Naïve Bayes

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

Exercício 1

First Question

Dada a tabela ao lado, se uma pessoa tem 35 anos e ganha $50.000,00 (renda media), ele compra um computador?

In [2]:
```python
import pandas as pd

data = {'rec':            ['r1',          'r2',        'r3',      'r4',    'r5',          'r6',          'r7',
        'Age':            ['<=30',      '<=30',  '31...40',    '>40',   '>40',         '>40',     '31...40',
        'Income':         ['High',      'High',     'High', 'Medium',   'Low',         'Low',         'Low',
        'Student':        [  'No',        'No',       'No',     'No',   'Yes',         'Yes',         'Yes',
        'Credit_rating': ['Fair', 'Excellent',     'Fair',   'Fair',  'Fair',  'Excellent',  'Excellent',
        'Buys_computer': [  'No',        'No',      'Yes',    'Yes',   'Yes',          'No',         'Yes',
        }

df = pd.DataFrame (data, columns = ['rec','Age','Income','Student','Credit_rating','Buys_computer'])

df
```

Out[2]:

	rec	Age	Income	Student	Credit_rating	Buys_computer
0	r1	<=30	High	No	Fair	No
1	r2	<=30	High	No	Excellent	No
2	r3	31...40	High	No	Fair	Yes
3	r4	>40	Medium	No	Fair	Yes
4	r5	>40	Low	Yes	Fair	Yes
5	r6	>40	Low	Yes	Excellent	No
6	r7	31...40	Low	Yes	Excellent	Yes
7	r8	<=30	Medium	No	Fair	No
8	r9	<=30	Low	Yes	Fair	Yes
9	r10	>40	Medium	Yes	Fair	Yes
10	r11	<=30	Medium	Yes	Excellent	Yes
11	r12	31...40	Medium	No	Excellent	Yes
12	r13	31...40	High	Yes	Fair	Yes
13	r14	>40	Medium	No	Excellent	No

IMPORTANTE: Este exercício será solucionado de duas maneiras:

1º Solução) Considerando P=0 para os casos em que não existir nenhuma amostra que atenda à condição analisada;

2º Solução) Utilizando a fórmula onde:
P= (nc+mxp)/(n+m)
Considerando que m=1 (peso dado à priori)
nc - quantidade de amostras que atendem à condição
p=m/(total de casos)=1/14
n - depende da condição analisada, será igual ao número total de casos da condição *yes* OU número total de casos da condição *no*.

1º Solução

In [3]:
```python
#Definição das funções
def prob_yes(column,cond):
    """Probability of buy a computer with the input conditions"""
    if ((((df.loc[(df['Buys_computer']=='Yes') & (df[f'{column}']==f'{cond}'),f'{column}']).value_counts()).
        x= 0
    else:
        x= df.loc[(df['Buys_computer']=='Yes') & (df[f'{column}']==f'{cond}'),f'{column}'].value_counts()[0]
    return(x)

def prob_no(column,cond):
    """Probability of not buy a computer with the input conditions"""
    if ((((df.loc[(df['Buys_computer']=='No') & (df[f'{column}']==f'{cond}'),f'{column}']).value_counts()).a
        x= 0
    else:
        x= df.loc[(df['Buys_computer']=='No') & (df[f'{column}']==f'{cond}'),f'{column}'].value_counts()[0]/
    return(x)

#Cálculo das probabilidades de comprar ou não um computador
p_buy     = df[['Buys_computer']].value_counts()[0]/df['Buys_computer'].shape[0]
p_not_buy = df[['Buys_computer']].value_counts()[1]/df['Buys_computer'].shape[0]
```

In [4]:
```python
#Solução:
print('x’ = (Age=31...40,Income=Medium)')
print('\n')

#P_buy
print('P(Age=31...40 | Buys_computer=Yes)=',  prob_yes('Age','31...40'))
print('P(Income=Medium | Buys_computer=Yes)=',prob_yes('Income','Medium'))
print('P(Buys_computer=Yes)=',p_buy)
print('\n')

#P_not_buy
print('P(Age=31...40 | Buys_computer=No)=',  prob_no('Age','31...40'))
print('P(Income=Medium | Buys_computer=No)=',prob_no('Income','Medium'))
print('P(Buys_computer=No)=',p_not_buy)
print('\n')

#Resultado:
p_yes = prob_yes('Age','31...40')*prob_yes('Income','Medium')*p_buy
p_no = prob_no('Age','31...40')*prob_no('Income','Medium')*p_not_buy
print('P(Yes|x’)=',p_yes)
print('P(No|x’)=',p_no)
if(p_yes>p_no):
    print("Yes, he buys a computer.")
else:
    print("No, he didn't buy a computer.")
```

```
x’ = (Age=31...40,Income=Medium)


P(Age=31...40 | Buys_computer=Yes)= 0.4444444444444444
P(Income=Medium | Buys_computer=Yes)= 0.4444444444444444
P(Buys_computer=Yes)= 0.6428571428571429


P(Age=31...40 | Buys_computer=No)= 0
P(Income=Medium | Buys_computer=No)= 0.4
P(Buys_computer=No)= 0.35714285714285715


P(Yes|x’)= 0.12698412698412698
P(No|x’)= 0.0
Yes, he buys a computer.
```

2º Solução

In [5]:
```python
#Definição das funções
def prob_yes2(column,cond):
    """Probability of buy a computer with the input conditions"""
    if ((((df.loc[(df['Buys_computer']=='Yes') & (df[f'{column}']==f'{cond}'),f'{column}']).value_counts()).
        x= ((0 + 1/14)/(9+1))
    else:
        x= ((df.loc[(df['Buys_computer']=='Yes') & (df[f'{column}']==f'{cond}'),f'{column}'].value_counts()[
    return(x)

def prob_no2(column,cond):
    """Probability of not buy a computer with the input conditions"""
    if ((((df.loc[(df['Buys_computer']=='No') & (df[f'{column}']==f'{cond}'),f'{column}']).value_counts()).a
        x= ((0 + 1/14)/(5+1))
    else:
        x= ((df.loc[(df['Buys_computer']=='No') & (df[f'{column}']==f'{cond}'),f'{column}'].value_counts()[0
    return(x)

#Cálculo das probabilidades de comprar ou não um computador
p_buy     = df[['Buys_computer']].value_counts()[0]/df['Buys_computer'].shape[0]
p_not_buy = df[['Buys_computer']].value_counts()[1]/df['Buys_computer'].shape[0]
```

In [6]:
```python
#Solução:
print('x’ = (Age=31...40,Income=Medium)')
print('\n')

#P_buy
print('P(Age=31...40 | Buys_computer=Yes)=',  prob_yes2('Age','31...40'))
print('P(Income=Medium | Buys_computer=Yes)=',prob_yes2('Income','Medium'))
print('P(Buys_computer=Yes)=',p_buy)
print('\n')

#P_not_buy
print('P(Age=31...40 | Buys_computer=No)=',  prob_no2('Age','31...40'))
print('P(Income=Medium | Buys_computer=No)=',prob_no2('Income','Medium'))
print('P(Buys_computer=No)=',p_not_buy)
print('\n')

#Resultado:
p_yes = prob_yes2('Age','31...40')*prob_yes2('Income','Medium')*p_buy
p_no = prob_no2('Age','31...40')*prob_no2('Income','Medium')*p_not_buy
print('P(Yes|x’)=',p_yes)
print('P(No|x’)=',p_no)
if(p_yes>p_no):
    print("Yes, he buys a computer.")
else:
    print("No, he didn't buy a computer.")
```

```
x’ = (Age=31...40,Income=Medium)


P(Age=31...40 | Buys_computer=Yes)= 0.40714285714285714
P(Income=Medium | Buys_computer=Yes)= 0.40714285714285714
P(Buys_computer=Yes)= 0.6428571428571429


P(Age=31...40 | Buys_computer=No)= 0.011904761904761904
P(Income=Medium | Buys_computer=No)= 0.3452380952380953
P(Buys_computer=No)= 0.35714285714285715


P(Yes|x’)= 0.1065634110787172
P(No|x’)= 0.0014678490443796569
Yes, he buys a computer.
```

Second Question

E se a pessoa for estudante e tiver um crédito bom (fair)?

1º Solução

In [7]:
```python
#Solução:
print('x’ = (Age=31...40, Income=Medium, Student=Yes, Credit_rating=Fair)')
print('\n')

#P_buy
print('P(Age=31...40 | Buys_computer=Yes)=',  prob_yes('Age','31...40'))
print('P(Income=Medium | Buys_computer=Yes)=',prob_yes('Income','Medium'))
print('P(Student=Yes | Buys_computer=Yes)=',prob_yes('Student','Yes'))
print('P(Credit_rating=Fair | Buys_computer=Yes)=',prob_yes('Credit_rating','Fair'))
print('P(Buys_computer=Yes)=',p_buy)
print('\n')

#P_not_buy
print('P(Age=31...40 | Buys_computer=No)=',  prob_no('Age','31...40'))
print('P(Income=Medium | Buys_computer=No)=',prob_no('Income','Medium'))
print('P(Student=Yes | Buys_computer=No)=',prob_no('Student','Yes'))
print('P(Credit_rating=Fair | Buys_computer=No)=',prob_no('Credit_rating','Fair'))
print('P(Buys_computer=No)=',p_not_buy)
print('\n')

#Resultado:
p_yes = prob_yes('Age','31...40')*prob_yes('Income','Medium')*prob_yes('Student','Yes')*prob_yes('Credit_rat
p_no = prob_no('Age','31...40')*prob_no('Income','Medium')*prob_no('Student','Yes')*prob_no('Credit_rating',
print('P(Yes|x’)=',p_yes)
print('P(No|x’)=',p_no)
if(p_yes>p_no):
    print("Yes, he buys a computer.")
else:
    print("No, he didn't buy a computer.")
```

```
x’ = (Age=31...40, Income=Medium, Student=Yes, Credit_rating=Fair)


P(Age=31...40 | Buys_computer=Yes)= 0.4444444444444444
P(Income=Medium | Buys_computer=Yes)= 0.4444444444444444
P(Student=Yes | Buys_computer=Yes)= 0.6666666666666666
P(Credit_rating=Fair | Buys_computer=Yes)= 0.6666666666666666
P(Buys_computer=Yes)= 0.6428571428571429


P(Age=31...40 | Buys_computer=No)= 0
P(Income=Medium | Buys_computer=No)= 0.4
P(Student=Yes | Buys_computer=No)= 0.2
P(Credit_rating=Fair | Buys_computer=No)= 0.4
P(Buys_computer=No)= 0.35714285714285715


P(Yes|x’)= 0.056437389770723094
P(No|x’)= 0.0
Yes, he buys a computer.
```

2º Solução

In [8]:
```python
#Solução:
print('x’ = (Age=31...40, Income=Medium, Student=Yes, Credit_rating=Fair)')
print('\n')

#P_buy
print('P(Age=31...40 | Buys_computer=Yes)=',  prob_yes2('Age','31...40'))
print('P(Income=Medium | Buys_computer=Yes)=',prob_yes2('Income','Medium'))
print('P(Student=Yes | Buys_computer=Yes)=',prob_yes2('Student','Yes'))
print('P(Credit_rating=Fair | Buys_computer=Yes)=',prob_yes2('Credit_rating','Fair'))
print('P(Buys_computer=Yes)=',p_buy)
print('\n')

#P_not_buy
print('P(Age=31...40 | Buys_computer=No)=',  prob_no2('Age','31...40'))
print('P(Income=Medium | Buys_computer=No)=',prob_no2('Income','Medium'))
print('P(Student=Yes | Buys_computer=No)=',prob_no2('Student','Yes'))
print('P(Credit_rating=Fair | Buys_computer=No)=',prob_no2('Credit_rating','Fair'))
print('P(Buys_computer=No)=',p_not_buy)
print('\n')

#Resultado:
p_yes = prob_yes2('Age','31...40')*prob_yes2('Income','Medium')*prob_yes2('Student','Yes')*prob_yes2('Credit
p_no = prob_no2('Age','31...40')*prob_no2('Income','Medium')*prob_no2('Student','Yes')*prob_no2('Credit_rati
print('P(Yes|x’)=',p_yes)
print('P(No|x’)=',p_no)
if(p_yes>p_no):
    print("Yes, he buys a computer.")
else:
    print("No, he didn't buy a computer.")
```

```
x’ = (Age=31...40, Income=Medium, Student=Yes, Credit_rating=Fair)


P(Age=31...40 | Buys_computer=Yes)= 0.40714285714285714
P(Income=Medium | Buys_computer=Yes)= 0.40714285714285714
P(Student=Yes | Buys_computer=Yes)= 0.6071428571428571
P(Credit_rating=Fair | Buys_computer=Yes)= 0.6071428571428571
P(Buys_computer=Yes)= 0.6428571428571429


P(Age=31...40 | Buys_computer=No)= 0.011904761904761904
P(Income=Medium | Buys_computer=No)= 0.3452380952380953
P(Student=Yes | Buys_computer=No)= 0.17857142857142858
P(Credit_rating=Fair | Buys_computer=No)= 0.3452380952380953
P(Buys_computer=No)= 0.35714285714285715


P(Yes|x’)= 0.03928166556345569
P(No|x’)= 9.049239431762341e-05
Yes, he buys a computer.
```

Exercício 2

Use o Naïve Bayes Gaussiano para classificar o dataset Iris, e mais dois de sua escolha (os mesmos do trabalho anterior).

Importante: Para solucionar este exercício, primeiramente será aplicado o PCA a fim de reduzir as dimensões dos datasets para uma dimensão e posteriormente, será aplicado o método Naïve Bayes Gaussiano. Como o PCA já foi implementado em aulas anteriores, será utilizada a biblioteca sckit-learn para esta etapa.

In [9]:
```python
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
```

Dataset Iris

Classificate into 'setosa', 'versicolor' or 'virginica'.

In [10]:
```python
#Load the dataset
iris = load_iris()
X = iris['data']
iris['target_names']
y = iris['target']

#PCA to reduce for dimentions into only one
pca = PCA(n_components = 1)
pca.fit(iris['data'])
Transformed_X = pca.transform(iris['data'])
Transformed_X

#Separate iris samples - 0-setosa, 1-versicolor, 2-virginica
setosa = Transformed_X[y==0]
versicolor = Transformed_X[y==1]
virginica = Transformed_X[y==2]
```

In [11]:
```python
#Função para calcular a probabilidade
def prob_calc(std,mean):
    z=[]
    for x in Transformed_X:
        z.append((1/(std*(2*np.pi)**(1/2))) * (np.exp(-((x-mean)**2) / (2*std**2))))
    return z
```

In [12]:
```python
prob_0=prob_calc(np.std(setosa),np.mean(setosa))
prob_1=prob_calc(np.std(versicolor),np.mean(versicolor))
prob_2=prob_calc(np.std(virginica),np.mean(virginica))

classification=[]
for i in range(len(prob_0)):
    if ((prob_0[i]>prob_1[i])&(prob_0[i]>prob_2[i])):
        classification.append(0)
    if ((prob_1[i]>prob_0[i])&(prob_1[i]>prob_2[i])):
        classification.append(1)
    if ((prob_2[i]>prob_0[i])&(prob_2[i]>prob_1[i])):
        classification.append(2)
print(classification)
print('\n')
x=[]
c=0
e=0
for i in range(len(classification)):
    if (classification[i]==y[i]):
        c=c+1
    else:
        e=e+1
print('Porcentagem de acerto:',(c/y.shape[0])*100)
print('Porcentagem de erro:',(e/y.shape[0])*100)
```

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


Porcentagem de acerto: 94.0
Porcentagem de erro: 6.0
```

Interpretação do resultado: Podemos observar que a porcentagem de erro é muito pequena comparada à porcentagem de acerto, e portanto, o método Naïve Bayes teve apresentou um resultado satisfatório para a classificação do dataset Iris.

Dataset Wholesale customers data.csv

Classificate into regions (Regions - Lisnon, Oporto or Other (Nominal)).

In [26]:
```python
# Importando o dataset
X = pd.read_csv('data/Wholesale customers data.csv')

#classify into region:
region = X['Region']
X.drop(columns=['Region','Channel'], inplace=True)

#PCA to reduce for dimentions into only one
pca = PCA(n_components = 1)
pca.fit(X)
Transformed_X = pca.transform(X)

#Separate classes into 03 samples
r1 = Transformed_X[region==1]
r2 = Transformed_X[region==2]
r3 = Transformed_X[region==3]
```

In [27]:
```python
prob_1=prob_calc(np.std(one),np.mean(r1))
prob_2=prob_calc(np.std(two),np.mean(r2))
prob_3=prob_calc(np.std(three),np.mean(r3))

classification=[]
for i in range(len(prob_1)):
    if ((prob_1[i]>prob_2[i])&(prob_1[i]>prob_3[i])):
        classification.append(1)
    if ((prob_2[i]>prob_1[i])&(prob_2[i]>prob_3[i])):
        classification.append(2)
    if ((prob_3[i]>prob_1[i])&(prob_3[i]>prob_2[i])):
        classification.append(3)
print(classification)
print('\n')

#Análise do resultado obtido
x=[]
c=0
e=0
for i in range(len(classification)):
    if (classification[i]==region[i]):
        c=c+1
    else:
        e=e+1
print('Porcentagem de acerto:',(c/region.shape[0])*100)
print('Porcentagem de erro:',(e/region.shape[0])*100)
```

```
[2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 2, 2, 2, 1, 2, 2, 2, 3, 3, 3, 2, 2, 2, 2, 3, 2, 2, 1, 3, 2, 2,
3, 2, 2, 3, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 3, 2, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1,
2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 3, 3, 3, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2,
2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 3, 3, 1, 3, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 2,
2, 3, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2,
2, 3, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 3, 2, 3, 2, 2, 3, 3, 2, 2, 3, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 3, 3, 3, 3, 2, 2,
2, 3, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
3, 3, 2, 2, 2, 2, 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2,
1, 2, 2, 2, 2, 2, 2, 2, 3, 2, 3, 1, 2, 2, 2, 2, 2, 3, 2, 2, 3, 2, 3, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 3, 2, 2,
2, 2, 2, 2, 2, 3, 3, 3, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2, 2,
1, 2, 2, 3, 3, 2, 2, 2]


Porcentagem de acerto: 22.272727272727273
Porcentagem de erro: 77.72727272727272
```

Interpretação do resultado: Podemos observar que a porcentagem de erro é muito superior à % de acerto. Portanto, neste caso não é recomendado classificar o dataset através de regiões utilizando o método Naive Bayes, uma vez que o resultado obtido não terá uma precisão satisfatória.

Wine dataset

Classificate into 'class_0', 'class_1' or 'class_2'.

In [24]:
```python
#Import dataset
from sklearn.datasets import load_wine
wine = load_wine()
X = wine['data']
y = wine['target']

#PCA to reduce for dimentions into only one
pca = PCA(n_components = 1)
pca.fit(X)
Transformed_X = pca.transform(X)
Transformed_X

#Separate classes into 03 samples
c0 = Transformed_X[y==0]
c1 = Transformed_X[y==1]
c2 = Transformed_X[y==2]
```

In [25]:
```python
prob_0=prob_calc(np.std(c0),np.mean(c0))
prob_1=prob_calc(np.std(c1),np.mean(c1))
prob_2=prob_calc(np.std(c2),np.mean(c2))

classification=[]
for i in range(len(prob_0)):
    if ((prob_0[i]>prob_1[i])&(prob_0[i]>prob_2[i])):
        classification.append(0)
    if ((prob_1[i]>prob_0[i])&(prob_1[i]>prob_2[i])):
        classification.append(1)
    if ((prob_2[i]>prob_0[i])&(prob_2[i]>prob_1[i])):
        classification.append(2)
print(classification)
print('\n')
#Análise do resultado obtido
x=[]
c=0
e=0
for i in range(len(classification)):
    if (classification[i]==y[i]):
        c=c+1
    else:
        e=e+1
print('Porcentagem de acerto:',(c/y.shape[0])*100)
print('Porcentagem de erro:',(e/y.shape[0])*100)
```

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


Porcentagem de acerto: 72.47191011235955
Porcentagem de erro: 27.52808988764045
```

Interpretação do resultado: Podemos observar que a porcentagem de erro é muito pequena comparada à porcentagem de acerto, e portanto, o método Naïve Bayes teve apresentou um resultado satisfatório para a classificação do dataset Wine.