Importación de librerías.

In [1]:
```
#Librerias
import sklearn
import numpy as np
import pandas as pd
import urllib.request
import csv

from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB

from sklearn.model_selection import train_test_split

from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
```

Importación de datos.

In [2]:
```
#Tratamiento de datos
s = "house-votes-84.data"
myfile = pd.read_csv(s,encoding = 'utf8',header = None,sep=",")

names=["Class Name", "handicapped-infants", "water-project-cost-sharing", "ado
ption-of-the-budget-resolution", "physician-fee-freeze", "el-salvador-aid", "r
eligious-groups-in-schools", "anti-satellite-test-ban", "aid-to-nicaraguan-con
tras","mx-missile", "immigration","synfuels-corporation-cutback","education-sp
ending","superfund-right-to-sue","crime","duty-free-exports","export-administr
ation-act-south-africa"]
for i in range (0,myfile.shape[1]):
    myfile.rename(columns={myfile.columns[i]:names[i]}, inplace = True)
```

Tras leer los datos, los siguientes pasos serían:

1. Transformar las variables categóricas a númericas (y:1,n:0), (demócratas:0, republicanos:0).
2. Tratar los valores perdidos. La estrategia de eliminar las filas con valores indefinidos nos hace perder demasiada información, dado que tenemos al rededor de 200 (46%) filas con valores perdidos. Por ello, pasamos a tratar estos datos como el valor más repetido en la misma columna, es decir, el valor más frecuente en dicha carácteristica de los congresistas.

In [3]:
```
#Variables categoricas a numericas
myfile.values[myfile.values=="y"]=1
myfile.values[myfile.values=="n"]=0
myfile.values[myfile.values=="?"]=np.nan

#Codificar las variable dependiente
labelencoder_X=LabelEncoder()
myfile.values[:,0]=labelencoder_X.fit_transform(myfile.values[:,0])

#Variables desconocidas
imputer=SimpleImputer(missing_values=np.nan, strategy="median")
imputer.fit(myfile.values[:,1:17])
myfile.values[:,1:17]=imputer.transform(myfile.values[:,1:17])
```

Definimos los datos de entrenamiento y test de nuestro modelo. Elegimos el 75%de los datos como datos de test y 25% como dato de entremaiento.

In [4]:
```
#Definir variables independientes e independientes
X=myfile.values[:,1:myfile.shape[1]]
y=myfile.values[:,0]
#Dividir datos de entrenamiento y test
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25)

X_train = X_train.astype(np.float64)
y_train = y_train.astype(np.float64)
X_test = X_test.astype(np.float64)
y_test = y_test.astype(np.float64)
```

Se realizan pruebas con el modelo de Bernouilli y el de Gaus. Se imprime por pantalla el modelo de Gaus por mayor calidad en el modelo y datos predecidos.

In [5]:
```
#Modelos

#Gaus
GausNB=GaussianNB()
GausNB.fit(X_train, y_train)
print(GausNB)
y_pred= GausNB.predict(X_test)
gausac=accuracy_score(y_test,y_pred)
print("Precisión: ", gausac)
print(metrics.classification_report(y_test,y_pred))
print(metrics.confusion_matrix(y_test,y_pred),"\n")

#Bernoulli
#BernNB=BernoulliNB(binarize=True)
#BernNB.fit(X_train, y_train)
#print(BernNB)
#y_pred=BernNB.predict(X_test)
#bernac=accuracy_score(y_test, y_pred)
#print("Precision: ", bernac)
#print(metrics.classification_report(y_test,y_pred))
#print(metrics.confusion_matrix(y_test,y_pred))
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
Precisión:  0.944954128440367
              precision    recall  f1-score   support

         0.0       0.95      0.95      0.95        61
         1.0       0.94      0.94      0.94        48

   micro avg       0.94      0.94      0.94       109
   macro avg       0.94      0.94      0.94       109
weighted avg       0.94      0.94      0.94       109

[[58  3]
 [ 3 45]]
```