** Naive Bayes Model ** En este documento se desarrolla y analiza un modelo Naive Bayes para la preaprobación de créditos de consumo en una FINTECH a partir de la calidad de las variables socioeconómicas que definen un solicitante de crédito. Este modelo fue implementado utilizado para ello la libreria sklearn.

0. Se procede con la carga de las librerías

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
```

1. Se procede con la carga de los datos

```
nxl="/content/0. SolicitantesCrédito(USD).xlsx"
XDB=pd.read_excel(nxl,sheet_name=1)

XDB=XDB[['Edad','Ingresos','Egresos','Monto (EAD)','Plazo','Cuota (COP)','Estrato']]
XDB=XDB.dropna()

XD=np.array(XDB[['Edad','Ingresos','Egresos','Monto (EAD)','Plazo','Cuota (COP)']])
yd=np.array(XDB[['Estrato']])
```

2. Se crea el modelo de clasificación

```
#Se procede con la construcción del modelo propuesto
mnb=GaussianNB()
mnb.fit(XD,yd)
ydp=mnb.predict(XD)
ydpdf=pd.DataFrame(ydp)

#Para obtener la información del modelo
u=mnb.theta_
sigma=np.sqrt(mnb.var_)
cv=u/sigma
LI=u-sigma; LS=u+sigma
ndc=mnb.class_count_
pdnc=mnb.class_prior_  #% de los datos por categoría de riesgo

#Evaluar el comportamiento dle modelo
cm=confusion_matrix(yd,ydp)
df=pd.DataFrame(cm)
df.to_excel("MatrizConfusión.xlsx")

cm=cm/np.sum(cm,axis=1)[:,np.newaxis]
VP=cm[0,0];FN=cm[0,1];FP=cm[1,0];VN=cm[1,1]

print(cm)
print(cm/np.sum(cm,axis=0))

#Se procede con la estimación de un valor particular
X=np.array([[34,850,250,1250,24,52]])
Vp=np.exp(-0.5*((u[:,:]-X)/sigma)**2)
np.where(np.max(np.prod(Vp,axis=1))==np.prod(Vp,axis=1))
```

```
[[0.66272189 0.33136095 0.00591716 0.         0.         0.        ]
 [0.34182909 0.40429785 0.24187906 0.         0.0089955  0.0029985 ]
 [0.22240595 0.28400165 0.38238942 0.07895825 0.02687061 0.00537412]
 [0.17052023 0.21531792 0.29190751 0.1300578  0.17919075 0.01300578]
 [0.12569316 0.16266174 0.24399261 0.12754159 0.3142329  0.025878  ]
 [0.15       0.1        0.1        0.15       0.45       0.05      ]]
[[0.39608753 0.22125539 0.00467359 0.         0.         0.        ]
 [0.20430023 0.26995661 0.19104477 0.         0.00918574 0.03083088]
 [0.13292487 0.18963278 0.30202489 0.16227933 0.02743887 0.05525725]
 [0.10191445 0.14377147 0.23055904 0.26730194 0.18298032 0.13372672]
 [0.07512275 0.10861203 0.19271412 0.26213048 0.32087837 0.2660802 ]
 [0.08965017 0.06677172 0.07898359 0.30828824 0.4595167  0.51410495]]
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column
  y = column_or_1d(y, warn=True)
(array([4]),)
```

```
#Como se obtienen los valores del histograma
XD3=np.array(XD)
freq,bins=np.histogram(XD3[:,0],bins=10)

#Se procede con la estimación de los valores de referencia
max=np.max(XD3[:,0])
min=np.min(XD3[:,0])
```

```
rango=max-min
dx=10
ti=rango/dx

LI=np.zeros((10,1)); LS=np.zeros((10,1)); fi=np.zeros((10,1))
MC=np.zeros((10,1)); fi=np.zeros((10,1))
for i in range(10):
  LI[i,]=min+i*ti
  LS[i,]=min+(i+1)*ti
  MC[i,]=(LI[i,]+LS[i,])/2
  print(i,min+i*ti,min+(i+1)*ti,MC[i,])
  fi[i,]=np.sum((XD3[:,0]>=LI[i,])&(XD3[:,0]<LS[i,]))

fi=fi/np.sum(fi)
u2=np.sum(MC*fi)
print("La media de los datos es:",np.sum(MC*fi))
np.sqrt(np.sum(((MC-u2)**2)*fi))
```

```
0 21.0 26.2 [23.6]
1 26.2 31.4 [28.8]
2 31.4 36.6 [34.]
3 36.6 41.8 [39.2]
4 41.8 47.0 [44.4]
5 47.0 52.2 [49.6]
6 52.2 57.4 [54.8]
7 57.4 62.6 [60.]
```