import pandas as pd Email

import numpy as np import seaborn as sns

import matplotlib.pyplot as plt import warnings

warnings.filterwarnings('ignore')

from sklearn.model\_selection import train\_test\_split from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier from sklearn import metrics

from sklearn.preprocessing import scale

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

df = pd.read\_csv("emails.csv")

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

df.info()

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

df.head()

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

df.isnull().sum()

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

df.columns

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

df.dropna(inplace=True)

df.drop(['Email No.'] , axis=1, inplace=True)

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

df.head()

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

X = df.drop(['Prediction'], axis=1) # Features y = df['Prediction'] # Target variable

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

print("\nFeatures (X) and Target (y) separated:") print(X.head())

print(y.head())

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

# Normalize the feature set X = scale(X)

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

# Split the data into training and testing sets (70% training, 30% testing)7

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

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

print("\nTraining and Testing data shapes:") print("X\_train shape:", X\_train.shape)

print("X\_test shape:", X\_test.shape) print("y\_train shape:", y\_train.shape) print("y\_test shape:", y\_test.shape)

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

# KNN Classifier

knn = KNeighborsClassifier(n\_neighbors=3) knn.fit(X\_train, y\_train)

y\_pred\_knn = knn.predict(X\_test)

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

# KNN Evaluation

print("\nKNN Classifier Results:")

print("Predictions:", y\_pred\_knn) # 1 for spam, 0 for not spam

print("KNN Accuracy:", metrics.accuracy\_score(y\_test, y\_pred\_knn))

print("KNN Confusion Matrix:\n", metrics.confusion\_matrix(y\_test, y\_pred\_knn))

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

# SVM Classifier

svm = SVC() # Support Vector Machine model svm.fit(X\_train, y\_train)

y\_pred\_svm = svm.predict(X\_test)

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

# SVM Evaluation

print("\nSVM Classifier Results:") print("Predictions:", y\_pred\_svm)

print("SVM Accuracy:", metrics.accuracy\_score(y\_test, y\_pred\_svm))

print("SVM Confusion Matrix:\n", metrics.confusion\_matrix(y\_true=y\_test, y\_pred=y\_pred\_svm))

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

print("\nKNN Classifier - Additional Metrics:")

print("Precision:", metrics.precision\_score(y\_test, y\_pred\_knn)) print("Recall:", metrics.recall\_score(y\_test, y\_pred\_knn))

print("F1-Score:", metrics.f1\_score(y\_test, y\_pred\_knn))

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

print("\nSVM Classifier - Additional Metrics:")

print("Precision:", metrics.precision\_score(y\_test, y\_pred\_svm)) print("Recall:", metrics.recall\_score(y\_test, y\_pred\_svm))

print("F1-Score:", metrics.f1\_score(y\_test, y\_pred\_svm))

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