

Experiment:- 4

Student Name: Anshuman Singh

Branch: CSE

Semester: 5th

Subject Name: MACHINE LEARNING LAB

UID: 20BCS2665

Section/Group: 20BCS_WM_902/A

Subject Code: 20CSP-317

Aim/Overview of the practical: Implement SVM on any data set and analyze the accuracy with Logistic regression.

Task to be done:

Implement SVM on any data set.

Apparatus/Simulator used:

- Jupyter Notebook/Google Collab
- Python
- pandas Library
- seaborn Library
- Standard Dataset

Code and Output:

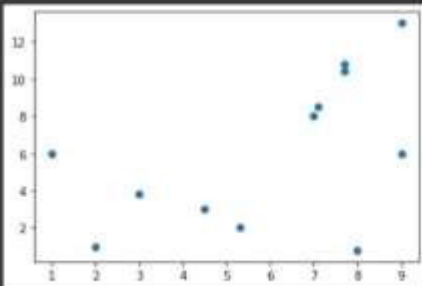


+ Code + Text

```
[1] import matplotlib.pyplot as plt
import numpy as np
from sklearn import svm
```

```
[15] # linear data
X = np.array([2, 9, 3, 8, 1, 9, 7, 7.7, 5.3, 4.5, 7.7, 7.1])
y = np.array([1, 6, 3.8, 9.8, 6, 13, 8, 10.4, 2, 3, 10.8, 8.3])
```

```
[16] # show unclassified data
plt.scatter(X, y)
plt.show()
```



```
[17] # shaping data for training the model
training_X = np.vstack((X, y)).T
training_y = [0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1]
```

```
# define the model
clf = svm.SVC(kernel='linear', C=1.0)
```

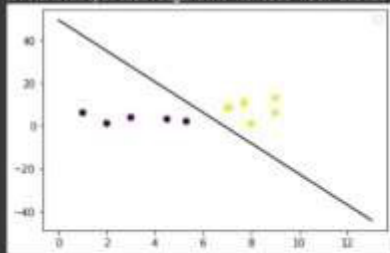
```
[19] # train the model
clf.fit(training_X, training_y)

SVC(kernel='linear')
```

```
SVC(kernel='linear')
```

```
# get the weight values for the linear equation from the trained SVM model  
w = clf.coef_[0]  
  
# get the y-offset for the linear equation  
s = -w[0] / w[1]  
  
# make the x-axis space for the data points  
XX = np.linspace(0, 13)  
  
# get the y-values to plot the decision boundary  
yy = s * XX - clf.intercept_[0] / w[1]  
  
# plot the decision boundary  
plt.plot(XX, yy, 'k-')  
  
# show the plot visually  
plt.scatter(training_X[:, 0], training_X[:, 1], c=training_y)  
plt.legend()  
plt.show()
```

WARNING:matplotlib.legend.No handles with labels found to put in legend.



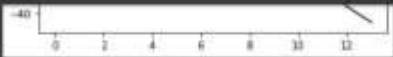
0s completed at 3:43 PM



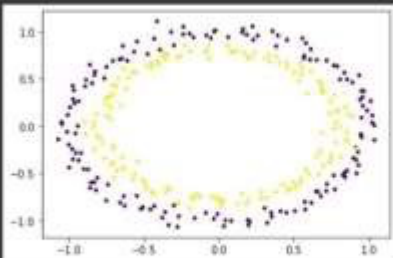
exp 4 ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text



```
[21] import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets  
from sklearn import svm  
  
[23] # non-linear data  
circle_X, circle_y = datasets.make_circles(n_samples=300, noise=0.05)  
  
[24] # show raw non-linear data  
plt.scatter(circle_X[:, 0], circle_X[:, 1], c=circle_y, marker='o')  
plt.show()
```



```
[25] # make non-linear algorithm for model  
nonlinear_clf = svm.SVC(kernel='rbf', C=1.0)
```

```
exp 4 ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

[25] # make non-linear algorithms for model
nonlinear_clf = svm.SVC(kernel='rbf', C=1, 0)

[26] # training non-linear model
nonlinear_clf.fit(circle_X, circle_y)

SVC()

[27] # Plot the decision boundary for a non-linear SVM problem
def plot_decision_boundary(model, ax=None):
    if ax is None:
        ax = plt.gca()

    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)

    # shape data
    xy = np.vstack([X.ravel(), Y.ravel()]).T

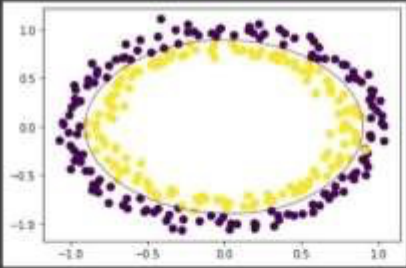
    # get the decision boundary based on the model
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary
    ax.contour(X, Y, P,
               levels=[0], alpha=0.5,
               linestyle='--')
```

```
exp 4 ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

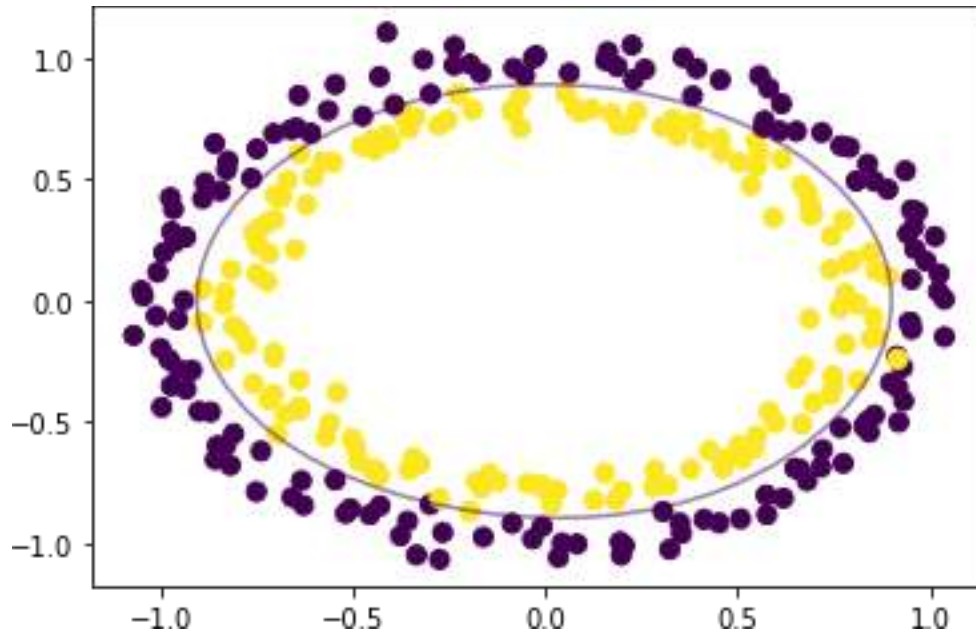
[27] ax.contour(X, Y, P,
             levels=[0], alpha=0.5,
             linestyle='--')

[28] # plot data and decision boundary
plt.scatter(circle_X[:, 0], circle_X[:, 1], c=circle_y, s=50)
plot_decision_boundary(nonlinear_clf)
plt.scatter(nonlinear_clf.support_vectors[:, 0], nonlinear_clf.support_vectors[:, 1], s=50, lw=1, facecolors='none')
plt.show()
```



The figure displays a 2D scatter plot with two classes of data points: purple and yellow. The data points are arranged in a ring-like pattern. A dashed line represents the decision boundary, which is non-linear and separates the two classes. The plot also shows the support vectors of the decision boundary, which are the points on the boundary that are closest to the origin of the decision boundary.

FINAL OUTPUT:



Learning outcomes (What I have learnt):

1. To understand Data Visualization.
2. Learn about pandas', matplotlib and seaborn library/package of python.
3. Learn about the different methods/functions that are needed to generate different types of graphs, charts and plots of the given dataset.
4. Leaned about regression line, KDE.