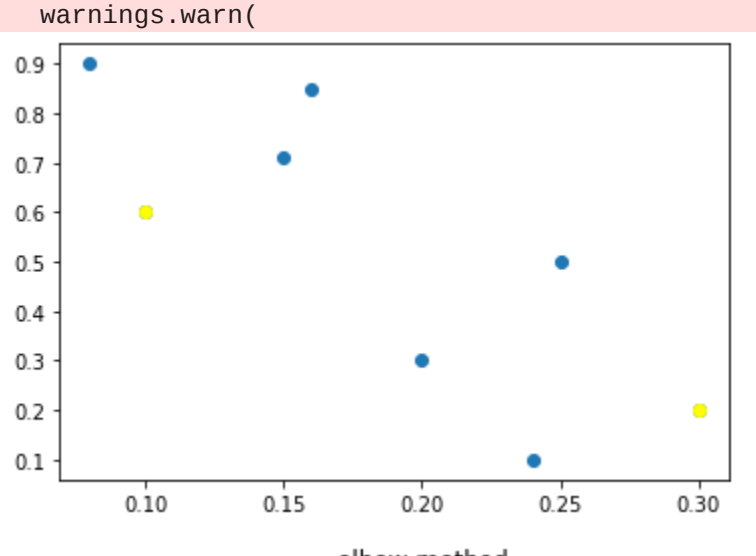
In [14]:

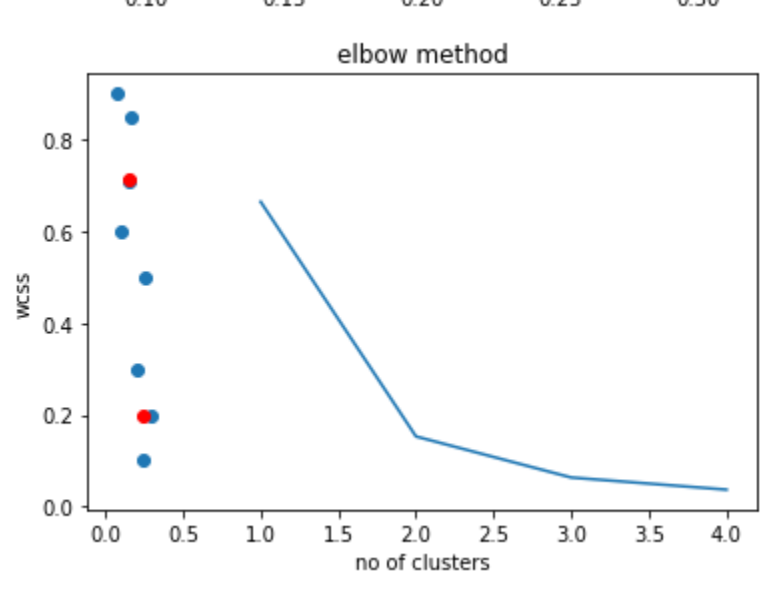
```python
import numpy as np
import pandas as pandas
import math
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
from sklearn.datasets import load_boston
```

In [1]:

```python
#Assi 1
#Kmeans  We have given a collection of 8 points.
#P1=[0.1,0.6] P2=[0.15,0.71] P3=[0.08,0.9] P4=[0.16, 0.85] P5=[0.2,0.3] P6=[0.25,0.5] P7=[0.24,0.1] P8=[0.3,0.2].
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
data=[[0.1,0.6],[0.08,0.9],[0.15,0.71],[0.16,0.85],[0.2,0.3],[0.25,0.5],[0.24,0.1],[0.3,0.2]]
df=np.array(data)
centriod=[[0.1,0.6],[0.3,0.2]]
centriod=np.array(centriod)

model=KMeans(n_clusters=2,init=centriod,n_init=1,random_state=0)
model.fit(data)

plt.figure()
plt.scatter(df[:,0],df[:,1])
plt.scatter(centriod[:,0],centriod[:,1],color='yellow',marker='o')

model.labels_
print("p6 belongs to",model.predict([[0.25,0.5]]))
print("population of m2",sum(model.labels_==1))
print("centriod m1::",model.cluster_centers_[0])
print("centriod m2::",model.cluster_centers_[1])

plt.figure()
plt.scatter(df[:,0],df[:,1])
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],color='red',marker='o')

wcss=[]
for i in range(1,5):
    model=KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    model.fit(data)
    wcss.append(model.inertia_)

plt.plot(range(1,5),wcss)
plt.title("elbow method")
plt.xlabel("no of clusters")
plt.ylabel("wcss")
plt.show()
```

```
p6 belongs to [0]
population of m2 3
centriod m1:: [0.148 0.712]
centriod m2:: [0.24666667 0.2       ]
C:\Users\user\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than av
ailable threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

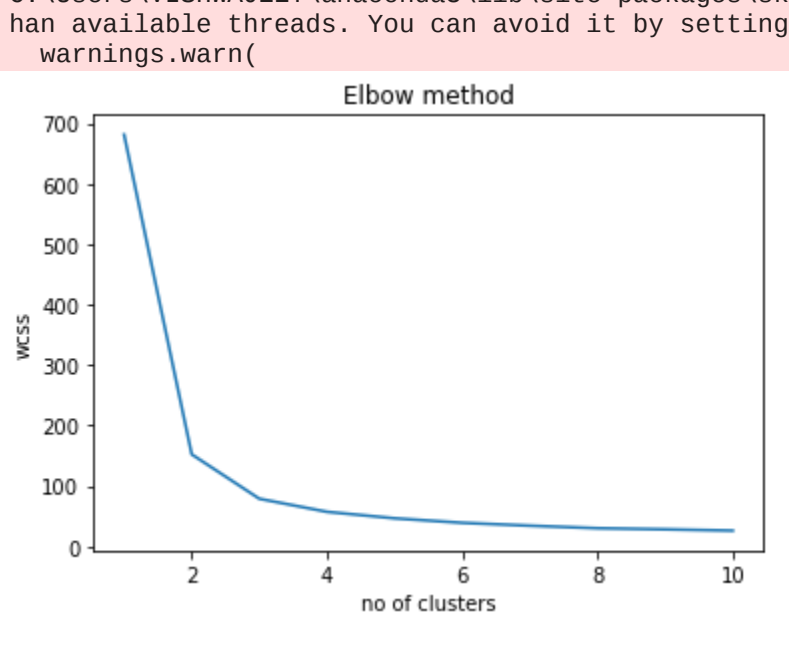




In [45]:

```python
#ass 2
#Apply K-Means Clustering technique of machine learning to analyze the Iris dataset.
#Use Elbow method to find best value of K.

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans

iris=load_iris()
wcss=[]
for i in range(1,11):
    model=KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    model.fit(iris.data)
    wcss.append(model.inertia_)

plt.plot(range(1,11),wcss)
plt.title("Elbow method")
plt.xlabel("no of clusters")
plt.ylabel("wcss")
plt.show()
```

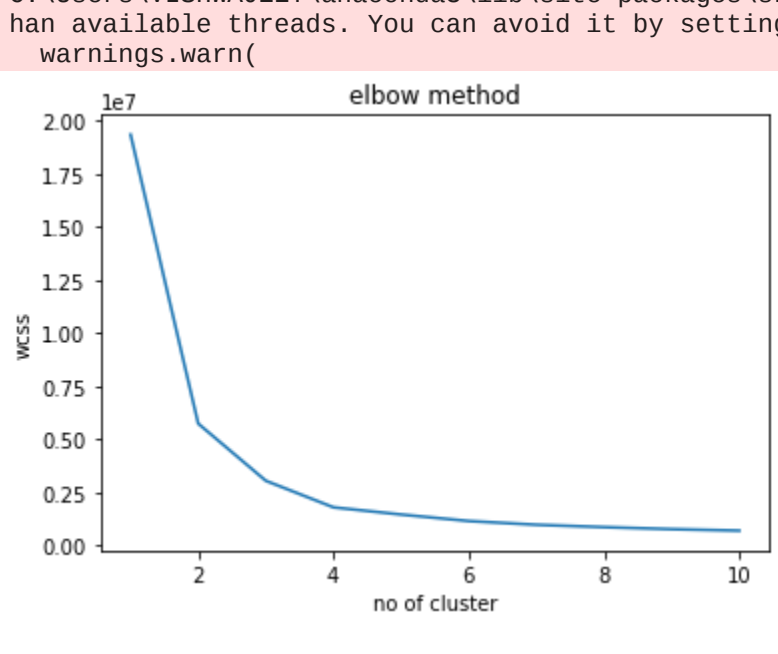
```
C:\Users\VISHWAJEET\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks t
han available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [48]:

```python
#ass 3 boston data set eblow method
#Apply K-Means Clustering technique to analyze the Bostan dataset.Elbow method to find best value of K.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
from sklearn.cluster import KMeans

boston=load_boston()
wcss=[]

for i in range(1,11):
    model=KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_state=0)
    model.fit(boston.data)
    wcss.append(model.inertia_)

plt.plot(range(1,11),wcss)
plt.title("elbow method")
plt.xlabel("no of cluster")
plt.ylabel("wcss")
plt.show()
```

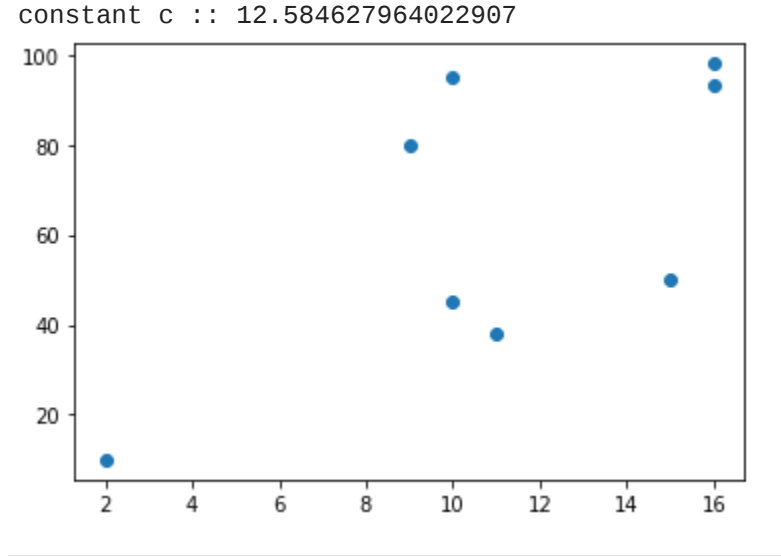
```
C:\Users\VISHWAJEET\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks t
han available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```



In [3]:

```python
#assignmnt 4 linear regression
#Apply Linear Regression technique to solve the given problem.
#The following table shows the results of a recently conducted study on the correlation of the
#number of hours spent driving with the risk of developing acute backache. Find the
#equation of the best fit line for this data.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

data = [[10,95],[9,80],[2,10],[15,50],[10,45],[16,98],[11,38],[16,93]]
data=pd.DataFrame(data)

x=data.iloc[:,:-1]
y=data.iloc[:,-1]

lr=LinearRegression()
lr.fit(x,y)

plt.scatter(x,y)

print("accuracy ::",lr.score(x,y))
print("slope m ::",lr.coef_)
print("constant c ::",lr.intercept_)
```

```
accuracy :: 0.43709481451010035
slope m :: [4.58789861]
constant c :: 12.584627964022907
```



In [60]:

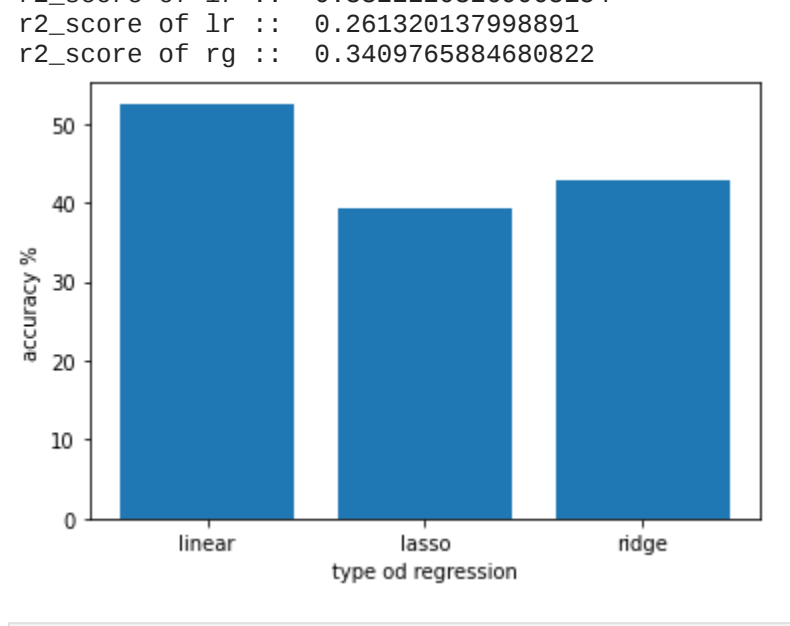
```python
#asss5 linear regression on dabetes
#5 Apply Linear Regression technique of machine learning to analyze the Diabetes dataset.Display accuracy of the model.
#Find the equation of the best fit line for this data.

import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_diabetes

dia=load_diabetes()

x=dia.data
y=dia.target

lr=LinearRegression()
lr.fit(x,y)

print("accuracy by score",lr.score(x,y))
print("slope m " ,lr.coef_.mean())

cvs_lr=cross_val_score(lr,x,y,cv=5)
cv_mlr=cvs_lr.mean()
print("accuracy by cross validation ==",cvs_lr.mean())
```

```
accuracy by score 0.5177494254132934
slope m  137.59740575237754
accuracy by cross validation == 0.4823181221114939
```

In [72]:

```python
#assig 6 lr,ridge,lasso
#Apply Linear, Ridge, Lasso Regression technique of machine learning to analyze and build the model of the Diabetes dataset.
#Display and compare the accuracy (Cross-Validation, R2 Score) of all the models.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
from sklearn.datasets import load_diabetes
from sklearn.model_selection import cross_val_score

dia=load_diabetes()
x=dia.data
y=dia.target

X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.2,random_state=0)

lr=LinearRegression()
lr.fit(X_train,Y_train)

cr_lr=cross_val_score(lr,X_train,Y_train,cv=5)
cr_lr.mean()

predicated=lr.predict(np.array(X_test))
print("r2_score of lr :: ",r2_score(Y_test,predicated))

#lasso
ls=Lasso()
ls.fit(X_train,Y_train)

cr_ls=cross_val_score(ls,X_train,Y_train,cv=5)

predicated_ls=ls.predict(np.array(X_test))
print("r2_score of lr :: ",r2_score(Y_test,predicated_ls))

#ridge
rg=Ridge()
rg.fit(X_train,Y_train)

cr_rg=cross_val_score(rg,X_train,Y_train,cv=5)

predicated_rg=rg.predict(np.array(X_test))
print("r2_score of rg :: ",r2_score(Y_test,predicated_rg))

#plot graph

x=['linear','lasso','ridge']
y=[100*cr_lr.mean(),100*cr_ls.mean(),100*cr_rg.mean()]
plt.bar(x,y)
plt.xlabel("type od regression")
plt.ylabel("accuracy %")
plt.show()
```

```
r2_score of lr ::  0.33222203269065154
r2_score of lr ::  0.261320137998891
r2_score of rg ::  0.3409765884680822
```



In [15]:

```python
#7 desioci tree buy
#Apply Decision Tree Classification technique to solve given problem:
#A dataset collected in a cosmetics shop showing details of customers and whether or not
#they responded to a special offer to buy a new lip-stick is shown in table below. Use this
#dataset to build a decision tree, with Buys as the target variable, to help in buying lip-sticks
#in the future. Find the root node of decision tree. According to the decision tree you have
#made from previous training data set, what is the decision for the test data: [Age < 21,
#Income = Low, Gender = Female, Marital Status = Married]?

import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier

dataset = {
'Id':[1,2,3,4,5,6,7,8,9,10,11,12,13,14],
'Age':['<21','<21','21-35','>35','>35','>35','21-35','<21','<21','>35','<21','21-35','21-35','>35'],
'Income':['High','High','High','Medium','Low','Low','Low','Medium','Low','Medium','Medium','Medium','High','Medium'],
'Gender':['Male','Male','Male','Male','Female','Female','Female','Male','Female','Female','Female','Male','Female','Male'],
'MaritalStatus':['Single','Married','Single','Single','Single','Married','Married','Single','Married','Single','Married','Married','Single','Married'],
'Buys':['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']}

df=pd.DataFrame(dataset)
print(df)

X=df.iloc[:,1:-1]
Y=df["Buys"]

le=LabelEncoder();
x=X.apply(le.fit_transform)
print(x)

print("Age:",list(zip(df.iloc[:,0], x.iloc[:,0])))
print("Income:",list(zip(df.iloc[:,1], x.iloc[:,1])))
print("Gender:",list(zip(df.iloc[:,2], x.iloc[:,2])))
print("MaritialStatus:",list(zip(df.iloc[:,3], x.iloc[:,3])))

dt=DecisionTreeClassifier()
dt.fit(x,Y)

print(Y)

query=np.array([1,1,0,0])
pred=dt.predict([query])
print("ans is ",pred[0])
```

```
    Id    Age  Income  Gender MaritalStatus Buys
0    1    <21    High    Male        Single   No
1    2    <21    High    Male       Married   No
2    3  21-35    High    Male        Single  Yes
3    4    >35  Medium    Male        Single  Yes
4    5    >35     Low  Female        Single  Yes
5    6    >35     Low  Female       Married   No
6    7  21-35     Low  Female       Married  Yes
7    8    <21  Medium    Male        Single   No
8    9    <21     Low  Female       Married  Yes
9   10    >35  Medium  Female        Single  Yes
10  11    <21  Medium  Female       Married  Yes
11  12  21-35  Medium    Male       Married  Yes
12  13  21-35    High  Female        Single  Yes
13  14    >35  Medium    Male       Married   No
    Age  Income  Gender  MaritalStatus
0     1       0       1              1
1     1       0       1              0
2     0       0       1              1
3     2       2       1              1
4     2       1       0              1
5     2       1       0              0
6     0       1       0              0
7     1       2       1              1
8     1       1       0              0
9     2       2       0              1
10    1       2       0              0
11    0       2       1              0
12    0       0       0              1
13    2       2       1              0
Age: [(1, 1), (2, 1), (3, 0), (4, 2), (5, 2), (6, 2), (7, 0), (8, 1), (9, 1), (10, 2), (11, 1), (12, 0), (13, 0), (14, 2)]
Income: [('<21', 0), ('<21', 0), ('21-35', 0), ('>35', 2), ('>35', 1), ('>35', 1), ('21-35', 1), ('<21', 2), ('<21', 1), ('>35', 2), ('<21', 2), ('21-35', 2), ('21-35', 0), ('>35',
2)]
Gender: [('High', 1), ('High', 1), ('High', 1), ('Medium', 1), ('Low', 0), ('Low', 0), ('Low', 0), ('Medium', 1), ('Low', 0), ('Medium', 0), ('Medium', 0), ('Medium', 1), ('High',
0), ('Medium', 1)]
MaritialStatus: [('Male', 1), ('Male', 0), ('Male', 1), ('Male', 1), ('Female', 1), ('Female', 0), ('Female', 0), ('Male', 1), ('Female', 0), ('Female', 1), ('Female', 0), ('Male',
0), ('Female', 1), ('Male', 0)]
0      No
1      No
2     Yes
3     Yes
4     Yes
5      No
6     Yes
7      No
8     Yes
9     Yes
10    Yes
11    Yes
12    Yes
13     No
Name: Buys, dtype: object
ans is  Yes
```

In [18]:

```python
#ass 8 knn color
#Apply k-NN Classification technique to solve given problem:
#In the following diagram let blue circles indicate positive examples and orange squares
#indicate negative examples. We want to use k-NN algorithm for classifying the points. If
#k=3, find the class of the point (6,6).

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score,accuracy_score,confusion_matrix
from sklearn.neighbors import KNeighborsClassifier

data = [[2,4,'negative'],[4,6,'negative'],[4,4,'positive'],[4,2,'negative'],[6,4,'negative'],[6,2,'positive']]
data=np.array(data)

x=data[:,:-1]
y=data[:,-1]

knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x,y)

y_predict=knn.predict(x)

x_train=np.array([[6,6]])
Y_pre=knn.predict(x_train)

print("class of point",Y_pre)
print("accuracy score",accuracy_score(y,y_predict))
print("confusion matrix ::",confusion_matrix(y,y_predict))
```

```
class of point ['negative']
accuracy score 0.3333333333333333
confusion matrix :: [[2 2]
 [2 0]]
C:\Users\VISHWAJEET\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: FutureWarning: Arrays of bytes/strings is being converted to decimal numbers if dtype='numeric'. Thi
s behavior is deprecated in 0.24 and will be removed in 1.1 (renaming of 0.26). Please convert your data to numeric values explicitly instead.
  return f(*args, **kwargs)
C:\Users\VISHWAJEET\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: FutureWarning: Arrays of bytes/strings is being converted to decimal numbers if dtype='numeric'. Thi
s behavior is deprecated in 0.24 and will be removed in 1.1 (renaming of 0.26). Please convert your data to numeric values explicitly instead.
  return f(*args, **kwargs)
```

In [14]:

```python
#ass 9 desicion tree
#9 Consider the following training data set. Write a program to construct a decision tree
#using ID3 algorithm. Display Accuracy measures for the same and predict a class of
#suitable query.


import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier

datasets = {
'Day':['D1','D2','D3','D4','D5','D6','D7','D8','D9','D10','D11','D12','D13','D14'],
'OutLook':['Sunny','Sunny','Overcast','Rain','Rain','Rain','Overcast','Sunny','Sunny','Rain','Sunny','Overcast','Overcast','Rain'],
'Temperature':['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild','Cool','Mild','Mild','Mild','Hot','Mild'],
'Humidity':['High','High','High','High','Normal','Normal','Normal','High','Normal','Normal','Normal','High','Normal','High'],
'Wind':['Weak','Strong','Weak','Weak','Weak','Strong','Strong','Weak','Weak','Weak','Strong','Strong','Weak','Strong'],
'PlayTennis':['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']}

df=pd.DataFrame(datasets)
print(df)

X=df.iloc[:,1:-1]
Y=df["PlayTennis"]

laen=LabelEncoder();
x=X.apply(laen.fit_transform)
print(x)

print("Outllok:",list(zip(df.iloc[:,0], x.iloc[:,0])))
print("Temperature:",list(zip(df.iloc[:,1], x.iloc[:,1])))
print("Humidity:",list(zip(df.iloc[:,2], x.iloc[:,2])))
print("Wind:",list(zip(df.iloc[:,3], x.iloc[:,3])))

print(Y)

dtc=DecisionTreeClassifier()
dtc.fit(x,Y)

queryy=np.array([1,1,0,0])
predi=dtc.predict([queryy])
print("ans is ::", predi[0])
```

```
    Id    Age  Income  Gender MaritalStatus Buys
0    1    <21    High    Male        Single   No
1    2    <21    High    Male       Married   No
2    3  21-35    High    Male        Single  Yes
3    4    >35  Medium    Male        Single  Yes
4    5    >35     Low  Female        Single  Yes
5    6    >35     Low  Female       Married   No
6    7  21-35     Low  Female       Married  Yes
7    8    <21  Medium    Male        Single   No
8    9    <21     Low  Female       Married  Yes
9   10    >35  Medium  Female        Single  Yes
10  11    <21  Medium  Female       Married  Yes
11  12  21-35  Medium    Male       Married  Yes
12  13  21-35    High  Female        Single  Yes
13  14    >35  Medium    Male       Married   No
    Age  Income  Gender  MaritalStatus
0     1       0       1              1
1     1       0       1              0
2     0       0       1              1
3     2       2       1              1
4     2       1       0              1
5     2       1       0              0
6     0       1       0              0
7     1       2       1              1
8     1       1       0              0
9     2       2       0              1
10    1       2       0              0
11    0       2       1              0
12    0       0       0              1
13    2       2       1              0
Age: [(1, 1), (2, 1), (3, 0), (4, 2), (5, 2), (6, 2), (7, 0), (8, 1), (9, 1), (10, 2), (11, 1), (12, 0), (13, 0), (14, 2)]
Income: [('<21', 0), ('<21', 0), ('21-35', 0), ('>35', 2), ('>35', 1), ('>35', 1), ('21-35', 1), ('<21', 2), ('<21', 1), ('>35', 2), ('<21', 2), ('21-35', 2), ('21-35', 0), ('>35',
2)]
Gender: [('High', 1), ('High', 1), ('High', 1), ('Medium', 1), ('Low', 0), ('Low', 0), ('Low', 0), ('Medium', 1), ('Low', 0), ('Medium', 0), ('Medium', 0), ('Medium', 1), ('High',
0), ('Medium', 1)]
MaritialStatus: [('Male', 1), ('Male', 0), ('Male', 1), ('Male', 1), ('Female', 1), ('Female', 0), ('Female', 0), ('Male', 1), ('Female', 0), ('Female', 1), ('Female', 0), ('Male',
0), ('Female', 1), ('Male', 0)]
0      No
1      No
2     Yes
3     Yes
4     Yes
5      No
6     Yes
7      No
8     Yes
9     Yes
10    Yes
11    Yes
12    Yes
13     No
Name: Buys, dtype: object
ans is  Yes
```

In [19]:

```python
# ass 10 good bad using knn
#Consider tissue paper factory application. Apply KNN algorithm to find class of new
#tissue paper (X1= 3, X2=7). Assume K=3

import pandas as np
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix,accuracy_score,r2_score

data=[[7,7,'BAD'],[7,4,'BAD'],[3,4,'GOOD'],[1,4,'GOOD']]
data=np.array(data)

x=data[:,:-1]
y=data[:,-1]

knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(x,y)

y_pre=knn.predict(x)

X_train=np.array([[3,7]])
y_ans=knn.predict(X_train)

print("classs of point is ::",y_ans)
print("accuracy is :: ",accuracy_score(y,y_pre))
print("confusion matrix ",confusion_matrix(y,y_pre))
```

```
classs of point is :: ['GOOD']
accuracy is ::  1.0
confusion matrix  [[2 0]
 [0 2]]
C:\Users\VISHWAJEET\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: FutureWarning: Arrays of bytes/strings is being converted to decimal numbers if dtype='numeric'. Thi
s behavior is deprecated in 0.24 and will be removed in 1.1 (renaming of 0.26). Please convert your data to numeric values explicitly instead.
  return f(*args, **kwargs)
C:\Users\VISHWAJEET\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: FutureWarning: Arrays of bytes/strings is being converted to decimal numbers if dtype='numeric'. Thi
s behavior is deprecated in 0.24 and will be removed in 1.1 (renaming of 0.26). Please convert your data to numeric values explicitly instead.
  return f(*args, **kwargs)
```

In [ ]: