PML-LAB-2_maha18

March 15, 2021

0.0.1 Lab2. Pizza Liking Prediction using kNN

0.0.2 Step2. [Import dataset]. Using Pandas, import "pizza.csv" file and print properties such as head(), shape, columns and info.

[5]:
```
import pandas as pd
import csv
```

[6]:
```
piz=pd.read_csv("pizza.csv")
```

[7]:
```
piz.head()
```

[7]:

	age	weight	likepizza
0	50	65	0
1	20	55	1
2	15	40	1
3	70	65	0
4	30	70	1

[8]:
```
piz.tail()
```

[8]:

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

[9]:
```
piz.shape
```

[9]: (6, 3)

[10]:
```
df = pd.read_csv("pizza.csv")
```

[11]:
```
df
```

[11]:

	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
```

[12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         6 non-null      int64
 1   weight      6 non-null      int64
 2   likepizza   6 non-null      int64
dtypes: int64(3)
memory usage: 272.0 bytes
```

[13]:
```
res = df.columns
```

[14]:
```
print(res)
```

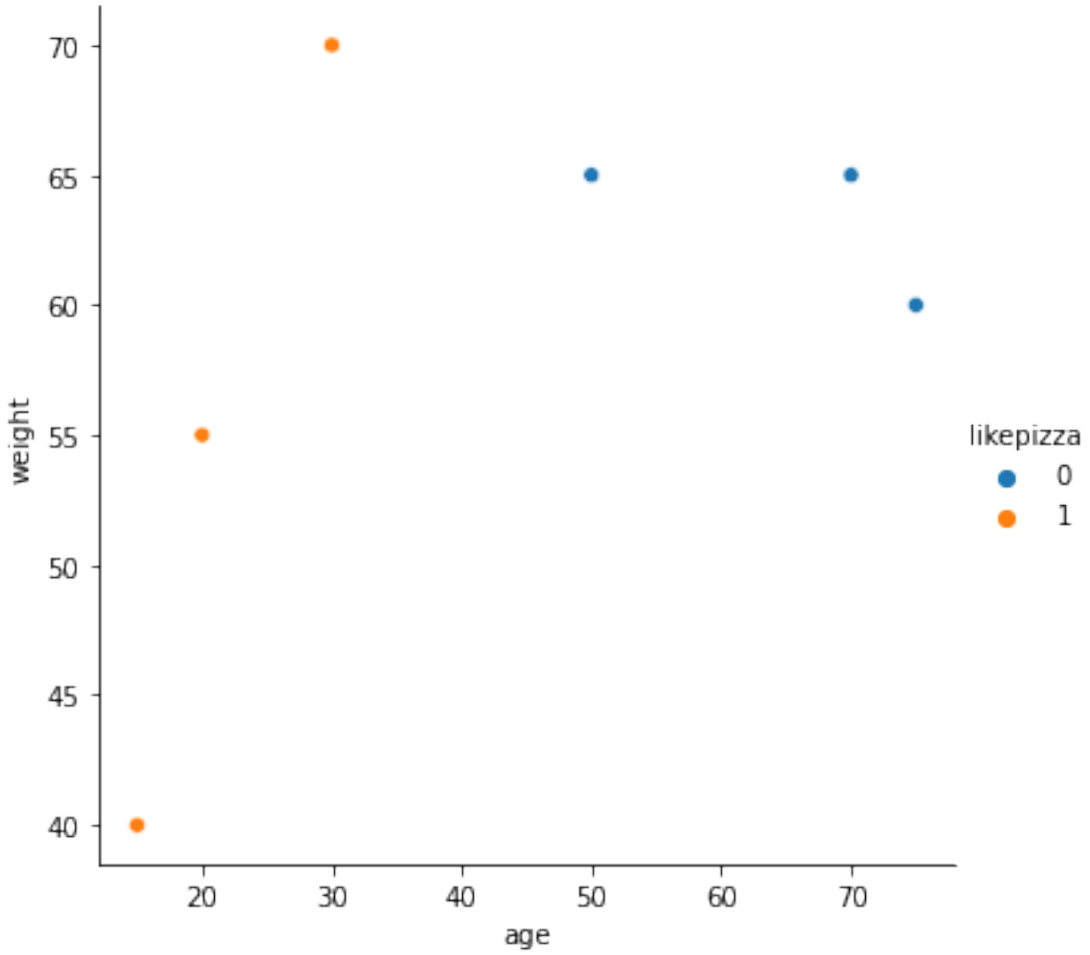
```
Index(['age', 'weight', 'likepizza'], dtype='object')
```

0.0.3 Step3. [Visualize Relationships]. Plot relplot between "age" and "weight", with hue as "likePizza"

[15]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

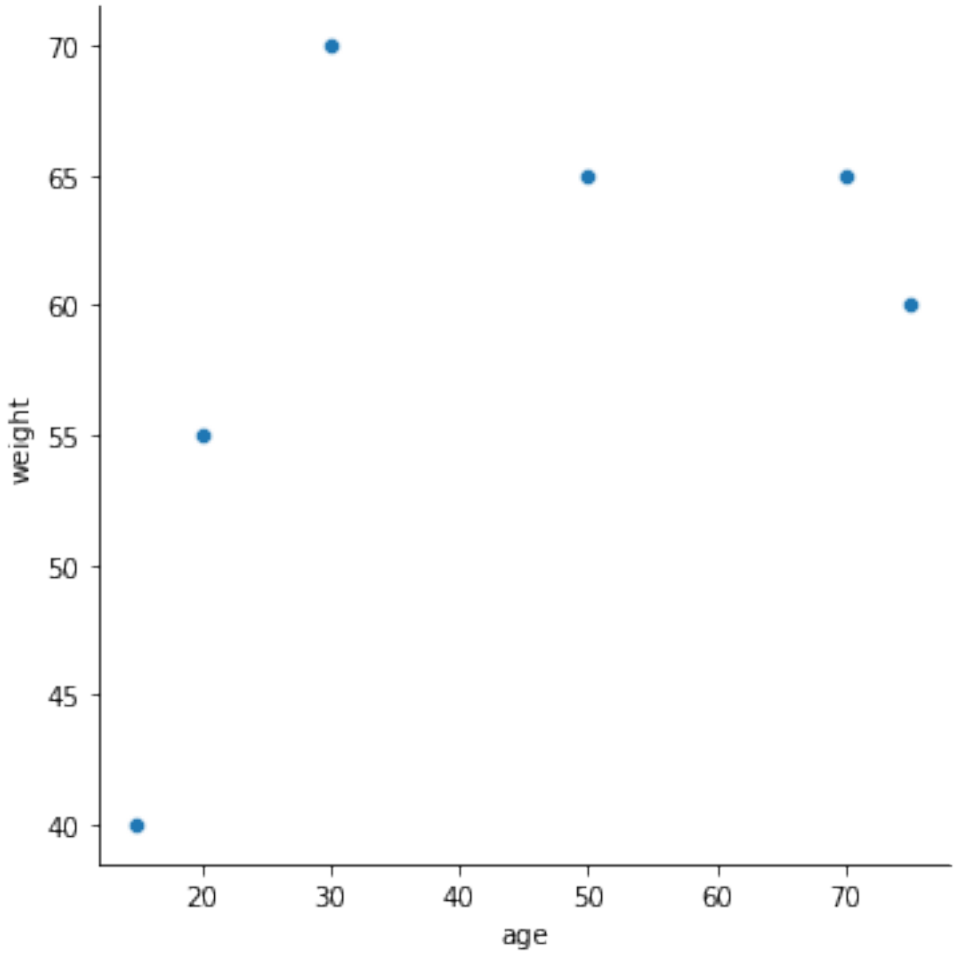
[16]:
```
sns.relplot(x="age", y="weight", hue="likepizza",data=piz)
```

[16]:
```
<seaborn.axisgrid.FacetGrid at 0x20695e6b7c0>
```



[17]:
```
sns.relplot(x='age', y='weight',data=piz, kind='scatter')
```

[17]: <seaborn.axisgrid.FacetGrid at 0x20695e38340>



[20]:
```
y=piz.likepizza
```

0.0.4 Step4. [Prepare X matrix and y vector]. Extract "age" and "weight" columns and store into new dataframe X. Similarly, extract "likePizza" column and store into y.

[21]:
```
dat = ['age','weight']
X=piz[dat]
```

[22]:
```
X
```

[22]:

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

```
5   75      60
```

[23]:
```
y
```

[23]:
```
0    0
1    1
2    1
3    0
4    1
5    0
Name: likepizza, dtype: int64
```

0.0.5 Step5. [Examine X and y]. Print X, y, type of X and type of y.

[24]:
```
X.dtypes
```

[24]:
```
age       int64
weight    int64
dtype: object
```

[25]:
```
y.dtype
```

[25]:
```
dtype('int64')
```

0.0.6 Step6. [Model building]. Create KNeighborsClassifier(n_neighbors=2) from sklearn and perform fit onX and y.

[26]:
```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\programdata\anaconda3\lib\site-
packages (0.0)
Requirement already satisfied: scikit-learn in
c:\programdata\anaconda3\lib\site-packages (from sklearn) (0.23.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->sklearn) (2.1.0)
Requirement already satisfied: scipy>=0.19.1 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.5.2)
Requirement already satisfied: joblib>=0.11 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->sklearn) (0.17.0)
Requirement already satisfied: numpy>=1.13.3 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.19.2)
```

[27]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[28]:
```
pizza_piz = KNeighborsClassifier(n_neighbors=2)
pizza_piz.fit(X, y)
```

[28]:
```
KNeighborsClassifier(n_neighbors=2)
```

0.0.7 Step7. [Model testing]. Using your KNN model, predict if a person will like Pizza or not.

[29]:
```
print(pizza_piz.predict(X))
```

```
[0 1 1 0 1 0]
```

0.0.8 Will a person who is 25 years with weight 50 kgs like Pizza or not? – The answer should be 1 (ie., YES)

[30]:
```
new=[25,50]
pizza_piz.predict([new])
```

[30]:
```
array([1], dtype=int64)
```

0.0.9 Will a person who is 60 years with weight 60 kgs like Pizza or not? – The answer should be 0 (ie., NO)

[31]:
```
new=[60,60]
pizza_piz.predict([new])
```

[31]:
```
array([0], dtype=int64)
```

0.0.10 Step.8 [Change n_neighbors = 3]. Now, create new model, perform fit and predict steps. Check results for the above 2 queries. Are they same?

[32]:
```
pizza_piz = KNeighborsClassifier(n_neighbors=3)
pizza_piz.fit(X, y)
```

[32]:
```
KNeighborsClassifier(n_neighbors=3)
```

[33]:
```
print(pizza_piz.predict(X))
```

```
[0 1 1 0 1 0]
```

[34]:
```
new=[25,50]
pizza_piz.predict([new])
```

[34]:
```
array([1], dtype=int64)
```

[35]:
```
new=[60,60]
pizza_piz.predict([new])
```

[35]:
```
array([0], dtype=int64)
```

0.0.11 Step9. [Predict on entire dataset]. Now, perform prediction on entire X matrix and store result as y_pred.

[36]:
```
y_pred=pizza_piz.predict(X)
```

[37]:
```
y_pred
```

[37]:
```
array([0, 1, 1, 0, 1, 0], dtype=int64)
```

0.0.12 Step10. [Accuracy function]. Create a function accuracy() and returns accuracy.

[74]:
```
def accuracy(actual,pred):
    return sum(actual == pred) / float(actual.shape[0])
```

0.0.13 Step11. [Find accuracy]. Call accuracy() with y and y_pred as parameters and print accuracy score. Are you getting score as 1.0 ?

[76]:
```
accuracy(y,y_pred)
```

[76]:
```
0.5
```

0.0.14 Step12. [Prediction on Test Set]

Using Pandas, import "pizza_test.csv" file and print properties such as head(), shape, columns and info.Using KNN model with n_neighbors=2, that you created previously, perform prediction on X values frompizza_test dataframe. Call accuracy function and print accuracy score. Are you getting a score of 0.5?That is, our model has predicted 2 samples correctly and two wrongly, out of 4 samples in the test set.

[38]:
```
pizz=pd.read_csv("pizza_test.csv")
```

[39]:
```
pizz.head()
```

[39]:

	age	weight	likepizza
0	48	68	1
1	35	45	1
2	15	40	0
3	55	65	0

[40]:
```
pizz.tail()
```

[40]:

	age	weight	likepizza
0	48	68	1
1	35	45	1
2	15	40	0
3	55	65	0

[41]:
```
pizz.shape
```

[41]:
```
(4, 3)
```

[42]:
```
df = pd.read_csv("pizza_test.csv")
```

[43]:
```
df
```

[43]:

	age	weight	likepizza
0	48	68	1
1	35	45	1
2	15	40	0
3	55	65	0

[44]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         4 non-null      int64
 1   weight      4 non-null      int64
 2   likepizza   4 non-null      int64
dtypes: int64(3)
memory usage: 224.0 bytes
```

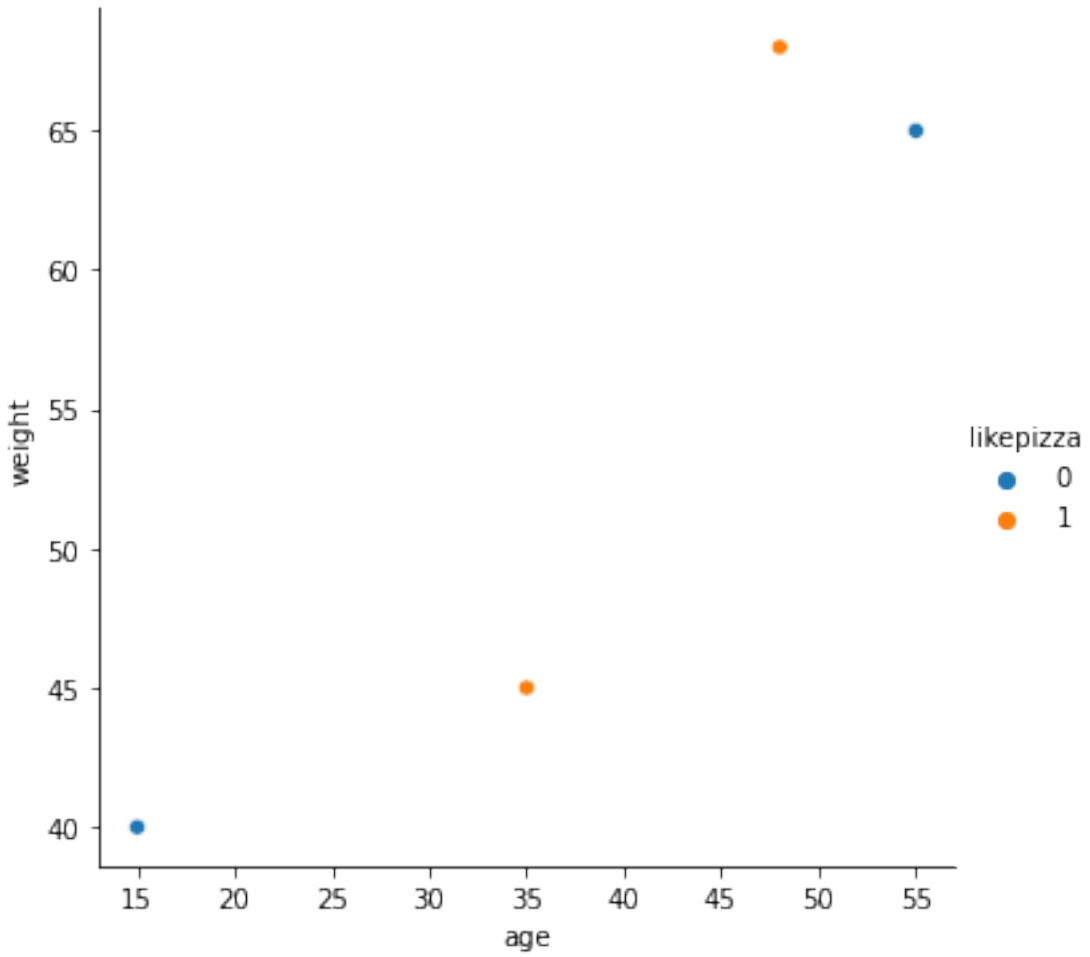
[45]:
```
res = df.columns
```

[46]:
```
print(res)
```

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

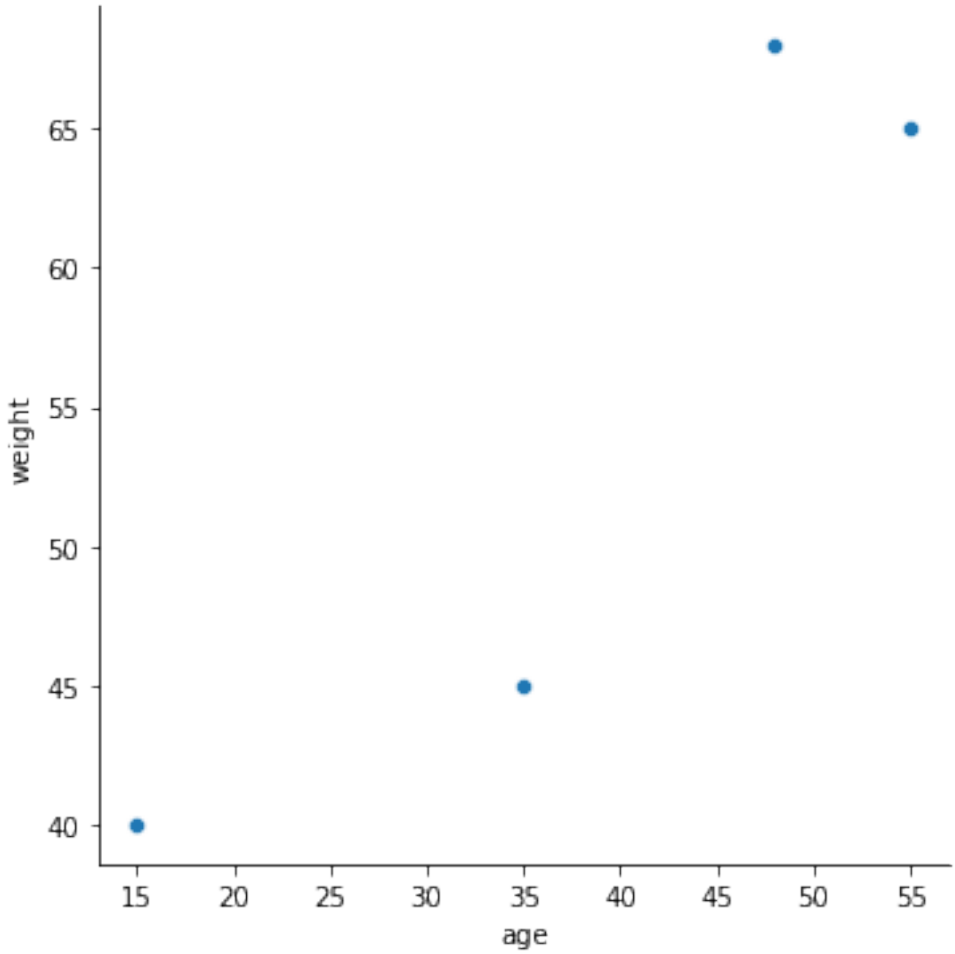
[47]:
```
sns.relplot(x="age", y="weight", hue="likepizza",data=pizz)
```

[47]:
```
<seaborn.axisgrid.FacetGrid at 0x20697feb880>
```



[48]:
```
sns.relplot(x='age', y='weight',data=pizz, kind='scatter')
```

[48]: <seaborn.axisgrid.FacetGrid at 0x206980419a0>



[49]:
```
rk = pizz.dropna(axis=0)
```

[50]:
```
rk
```

[50]:

	age	weight	likepizza
0	48	68	1
1	35	45	1
2	15	40	0
3	55	65	0

[51]:
```
y=pizz.likepizza
```

[52]:
```
datt = ['age','weight']
X=pizz[datt]
```

[53]:
```
X
```

[53]:
```
   age  weight
0   48      68
1   35      45
2   15      40
3   55      65
```

[54]:
```
y
```

[54]:
```
0    1
1    1
2    0
3    0
Name: likepizza, dtype: int64
```

[55]:
```
X.dtypes
```

[55]:
```
age       int64
weight    int64
dtype: object
```

[56]:
```
y.dtypes
```

[56]:
```
dtype('int64')
```

[57]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[81]:
```
piz1 = KNeighborsClassifier(n_neighbors=2)
piz1.fit(X, y)
```

[81]:
```
KNeighborsClassifier(n_neighbors=2)
```

[83]:
```
(piz1.predict(X))
```

[83]:
```
array([0, 0, 0, 0], dtype=int64)
```

[84]:
```
def accuracy(actual,pred):
    return sum(actual == pred) / float(actual.shape[0])
```

[85]:
```
y_pred=piz1.predict(X)
```

[86]:
```
accuracy(y,y_pred)
```

[86]:
```
0.5
```

0.0.15 step13. [Find best value for k]. If you want to improve the accuracy of your model, then you should use the best value k for the nearest neighbors.

[77]:
```
score = []
for k in range(1,4):
    best = KNeighborsClassifier(n_neighbors=k)
    best.fit(X, y)
    best.predict(X)
    y_predt = best.predict(X)
    acc=accuracy(y,y_predt)
    score.append((k,acc))
```

[78]:
```
score
```

[78]:
```
[(1, 1.0), (2, 0.5), (3, 0.5)]
```

0.0.16 Step14. [accuracy_score function]. Call accuracy_score() function with y_test and y_pred values. You can import as "from sklearn.metrics import accuracy_score".

[88]:
```
from sklearn.metrics import accuracy_score
```

[89]:
```
accuracy_score(y,y_predt)
```

[89]:
```
0.5
```