**“Experiment – 4”**

Name: **SUMIT KUMAR** UID: **20BCS8226**

Branch: **BE-CSE** Section/Group: **MM 808 A**

Semester: **5th** Date of Submission: **23/10/22**

Subject Name: **Machine Learning Lab** Subject Code: **20CSP-317**

**1. Aim/Overview of the practical:**

Implement Support Vector Machine on any data set and analyze the accuracy with Logistic regression.

**2. Task to be done/ Which logistics used:**

Implement SVM on any data set using sklearn.

**3. Steps for experiment/practical/Code:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn import svm

# a linear data

x=np.array([1,5,1.5,8,1,9,7,8.7,2.3,5.5,7.7,6.1,8,6])

y=np.array([2,8,1.8,8,0.6,11,10,9,4,3,8.8,7.5,12,5])

plt.scatter(x,y)

plt.show()

training\_X=np.vstack((x,y)).T

training\_Y=[0,1,0,1,0,1,1,1,0,0,1,1,1,0]

#DEFINE MODEL

clf=svm.SVC(kernel='linear', C=1.0)

clf.fit(training\_X,training\_Y)

# get the weight value for the linear equation from the trained SVM model

w=clf.coef\_[0]

# get the y-offset for the liner equation

a=-w[0]/w[1]

#make the x-axis space fo the data points

xx=np.linspace(0,10)

#get the y-values to plot the decision boundary

yy=a\*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()

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets

from sklearn import svm

circle\_x,circle\_y=datasets.make\_circles(n\_samples=300,noise=0.05)

plt.scatter(circle\_x[:,0],circle\_x[:,1],c=circle\_y,marker='.')

plt.show()

#make non-linear algorithm for model

nonlinear\_clf = svm.SVC(kernel='rbf',C=1.0)

#training non-linear model

nonlinear\_clf.fit(circle\_x,circle\_y)

#plot the decision boundary for 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

  P=model.decision\_function(xy).reshape(X.shape)

  #plot decision boundary

  ax.contour(X,Y,P,

             levels=[0],alpha=0.5,

             linestyles=['-'])

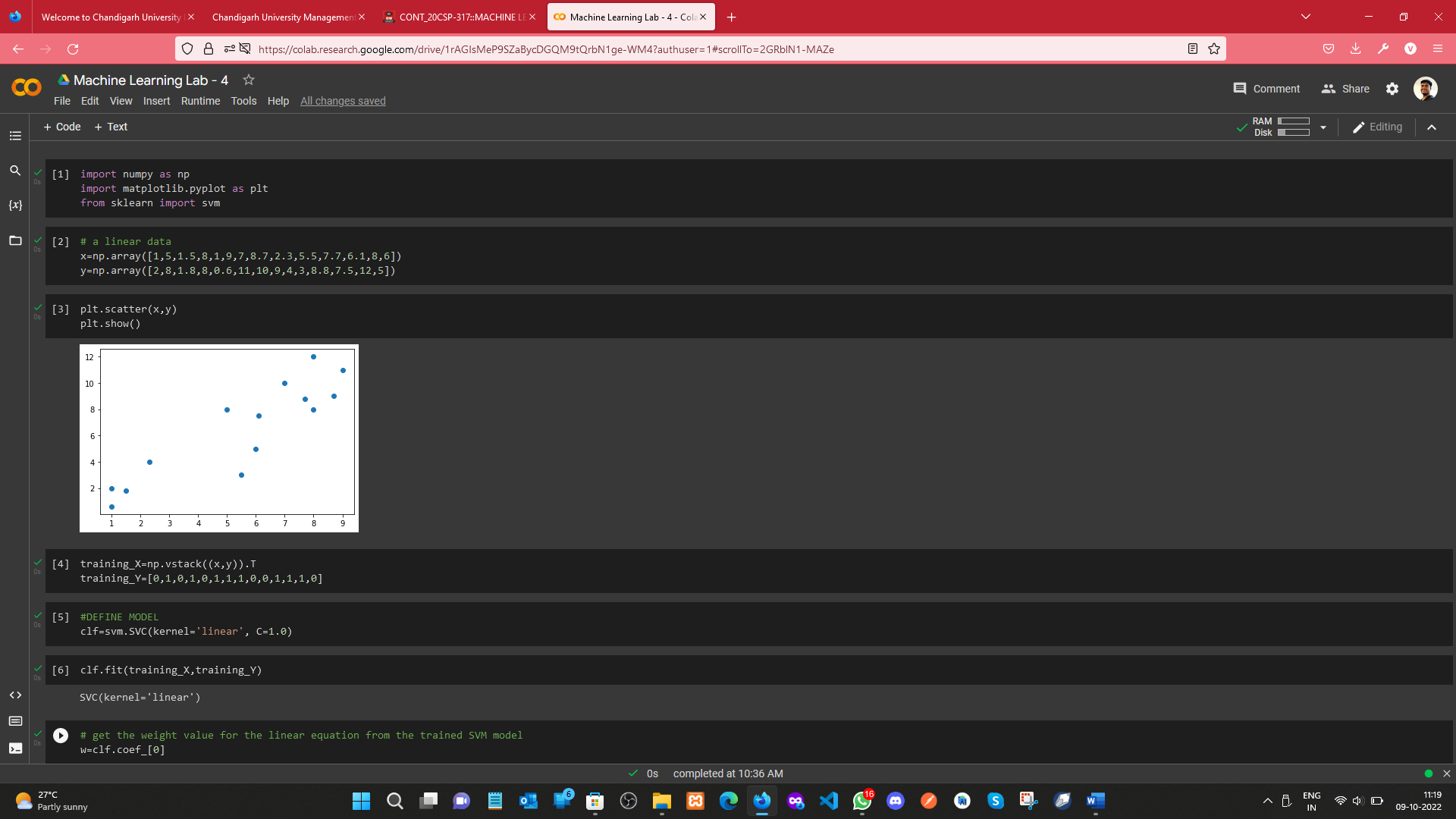
plt.scatter(circle\_x[:,0],circle\_x[:,1],c=circle\_y,s=50)

