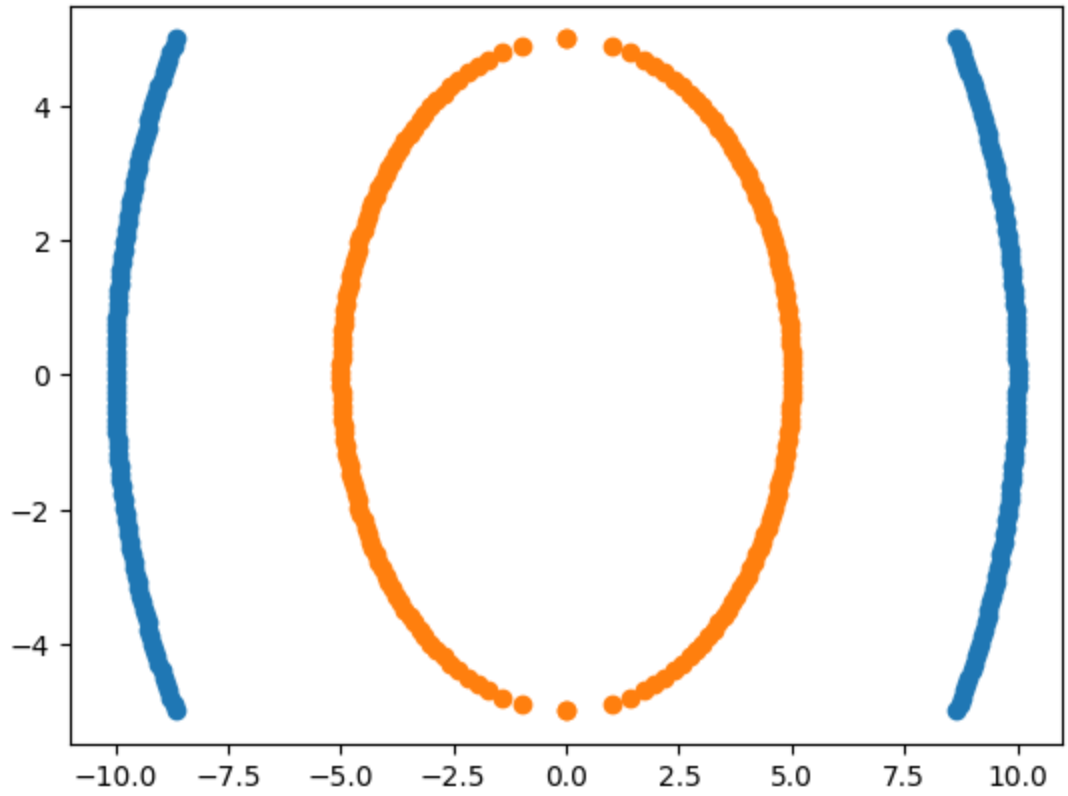
Support Vector Machine(Classification)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5.0, 5.0, 100)  ## here making circle equation,feature 1
y = np.sqrt(10**2 - x**2)   ## 10 is radius ,feature 2
y=np.hstack([y,-y])
x=np.hstack([x,-x])
```

In [2]:
```
x1 = np.linspace(-5.0, 5.0, 100) #feature 1
y1 = np.sqrt(5**2 - x1**2) # feature 2
y1=np.hstack([y1,-y1])
x1=np.hstack([x1,-x1])
```

In [3]:
```
plt.scatter(y,x)
plt.scatter(y1,x1)
```

Out[3]:
```
<matplotlib.collections.PathCollection at 0x2c5700a57f0>
```



In [4]:
```
import pandas as pd
df1 =pd.DataFrame(np.vstack([y,x]).T,columns=['X1','X2'])
df1['Y']=0  # creating 1 class
df2 =pd.DataFrame(np.vstack([y1,x1]).T,columns=['X1','X2'])
df2['Y']=1    # creating another class
df = df1.append(df2)
df.head(5)
```

```
C:\Users\jayes\AppData\Local\Temp\ipykernel_3560\3774610393.py:6: FutureWarning: The fra
me.append method is deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  df = df1.append(df2)
```

Out[4]:

	X1	X2	Y

0	8.660254	-5.00000	0
1	8.717792	-4.89899	0
2	8.773790	-4.79798	0
3	8.828277	-4.69697	0
4	8.881281	-4.59596	0

In [5]:
```
### Independent and Dependent features
X = df.iloc[:, :2]
y = df.Y
```

In [6]:
```
y
```

Out[6]:
```
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
```

In [7]:
```
## Split the dataset into train and test
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 [8]:
```
y_train
```

Out[8]:
```
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
```

In [9]:
```
from sklearn.svm import SVC
classifier=SVC(kernel="linear")
classifier.fit(X_train,y_train)
```

Out[9]:
```
SVC(kernel='linear')
```

In [10]:
```
from sklearn.metrics import accuracy_score
y_pred = classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[10]:
```
0.45
```

In [11]:
```
df.head()
```

Out[11]:

	X1	X2	Y
0	8.660254	-5.00000	0
1	8.717792	-4.89899	0
2	8.773790	-4.79798	0
3	8.828277	-4.69697	0
4	8.881281	-4.59596	0

In [12]:
```
from sklearn.svm import SVC
classifier=SVC(kernel="rbf")
classifier.fit(X_train,y_train)
```

Out[12]:
▾ SVC

SVC()

In [13]:
```
from sklearn.metrics import accuracy_score
y_pred = classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[13]:
```
1.0
```

In [14]:
```
df.head()
```

Out[14]:

	X1	X2	Y
0	8.660254	-5.00000	0
1	8.717792	-4.89899	0
2	8.773790	-4.79798	0
3	8.828277	-4.69697	0
4	8.881281	-4.59596	0

Polynomial Kernel

In [15]:
```
# We need to find components for the Polynomical Kernel
#X1,X2,X1_square,X2_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[15]:

	X1	X2	Y	X1_Square	X2_Square	X1*X2
0	8.660254	-5.00000	0	75.000000	25.000000	-43.301270
1	8.717792	-4.89899	0	75.999898	24.000102	-42.708375
2	8.773790	-4.79798	0	76.979390	23.020610	-42.096467
3	8.828277	-4.69697	0	77.938476	22.061524	-41.466150
4	8.881281	-4.59596	0	78.877155	21.122845	-40.818009

In [16]:
```
### Independent and Dependent features
X = df[['X1','X2','X1_Square','X2_Square','X1*X2']]
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
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.25,
                                                    random_state = 0)

X_train
```

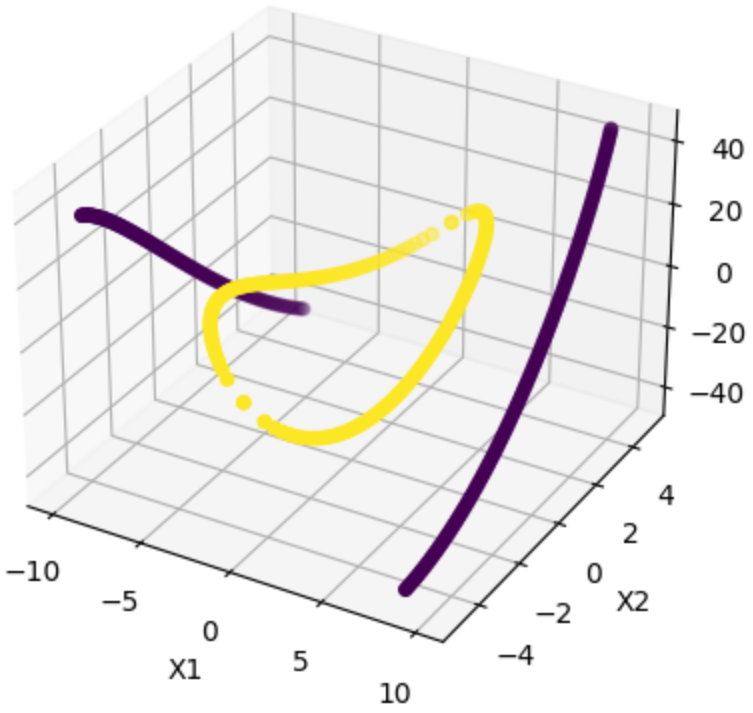
Out[18]:

	X1	X2	X1_Square	X2_Square	X1*X2
50	4.999745	0.050505	24.997449	0.002551	0.252512
63	9.906589	1.363636	98.140496	1.859504	13.508984
112	-3.263736	3.787879	10.651974	14.348026	-12.362637
159	-9.953852	-0.959596	99.079176	0.920824	9.551676
83	3.680983	3.383838	13.549638	11.450362	12.455852
...	...	...	...	...	...
123	-4.223140	2.676768	17.834915	7.165085	-11.304366
192	-9.031653	-4.292929	81.570758	18.429242	38.772248
117	-9.445795	3.282828	89.223038	10.776962	-31.008922
47	9.996811	-0.252525	99.936231	0.063769	-2.524447
172	-9.738311	-2.272727	94.834711	5.165289	22.132526

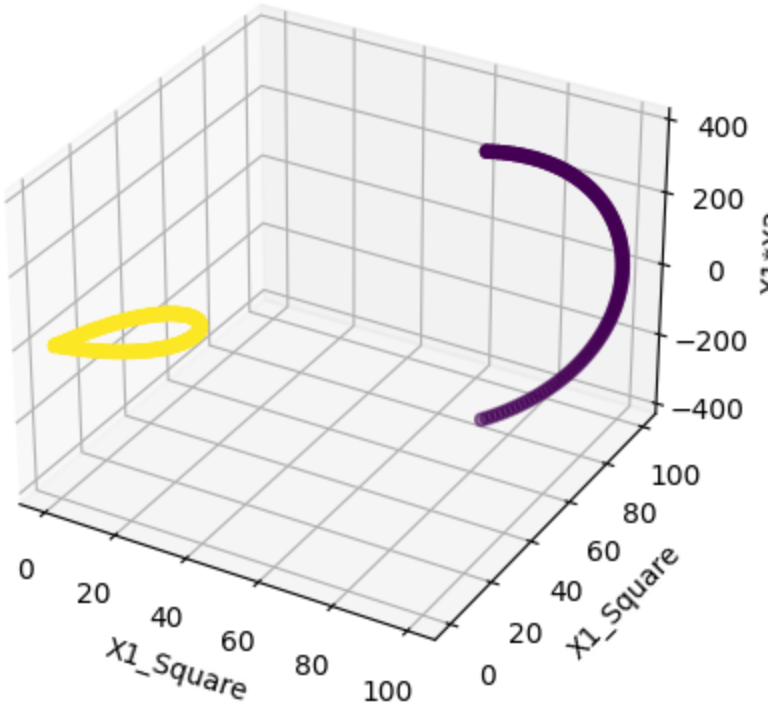
300 rows × 5 columns

In [27]:
```
import matplotlib.pyplot as plt
import pandas as pd



fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = df['X1']
y = df['X2']
z = df['X1'] * df['X2']
colors = df['Y']

ax.scatter3D(x, y, z, c=colors, cmap='viridis')

ax.set_xlabel('X1')
ax.set_ylabel('X2')
```

```
ax.set_zlabel('X1*X2')

plt.show()
```



In [26]:
```
import matplotlib.pyplot as plt
import pandas as pd



fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = df['X1_Square']
y = df['X1_Square']
z = df['X1_Square'] * df['X2']
colors = df['Y']

ax.scatter3D(x, y, z, c=colors, cmap='viridis')

ax.set_xlabel('X1_Square')
ax.set_ylabel('X1_Square')
ax.set_zlabel('X1*X2')

plt.show()
```



In [21]:
```
classifier = SVC(kernel="linear")
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[21]:
```
1.0
```

In [30]:
```
!jupyter nbconvert --to webpdf --allow-chromium-download SVM_KERNALS.ipynb
```

```
[NbConvertApp] Converting notebook SVM_KERNALS.ipynb to webpdf
[NbConvertApp] Building PDF
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 360487 bytes to SVM_KERNALS.pdf
```

In [ ]: