINGS         0
        class                         0
        Length: 216, dtype: int64
```

```
In [7]: df.duplicated()
```

```
Out[7]: 0        False
        1        False
        2        False
        3        False
        4        False
                 ...
        15031     True
        15032    False
        15033    False
        15034     True
        15035     True
        Length: 15036, dtype: bool
```

```
In [8]:  df.drop_duplicates(inplace=True)
```

```
In [9]:  df['class'].unique()
```

```
Out[9]:  array(['S', 'B'], dtype=object)
```

```
In [10]:  label=LabelEncoder()
```

```
In [11]:  df['class']=label.fit_transform(df['class'])
```

```
In [12]:  df['class'].unique()
```

```
Out[12]:  array([1, 0])
```

```
In [13]:  x=df.drop(columns=['class'],axis=1)
          y=df['class']
```

```
In [14]:  x=x.replace('?',' ')
```

```
In [15]:  x=x.replace(' ',)
```

```
In [16]:  x.dropna(inplace=True)
```

In [17]:
```
x
```

Out[17]:

VIFI_STATE	WRITE_EXTERNAL_STORAGE	ACCESS_FINE_LOCATION	SET_WALLPAPER_HINTS	SET_PREFERRED_APPLICATIONS	WRI
0	1	0	0	0	
0	1	0	0	0	
0	0	0	0	0	
1	1	1	0	0	
1	0	1	0	0	
...	...	...	...	...	
0	1	0	0	0	
1	1	0	0	0	
1	1	1	0	0	
1	1	0	0	0	
1	1	0	0	0	

In [18]:
```
y
```

Out[18]:
```
0        1
1        1
2        1
3        1
4        1
        ..
15026    0
15028    0
15029    0
15032    0
15033    0
Name: class, Length: 8171, dtype: int32
```

```
In [19]:  pca=PCA(n_components=10)
          x_pca=pca.fit_transform(x)

In [20]:  x_train,x_test,y_train,y_test=train_test_split(x_pca,y,test_size=0.2, random_state=42)

In [21]:  dtc=DecisionTreeClassifier()

In [22]:  dtc.fit(x_train,y_train)

Out[22]:  ▾ DecisionTreeClassifier

          DecisionTreeClassifier()

In [23]:  dtc_pred=dtc.predict(x_test)

In [24]:  dtc_pred[: ]

Out[24]:  array([0, 0, 0, ..., 0, 0, 0])

In [25]:  dtc_accuracy=accuracy_score(y_test, dtc_pred)

In [26]:  dtc_accuracy

Out[26]:  0.9259938837920489

In [27]:  f1_score(y_test, dtc_pred)

Out[27]:  0.8263988522238164

In [28]:  dtc_confusion_matrix=confusion_matrix(y_test, dtc_pred)

In [29]:  dtc_confusion_matrix

Out[29]:  array([[1226,   45],
                 [  76,  288]], dtype=int64)
```

```
In [30]:  TP=dtc_confusion_matrix[1][1]

In [31]:  TP

Out[31]:  288

In [32]:  FP=dtc_confusion_matrix[0][1]

In [33]:  FP

Out[33]:  45

In [34]:  dtc_classification_report=classification_report(y_test,dtc_pred)

In [35]:  print(dtc_classification_report)

                        precision    recall  f1-score   support

                   0       0.94      0.96      0.95      1271
                   1       0.86      0.79      0.83       364

            accuracy                           0.93      1635
           macro avg       0.90      0.88      0.89      1635
        weighted avg       0.92      0.93      0.92      1635


In [36]:  dtc_mse=mean_squared_error(y_test,dtc_pred)

In [37]:  dtc_mse

Out[37]:  0.07400611620795107

In [38]:  rfc=RandomForestClassifier()
In [39]: rfc.fit(x_train,y_train)
```

```
Out[39]: ▾ RandomForestClassifier
         RandomForestClassifier()
```

```
In [40]: rfc_pred=rfc.predict(x_test)
```

```
In [41]: rfc_pred[: ]
```

```
Out[41]: array([0, 0, 0, ..., 0, 0, 0])
```

```
In [42]: rfc_accuracy=accuracy_score(y_test, rfc_pred)
```

```
In [43]: rfc_accuracy
```

```
Out[43]: 0.9559633027522936
```

```
In [44]: f1_score(y_test, rfc_pred)
```

```
Out[44]: 0.8950437317784256
```

```
In [45]: rfc_confusion_matrix=confusion_matrix(y_test, rfc_pred)
```

```
In [46]: rfc_confusion_matrix
```

```
Out[46]: array([[1256,   15],
                [  57,  307]], dtype=int64)
```

```
In [47]: TP=rfc_confusion_matrix[1][1]
```

```
In [48]: TP
```

```
Out[48]: 307
```

In [49]:
```
FP=rfc_confusion_matrix[0][1]
```

In [50]:
```
FP
```

Out[50]:
```
15
```

In [51]:
```
rfc_classification_report=classification_report(y_test,rfc_pred)
```

In [52]:
```
print(rfc_classification_report)
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      1271
           1       0.95      0.84      0.90       364

    accuracy                           0.96      1635
   macro avg       0.96      0.92      0.93      1635
weighted avg       0.96      0.96      0.95      1635
```

In [53]:
```
rfc_mse=mean_squared_error(y_test,rfc_pred)
```

In [54]:
```
rfc_mse
```

Out[54]:
```
0.044036697247706424
```

In [55]:
```
svm=SVC(random_state=42)
```

In [56]:
```
svm.fit(x_train,y_train)
```

Out[56]:
```
▾         SVC
SVC(random_state=42)
```

In [57]:
```
svm_pred=svm.predict(x_test)
```

```
In [58]:  svm_pred[: ]

Out[58]: array([0, 0, 0, ..., 0, 0, 0])

In [59]:  svm_accuracy=accuracy_score(y_test, svm_pred)

In [60]:  svm_accuracy

Out[60]: 0.9522935779816514

In [61]:  f1_score(y_test, svm_pred)

Out[61]: 0.8862973760932944

In [62]:  svm_confusion_matrix=confusion_matrix(y_test, svm_pred)

In [63]:  svm_confusion_matrix

Out[63]: array([[1253,   18],
                [  60,  304]], dtype=int64)

In [64]:  TP=svm_confusion_matrix[1][1]

In [65]:  TP

Out[65]: 304

In [66]:  FP=svm_confusion_matrix[0][1]

In [67]:  FP

Out[67]: 18

In [68]:  svm_classification_report=classification_report(y_test,svm_pred)
In [69]: print(svm_classification_report)

              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1271
           1       0.94      0.84      0.89       364

    accuracy                           0.95      1635
   macro avg       0.95      0.91      0.93      1635
weighted avg       0.95      0.95      0.95      1635
```

```
In [70]: svm_mse=mean_squared_error(y_test,svm_pred)
```

```
In [71]: svm_mse

Out[71]: 0.047706422018348627
```