In [3]:
```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [5]:
```
df=pd.read_csv("f:/dataset/classification/fruits.csv")
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1)
```

without feature scaling

In [6]:
```
model=KNeighborsClassifier()
model.fit(X_train,y_train)
pred_train=model.predict(X_train)
pred_test=model.predict(X_test)
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))
```

```
Train Score: 0.75
Test Score: 1.0
```

with feature scaling

In [7]:
```
sc=MinMaxScaler(feature_range=(0,1))
X_train_new=sc.fit_transform(X_train)
X_test_new=sc.transform(X_test)

model=KNeighborsClassifier()
model.fit(X_train_new,y_train)
pred_train=model.predict(X_train_new)
pred_test=model.predict(X_test_new)
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))
```

```
Train Score: 0.9375
Test Score: 1.0
```

In [33]:
```
from sklearn.preprocessing import MaxAbsScaler
sc=MaxAbsScaler()
X_train_new=sc.fit_transform(X_train)
X_test_new=sc.transform(X_test)

model=KNeighborsClassifier()
model.fit(X_train_new,y_train)
pred_train=model.predict(X_train_new)
pred_test=model.predict(X_test_new)
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))
```

```
Train Score: 0.9375
Test Score: 1.0
```

In [34]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train_new=sc.fit_transform(X_train)
X_test_new=sc.transform(X_test)

model=KNeighborsClassifier()
model.fit(X_train_new,y_train)
```

```
pred_train=model.predict(X_train_new)
pred_test=model.predict(X_test_new)
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))
```

```
Train Score: 0.9375
Test Score: 1.0
```

In [11]:
```
d=float(input("enter dim:"))
w=float(input("enter wt:"))

#model.predict([[d,w]])
sample=sc.transform([[d,w]])
model.predict(sample)
```

Out[11]:
```
array(['Banana'], dtype=object)
```

In [12]:
```
df
```

Out[12]:

	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 [13]:
```
df.describe()
```

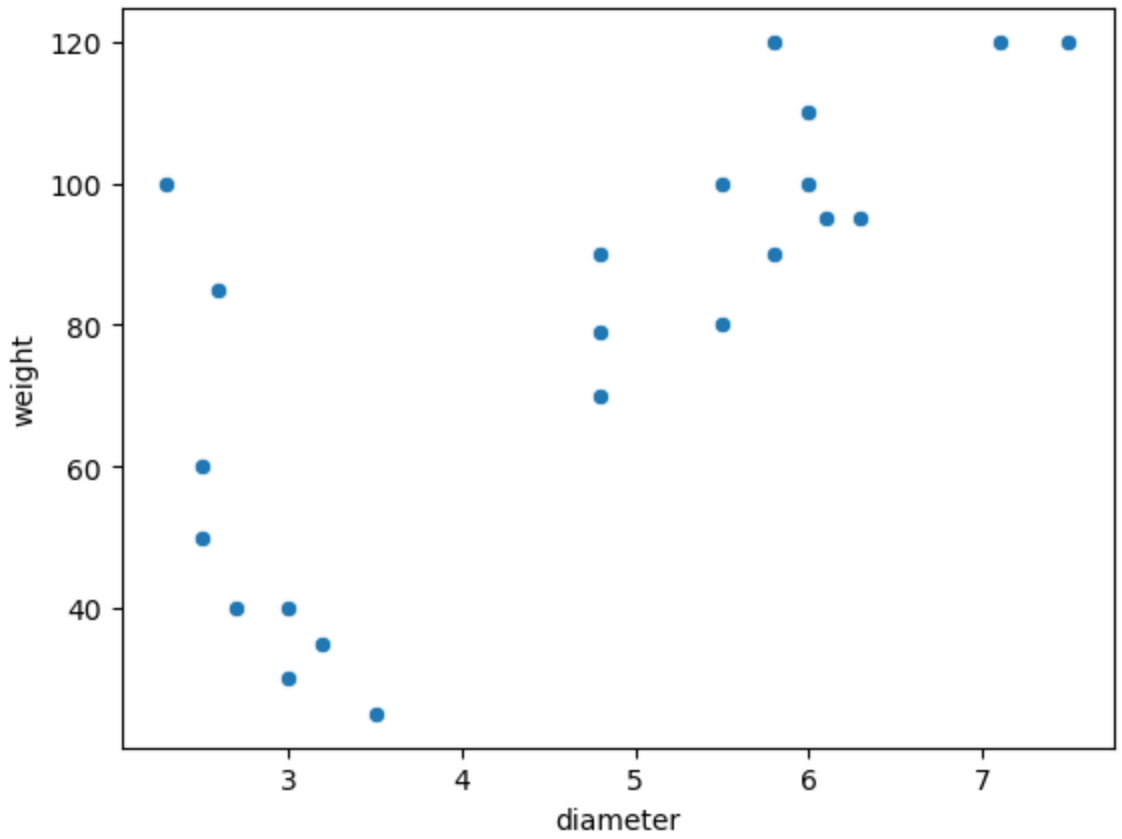
Out[13]:

	diameter	weight

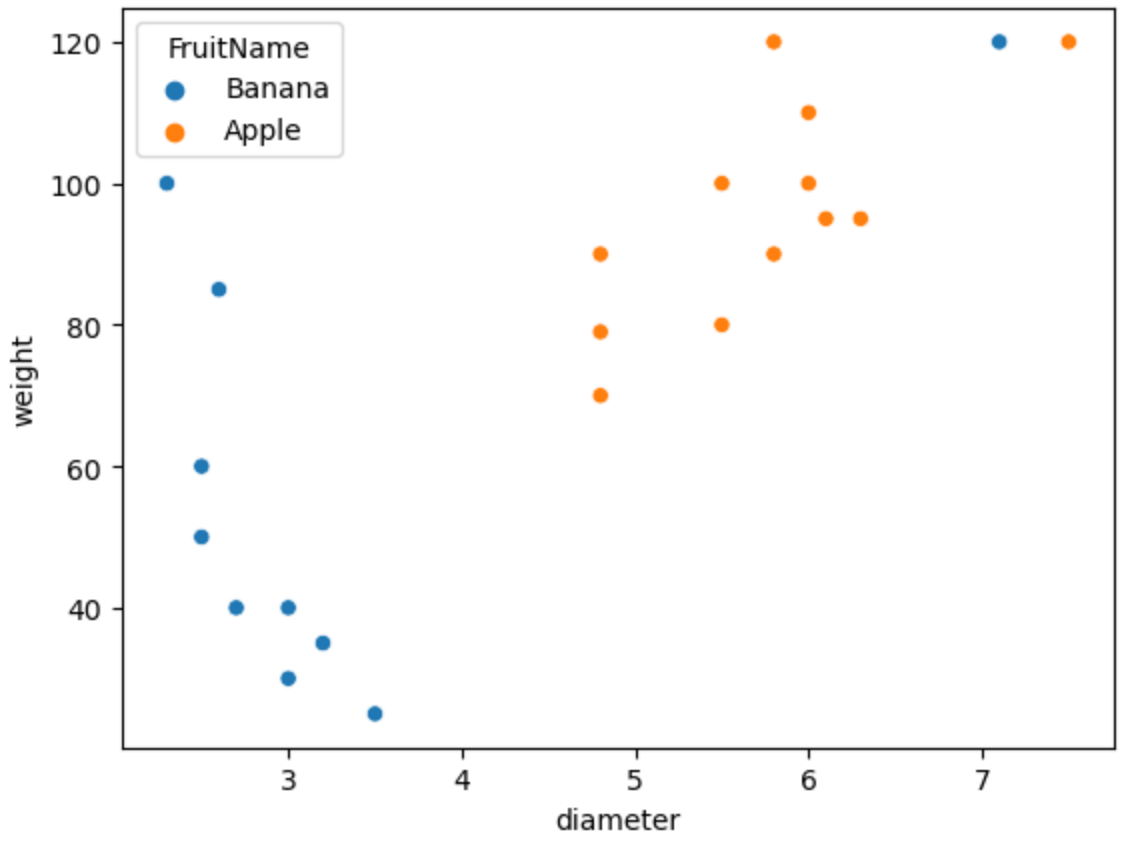
count	22.000000	22.000000
mean	4.604545	78.818182
std	1.666898	30.624665
min	2.300000	25.000000
25%	3.000000	52.500000
50%	4.800000	87.500000
75%	5.950000	100.000000
max	7.500000	120.000000

In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sb
```

In [15]:
```
sb.scatterplot(x=df.diameter,y=df.weight)
plt.show()
```



In [16]:
```
sb.scatterplot(x=df.diameter,y=df.weight,hue=df.FruitName)
plt.show()
```



Parameter v/s Hyper-parameter

- **parameters** are values that an object learn from data
- **hyper-parameters** are values that we pass explicitly to object.
- exp:
  - in scaler object,
    - minrange,maxrange are hyper-parameters
    - xmax,xmin are parameters

feature scaling with model hyper-parameter tuning

In [26]:
```
sc=MinMaxScaler(feature_range=(0,1))
X_train_new=sc.fit_transform(X_train)
X_test_new=sc.transform(X_test)
for i in range(1,10):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(X_train_new,y_train)
    pred_train=model.predict(X_train_new)
    pred_test=model.predict(X_test_new)
    print("---Neighbors:----",i)
    print("Train Score:",accuracy_score(y_train,pred_train))
    print("Test Score:",accuracy_score(y_test,pred_test))
```

```
---Neighbors:---- 1
Train Score: 1.0
Test Score: 1.0
---Neighbors:---- 2
Train Score: 0.9375
Test Score: 1.0
---Neighbors:---- 3
Train Score: 0.9375
Test Score: 1.0
---Neighbors:---- 4
```

```
Train Score: 0.9375
Test Score: 1.0
---Neighbors:---- 5
Train Score: 0.9375
Test Score: 1.0
---Neighbors:---- 6
Train Score: 0.8125
Test Score: 1.0
---Neighbors:---- 7
Train Score: 0.875
Test Score: 1.0
---Neighbors:---- 8
Train Score: 0.6875
Test Score: 0.6666666666666666
---Neighbors:---- 9
Train Score: 0.875
Test Score: 0.8333333333333334
```

In [32]:

```
sc=MinMaxScaler(feature_range=(0,1))
X_train_new=sc.fit_transform(X_train)
X_test_new=sc.transform(X_test)

model=KNeighborsClassifier(n_neighbors=5,metric="euclidean")
model.fit(X_train_new,y_train)
pred_train=model.predict(X_train_new)
pred_test=model.predict(X_test_new)
print("---Euclidean:----")
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))


model=KNeighborsClassifier(n_neighbors=5,metric="manhattan")
model.fit(X_train_new,y_train)
pred_train=model.predict(X_train_new)
pred_test=model.predict(X_test_new)
print("---Manhattan:----")
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))

model=KNeighborsClassifier(n_neighbors=5,metric="minkowski",p=1)
model.fit(X_train_new,y_train)
pred_train=model.predict(X_train_new)
pred_test=model.predict(X_test_new)
print("---Minkowski with p=1:----")
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))

model=KNeighborsClassifier(n_neighbors=5,metric="minkowski",p=2)
model.fit(X_train_new,y_train)
pred_train=model.predict(X_train_new)
pred_test=model.predict(X_test_new)
print("---Minkowski with p=2:----")
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))

model=KNeighborsClassifier(n_neighbors=5,metric="minkowski",p=3)
model.fit(X_train_new,y_train)
pred_train=model.predict(X_train_new)
pred_test=model.predict(X_test_new)
print("---Minkowski with p=3:----")
print("Train Score:",accuracy_score(y_train,pred_train))
print("Test Score:",accuracy_score(y_test,pred_test))
```

```
---Euclidean:----
Train Score: 0.9375
Test Score: 1.0
```

```
---Manhattan:----
Train Score: 0.9375
Test Score: 1.0
---Minkowski with p=1:----
Train Score: 0.9375
Test Score: 1.0
---Minkowski with p=2:----
Train Score: 0.9375
Test Score: 1.0
---Minkowski with p=3:----
Train Score: 0.9375
Test Score: 1.0
```

In [ ]: