UCS2612 Machine Learning Laboratory

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

Name : **BALAJI V A**

Roll No : **3122 21 5001 018**

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)
```

```
The Shape Of the dataset is :  (4601, 58)


The Attributes of the dataset is :  Index(['word_freq_make', 'word_freq_address', 'word_freq_all', 'word_freq_3d',
       'word_freq_our', 'word_freq_over', 'word_freq_remove',
       'word_freq_internet', 'word_freq_order', 'word_freq_mail',
       'word_freq_receive', 'word_freq_will', 'word_freq_people',
       'word_freq_report', 'word_freq_addresses', 'word_freq_free',
       'word_freq_business', 'word_freq_email', 'word_freq_you',
       'word_freq_credit', 'word_freq_your', 'word_freq_font', 'word_freq_000',
       'word_freq_money', 'word_freq_hp', 'word_freq_hpl', 'word_freq_george',
       'word_freq_650', 'word_freq_lab', 'word_freq_labs', 'word_freq_telnet',
       'word_freq_857', 'word_freq_data', 'word_freq_415', 'word_freq_85',
       'word_freq_technology', 'word_freq_1999', 'word_freq_parts',
       'word_freq_pm', 'word_freq_direct', 'word_freq_cs', 'word_freq_meeting',
       'word_freq_original', 'word_freq_project', 'word_freq_re',
       'word_freq_edu', 'word_freq_table', 'word_freq_conference',
       'char_freq_%3B', 'char_freq_%28', 'char_freq_%5B', 'char_freq_%21',
       'char_freq_%24', 'char_freq_%23', 'capital_run_length_average',
       'capital_run_length_longest', 'capital_run_length_total', 'class'],
      dtype='object')
```

```
Pre-Processing the data (Handling missing values)
```

```
print("The Number of Missing Values in the dataset\n")
Email_df.isnull().sum()
```

```
The Number of Missing Values in the dataset


word_freq_make                0
word_freq_address             0
word_freq_all                 0
word_freq_3d                  0
word_freq_our                 0
word_freq_over                0
word_freq_remove              0
word_freq_internet            0
word_freq_order               0
word_freq_mail                0
word_freq_receive             0
word_freq_will                0
word_freq_people              0
word_freq_report              0
word_freq_addresses           0
word_freq_free                0
word_freq_business            0
word_freq_email               0
word_freq_you                 0
word_freq_credit              0
word_freq_your                0
word_freq_font                0
word_freq_000                 0
word_freq_money               0
word_freq_hp                  0
...
capital_run_length_average    0
capital_run_length_longest    0
capital_run_length_total      0
class                         0
dtype: int64
```

```
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)
```

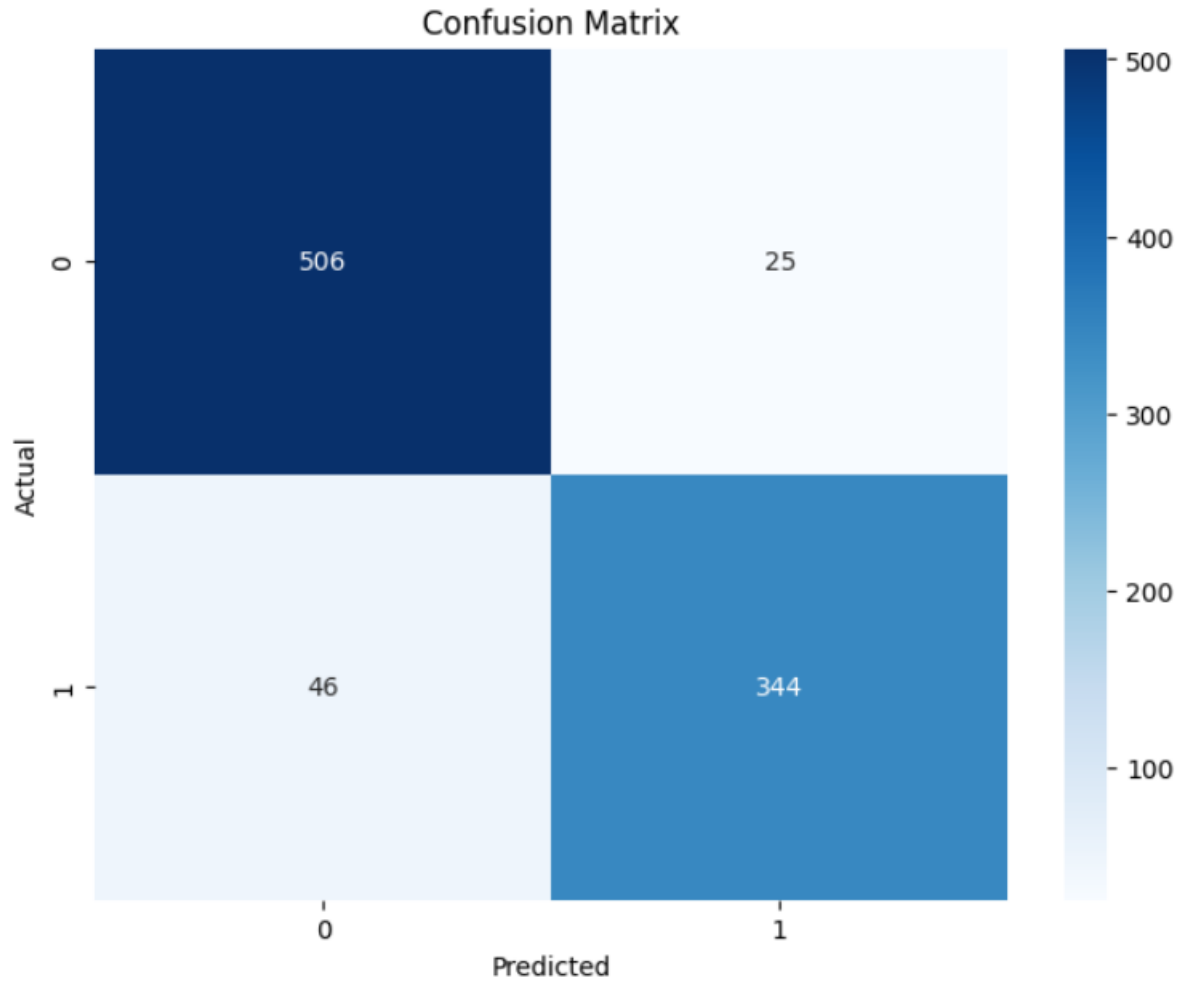
Train the model. Test the model. Measure the performance of the trained model.

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)
```

```
Accuracy (Linear Kernel): 0.9229098805646037
```

```
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()
```



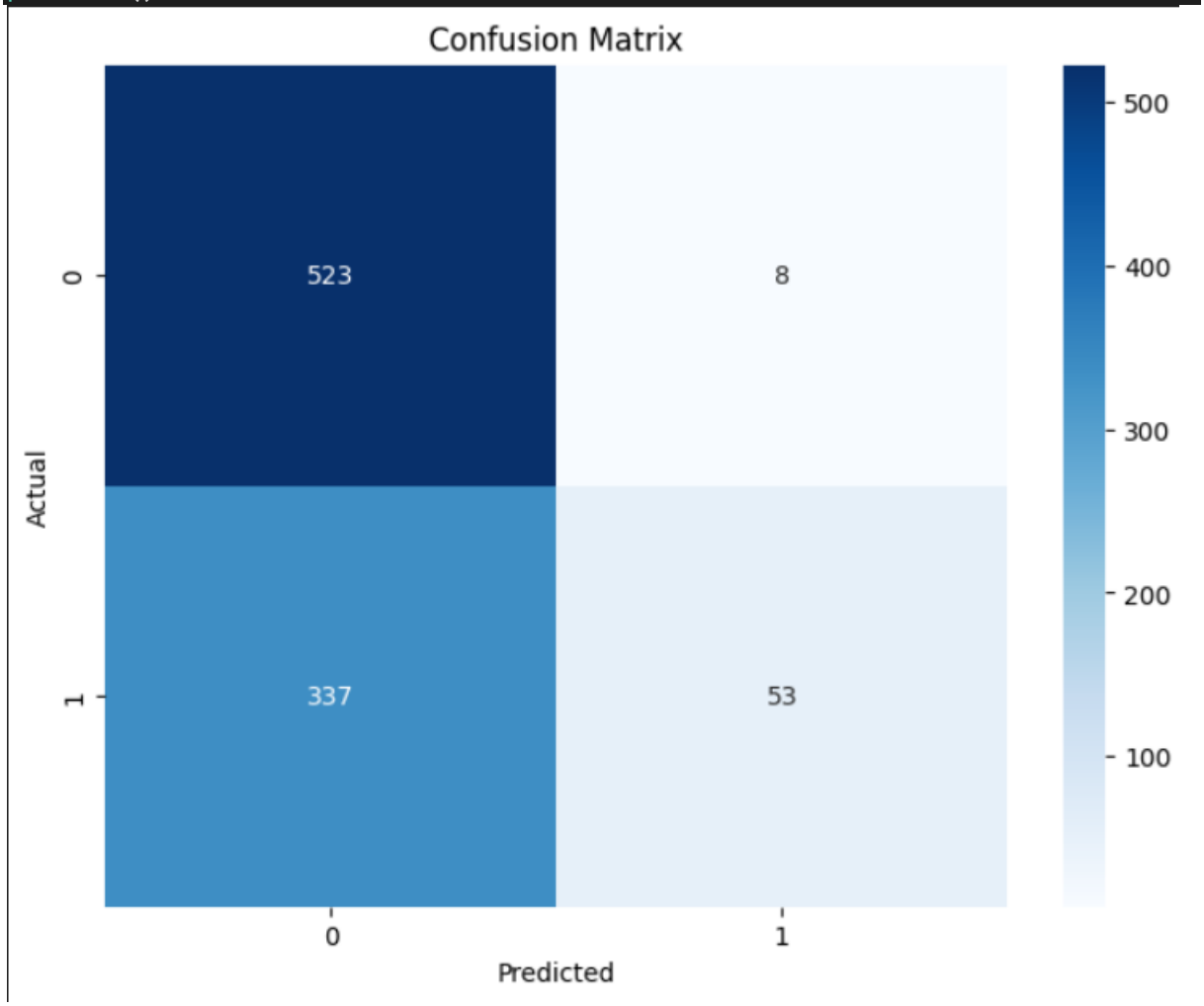
2) Polynomial kernal

```
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)
```

```
Accuracy (Polynomial Kernel): 0.6254071661237784
```

```
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()
```



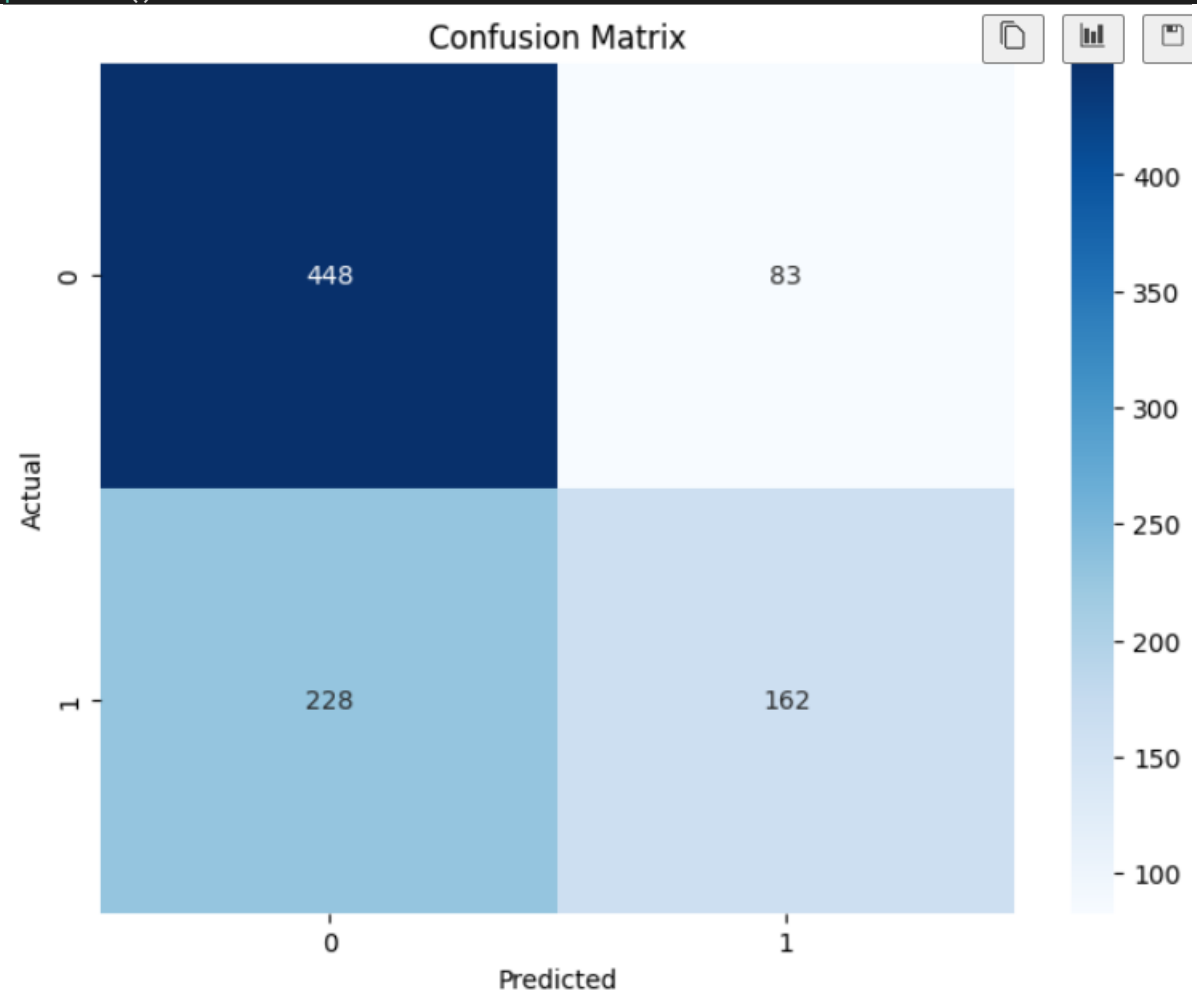
3) RBF

```
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)
```

```
Accuracy (RBF Kernel): 0.6623235613463626
```

```
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()
```



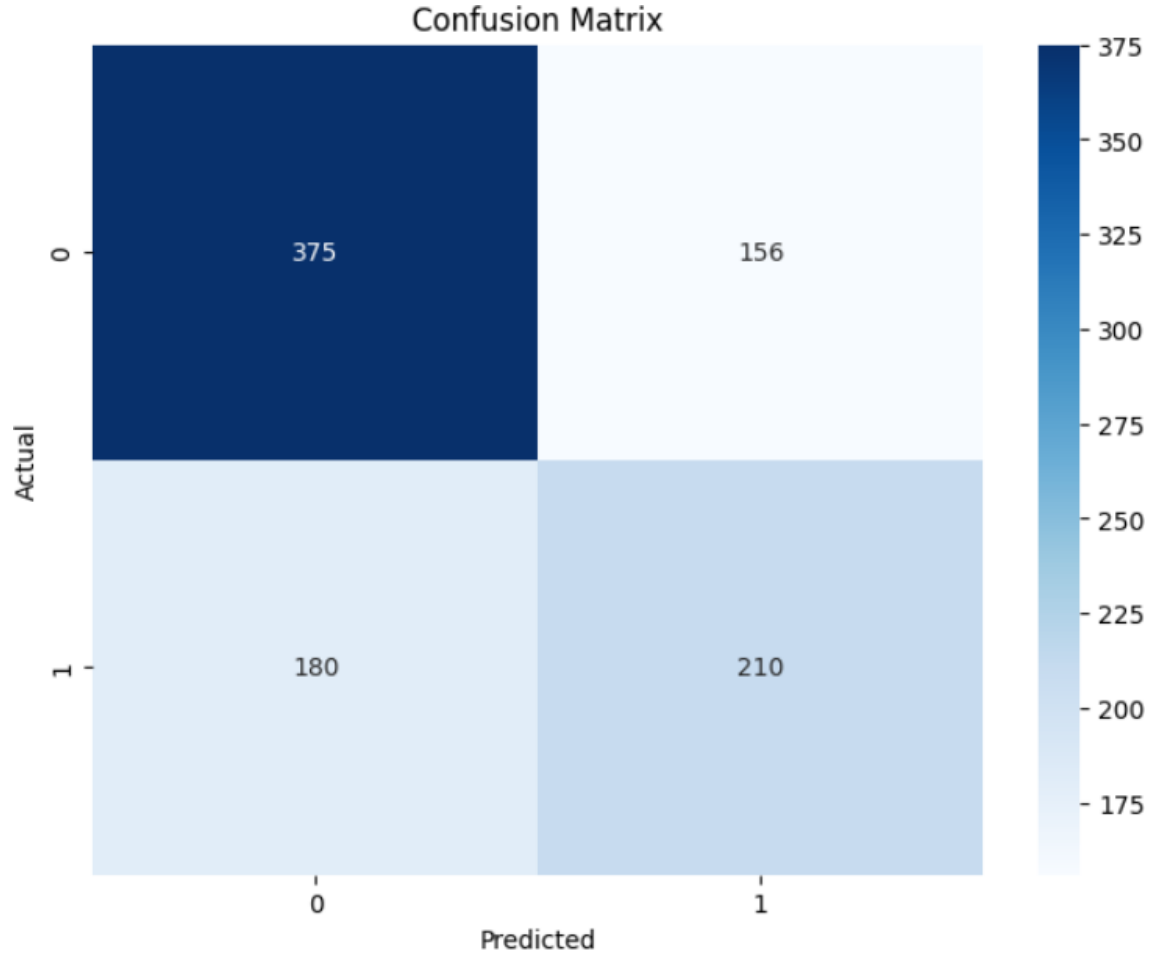
4) Sigmoid kernal

```
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)
```

```
Accuracy (Sigmoid Kernel): 0.6351791530944625
```

```
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)
```

```
Accuracy (Linear Kernel)     :  92.29098805646036
Accuracy (Polynomial Kernel) :  62.54071661237784
Accuracy (RBF Kernel)        :  66.23235613463626
Accuracy (Sigmoid Kernel)    :  63.51791530944625
```

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)
```

```
The Shape Of The Train dataset :  (60000, 28, 28)

The Shape Of The Train dataset :  (10000, 28, 28)
```

```
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)
```

```
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)
```

```
Accuracy (Linear Kernel): 0.9404
```

```
2) 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)
```

```
Accuracy (Polynomial Kernel): 0.9771
```

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)
```

```
Accuracy (RBF Kernel): 0.9792
```

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)
```

```
Accuracy (Sigmoid Kernel): 0.7759
```

Conclusion :

```
print("Linear Kernel Accuracy      : ", accuracy_linear)
print("RBF Kernel Accuracy         : ", accuracy_rbf)
print("Polynomial Kernel Accuracy : ", accuracy_poly)
print("Sigmoid Kernel Accuracy     : ", accuracy_sigmoid)
```

```
Linear Kernel Accuracy     :  0.9404
RBF Kernel Accuracy        :  0.9792
Polynomial Kernel Accuracy :  0.9771
Sigmoid Kernel Accuracy    :  0.7759
```

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/balaji0920/ML-Assignment-4.git

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.**