Naive Bayes Algorithm

Assignment 20

In [1]:
```
#use for classification problem
```

In [2]:
```
from sklearn import preprocessing
```

In [3]:
```
from sklearn.naive_bayes import GaussianNB
```

In [4]:
```
weather=['Sunny','Sunny',
        'Overcast','Rainy',
        'Rainy','Rainy',
        'Overcast','Sunny',
        'Sunny','Rainy',
        'Sunny','Overcast',
        'Overcast','Rainy']

temp=['High','High',
     'High','Medium',
     'Low','Low',
     'Low','Medium',
      'Low','Medium',
      'Medium','Medium',
      'High','Medium'
     ]

play=['No','No',
     'Yes','Yes',
     'Yes','No',
     'Yes','No',
     'Yes','Yes',
     'Yes','Yes',
     'Yes','No']
```

Creating label encoder

In [5]:
```
le=preprocessing.LabelEncoder()
```

Converting string labels into numbers

In [6]:
```
weather_encoded=le.fit_transform(weather)
temp_encoded=le.fit_transform(temp)
label=le.fit_transform(play)
```

In [7]:
```
print(weather_encoded,temp_encoded,label)
```

```
[2 2 0 1 1 1 0 2 2 1 2 0 0 1] [0 0 0 2 1 1 1 2 1 2 2 2 0 2] [0 0 1 1 1 0 1 0
1 1 1 1 1 0]
```

In [8]:
```
# 0--> Overcast 1 --> Rainy 2 -->Sunny
# 0-->High 1--> Low 2-->Medium
# 0-->No 1-->Yes
```

Combine weather and humidity in single tuple or feature

In [10]:
```
features=list(zip(weather_encoded,temp_encoded))
```

Create a gaussian classifier

In [12]:
```
model=GaussianNB()
```

In [13]:
```
model.fit(features,label)   #Train the model using training set
```

Out[13]:
```
GaussianNB()
```

In [14]:
```
print("Enter Weather and Humidity Conditions :")
w,h=map(int,input().split())

predicted=model.predict([[w,h]])

print(predicted)

if predicted==1:
    print('Yes')
else:
    print('No')
```

```
Enter Weather and Humidity Conditions :
1 2
[1]
Yes
```

In [15]:
```
#For Weather :  0--> Overcast 1 --> Rainy 2 -->Sunny
#For Humidity : 0-->High 1--> Low 2-->Medium
```

In [16]:
```
w=input("Enter the weather condition: 0--> Overcast 1 --> Rainy 2 -->Sunny")
h=input("Enter the temperature as 0-->High 1--> Low 2-->Medium ")

if w=='Overcast':
    w=0

elif w=='Rainy':
    w=1

else:
    w=2

if h=='High':

    h=0

elif h=='Low':
    h=1

else:
    h=2
predicted=model.predict([[w,h]])


if predicted==1:
    print('Yes,you play ')
else:
    print('No,you dont play')
```

```
Enter the weather condition: 0--> Overcast 1 --> Rainy 2 -->Sunnyovercast
Enter the temperature as 0-->High 1--> Low 2-->Medium 1
No,you dont play
```

Assignment 20

Example Iris dataset

In [17]:
```
import seaborn as sns
```

In [18]:
```
ir=sns.load_dataset('iris')
ir.head(140)
```

Out[18]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
135	7.7	3.0	6.1	2.3	virginica
136	6.3	3.4	5.6	2.4	virginica
137	6.4	3.1	5.5	1.8	virginica
138	6.0	3.0	4.8	1.8	virginica
139	6.9	3.1	5.4	2.1	virginica

140 rows × 5 columns

In [20]:
```
from sklearn.datasets import load_iris
```

In [21]:
```
iris=load_iris()
```

In [22]:
```
iris.data[:5]
```

Out[22]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

In [23]:
```
iris.target[:10]
```

Out[23]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [24]:
```
x=iris.data
y=iris.target
```

In [25]:
```
from sklearn.model_selection import train_test_split
```

In [26]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=
```

In [27]:
```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
```

In [28]:
```
model=nb.fit(x_train,y_train)
```

In [29]:
```
predictions=nb.predict(x_test)
```

In [30]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

In [31]:
```
r = classification_report(y_test,predictions)
print(r)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       0.86      0.92      0.89        13
           2       0.93      0.87      0.90        15

    accuracy                           0.93        45
   macro avg       0.93      0.93      0.93        45
weighted avg       0.93      0.93      0.93        45
```

In [32]:
```
cm=confusion_matrix(y_test,predictions).ravel()
```

In [33]:
```
cm
```

Out[33]:
```
array([17,  0,  0,  0, 12,  1,  0,  2, 13], dtype=int64)
```

In [40]:
```
ir.describe()
```

Out[40]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [39]:
```
sl=input("Enter the sepal_length(4.3 - 7.9 ): ")
sw=input("Enter the sepal_width(2.0 - 4.4): ")
pl=input("Enter the petal_length(1.0 - 6.9): ")
pw=input("Enter the petal_width(0.1 - 2.5): ")

predicted=model.predict([[sl,sw,pl,pw]])

print(predicted)

if predicted==0:
    print("It is setosa.")
elif predicted==1:
    print("It is Versicolor.")
else:
    print("It is a Virginica.")
```

```
Enter the sepal_length(4.3 - 7.9 ): 4.4
Enter the sepal_width(2.0 - 4.4): 4
Enter the petal_length(1.0 - 6.9): 6.6
Enter the petal_width(0.1 - 2.5): 2
[2]
It is a Virginica.

C:\Users\keerti chouhan\anaconda3\lib\site-packages\sklearn\base.py:566: Futu
reWarning: Arrays of bytes/strings is being converted to decimal numbers if d
type='numeric'. This 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 i
nstead.
  X = check_array(X, **check_params)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: