UCS2612 Machine Learning Laboratory

Name: Harini Mohan **Register No :** 3122215001029 **Class & Section:** CSE A VI

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110
(An Autonomous Institution, Affiliated to Anna University, Chennai)

UCS2612 Machine Learning Laboratory

Academic Year: 2023-2024 Even **Batch: 2021-2025**
Faculty In-charges: Y.V. Lokeswari & Nilu R Salim **VI Semester A & B**

A. No. : 4 . **Classification of Email Spam and MNIST data using Support Vector Machines**

Download the Email spam dataset from the link given below:

https://www.kaggle.com/datasets/somesh24/spambase

The "spam" concept is diverse: advertisements for products/websites, make money fast schemes, chain letters, pornography. Our collection of spam e-mails came from our postmaster and individuals who had filed spam. Our collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

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. **[CO1, K3]**

Download the MNIST dataset from the link given below:

https://archive.ics.uci.edu/dataset/683/mnist+database+of+handwritten+digits

THE MNIST DATABASE: http://yann.lecun.com/exdb/mnist/

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. **[CO1, K3]**

Use the following steps to do implementation:

1. Loading the dataset.
2. Pre-Processing the data (Handling missing values, Encoding, Normalization, Standardization).
3. Exploratory Data Analysis.
4. Feature Engineering Techniques.
5. Split the data into training, testing and validation sets.
6. Train the model.
7. Test the model.
8. Measure the performance of the trained model.
9. Represent the results using graphs.

Aim:

To classify email as spam or ham using SVM ML model and SVM for MNIST data.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split



df = pd.read_csv("archive/spambase_csv.csv")
df.info()
df.shape
df.head()
df.isna().sum()
df.isnull().sum()
x=df.drop(['class'],axis=1) #------->its dropping class colum....since other columns are fearues column
y=df['class']   #------>this is to seperate target column from the rest

import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(x)

# Create a scatter plot
```

```
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis', s=10, alpha=0.5)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Email Spam/Ham Classification (PCA)')
plt.colorbar(label='Class')
plt.show()
x_train, x_test, y_train, y_test = train_test_split(x,y,random_state=11,test_size=0.2)
from sklearn import svm
from sklearn.svm import SVC
model = SVC(random_state = 0)
model.fit(x_train, y_train)
model.score(x_test,y_test)
import joblib
joblib.dump(model, 'svm_model.pkl')

loaded_model = joblib.load('svm_model.pkl')
predictions = loaded_model.predict(x_test)
print(y_test)

print(predictions)
print(len(predictions))

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, predictions)
print("Test Accuracy:", accuracy)

train_predictions = model.predict(x_train)
```

```
# Calculate the training accuracy by comparing the predicted labels with the actual labels
train_accuracy = accuracy_score(y_train, train_predictions)


print("Training Accuracy:", train_accuracy)
```

Using different Kernel functions:

```
linearSVM = svm.SVC(kernel='linear')
polynomialSVM = svm.SVC(kernel='poly', degree=3)
rbfSVM = svm.SVC(kernel='rbf')
sigmoidSVM = svm.SVC(kernel='sigmoid')
linearSVM.fit(x_train,y_train)

linear_predictions = linearSVM.predict(x_test)

polynomialSVM.fit(x_train,y_train)

poly_predictions = polynomialSVM.predict(x_test)

rbfSVM.fit(x_train,y_train)

rbf_predictions = rbfSVM.predict(x_test)

sigmoidSVM.fit(x_train,y_train)
sigmoid_predictions = sigmoidSVM.predict(x_test)



linear_accuracy = accuracy_score(y_test, linear_predictions)
poly_accuracy = accuracy_score(y_test, poly_predictions)
rbf_accuracy = accuracy_score(y_test, rbf_predictions)
sigmoid_accuracy = accuracy_score(y_test, sigmoid_predictions)
```

```
print("Linear SVM Accuracy:", linear_accuracy)
print("Polynomial SVM Accuracy:", poly_accuracy)
print("RBF SVM Accuracy:", rbf_accuracy)
print("Sigmoid SVM Accuracy:", sigmoid_accuracy)
```

TRAINING ACCURACIES

```
svm_linear_model = svm.SVC(kernel='linear')
svm_linear_model.fit(x_train, y_train)

train_predictions_linear = svm_linear_model.predict(x_train)
train_accuracy_linear = accuracy_score(y_train, train_predictions_linear)
print("Training Accuracy (Linear Kernel):", train_accuracy_linear)
svm_poly_model = svm.SVC(kernel='poly')
svm_poly_model.fit(x_train, y_train)

train_predictions_poly = svm_poly_model.predict(x_train)
train_accuracy_poly = accuracy_score(y_train, train_predictions_poly)
print("Training Accuracy (poly Kernel):", train_accuracy_poly)
svm_rbf_model = svm.SVC(kernel='rbf')
svm_rbf_model.fit(x_train, y_train)

train_predictions_rbf = svm_rbf_model.predict(x_train)
train_accuracy_rbf = accuracy_score(y_train, train_predictions_rbf)
print("Training Accuracy (rbf Kernel):", train_accuracy_rbf)
svm_sigmoid_model = svm.SVC(kernel='sigmoid')
svm_sigmoid_model.fit(x_train, y_train)

train_predictions_sigmoid = svm_sigmoid_model.predict(x_train)
train_accuracy_sigmoid = accuracy_score(y_train, train_predictions_sigmoid)
```

```
print("Training Accuracy (sigmoid Kernel):", train_accuracy_sigmoid)
print("\nTraining Accuracy\n")
print("Training Accuracy (Linear Kernel):", train_accuracy_linear)
print("Training Accuracy (poly Kernel):", train_accuracy_poly)
print("Training Accuracy (rbf Kernel):", train_accuracy_rbf)
print("Training Accuracy (sigmoid Kernel):", train_accuracy_sigmoid)



print("\n\nTesting Accuracy\n")
print("Linear SVM Accuracy:", linear_accuracy)
print("Polynomial SVM Accuracy:", poly_accuracy)
print("RBF SVM Accuracy:", rbf_accuracy)
print("Sigmoid SVM Accuracy:", sigmoid_accuracy)
from sklearn.metrics import precision_score, f1_score, roc_curve, auc

print("Other metrics:")

predictionsLinear = svm_linear_model.predict(x_test)

# Calculate precision
precisionLinear = precision_score(y_test, predictionsLinear)

# Calculate F1 score
f1Linear = f1_score(y_test, predictionsLinear)

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, predictionsLinear)
```

```
# Calculate AUC score
auc_scoreLinear = auc(fpr, tpr)


print("Linear:", precisionLinear)
print("F1 Score:", f1Linear)
print("AUC Score:", auc_scoreLinear)
print("\n\n_________________________________________\n\n")


predictionsPoly = svm_poly_model.predict(x_test)

# Calculate precision
precisionPoly = precision_score(y_test, predictionsPoly)

# Calculate F1 score
f1Poly = f1_score(y_test, predictionsPoly)

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, predictionsPoly)

# Calculate AUC score
auc_scorePoly = auc(fpr, tpr)

print("Precision:", precisionPoly)
print("F1 Score:", f1Poly)
print("AUC Score:", auc_scorePoly)

print("\n\n_________________________________________\n\n")
```

```
predictionsrbf = svm_rbf_model.predict(x_test)

# Calculate precision
precisionrbf = precision_score(y_test, predictionsrbf)

# Calculate F1 score
f1rbf = f1_score(y_test, predictionsrbf)

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, predictionsrbf)

# Calculate AUC score
auc_scorerbf = auc(fpr, tpr)

print("Precision:", precisionrbf)
print("F1 Score:", f1rbf)
print("AUC Score:", auc_scorerbf)
print("\n\n________________________________________\n\n")


predictionsSig = svm_sigmoid_model.predict(x_test)

# Calculate precision
precisionSig = precision_score(y_test, predictionsSig)

# Calculate F1 score
f1Sig = f1_score(y_test, predictionsSig)
```

```
# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, predictionsSig)


# Calculate AUC score
auc_scoreSig = auc(fpr, tpr)


print("Precision:", precisionSig)
print("F1 Score:", f1Sig)
print("AUC Score:", auc_scoreSig)
```

INFERENCE:

In Support Vector Machine (SVM) models, the kernel function plays a crucial role in transforming the input data into a higher-dimensional space, where it might be easier to classify the data using a linear decision boundary.

1. Linear kernel:

It computes the dot product between the input feature vectors, which effectively calculates the similarity between them.

2. Polynomial Kernel:

The polynomial kernel function is used to handle nonlinear relationships between the features.

It maps the data into a higher-dimensional space using polynomial functions.

3. Radial Basis Function (RBF) Kernel:

The RBF kernel, also known as the Gaussian kernel, is widely used in SVMs due to its flexibility.

It maps the data into an infinite-dimensional space using Gaussian radial basis functions.

The RBF kernel considers all possible transformations of the input data into a higher-dimensional space.

4. Sigmoid Kernel:

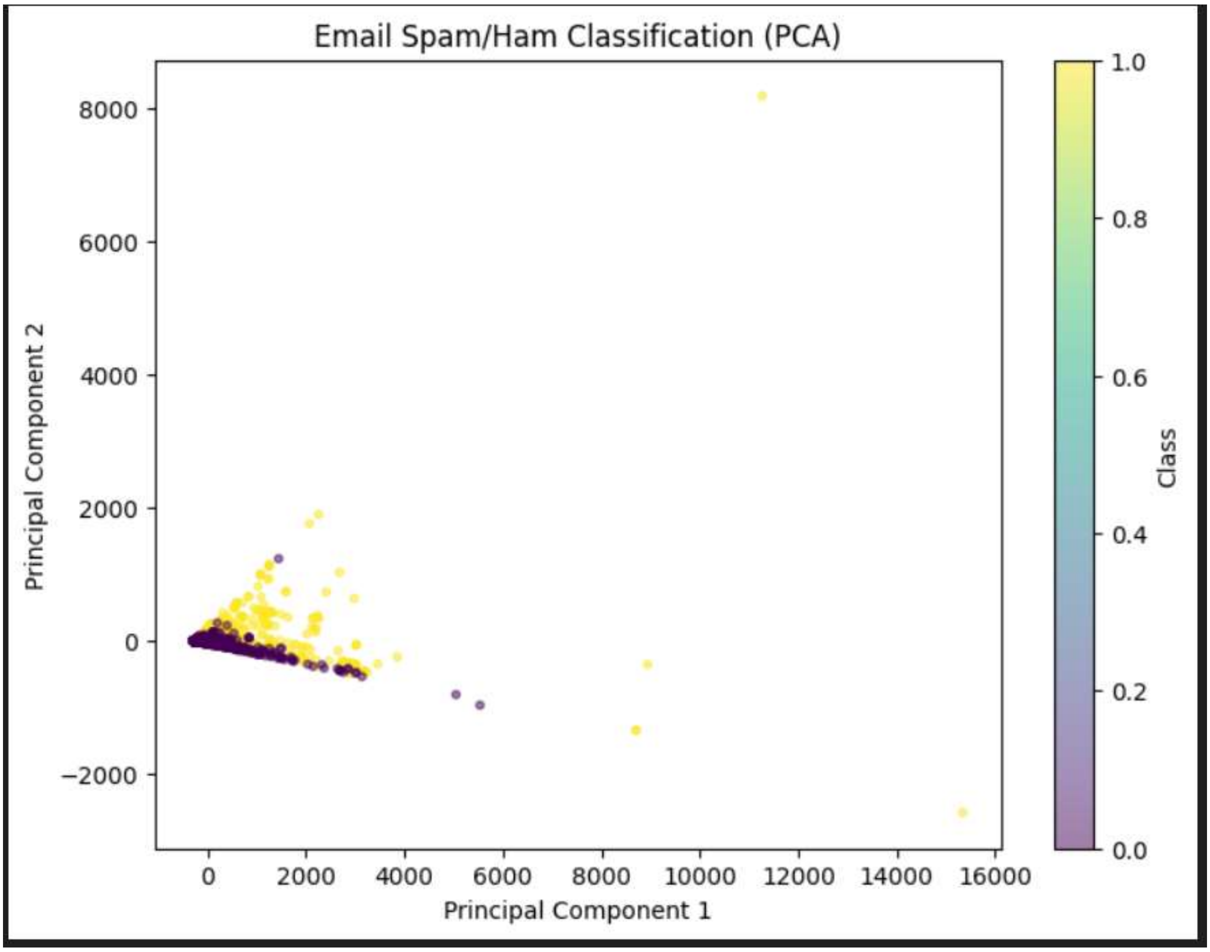
The sigmoid kernel is another kernel function used in SVMs.

It is based on the hyperbolic tangent function and is suitable for classification problems.

For spam mail classification, Linear kernel performs better.

The data is more linearly separable, meaning a linear decision boundary can effectively separate spam and non-spam emails.

Output:



```
Training Accuracy

Training Accuracy (Linear Kernel): 0.9339673913043478
Training Accuracy (poly Kernel): 0.6551630434782608
Training Accuracy (rbf Kernel): 0.7125
Training Accuracy (sigmoid Kernel): 0.6432065217391304


Testing Accuracy

Linear SVM Accuracy: 0.9381107491856677
Polynomial SVM Accuracy: 0.6449511400651465
RBF SVM Accuracy: 0.6905537459283387
Sigmoid SVM Accuracy: 0.6547231270358306
```

```
Other metrics:
Linear: 0.9258241758241759
F1 Score: 0.9220246238030096
AUC Score: 0.9347598343481639

________________________________________

Precision: 0.8448275862068966
F1 Score: 0.23058823529411765
AUC Score: 0.5586347495057005

________________________________________

Precision: 0.6782608695652174
F1 Score: 0.5226130653266332
AUC Score: 0.6457470563353958

________________________________________

Precision: 0.5649867374005305
F1 Score: 0.5725806451612904
AUC Score: 0.6421762952616099
```

INFERENCE:

In Support Vector Machine (SVM) models, the kernel function plays a crucial role in transforming the input data into a higher-dimensional space, where it might be easier to classify the data using a linear decision boundary.

1. Linear kernel: It computes the dot product between the input feature vectors, which effectively calculates the similarity between them.
2. Polynomial Kernel: The polynomial kernel function is used to handle nonlinear relationships between the features. It maps the data into a higher-dimensional space using polynomial functions.
3. Radial Basis Function (RBF) Kernel: The RBF kernel, also known as the Gaussian kernel, is widely used in SVMs due to its flexibility. It maps the data into an infinite-dimensional space using Gaussian radial basis functions. The RBF kernel considers all possible transformations of the input data into a higher-dimensional space.
4. Sigmoid Kernel: The sigmoid kernel is another kernel function used in SVMs. It is based on the hyperbolic tangent function and is suitable for classification problems.

For spam mail classification, Linear kernel performs better. The data is more linearly separable, meaning a linear decision boundary can effectively separate spam and non-spam emails.

MNIST

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split
from sklearn.svm import SVC


from sklearn import svm
from sklearn.svm import SVC
train_data = pd.read_csv("archive\MNIST\mnist_train.csv") #reading the csv files using pandas
test_data = pd.read_csv("archive\MNIST\mnist_test.csv")
df = train_data


df.describe()
df.shape
df.head()
df.isnull().sum()
df.columns
order = list(np.sort(df['label'].unique()))
print(order)
```

```
y = train_data['label']

X = train_data.drop(columns = 'label')

print(train_data.shape)
## Normalization

X = X/255.0
test_data = test_data/255.0

print("X:", X.shape)
print("test_data:", test_data.shape)
from sklearn.preprocessing import scale
X_scaled = scale(X)

# train test split
x_train, x_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.3, train_size = 0.2
,random_state = 10)
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

linearSVM = svm.SVC(kernel='linear')
polynomialSVM = svm.SVC(kernel='poly', degree=3)
rbfSVM = svm.SVC(kernel='rbf')
sigmoidSVM = svm.SVC(kernel='sigmoid')
linearSVM.fit(x_train,y_train)
linear_predictions = linearSVM.predict(x_test)

polynomialSVM.fit(x_train,y_train)
```

```
poly_predictions = polynomialSVM.predict(x_test)
rbfSVM.fit(x_train,y_train)
rbf_predictions = rbfSVM.predict(x_test)

sigmoidSVM.fit(x_train,y_train)
sigmoid_predictions = sigmoidSVM.predict(x_test)

linear_accuracy = accuracy_score(y_test, linear_predictions)
poly_accuracy = accuracy_score(y_test, poly_predictions)
rbf_accuracy = accuracy_score(y_test, rbf_predictions)
sigmoid_accuracy = accuracy_score(y_test, sigmoid_predictions)

print("Linear SVM Accuracy:", linear_accuracy)
print("Polynomial SVM Accuracy:", poly_accuracy)
print("RBF SVM Accuracy:", rbf_accuracy)
print("Sigmoid SVM Accuracy:", sigmoid_accuracy)
from sklearn import metrics

print("\nConfusion matrix for linear kernel\n" )
print(metrics.confusion_matrix(y_true=y_test, y_pred=linear_predictions))
print("\nConfusion matrix for poly kernel\n" )
print(metrics.confusion_matrix(y_true=y_test, y_pred=poly_predictions))
print("\nConfusion matrix for rbf kernel\n" )
print(metrics.confusion_matrix(y_true=y_test, y_pred=rbf_predictions))
print("\nConfusion matrix for sigmoid kernel\n" )
print(metrics.confusion_matrix(y_true=y_test, y_pred=sigmoid_predictions))
```

TRAINING ACCURACIES

```
svm_linear_model = svm.SVC(kernel='linear')
svm_linear_model.fit(x_train, y_train)

train_predictions_linear = svm_linear_model.predict(x_train)
```

```
train_accuracy_linear = accuracy_score(y_train, train_predictions_linear)
print("Training Accuracy (Linear Kernel):", train_accuracy_linear)
svm_poly_model = svm.SVC(kernel='poly')
svm_poly_model.fit(x_train, y_train)

train_predictions_poly = svm_poly_model.predict(x_train)
train_accuracy_poly = accuracy_score(y_train, train_predictions_poly)
print("Training Accuracy (poly Kernel):", train_accuracy_poly)
svm_rbf_model = svm.SVC(kernel='rbf')
svm_rbf_model.fit(x_train, y_train)

train_predictions_rbf = svm_rbf_model.predict(x_train)
train_accuracy_rbf = accuracy_score(y_train, train_predictions_rbf)
print("Training Accuracy (rbf Kernel):", train_accuracy_rbf)
svm_sigmoid_model = svm.SVC(kernel='sigmoid')
svm_sigmoid_model.fit(x_train, y_train)

train_predictions_sigmoid = svm_sigmoid_model.predict(x_train)
train_accuracy_sigmoid = accuracy_score(y_train, train_predictions_sigmoid)
print("Training Accuracy (sigmoid Kernel):", train_accuracy_sigmoid)
print("SVM model accuracies for different kernels\n")


print("Training accuracis:")
print("\n\t\tLinear kerenel: ",train_accuracy_linear)
print("\n\t\tpolynomial kerenel: ",train_accuracy_poly)
print("\n\t\trbf kerenel: ",train_accuracy_rbf)
print("\n\t\tSigmoid kerenel: ",train_accuracy_sigmoid)


print("\n\nTesting accuracis:")
```

```
print("\n\t\tLinear kerenel: ",linear_accuracy)
print("\n\t\tpolynomial kerenel: ",poly_accuracy)
print("\n\t\trbf kerenel: ",rbf_accuracy)
print("\n\t\tSigmoid kerenel: ",sigmoid_accuracy)
```

All kernels of SVM models produce a good accuracy.

OUTPUT:

```
SVM model accuracies for different kernels

Training accuracis:

                Linear kerenel:  1.0

                polynomial kerenel:  0.95025

                rbf kerenel:  0.98075

                Sigmoid kerenel:  0.9099166666666667


Testing accuracis:

                Linear kerenel:  0.9103333333333333

                polynomial kerenel:  0.9132222222222223

                rbf kerenel:  0.943

                Sigmoid kerenel:  0.9010555555555556
```

All kernels of SVM models produce a good accuracy.

```
Confusion matrix for linear kernel

[[1719    0   10    5    3   16   12    1    6    0]
 [   1 1951   11    5    5    4    0    3   11    1]
 [  11   26 1676   30   23    5   23   19   17    1]
 [  10    4   47 1627    4   66    5   19   42   10]
 [   4    8   21    1 1658    5   14    6    5   50]
 [  21    9   19   87   12 1423   30    1   39   11]
 [  20    7   23    1   14   20 1666    2    4    0]
 [   7   13   19   16   31    4    2 1774    5   93]
 [  25   44   49   54   12   58   18   11 1436   17]
 [   4   11   19   23   90    9    1   69   20 1456]]

Confusion matrix for poly kernel

[[1649    0    7    2    8    9   11    1   84    1]
 [   0 1941    8    5    6    0    2    1   28    1]
 [   4    8 1576   15   53    2    5    7  159    2]
 [   1    2   15 1644    6   24    0   11  113   18]
 [   0    5   14    0 1685    3    4    0    5   56]
 [   2    1    1   33   27 1388   18    2  149   31]
 [   3    4    3    0   26   14 1657    0   50    0]
 [   1   14    5    1   76    1    0 1692   28  146]
 [   3    6   14   12   11   12    4    1 1650   11]
 [   2    6    4   14   62    4    0   16   38 1556]]
```

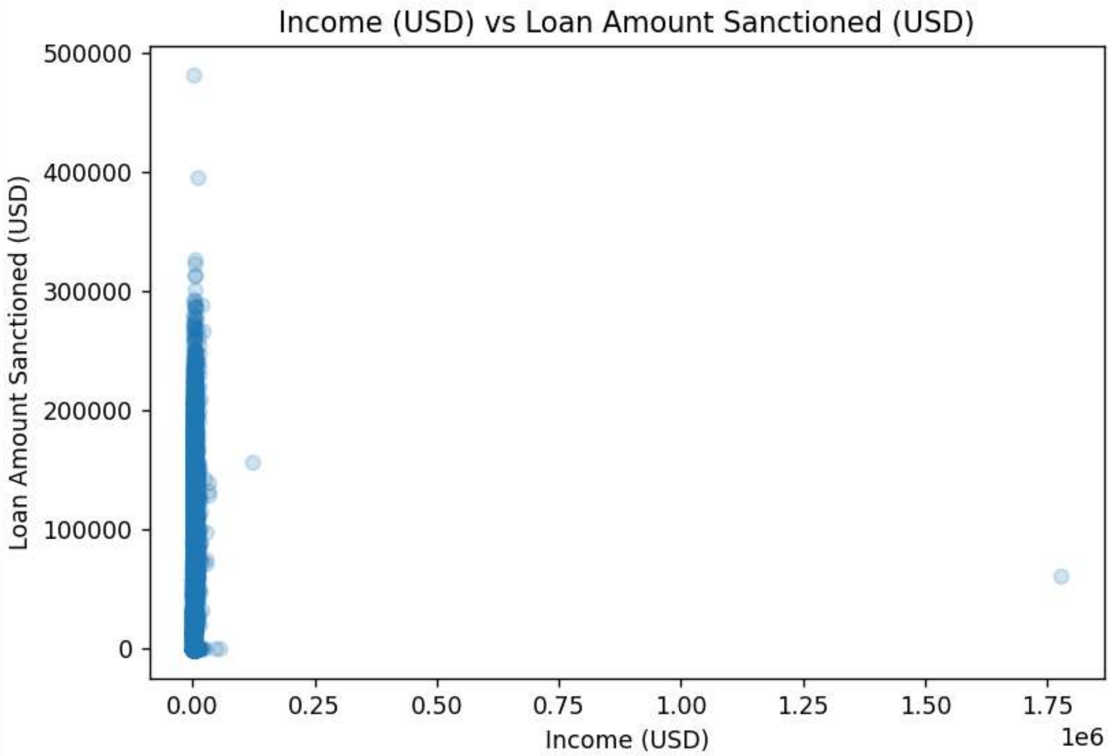
```
Confusion matrix for rbf kernel

[[1722    0   15    4    1    6   13    2    8    1]
 [   1 1947   21    7    5    0    1    2    7    1]
 [   5    6 1747   11   12    3   14   16   14    3]
 [   2    3   52 1685    1   37    2   21   26    5]
 [   1    5   31    1 1664    5    9    7    4   45]
 [   3    5   28   33    3 1526   27    5   15    7]
 [   8    4   21    0    4   14 1698    1    7    0]
 [   3   11   52    7   14    0    0 1828    2   47]
 [  10   22   32   18    9   26   11    5 1585    6]
 [   3    5   25   17   23    4    0   38   15 1572]]

Confusion matrix for sigmoid kernel

[[1695    0   19    4    4   26    9    1   12    2]
 [   1 1940   15    8    3    8    0    2   11    4]
 [  23   19 1593   26   26    6   84   12   34    8]
 [   8    4   56 1603    5   86    6   26   30   10]
 [   4    7   33    2 1623    6   13   10    3   71]
 [  16   21   22   62   15 1424   29    4   36   23]
 [  19   11   41    0   20   26 1633    0    7    0]
 [   7   20   25   25   22    2    4 1749    5  105]
 [  20   49   43   38    8   57   14    8 1473   14]
 [   7    8   23   23   71    9    0   64   11 1486]]
```

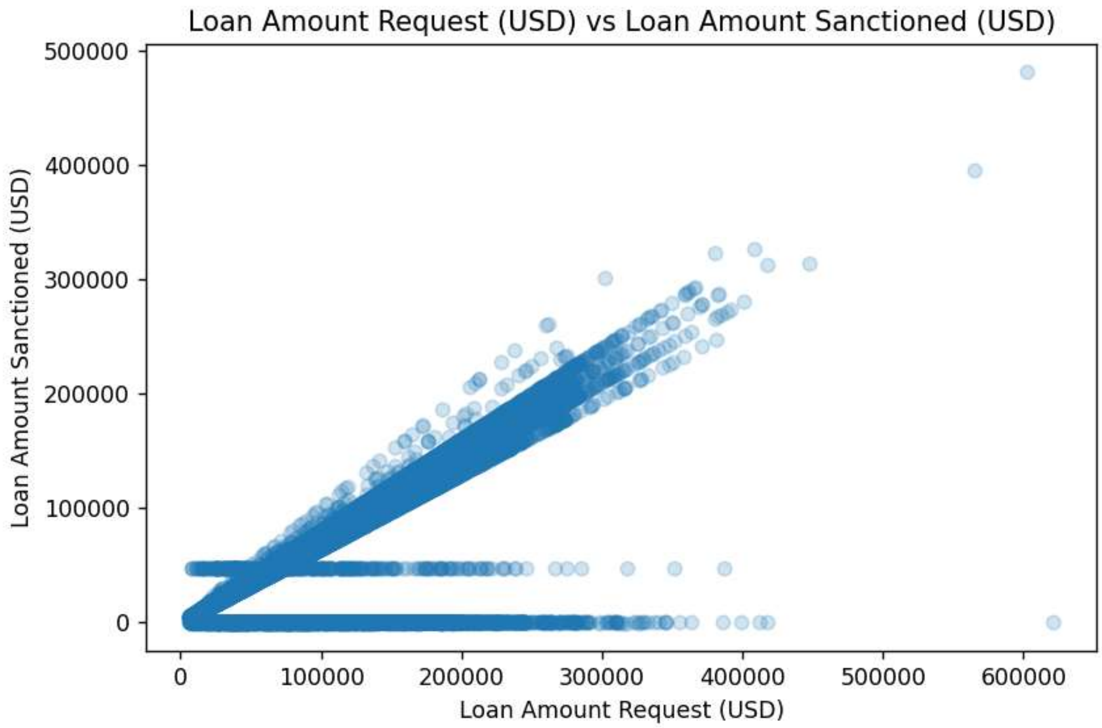
Learning outcome:

1. Learnt to implement SVM model.

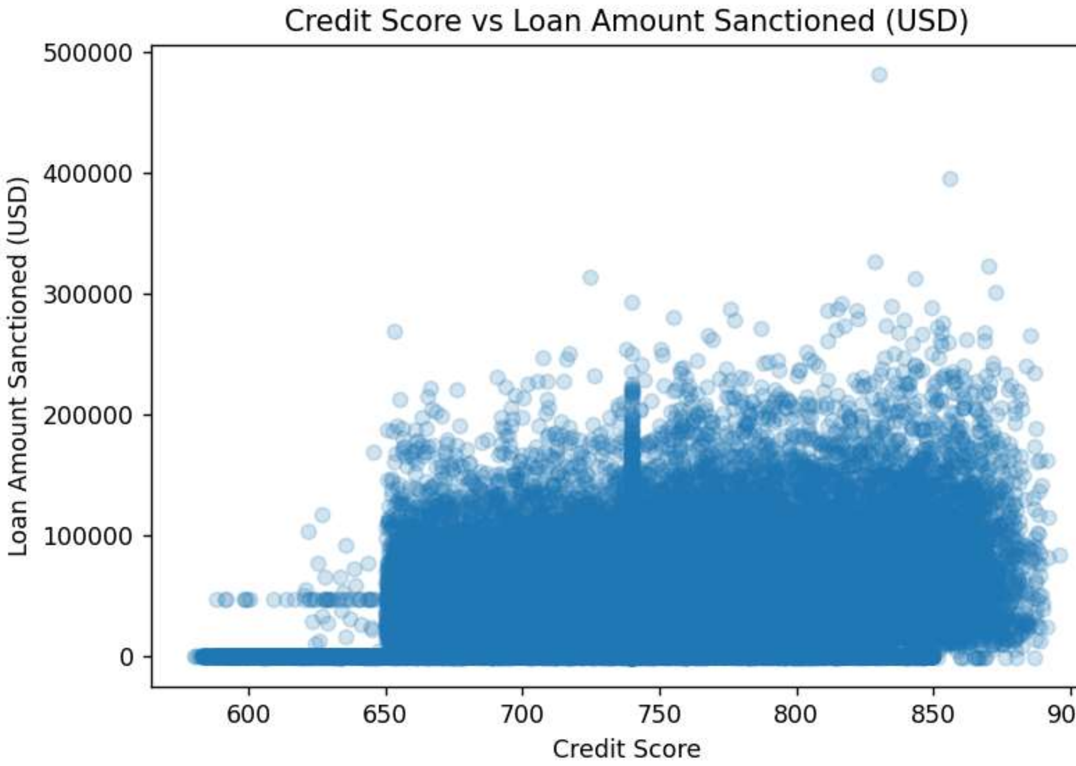
2. Different kernels in svm to build a model.

Harini Mohan 3122215001029

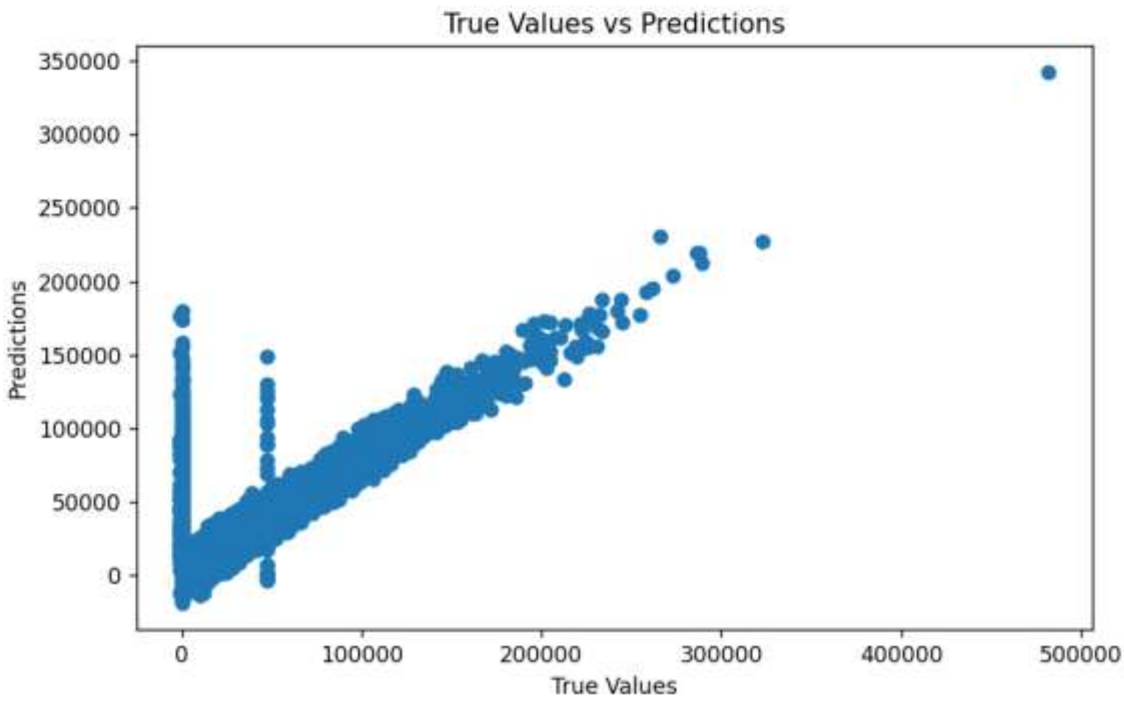




Harini Mohan 3122215001029



```
Mean Squared Error: 945297926.63202
Root Mean Squared Error (RMSE): 30745.697693043494
Mean Absolute Error (MAE): 21560.77968131055
R-squared (Coefficient of Determination): 0.5824179712653825

--------------------------------------------------
```



Learning outcome:

1. Learnt to implement linear regression model.

2. Learnt to handle missing values in a dataset.

3. Learnt about the evaluation metrics used for regression models.

4. Learnt to visualize the results and exploratory data analysis methods.