Evasion Aware Android Malware Detection System based on Optimised KNN Model

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
Start coding or generate with AI.
```

Data Processing and Model Building

Data Processing

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.exceptions import NotFittedError
from sklearn.pipeline import Pipeline
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
```

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/CombineFeatures_PowerDroid.csv", index_col=False)
```

```
df.shape
```

```
(35999, 1434)
```

```
df.head()
```

	openInputStream	getCellLocation	getAddress	start	sendBroadcast	setContentView	setName	getScanResults	query	getI
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	1	0	1	0	0	1	
2	1	0	0	1	1	1	0	0	1	
3	0	0	0	1	0	1	0	0	1	
4	0	0	0	1	1	1	0	0	1	

5 rows × 1434 columns

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
X.shape
```

```
(35999, 1433)
```

```
train=df.sample(frac=0.8,random_state=200) #random state is a seed value
test=df.drop(train.index)
```

```
train.to_csv('/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/train.csv',index=False)
```

```
test.to_csv('/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/test.csv',index=False)
```

```
train[train > 0] = 1
test[test > 0] = 1

X_train[X_train > 0] = 1
X_test[X_test > 0] = 1
```

```
X_test
```

	openInputStream	getCellLocation	getAddress	start	sendBroadcast	setContentView	setName	getScanResults	query
23872	1	0	1	1	0	1	1	0	1
9088	0	1	0	1	1	1	0	1	1
31766	0	0	0	1	0	1	0	0	0
9662	1	0	1	1	1	1	1	0	1
2281	0	0	0	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...
33243	1	0	0	1	0	1	0	0	1
19507	0	0	0	1	0	1	0	0	0
20909	0	0	0	0	0	1	0	0	0
18462	0	0	0	1	0	1	1	0	0
22220	0	0	1	1	1	1	1	0	0

7200 rows × 1433 columns

```
X_train
```

	openInputStream	getCellLocation	getAddress	start	sendBroadcast	setContentView	setName	getScanResults	query
20583	1	0	0	1	1	1	1	0	1
16529	0	0	0	1	0	0	0	0	0
7088	0	1	0	1	1	1	0	0	1
7182	0	0	0	1	1	1	0	0	1
6668	0	0	0	1	1	1	0	0	0
...	...	...	...	...	...	...	...	...	...
13154	0	0	0	1	1	1	0	0	0
7522	0	0	0	1	1	1	0	0	0
16342	1	0	0	1	1	1	0	0	1
2097	0	0	0	1	1	1	0	0	0
33655	0	0	0	1	0	1	0	0	0

28799 rows × 1433 columns

```
X_test.to_csv('/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/mainTestData-X_savedroid.csv', index=False)
```

```
y_test.to_csv('/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/mainTestData-Y_savedroid.csv', index=False)
```

```
Xn_train=train
Xn_test=test
yn_train=train
yn_test=test
```

Default Models Traings

```
svm_full_model = SVC(kernel='rbf', C=1.0, gamma='scale')
rf_full_model = RandomForestClassifier(n_estimators=100, max_depth=None, random_state=200)
dt_full_model = DecisionTreeClassifier(criterion='gini', max_depth=None, random_state=200)
```

```
nb_full_model = GaussianNB()
ada_full_model = AdaBoostClassifier(n_estimators=50, learning_rate=1.0, random_state=200)
knn_full_model = KNeighborsClassifier(n_neighbors=80, algorithm='auto')
#lr_full_model = LogisticRegression(penalty='l2', C=1.0, solver='lbfgs', max_iter=1000, random_state=200)
lr_full_model = Pipeline([
    ('scaler', StandardScaler()),
    ('logistic', LogisticRegression(solver='lbfgs', max_iter=3000))
])
```

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

```
SVC
SVC()
```

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

```
RandomForestClassifier
RandomForestClassifier(random_state=200)
```

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

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=200)
```

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

```
GaussianNB
GaussianNB()
```

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

```
AdaBoostClassifier
AdaBoostClassifier(random_state=200)
```

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

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=80)
```

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

```
Pipeline
StandardScaler
LogisticRegression
```

Default Model Testing

```
y_pred_svm_full_model = svm_full_model.predict(X_test)
```

```
y_pred_rf_full_model = rf_full_model.predict(X_test)
```

```
y_pred_dt_full_model = dt_full_model.predict(X_test)
```

```
y_pred_nb_full_model = nb_full_model.predict(X_test)
```

```
y_pred_ada_full_model = ada_full_model.predict(X_test)
```

```
y_pred_knn_full_model = knn_full_model.predict(X_test)
```

```
y_pred_lr_full_model = lr_full_model.predict(X_test)
```

```
metrics = {
    'Model': ['SVM', 'Random Forest', 'Decision Tree', 'Naive Bayes', 'AdaBoost', 'KNN', 'Logistic Regression']
```

```
    'Model': ['SVM', 'Random Forest', 'Decision Tree', 'Naive Bayes', 'AdaBoost', 'KNN', 'Logistic Regression'],
    'Accuracy': [
        round(accuracy_score(y_test, y_pred_svm_full_model), 3)*100,
        round(accuracy_score(y_test, y_pred_rf_full_model), 3)*100,
        round(accuracy_score(y_test, y_pred_dt_full_model), 3)*100,
        round(accuracy_score(y_test, y_pred_nb_full_model), 3)*100,
        round(accuracy_score(y_test, y_pred_ada_full_model), 3)*100,
        round(accuracy_score(y_test, y_pred_knn_full_model), 3)*100,
        round(accuracy_score(y_test, y_pred_lr_full_model), 3)*100
    ],
    'Precision': [
        round(precision_score(y_test, y_pred_svm_full_model, average='binary'), 3)*100,
        round(precision_score(y_test, y_pred_rf_full_model, average='binary'), 3)*100,
        round(precision_score(y_test, y_pred_dt_full_model, average='binary'), 3)*100,
        round(precision_score(y_test, y_pred_nb_full_model, average='binary'), 3)*100,
        round(precision_score(y_test, y_pred_ada_full_model, average='binary'), 3)*100,
        round(precision_score(y_test, y_pred_knn_full_model, average='binary'), 3)*100,
        round(precision_score(y_test, y_pred_lr_full_model, average='binary'), 3)*100
    ],
    'Recall': [
        round(recall_score(y_test, y_pred_svm_full_model, average='binary'), 3)*100,
        round(recall_score(y_test, y_pred_rf_full_model, average='binary'), 3)*100,
        round(recall_score(y_test, y_pred_dt_full_model, average='binary'), 3)*100,
        round(recall_score(y_test, y_pred_nb_full_model, average='binary'), 3)*100,
        round(recall_score(y_test, y_pred_ada_full_model, average='binary'), 3)*100,
        round(recall_score(y_test, y_pred_knn_full_model, average='binary'), 3)*100,
        round(recall_score(y_test, y_pred_lr_full_model, average='binary'), 3)*100
    ],
    'F-Measure': [
        round(f1_score(y_test, y_pred_svm_full_model, average='binary'), 3)*100,
        round(f1_score(y_test, y_pred_rf_full_model, average='binary'), 3)*100,
        round(f1_score(y_test, y_pred_dt_full_model, average='binary'), 3)*100,
        round(f1_score(y_test, y_pred_nb_full_model, average='binary'), 3)*100,
        round(f1_score(y_test, y_pred_ada_full_model, average='binary'), 3)*100,
        round(f1_score(y_test, y_pred_knn_full_model, average='binary'), 3)*100,
        round(f1_score(y_test, y_pred_lr_full_model, average='binary'), 3)*100
    ],
    'Confusion Matrix': [
        confusion_matrix(y_test, y_pred_svm_full_model.round()),
        confusion_matrix(y_test, y_pred_rf_full_model.round()),
        confusion_matrix(y_test, y_pred_dt_full_model.round()),
        confusion_matrix(y_test, y_pred_nb_full_model.round()),
        confusion_matrix(y_test, y_pred_ada_full_model.round()),
        confusion_matrix(y_test, y_pred_knn_full_model.round()),
        confusion_matrix(y_test, y_pred_lr_full_model.round())
    ]
}

# Create DataFrame for the main metrics
df_metrics = pd.DataFrame(metrics, columns=['Model', 'Accuracy', 'Precision', 'Recall', 'F-Measure'])

print(df_metrics)
print("\n>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>")
print("\t\t\tConfusion Matrices")
print(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> \n")

for model, cm in zip(metrics['Model'], metrics['Confusion Matrix']):
    print(f"{model} Confusion Matrix:")
    print(f"{cm}\n")
```

```
                 Model  Accuracy  Precision  Recall  F-Measure
0                  SVM      99.3       99.3    99.3       99.3
1        Random Forest      99.4       99.4    99.4       99.4
2        Decision Tree      98.3       98.3    98.4       98.3
3          Naive Bayes      66.0       59.7    99.1       74.5
4             AdaBoost      98.0       98.1    97.9       98.0
5                  KNN      97.8       98.4    97.3       97.8
6  Logistic Regression      98.9       98.8    99.1       98.9

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                        Confusion Matrices
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

SVM Confusion Matrix:
[[3564   24]
 [  27 3585]]

Random Forest Confusion Matrix:
[[3565   23]
 [  20 3592]]

Decision Tree Confusion Matrix:
[[3525   63]
 [  57 3555]]
```

```
Naive Bayes Confusion Matrix:
[[1171 2417]
 [  34 3578]]

AdaBoost Confusion Matrix:
[[3518   70]
 [  76 3536]]

KNN Confusion Matrix:
[[3531   57]
 [  98 3514]]

Logistic Regression Confusion Matrix:
[[3546   42]
 [  34 3578]]
```

Optimised KNN

```
knn_optimised_model = KNeighborsClassifier(n_neighbors=3, metric='manhattan', weights='distance', algorithm='auto')
```

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

```
KNeighborsClassifier
KNeighborsClassifier(metric='manhattan', n_neighbors=3, weights='distance')
```

```
y_pred_knn_optimised_model = knn_optimised_model.predict(X_test)
```

```
# Calculate the metrics
accuracy = round(accuracy_score(y_test, y_pred_knn_optimised_model), 3) * 100
precision = round(precision_score(y_test, y_pred_knn_optimised_model, average='binary'), 3) * 100
recall = round(recall_score(y_test, y_pred_knn_optimised_model, average='binary'), 3) * 100
f_measure = round(f1_score(y_test, y_pred_knn_optimised_model, average='binary'), 3) * 100
conf_matrix = confusion_matrix(y_test, y_pred_knn_optimised_model)

# Create a dictionary for the metrics without confusion matrix for the DataFrame
metrics_optimised = {
    'Metric': ['Accuracy', 'Precision', 'Recall', 'F-Measure'],
    'Value': [accuracy, precision, recall, f_measure]
}

# Create DataFrame for the main metrics
df_metrics_optimised = pd.DataFrame(metrics_optimised)

print(df_metrics_optimised)

print("\n>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>")
print("\t\t\tConfusion Matrices_optimised_knn")
print(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> \n")

print("Confusion Matrix for KNN Optimised Model:")
print(conf_matrix)
```

```
      Metric  Value
0   Accuracy   99.3
1  Precision   99.1
2     Recall   99.6
3  F-Measure   99.3

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                        Confusion Matrices_optimised_knn
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Confusion Matrix for KNN Optimised Model:
[[3555   33]
 [  15 3597]]
```

Exporting the Optimised Model as a Pickle file

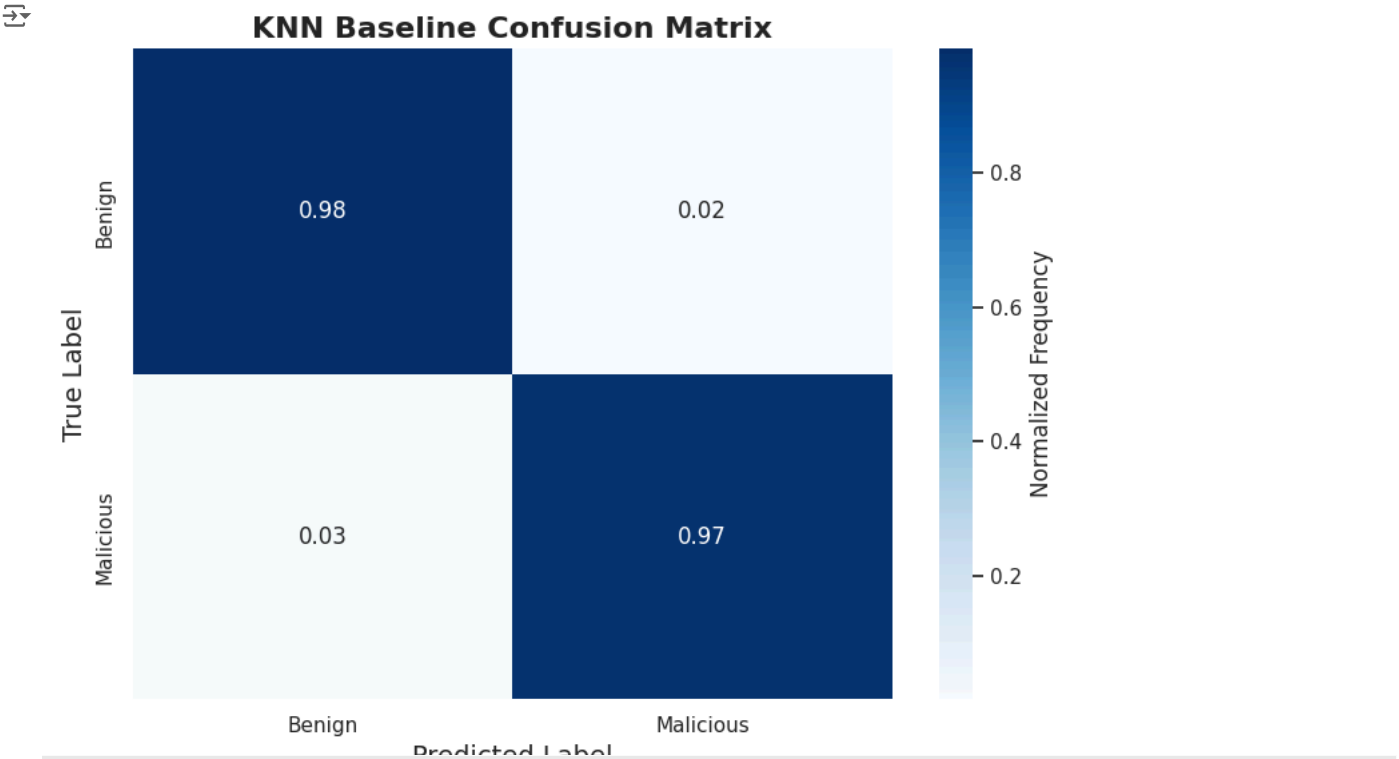
```
import pickle
```

```
with open('savedroid.pkl', 'wb') as file:
         pickle.dump(knn_optimised_model, file)
```

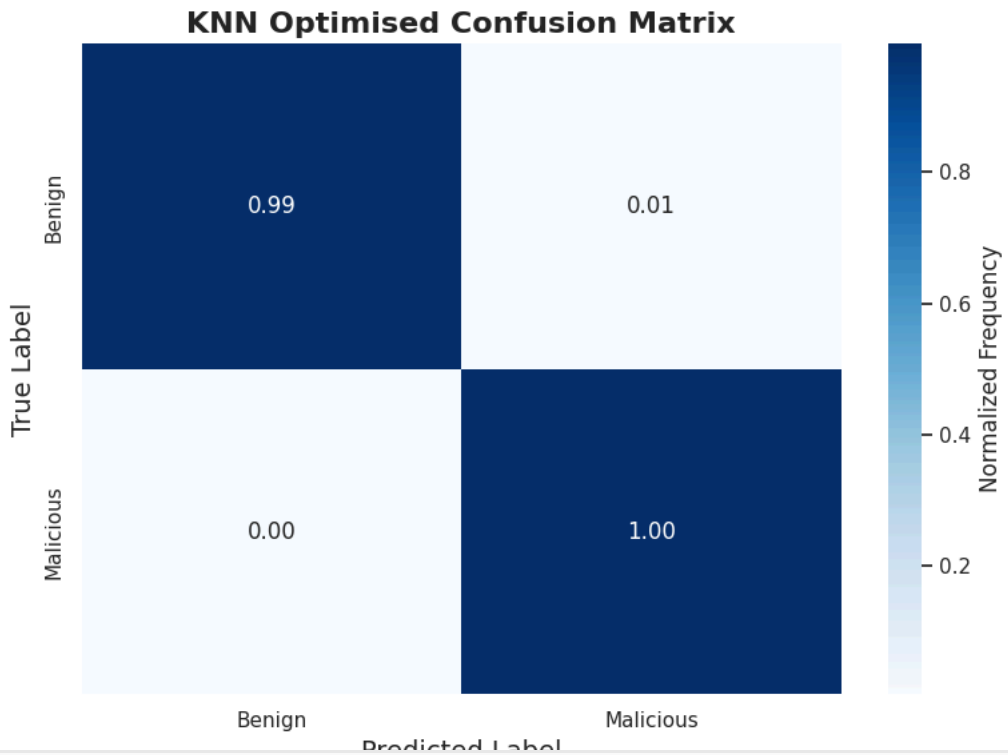
Comparing Baseline and Optimised KNN Confusion Matrix

```
# Function to generate and display a detailed confusion matrix
def plot_confusion_matrix(y_true, y_pred, classes, title):
    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]  # Normalize

    # Set up the plot
    plt.figure(figsize=(8, 6))
    sns.set(style="white")  # White background for a clean look
    sns.heatmap(cm_normalized, annot=True, fmt='.2f', cmap='Blues',
                xticklabels=classes, yticklabels=classes, cbar_kws={'label': 'Normalized Frequency'})

    # Titles and labels with IEEE style considerations
    plt.title(title, fontsize=16, fontweight='bold')
    plt.xlabel('Predicted Label', fontsize=14)
    plt.ylabel('True Label', fontsize=14)

    # Fine-tune layout for publication quality
    plt.tight_layout()
    plt.show()
```

```
plot_confusion_matrix(y_test, y_pred_knn_full_model, ['Benign', 'Malicious'], 'KNN Baseline Confusion Matrix')
```




```
plot_confusion_matrix(y_test, y_pred_knn_optimised_model, ['Benign', 'Malicious'], 'KNN Optimised Confusion Matrix')
```



Performing Advarsarial Attack - Feature Injection (FI) With Discriminating Features

```
# Getting the top 35 features based on the frequency in to a list
Top_features = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features.append(w)

Top_features1 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features1.append(w)

Top_features2 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features2.append(w)

Top_features3 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features3.append(w)

Top_features4 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features4.append(w)

Top_features5 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features5.append(w)

Top_features6 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features6.append(w)

Top_features_optimised1 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
```

```
    for feature in f:
        w = feature.strip('\n')
        Top_features_optimised1.append(w)
```

```
print(Top_features1)
```

```
['openInputStream', 'startListening', 'isWiredHeadsetOn', 'stopListening', 'setMode', 'addMessage', 'getPassword', 'getA
```

```
Adv_Test=Xn_test.copy()
Adv_Test1=Xn_test.copy()
Adv_Test2=Xn_test.copy()
Adv_Test3=Xn_test.copy()
Adv_Test4=Xn_test.copy()
Adv_Test5=Xn_test.copy()
Adv_Test6=Xn_test.copy()

Adv_Test_optimised1=Xn_test.copy()
```

```
Top_features = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/Discriminating features.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features.append(w)

Adv_Test=Xn_test.copy()

Er_List_adv_svm = []
for feature in Top_features:
    col2 = feature
    Adv_Test.loc[Adv_Test.Label == 1, col2] = 1
    Adv_x_test = Adv_Test.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test['Label']
    Adv_y_pred_Full_Model = svm_full_model.predict(Adv_x_test)
    #-----------------------------------------
    precision_adv_svm = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall_adv_svm = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score_adv_svm = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con_adv_svm = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-----------------------------------------------
    er = con_adv_svm[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_svm.append(er)
    #-----------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision_adv_svm)
    print('Recall: %.3f' % recall_adv_svm)
    print('F-Measure: %.3f' % score_adv_svm)
    print('CM', con_adv_svm)
    print('-------------------------------------------------')
    print('EVASION RATE_svm: ', er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

```
Accuracy     : 0.9928
Precision: 0.995
Recall: 0.990
F-Measure: 0.993
CM [[3570   17]
 [  35 3578]]
-------------------------------------------------
EVASION RATE_svm:  0.01
-------------------------------------------------
*************************************************************************
Accuracy     : 0.9904
Precision: 0.995
Recall: 0.986
F-Measure: 0.990
CM [[3570   17]
 [  52 3561]]
-------------------------------------------------
EVASION RATE_svm:  0.01
-------------------------------------------------
*************************************************************************
Accuracy     : 0.9894
Precision: 0.995
Recall: 0.984
F-Measure: 0.989
CM [[3570   17]
 [  59 3554]]
-------------------------------------------------
EVASION RATE_svm:  0.02
```

```
--------------------------------------------------
**********************************************************************
Accuracy     : 0.989
Precision: 0.995
Recall: 0.983
F-Measure: 0.989
CM [[3570   17]
 [  62 3551]]
--------------------------------------------------
EVASION RATE_svm:  0.02
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9865
Precision: 0.995
Recall: 0.978
F-Measure: 0.986
CM [[3570   17]
 [  80 3533]]
--------------------------------------------------
EVASION RATE_svm:  0.02
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9765
Precision: 0.995
Recall: 0.958
F-Measure: 0.976
CM [[3570   17]
 [ 152 3461]]
--------------------------------------------------
EVASION RATE_svm:  0.04
```

```
Adv_Test.shape
```

```
(7200, 1434)
```

```
Adv_Test.shape
```

```
(7200, 1434)
```

SVM_fi

```
Er_List_adv_svm = []
for feature in Top_features:
    col2 = feature
    Adv_Test.loc[Adv_Test.Label == 1, col2] = 1
    Adv_x_test = Adv_Test.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test['Label']
    Adv_y_pred_Full_Model = svm_full_model.predict(Adv_x_test)
    #-----------------------------------------------
    precision_adv_svm = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall_adv_svm = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score_adv_svm = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con_adv_svm = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er = con_adv_svm[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_svm.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision_adv_svm)
    print('Recall: %.3f' % recall_adv_svm)
    print('F-Measure: %.3f' % score_adv_svm)
    print('CM', con_adv_svm)
    print('--------------------------------------------------')
    print('EVASION RATE_svm: ', er)
    print('--------------------------------------------------')
    print("**********************************************************************")
```

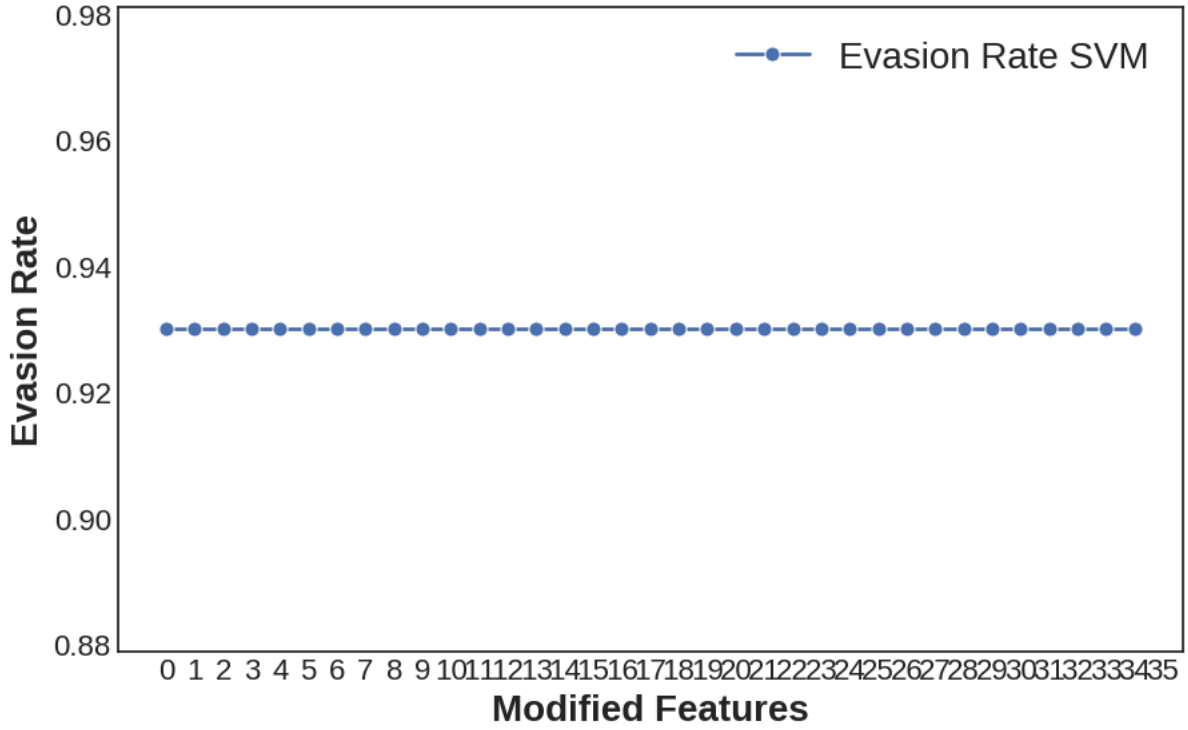
```
Accuracy     : 0.5317
Precision: 0.938
Recall: 0.071
F-Measure: 0.133
CM [[3570   17]
 [3355  258]]
--------------------------------------------------
EVASION RATE_svm:  0.93
--------------------------------------------------
**********************************************************************
Accuracy     : 0.5317
Precision: 0.938
Recall: 0.071
F-Measure: 0.133
CM [[3570   17]
 [3355  258]]
```

```
--------------------------------------------------
EVASION RATE_svm:  0.93
--------------------------------------------------
********************************************************************
Accuracy     : 0.5317
Precision: 0.938
Recall: 0.071
F-Measure: 0.133
CM [[3570   17]
 [3355  258]]
--------------------------------------------------
EVASION RATE_svm:  0.93
--------------------------------------------------
********************************************************************
Accuracy     : 0.5317
Precision: 0.938
Recall: 0.071
F-Measure: 0.133
CM [[3570   17]
 [3355  258]]
--------------------------------------------------
EVASION RATE_svm:  0.93
--------------------------------------------------
********************************************************************
Accuracy     : 0.5317
Precision: 0.938
Recall: 0.071
F-Measure: 0.133
CM [[3570   17]
 [3355  258]]
--------------------------------------------------
EVASION RATE_svm:  0.93
--------------------------------------------------
********************************************************************
Accuracy     : 0.5317
Precision: 0.938
Recall: 0.071
F-Measure: 0.133
CM [[3570   17]
 [3355  258]]
--------------------------------------------------
EVASION RATE_svm:  0.93
```

```
Less_ER=Er_List_adv_svm[0:35]
```

```
df = pd.DataFrame (Less_ER,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate SVM"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-126-1fe23fe2f53b>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



RF_fi

```
Er_List_adv_rf = []
for feature in Top_features1:
    col2 = feature
    Adv_Test1.loc[Adv_Test1.Label == 1, col2] = 1
    Adv_x_test = Adv_Test1.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test1['Label']
    Adv_y_pred_Full_Model = rf_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision_adv_rf = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall_adv_rf = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score_adv_rf = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con_adv_rf = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er = con_adv_rf[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_rf.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision_adv_rf)
    print('Recall: %.3f' % recall_adv_rf)
    print('F-Measure: %.3f' % score_adv_rf)
    print('CM', con_adv_rf)
    print('-------------------------------------------------')
    print('EVASION RATE_rf: ', er)
    print('-------------------------------------------------')
    print("**********************************************************************")
```

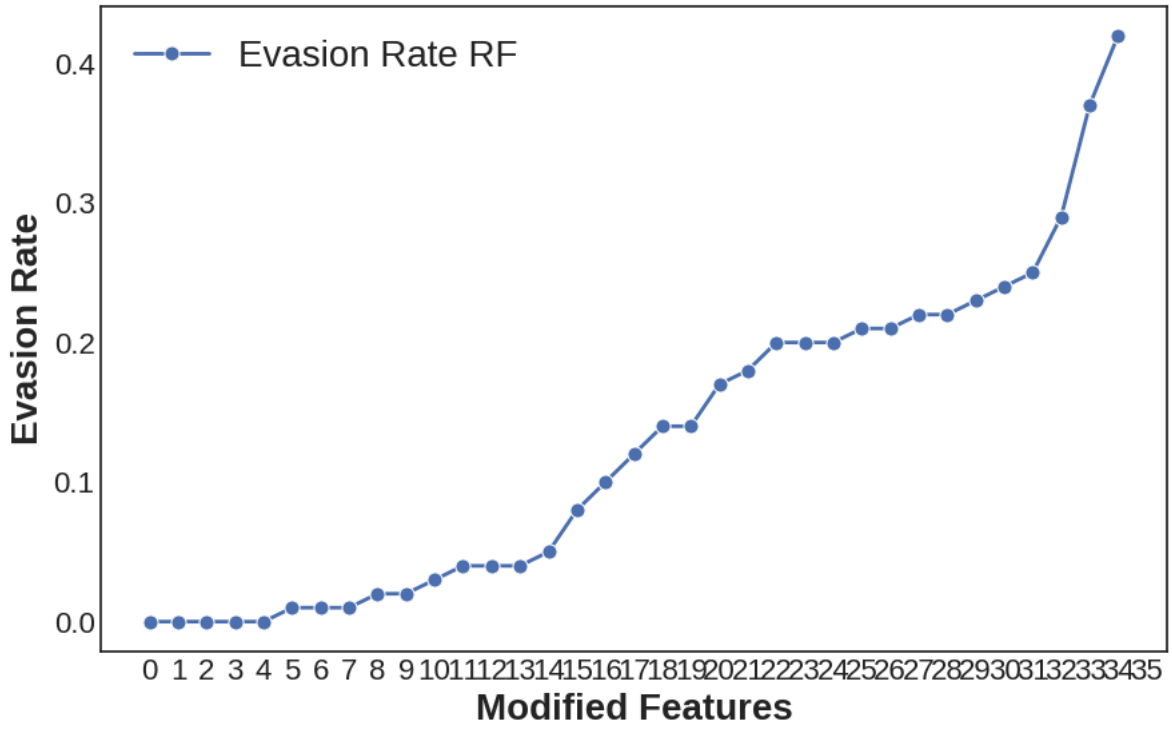
```
Accuracy     : 0.9979
Precision: 0.998
Recall: 0.998
F-Measure: 0.998
CM [[3581    6]
 [   9 3604]]
-------------------------------------------------
EVASION RATE_rf:  0.0
-------------------------------------------------
**********************************************************************
Accuracy     : 0.9976
Precision: 0.998
Recall: 0.997
F-Measure: 0.998
CM [[3581    6]
 [  11 3602]]
-------------------------------------------------
EVASION RATE_rf:  0.0
-------------------------------------------------
```

```
**************************************************************************
Accuracy     : 0.9976
Precision: 0.998
Recall: 0.997
F-Measure: 0.998
CM [[3581    6]
 [  11 3602]]
--------------------------------------------------
EVASION RATE_rf:  0.0
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9976
Precision: 0.998
Recall: 0.997
F-Measure: 0.998
CM [[3581    6]
 [  11 3602]]
--------------------------------------------------
EVASION RATE_rf:  0.0
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9978
Precision: 0.998
Recall: 0.997
F-Measure: 0.998
CM [[3581    6]
 [  10 3603]]
--------------------------------------------------
EVASION RATE_rf:  0.0
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9961
Precision: 0.998
Recall: 0.994
F-Measure: 0.996
CM [[3581    6]
 [  22 3591]]
--------------------------------------------------
EVASION RATE_rf:  0.01
```

```
Less_ER1=Er_List_adv_rf[0:35]
```

```
df = pd.DataFrame (Less_ER1,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate RF"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-129-f57280458bdd>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



DT_fi

```
Er_List_adv_dt = []
for feature in Top_features2:
    col2 = feature
    Adv_Test2.loc[Adv_Test2.Label == 1, col2] = 1
    Adv_x_test = Adv_Test2.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test2['Label']
    Adv_y_pred_Full_Model = dt_full_model.predict(Adv_x_test)
    #-----------------------------------------------
    precision_adv_dt = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall_adv_dt = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score_adv_dt = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con_adv_dt = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-----------------------------------------------------
    er = con_adv_dt[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_dt.append(er)
    #-----------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision_adv_dt)
    print('Recall: %.3f' % recall_adv_dt)
    print('F-Measure: %.3f' % score_adv_dt)
    print('CM', con_adv_dt)
    print('-----------------------------------------------------')
    print('EVASION RATE_dt: ', er)
    print('-----------------------------------------------------')
    print("*************************************************************************")
```

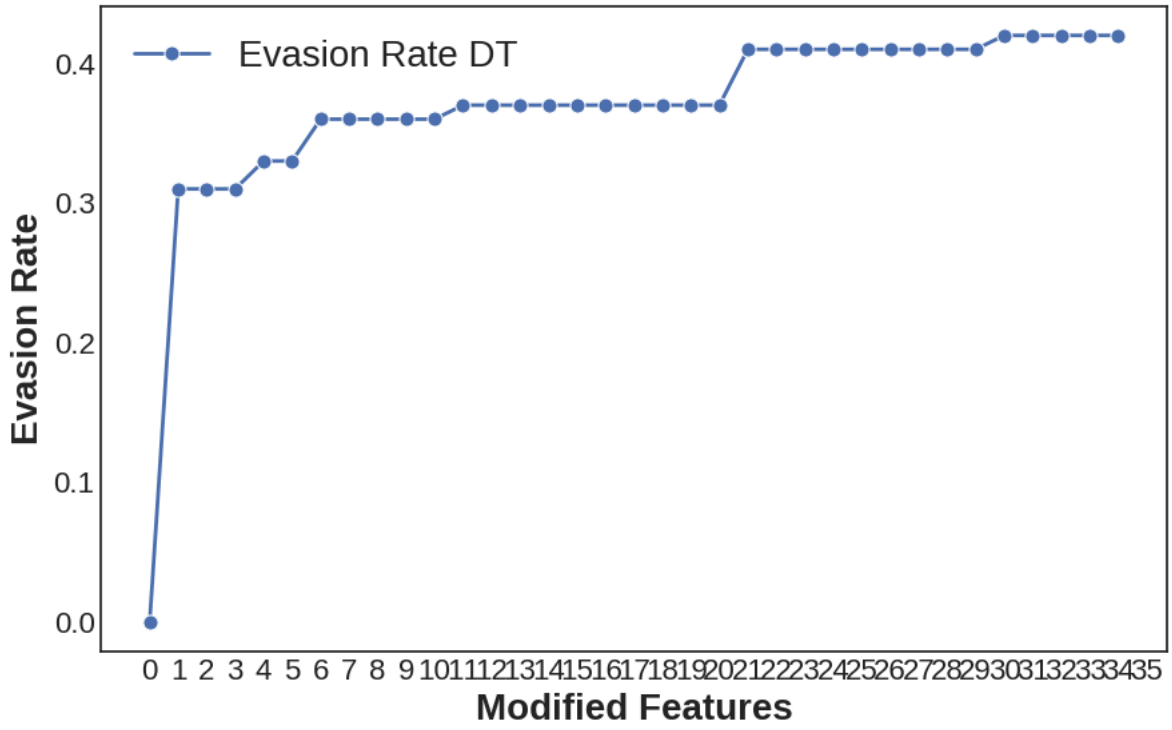
```
Accuracy     : 0.9969
Precision: 0.997
Recall: 0.997
F-Measure: 0.997
CM [[3577   10]
 [  12 3601]]
-----------------------------------------------------
EVASION RATE_dt:  0.0
-----------------------------------------------------
*************************************************************************
Accuracy     : 0.8444
Precision: 0.996
Recall: 0.693
F-Measure: 0.817
CM [[3577   10]
 [1110 2503]]
-----------------------------------------------------
EVASION RATE_dt:  0.31
-----------------------------------------------------
```

```
**************************************************************************
Accuracy     : 0.8444
Precision: 0.996
Recall: 0.693
F-Measure: 0.817
CM [[3577   10]
 [1110 2503]]
--------------------------------------------------
EVASION RATE_dt:  0.31
--------------------------------------------------
**************************************************************************
Accuracy     : 0.8444
Precision: 0.996
Recall: 0.693
F-Measure: 0.817
CM [[3577   10]
 [1110 2503]]
--------------------------------------------------
EVASION RATE_dt:  0.31
--------------------------------------------------
**************************************************************************
Accuracy     : 0.8306
Precision: 0.996
Recall: 0.665
F-Measure: 0.798
CM [[3577   10]
 [1210 2403]]
--------------------------------------------------
EVASION RATE_dt:  0.33
--------------------------------------------------
**************************************************************************
Accuracy     : 0.8306
Precision: 0.996
Recall: 0.665
F-Measure: 0.798
CM [[3577   10]
 [1210 2403]]
--------------------------------------------------
EVASION RATE dt:  0.33
```

```
Less_ER2=Er_List_adv_dt[0:35]
```

```
df = pd.DataFrame (Less_ER2,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate DT"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-132-13ea2e8f7696>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



NB_fi

```
Er_List_adv_nb = []
for feature in Top_features3:
    col2 = feature
    Adv_Test3.loc[Adv_Test3.Label == 1, col2] = 1
    Adv_x_test = Adv_Test3.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test3['Label']
    Adv_y_pred_Full_Model = nb_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er = con[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_nb.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM', con)
    print('-------------------------------------------------')
    print('EVASION RATE_nb: ', er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

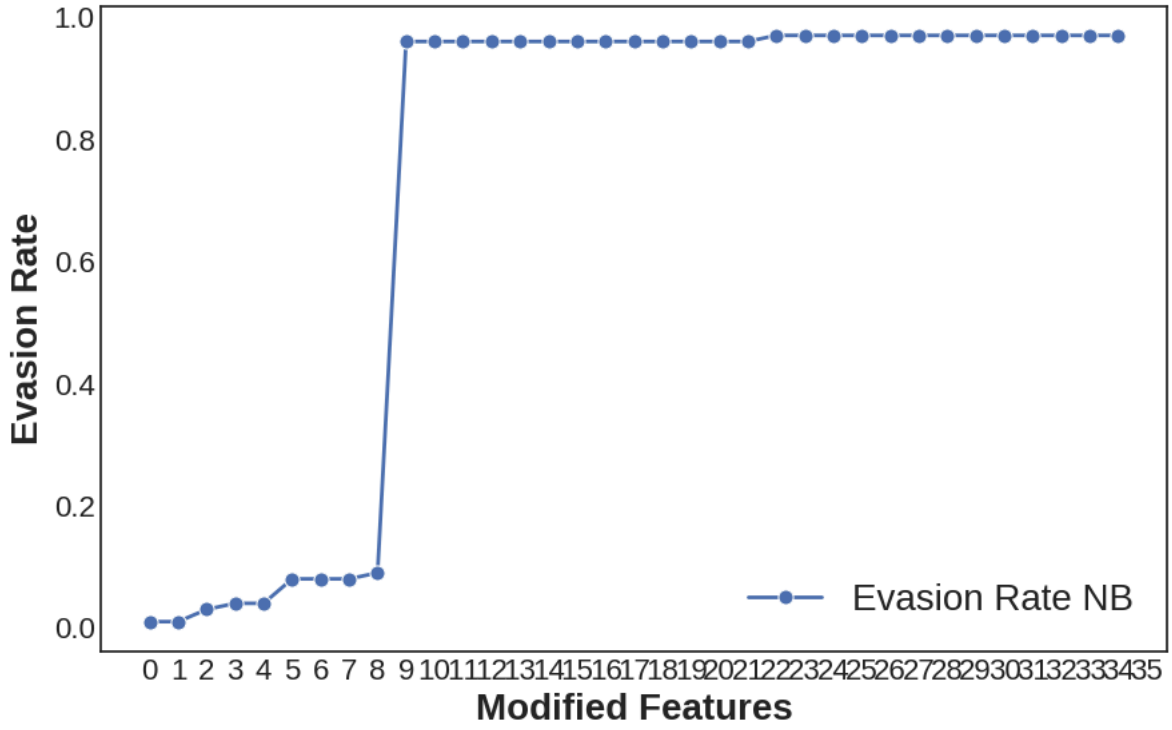
```
Accuracy     : 0.6671
Precision: 0.602
Recall: 0.992
F-Measure: 0.749
CM [[1219 2368]
 [  29 3584]]
-------------------------------------------------
EVASION RATE_nb:  0.01
-------------------------------------------------
*************************************************************************
Accuracy     : 0.6668
Precision: 0.602
Recall: 0.991
F-Measure: 0.749
CM [[1219 2368]
 [  31 3582]]
-------------------------------------------------
EVASION RATE_nb:  0.01
-------------------------------------------------
```

```
**************************************************************************
Accuracy     : 0.6574
Precision: 0.597
Recall: 0.973
F-Measure: 0.740
CM [[1219 2368]
 [  99 3514]]
--------------------------------------------------
EVASION RATE_nb:  0.03
--------------------------------------------------
**************************************************************************
Accuracy     : 0.6532
Precision: 0.595
Recall: 0.964
F-Measure: 0.736
CM [[1219 2368]
 [ 129 3484]]
--------------------------------------------------
EVASION RATE_nb:  0.04
--------------------------------------------------
**************************************************************************
Accuracy     : 0.6518
Precision: 0.595
Recall: 0.962
F-Measure: 0.735
CM [[1219 2368]
 [ 139 3474]]
--------------------------------------------------
EVASION RATE_nb:  0.04
--------------------------------------------------
**************************************************************************
Accuracy     : 0.6329
Precision: 0.585
Recall: 0.924
F-Measure: 0.716
CM [[1219 2368]
 [ 275 3338]]
--------------------------------------------------
EVASION RATE nb:  0.08
```

```
Less_ER3=Er_List_adv_nb[0:35]
```

```
df = pd.DataFrame (Less_ER3,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate NB"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-135-1684313fc31d>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



ADA_fi

```
Er_List_adv_ada = []
for feature in Top_features4:
    col2 = feature
    Adv_Test4.loc[Adv_Test4.Label == 1, col2] = 1
    Adv_x_test = Adv_Test4.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test4['Label']
    Adv_y_pred_Full_Model = ada_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision_adv_ada = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall_adv_ada = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score_adv_ada = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con_adv_ada = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er = con_adv_ada[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_ada.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision_adv_ada)
    print('Recall: %.3f' % recall_adv_ada)
    print('F-Measure: %.3f' % score_adv_ada)
    print('CM', con_adv_ada)
    print('-------------------------------------------------')
    print('EVASION RATE_ada: ', er)
    print('-------------------------------------------------')
    print("**************************************************************************")
```

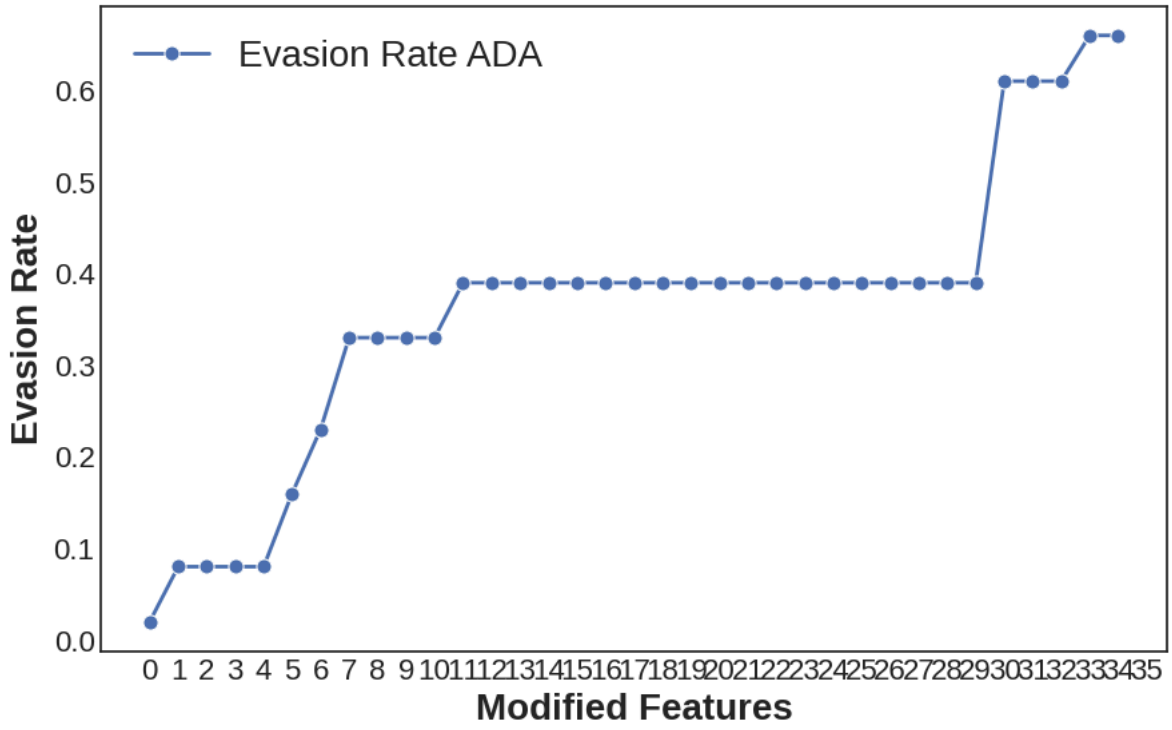
```
Accuracy     : 0.9762
Precision: 0.977
Recall: 0.976
F-Measure: 0.976
CM [[3503   84]
 [  87 3526]]
-------------------------------------------------
EVASION RATE_ada:  0.02
-------------------------------------------------
**************************************************************************
Accuracy     : 0.9474
Precision: 0.975
Recall: 0.918
F-Measure: 0.946
CM [[3503   84]
 [ 295 3318]]
-------------------------------------------------
EVASION RATE_ada:  0.08
-------------------------------------------------
```

```
********************************************************************
Accuracy     : 0.9474
Precision: 0.975
Recall: 0.918
F-Measure: 0.946
CM [[3503   84]
 [ 295 3318]]
--------------------------------------------------
EVASION RATE_ada:  0.08
--------------------------------------------------
********************************************************************
Accuracy     : 0.9474
Precision: 0.975
Recall: 0.918
F-Measure: 0.946
CM [[3503   84]
 [ 295 3318]]
--------------------------------------------------
EVASION RATE_ada:  0.08
--------------------------------------------------
********************************************************************
Accuracy     : 0.9474
Precision: 0.975
Recall: 0.918
F-Measure: 0.946
CM [[3503   84]
 [ 295 3318]]
--------------------------------------------------
EVASION RATE_ada:  0.08
--------------------------------------------------
********************************************************************
Accuracy     : 0.9058
Precision: 0.973
Recall: 0.836
F-Measure: 0.899
CM [[3503   84]
 [ 594 3019]]
--------------------------------------------------
EVASION RATE ada:  0.16
```

```
Less_ER4=Er_List_adv_ada[0:35]
```

```
df = pd.DataFrame (Less_ER4,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate ADA"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-138-88df41d8cbc4>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



KNN_fi

```
Er_List_adv_knn = []
for feature in Top_features5:
    col2 = feature
    Adv_Test5.loc[Adv_Test5.Label == 1, col2] = 1
    Adv_x_test = Adv_Test5.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test5['Label']
    Adv_y_pred_Full_Model = knn_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision_adv_knn = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall_adv_knn = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score_adv_knn = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con_adv_knn = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er = con_adv_knn[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_knn.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision_adv_knn)
    print('Recall: %.3f' % recall_adv_knn)
    print('F-Measure: %.3f' % score_adv_knn)
    print('CM', con_adv_knn)
    print('-------------------------------------------------')
    print('EVASION RATE_knn: ', er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

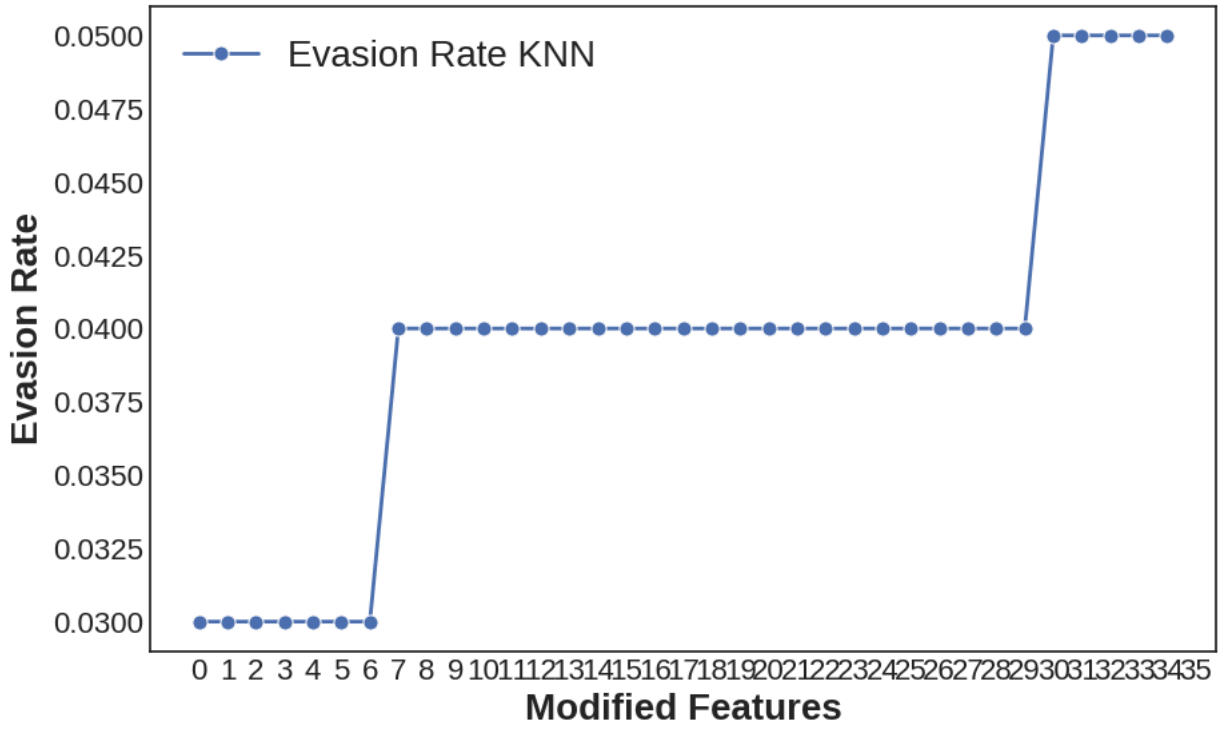
```
Accuracy     : 0.9776
Precision: 0.984
Recall: 0.971
F-Measure: 0.978
CM [[3529   58]
 [ 103 3510]]
-------------------------------------------------
EVASION RATE_knn:  0.03
-------------------------------------------------
*************************************************************************
Accuracy     : 0.9774
Precision: 0.984
Recall: 0.971
F-Measure: 0.977
CM [[3529   58]
 [ 105 3508]]
-------------------------------------------------
EVASION RATE_knn:  0.03
-------------------------------------------------
```

```
**************************************************************************
Accuracy     : 0.9772
Precision: 0.984
Recall: 0.971
F-Measure: 0.977
CM [[3529   58]
 [ 106 3507]]
--------------------------------------------------
EVASION RATE_knn:  0.03
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9767
Precision: 0.984
Recall: 0.970
F-Measure: 0.977
CM [[3529   58]
 [ 110 3503]]
--------------------------------------------------
EVASION RATE_knn:  0.03
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9764
Precision: 0.984
Recall: 0.969
F-Measure: 0.976
CM [[3529   58]
 [ 112 3501]]
--------------------------------------------------
EVASION RATE_knn:  0.03
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9754
Precision: 0.984
Recall: 0.967
F-Measure: 0.975
CM [[3529   58]
 [ 119 3494]]
--------------------------------------------------
EVASION RATE knn:  0.03
```

```
Less_ER5=Er_List_adv_knn[0:35]
```

```
df = pd.DataFrame (Less_ER5,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate KNN"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-141-17804520411a>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



Optimised KNN

```
Er_List_adv_optimised_knn = []
for feature in Top_features_optimised1:
    col2 = feature
    Adv_Test_optimised1.loc[Adv_Test_optimised1.Label == 1, col2] = 1
    Adv_x_test = Adv_Test_optimised1.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test_optimised1['Label']
    Adv_y_pred_Full_Model = knn_optimised_model.predict(Adv_x_test)
    #-----------------------------------------------
    precision_adv_knn = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall_adv_knn = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score_adv_knn = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con_adv_knn = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er = con_adv_knn[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_optimised_knn.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision_adv_knn)
    print('Recall: %.3f' % recall_adv_knn)
    print('F-Measure: %.3f' % score_adv_knn)
    print('CM', con_adv_knn)
    print('-------------------------------------------------')
    print('EVASION RATE_knn_optimised: ', er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

Show hidden output

```
Less_ER_optimised_knn=Er_List_adv_optimised_knn[0:35]
```

```
df = pd.DataFrame (Less_ER_optimised_knn,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate KNN_optimised"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

LR_fi

```
Er_List_adv_lr = []
for feature in Top_features6:
    col2 = feature
    Adv_Test6.loc[Adv_Test6.Label == 1, col2] = 1
    Adv_x_test = Adv_Test6.drop(['Label'], axis=1)
    Adv_y_test = Adv_Test6['Label']
    Adv_y_pred_Full_Model = lr_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision_adv_lr = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall_adv_lr = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score_adv_lr = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con_adv_lr = confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er = con_adv_lr[1][0] / 3613
    er = round(er, 2)
    Er_List_adv_lr.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision_adv_lr)
    print('Recall: %.3f' % recall_adv_lr)
    print('F-Measure: %.3f' % score_adv_lr)
    print('CM', con_adv_lr)
    print('-------------------------------------------------')
    print('EVASION RATE_lr: ', er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-94856c3d4765> in <cell line: 2>()
      1 Er_List_adv_lr = []
----> 2 for feature in Top_features6:
      3     col2 = feature
      4     Adv_Test6.loc[Adv_Test6.Label == 1, col2] = 1
      5     Adv_x_test = Adv_Test6.drop(['Label'], axis=1)

NameError: name 'Top_features6' is not defined
```

```
Less_ER6=Er_List_adv_lr[0:35]
```

```
df = pd.DataFrame (Less_ER6,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate LR"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

Double-click (or enter) to edit

Performing Feature Removal (FR) Attack

```
# Getting the top 35 features based on the frequency in to a list
Top_features_fr1 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/MilineFeature.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features_fr1.append(w)

Top_features_fr2 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/MilineFeature.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features_fr2.append(w)

Top_features_fr3 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/MilineFeature.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features_fr3.append(w)

Top_features_fr4 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/MilineFeature.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features_fr4.append(w)

Top_features_fr5 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/MilineFeature.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features_fr5.append(w)

Top_features_fr6 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/MilineFeature.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features_fr6.append(w)

Top_features_fr7 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/MilineFeature.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features_fr7.append(w)

Top_features_optimised2 = []
with open("/content/drive/MyDrive/Colab Notebooks/CIS4517_Project/MilineFeature.txt", "r") as f:
    for feature in f:
        w = feature.strip('\n')
        Top_features_optimised2.append(w)
```

```
print(Top_features_optimised2)
```

```
['android.permission.SEND', 'getConnectionInfo', 'android.permission.READ_PHONE_STATE', 'getrlimit', 'send', 'android.pe
```

```
Adv_Test_fr1=Xn_test.copy()
Adv_Test_fr2=Xn_test.copy()
Adv_Test_fr3=Xn_test.copy()
Adv_Test_fr4=Xn_test.copy()
Adv_Test_fr5=Xn_test.copy()
Adv_Test_fr6=Xn_test.copy()
Adv_Test_fr7=Xn_test.copy()

Adv_Test_optimised2=Xn_test.copy()
```

SVM

```
Er_List_fr_svm=[]
for feature in Top_features_fr1:
    col2=feature
    Adv_Test_fr1.loc[Adv_Test_fr1.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fr1.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fr1['Label']
    Adv_y_pred_Full_Model = svm_full_model.predict(Adv_x_test)
    #-----------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-----------------------------------------------
```

```
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fr_svm.append(er)
    #-----------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-----------------------------------------------')
    print('EVASION RATE_fr_svm: ',er)
    print('-----------------------------------------------')
    print("**********************************************************************")
```

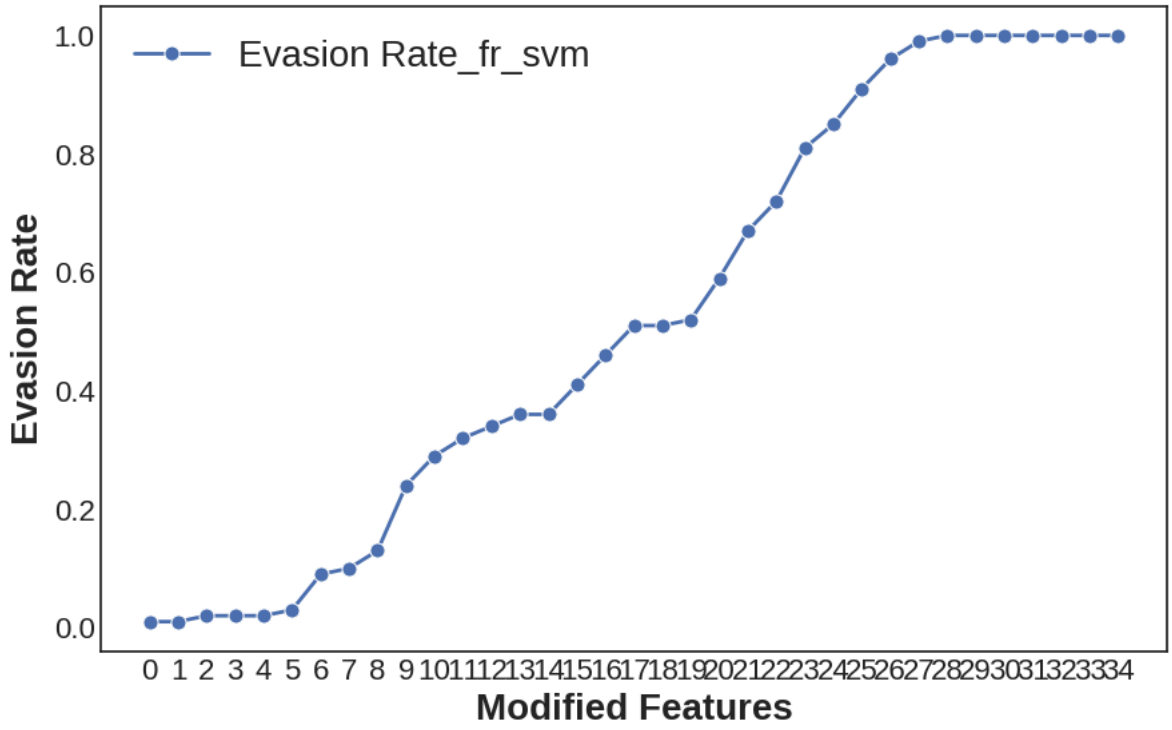
```
Accuracy     : 0.9919
Precision: 0.995
Recall: 0.989
F-Measure: 0.992
CM [[3570   17]
 [  41 3572]]
-----------------------------------------------
EVASION RATE_fr_svm:  0.01
-----------------------------------------------
**********************************************************************
Accuracy     : 0.9915
Precision: 0.995
Recall: 0.988
F-Measure: 0.992
CM [[3570   17]
 [  44 3569]]
-----------------------------------------------
EVASION RATE_fr_svm:  0.01
-----------------------------------------------
**********************************************************************
Accuracy     : 0.9892
Precision: 0.995
Recall: 0.983
F-Measure: 0.989
CM [[3570   17]
 [  61 3552]]
-----------------------------------------------
EVASION RATE_fr_svm:  0.02
-----------------------------------------------
**********************************************************************
Accuracy     : 0.9879
Precision: 0.995
Recall: 0.981
F-Measure: 0.988
CM [[3570   17]
 [  70 3543]]
-----------------------------------------------
EVASION RATE_fr_svm:  0.02
-----------------------------------------------
**********************************************************************
Accuracy     : 0.9854
Precision: 0.995
Recall: 0.976
F-Measure: 0.985
CM [[3570   17]
 [  88 3525]]
-----------------------------------------------
EVASION RATE_fr_svm:  0.02
-----------------------------------------------
**********************************************************************
Accuracy     : 0.9832
Precision: 0.995
Recall: 0.971
F-Measure: 0.983
CM [[3570   17]
 [ 104 3509]]
-----------------------------------------------
EVASION RATE_fr_svm:  0.03
```

```
Less_ER_fr1=Er_List_fr_svm[0:35]
```

```
df = pd.DataFrame (Less_ER_fr1,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fr_svm"], prop={"size":20})
plt.xticks(list(range(0, 35)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
```

```
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-78-7ccb1a0b903e>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-white')
```



RF

```
Er_List_fr_rf=[]
for feature in Top_features_fr2:
    col2=feature
    Adv_Test_fr2.loc[Adv_Test_fr2.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fr2.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fr2['Label']
    Adv_y_pred_Full_Model = rf_full_model.predict(Adv_x_test)
    #-----------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-----------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fr_rf.append(er)
    #-----------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-----------------------------------------------------')
    print('EVASION RATE_fr_rf: ',er)
    print('-----------------------------------------------------')
    print("**************************************************************************")
```

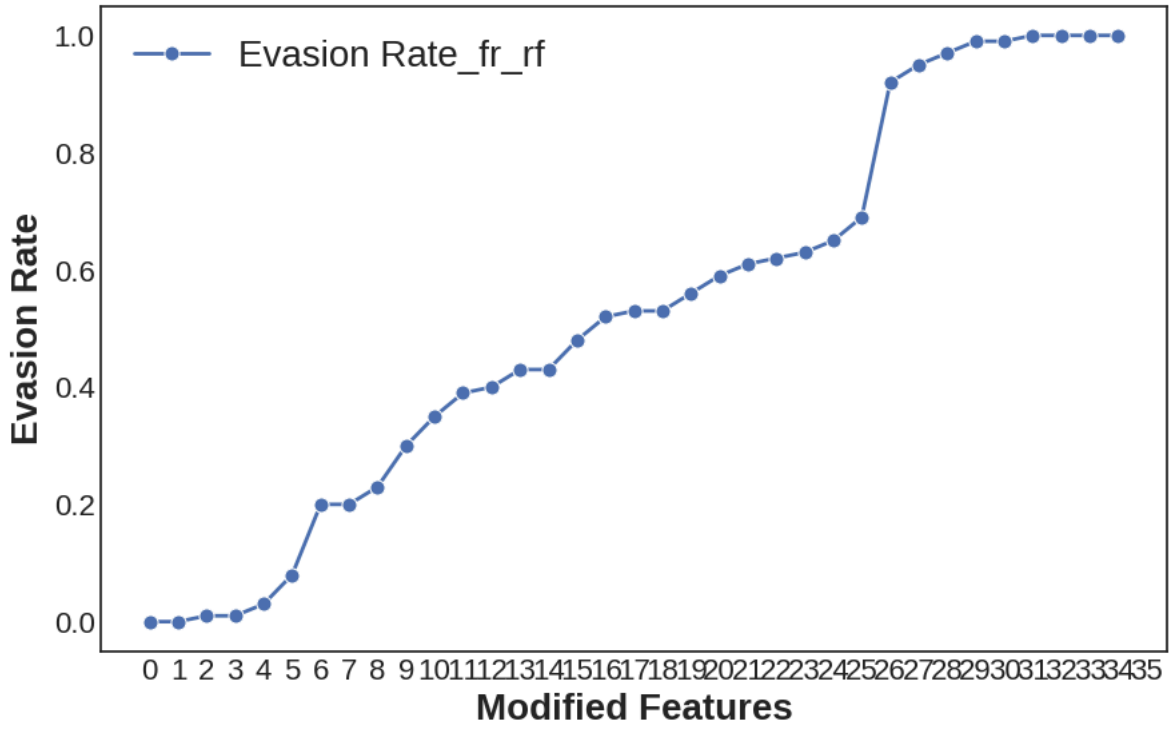
```
Accuracy     : 0.9981
Precision: 0.998
Recall: 0.998
F-Measure: 0.998
CM [[3581    6]
 [   8 3605]]
-----------------------------------------------------
EVASION RATE_fr_rf:  0.0
-----------------------------------------------------
**************************************************************************
Accuracy     : 0.9979
Precision: 0.998
Recall: 0.998
F-Measure: 0.998
CM [[3581    6]
 [   9 3604]]
```

```
--------------------------------------------------
EVASION RATE_fr_rf:  0.0
--------------------------------------------------
**********************************************************************
Accuracy     : 0.995
Precision: 0.998
Recall: 0.992
F-Measure: 0.995
CM [[3581    6]
 [  30 3583]]
--------------------------------------------------
EVASION RATE_fr_rf:  0.01
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9922
Precision: 0.998
Recall: 0.986
F-Measure: 0.992
CM [[3581    6]
 [  50 3563]]
--------------------------------------------------
EVASION RATE_fr_rf:  0.01
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9817
Precision: 0.998
Recall: 0.965
F-Measure: 0.981
CM [[3581    6]
 [ 126 3487]]
--------------------------------------------------
EVASION RATE_fr_rf:  0.03
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9596
Precision: 0.998
Recall: 0.921
F-Measure: 0.958
CM [[3581    6]
 [ 285 3328]]
--------------------------------------------------
EVASION RATE_fr_rf:  0.08
```

```
Less_ER_fr2=Er_List_fr_rf[0:35]
```

```
df = pd.DataFrame (Less_ER_fr2,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fr_rf"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-81-2524942abf86>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-white')
```



DT

```
Er_List_fr_dt=[]
for feature in Top_features_fr3:
    col2=feature
    Adv_Test_fr3.loc[Adv_Test_fr3.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fr3.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fr3['Label']
    Adv_y_pred_Full_Model = dt_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fr_dt.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fr_dt: ',er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

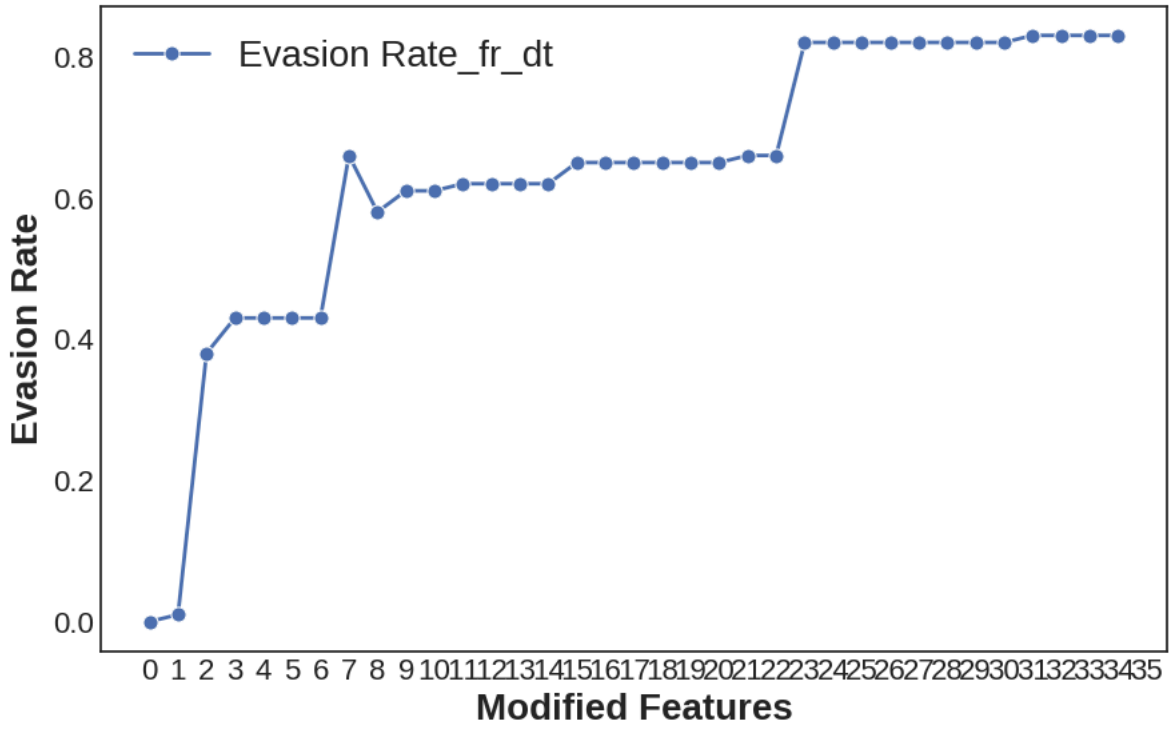
```
Accuracy     : 0.9968
Precision: 0.997
Recall: 0.996
F-Measure: 0.997
CM [[3577   10]
 [  13 3600]]
-------------------------------------------------
EVASION RATE_fr_dt:  0.0
-------------------------------------------------
*************************************************************************
Accuracy     : 0.9925
Precision: 0.997
Recall: 0.988
F-Measure: 0.992
CM [[3577   10]
 [  44 3569]]
-------------------------------------------------
EVASION RATE_fr_dt:  0.01
-------------------------------------------------
```

```
**************************************************************************
Accuracy     : 0.8074
Precision: 0.996
Recall: 0.619
F-Measure: 0.763
CM [[3577   10]
 [1377 2236]]
--------------------------------------------------
EVASION RATE_fr_dt:  0.38
--------------------------------------------------
**************************************************************************
Accuracy     : 0.7843
Precision: 0.995
Recall: 0.573
F-Measure: 0.727
CM [[3577   10]
 [1543 2070]]
--------------------------------------------------
EVASION RATE_fr_dt:  0.43
--------------------------------------------------
**************************************************************************
Accuracy     : 0.7843
Precision: 0.995
Recall: 0.573
F-Measure: 0.727
CM [[3577   10]
 [1543 2070]]
--------------------------------------------------
EVASION RATE_fr_dt:  0.43
--------------------------------------------------
**************************************************************************
Accuracy     : 0.7843
Precision: 0.995
Recall: 0.573
F-Measure: 0.727
CM [[3577   10]
 [1543 2070]]
--------------------------------------------------
EVASION RATE fr dt:  0.43
```

```
Less_ER_fr3=Er_List_fr_dt[0:35]
```

```
df = pd.DataFrame (Less_ER_fr3,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fr_dt"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-84-f9ef94f121b6>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-white')
```



NB

```
Er_List_fr_nb=[]
for feature in Top_features_fr4:
    col2=feature
    Adv_Test_fr4.loc[Adv_Test_fr4.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fr4.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fr4['Label']
    Adv_y_pred_Full_Model = nb_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fr_nb.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fr_nb: ',er)
    print('-------------------------------------------------')
    print("**************************************************************************")
```

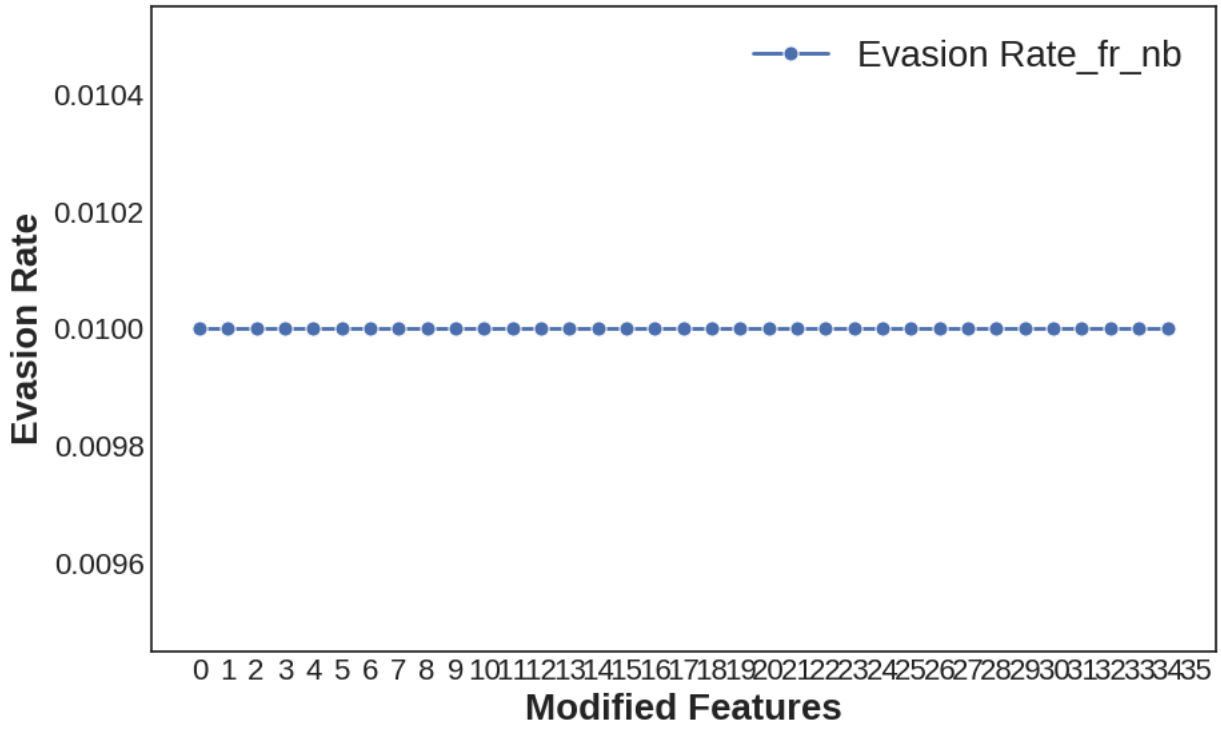
```
Accuracy     : 0.6671
Precision: 0.602
Recall: 0.992
F-Measure: 0.749
CM [[1219 2368]
 [  29 3584]]
-------------------------------------------------
EVASION RATE_fr_nb:  0.01
-------------------------------------------------
**************************************************************************
Accuracy     : 0.6671
Precision: 0.602
Recall: 0.992
F-Measure: 0.749
CM [[1219 2368]
 [  29 3584]]
-------------------------------------------------
EVASION RATE_fr_nb:  0.01
-------------------------------------------------
```

```
********************************************************************
Accuracy     : 0.6671
Precision: 0.602
Recall: 0.992
F-Measure: 0.749
CM [[1219 2368]
 [  29 3584]]
--------------------------------------------------
EVASION RATE_fr_nb:  0.01
--------------------------------------------------
********************************************************************
Accuracy     : 0.6671
Precision: 0.602
Recall: 0.992
F-Measure: 0.749
CM [[1219 2368]
 [  29 3584]]
--------------------------------------------------
EVASION RATE_fr_nb:  0.01
--------------------------------------------------
********************************************************************
Accuracy     : 0.6671
Precision: 0.602
Recall: 0.992
F-Measure: 0.749
CM [[1219 2368]
 [  29 3584]]
--------------------------------------------------
EVASION RATE_fr_nb:  0.01
--------------------------------------------------
********************************************************************
Accuracy     : 0.6669
Precision: 0.602
Recall: 0.992
F-Measure: 0.749
CM [[1219 2368]
 [  30 3583]]
--------------------------------------------------
EVASION RATE fr nb:  0.01
```

```
Less_ER_fr4=Er_List_fr_nb[0:35]
```

```
df = pd.DataFrame (Less_ER_fr4,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fr_nb"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-87-b24c6d273682>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-white')
```



ADA

```
Er_List_fr_ada=[]
for feature in Top_features_fr5:
    col2=feature
    Adv_Test_fr5.loc[Adv_Test_fr5.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fr5.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fr5['Label']
    Adv_y_pred_Full_Model = ada_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fr_ada.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fr_ada: ',er)
    print('-------------------------------------------------')
    print("**************************************************************************")
```

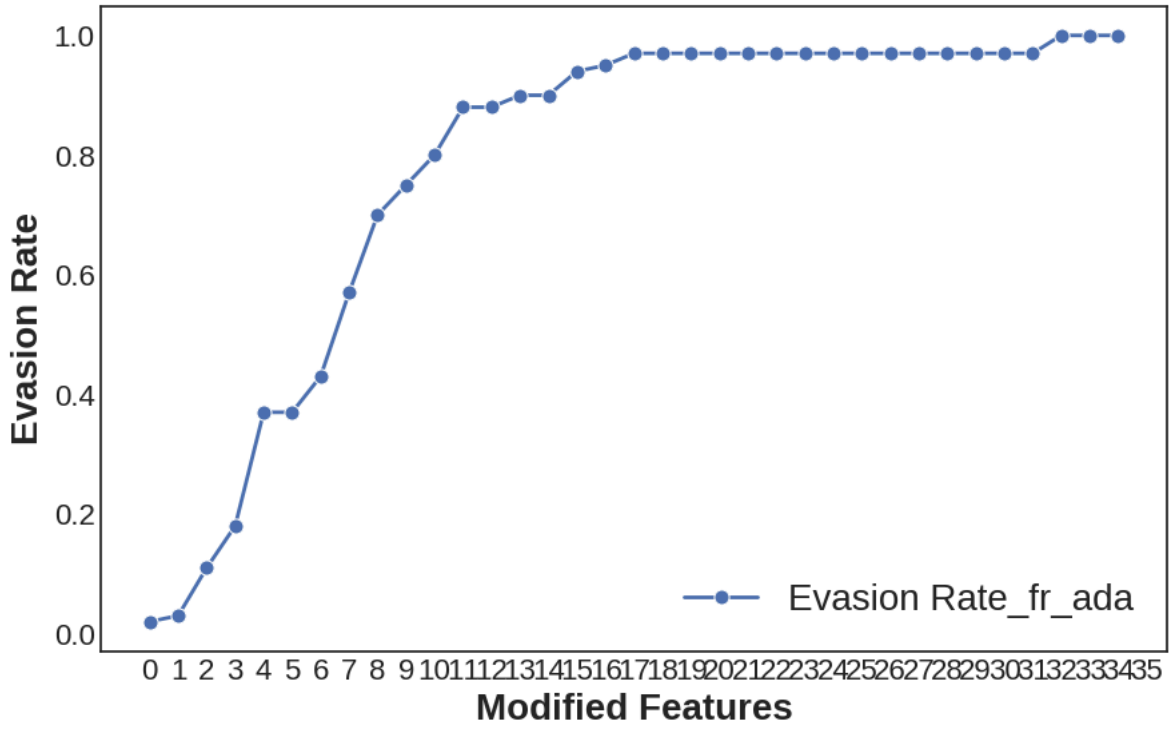
```
Accuracy     : 0.9762
Precision: 0.977
Recall: 0.976
F-Measure: 0.976
CM [[3503   84]
 [  87 3526]]
-------------------------------------------------
EVASION RATE_fr_ada:  0.02
-------------------------------------------------
**************************************************************************
Accuracy     : 0.9735
Precision: 0.977
Recall: 0.970
F-Measure: 0.973
CM [[3503   84]
 [ 107 3506]]
-------------------------------------------------
EVASION RATE_fr_ada:  0.03
-------------------------------------------------
```

```
**************************************************************************
Accuracy     : 0.9332
Precision: 0.975
Recall: 0.890
F-Measure: 0.930
CM [[3503   84]
 [ 397 3216]]
--------------------------------------------------
EVASION RATE_fr_ada:  0.11
--------------------------------------------------
**************************************************************************
Accuracy     : 0.8997
Precision: 0.973
Recall: 0.823
F-Measure: 0.892
CM [[3503   84]
 [ 638 2975]]
--------------------------------------------------
EVASION RATE_fr_ada:  0.18
--------------------------------------------------
**************************************************************************
Accuracy     : 0.8051
Precision: 0.965
Recall: 0.635
F-Measure: 0.766
CM [[3503   84]
 [1319 2294]]
--------------------------------------------------
EVASION RATE_fr_ada:  0.37
--------------------------------------------------
**************************************************************************
Accuracy     : 0.8051
Precision: 0.965
Recall: 0.635
F-Measure: 0.766
CM [[3503   84]
 [1319 2294]]
--------------------------------------------------
EVASION RATE fr ada:  0.37
```

```
Less_ER_fr5=Er_List_fr_ada[0:35]
```

```
df = pd.DataFrame (Less_ER_fr5,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fr_ada"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-90-3c5d03ba30f9>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-white')
```



KNN

```
Er_List_fr_knn=[]
for feature in Top_features_fr6:
    col2=feature
    Adv_Test_fr6.loc[Adv_Test_fr6.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fr6.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fr6['Label']
    Adv_y_pred_Full_Model = knn_full_model.predict(Adv_x_test)
    #-----------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-----------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fr_knn.append(er)
    #-----------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-----------------------------------------------------')
    print('EVASION RATE_fr_knn: ',er)
    print('-----------------------------------------------------')
    print("**************************************************************************")
```

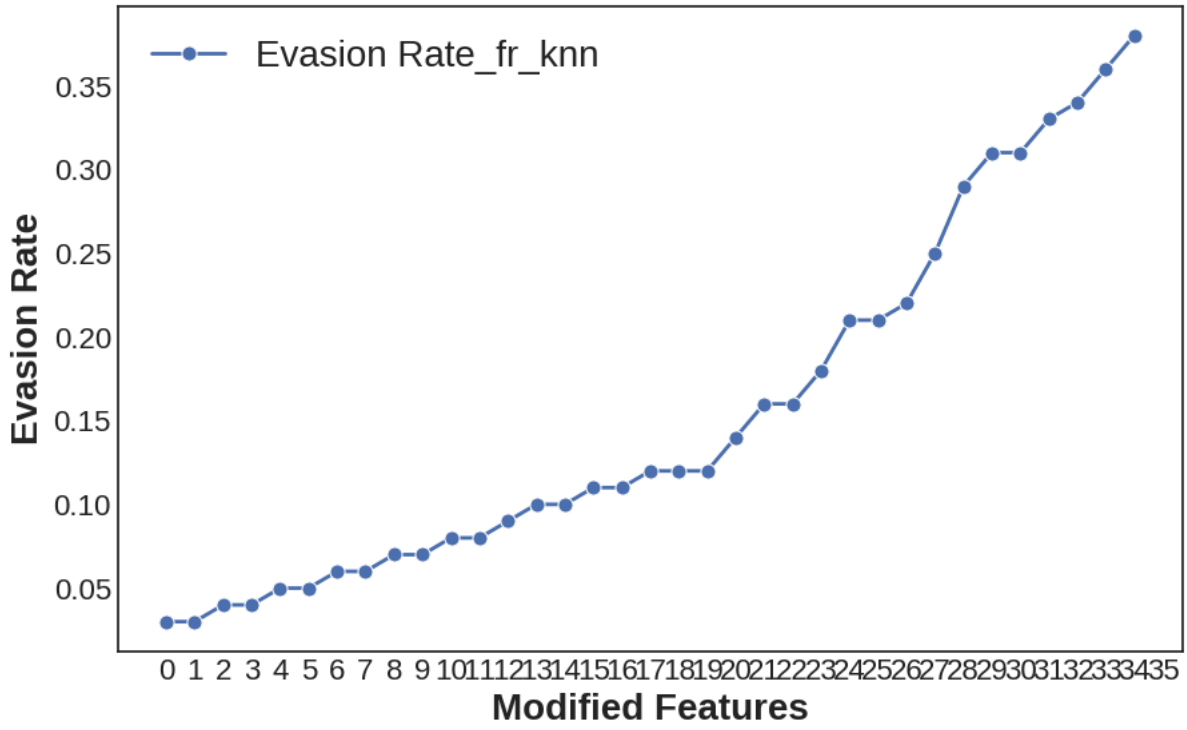
```
Accuracy     : 0.9767
Precision: 0.984
Recall: 0.970
F-Measure: 0.977
CM [[3529   58]
 [ 110 3503]]
-----------------------------------------------------
EVASION RATE_fr_knn:  0.03
-----------------------------------------------------
**************************************************************************
Accuracy     : 0.975
Precision: 0.984
Recall: 0.966
F-Measure: 0.975
CM [[3529   58]
 [ 122 3491]]
-----------------------------------------------------
EVASION RATE_fr_knn:  0.03
-----------------------------------------------------
```

```
**************************************************************************
Accuracy     : 0.9712
Precision: 0.984
Recall: 0.959
F-Measure: 0.971
CM [[3529   58]
 [ 149 3464]]
--------------------------------------------------
EVASION RATE_fr_knn:  0.04
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9703
Precision: 0.983
Recall: 0.957
F-Measure: 0.970
CM [[3529   58]
 [ 156 3457]]
--------------------------------------------------
EVASION RATE_fr_knn:  0.04
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9679
Precision: 0.983
Recall: 0.952
F-Measure: 0.968
CM [[3529   58]
 [ 173 3440]]
--------------------------------------------------
EVASION RATE_fr_knn:  0.05
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9653
Precision: 0.983
Recall: 0.947
F-Measure: 0.965
CM [[3529   58]
 [ 192 3421]]
--------------------------------------------------
EVASION RATE fr knn:  0.05
```

```
Less_ER_fr6=Er_List_fr_knn[0:35]
```

```
df = pd.DataFrame (Less_ER_fr6,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fr_knn"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-93-3e69503512f8>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-white')
```



KNN Optimised

```
Er_List_fr_knn_optimised=[]
for feature in Top_features_optimised2:
    col2=feature
    Adv_Test_optimised2.loc[Adv_Test_optimised2.Label == 1, col2]=0
    Adv_x_test=Adv_Test_optimised2.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_optimised2['Label']
    Adv_y_pred_Full_Model = knn_optimised_model.predict(Adv_x_test)
    #-----------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-----------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fr_knn_optimised.append(er)
    #-----------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-----------------------------------------------')
    print('EVASION RATE_fr_knn_optimised: ',er)
    print('-----------------------------------------------')
    print("**************************************************************************")
```

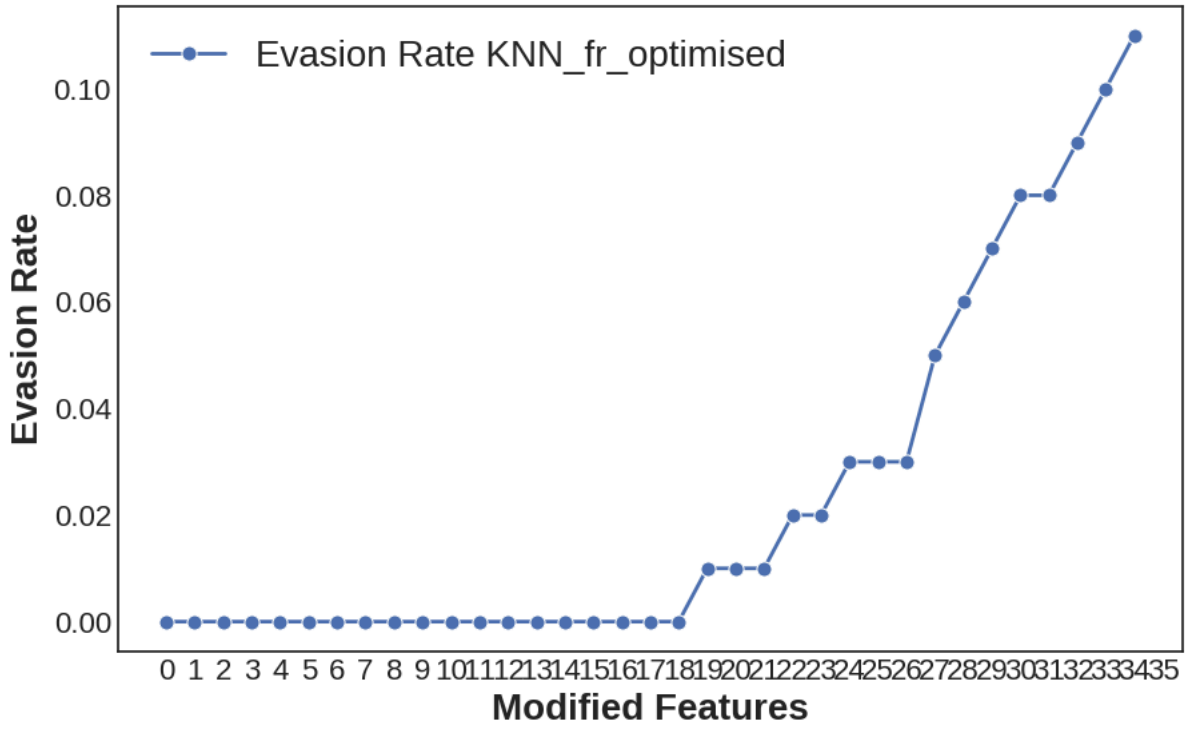
```
Accuracy     : 0.9981
Precision: 0.998
Recall: 0.999
F-Measure: 0.998
CM [[3578    9]
 [   5 3608]]
-----------------------------------------------
EVASION RATE_fr_knn_optimised:  0.0
-----------------------------------------------
**************************************************************************
Accuracy     : 0.9981
Precision: 0.998
Recall: 0.999
F-Measure: 0.998
CM [[3578    9]
 [   5 3608]]
-----------------------------------------------
EVASION RATE_fr_knn_optimised:  0.0
-----------------------------------------------
**************************************************************************
```

```
Accuracy     : 0.9981
Precision: 0.998
Recall: 0.999
F-Measure: 0.998
CM [[3578    9]
 [   5 3608]]
------------------------------------------------
EVASION RATE_fr_knn_optimised:  0.0
------------------------------------------------
**********************************************************************
Accuracy     : 0.9979
Precision: 0.998
Recall: 0.998
F-Measure: 0.998
CM [[3578    9]
 [   6 3607]]
------------------------------------------------
EVASION RATE_fr_knn_optimised:  0.0
------------------------------------------------
**********************************************************************
Accuracy     : 0.9976
Precision: 0.998
Recall: 0.998
F-Measure: 0.998
CM [[3578    9]
 [   8 3605]]
------------------------------------------------
EVASION RATE_fr_knn_optimised:  0.0
------------------------------------------------
**********************************************************************
Accuracy     : 0.9976
Precision: 0.998
Recall: 0.998
F-Measure: 0.998
CM [[3578    9]
 [   8 3605]]
------------------------------------------------
EVASION RATE_fr_knn_optimised:  0.0
```

```
Less_ER_optimised_fr_knn=Er_List_fr_knn_optimised[0:35]
```

```
df = pd.DataFrame (Less_ER_optimised_fr_knn,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate KNN_fr_optimised"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-96-53b566f3689c>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-white')
```



LR

```
Er_List_fr_lr=[]
for feature in Top_features_fr7:
    col2=feature
    Adv_Test_fr7.loc[Adv_Test_fr7.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fr7.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fr7['Label']
    Adv_y_pred_Full_Model = lr_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fr_lr.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model), 4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fr_lr: ',er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

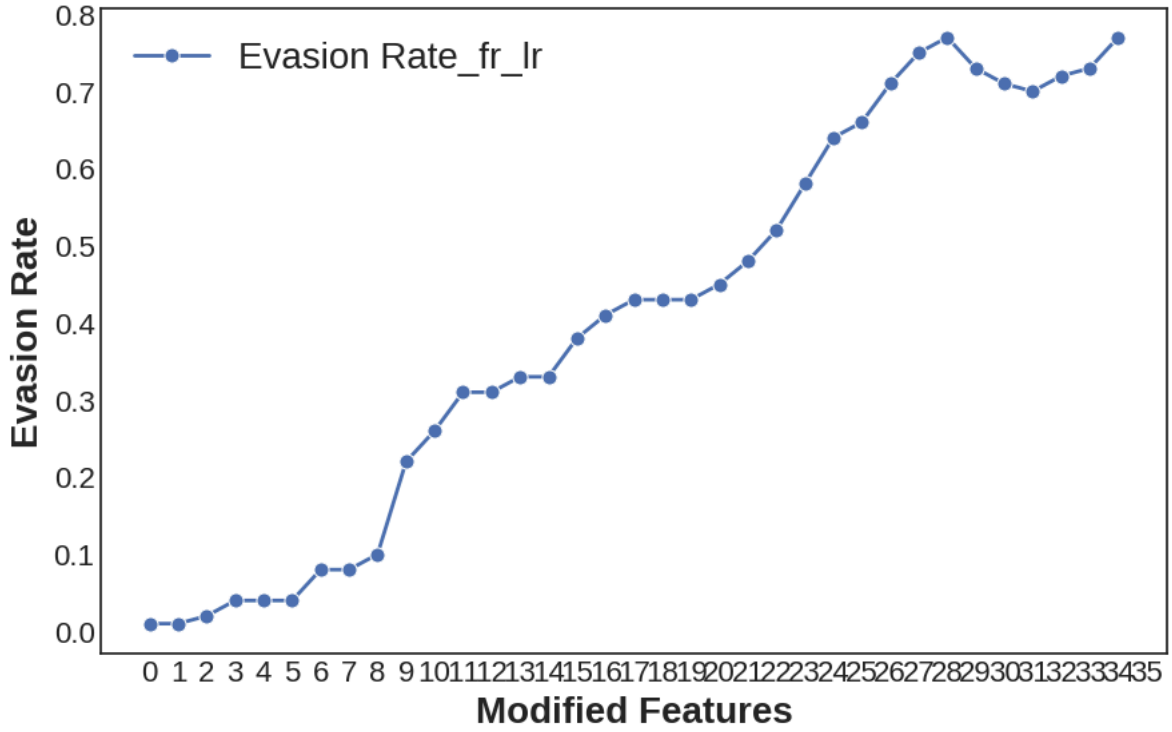
```
Accuracy     : 0.9906
Precision: 0.994
Recall: 0.988
F-Measure: 0.991
CM [[3564   23]
 [  45 3568]]
-------------------------------------------------
EVASION RATE_fr_lr:  0.01
-------------------------------------------------
*************************************************************************
Accuracy     : 0.9897
Precision: 0.994
Recall: 0.986
F-Measure: 0.990
CM [[3564   23]
 [  51 3562]]
-------------------------------------------------
EVASION RATE_fr_lr:  0.01
-------------------------------------------------
```

```
**************************************************************************
Accuracy     : 0.9847
Precision: 0.994
Recall: 0.976
F-Measure: 0.985
CM [[3564   23]
 [  87 3526]]
--------------------------------------------------
EVASION RATE_fr_lr:  0.02
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9783
Precision: 0.993
Recall: 0.963
F-Measure: 0.978
CM [[3564   23]
 [ 133 3480]]
--------------------------------------------------
EVASION RATE_fr_lr:  0.04
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9756
Precision: 0.993
Recall: 0.958
F-Measure: 0.975
CM [[3564   23]
 [ 153 3460]]
--------------------------------------------------
EVASION RATE_fr_lr:  0.04
--------------------------------------------------
**************************************************************************
Accuracy     : 0.9768
Precision: 0.993
Recall: 0.960
F-Measure: 0.976
CM [[3564   23]
 [ 144 3469]]
--------------------------------------------------
EVASION RATE fr lr:  0.04
```

```
Less_ER_fr7=Er_List_fr_lr[0:35]
```

```
df = pd.DataFrame (Less_ER_fr7,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fr_lr"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-99-70158d6a8fce>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecate
  plt.style.use('seaborn-white')
```



With Feature Injection (Mimicry) and Removal (FR)

```
Adv_Test3=test.copy()

Adv_Test_fir1=Xn_test.copy()
Adv_Test_fir2=Xn_test.copy()
Adv_Test_fir3=Xn_test.copy()
Adv_Test_fir4=Xn_test.copy()
Adv_Test_fir5=Xn_test.copy()
Adv_Test_fir6=Xn_test.copy()
Adv_Test_fir7=Xn_test.copy()

Adv_Test_optimised3=Xn_test.copy()
```

SVM_fir

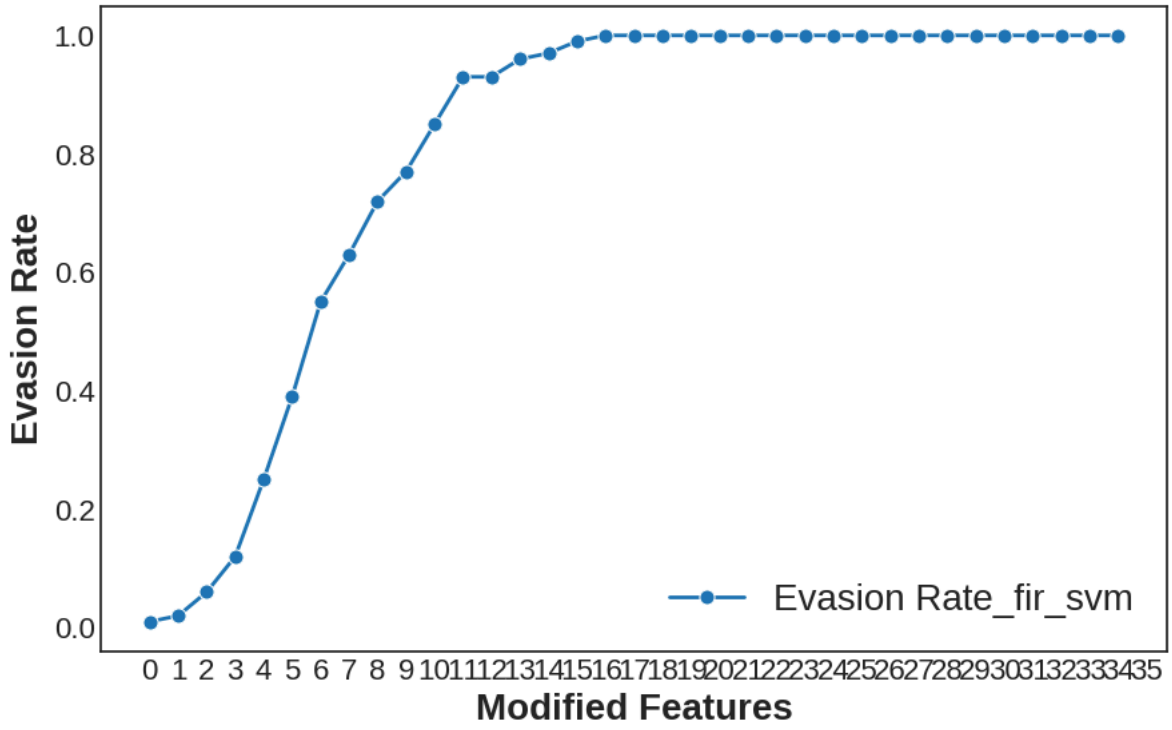
```
Er_List_fir_svm=[]
for i in range(0,35):
    col1=Top_features[i]
    col2=Top_features_fr1[i]
    Adv_Test_fir1.loc[Adv_Test_fir1.Label == 1, col1] =1
    Adv_Test_fir1.loc[Adv_Test_fir1.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fir1.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fir1['Label']
    Adv_y_pred_Full_Model = svm_full_model.predict(Adv_x_test)
    #-----------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-----------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fir_svm.append(er)
    #-----------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model),4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-----------------------------------------------')
    print('EVASION RATE_fir_svm: ',er)
    print('-----------------------------------------------')
    print("**********************************************************************")
```

```
Accuracy     : 0.9907
Precision: 0.995
Recall: 0.986
F-Measure: 0.991
CM [[3570   17]
 [  50 3563]]
--------------------------------------------------
EVASION RATE_fir_svm:  0.01
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9879
Precision: 0.995
Recall: 0.981
F-Measure: 0.988
CM [[3570   17]
 [  70 3543]]
--------------------------------------------------
EVASION RATE_fir_svm:  0.02
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9685
Precision: 0.995
Recall: 0.942
F-Measure: 0.968
CM [[3570   17]
 [ 210 3403]]
--------------------------------------------------
EVASION RATE_fir_svm:  0.06
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9394
Precision: 0.995
Recall: 0.884
F-Measure: 0.936
CM [[3570   17]
 [ 419 3194]]
--------------------------------------------------
EVASION RATE_fir_svm:  0.12
--------------------------------------------------
**********************************************************************
Accuracy     : 0.8712
Precision: 0.994
Recall: 0.748
F-Measure: 0.854
CM [[3570   17]
 [ 910 2703]]
--------------------------------------------------
EVASION RATE_fir_svm:  0.25
--------------------------------------------------
**********************************************************************
Accuracy     : 0.8036
Precision: 0.992
Recall: 0.613
F-Measure: 0.758
CM [[3570   17]
 [1397 2216]]
--------------------------------------------------
EVASION RATE_fir_svm:  0.39
```

```
Less_ER_fir1=Er_List_fir_svm[0:35]
```

```
df = pd.DataFrame (Less_ER_fir1,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fir_svm"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-107-606f6f4a2557>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



RF_fir

```
Er_List_fir_rf=[]
for i in range(0,35):
    col1=Top_features1[i]
    col2=Top_features_fr2[i]
    Adv_Test_fir2.loc[Adv_Test_fir2.Label == 1, col1] =1
    Adv_Test_fir2.loc[Adv_Test_fir2.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fir2.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fir2['Label']
    Adv_y_pred_Full_Model = rf_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fir_rf.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model),4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fir_rf: ',er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

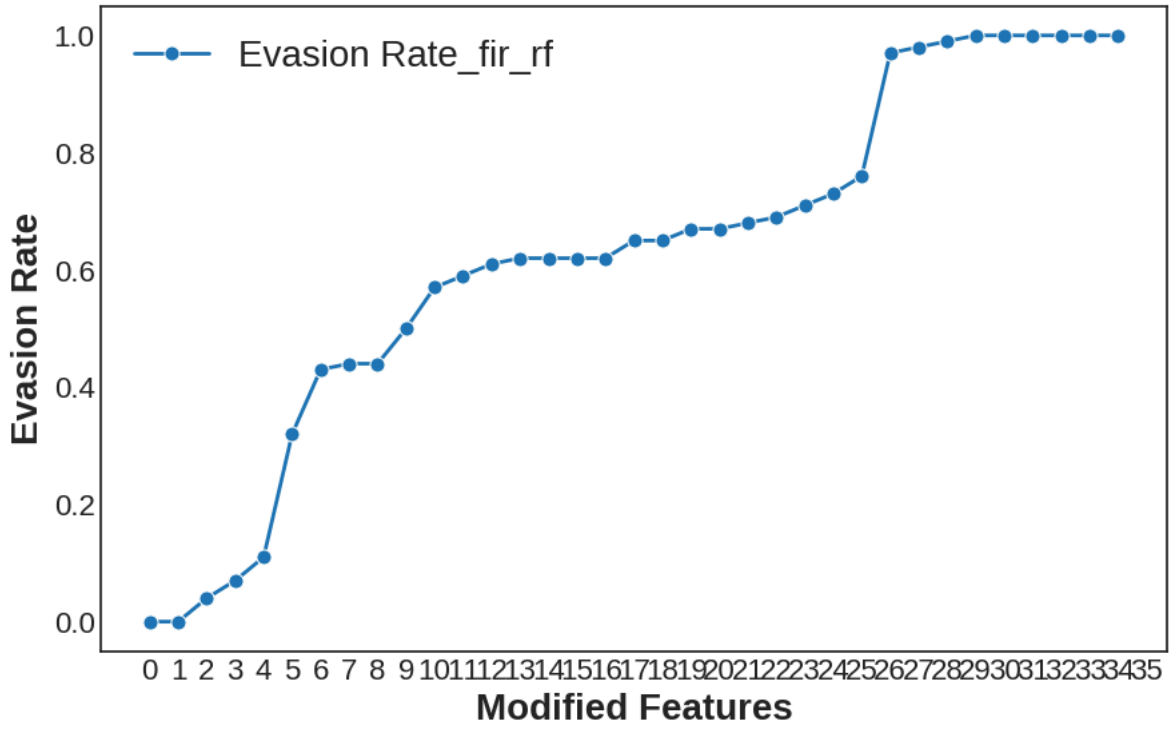
```
Accuracy     : 0.9981
Precision: 0.998
Recall: 0.998
F-Measure: 0.998
CM [[3580    7]
 [   7 3606]]
-------------------------------------------------
EVASION RATE_fir_rf:  0.0
-------------------------------------------------
*************************************************************************
Accuracy     : 0.9968
Precision: 0.998
Recall: 0.996
F-Measure: 0.997
CM [[3580    7]
 [  16 3597]]
-------------------------------------------------
```

```
EVASION RATE_fir_rf:  0.0
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9771
Precision: 0.998
Recall: 0.956
F-Measure: 0.977
CM [[3580    7]
 [ 158 3455]]
--------------------------------------------------
EVASION RATE_fir_rf:  0.04
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9622
Precision: 0.998
Recall: 0.927
F-Measure: 0.961
CM [[3580    7]
 [ 265 3348]]
--------------------------------------------------
EVASION RATE_fir_rf:  0.07
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9431
Precision: 0.998
Recall: 0.888
F-Measure: 0.940
CM [[3580    7]
 [ 403 3210]]
--------------------------------------------------
EVASION RATE_fir_rf:  0.11
--------------------------------------------------
**********************************************************************
Accuracy     : 0.8378
Precision: 0.997
Recall: 0.679
F-Measure: 0.808
CM [[3580    7]
 [1161 2452]]
--------------------------------------------------
EVASION RATE_fir_rf:  0.32
```

```
Less_ER_fir2=Er_List_fir_rf[0:35]
```

```
df = pd.DataFrame (Less_ER_fir2,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fir_rf"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-110-4568426ee400>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



DT_fir

```
Er_List_fir_dt=[]
for i in range(0,35):
    col1=Top_features2[i]
    col2=Top_features_fr3[i]
    Adv_Test_fir3.loc[Adv_Test_fir3.Label == 1, col1] =1
    Adv_Test_fir3.loc[Adv_Test_fir3.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fir3.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fir3['Label']
    Adv_y_pred_Full_Model = dt_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fir_dt.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model),4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fir_dt: ',er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

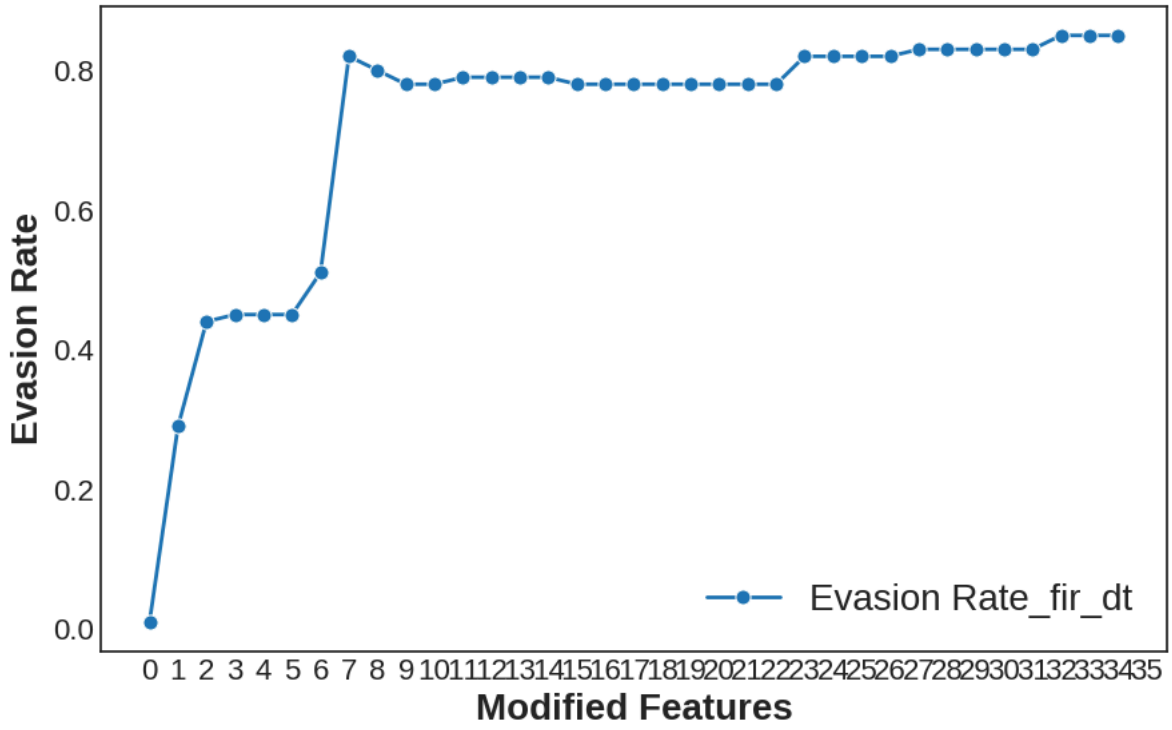
```
Accuracy     : 0.9946
Precision: 0.996
Recall: 0.994
F-Measure: 0.995
CM [[3571   16]
 [  23 3590]]
-------------------------------------------------
EVASION RATE_fir_dt:  0.01
-------------------------------------------------
*************************************************************************
Accuracy     : 0.8544
Precision: 0.994
Recall: 0.714
F-Measure: 0.831
CM [[3571   16]
 [1032 2581]]
-------------------------------------------------
```

```
EVASION RATE_fir_dt:  0.29
--------------------------------------------------
******************************************************************
Accuracy     : 0.7762
Precision: 0.992
Recall: 0.559
F–Measure: 0.715
CM [[3571   16]
 [1595 2018]]
--------------------------------------------------
EVASION RATE_fir_dt:  0.44
--------------------------------------------------
******************************************************************
Accuracy     : 0.7712
Precision: 0.992
Recall: 0.549
F–Measure: 0.706
CM [[3571   16]
 [1631 1982]]
--------------------------------------------------
EVASION RATE_fir_dt:  0.45
--------------------------------------------------
******************************************************************
Accuracy     : 0.7712
Precision: 0.992
Recall: 0.549
F–Measure: 0.706
CM [[3571   16]
 [1631 1982]]
--------------------------------------------------
EVASION RATE_fir_dt:  0.45
--------------------------------------------------
******************************************************************
Accuracy     : 0.7712
Precision: 0.992
Recall: 0.549
F–Measure: 0.706
CM [[3571   16]
 [1631 1982]]
--------------------------------------------------
EVASION RATE_fir_dt:  0.45
```

```
Less_ER_fir3=Er_List_fir_dt[0:35]
```

```
df = pd.DataFrame (Less_ER_fir3,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fir_dt"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-113-73f8a697c292>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



NB_fir

```
Er_List_fir_nb=[]
for i in range(0,35):
    col1=Top_features3[i]
    col2=Top_features_fr4[i]
    Adv_Test_fir4.loc[Adv_Test_fir4.Label == 1, col1] =1
    Adv_Test_fir4.loc[Adv_Test_fir4.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fir4.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fir4['Label']
    Adv_y_pred_Full_Model = nb_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fir_nb.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model),4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fir_nb: ',er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

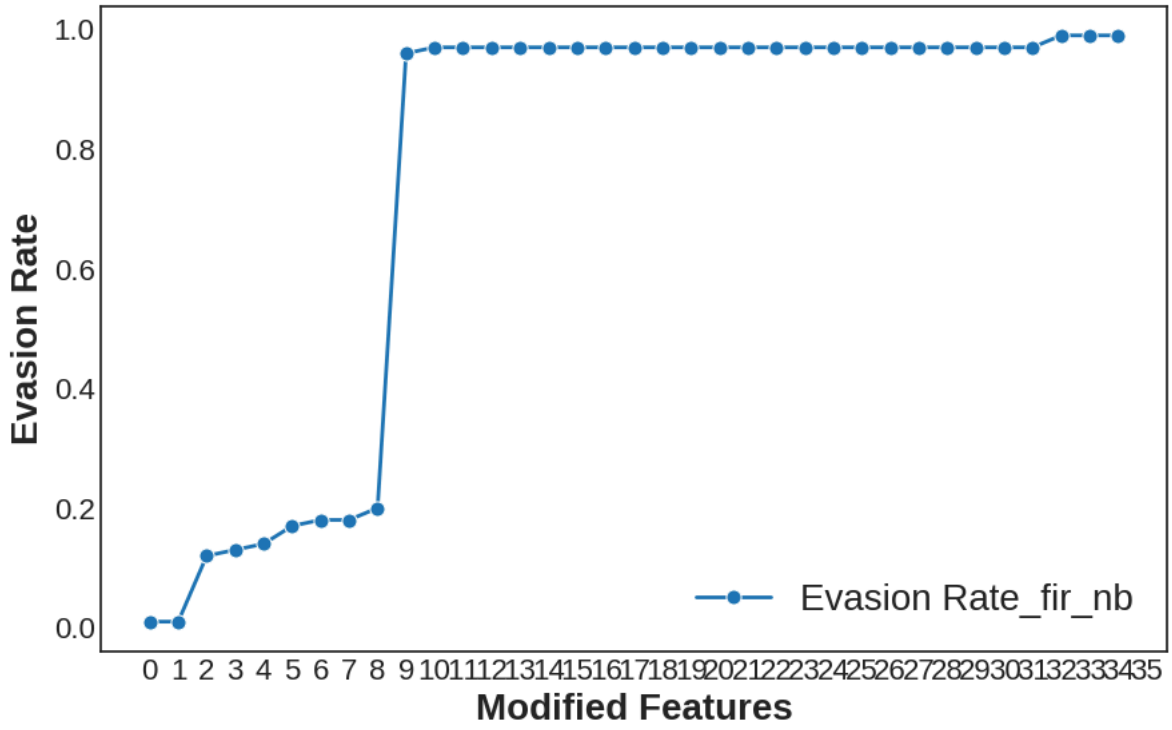
```
Accuracy     : 0.6762
Precision: 0.609
Recall: 0.990
F-Measure: 0.754
CM [[1292 2295]
 [  36 3577]]
-------------------------------------------------
EVASION RATE_fir_nb:  0.01
-------------------------------------------------
*************************************************************************
Accuracy     : 0.6757
Precision: 0.609
Recall: 0.989
F-Measure: 0.754
CM [[1292 2295]
 [  40 3573]]
-------------------------------------------------
```

```
EVASION RATE_fir_nb:  0.01
--------------------------------------------------
**********************************************************************
Accuracy     : 0.619
Precision: 0.580
Recall: 0.876
F-Measure: 0.698
CM [[1292 2295]
 [ 448 3165]]
--------------------------------------------------
EVASION RATE_fir_nb:  0.12
--------------------------------------------------
**********************************************************************
Accuracy     : 0.614
Precision: 0.577
Recall: 0.866
F-Measure: 0.692
CM [[1292 2295]
 [ 484 3129]]
--------------------------------------------------
EVASION RATE_fir_nb:  0.13
--------------------------------------------------
**********************************************************************
Accuracy     : 0.6108
Precision: 0.575
Recall: 0.860
F-Measure: 0.689
CM [[1292 2295]
 [ 507 3106]]
--------------------------------------------------
EVASION RATE_fir_nb:  0.14
--------------------------------------------------
**********************************************************************
Accuracy     : 0.596
Precision: 0.566
Recall: 0.830
F-Measure: 0.673
CM [[1292 2295]
 [ 614 2999]]
--------------------------------------------------
EVASION RATE_fir_nb:  0.17
```

```
Less_ER_fir4=Er_List_fir_nb[0:35]
```

```
df = pd.DataFrame (Less_ER_fir4,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fir_nb"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-116-47b3cfc755c0>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



ADA_fir

```
Er_List_fir_ada=[]
for i in range(0,35):
    col1=Top_features4[i]
    col2=Top_features_fr5[i]
    Adv_Test_fir5.loc[Adv_Test_fir5.Label == 1, col1] =1
    Adv_Test_fir5.loc[Adv_Test_fir5.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fir5.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fir5['Label']
    Adv_y_pred_Full_Model = ada_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fir_ada.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model),4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fir_ada: ',er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

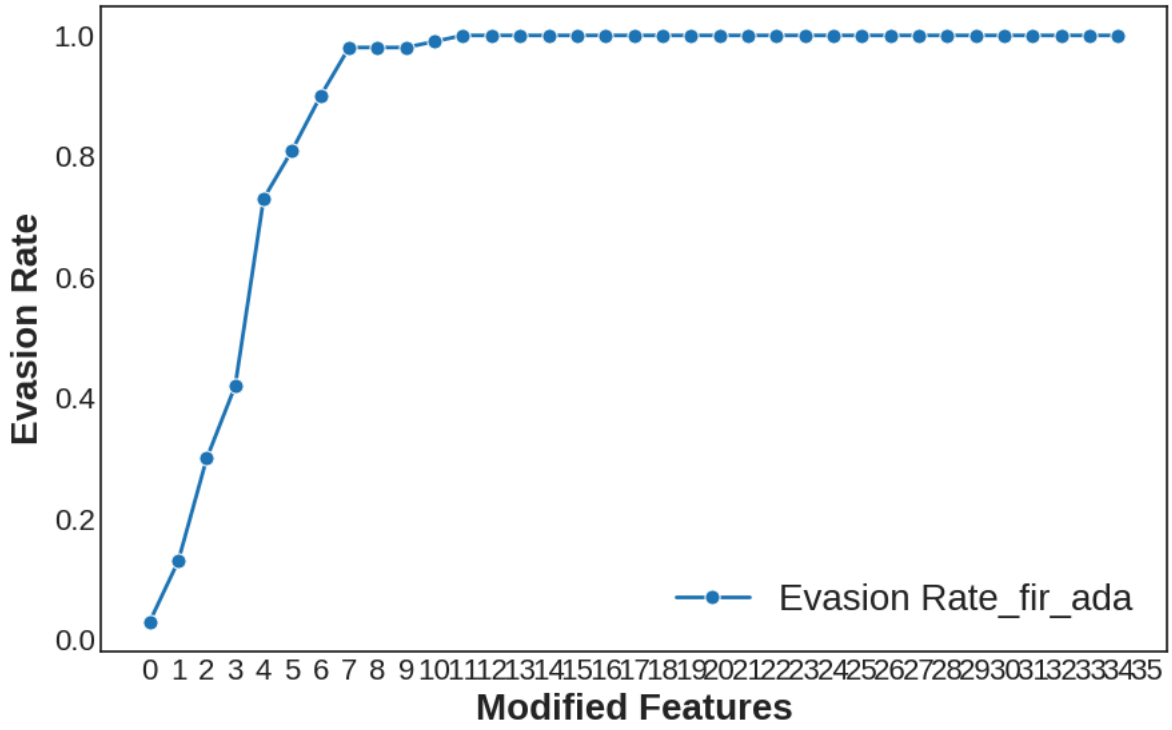
```
Accuracy     : 0.9765
Precision: 0.979
Recall: 0.974
F-Measure: 0.977
CM [[3513   74]
 [  95 3518]]
-------------------------------------------------
EVASION RATE_fir_ada:  0.03
-------------------------------------------------
**********************************************************************
Accuracy     : 0.9253
Precision: 0.977
Recall: 0.872
F-Measure: 0.921
CM [[3513   74]
 [ 464 3149]]
-------------------------------------------------
```

```
EVASION RATE_fir_ada:  0.13
--------------------------------------------------
**********************************************************************
Accuracy     : 0.8386
Precision: 0.972
Recall: 0.699
F-Measure: 0.813
CM [[3513   74]
 [1088 2525]]
--------------------------------------------------
EVASION RATE_fir_ada:  0.3
--------------------------------------------------
**********************************************************************
Accuracy     : 0.7796
Precision: 0.966
Recall: 0.581
F-Measure: 0.726
CM [[3513   74]
 [1513 2100]]
--------------------------------------------------
EVASION RATE_fir_ada:  0.42
--------------------------------------------------
**********************************************************************
Accuracy     : 0.6211
Precision: 0.928
Recall: 0.265
F-Measure: 0.413
CM [[3513   74]
 [2654  959]]
--------------------------------------------------
EVASION RATE_fir_ada:  0.73
--------------------------------------------------
**********************************************************************
Accuracy     : 0.5817
Precision: 0.901
Recall: 0.187
F-Measure: 0.309
CM [[3513   74]
 [2938  675]]
--------------------------------------------------
EVASION RATE_fir_ada:  0.81
```

```
Less_ER_fir5=Er_List_fir_ada[0:35]
```

```
df = pd.DataFrame (Less_ER_fir5,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fir_ada"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-119-c2d298111864>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



KNN_fir

```
Er_List_fir_knn=[]
for i in range(0,35):
    col1=Top_features5[i]
    col2=Top_features_fr6[i]
    Adv_Test_fir6.loc[Adv_Test_fir6.Label == 1, col1] =1
    Adv_Test_fir6.loc[Adv_Test_fir6.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fir6.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fir6['Label']
    Adv_y_pred_Full_Model = knn_full_model.predict(Adv_x_test)
    #-----------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-----------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fir_knn.append(er)
    #-----------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model),4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-----------------------------------------------')
    print('EVASION RATE_fir_knn: ',er)
    print('-----------------------------------------------')
    print("*************************************************************************")
```

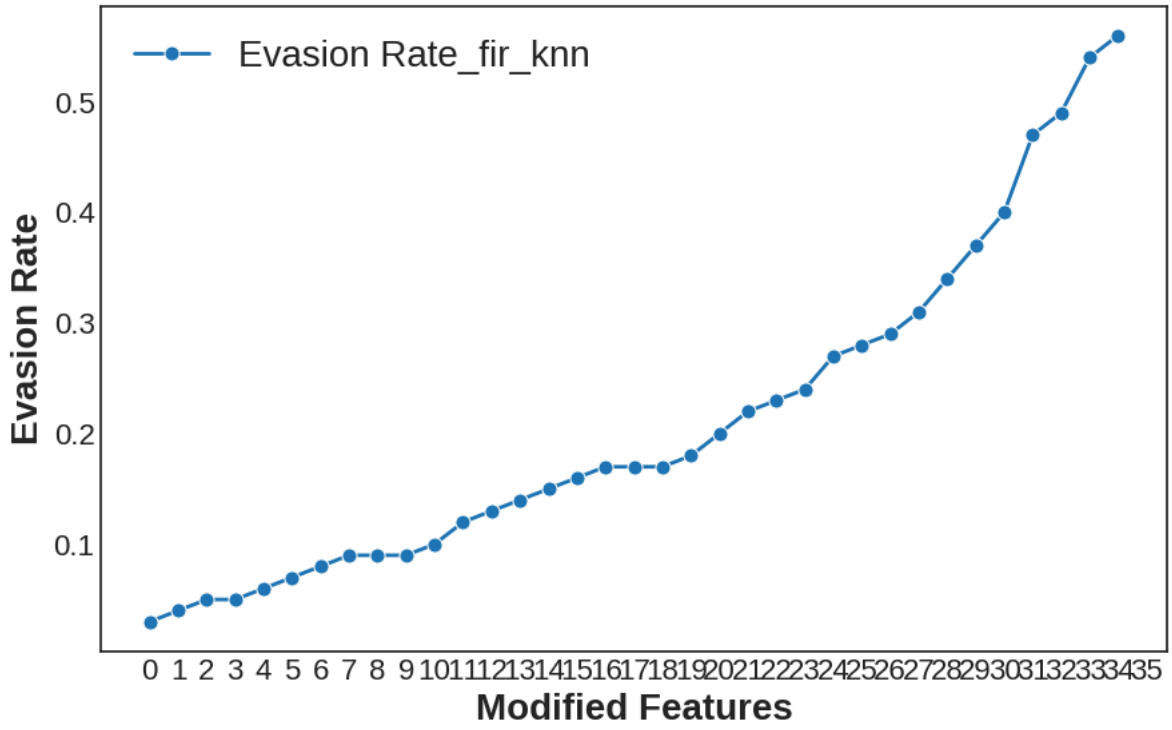
```
Accuracy     : 0.9756
Precision: 0.983
Recall: 0.968
F-Measure: 0.975
CM [[3526   61]
 [ 115 3498]]
-------------------------------------------------
EVASION RATE_fir_knn:  0.03
-------------------------------------------------
*****************************************************************
Accuracy     : 0.9721
Precision: 0.983
Recall: 0.961
F-Measure: 0.972
CM [[3526   61]
 [ 140 3473]]
-------------------------------------------------
```

```
EVASION RATE_fir_knn:  0.04
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9686
Precision: 0.983
Recall: 0.954
F-Measure: 0.968
CM [[3526   61]
 [ 165 3448]]
--------------------------------------------------
EVASION RATE_fir_knn:  0.05
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9665
Precision: 0.983
Recall: 0.950
F-Measure: 0.966
CM [[3526   61]
 [ 180 3433]]
--------------------------------------------------
EVASION RATE_fir_knn:  0.05
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9615
Precision: 0.982
Recall: 0.940
F-Measure: 0.961
CM [[3526   61]
 [ 216 3397]]
--------------------------------------------------
EVASION RATE_fir_knn:  0.06
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9578
Precision: 0.982
Recall: 0.933
F-Measure: 0.957
CM [[3526   61]
 [ 243 3370]]
--------------------------------------------------
EVASION RATE_fir_knn:  0.07
```

```
Less_ER_fir6=Er_List_fir_knn[0:35]
```

```
df = pd.DataFrame (Less_ER_fir6,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fir_knn"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-122-828033ce0863>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



Optimised KNN - FIR

```
Er_List_fir_knn_optimised=[]
for i in range(0,35):
    col1=Top_features_optimised1[i]
    col2=Top_features_optimised2[i]

    Adv_Test_optimised3.loc[Adv_Test_optimised3.Label == 1, col1] =1
    Adv_Test_optimised3.loc[Adv_Test_optimised3.Label == 1, col2] =0
    Adv_x_test=Adv_Test_optimised3.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_optimised3['Label']
    Adv_y_pred_Full_Model = knn_optimised_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fir_knn_optimised.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model),4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fir_optimised_knn: ',er)
    print('-------------------------------------------------')
    print("**************************************************************************")
```

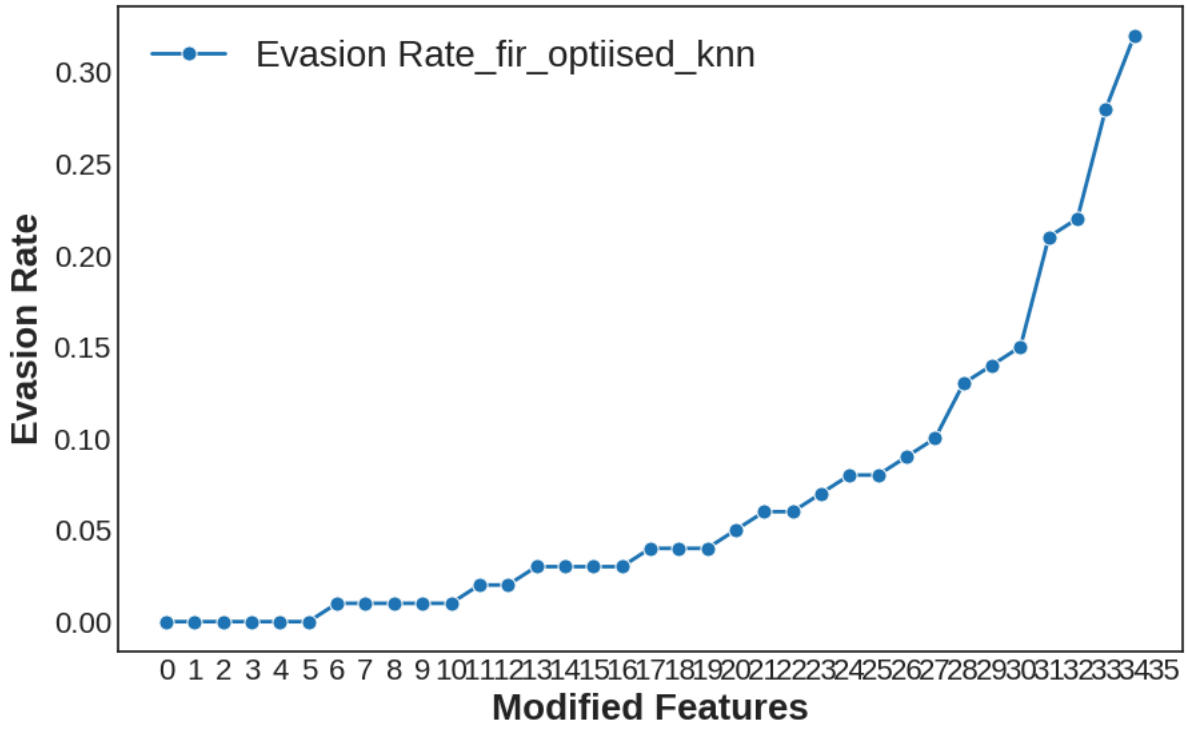
```
Accuracy     : 0.9981
Precision: 0.997
Recall: 0.999
F-Measure: 0.998
CM [[3576   11]
 [   3 3610]]
-------------------------------------------------
EVASION RATE_fir_optimised_knn:  0.0
-------------------------------------------------
**************************************************************************
Accuracy     : 0.9979
Precision: 0.997
Recall: 0.999
F-Measure: 0.998
CM [[3576   11]
 [   4 3609]]
```

```
--------------------------------------------------
EVASION RATE_fir_optimised_knn:  0.0
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9972
Precision: 0.997
Recall: 0.998
F-Measure: 0.997
CM [[3576   11]
 [   9 3604]]
--------------------------------------------------
EVASION RATE_fir_optimised_knn:  0.0
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9968
Precision: 0.997
Recall: 0.997
F-Measure: 0.997
CM [[3576   11]
 [  12 3601]]
--------------------------------------------------
EVASION RATE_fir_optimised_knn:  0.0
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9967
Precision: 0.997
Recall: 0.996
F-Measure: 0.997
CM [[3576   11]
 [  13 3600]]
--------------------------------------------------
EVASION RATE_fir_optimised_knn:  0.0
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9962
Precision: 0.997
Recall: 0.996
F-Measure: 0.996
CM [[3576   11]
 [  16 3597]]
--------------------------------------------------
EVASION RATE_fir_optimised_knn:  0.0
```

```
Less_ER_fir_optimised = Er_List_fir_knn_optimised[0:35]
```

```
df = pd.DataFrame (Less_ER_fir_optimised,columns=['Evasion Rate'])
# figure size in inches
rcParams['figure.figsize'] = 10.3,6.27
sns.lineplot(data=df, markers = True,markersize=8,linewidth=2.0)
#plt.gca().invert_yaxis()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend(["Evasion Rate_fir_optiised_knn"], prop={"size":20})
plt.xticks(list(range(0, 36)))
plt.style.use('seaborn-white')
plt.ylabel("Evasion Rate", size=20,weight = 'bold')
plt.xlabel("Modified Features", size=20,weight = 'bold')
plt.show()
```

```
<ipython-input-125-343c606be4be>:10: MatplotlibDeprecationWarning: The seaborn styles shipped by Matplotlib are deprecat
  plt.style.use('seaborn-white')
```



LR_fir

```
Er_List_fir_lr=[]
for i in range(0,35):
    col1=Top_features6[i]
    col2=Top_features_fr7[i]
    Adv_Test_fir7.loc[Adv_Test_fir7.Label == 1, col1] =1
    Adv_Test_fir7.loc[Adv_Test_fir7.Label == 1, col2] =0
    Adv_x_test=Adv_Test_fir7.drop(['Label'], axis=1)
    Adv_y_test=Adv_Test_fir7['Label']
    Adv_y_pred_Full_Model = lr_full_model.predict(Adv_x_test)
    #-------------------------------------------
    precision = precision_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    recall = recall_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    score = f1_score(Adv_y_test, Adv_y_pred_Full_Model, average='binary')
    con=confusion_matrix(Adv_y_test, Adv_y_pred_Full_Model.round())
    #-------------------------------------------------
    er=con[1][0]/3613
    er=round(er, 2)
    Er_List_fir_lr.append(er)
    #-------------------------------------------------
    print("Accuracy     :", round(accuracy_score(Adv_y_test, Adv_y_pred_Full_Model),4))
    print('Precision: %.3f' % precision)
    print('Recall: %.3f' % recall)
    print('F-Measure: %.3f' % score)
    print('CM',con)
    print('-------------------------------------------------')
    print('EVASION RATE_fir_lr: ',er)
    print('-------------------------------------------------')
    print("*************************************************************************")
```

```
Accuracy     : 0.9908
Precision: 0.994
Recall: 0.988
F-Measure: 0.991
CM [[3565   22]
 [  44 3569]]
-------------------------------------------------
EVASION RATE_fir_lr:  0.01
-------------------------------------------------
*************************************************************************
Accuracy     : 0.9829
Precision: 0.994
Recall: 0.972
F-Measure: 0.983
CM [[3565   22]
 [ 101 3512]]
-------------------------------------------------
```

```
EVASION RATE_fir_lr:  0.03
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9785
Precision: 0.994
Recall: 0.963
F-Measure: 0.978
CM [[3565   22]
 [ 133 3480]]
--------------------------------------------------
EVASION RATE_fir_lr:  0.04
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9583
Precision: 0.993
Recall: 0.923
F-Measure: 0.957
CM [[3565   22]
 [ 278 3335]]
--------------------------------------------------
EVASION RATE_fir_lr:  0.08
--------------------------------------------------
**********************************************************************
Accuracy     : 0.9329
Precision: 0.993
Recall: 0.872
F-Measure: 0.929
CM [[3565   22]
 [ 461 3152]]
--------------------------------------------------
EVASION RATE_fir_lr:  0.13
--------------------------------------------------
**********************************************************************
Accuracy     : 0.8544
```