Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2022-23

Experiment 8

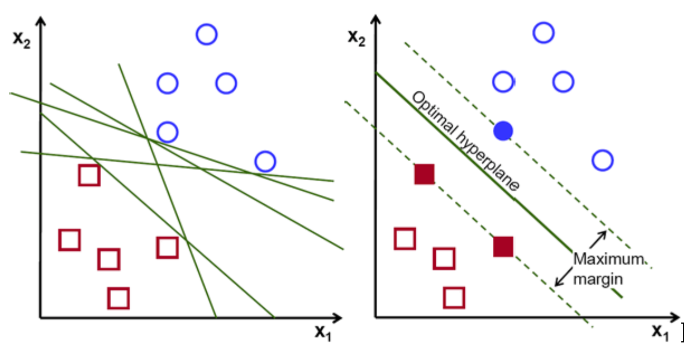
Bhuvi Ghosh
60009210191

(SVM)

Aim: Perform SVM using soft margin SVC, Kernels and improve the accuracies using hyperparameter tuning.

Theory:

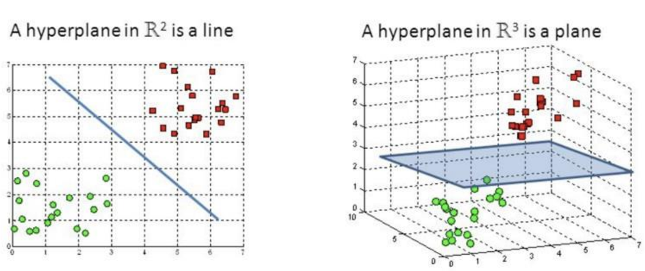
The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.



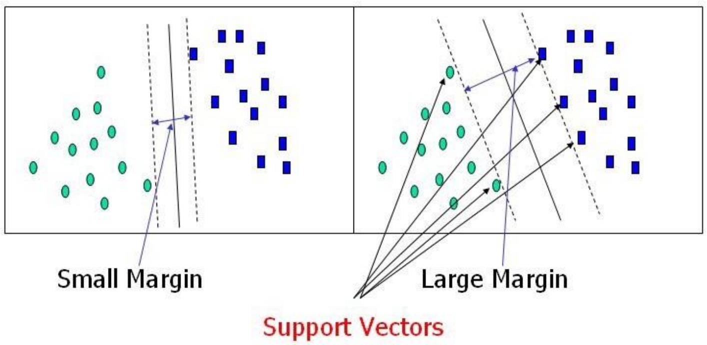
Possible hyperplanes

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Hyperplanes and Support Vectors



Hyperplanes in 2D and 3D feature space: Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

Large Margin Intuition

In logistic regression, we take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function. If the squashed value is greater than a threshold value(0.5) we assign it a label 1, else we assign it a label 0. In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin.

Cost Function and Gradient Update: In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

$$c(x, y, f(x)) = (1 - y * f(x))_+$$

Hinge loss function (function on left can be represented as a function on the right)

The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions looks as below.

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n}(1 - y_i \langle x_i, w \rangle)_+$$

Loss function for SVM

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

$$\frac{\delta}{\delta w_k}\lambda \parallel w \parallel^2 \quad = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k}\left(1 - y_i \langle x_i, w \rangle\right)_+ \quad = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Gradients

When there is no misclassification, i.e our model correctly predicts the class of our data point, we only have to update the gradient from the regularization parameter.

$$w = w - \alpha \cdot (2\lambda w)$$

Gradient Update — No misclassification

When there is a misclassification, i.e our model make a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform gradient update.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

Gradient Update — Misclassification

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:

Dataset 1: IRIS.cvs

Dataset 2: mnist_784 : The MNIST database of handwritten digits with 784 features, raw data available at: http://yann.lecun.com/exdb/mnist/. It can be split in a training set of the first 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting. The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

Task 1: Build a linear classifier on Dataset 1 using SVC.
Task 2: Build a classifier on Dataset 1 using Linear, Polynomial and RBF kernel and show the decision boundary using matplotlib.
Task 3. Find the accuracy of svc classifier (M1) built on Dataset 3 using linear csv and RBF kernel.
Task 4: Improve the accuracy of M1 by varying C and gamma values and using RandomizedSearchCV.
Task 5: Calculate the computational time of Task 3 and 4.

Linear SVM

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
from sklearn.datasets import load_iris
df = load_iris()
```

In [3]:
```
X=df.data
y=df.target
```

In [4]:
```
y
```

Out[4]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [5]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
```

In [6]:
```
from sklearn.svm import SVC
model=SVC(kernel='linear')
model.fit(X_train,y_train)
```

Out[6]:
```
▾ SVC
SVC(kernel='linear')
```

In [7]:
```
y_pred=model.predict(X_test)
```

In [8]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

In [9]:
```
!pip install mlxtend
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
ic/simple/
Requirement already satisfied: mlxtend in /usr/local/lib/python3.10/dist-packages (0.14.
0)
Requirement already satisfied: scipy>=0.17 in /usr/local/lib/python3.10/dist-packages (f
rom mlxtend) (1.10.1)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.10/dist-pack
ages (from mlxtend) (1.2.2)
Requirement already satisfied: pandas>=0.17.1 in /usr/local/lib/python3.10/dist-packages
```

```
(from mlxtend) (1.5.3)
Requirement already satisfied: matplotlib>=1.5.1 in /usr/local/lib/python3.10/dist-packa
ges (from mlxtend) (3.7.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (fr
om mlxtend) (67.7.2)
Requirement already satisfied: numpy>=1.10.4 in /usr/local/lib/python3.10/dist-packages
(from mlxtend) (1.22.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages
(from matplotlib>=1.5.1->mlxtend) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib>=1.5.1->mlxtend) (1.0.7)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-pa
ckages (from matplotlib>=1.5.1->mlxtend) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packa
ges (from matplotlib>=1.5.1->mlxtend) (1.4.4)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packa
ges (from matplotlib>=1.5.1->mlxtend) (4.39.3)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages
(from matplotlib>=1.5.1->mlxtend) (8.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib>=1.5.1->mlxtend) (3.0.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-package
s (from matplotlib>=1.5.1->mlxtend) (23.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages
(from pandas>=0.17.1->mlxtend) (2022.7.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
(from scikit-learn>=0.18->mlxtend) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from scikit-learn>=0.18->mlxtend) (3.1.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
python-dateutil>=2.7->matplotlib>=1.5.1->mlxtend) (1.16.0)
```

In [10]:

```
!pip install mlxtend --upgrade --no-deps
```

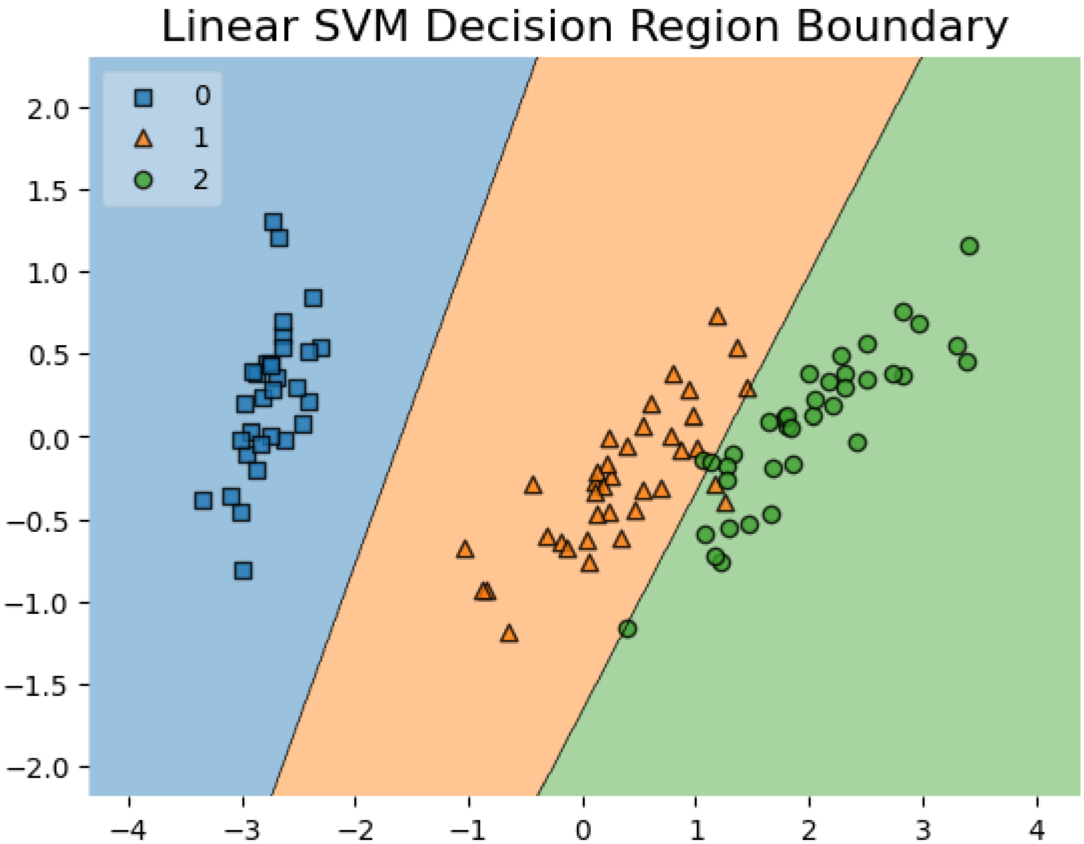
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
ic/simple/
Requirement already satisfied: mlxtend in /usr/local/lib/python3.10/dist-packages (0.14.
0)
Collecting mlxtend
  Downloading mlxtend-0.22.0-py2.py3-none-any.whl (1.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 14.9
 MB/s eta 0:00:00
Installing collected packages: mlxtend
  Attempting uninstall: mlxtend
    Found existing installation: mlxtend 0.14.0
    Uninstalling mlxtend-0.14.0:
      Successfully uninstalled mlxtend-0.14.0
Successfully installed mlxtend-0.22.0
```

In [34]:

```
from sklearn.decomposition import PCA
from mlxtend.plotting import plot_decision_regions
import matplotlib.pyplot as plt

clf = SVC(kernel='linear')
pca = PCA(n_components = 2)
X_train2 = pca.fit_transform(X_train)
clf.fit(X_train2, y_train)
plot_decision_regions(X_train2, y_train, clf=clf, legend=2)

plt.title('Linear SVM Decision Region Boundary', size=16)
```

Out[34]:

```
Text(0.5, 1.0, 'Linear SVM Decision Region Boundary')
```



Non-Linear SVM

In [14]:
```
import pandas as pd
import numpy as np
```

In [15]:
```
from sklearn.datasets import load_iris
df = load_iris()
```

In [16]:
```
X=df.data
y=df.target
```

In [17]:
```
y
```

Out[17]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [18]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
```

In [19]:
```
from sklearn.svm import SVC
model=SVC(kernel='rbf')
model.fit(X_train,y_train)
```

Out[19]:

▾ SVC

SVC()

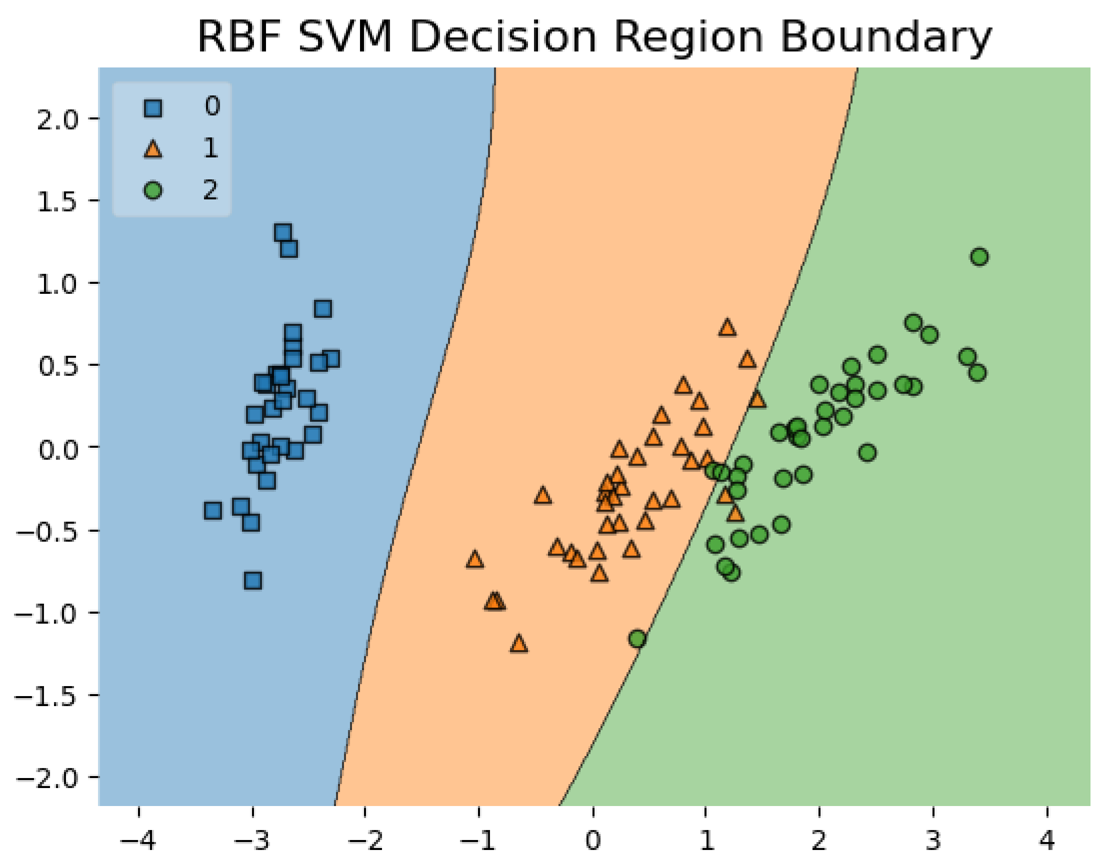
In [20]:
```
y_pred=model.predict(X_test)
```

In [21]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

In [33]:
```
from sklearn.decomposition import PCA
from mlxtend.plotting import plot_decision_regions
import matplotlib.pyplot as plt

clf  = SVC(kernel='rbf')
pca = PCA(n_components = 2)
X_train2  =  pca.fit_transform(X_train)
clf.fit(X_train2, y_train)
plot_decision_regions(X_train2,  y_train,  clf=clf,  legend=2)

plt.title('RBF SVM Decision Region Boundary', size=16)
```

Out[33]:
```
Text(0.5, 1.0, 'RBF SVM Decision Region Boundary')
```



Polynomial SVM

In [23]:
```
import pandas as pd
import numpy as np
```

In [24]:
```
from sklearn.datasets import load_iris
df = load_iris()
```

In [25]:
```
X=df.data
y=df.target
```

In [26]:
```
y
```

Out[26]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [27]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
```

In [28]:
```
from sklearn.svm import SVC
model=SVC(kernel='poly',degree=3)
model.fit(X_train,y_train)
```

Out[28]:
```
▾ SVC
SVC(kernel='poly')
```

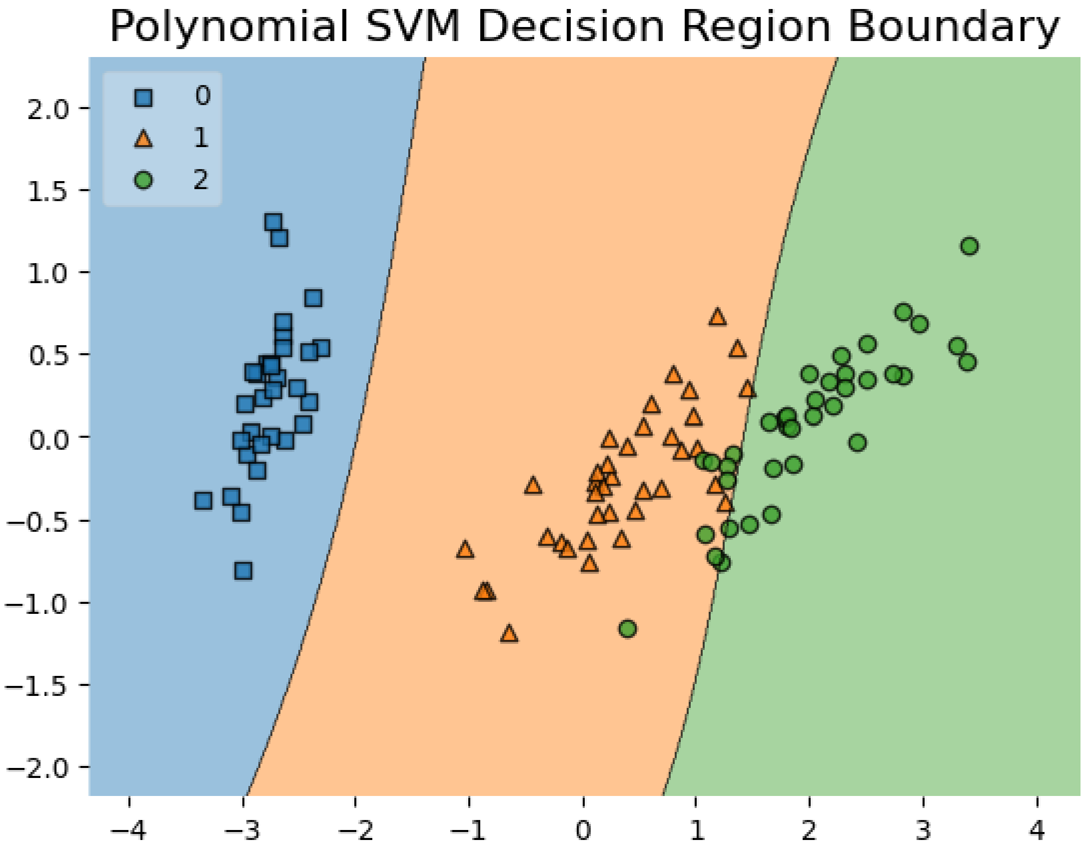
In [29]:
```
y_pred=model.predict(X_test)
```

In [30]:
```
from sklearn.metrics import classification_report,accuracy_score
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.92      0.96        13
           2       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```

In [32]:
```
from sklearn.decomposition import PCA
from mlxtend.plotting import plot_decision_regions
import matplotlib.pyplot as plt

clf = SVC(kernel='poly')
pca = PCA(n_components = 2)
X_train2 = pca.fit_transform(X_train)
clf.fit(X_train2, y_train)
plot_decision_regions(X_train2, y_train, clf=clf, legend=2)

plt.title('Polynomial SVM Decision Region Boundary', size=16)
```

Out[32]:
```
Text(0.5, 1.0, 'Polynomial SVM Decision Region Boundary')
```

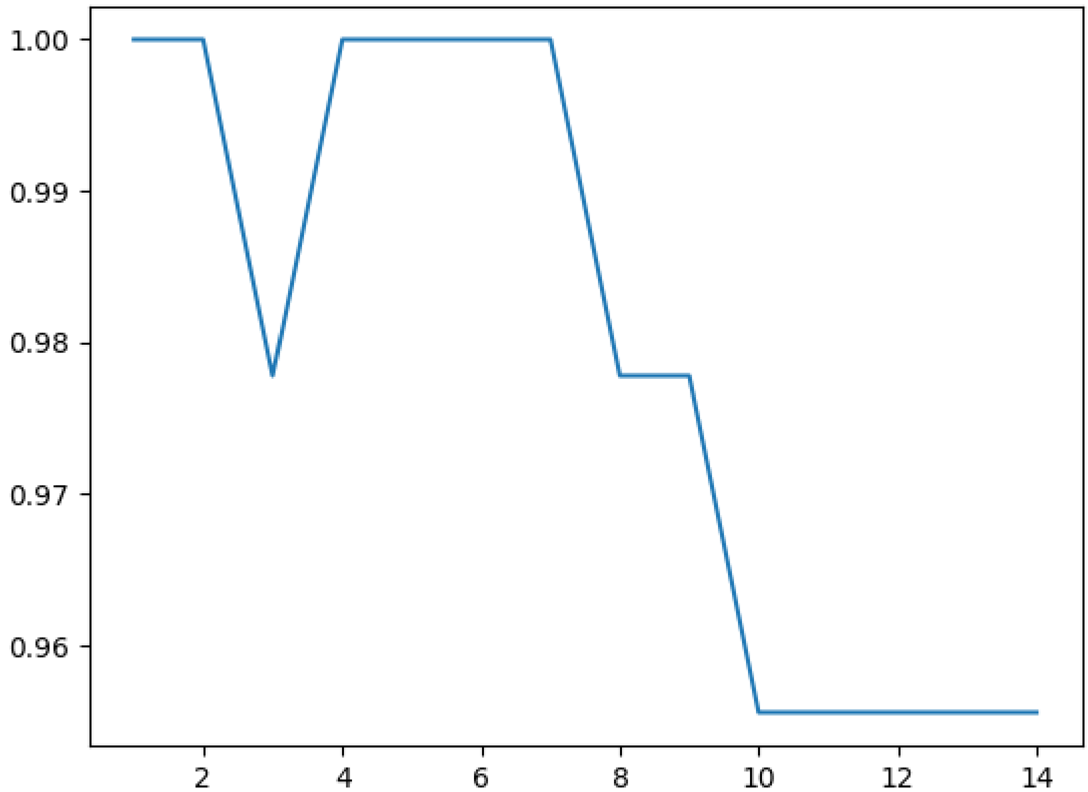


Change with degree in Polynomial

In [35]:
```
deg=[]
acc=[]
for i in range(1,15):
  model=SVC(kernel='poly',degree=i)
  model.fit(X_train,y_train)
  y_pred=model.predict(X_test)
  deg.append(i)
  acc.append(accuracy_score(y_test,y_pred))
```

In [36]:
```
import seaborn as sns
sns.lineplot(x=deg,y=acc)
```

Out[36]:
```
<Axes: >
```

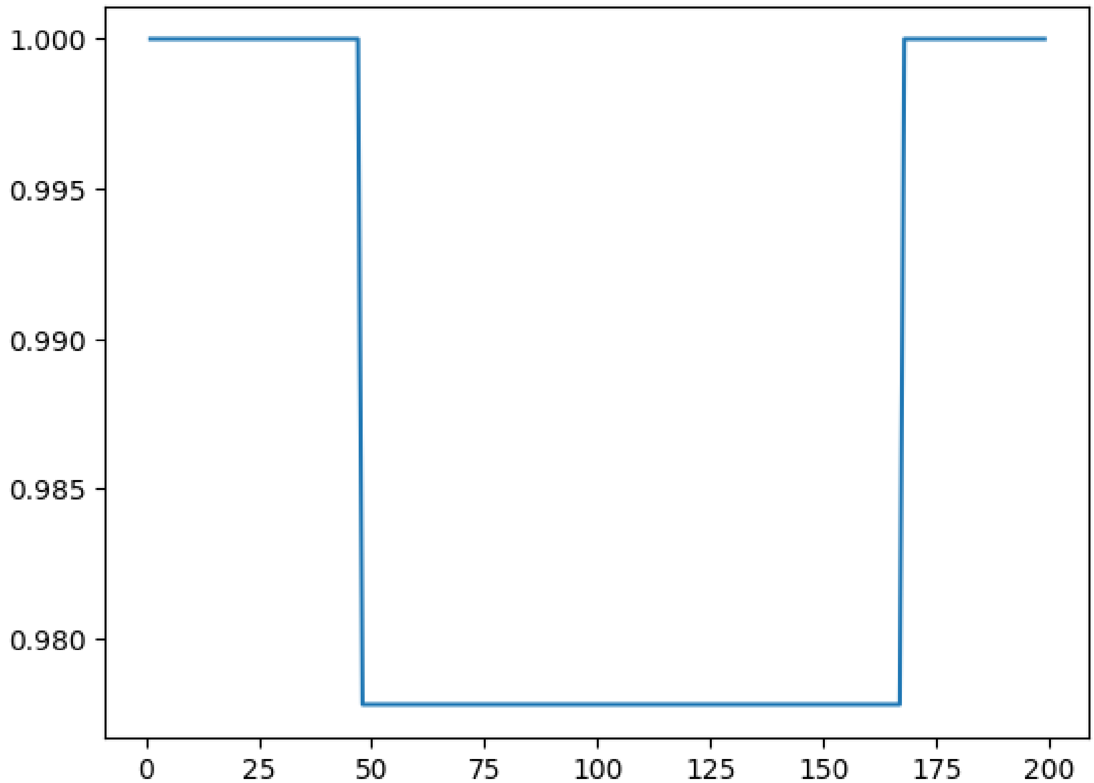


Change with c value in RBF

In [37]:
```
deg=[]
acc=[]
for i in range(1,200):
  model=SVC(kernel='rbf',C=i)
  model.fit(X_train,y_train)
  y_pred=model.predict(X_test)
  deg.append(i)
  acc.append(accuracy_score(y_test,y_pred))
```

In [38]:
```
import seaborn as sns
sns.lineplot(x=deg,y=acc)
```

Out[38]:
```
<Axes: >
```


1.000
0.995
0.990
0.985
0.980
0
25
50
75
100
125
150
175
200

In [1]:
```python
import pandas as pd
```

In [2]:
```python
!pip install python-mnist
```

```
Collecting python-mnist
  Downloading python_mnist-0.7-py2.py3-none-any.whl (9.6 kB)
Installing collected packages: python-mnist
Successfully installed python-mnist-0.7
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting
 behaviour with the system package manager. It is recommended to use a virtual environme
nt instead: https://pip.pypa.io/warnings/venv
```

In [3]:
```python
!pip uninstall --yes mnist
```

```
Found existing installation: mnist 0.2.2
Uninstalling mnist-0.2.2:
  Successfully uninstalled mnist-0.2.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting
 behaviour with the system package manager. It is recommended to use a virtual environme
nt instead: https://pip.pypa.io/warnings/venv
```

SVM (Kernel: RBF)

In [4]:
```python
from mnist.loader import MNIST
mndata = MNIST('/kaggle/input/djsce-data-science-svm-dataset')
images, labels = mndata.load_training()
```

In [5]:
```python
import random
print(mndata.display(images[2]))
```

```
............................
............................
............................
............................
............................
.....................@......
............................
.....................@......
....@...............@.......
....@...............@.......
....@...............@.......
....@..............@@.......
....@..............@@.......
....@.............@@........
....@.........@@@.@@........
....@@@@@@@@@.....@@........
..................@@........
..................@.........
..................@.........
..................@.........
..................@.........
..................@.........
..................@.........
..................@.........
..................@.........
............................
............................
............................
```

In [6]:
```python
X_train=pd.DataFrame(images)
```

In [7]:
```
y_tr=list(labels)
```

In [8]:
```
y_train=pd.DataFrame(y_tr)
```

In [9]:
```
from sklearn.svm import SVC
clf = SVC(kernel='rbf')
clf.fit(X_train, y_train)
```

Out[9]:
```
SVC()
```

In [10]:
```
images, labels = mndata.load_testing()
X_test=pd.DataFrame(images)
y_ts=list(labels)
y_test=pd.DataFrame(y_ts)
```

In [11]:
```
y_pred=clf.predict(X_test)
```

In [12]:
```
from sklearn.metrics import accuracy_score,classification_report
print('Testing Accuracy:',accuracy_score(y_pred,y_test))
print(classification_report(y_test,y_pred))
```

```
Testing Accuracy: 0.9792
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.97      0.98      1032
           3       0.97      0.99      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.99      0.98      0.98       892
           6       0.99      0.99      0.99       958
           7       0.98      0.97      0.97      1028
           8       0.97      0.98      0.97       974
           9       0.97      0.96      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

In [13]:
```
y_pred2=clf.predict(X_train)
```

In [14]:
```
from sklearn.metrics import accuracy_score,classification_report
print('Training Accuracy',accuracy_score(y_pred2,y_train))
print(classification_report(y_train,y_pred2))
```

```
Training Accuracy 0.9899166666666667
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      5923
           1       0.99      0.99      0.99      6742
           2       0.99      0.99      0.99      5958
           3       0.99      0.98      0.99      6131
           4       0.99      0.99      0.99      5842
           5       0.99      0.99      0.99      5421
           6       0.99      1.00      1.00      5918
           7       0.99      0.99      0.99      6265
           8       0.99      0.99      0.99      5851
           9       0.98      0.98      0.98      5949
```

```
    accuracy                           0.99     60000
   macro avg       0.99      0.99      0.99     60000
weighted avg       0.99      0.99      0.99     60000
```

SVM (Kernel: Linear)

In [ ]:
```
from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
```

In [ ]:
```
y_pred=clf.predict(X_test)
```

In [ ]:
```
from  sklearn.metrics  import  accuracy_score,classification_report
print('Testing Accuracy:',accuracy_score(y_pred,y_test))
print(classification_report(y_test,y_pred))
```

In [ ]:
```
y_pred2=clf.predict(X_train)
```

In [ ]:
```
from  sklearn.metrics  import  accuracy_score,classification_report
print('Training Accuracy',accuracy_score(y_pred2,y_train))
print(classification_report(y_train,y_pred2))
```

SVM (Randomized Search CV)

In [ ]:
```
import pandas as pd
import numpy as np
from  sklearn.metrics  import  classification_report,  confusion_matrix
from  sklearn.datasets  import  load_breast_cancer
from sklearn.svm import SVC

from  sklearn.model_selection  import  GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001], 'ker

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train,  y_train)
```

In [ ]:
```
from  sklearn.metrics  import  accuracy_score,classification_report
grid_predictions = grid.predict(X_test)

# print classification report
print('Testing  Accuracy:',accuracy_score(grid_predictions,y_test))
print(classification_report(y_test, grid_predictions))
```

In [ ]:
```
from  sklearn.metrics  import  accuracy_score,classification_report
grid_predictions2 = grid.predict(X_train)

# print classification report
```

```
print('Training Accuracy:',accuracy_score(grid_predictions2,y_train))
print(classification_report(y_train, grid_predictions2))
```