importing libraries

In [85]:
```
import numpy as np
import pandas as pd
```

reading dataset

In [86]:
```
df = pd.read_csv('D:\Downloads\play_tennis.csv')
df
```

Out[86]:

	day	outlook	temp	humidity	wind	play
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes
5	D6	Rain	Cool	Normal	Strong	No
6	D7	Overcast	Cool	Normal	Strong	Yes
7	D8	Sunny	Mild	High	Weak	No
8	D9	Sunny	Cool	Normal	Weak	Yes
9	D10	Rain	Mild	Normal	Weak	Yes
10	D11	Sunny	Mild	Normal	Strong	Yes
11	D12	Overcast	Mild	High	Strong	Yes
12	D13	Overcast	Hot	Normal	Weak	Yes
13	D14	Rain	Mild	High	Strong	No

In [87]:
```
df.shape
df.head()
```

Out[87]:

	day	outlook	temp	humidity	wind	play
0	D1	Sunny	Hot	High	Weak	No
1	D2	Sunny	Hot	High	Strong	No
2	D3	Overcast	Hot	High	Weak	Yes
3	D4	Rain	Mild	High	Weak	Yes
4	D5	Rain	Cool	Normal	Weak	Yes

spliting data in feature and target

In [88]:
```
# Show the list of columns
feature=list(df.columns[0:5])
```

```
print("Feature columns: \n{}".format(feature))
#Separate the data into feature data and target data (X_all and y_all, respectively)
x= df[feature].values
y= df['play'].values
x
```

Out[88]:

```
Feature columns:
['day', 'outlook', 'temp', 'humidity', 'wind']
array([['D1', 'Sunny', 'Hot', 'High', 'Weak'],
       ['D2', 'Sunny', 'Hot', 'High', 'Strong'],
       ['D3', 'Overcast', 'Hot', 'High', 'Weak'],
       ['D4', 'Rain', 'Mild', 'High', 'Weak'],
       ['D5', 'Rain', 'Cool', 'Normal', 'Weak'],
       ['D6', 'Rain', 'Cool', 'Normal', 'Strong'],
       ['D7', 'Overcast', 'Cool', 'Normal', 'Strong'],
       ['D8', 'Sunny', 'Mild', 'High', 'Weak'],
       ['D9', 'Sunny', 'Cool', 'Normal', 'Weak'],
       ['D10', 'Rain', 'Mild', 'Normal', 'Weak'],
       ['D11', 'Sunny', 'Mild', 'Normal', 'Strong'],
       ['D12', 'Overcast', 'Mild', 'High', 'Strong'],
       ['D13', 'Overcast', 'Hot', 'Normal', 'Weak'],
       ['D14', 'Rain', 'Mild', 'High', 'Strong']], dtype=object)
```

naive bayes algorithm

In [89]:

```
#function to calculate conditional probablity
def conditional_prob(feature,col_num,out):
    num=0
    den=0
    for i in range(len(y)):
        if ((x[i][col_num].lower()==feature.lower()) and (y[i].lower()==out.lower())
            num=num+1
    for item in y:
        if (item.lower()==out.lower()):
            den=den+1
    return num/den
```

In [90]:

```
#function to calculate prior probablity
def prior_prob(out):
    num=0
    for item in y:
        if (item.lower()==out.lower()):
            num=num+1
    return num/len(y)
```

In [91]:

```
def nb(new_instance):
    p_yes=prior_prob("Yes")
    for i in range(len(feature)-1):
        p_yes=p_yes*conditional_prob(new_instance[i],i+1,"Yes")
    p_no=prior_prob("No")
    for i in range(len(feature)-1):
        p_no=p_no*conditional_prob(new_instance[i],i+1,"No")
    #normalization
    pnb_y=p_yes/(p_yes+p_no)
    pnb_n=p_no/(p_yes+p_no)
    print("probablity of yes : ",pnb_y)
    print("probablity of no : ",pnb_n)
    if(pnb_y>pnb_n):
        print("classification = yes")
```

```
    else:
         print("classification = no")
```

new instance for classification

In [92]:

```
new_instance=[]
print("enter new instance : ")
for i in range(len(feature)-1):
    new_instance.append(input())
nb(new_instance)
```

```
enter new instance :
sunny
cool
normal
strong
probablity of yes :  0.6729475100942126
probablity of no :  0.32705248990578734
classification = yes
```