Gaussian Naive Byes

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv ('naive_bayes_dataset.csv')
dataset = data
dataset
```

Out[2]:

	Age	Income	Student	Credit_Rating	Buys_Computer
0	<=30	high	no	fair	no
1	<=30	high	no	excellent	no
2	31-40	high	no	fair	yes
3	>40	medium	no	fair	yes
4	>40	low	yes	fair	yes
5	>40	low	yes	excellent	no
6	31-40	low	yes	excellent	yes
7	<=30	medium	no	fair	no
8	<=30	low	yes	fair	yes
9	>40	medium	yes	fair	yes
10	<=30	medium	yes	excellent	yes
11	31-40	medium	no	excellent	yes
12	31-40	high	yes	fair	yes
13	>40	medium	no	excellent	no

In [3]:
```
data.head()
```

Out[3]:

	Age	Income	Student	Credit_Rating	Buys_Computer
0	<=30	high	no	fair	no
1	<=30	high	no	excellent	no
2	31-40	high	no	fair	yes
3	>40	medium	no	fair	yes
4	>40	low	yes	fair	yes

Feture Selection

In [4]:
```
data.Age.unique()
```

Out[4]:
```
array(['<=30', '31-40', '>40'], dtype=object)
```

In [5]:
```
data.Income.unique()
```

Out[5]:
```
array(['high', 'medium', 'low'], dtype=object)
```

In [6]:
```
data.Credit_Rating.unique()
```

Out[6]:
```
array(['fair', 'excellent'], dtype=object)
```

In [7]:
```
data.Buys_Computer.unique()
```

Out[7]:
```
array(['no', 'yes'], dtype=object)
```

Transforming data

In [8]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [9]:
```
data['Age']=le.fit_transform(data['Age'])
data['Income']=le.fit_transform(data['Income'])
data['Student']=le.fit_transform(data['Student'])
data['Credit_Rating']=le.fit_transform(data['Credit_Rating'])
data['Buys_Computer']=le.fit_transform(data['Buys_Computer'])
```

In [10]:
```
x=data.drop(['Buys_Computer'],axis=1)
x.head()
```

Out[10]:

	Age	Income	Student	Credit_Rating
0	1	0	0	1
1	1	0	0	0
2	0	0	0	1
3	2	2	0	1
4	2	1	1	1

In [12]:
```
y=data['Buys_Computer']
y.head()
```

Out[12]:
```
0   0
1   0
2   1
3   1
4   1
Name: Buys_Computer, dtype: int32
```

Gaussian Naive Byes

In [13]:
```
from sklearn.naive_bayes import GaussianNB


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y)

gaussianmodel = GaussianNB()
gaussianmodel.fit(x_train,y_train)
```

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

In [14]:
```
gaussianmodel.score(x_test,y_test)
```

Out[14]:
```
1.0
```

In [ ]:
```

```

For Multi nomial Gaussian

In [15]:
```
prior = dataset.groupby('Buys_Computer').size().div(len(data))
prior
```

Out[15]:
```
Buys_Computer
0   0.357143
1   0.642857
dtype: float64
```

In [16]:
```
likelihood = {}
likelihood['Credit_Rating'] = dataset.groupby(['Buys_Computer', 'Credit_Rating']).size().div(len(dataset)).div(pr
likelihood['Age'] = dataset.groupby(['Buys_Computer', 'Age']).size().div(len(dataset)).div(prior)
likelihood['Income'] = dataset.groupby(['Buys_Computer', 'Income']).size().div(len(dataset)).div(prior)
likelihood['Student'] = dataset.groupby(['Buys_Computer', 'Student']).size().div(len(dataset)).div(prior)

likelihood
```

Out[16]:
```
{'Credit_Rating': Buys_Computer  Credit_Rating
0           0              0.600000
                           1              0.400000
1           0              0.333333
                           1              0.666667
dtype: float64,
'Age': Buys_Computer  Age
0           1     0.600000
                           2     0.400000
1           0     0.444444
                           1     0.222222
                           2     0.333333
dtype: float64,
'Income': Buys_Computer  Income
0           0       0.400000
                           1       0.200000
                           2       0.400000
1           0       0.222222
                           1       0.333333
                           2       0.444444
dtype: float64,
'Student': Buys_Computer  Student
0           0       0.800000
                           1       0.200000
1           0       0.333333
                           1       0.666667
dtype: float64}
```

In [17]:
```
dataset
```

Out[17]:

	Age	Income	Student	Credit_Rating	Buys_Computer
0	1	0	0	1	0
1	1	0	0	0	0
2	0	0	0	1	1
3	2	2	0	1	1
4	2	1	1	1	1
5	2	1	1	0	0
6	0	1	1	0	1
7	1	2	0	1	0
8	1	1	1	1	1
9	2	2	1	1	1
10	1	2	1	0	1
11	0	2	0	0	1
12	0	0	1	1	1
13	2	2	0	0	0

In [18]:
```
from sklearn.preprocessing import LabelEncoder
encoded_data = dataset.apply(LabelEncoder().fit_transform)
```

In [19]:
```
from sklearn.naive_bayes import MultinomialNB
import numpy as np
clf = MultinomialNB()
clf.fit(encoded_data.drop(['Buys_Computer'], axis=1), encoded_data['Buys_Computer'])
```

Out[19]:
```
MultinomialNB()
```

In [20]:
```
X = np.array([1,2,1,1])
print(X.shape)
print (clf._joint_log_likelihood(X.reshape(1,-1)))
print ("Prediction of : ", clf.predict(X.reshape(1,-1)))
```

```
(4,)
[[-8.29709436 -7.15971488]]
Prediction of :  [1]
```

In [ ]: