Laboratory Practice III – Practical 2

Name: Anuj Sachin Dhole
Roll No: B21042
Class: BE CE A
Subject: Laboratory Practice III (Machine Learning)

Assignment 2 – Email Spam Detection using KNN and SVM

In [16]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, recall_score, precision_score
from sklearn.svm import SVC
```

In [17]:

```
# Load the dataset
df = pd.read_csv('emails.csv')

X = df.drop(columns=['Email No.', 'spam'])  #feature

y = df['spam'] #target
```

In [18]:

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

Out[18]:

```
array([[-0.56544926, -0.64908256, -0.2938948 , ..., -0.32904848,
        -0.07097072, -0.63913749],
       [ 0.11575699,  0.71450797,  1.33733653, ...,  0.03067224,
        -0.07097072, -0.63913749],
       [-0.56544926, -0.64908256, -0.2938948 , ..., -0.32904848,
        -0.07097072, -0.63913749],
       ...,
       [-0.56544926, -0.64908256, -0.2938948 , ..., -0.32904848,
        -0.07097072,  1.56460858],
       [-0.3951477 ,  0.0851585 , -0.2938948 , ...,  0.03067224,
        -0.07097072,  1.56460858],
       [ 1.30786793,  1.86831533, -0.0102024 , ..., -0.32904848,
        -0.07097072, -0.63913749]])
```

In [19]:

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

K-Nearest Neighbours

In [20]:

```
# Initialize KNN
knn = KNeighborsClassifier(n_neighbors=5)

# Train KNN
knn.fit(X_train, y_train)

# Predict on test set
y_pred = knn.predict(X_test)

# Evaluate performance
print("KNN Accuracy:", accuracy_score(y_test, y_pred))
error_rate = 1 - accuracy_score(y_test, y_pred)
print("Error Rate:", error_rate)

print("KNN Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

precision = precision_score(y_test, y_pred)
print("Precision:", precision)
recall = recall_score(y_test, y_pred)
print("Recall:", recall)
```

```
KNN Accuracy: 0.9971014492753624
Error Rate: 0.0028985507246376274
KNN Confusion Matrix:
 [[1024    0]
 [   3    8]]
Precision: 1.0
Recall: 0.7272727272727273
```

Linear SVM

In [6]:

```
# Initialize SVM with linear kernel
svm = SVC(kernel='linear', random_state=42)

# Train SVM
svm.fit(X_train, y_train)
```

Out[6]:

```
SVC(kernel='linear', random_state=42)
```

Non-linear SVM

In [22]:

```
y_pred = svm.predict(X_test)

print("SVM Accuracy:", accuracy_score(y_test, y_pred))

print("SVM Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
SVM Accuracy: 0.9942028985507246
SVM Confusion Matrix:
 [[1020    3    1]
 [   2    9    0]
 [   0    0    0]]
```

In [8]:

```
# SVM with polynomial kernel
svm_poly = SVC(kernel='poly', degree=3, coef0=1, C=1, random_state=42)


svm_poly.fit(X_train, y_train)

y_pred = svm_poly.predict(X_test)

# Accuracy
print("Polynomial SVM Accuracy:", accuracy_score(y_test, y_pred))

# Confusion Matrix
print("Polynomial SVM Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Polynomial SVM Accuracy: 0.9951690821256038
Polynomial SVM Confusion Matrix:
 [[1022    2]
 [   3    8]]
```