SVM Kernels Indepth Intuition And Practical Explanation

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

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

In [3]:
```
x.shape
```

Out[3]:
```
(100,)
```

In [4]:
```
y.shape
```

Out[4]:
```
(100,)
```

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

In [6]:
```
x.shape
```

Out[6]:
```
(200,)
```

In [7]:
```
y.shape
```

Out[7]:
```
(200,)
```

In [8]:
```
x1 = np.linspace(-5.0, 5.0, 100)
y1 = np.sqrt(5**2 - x1**2)
y1=np.hstack([y1,-y1])
x1=np.hstack([x1,-x1])
```

In [9]:
```
y1.shape
```

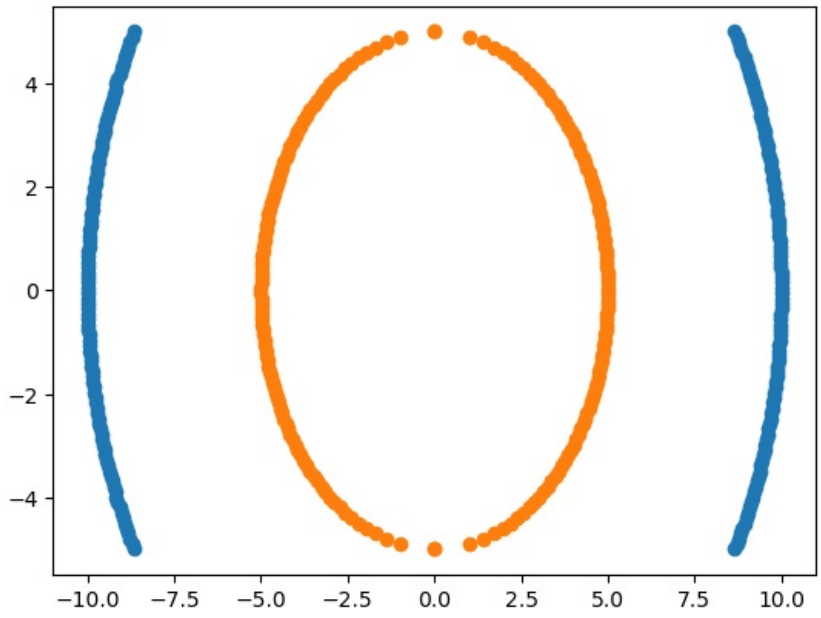
Out[9]:
```
(200,)
```

In [10]:
```
x1.shape
```

Out[10]:
```
(200,)
```

In [11]:
```
plt.scatter(y,x)
plt.scatter(y1,x1)
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x25935ba63d0>
```



In [ ]:
```
```

In [12]:
```
# Change a DataFrame (x,y)
df1 = pd.DataFrame(np.vstack([y,x]).T,columns = ['X1','X2'])
df1['Y'] = 0
```

```
# Change a DataFrame (x1,y1)
df2 = pd.DataFrame(np.vstack([y1,x1]).T,columns = ['X1','X2'])
df2['Y'] = 1
```

In [13]:
```
#append the two dataframe(df1,df2)
df = df1.append(df2)
df.head()
```

```
C:\Users\dines\AppData\Local\Temp\ipykernel_13024\1397825995.py:2: FutureWarning: The frame.append method is de
precated and will be removed from pandas in a future version. Use pandas.concat instead.
  df = df1.append(df2)
```

Out[13]:

	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 [14]:
```
x = df.iloc[:,:2]
y = df['Y']
```

In [15]:
```
x.head()
```

Out[15]:

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

In [16]:
```
y
```

Out[16]:
```
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 [17]:
```
#Split the dataset into train_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 [18]:
```
x_train.shape
```

Out[18]:
```
(300, 2)
```

In [19]:
```
y_train.shape
```

Out[19]:
```
(300,)
```

In [20]:
```
x_test.shape
```

Out[20]:
```
(100, 2)
```

In [21]:
```
x_test.shape
```

Out[21]:
```
(100, 2)
```

In [ ]:
```

```

Base_Model building

In [22]:
```
from sklearn.svm import SVC
```

In [23]:
```
cls = SVC(kernel = "rbf")
cls.fit(x_train,y_train)
```

Out[23]:
```
SVC()
```

In [24]:
```
#Evaluation Metrics
from sklearn.metrics import accuracy_score
y_pred = cls.predict(x_test)
accuracy_score(y_test,y_pred)
```

Out[24]:
```
1.0
```

In [ ]:

In [25]:
```
df.head()
```

Out[25]:

	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 [ ]:

Polynomial Kernel

In [26]:
```
# 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[26]:

	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 [27]:
```
df['X1'].shape
```

Out[27]:
```
(400,)
```

In [28]:
```
# Independent and Dependent features
x = df[['X1','X2','X1_Square','X2_Square','X1*X2']]
y = df['Y']
```

In [29]:
```
y
```

Out[29]:
```
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 [30]:
```
#train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25,random_state = 0)
```

In [31]:
```
x_train.shape
```

Out[31]:
```
(300, 5)
```

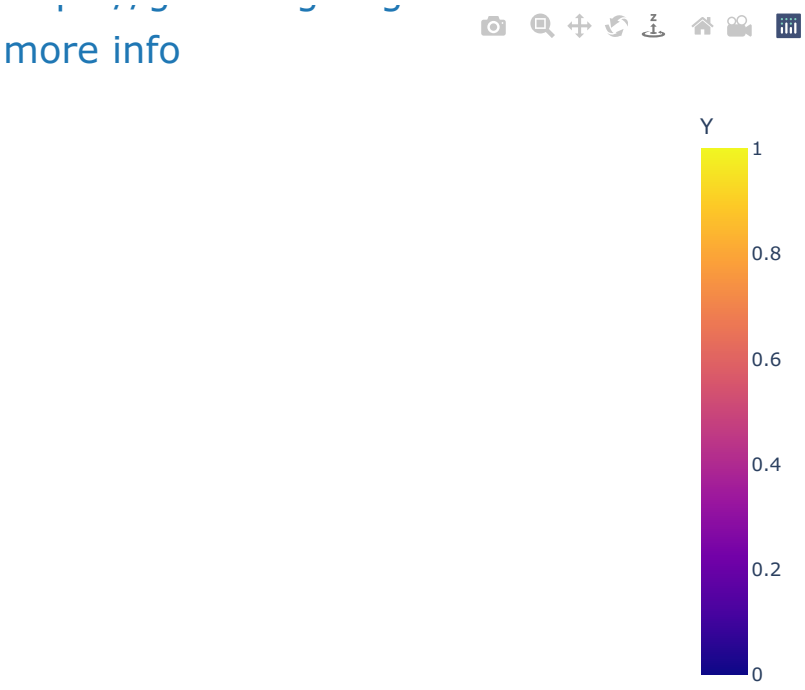
In [32]:
```
y_train.shape
```

Out[32]:
```
(300,)
```

In [33]:
```
#import poltly
import plotly.express as px

fig = px.scatter_3d(df, x="X1", y ="X2",z="X1*X2",color = 'Y')

fig.show()
```



In [34]:
```
fig = px.scatter_3d(df, x="X1_Square", y ="X1_Square",z="X1*X2",color = 'Y')

fig.show()
```

In [35]:
```
classifier = SVC(kernel="linear")
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
accuracy_score(y_test, y_pred)
```

Out[35]:
```
1.0
```

In [ ]:

In [ ]: