LAB 5
Non-Linear Support Vector Machines

1. Create a contrived 2-class dataset that are non-linearly separable. Each sample is 2-dimensional. Plot the dataset using 2-d plot

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
print(plt.style.available)
plt.style.use('seaborn-dark-palette')
from sklearn.datasets import make_circles
from mpl_toolkits.mplot3d import Axes3D
```

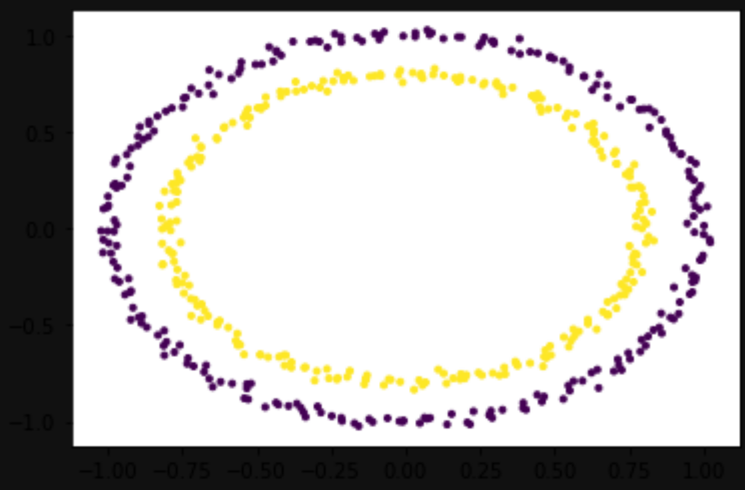
```
['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast',
'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorbl
ind', 'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgrid', 'seaborn-deep', 'seab
orn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 's
eaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblin
d10']
```

In [2]:

```
X, Y = make_circles(n_samples = 500, noise = 0.02)
```

In [3]:

```
plt.scatter(X[:, 0], X[:, 1], c = Y, marker = '.')
plt.show()
```



2. Write a program to visualize Gaussian kernel with various values for $\sigma \; and \; \mu$

In [4]:

```
X[:5]
```

Out[4]:

```
array([[ 0.77667872, -0.18866887],
       [ 0.31166387,  0.73430756],
       [ 1.01252788,  0.11270041],
       [-0.43491656, -0.90224886],
       [ 0.96693617,  0.06296941]])
```

In [5]:

```
# adding a new dimension to X
X1 = X[:, 0].reshape((-1, 1))
X2 = X[:, 1].reshape((-1, 1))
X3 = (X1**2 + X2**2)
X3[:5]
```

Out[5]:

```
array([[0.63882578],
       [0.63634197],
       [1.03791409],
       [1.00320543],
       [0.93893069]])
```

In [6]:

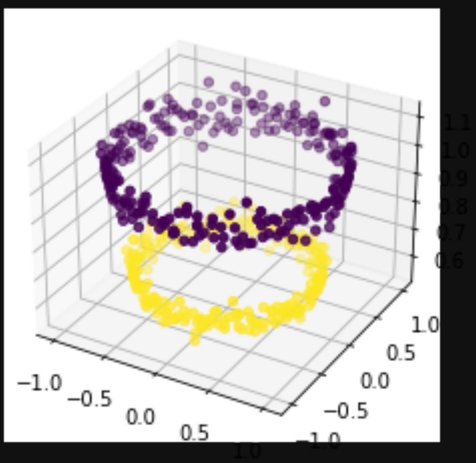
```
X = np.hstack((X, X3))
X[:5]
```

Out[6]:

```
array([[ 0.77667872, -0.18866887,  0.63882578],
       [ 0.31166387,  0.73430756,  0.63634197],
       [ 1.01252788,  0.11270041,  1.03791409],
       [-0.43491656, -0.90224886,  1.00320543],
       [ 0.96693617,  0.06296941,  0.93893069]])
```

In [7]:

```
# visualizing data in higher dimension
fig = plt.figure()
axes = fig.add_subplot(111, projection = '3d')
axes.scatter(X1, X2, X3, c = Y, depthshade = True)
plt.show()
```



3. Generate classification dataset with 2 classes where each sample is in 2-d space.

NB: Use function make_gaussian_quantiles https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_gaussian_quantiles.html

from sklearn.svm import SVC

Convert this dataset into 3d using RBF function and plot the dataset

In [8]:

```
from sklearn.datasets import make_gaussian_quantiles
import pandas as pd
import seaborn as sns
from sklearn.gaussian_process.kernels import RBF
from sklearn.preprocessing import StandardScaler
from scipy.spatial import distance
from sklearn import svm, datasets
```

In [9]:

```
X1, y1 = make_gaussian_quantiles(cov=3.0, n_samples=10000, n_features=2,
n_classes=2, random_state=1)
print(X1[:5])
print(y1[:5])
```

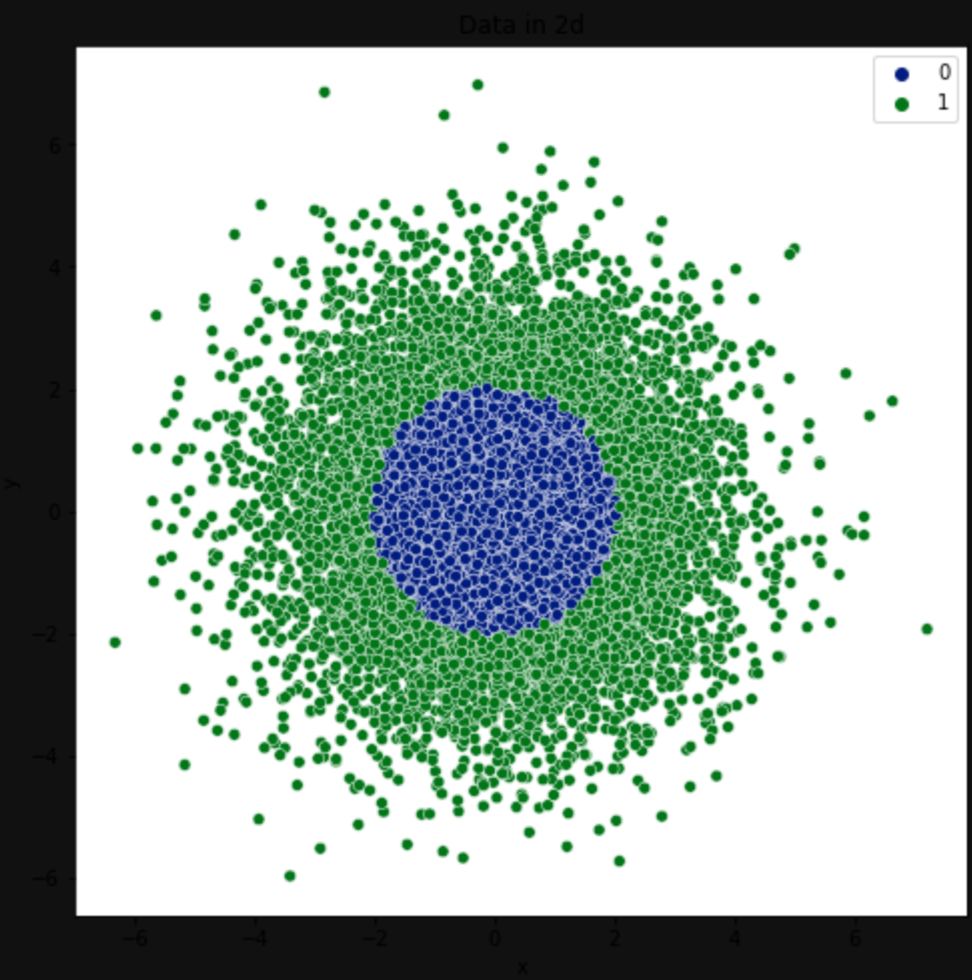
```
[[ 0.7597722   1.41831633]
 [ 2.42989589 -2.97483904]
 [-1.31266241 -3.83763022]
 [ 1.54424685  0.90423614]
 [ 0.67590516  3.47166431]]
[0 1 1 0 1]
```

In [10]:

```
X1 = pd.DataFrame(X1,columns=['x','y'])
y1 = pd.Series(y1)
```

In [11]:

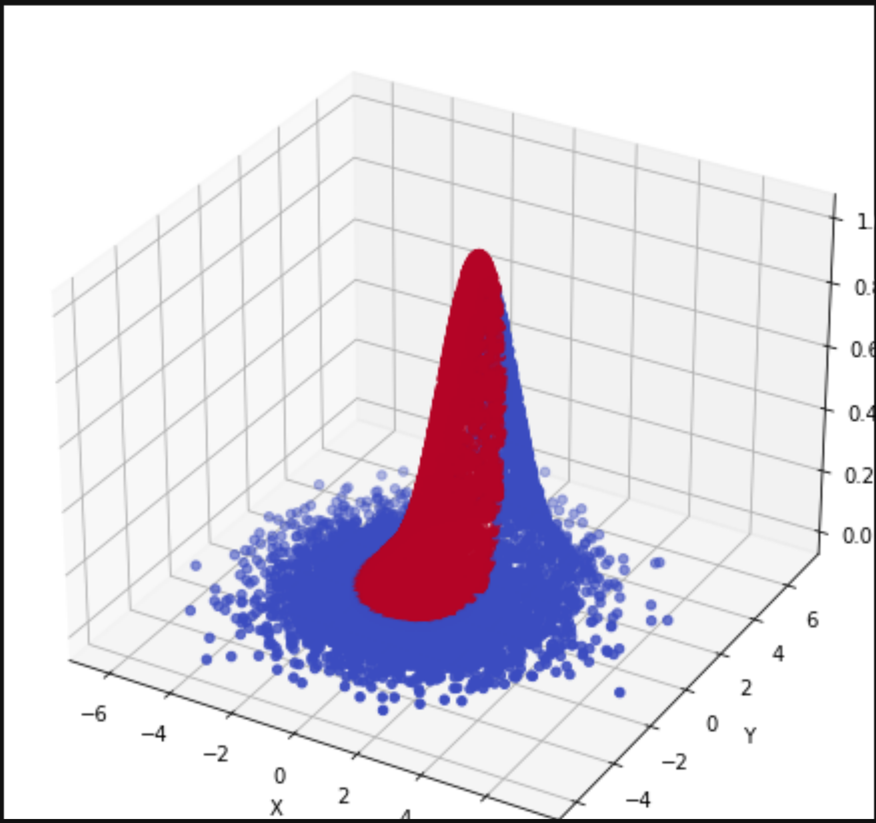
```
f, (ax1) = plt.subplots(nrows=1, ncols=1, figsize=(8,8))
sns.scatterplot(x = X1['x'], y = X1['y'], hue=y1, ax=ax1)
ax1.set_title("Data in 2d")
plt.show()
```



In [12]:

```
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')
colors = {
0: '#b40426',
1: '#3b4cc0',
2: '#f2da0a',
3: '#fe5200'
}

# Generate Z component
z = RBF(1.0).__call__(X1)[0]
print(z)
# Plot
colors = list(map(lambda x: colors[x], y1))
ax.scatter(X1['x'], X1['y'], z, c=colors, marker='o')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```

```
[1.00000000e+00 1.59737526e-05 1.17130203e-07 ... 1.26758885e-01
 2.67087386e-09 1.53720231e-01]
```



4. Fit the dataset created in Q.3 using SVM classifier with kernel = rbf, linear, and poly & various values for penalty term C.
    You may split the dataset into 80%-20% split.
    Plot 3 accuracy graphs.
    One with RBF with varying C.
    Second with Linear with varying C.
    Third with Polynomial kernel with varying C

In [13]:

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import plotly as pltly
import plotly.express as px
import plotly.graph_objects as go
pltly.offline.init_notebook_mode()
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.2,
random_state=0)
```

In [14]:

```python
def Plot_3D(X, X_test, y_test, clf, C, kernel):

    # Specify a size of the mesh to be used
    mesh_size = 5
    margin = 1

    # Create a mesh grid on which we will run our model
    x_min, x_max = X.iloc[:, 0].fillna(X.mean()).min() - margin, X.iloc[:, 0].fillna(X.mean()).max() + margin
    y_min, y_max = X.iloc[:, 1].fillna(X.mean()).min() - margin, X.iloc[:, 1].fillna(X.mean()).max() + margin
    xrange = np.arange(x_min, x_max, mesh_size)
    yrange = np.arange(y_min, y_max, mesh_size)
    xx, yy = np.meshgrid(xrange, yrange)

    # Calculate predictions on grid
    Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
    Z = Z.reshape(xx.shape)
    print(Z)
    # Create a 3D scatter plot with predictions
    fig = px.scatter_3d(x=X_test['x'], y=X_test['y'], z=y_test,
                     opacity=0.8, color_discrete_sequence=['black'],
                 width=500, height=500)

    # Set figure title and colors
    print("-----------------SVM Prediction Surface-----------------")
    title_text = "C=" + str(C) + " Kernel=" + str(kernel)
    fig.update_layout(title_text=title_text,
                      paper_bgcolor = 'white', margin=dict(l=5, r=5, t=40, b=20),
                      scene = dict(xaxis=dict(backgroundcolor='white',
                                              color='black',
                                              gridcolor='#f0f0f0'),
                                   yaxis=dict(backgroundcolor='white',
                                              color='black',
                                              gridcolor='#f0f0f0'
                                              ),
                                   zaxis=dict(backgroundcolor='lightgrey',
                                              color='black',
                                              gridcolor='#f0f0f0',
                                              )))
    # Update marker size
    fig.update_traces(marker=dict(size=1))

    # Add prediction plane
```

```
    # Add prediction plane
    fig.add_traces(go.Surface(x=xrange, y=yrange, z=Z, name='SVM
Prediction',
                              colorscale='RdBu', showscale=False,
                              contours = {"z": {"show": True, "start": 0.2,
"end": 0.8, "size": 0.05}}))
    fig.show()
```

In [15]:

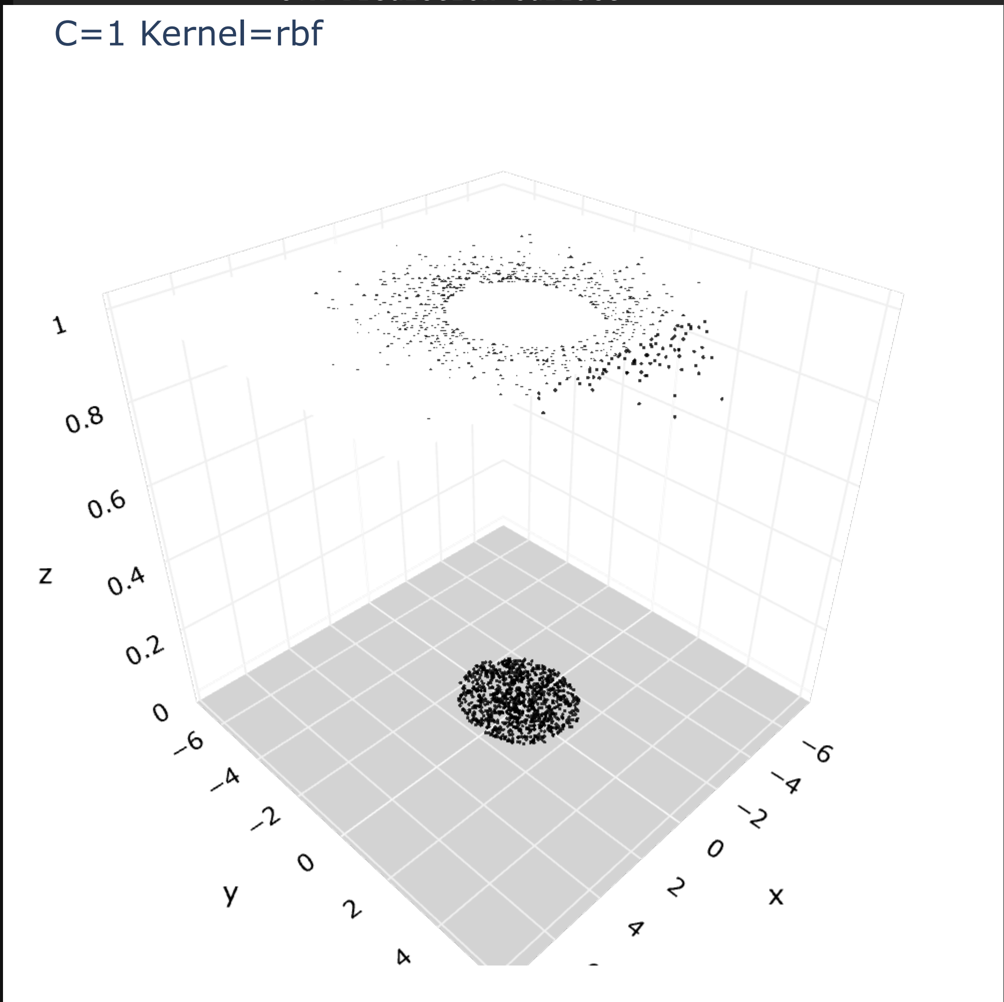
```
def fitting(X, y, C, gamma, kernel):
    print("--------------------------------------------------------Kernel = 
", kernel, " C = ", C, "
-----------------------------------------------------------")
    # Create training and testing samples
    X_train, X_test, y_train, y_test = train_test_split(X, y, 
test_size=0.2, random_state=0)

    # Fit the model
    # Note, available kernels: {'linear', 'poly', 'rbf', 'sigmoid', 
'precomputed'}, default='rbf'
    model = SVC(kernel=kernel, probability=True, C=C, gamma=gamma)
    clf = model.fit(X_train, y_train)

    # Predict class labels on training data
    pred_labels_tr = model.predict(X_train)
    # Predict class labels on a test data
    pred_labels_te = model.predict(X_test)

    # Use score method to get accuracy of the model
    print('--------------- Evaluation on Test Data ----------------')
    score_te = model.score(X_test, y_test)
    print('Accuracy Score: ', score_te)
    # Look at classification report to evaluate the model
    print(classification_report(y_test, pred_labels_te))
    print('-----------------------------------------------------------')

    print('------------- Evaluation on Training Data ---------------')
    score_tr = model.score(X_train, y_train)
    print('Accuracy Score: ', score_tr)
    # Look at classification report to evaluate the model
    print(classification_report(y_train, pred_labels_tr))
    print('-----------------------------------------------------------')

    # Return relevant data for chart plotting
    Plot_3D(X, X_test, y_test, clf, C, kernel)
    return X_train, X_test, y_train, y_test, clf
```

In [19]:

```
C = [1,2,3]
for c in C:
    rbf_X_train, rbf_X_test, rbf_y_train, rbf_y_test, rbf_clf = fitting(X1, 
y1, c, 'scale', 'rbf')
```

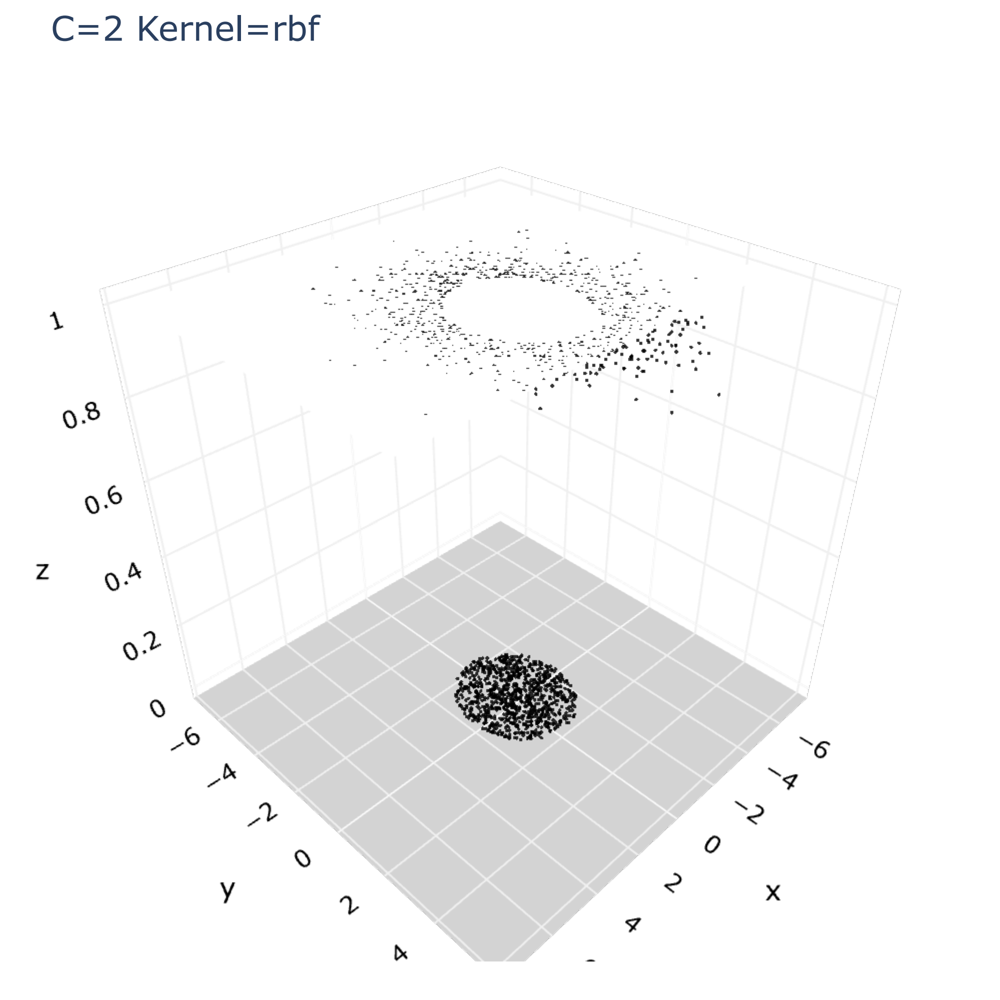
```
--------------------------------------------------------Kernel =  rbf  C =  1
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
```

```
Accuracy Score:  0.9975
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       987
           1       1.00      1.00      1.00      1013

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
Accuracy Score:  0.997625
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4013
           1       1.00      1.00      1.00      3987

    accuracy                           1.00      8000
   macro avg       1.00      1.00      1.00      8000
weighted avg       1.00      1.00      1.00      8000

--------------------------------------------------------
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
-----------------SVM Prediction Surface-----------------
```



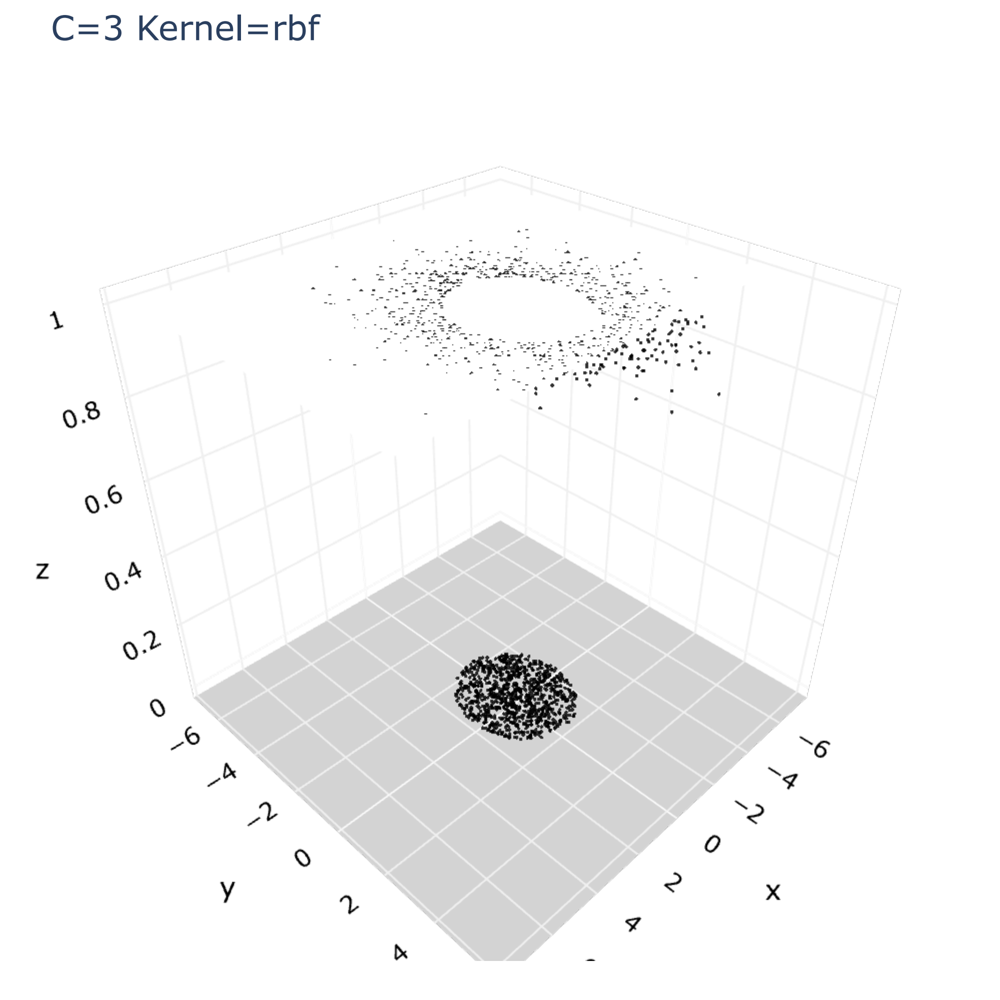

```
--------------------------------------------------------Kernel =  rbf  C =  2
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
Accuracy Score:  0.998
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       987
           1       1.00      1.00      1.00      1013

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
```

```
Accuracy Score:  0.997875
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4013
           1       1.00      1.00      1.00      3987

    accuracy                           1.00      8000
   macro avg       1.00      1.00      1.00      8000
weighted avg       1.00      1.00      1.00      8000

--------------------------------------------------------
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
-----------------SVM Prediction Surface-----------------
```




```
--------------------------------------------------------Kernel =  rbf  C =  3
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
Accuracy Score:  0.999
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       987
           1       1.00      1.00      1.00      1013

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
Accuracy Score:  0.9985
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4013
           1       1.00      1.00      1.00      3987

    accuracy                           1.00      8000
   macro avg       1.00      1.00      1.00      8000
weighted avg       1.00      1.00      1.00      8000

--------------------------------------------------------
[[1. 1. 1. 1.]
```

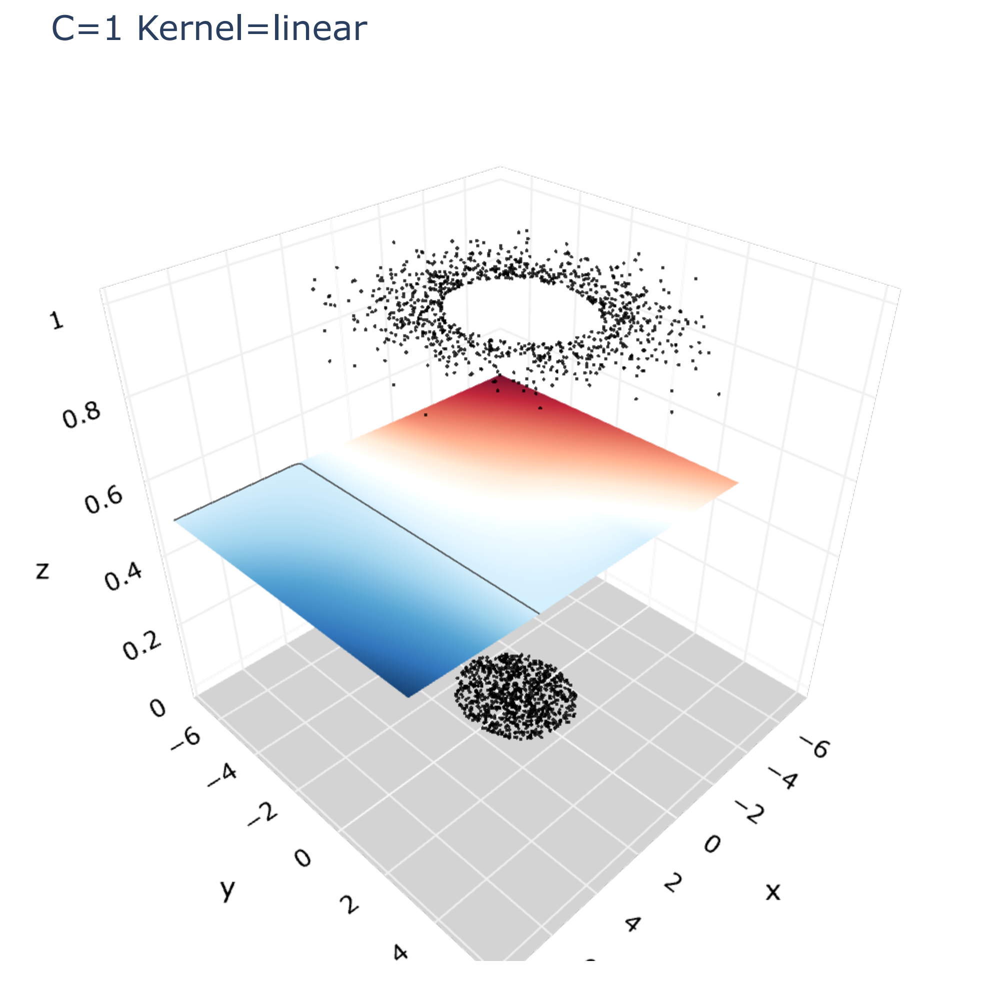
```
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
-----------------SVM Prediction Surface-----------------
```



In [17]:

```
C = [1,2,3]
for c in C:
    poly_X_train, poly_X_test, poly_y_train, poly_y_test, poly_clf = fitting(X1, y1, c, 'scale', 'linear')
```

```
--------------------------------------------------------Kernel =  linear  C =  1
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
Accuracy Score:  0.623
              precision    recall  f1-score   support

           0       0.57      0.92      0.71       987
           1       0.81      0.34      0.47      1013

    accuracy                           0.62      2000
   macro avg       0.69      0.63      0.59      2000
weighted avg       0.69      0.62      0.59      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
Accuracy Score:  0.635
              precision    recall  f1-score   support

           0       0.59      0.93      0.72      4013
           1       0.82      0.34      0.48      3987

    accuracy                           0.64      8000
   macro avg       0.70      0.63      0.60      8000
weighted avg       0.70      0.64      0.60      8000

--------------------------------------------------------
[[0.48517194 0.49067862 0.5        0.5       ]
 [0.4888388  0.49434735 0.5        0.50536745]
 [0.49250702 0.5        0.5        0.50903633]]
-----------------SVM Prediction Surface-----------------
```




```
--------------------------------------------------------Kernel =  linear  C =  2
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
Accuracy Score:  0.623
              precision    recall  f1-score   support

           0       0.57      0.92      0.71       987
           1       0.81      0.34      0.47      1013

    accuracy                           0.62      2000
   macro avg       0.69      0.63      0.59      2000
weighted avg       0.69      0.62      0.59      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
Accuracy Score:  0.635
              precision    recall  f1-score   support

           0       0.59      0.93      0.72      4013
           1       0.82      0.34      0.48      3987

    accuracy                           0.64      8000
   macro avg       0.70      0.63      0.60      8000
weighted avg       0.70      0.64      0.60      8000

--------------------------------------------------------
[[0.4590701  0.47538658 0.49177075 0.50817536]
 [0.46992419 0.48629149 0.5        0.51909176]
 [0.48081606 0.5        0.51361468 0.52999462]]
-----------------SVM Prediction Surface-----------------
```

```
--------------------------------------------------------Kernel =  linear  C =  3
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
Accuracy Score:  0.623
              precision    recall  f1-score   support

           0       0.57      0.92      0.71       987
           1       0.81      0.34      0.47      1013

    accuracy                           0.62      2000
   macro avg       0.69      0.63      0.59      2000
weighted avg       0.69      0.62      0.59      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
Accuracy Score:  0.635
              precision    recall  f1-score   support

           0       0.59      0.93      0.72      4013
           1       0.82      0.34      0.48      3987

    accuracy                           0.64      8000
   macro avg       0.70      0.63      0.60      8000
weighted avg       0.70      0.64      0.60      8000

--------------------------------------------------------
[[0.56500794 0.53955224 0.51400435 0.48840352]
 [0.54806764 0.52255076 0.5        0.47138747]
 [0.53108788 0.50551353 0.47991941 0.45445861]]
-----------------SVM Prediction Surface-----------------
```

In [18]:
```
C = [1,2,3]
for c in C:
    poly_X_train, poly_X_test, poly_y_train, poly_y_test, poly_clf =
fitting(X1, y1, c, 'scale', 'poly')
```

```
--------------------------------------------------------Kernel =  poly  C =  1
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
Accuracy Score:  0.505
              precision    recall  f1-score   support

           0       0.50      1.00      0.67       987
           1       1.00      0.02      0.04      1013

    accuracy                           0.51      2000
   macro avg       0.75      0.51      0.36      2000
weighted avg       0.75      0.51      0.35      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
Accuracy Score:  0.519375
              precision    recall  f1-score   support

           0       0.51      1.00      0.68      4013
           1       1.00      0.04      0.07      3987

    accuracy                           0.52      8000
   macro avg       0.76      0.52      0.37      8000
weighted avg       0.75      0.52      0.37      8000

--------------------------------------------------------
[[0.64510829 0.56015944 0.56597733 0.58044104]
 [0.56257897 0.50706568 0.5        0.47336027]
 [0.51544009 0.5        0.49418745 0.42543161]]
-----------------SVM Prediction Surface-----------------
```

```
--------------------------------------------------------Kernel =  poly  C =  2
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
Accuracy Score:  0.505
              precision    recall  f1-score   support

           0       0.50      1.00      0.67       987
           1       1.00      0.02      0.04      1013

    accuracy                           0.51      2000
   macro avg       0.75      0.51      0.36      2000
weighted avg       0.75      0.51      0.35      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
Accuracy Score:  0.519375
              precision    recall  f1-score   support

           0       0.51      1.00      0.68      4013
           1       1.00      0.04      0.07      3987

    accuracy                           0.52      8000
   macro avg       0.76      0.52      0.37      8000
weighted avg       0.75      0.52      0.37      8000

--------------------------------------------------------
[[0.53786208 0.51422412 0.51583449 0.51990011]
 [0.51493596 0.5        0.5        0.49012383]
 [0.5        0.5        0.5        0.47660678]]
-----------------SVM Prediction Surface-----------------
```

```
--------------------------------------------------------Kernel =  poly  C =  3
--------------------------------------------------------
--------------- Evaluation on Test Data ----------------
Accuracy Score:  0.505
```

```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67       987
           1       1.00      0.02      0.04      1013

    accuracy                           0.51      2000
   macro avg       0.75      0.51      0.36      2000
weighted avg       0.75      0.51      0.35      2000

--------------------------------------------------------
------------- Evaluation on Training Data --------------
Accuracy Score:  0.519375
              precision    recall  f1-score   support

           0       0.51      1.00      0.68      4013
           1       1.00      0.04      0.07      3987

    accuracy                           0.52      8000
   macro avg       0.76      0.52      0.37      8000
weighted avg       0.75      0.52      0.37      8000

--------------------------------------------------------
[[0.6825755  0.57548906 0.58328661 0.60303341]
 [0.5789389  0.5        0.5        0.45827743]
 [0.51503986 0.49015612 0.48632833 0.39446072]]
-----------------SVM Prediction Surface-----------------
```