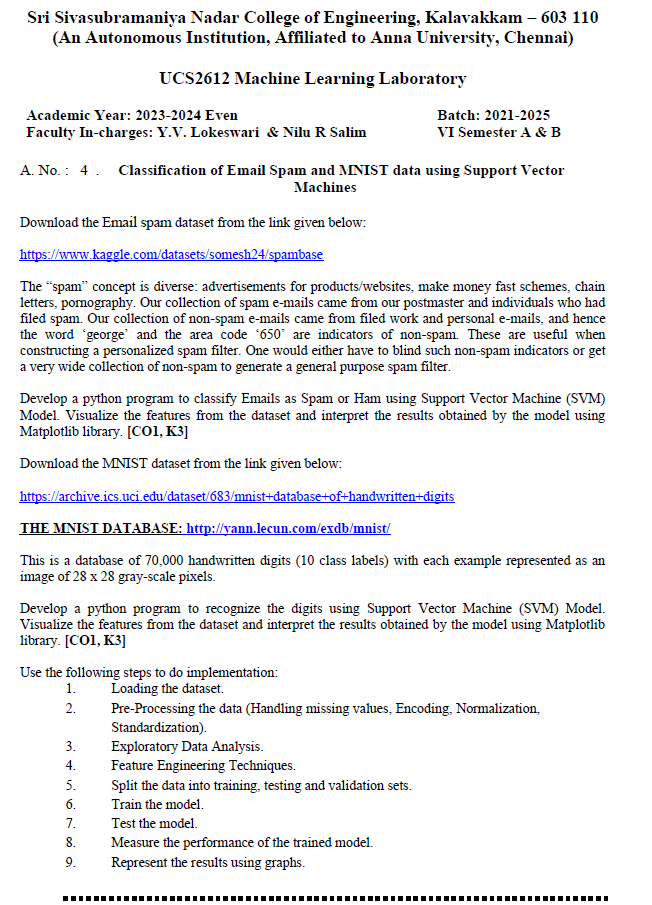
**UCS2612 Machine Learning Laboratory**

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



**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:

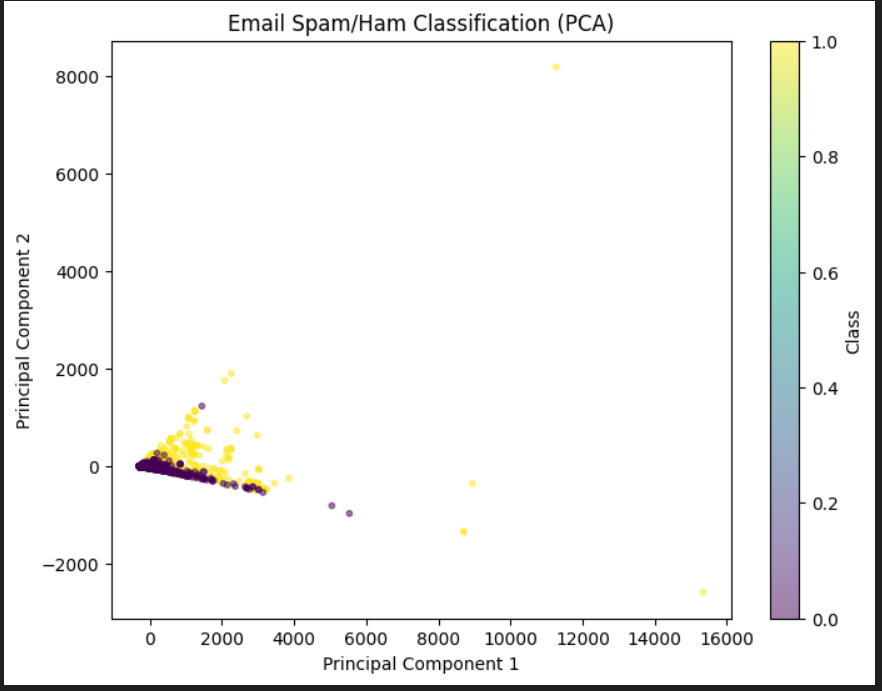
The sigmoid kernel is another kernel function used in SVMs.

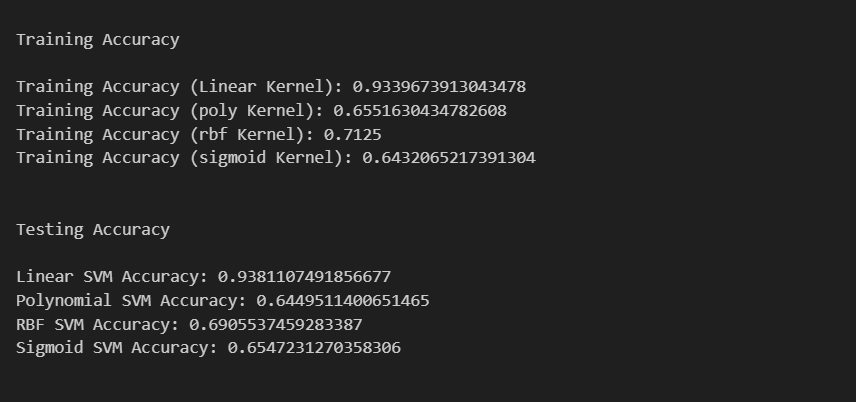
It is based on the hyperbolic tangent function and is suitable for classification problems.

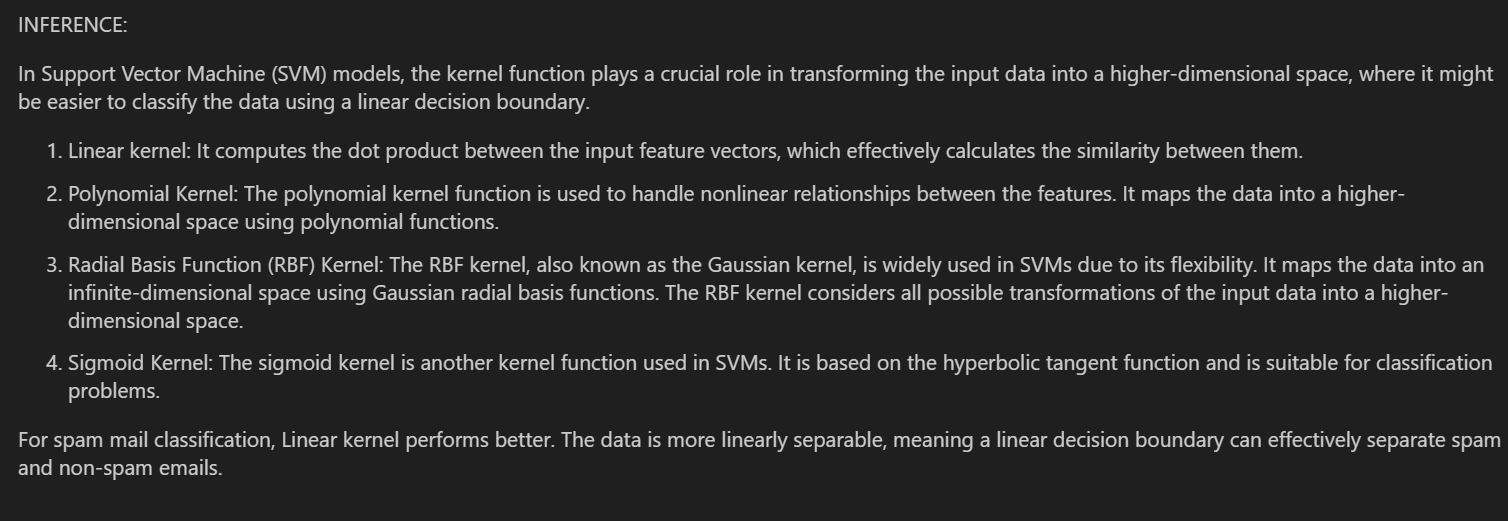
For spam mail classification, Linear kernel performs better.

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

**Output:**





**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)

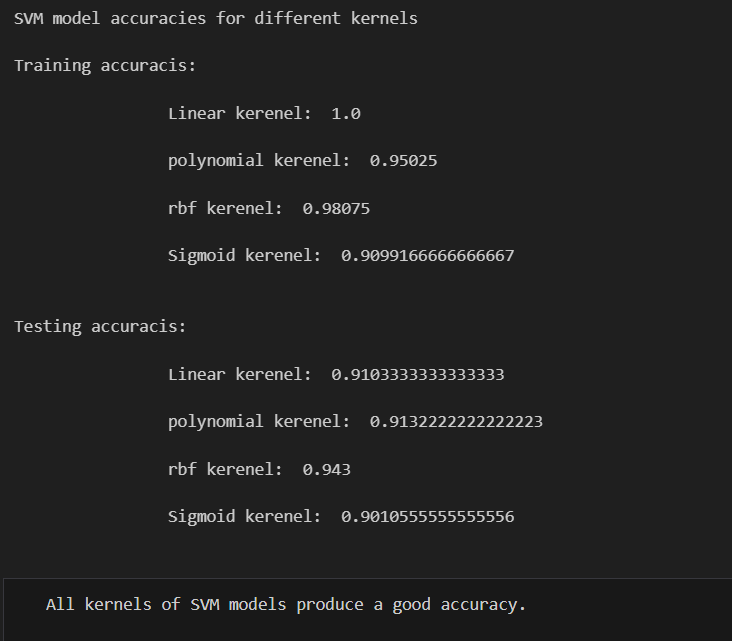
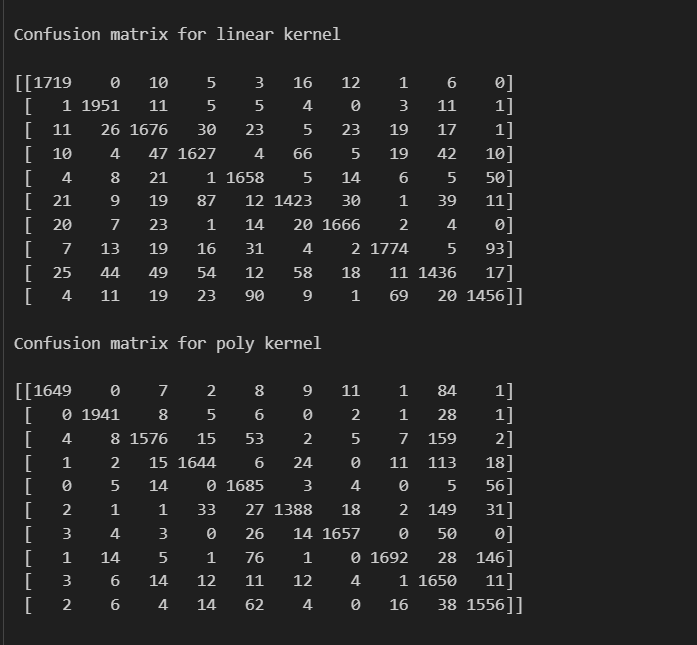
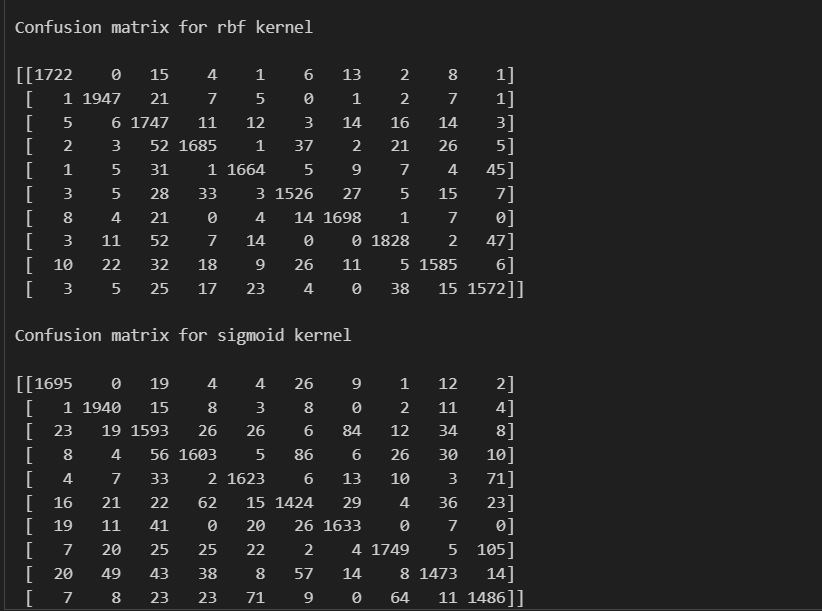
print("\n\t\tpolynomial kerenel: ",poly\_accuracy)

print("\n\t\trbf kerenel: ",rbf\_accuracy)

print("\n\t\tSigmoid kerenel: ",sigmoid\_accuracy)

All kernels of SVM models produce a good accuracy.

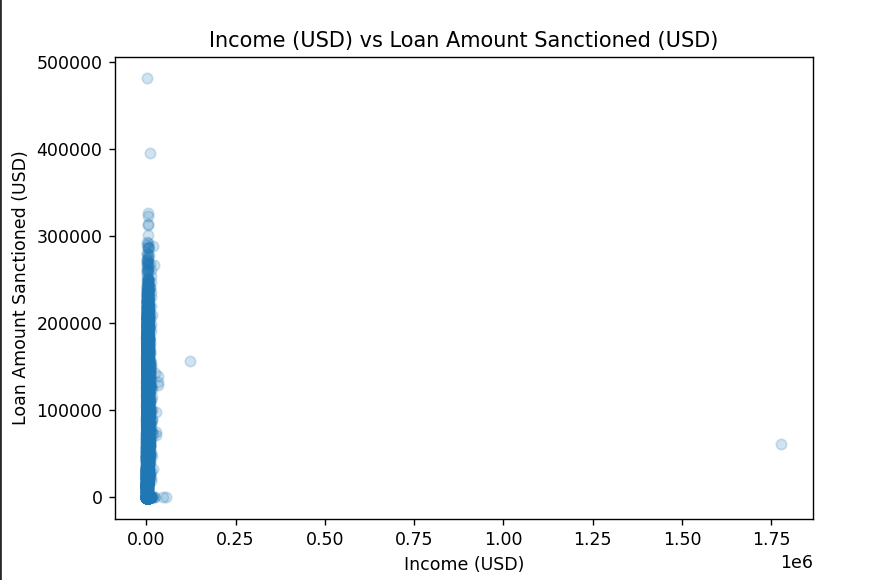
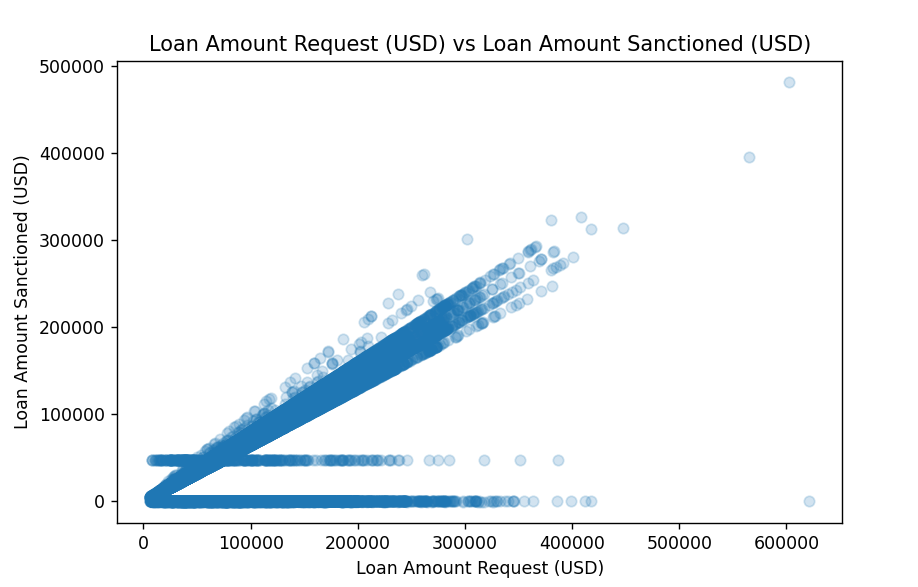
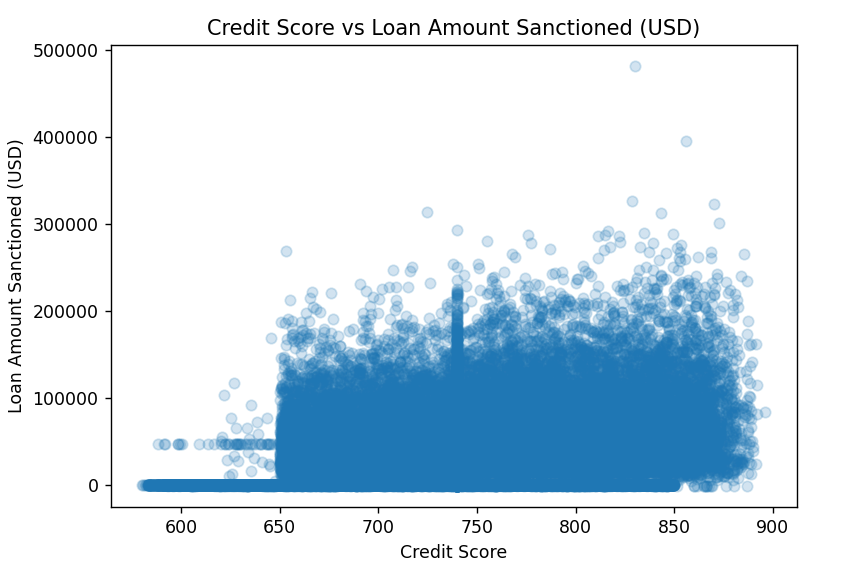
**OUTPUT:**

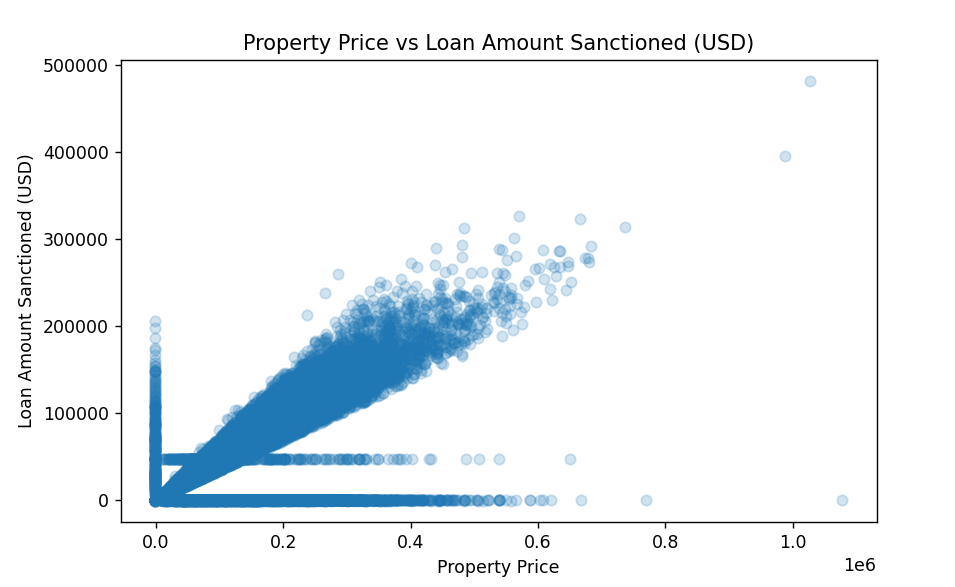
**** **** 

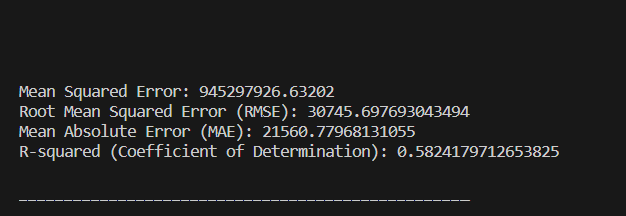
**Learning outcome:**

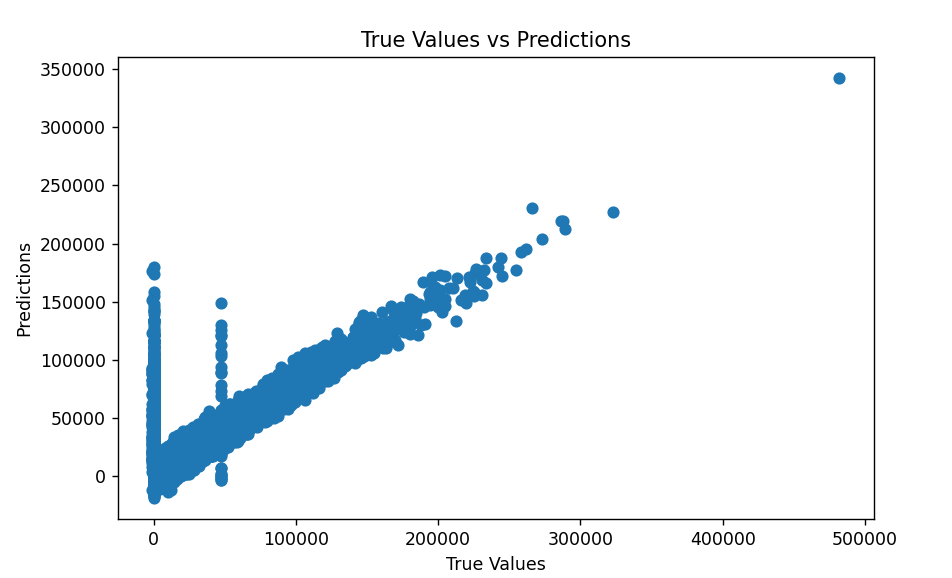
1. Learnt to implement SVM model.

2. Different kernels in svm to build a model.







**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.