HARI PRASATH S

226229110

Importing packages

In [12]:

```
import pandas as pd
```

In [13]:

```
import seaborn as sns
```

In [14]:

```
import matplotlib.pyplot as plt
```

In [15]:

```
from sklearn.neighbors import KNeighborsClassifier
```

STEP1: Importing csv file

STEP2: Printing using head()

In [16]:

```
pi = pd.read_csv("pizza.csv")
pi.head()
```

Out[16]:

	Age	Weight	Likepizza
0	50	65	0
1	20	55	1
2	15	40	1
3	70	65	0
4	30	70	1

Printing shape

In [17]:

```
pi.shape
```

Out[17]:

```
(6, 3)
```

Printing columns

In [18]:

```
pi.columns
```

Out[18]:

```
Index(['Age', 'Weight', 'Likepizza'], dtype='object')
```

Printing info

In [28]:

```
pi.info
```

Out[28]:

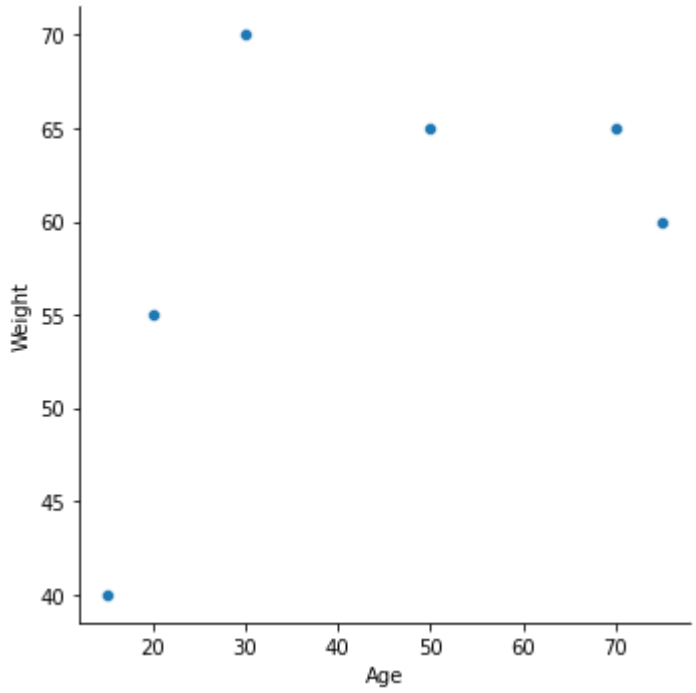
```
<bound method DataFrame.info of    Age  Weight  Likepizza
0   50      65          0
1   20      55          1
2   15      40          1
3   70      65          0
4   30      70          1
5   75      60          0>
```

In [ ]:

```
#step3:
```

In [23]:

```
sns.relplot(x='Age',y='Weight',data=pi,kind='scatter');
```



In [ ]:

```
#STEP 4:
#prepare x matric and y vector
```

In [24]:

```
Fix = ["Age","Weight"]
x = pi[Fix]
```

In [25]:

```
#STEP 5:
#printing x
x
```

Out[25]:

	Age	Weight
0	50	65
1	20	55
2	15	40
3	70	65
4	30	70
5	75	60

In [26]:

```
y = pi.Likepizza
```

In [27]:

```
#printing y
y
```

Out[27]:

```
0    0
1    1
2    1
3    0
4    1
5    0
Name: Likepizza, dtype: int64
```

In [12]:

```
#printing type of x
type(x)
```

Out[12]:

```
pandas.core.frame.DataFrame
```

In [13]:

```
#printing type of y
type(y)
```

Out[13]:

```
pandas.core.series.Series
```

In [21]:

```
piz_eat = KNeighborsClassifier(n_neighbors=2)
```

In [ ]:

```
#STEP 6:
#MODEL BUILDING
```

In [22]:

```
piz_eat.fit(x,y)
```

Out[22]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=2, p=2,
           weights='uniform')
```

In [ ]:

```
#STEP 7:
#predicting if person will like pizza or not
```

In [23]:

```
piz_eat.predict(x)
```

Out[23]:

```
array([0, 1, 1, 0, 1, 0], dtype=int64)
```

In [ ]:

```
#will a person who is 25 years with weight 50 kgs like pizza or not?
```

In [24]:

```
pre = [[25,50]]
print(piz_eat.predict(pre))
```

```
[1]
```

In [ ]:

```
#will a person who is 60 years with weight 60 kgs like pizza or not?
```

In [25]:

```
pre =[[60,60]]
print(piz_eat.predict(pre))
```

```
[0]
```

In [ ]:

```
#STEP 8:
#changing n_neighbors to 3
```

In [31]:

```
piz_eat1 = KNeighborsClassifier(n_neighbors=3)
piz_eat1.fit(x,y)
```

Out[31]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')
```

In [ ]:

```
#will a person who is 25 years with weight 50 kgs like pizza or not?
```

In [32]:

```
pred = [[25,50]]
print(piz_eat1.predict(pred))
```

```
[1]
```

In [ ]:

```
#will a person who is 25 years with weight 50 kgs like pizza or not?
```

In [33]:

```
pred = [[60,60]]
print(piz_eat1.predict(pred))
```

```
[0]
```

In [ ]:

```
#STEP 9:
#Predict on entire dataset
```

In [34]:

```
y_pred = piz_eat.predict
(x)
```

Out[34]:

	age	weight
0	50	65
1	20	55
2	15	40
3	70	65
4	30	70
5	75	60

In [ ]:

```
#STEP 10:
#Accuracy funtion:
```

In [35]:

```
def accuracy(actual, pred):
    return sum(actual == pred) / float(actual.shape[0])
#STEP 11:
#calling accuracy funtion
accuracy(y,y_pred)
```

Out[35]:

```
0.0
```

In [ ]:

```
#STEP 12:
#importing csv file
```

In [36]:

```
pi1 = pd.read_csv("pizza_test.csv")
```

In [ ]:

```
#printing using head()
```

In [37]:

```
pi1.head()
```

Out[37]:

	age	weight	pizza_like
0	48	58	1
1	35	45	1
2	15	40	0
3	55	65	0

In [38]:

```
pi1.shape
```

Out[38]:

```
<bound method DataFrame.info of    age  weight  pizza_like
0   48      58           1
1   35      45           1
2   15      40           0
3   55      65           0>
```

In [39]:

```
pi1.columns
```

Out[39]:

```
Index(['age', 'weight', 'pizza_like'], dtype='object')
```

In [40]:

```
pi1.info
```

Out[40]:

```
<bound method DataFrame.info of    age  weight  pizza_like
0   48      58           1
1   35      45           1
2   15      40           0
3   55      65           0>
```

In [41]:

```
Fix = ["age","weight"]
x = pi1[Fix]
```

In [42]:

```
x
```

Out[42]:

	age	weight
0	48	58
1	35	45
2	15	40
3	55	65

In [45]:

```
y = pi1.pizza_like
y
```

Out[45]:

```
0    1
1    1
2    0
3    0
Name: pizza_like, dtype: int64
```

In [46]:

```
pizza_eat = KNeighborsClassifier(n_neighbors=2)
pizza_eat.fit(x,y)
```

Out[46]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=2, p=2,
           weights='uniform')
```

In [47]:

```
pizza_eat.predict(x)
```

Out[47]:

```
array([0, 1, 0, 0], dtype=int64)
```

In [48]:

```
def accuracy(actual, pred):
    return sum(actual == pred) / (float(actual.shape[0]))


y_pred = pizza_eat.predict(x)
```

In [49]:

```
accuracy(y,y_pred)
```

Out[49]:

```
0.75
```

In [ ]:

```
#STEP 13:
#finding best value for k
```

In [50]:

```
scores = []

for k in range(1,4):
    best = KNeighborsClassifier(n_neighbors=k)
    best.fit(x,y)
    y_pred = best.predict(x)
    acc = accuracy(y,y_pred)
    scores.append((k, acc))


scores
```

Out[50]:

```
[(1, 1.0), (2, 0.75), (3, 0.5)]
```

In [ ]:

```
#STEP 14:
#importing accuracy score
#calling accuracy_score()
```

In [51]:

```
from sklearn.metrics import accuracy_score
```

In [52]:

```
accuracy_score(y,y_pred)
```

Out[52]:

```
0.5
```

In [ ]:

```

```