IRIS con NaiveBayes y Cross Validation



Adrián Yared Armas de la Nuez

**Contenido**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**1. Enunciado 2**](#_2c35568x6wg8)

[**2. Actividad 3**](#_gaz96zc6sqe8)

[**2.1 Código explicado 3**](#_t8cwbo6n4466)

[**2.1.1 Imports 3**](#_klshywv1sk4j)

[**2.1.2 Carga de los datos 3**](#_pdoqq9xrcct3)

[**2.1.3 División en train y test 3**](#_cuqn07pgkb5i)

[**2.1.4 Modelos Naive Bayes 3**](#_u8jl47tqd8ox)

[**2.1.5 Discretización de los datos para CategoricalNB 4**](#_ktbp8em18yn5)

[**2.1.7 Tabla solicitada 4**](#_a91fw2mukk80)

[**2.1.8 Mostrar los resultados 4**](#_wd681ibl8726)

[**2.2 Código completo 5**](#_3aqdhxjrhq7b)

[**2.3 Justificación de los resultados 6**](#_mgz246po6pqf)

[**3. Github y Colab 7**](#_lmdngf8dlin3)

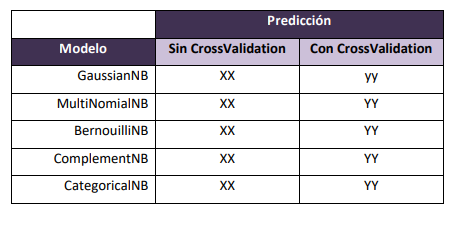
## 

## 

## 

## **1. Enunciado**

El objetivo de esta actividad es observar las diferencias en la Predicción en los modelos de NaiveBayes al utilizar o no la Validación cruzada en el entrenamiento de los modelos. Para ello anotaremos los resultados en la siguiente tabla comparativa:



En la primera columna indicamos los resultados obtenidos en la actividad 3.2 – Comparativa clasificadores NaiveBayes Como referencia se pueden utilizar los siguientes ejemplos:

Título: Ejemplo\_3\_2\_Iris\_NaiveBayes - GaussianNB.ipynb Url: <https://colab.research.google.com/drive/1tjF8jjv33sGhgTL3UjWlQh0iwMypjhAj?usp=sharing>

Título: Ejemplo\_3\_3\_Clasificación\_con\_Naive\_Bayes\_(Heart\_Diseases).ipynb Url:<https://colab.research.google.com/drive/1hwri6X-N_cHmpZs31-zyK2XwRyfA4EGN?usp=sharing>

Una vez realizados los ajustes y el entrenamiento, concluir si hay diferencias entre utilizar o no la Validación Cruzada (Cross Validation).

## 

## **2. Actividad**

### **2.1 Código explicado**

#### **2.1.1 Imports**

# Imports

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.naive\_bayes import GaussianNB, MultinomialNB, BernoulliNB, ComplementNB, CategoricalNB

from sklearn.preprocessing import KBinsDiscretizer

from sklearn.metrics import accuracy\_score

#### **2.1.2 Carga de los datos**

# Iris data Load

data = load\_iris()

X, y = data.data, data.target

#### **2.1.3 División en train y test**

# Train and test data split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

#### **2.1.4 Modelos Naive Bayes**

# Naive Bayes Models

models = {

"GaussianNB": GaussianNB(),

"MultinomialNB": MultinomialNB(),

"BernoulliNB": BernoulliNB(),

"ComplementNB": ComplementNB(),

}

#### 

#### **2.1.5 Discretización de los datos para CategoricalNB**

# Para CategoricalNB, data discretization

categorical\_encoder = KBinsDiscretizer(n\_bins=10, encode='ordinal', strategy='uniform')

X\_categorical = categorical\_encoder.fit\_transform(X)

models["CategoricalNB"] = CategoricalNB()

#### **2.1.7 Tabla solicitada**

Estructura:

# Results

results = {"Model": [], "Without Cross-Validation": [], "With Cross-Validation": []}

Relleno de los resultados:

for name, model in models.items():

# with NO cross validation

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

# with cross validation

if name == "CategoricalNB":

cv\_scores = cross\_val\_score(model, X\_categorical, y, cv=5)

else:

cv\_scores = cross\_val\_score(model, X, y, cv=5)

Guardado en la tabla:

**# Save results**

**results["Model"].append(name)**

**results["Without Cross-Validation"].append(accuracy)**

**results["With Cross-Validation"].append(cv\_scores.mean())**

#### **2.1.8 Mostrar los resultados**

# Show results

import pandas as pd

results\_df = pd.DataFrame(results)

print(results\_df)

### 

### **2.2 Código completo**

# Imports

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.naive\_bayes import GaussianNB, MultinomialNB, BernoulliNB, ComplementNB, CategoricalNB

from sklearn.preprocessing import KBinsDiscretizer

from sklearn.metrics import accuracy\_score

# Iris data Load

data = load\_iris()

X, y = data.data, data.target

# Train and test data split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Naive Bayes Models

models = {

"GaussianNB": GaussianNB(),

"MultinomialNB": MultinomialNB(),

"BernoulliNB": BernoulliNB(),

"ComplementNB": ComplementNB(),

}

# Para CategoricalNB, data discretization

categorical\_encoder = KBinsDiscretizer(n\_bins=10, encode='ordinal', strategy='uniform')

X\_categorical = categorical\_encoder.fit\_transform(X)

models["CategoricalNB"] = CategoricalNB()

# Results

results = {"Model": [], "Without Cross-Validation": [], "With Cross-Validation": []}

for name, model in models.items():

# with NO cross validation

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

# with cross validation

if name == "CategoricalNB":

cv\_scores = cross\_val\_score(model, X\_categorical, y, cv=5)

else:

cv\_scores = cross\_val\_score(model, X, y, cv=5)

# Save results

results["Model"].append(name)

results["Without Cross-Validation"].append(accuracy)

results["With Cross-Validation"].append(cv\_scores.mean())

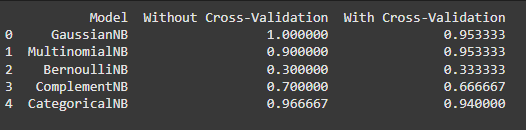
# Show results

import pandas as pd

results\_df = pd.DataFrame(results)

print(results\_df)

### **2.3 Justificación de los resultados**



Los resultados obtenidos parecen razonables y están alineados con el comportamiento esperado de los diferentes modelos Naive Bayes al aplicarse al conjunto de datos Iris.

**GaussianNB**

Trabaja bien con datos continuos y distribuciones gaussianas.

Resultados esperados: Una precisión alta (generalmente >90%), tanto con como sin validación cruzada.

Los resultados para GaussianNB (1.0 sin CV, 0.9533 con CV) son consistentes y razonables.

**MultinomialNB**

Este modelo es más adecuado para datos discretos o de conteo

Una precisión algo menor que GaussianNB

Los resultados (0.9 sin CV, 0.9533 con CV) sugieren que MultinomialNB se beneficia de la validación cruzada. Esto es plausible, ya que la CV suaviza posibles sesgos de la partición inicial.

**BernoulliNB**

Este modelo es más adecuado para datos binarios

Precisión significativamente baja

Los resultados (0.3 sin CV, 0.3333 con CV) reflejan este comportamiento esperado

**ComplementNB**

Es una variante de MultinomialNB diseñada para clases desbalanceadas

Un desempeño intermedio, mejor que BernoulliNB pero posiblemente inferior a GaussianNB

Los resultados (0.7 sin CV, 0.6667 con CV) son razonables para este caso

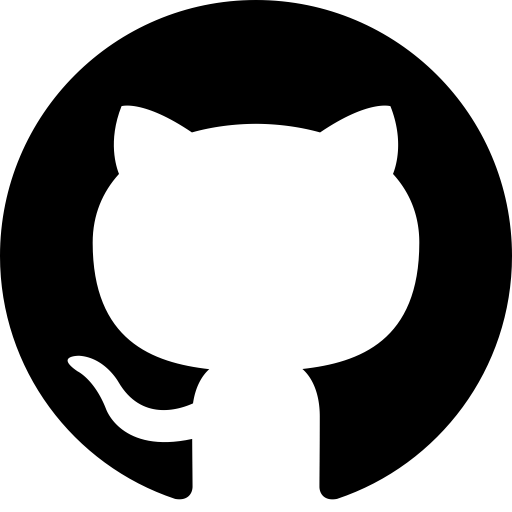
**CategoricalNB**

Este modelo está diseñado específicamente para datos categóricos. Para usarlo con Iris, discretizar los datos es necesario (como hice en el código con KBinsDiscretizer)

Una precisión muy alta, pero algo menor que GaussianNB, ya que los datos de Iris no son naturalmente categóricos, que es la especialidad de este

Los resultados (0.9667 sin CV, 0.94 con CV) son consistentes y razonables si los datos fueron discretizados correctamente

## **3. Github y Colab**

[](https://github.com/AdrianYArmas/IaBigData/tree/main/SNS/3%20%20-%20Algoritmos%20y%20herramientas%20para%20el%20aprendizaje%20supervisado%20/3.3%20IRIS%20con%20NaiveBayes%20y%20Cross%20Validation) [](https://colab.research.google.com/drive/1As4RS9v2ZmfWmigcLtcIxykHrW-aI6_i?usp=sharing)