In [ ]:

```
#SVM(Support Vectors Machine)
>SVC
>SVR

#Objective:cluster similar data points in such a manner that a decision boundary(hyper p
#can be used to separate classes.


#Types of SVM
>Linear SVM
>Non Linear SVM
```

In [1]:

```
import pandas as pd
```

In [2]:

```
df=pd.read_csv("f:/dataset/classification/fruits.csv")
df
```

Out[2]:

	diameter	weight	FruitName
0	3.0	30	Banana
1	6.0	100	Apple
2	6.1	95	Apple
3	3.2	35	Banana
4	5.5	80	Apple
5	7.1	120	Banana
6	2.5	60	Banana
7	2.3	100	Banana
8	4.8	70	Apple
9	4.8	79	Apple
10	5.8	120	Apple
11	2.6	85	Banana
12	6.0	110	Apple
13	6.3	95	Apple
14	3.0	40	Banana
15	3.5	25	Banana
16	5.5	100	Apple
17	7.5	120	Apple
18	2.5	50	Banana
19	2.7	40	Banana
20	4.8	90	Apple
21	5.8	90	Apple

In [3]:

```
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [4]:

```
from sklearn.svm import SVC
```

In [5]:
```
model=SVC(kernel='linear')
model.fit(X,y)
```

Out[5]:
```
▾        SVC
SVC(kernel='linear')
```

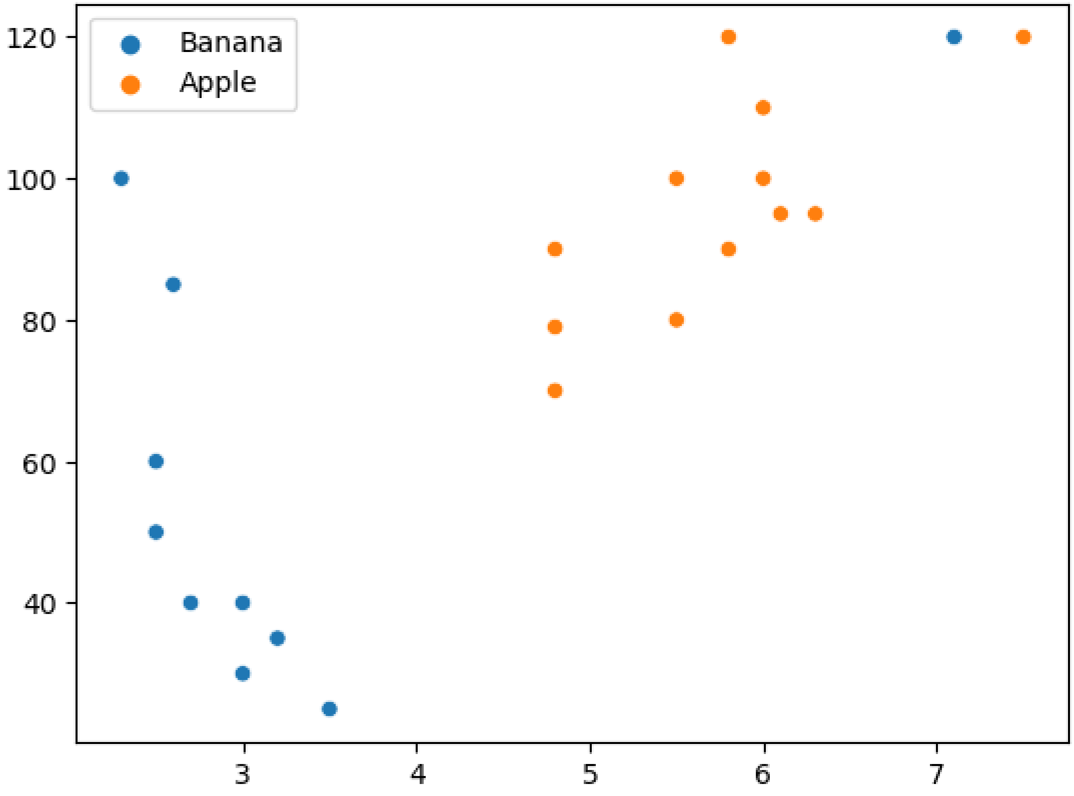
In [6]:
```
from sklearn.model_selection import cross_val_score
```

In [7]:
```
cross_val_score(model,X,y,cv=5).mean()
```

Out[7]:
```
0.96
```

In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sb
```

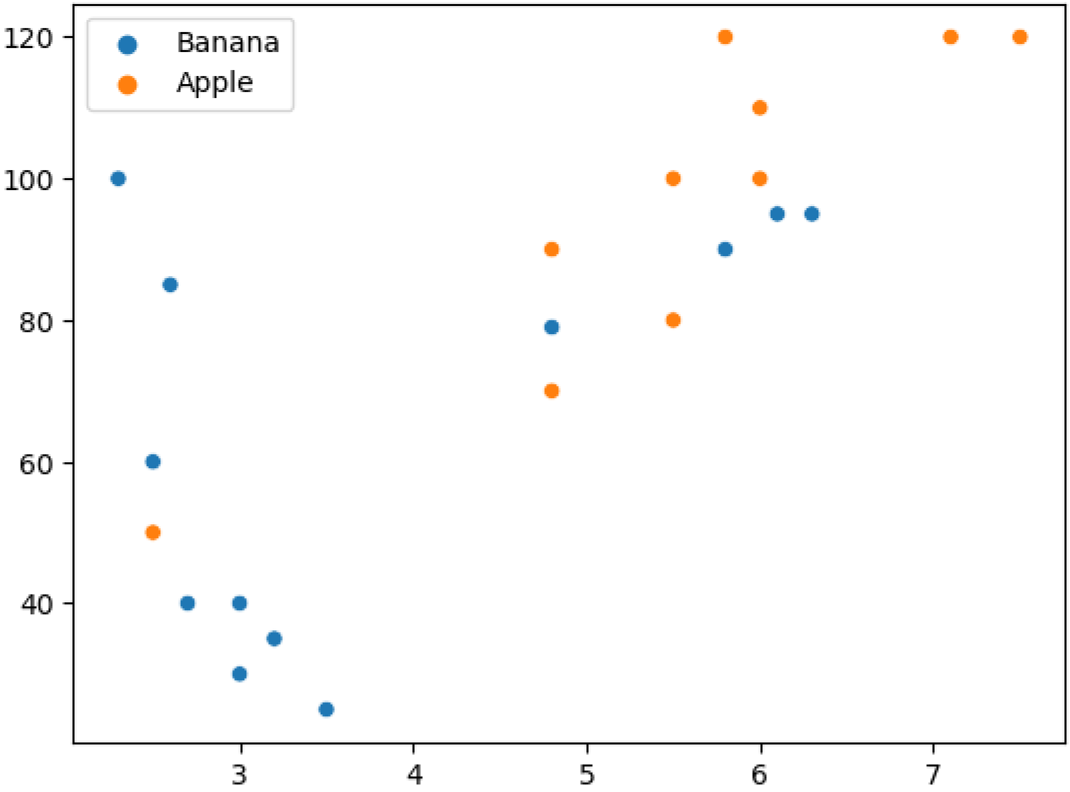
In [11]:
```
sb.scatterplot(x=X[:,0],y=X[:,1],hue=y)
plt.show()
```



In [12]:
```
df=pd.read_csv("f:/dataset/classification/fruits_svc.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
model=SVC(kernel='linear')
cross_val_score(model,X,y,cv=5).mean()
```

Out[12]:
```
0.6900000000000001
```

In [13]:
```
sb.scatterplot(x=X[:,0],y=X[:,1],hue=y)
plt.show()
```



In [15]:

```
df=pd.read_csv("f:/dataset/classification/fruits_svc.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
model=SVC(kernel='poly')
cross_val_score(model,X,y,cv=5).mean()
```

Out[15]:

```
0.6799999999999999
```

In [25]:

```
df=pd.read_csv("f:/dataset/classification/fruits_svc.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
model=SVC(kernel='poly',degree=7)
cross_val_score(model,X,y,cv=5).mean()
```

Out[25]:

```
0.73
```

In [34]:

```
df=pd.read_csv("f:/dataset/classification/fruits_svc.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
model=SVC(kernel='rbf',gamma=.5)
cross_val_score(model,X,y,cv=5).mean()
```

Out[34]:

```
0.73
```

In [45]:

```
df=pd.read_csv("f:/dataset/classification/fruits_svc.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
model=SVC(kernel='linear')
cross_val_score(model,X,y,cv=5).mean()
```

Out[45]:

```
0.6900000000000001
```

In [46]:

```
from sklearn.preprocessing import StandardScaler
```

In [49]:

```
df=pd.read_csv("f:/dataset/classification/fruits_svc.csv")
X=df.iloc[:,:-1].values
```

```
y=df.iloc[:,-1].values

sc=StandardScaler()
X=sc.fit_transform(X)

model=SVC(kernel='linear')
cross_val_score(model,X,y,cv=5).mean()
```

Out[49]:

```
0.6900000000000001
```

In [72]:

```
df=pd.read_csv("f:/dataset/classification/fruits_svc.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

sc=StandardScaler()
X=sc.fit_transform(X)

model=SVC(kernel='poly')
cross_val_score(model,X,y,cv=5).mean()
```

Out[72]:

```
0.73
```

In [71]:

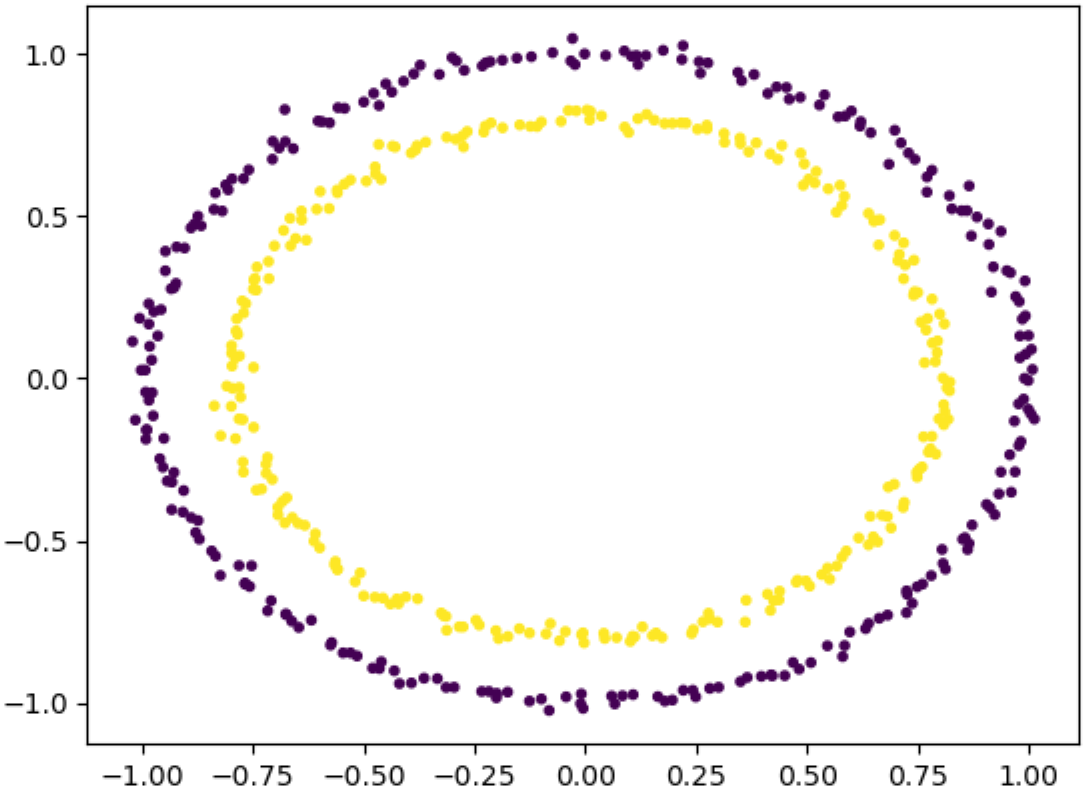
```
df=pd.read_csv("f:/dataset/classification/fruits_svc.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

sc=StandardScaler()
X=sc.fit_transform(X)

model=SVC(kernel='rbf')
cross_val_score(model,X,y,cv=5).mean()
```

Out[71]:

```
0.77
```

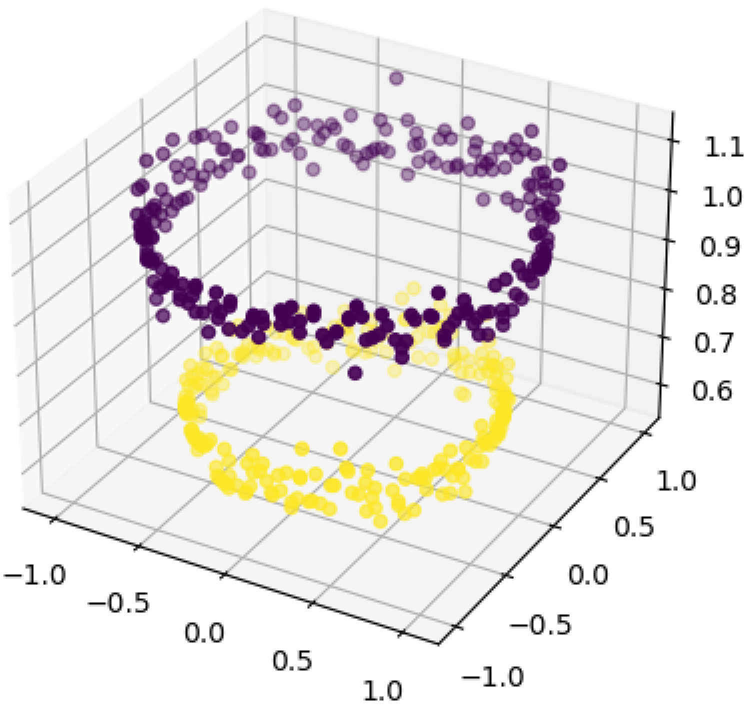
In [83]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from mpl_toolkits.mplot3d import Axes3D


X,y = make_circles(n_samples = 500,noise=.02)
plt.scatter(X[:, 0], X[:, 1],c=y,marker='.')
plt.show()
```



In [82]:

```
X1 = X[:, 0]
X2 = X[:, 1]
X3 = (X1**2 + X2**2)

fig = plt.figure()
axes=fig.add_subplot(projection = '3d')
axes.scatter(X1,X2,X3,c=y)
plt.show()
```



In [91]:

```
sample=[2.5,70]
sample=sc.transform([sample])
model=SVC(kernel='rbf',probability=True)
```

```
model.fit(X,y)
model.predict(sample)
```

Out[91]:

```
array([0], dtype=int64)
```

In [92]:

```
model.predict_proba(sample)
```

Out[92]:

```
array([[9.9999990e-01, 1.0000001e-07]])
```

In [ ]: