In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

LINESPACE CREATES 100 NUMBERS FROM THE RANGE -0.5 TO 5.0

In [8]:
```
x=np.linspace(-0.5,5.0,100)
y=np.sqrt(10**2-x**2)
y=np.hstack([y,-y])
x=np.hstack([x,-x])
```

In [3]:
```
x1=np.linspace(-5.0,5.0,100)
x1
```

Out[3]:
```
array([-5.        , -4.8989899 , -4.7979798 , -4.6969697 , -4.5959596 ,
       -4.49494949, -4.39393939, -4.29292929, -4.19191919, -4.09090909,
       -3.98989899, -3.88888889, -3.78787879, -3.68686869, -3.58585859,
       -3.48484848, -3.38383838, -3.28282828, -3.18181818, -3.08080808,
       -2.97979798, -2.87878788, -2.77777778, -2.67676768, -2.57575758,
       -2.47474747, -2.37373737, -2.27272727, -2.17171717, -2.07070707,
       -1.96969697, -1.86868687, -1.76767677, -1.66666667, -1.56565657,
       -1.46464646, -1.36363636, -1.26262626, -1.16161616, -1.06060606,
       -0.95959596, -0.85858586, -0.75757576, -0.65656566, -0.55555556,
       -0.45454545, -0.35353535, -0.25252525, -0.15151515, -0.05050505,
        0.05050505,  0.15151515,  0.25252525,  0.35353535,  0.45454545,
        0.55555556,  0.65656566,  0.75757576,  0.85858586,  0.95959596,
        1.06060606,  1.16161616,  1.26262626,  1.36363636,  1.46464646,
        1.56565657,  1.66666667,  1.76767677,  1.86868687,  1.96969697,
        2.07070707,  2.17171717,  2.27272727,  2.37373737,  2.47474747,
        2.57575758,  2.67676768,  2.77777778,  2.87878788,  2.97979798,
        3.08080808,  3.18181818,  3.28282828,  3.38383838,  3.48484848,
        3.58585859,  3.68686869,  3.78787879,  3.88888889,  3.98989899,
        4.09090909,  4.19191919,  4.29292929,  4.39393939,  4.49494949,
        4.5959596 ,  4.6969697 ,  4.7979798 ,  4.8989899 ,  5.        ])
```

Creating y1 variable, squaring the numbers and storing the dataset in y1

In [4]:
```
y1=np.sqrt(5**2-x1**2)
y1
```

Out[4]:
```
array([0.        , 0.99994898, 1.40690791, 1.71419826, 1.96904936,
       2.18984681, 2.38606299, 2.563349  , 2.72540153, 2.87479787,
       3.01342099, 3.14269681, 3.26373625, 3.37742495, 3.48448249,
       3.5855029 , 3.68098326, 3.77134438, 3.85694608, 3.93809873,
       4.0150721 , 4.08810229, 4.1573971 , 4.2231404 , 4.28549564,
       4.34460872, 4.40061029, 4.45361771, 4.50373673, 4.55106276,
       4.59568209, 4.63767284, 4.67710582, 4.71404521, 4.7485492 ,
       4.78067053, 4.81045693, 4.83795152, 4.86319318, 4.88621682,
       4.90705366, 4.92573145, 4.94227468, 4.95670471, 4.96903995,
       4.97929598, 4.98748561, 4.99361903, 4.99770379, 4.99974492,
       4.99974492, 4.99770379, 4.99361903, 4.98748561, 4.97929598,
       4.96903995, 4.95670471, 4.94227468, 4.92573145, 4.90705366,
       4.88621682, 4.86319318, 4.83795152, 4.81045693, 4.78067053,
       4.7485492 , 4.71404521, 4.67710582, 4.63767284, 4.59568209,
       4.55106276, 4.50373673, 4.45361771, 4.40061029, 4.34460872,
       4.28549564, 4.2231404 , 4.1573971 , 4.08810229, 4.0150721 ,
       3.93809873, 3.85694608, 3.77134438, 3.68098326, 3.5855029 ,
       3.48448249, 3.37742495, 3.26373625, 3.14269681, 3.01342099,
       2.87479787, 2.72540153, 2.563349  , 2.38606299, 2.18984681,
       1.96904936, 1.71419826, 1.40690791, 0.99994898, 0.        ])
```

hstack- Stacking the data in horizontal order parameters:(y1 data, coverting all y1 values to y1 negative values)

In [5]:
```
y1=np.hstack([y1,-y1])
y1
```

Out[5]:
```
array([ 0.        ,  0.99994898,  1.40690791,  1.71419826,  1.96904936,
        2.18984681,  2.38606299,  2.563349  ,  2.72540153,  2.87479787,
        3.01342099,  3.14269681,  3.26373625,  3.37742495,  3.48448249,
        3.5855029 ,  3.68098326,  3.77134438,  3.85694608,  3.93809873,
        4.0150721 ,  4.08810229,  4.1573971 ,  4.2231404 ,  4.28549564,
        4.34460872,  4.40061029,  4.45361771,  4.50373673,  4.55106276,
        4.59568209,  4.63767284,  4.67710582,  4.71404521,  4.7485492 ,
        4.78067053,  4.81045693,  4.83795152,  4.86319318,  4.88621682,
        4.90705366,  4.92573145,  4.94227468,  4.95670471,  4.96903995,
        4.97929598,  4.98748561,  4.99361903,  4.99770379,  4.99974492,
        4.99974492,  4.99770379,  4.99361903,  4.98748561,  4.97929598,
        4.96903995,  4.95670471,  4.94227468,  4.92573145,  4.90705366,
        4.88621682,  4.86319318,  4.83795152,  4.81045693,  4.78067053,
        4.7485492 ,  4.71404521,  4.67710582,  4.63767284,  4.59568209,
        4.55106276,  4.50373673,  4.45361771,  4.40061029,  4.34460872,
        4.28549564,  4.2231404 ,  4.1573971 ,  4.08810229,  4.0150721 ,
        3.93809873,  3.85694608,  3.77134438,  3.68098326,  3.5855029 ,
        3.48448249,  3.37742495,  3.26373625,  3.14269681,  3.01342099,
        2.87479787,  2.72540153,  2.563349  ,  2.38606299,  2.18984681,
        1.96904936,  1.71419826,  1.40690791,  0.99994898,  0.        ,
       -0.        , -0.99994898, -1.40690791, -1.71419826, -1.96904936,
       -2.18984681, -2.38606299, -2.563349  , -2.72540153, -2.87479787,
       -3.01342099, -3.14269681, -3.26373625, -3.37742495, -3.48448249,
       -3.5855029 , -3.68098326, -3.77134438, -3.85694608, -3.93809873,
       -4.0150721 , -4.08810229, -4.1573971 , -4.2231404 , -4.28549564,
       -4.34460872, -4.40061029, -4.45361771, -4.50373673, -4.55106276,
       -4.59568209, -4.63767284, -4.67710582, -4.71404521, -4.7485492 ,
       -4.78067053, -4.81045693, -4.83795152, -4.86319318, -4.88621682,
       -4.90705366, -4.92573145, -4.94227468, -4.95670471, -4.96903995,
       -4.97929598, -4.98748561, -4.99361903, -4.99770379, -4.99974492,
       -4.99974492, -4.99770379, -4.99361903, -4.98748561, -4.97929598,
       -4.96903995, -4.95670471, -4.94227468, -4.92573145, -4.90705366,
       -4.88621682, -4.86319318, -4.83795152, -4.81045693, -4.78067053,
       -4.7485492 , -4.71404521, -4.67710582, -4.63767284, -4.59568209,
       -4.55106276, -4.50373673, -4.45361771, -4.40061029, -4.34460872,
       -4.28549564, -4.2231404 , -4.1573971 , -4.08810229, -4.0150721 ,
       -3.93809873, -3.85694608, -3.77134438, -3.68098326, -3.5855029 ,
       -3.48448249, -3.37742495, -3.26373625, -3.14269681, -3.01342099,
       -2.87479787, -2.72540153, -2.563349  , -2.38606299, -2.18984681,
       -1.96904936, -1.71419826, -1.40690791, -0.99994898, -0.        ])
```

hstack-stacking the data in horizontal order parameters:(x1 data,converting all x1 values to x1 negative values)

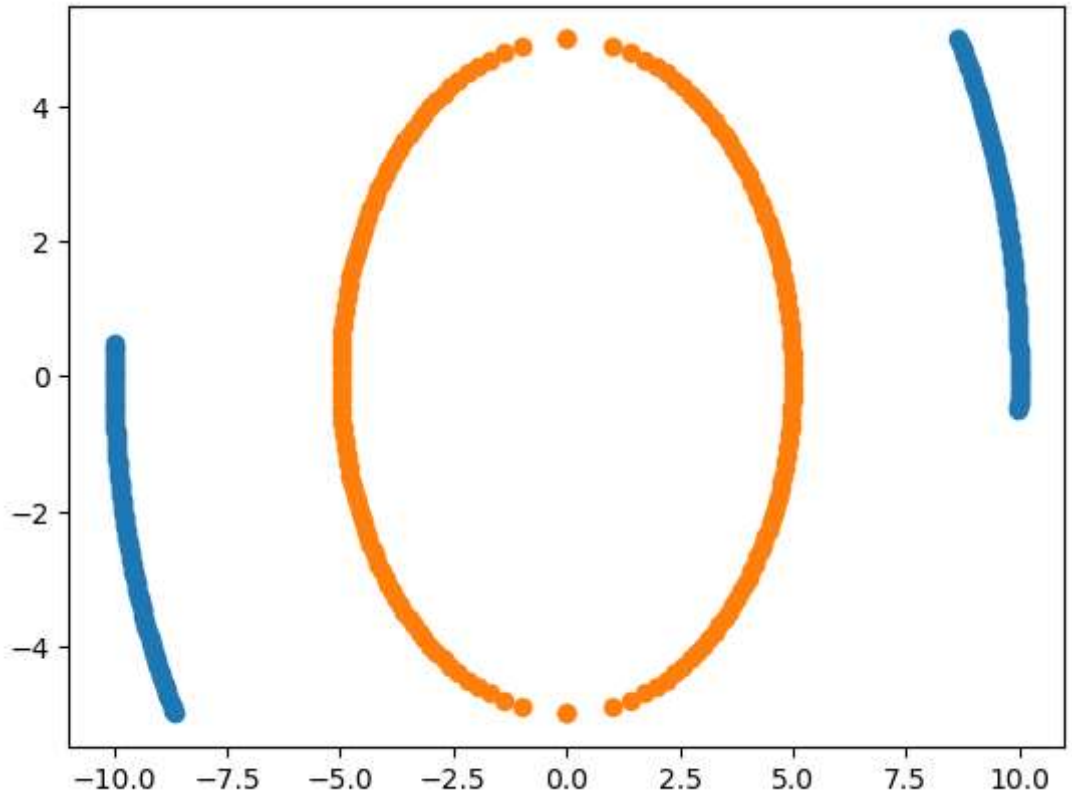
In [6]:
```
x1=np.hstack([x1,-x1])
x1
```

Out[6]:
```
array([-5.        , -4.8989899 , -4.7979798 , -4.6969697 , -4.5959596 ,
       -4.49494949, -4.39393939, -4.29292929, -4.19191919, -4.09090909,
       -3.98989899, -3.88888889, -3.78787879, -3.68686869, -3.58585859,
       -3.48484848, -3.38383838, -3.28282828, -3.18181818, -3.08080808,
       -2.97979798, -2.87878788, -2.77777778, -2.67676768, -2.57575758,
       -2.47474747, -2.37373737, -2.27272727, -2.17171717, -2.07070707,
       -1.96969697, -1.86868687, -1.76767677, -1.66666667, -1.56565657,
       -1.46464646, -1.36363636, -1.26262626, -1.16161616, -1.06060606,
       -0.95959596, -0.85858586, -0.75757576, -0.65656566, -0.55555556,
       -0.45454545, -0.35353535, -0.25252525, -0.15151515, -0.05050505,
        0.05050505,  0.15151515,  0.25252525,  0.35353535,  0.45454545,
        0.55555556,  0.65656566,  0.75757576,  0.85858586,  0.95959596,
        1.06060606,  1.16161616,  1.26262626,  1.36363636,  1.46464646,
        1.56565657,  1.66666667,  1.76767677,  1.86868687,  1.96969697,
        2.07070707,  2.17171717,  2.27272727,  2.37373737,  2.47474747,
        2.57575758,  2.67676768,  2.77777778,  2.87878788,  2.97979798,
        3.08080808,  3.18181818,  3.28282828,  3.38383838,  3.48484848,
        3.58585859,  3.68686869,  3.78787879,  3.88888889,  3.98989899,
        4.09090909,  4.19191919,  4.29292929,  4.39393939,  4.49494949,
        4.5959596 ,  4.6969697 ,  4.7979798 ,  4.8989899 ,  5.        ,
        5.        ,  4.8989899 ,  4.7979798 ,  4.6969697 ,  4.5959596 ,
        4.49494949,  4.39393939,  4.29292929,  4.19191919,  4.09090909,
        3.98989899,  3.88888889,  3.78787879,  3.68686869,  3.58585859,
        3.48484848,  3.38383838,  3.28282828,  3.18181818,  3.08080808,
        2.97979798,  2.87878788,  2.77777778,  2.67676768,  2.57575758,
        2.47474747,  2.37373737,  2.27272727,  2.17171717,  2.07070707,
        1.96969697,  1.86868687,  1.76767677,  1.66666667,  1.56565657,
        1.46464646,  1.36363636,  1.26262626,  1.16161616,  1.06060606,
        0.95959596,  0.85858586,  0.75757576,  0.65656566,  0.55555556,
        0.45454545,  0.35353535,  0.25252525,  0.15151515,  0.05050505,
       -0.05050505, -0.15151515, -0.25252525, -0.35353535, -0.45454545,
       -0.55555556, -0.65656566, -0.75757576, -0.85858586, -0.95959596,
       -1.06060606, -1.16161616, -1.26262626, -1.36363636, -1.46464646,
       -1.56565657, -1.66666667, -1.76767677, -1.86868687, -1.96969697,
       -2.07070707, -2.17171717, -2.27272727, -2.37373737, -2.47474747,
       -2.57575758, -2.67676768, -2.77777778, -2.87878788, -2.97979798,
       -3.08080808, -3.18181818, -3.28282828, -3.38383838, -3.48484848,
       -3.58585859, -3.68686869, -3.78787879, -3.88888889, -3.98989899,
       -4.09090909, -4.19191919, -4.29292929, -4.39393939, -4.49494949,
       -4.5959596 , -4.6969697 , -4.7979798 , -4.8989899 , -5.        ])
```

Using the above x,y,x1&y1 dataset 2 Dimensional scatter plot is created

In [9]:
```
plt.scatter(y,x)
plt.scatter(y1,x1)
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x25bb0e416a0>
```



For Outerline(Blue Color) are basically zero

For Innerline(Orange Color) are basically one

Vstack- Stacking the data in vertically to create a DataFrame

In [11]:
```
df1=pd.DataFrame(np.vstack([y,x]).T,columns=['X1','X2'])
df1['Y']=0
```

```
df2=pd.DataFrame(np.vstack([y1,x1]).T,columns=['X1','X2'])
df2['Y']=1
df=df1.append(df2)
df.head(5)
```

```
C:\Users\KIRAN\AppData\Local\Temp\ipykernel_22288\3102956401.py:5: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  df=df1.append(df2)
```

Out[11]:

	X1	X2	Y
0	9.987492	-0.500000	0
1	9.990119	-0.444444	0
2	9.992435	-0.388889	0
3	9.994443	-0.333333	0
4	9.996141	-0.277778	0

X- Independet features Y-Dependent features

In [15]:
```
X=df.iloc[:,:-1]
```

In [16]:
```
y=df.Y
```

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

Out[17]:
```
0      0
1      0
2      0
3      0
4      0
      ..
195    1
196    1
197    1
198    1
199    1
Name: Y, Length: 400, dtype: int64
```

Splitting the dataset into train and test with 25% of test data

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=0.25,random_state=0)
```

In [20]:
```
y_train
```

Out[20]:
```
50     1
63     0
112    1
159    0
83     1
      ..
123    1
192    0
117    0
47     0
172    0
Name: Y, Length: 300, dtype: int64
```

Importing SVM Class model from sklearn library

This is using Kernal as 'Linear'

In [21]:
```
from sklearn.svm import SVC
classifier=SVC(kernel='linear')
classifier.fit(X_train,y_train)
```

Out[21]:
```
SVC(kernel='linear')
```

Importing accuracy_score model from sklearn library metrics

In [22]:
```
from sklearn.metrics import accuracy_score
y_pred=classifier.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[22]:
```
0.45
```

The Accuracy Score is less than 50% not so good not linear seperable

In [23]:
```
df.head()
```

Out[23]:

	X1	X2	Y
0	9.987492	-0.500000	0
1	9.990119	-0.444444	0
2	9.992435	-0.388889	0
3	9.994443	-0.333333	0
4	9.996141	-0.277778	0

Applying Polynominal Kernel

In [24]:
```
# We need to findout components for the polynomial kernel
#X1,X2,X1_square,X1*X2
df['X1_Square']=df['X1']**2
df['X2_Square']=df['X2']**2
df['X1*X2']=(df['X1']*df['X2'])
df.head()
```

Out[24]:

	X1	X2	Y	X1_Square	X2_Square	X1*X2
0	9.987492	-0.500000	0	99.750000	0.250000	-4.993746
1	9.990119	-0.444444	0	99.802469	0.197531	-4.440053
2	9.992435	-0.388889	0	99.848765	0.151235	-3.885947
3	9.994443	-0.333333	0	99.888889	0.111111	-3.331481
4	9.996141	-0.277778	0	99.922840	0.077160	-2.776706

Independent features and Dependent features

In [25]:
```
X=df[['X1','X2','X1_Square','X2_Square','X1*X2']]
y=df['Y']
```

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

Out[26]:
```
0      0
1      0
2      0
3      0
4      0
      ..
195    1
196    1
197    1
198    1
199    1
Name: Y, Length: 400, dtype: int64
```

Splitting the dataset into train and test with 25% of Test data

In [27]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=0)
```

In [28]:
```
X_train
```

Out[28]:

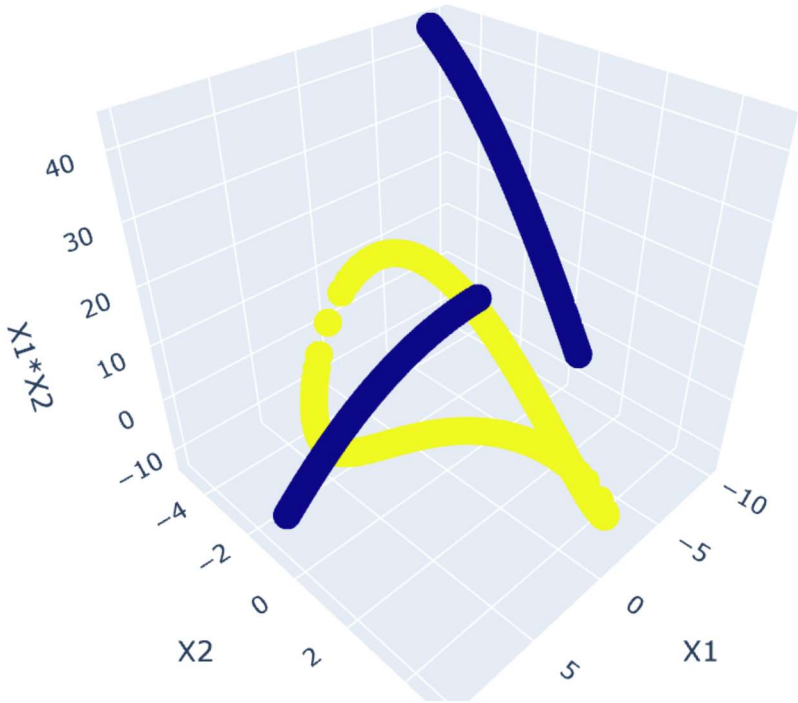
	X1	X2	X1_Square	X2_Square	X1*X2
50	4.999745	0.050505	24.997449	0.002551	0.252512
63	9.539392	3.000000	91.000000	9.000000	28.618176
112	-3.263736	3.787879	10.651974	14.348026	-12.362637
159	-9.606454	-2.777778	92.283951	7.716049	26.684593
83	3.680983	3.383838	13.549638	11.450362	12.455852
...	...	...	...	...	...
123	-4.223140	2.676768	17.834915	7.165085	-11.304366
192	-8.873424	-4.611111	78.737654	21.262346	40.916344
117	-9.990119	-0.444444	99.802469	0.197531	4.440053
47	9.774621	2.111111	95.543210	4.456790	20.635310
172	-9.367497	-3.500000	87.750000	12.250000	32.786239

300 rows × 5 columns

Importing Plotly.express for 3D Graph representation

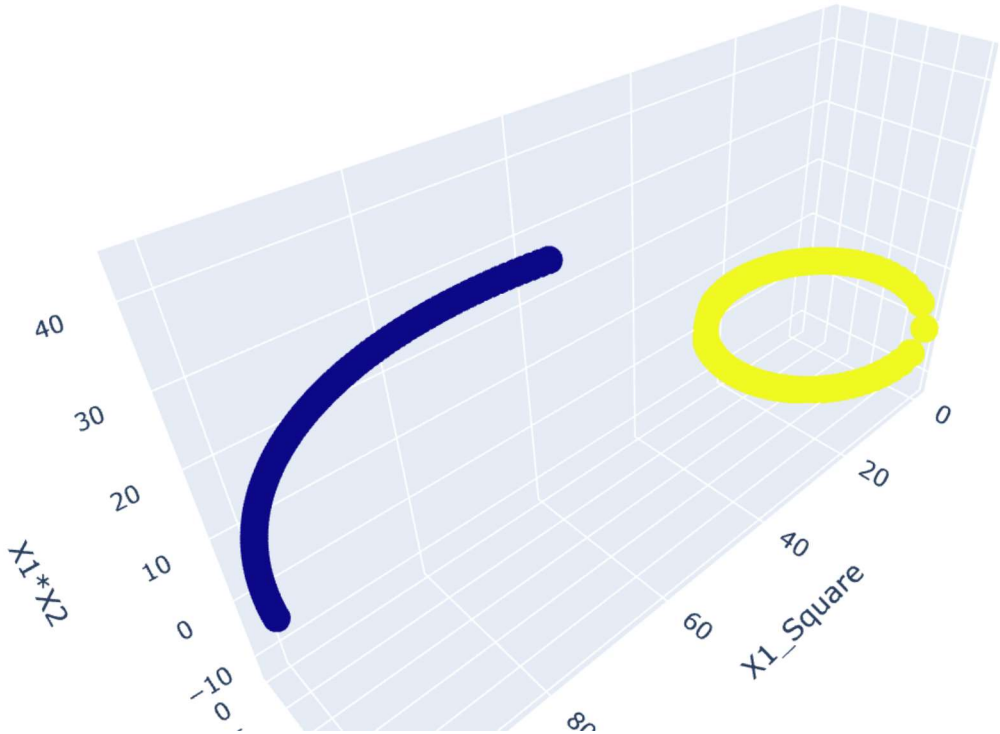
In the below diagram we cant draw a plane

In [30]:
```
import plotly.express as px
fig=px.scatter_3d(df,x='X1',y='X2',z='X1*X2',color='Y')
fig.show()
```



Using X1_Square & X2_Square features

In [31]:
```
fig=px.scatter_3d(df,x='X1_Square',y='X2_Square',z='X1*X2',color='Y')
fig.show()
```



In the above diagram we could draw a hyperplane

In [33]:
```
classifier=SVC(kernel='linear')
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[33]:
```
1.0
```

Accuracy is 100% it is clearly seperable

Let` use Logistic Regression to find the Accuracy Score

In [34]:
```
from sklearn.linear_model import LogisticRegression
class_reg=LogisticRegression()
```

In [35]:
```
class_reg.fit(X_train,y_train)
```

Out[35]:
```
LogisticRegression
LogisticRegression()
```

In [36]:
```
y_pred=class_reg.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[36]:
```
1.0
```

In Logistic Regression also the Accuracy Score is 100% means it is easily separable

In [ ]:
```

```