UCS2612 Machine Learning Laboratory

A4 : Classification of Email spam and MNIST data using Support Vector Machines

Name : **Anandh K**

Reg No : **3122 21 5001 009**

# ----------------------------------------------------------------------------------------------

4.a. Develop a python program to classify Emails as Spam or Ham using Support Vector Machine (SVM) Model. Visualize the features from the dataset and interpret the results obtained by the model using Matplotlib library.

**Code and Output:**

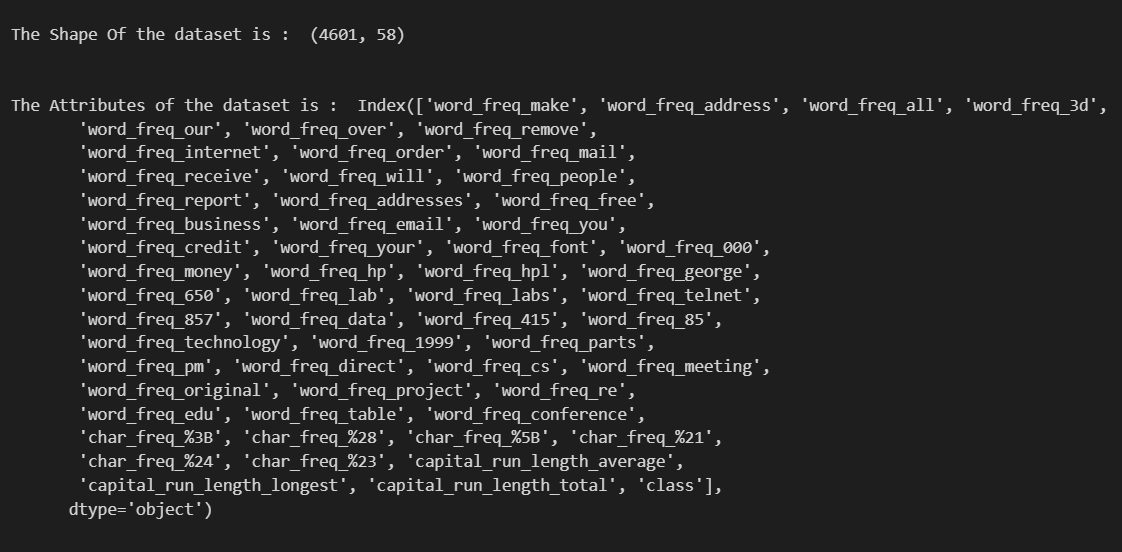
import numpy as np import pandas as pd

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

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

import matplotlib.pyplot as plt import seaborn as sns

from sklearn.decomposition import PCA



Loading the Dataset Email\_df=pd.read\_csv("spambase\_csv.csv")

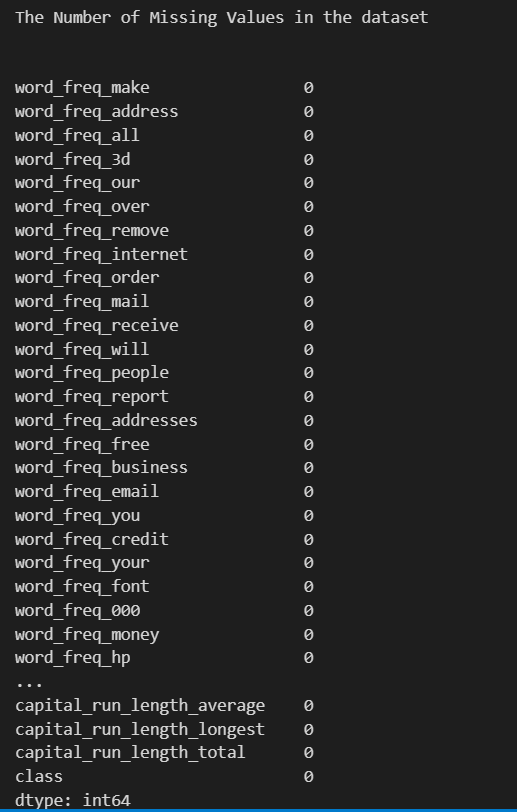
print("\n\nThe Shape Of the dataset is : ",Email\_df.shape)

print("\n\nThe Attributes of the dataset is : ",Email\_df.columns)

Pre-Processing the data (Handling missing values)

print("The Number of Missing Values in the dataset\n")

Email\_df.isnull().sum()



Feature Engineering Techniques

X = Email\_df.drop(columns=['class'])

y = Email\_df['class']

pca = PCA(n\_components=15) X\_pca = pca.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_pca, y, test\_size=0.2, random\_state=42)

Split the data into training, testing and validation sets.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2,

random\_state=42)

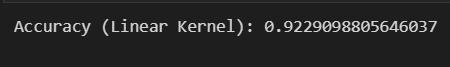
Train the model. Test the model. Measure the performance of the trained model.

**1) Linear Kernal**

svm\_linear = SVC(kernel='linear') svm\_linear.fit(X\_train, y\_train) y\_pred = svm\_linear.predict(X\_test)

accuracy\_linear = svm\_linear.score(X\_test, y\_test)

print("Accuracy (Linear Kernel):", accuracy\_linear)

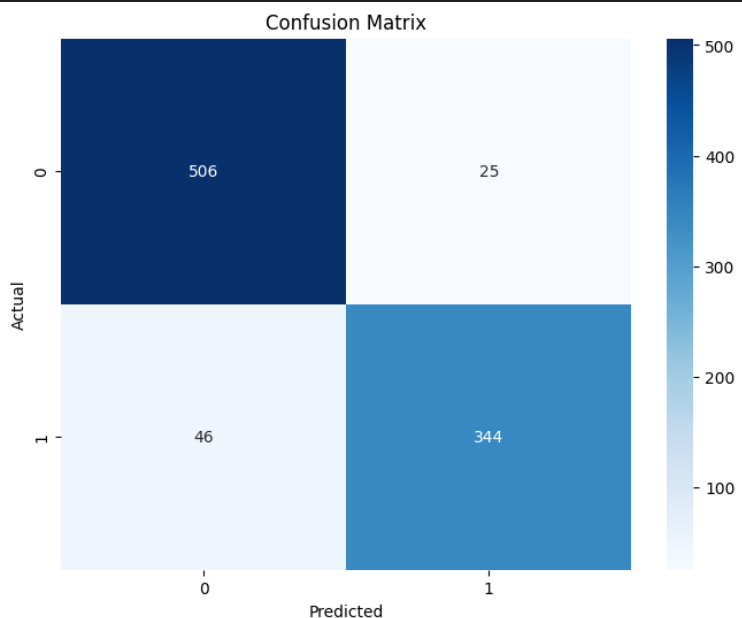


conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues") plt.xlabel("Predicted")

plt.ylabel("Actual") plt.title("Confusion Matrix")

plt.show()



## Polynomial kernal

svm\_poly = SVC(kernel='poly') svm\_poly.fit(X\_train, y\_train) y\_pred = svm\_poly.predict(X\_test)

accuracy\_poly = svm\_poly.score(X\_test, y\_test)

print("Accuracy (Polynomial Kernel):", accuracy\_poly)



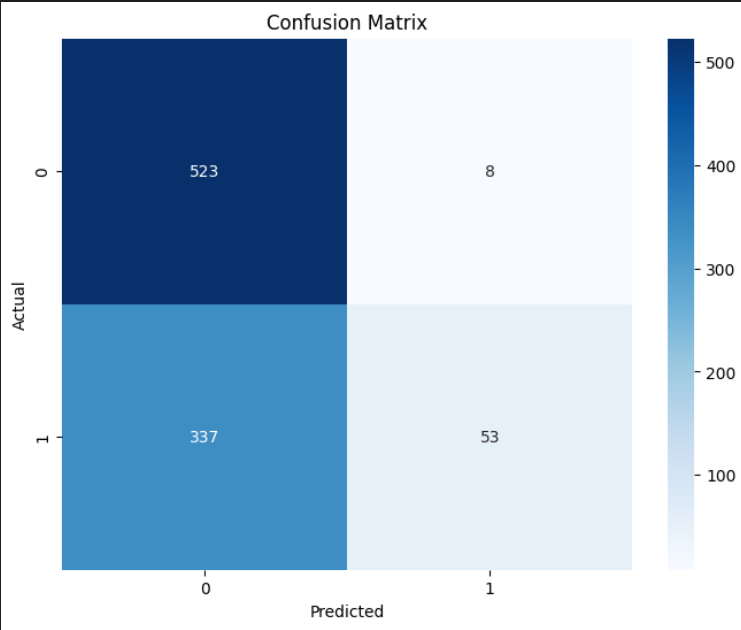
conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix") plt.show()



1. **RBF**

svm\_rbf = SVC(kernel='rbf') svm\_rbf.fit(X\_train, y\_train) y\_pred = svm\_rbf.predict(X\_test)

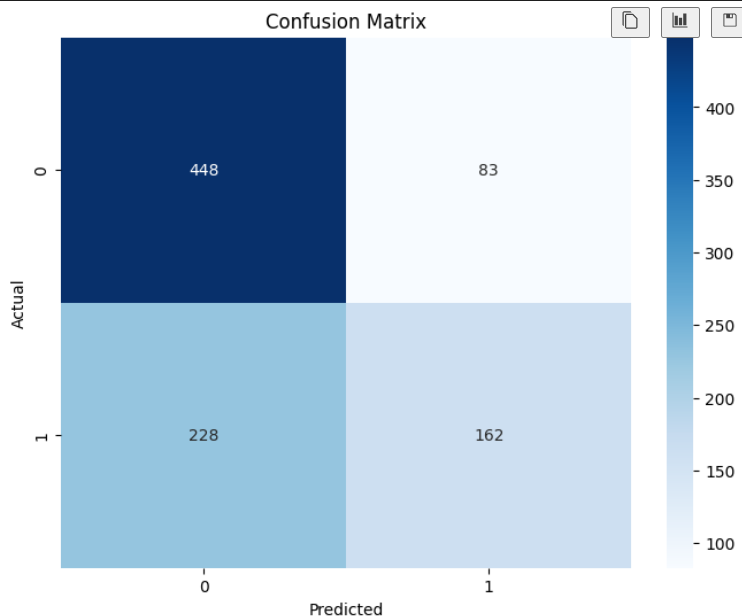
accuracy\_rbf = svm\_rbf.score(X\_test, y\_test)

print("Accuracy (RBF Kernel):", accuracy\_rbf)

A black background with white text  Description automatically generated

conf\_matrix = confusion\_matrix(y\_test, y\_pred) plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues")



plt.xlabel("Predicted") plt.ylabel("Actual") plt.title("Confusion Matrix")

plt.show()

## Sigmoid kernal

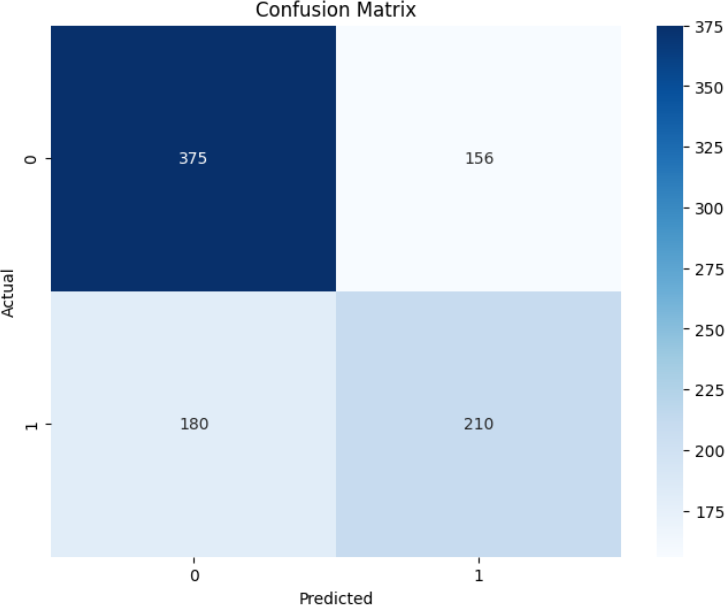
svm\_sigmoid = SVC(kernel='sigmoid') svm\_sigmoid.fit(X\_train, y\_train) y\_pred = svm\_sigmoid.predict(X\_test)

accuracy\_sigmoid = svm\_sigmoid.score(X\_test, y\_test)

print("Accuracy (Sigmoid Kernel):", accuracy\_sigmoid)



conf\_matrix = confusion\_matrix(y\_test, y\_pred)



plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues") plt.xlabel("Predicted")

plt.ylabel("Actual") plt.title("Confusion Matrix")

plt.show()

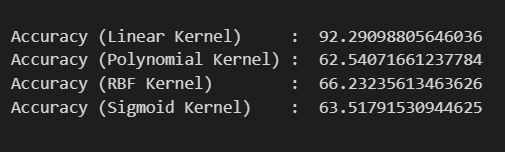
# Conclusion

print("\n\nAccuracy (Linear Kernel)

: ", accuracy\_linear\*100)

print("Accuracy (Polynomial Kernel) : ", accuracy\_poly\*100)

print("Accuracy (RBF Kernel) : ", accuracy\_rbf\*100) print("Accuracy (Sigmoid Kernel) : ", accuracy\_sigmoid\*100)



**From the among results we can easily understand that the linear kernal works best for the Email Spam ham Detection using the SVM wi the nearly 92 % Accuracy comparing than all other kernals.**

**4.B )** This is a database of 70,000 handwritten digits (10 class labels) with each example represented as an image of 28 x 28 gray-scale pixels. Develop a python program to recognize the digits using Support Vector Machine (SVM) Model. Visualize the features from the dataset and interpret the results obtained by the model using Matplotlib library

**Code and Output:**

Importing The Necessary Libraries

import cv2

import numpy as np

from skimage import io, color, exposure, feature from skimage.filters import gaussian

from skimage.segmentation import slic

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

from sklearn.metrics import accuracy\_score

Load The Dataset and Spliting the Data for Training and Testing

import numpy as np

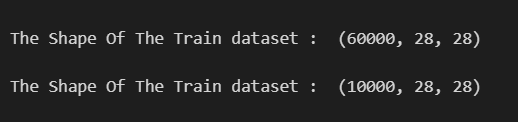
from tensorflow.keras import datasets

(x\_train, y\_train), (x\_test, y\_test) = datasets.mnist.load\_data()

Exploratory Data Analysis

print("\nThe Shape Of The Train dataset : ",x\_train.shape)

print("\nThe Shape Of The Train dataset : ",x\_test.shape)



x\_train = x\_train.reshape(x\_train.shape[0], -1) x\_test = x\_test.reshape(x\_test.shape[0], -1)

x\_train = x\_train.astype('float32') / 255.0 x\_test = x\_test.astype('float32') / 255.0

**the SVM model with different kernel functions (use linear, rbf, polynomial, sigmoid)**

## Linear kernal

svm\_linear = SVC(kernel='linear') svm\_linear.fit(x\_train, y\_train) y\_pred = svm\_linear.predict(x\_test)

accuracy\_linear = svm\_linear.score(x\_test, y\_test)

print("Accuracy (Linear Kernel):", accuracy\_linear)



1. **Polynomial kernal**

svm\_poly = SVC(kernel='poly')

svm\_poly.fit(x\_train, y\_train)

y\_pred = svm\_poly.predict(x\_test) accuracy\_poly = svm\_poly.score(x\_test, y\_test)

print("Accuracy (Polynomial Kernel):", accuracy\_poly)



**3) RBF Kernal**

svm\_rbf = SVC(kernel='rbf') svm\_rbf.fit(x\_train, y\_train) y\_pred = svm\_rbf.predict(x\_test)

accuracy\_rbf = svm\_rbf.score(x\_test, y\_test)

print("Accuracy (RBF Kernel):", accuracy\_rbf)

A black background with white text  Description automatically generated

**4) Sigmoid Kernal**

svm\_sigmoid = SVC(kernel='sigmoid') svm\_sigmoid.fit(x\_train, y\_train) y\_pred = svm\_sigmoid.predict(x\_test)

accuracy\_sigmoid = svm\_sigmoid.score(x\_test, y\_test)

print("Accuracy (Sigmoid Kernel):", accuracy\_sigmoid)

A black and white text  Description automatically generated

## Conclusion :

print("Linear Kernel Accuracy

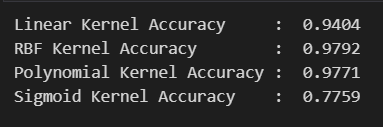
print("RBF Kernel Accuracy

: ", accuracy\_linear)

: ", accuracy\_rbf)

print("Polynomial Kernel Accuracy : ", accuracy\_poly)

print("Sigmoid Kernel Accuracy : ", accuracy\_sigmoid)



**From The above results all the kernal except the sigmoid kernal working very well for the MNIST dataset with the more that 95 % accuracy.**

## Github Link For The Project :

[**https://github.com/Anandh-007/Machine-learning-lab**](file:///C:\Users\anand\Downloads\ana.docx)

## Learning Outcome:

* + Better Understanding about the Support vector machine
  + SVM has the different kernals called

## Linear

* + - **Polinomial**

## RBF

* + - **Sigmoid**
  + I did the Email Spam and Ham using the SVM different Kernal
  + Except the linear kernal all other kernals works not well for the Email spam and Ham
  + For a MNIST Dataset we used the SVM with different Kernals
  + All the kernal works well for the MNIST dataset.
  + From the above results I conclude **that The Support vector Machine ( SVM ) Works well for the Image Dataset.**