plot\_decision\_boundary(nonlinear\_clf)

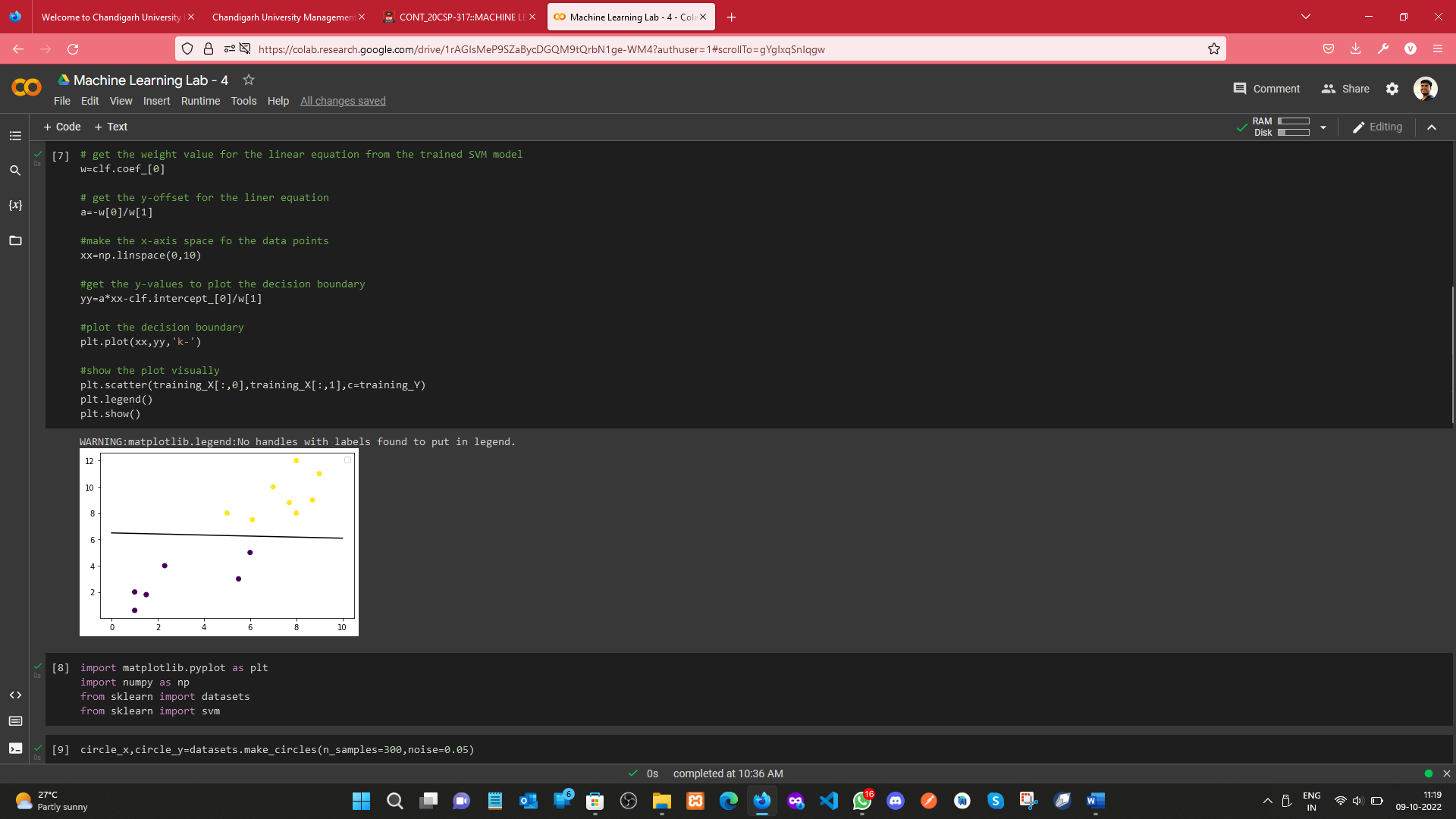
plt.scatter(nonlinear\_clf.support\_vectors\_[:,0], nonlinear\_clf.support\_vectors\_[:,1],s=50, lw=1, facecolors='none')

plt.show()

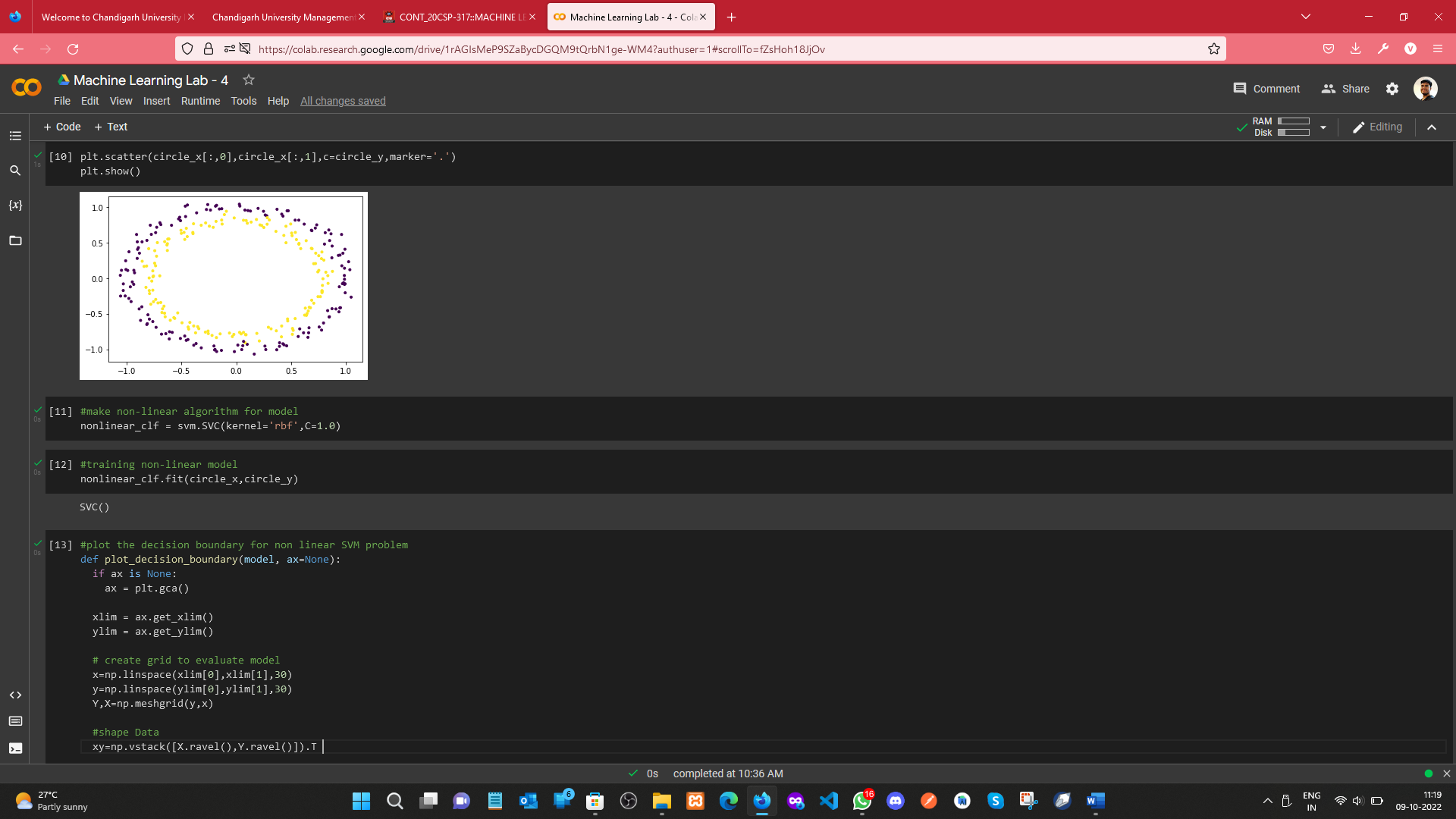
**4. Result/Output/Writing Summary:**



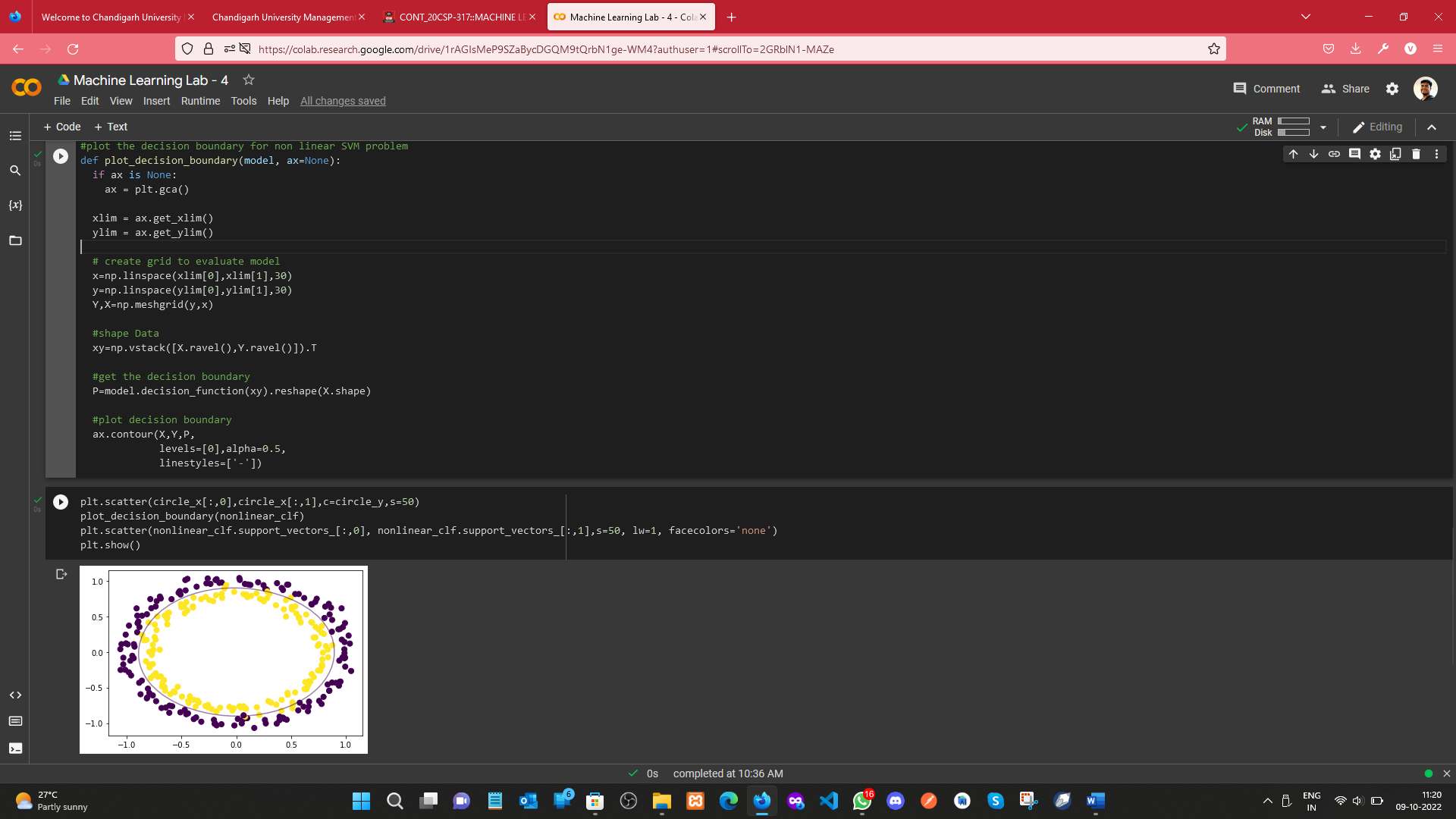
**Fig: Data Visualization Diagram 1**



**Fig: Data Visualization Diagram 2**



**Fig: Data Visualization Diagram 3**



**Fig: Data Visualization Diagram 4**

**Learning outcomes (What I have learnt):**

1. Understood the concept of SVM.
2. Learnt how to find the Hyperplane and Decision boundary.
3. Plotting the Hyperplane and Decision Boundary.

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |