

# Práctica '6' {

## [Naive Bayes]

< Profesora: Consuelo Varinia García Mendoza >

< Alumna: Vianey Maravilla Pérez

Diana Itzel García Rodríguez

}

{

Especificaciones:

- ◆ Los dataset utilizados para esta práctica serán iris.csv y emails.csv

**En el dataset emails.csv:**

- ◆ La primera columna indica el id del correo
- ◆ La última columna indica si el correo es spam o no
- ◆ El resto de las columnas son las palabras más comunes en todos los correos

}

{

Especificaciones:

**En el dataset iris.csv:**

- ◆ Las primeras 4 columnas son las características de las instancias
- ◆ La última columna es la clase

}

{

Especificaciones:

**Ambos dataset:**

- ◆ Carga el dataset
- ◆ Crea un conjunto de entrenamiento del 70% de las instancias y el conjunto de pruebas con el 30%(set random\_state = 0)
- ◆ Utilizando el conjunto de entrenamiento crea un conjunto de validación de 3 pliegues
- ◆ Utilizando el conjunto de validación realiza lo siguiente con cada pliegue:

}



01

{

[Código Fuente  
y Resultados]

}

# Código Fuente

```
In [1]: 1 # Importamos las librerías que se requieren para la realización de la práctica
        2
        3 import pandas as pd
        4 import numpy as np
        5 import matplotlib.pyplot as plt
        6
        7 from sklearn.model_selection import train_test_split
        8 from sklearn.model_selection import KFold
        9
       10 from sklearn.naive_bayes import GaussianNB
       11 from sklearn.naive_bayes import MultinomialNB
       12 from sklearn.naive_bayes import BernoulliNB
       13
       14 from sklearn.metrics import accuracy_score
       15 from sklearn.metrics import classification_report
       16 from sklearn.metrics import confusion_matrix
       17 from sklearn.metrics import ConfusionMatrixDisplay
       18
```

# Código Fuente

```
In [97]: 1 df = pd.read_csv(r"emails.csv", sep=',', engine='python')
          2 X = df.drop(df.columns[[0,len(df.columns)-1]],axis=1).values
          3 y = df['Prediction'].values
```

```
In [98]: 1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle = True, random_state=0)
```



# Código Fuente

```
In [99]: 1 # DIVISION EN K-PLIEGUES
2
3 class validation_set:
4     def __init__(self, X_train, y_train, X_test, y_test):
5         self.X_train = X_train
6         self.y_train = y_train
7         self.X_test = X_test
8         self.y_test = y_test
9
10
11
12 class test_set:
13     def __init__(self, X_test, y_test):
14         self.X_test = X_test
15         self.y_test = y_test
16
17
18
19 class data_set:
20     def __init__(self, validation_set, test_set):
21         self.validation_set = validation_set
22         self.test_set = test_set
23
24
25 # Numero de pliegues
26 n_pliegues = 3
27
28
29 validation_sets = []
30 kf = KFold(n_pliegues)
31
32
```

# Código Fuente

```
33 for train_index, test_index in kf.split(X_train):
34     print("TRAIN:", train_index, "\n", "TEST:",
35           test_index)
36     X_train_, X_test_ = X_train[train_index], X_train[test_index]
37     y_train_, y_test_ = y_train[train_index], y_train[test_index]
38     print(X_train_.shape)
39     print(X_test_.shape)
40     # print(y_train_.shape)
41     print(y_test_.shape)
42
43
44     validation_sets.append(validation_set(X_train_, y_train_, X_test_, y_test_))
45
46
47
48 my_test_set = test_set(X_test, y_test)
49
50
51 my_data_set = data_set(validation_sets, my_test_set)
52
```

# Código Fuente

```
1  TRAIN: [1207 1208 1209 ... 3617 3618 3619]
2  TEST: [ 0    1    2 ... 1204 1205 1206]
3
4  (2413, 3000)
5  (1207, 3000)
6  (1207,)
7  TRAIN: [ 0    1    2 ... 3617 3618 3619]
8  TEST: [1207 1208 1209 ... 2411 2412 2413]
9  (2413, 3000)
10 (1207, 3000)
11 (1207,)
12 TRAIN: [ 0    1    2 ... 2411 2412 2413]
13 TEST: [2414 2415 2416 ... 3617 3618 3619]
14 (2414, 3000)
    (1206, 3000)
    (1206,)
```

# Código Fuente

```
In [100]: 1 tiposModelo = ['Gauss', 'Multinomial']  
          2 nombreDataset = []  
          3 clasificadores = []  
          4 promediosExactitudesPliegues = []  
          5 exactitudesPrueba = []
```

# Código Fuente

```
In [70]: 1 # Modelo gaussiano aplicado a los tres pliegues
2 nPliegue = 0
3 exactitudes = []
4 dataset = 'emails.csv'
5 modelo = 'Gauss'
6 for pliegue in validation_sets:
7     nPliegue = nPliegue+1
8     clf = GaussianNB()
9     clf.fit(pliegue.X_train, pliegue.y_train)
10    y_predict = clf.predict(pliegue.X_test)
11    target_names = clf.classes_
12    print(target_names)
13    # target_names = target_names.tolist()
14    print ('-----Gaussian NB-----')
15    print(f'Pliegue: {nPliegue}')
16    print('Resultados de la predicción:')
17    print (y_predict)
18    print(y_predict.shape)
19    print('Probabilidades sin la funcion logaritmo')
20    print(f'Clases: {target_names}')
21    print(clf.predict_proba(pliegue.X_test))
22    exactitud = accuracy_score(pliegue.y_test, y_predict)
23    exactitudes.append(exactitud)
24    print(f'Accuracy: {exactitud}')
25    print(f'Numero de instancias predichas correctamente: {accuracy_score(pliegue.y_test, y_predict, normalize=False)}')
26    print('\nReporte de clasificación')
27    print(classification_report(pliegue.y_test, y_predict, target_names=target_names))
28    print('\nMatriz de confusión')
29    print(confusion_matrix(pliegue.y_test, y_predict, labels=clf.classes_))
30    print(exactitudes)
31    promedioExactitud = sum(exactitudes) / len(exactitudes)
32    promediosExactitudesPliegues.append(promedioExactitud)
33    nombreDataset.append(dataset)
34    clasificadores.append(modelo)
35    print(f"El promedio de exactitud de la predicción de los pliegues es: {promedioExactitud}")
```

# Código Fuente

```
['0', '1']
-----Gaussian NB-----
Pliegue: 1
Resultados de la prediccion:
[1 0 0 ... 0 0 0]
(1207,)
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[0. 1.]
 [1. 0.]
 [1. 0.]
 ...
 [1. 0.]
 [1. 0.]
 [1. 0.]]
Accuracy: 0.947804473902237
Numero de instancias predichas correctamente: 1144

Reporte de clasificación
      precision    recall  f1-score   support

      0       0.97       0.96       0.96         829
      1       0.91       0.93       0.92         378

   accuracy       0.94
  macro avg       0.94
weighted avg       0.95

Matriz de confusión
[[794  35]
 [ 28 350]]
['0', '1']
```

# Código Fuente

```

-----Gaussian NB-----
Pliegue: 2
Resultados de la prediccion:
[1 1 0 ... 1 0 0]
(1207,)
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[1.65305246e-225 1.00000000e+000]
 [0.00000000e+000 1.00000000e+000]
 [1.00000000e+000 0.00000000e+000]
 ...
 [0.00000000e+000 1.00000000e+000]
 [1.00000000e+000 0.00000000e+000]
 [1.00000000e+000 0.00000000e+000]]
Accuracy: 0.9428334714167357
Numero de instancias predichas correctamente: 1138

Reporte de clasificación

```

	precision	recall	f1-score	support
0	0.97	0.95	0.96	880
1	0.87	0.93	0.90	327
accuracy			0.94	1207
macro avg	0.92	0.94	0.93	1207
weighted avg	0.94	0.94	0.94	1207

```

Matriz de confusión
[[834 46]
 [ 23 304]]
['0', '1']

```

# Código Fuente

```
-----Gaussian NB-----
Pliegue: 3
Resultados de la prediccion:
[0 0 1 ... 1 0 1]
(1206,)
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[1. 0.]
 [1. 0.]
 [0. 1.]
 ...
 [0. 1.]
 [1. 0.]
 [0. 1.]]
Accuracy: 0.9494195688225538
Numero de instancias predichas correctamente: 1145

Reporte de clasificación
      precision    recall  f1-score   support

     0       0.98      0.95      0.96      852
     1       0.89      0.94      0.92      354

   accuracy          0.95
  macro avg       0.93      0.95      0.94
weighted avg       0.95      0.95      0.95

Matriz de confusión
[[811  41]
 [ 20 334]]
[0.947804473902237, 0.9428334714167357, 0.9494195688225538]
El promedio de exactitud de la predicción de los pliegues es: 0.9466858380471755
```



# Código Fuente

```
1 # Modelo multinomial
2 nPliegue = 0
3 exactitudes = []
4 dataset = 'emails.csv'
5 modelo = 'Multinomial'
6 for pliegue in validation_sets:
7     nPliegue = nPliegue+1
8     clf = MultinomialNB()
9     clf.fit(pliegue.X_train, pliegue.y_train)
10    y_predict = clf.predict(pliegue.X_test)
11    target_names = clf.classes_
12    print(target_names)
13    print('-----Multinomial NB-----')
14    print(f'Pliegue: {nPliegue}')
15    print('Resultados de la prediccion:')
16    print(y_predict)
17    print(y_predict.shape)
18    print('Probabilidades sin la funcion logaritmo')
19    print(f'Clases: {target_names}')
20    print(clf.predict_proba(pliegue.X_test))
21    exactitud = accuracy_score(pliegue.y_test, y_predict)
22    exactitudes.append(exactitud)
23    print(f'Accuracy: {exactitud}')
24    print(f'Numero de instancias predichas correctamente: {accuracy_score(pliegue.y_test, y_predict, normalize=False)}')
25    print('\nReporte de clasificación')
26    print(classification_report(pliegue.y_test, y_predict, target_names=target_names))
27    print('\nMatriz de confusión')
28    print(confusion_matrix(pliegue.y_test, y_predict, labels=clf.classes_))
29    print(exactitudes)
30    promedioExactitud = sum(exactitudes) / len(exactitudes)
31    promediosExactitudesPliegues.append(promedioExactitud)
32    nombreDataset.append(dataset)
33    clasificadores.append(modelo)
34    print(f"El promedio de exactitud de la predicción de los pliegues es: {promedioExactitud}")
```

# Código Fuente

```
[ '0', '1' ]
-----Multinomial NB-----
Pliegue: 1
Resultados de la prediccion:
[1 0 0 ... 0 0 0]
(1207,)
Probabilidades sin la funcion logaritmo
Clases: [ '0', '1' ]
[[2.71519188e-24 1.00000000e+00]
 [1.00000000e+00 1.35904582e-23]
 [1.00000000e+00 9.38001999e-16]
 ...
 [1.00000000e+00 2.43342953e-14]
 [1.00000000e+00 6.50230787e-61]
 [1.00000000e+00 5.16452354e-85]]
Accuracy: 0.9453189726594863
Numero de instancias predichas correctamente: 1141

Reporte de clasificación
      precision    recall  f1-score   support

     0       0.97      0.95      0.96         829
     1       0.89      0.94      0.92         378

 accuracy          0.95         1207
 macro avg          0.93         1207
weighted avg          0.95         1207

Matriz de confusión
[[785  44]
 [ 22 356]]
[ '0', '1' ]
```

# Código Fuente

```

-----Multinomial NB-----
Pliegue: 2
Resultados de la prediccion:
[0 1 0 ... 1 0 0]
(1207,)
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[9.20935643e-001 7.90643567e-002]
 [1.53904681e-082 1.00000000e+000]
 [1.00000000e+000 5.79196808e-177]
 ...
 [1.70916652e-029 1.00000000e+000]
 [1.00000000e+000 3.47712129e-134]
 [1.00000000e+000 7.14022280e-039]]
Accuracy: 0.9378624689312345
Numero de instancias predichas correctamente: 1132

Reporte de clasificación
              precision    recall  f1-score   support

         0       0.98        0.93        0.96         880
         1       0.84        0.95        0.89         327

   accuracy       0.94
  macro avg       0.91
weighted avg       0.94

Matriz de confusión
[[822  58]
 [ 17 310]]
['0', '1']

```

## Código Fuente

```

-----Multinomial NB-----
Pliegue: 3
Resultados de la prediccion:
[0 0 1 ... 1 0 1]
(1206,)
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[1.00000000e+000 1.36147654e-029]
 [1.00000000e+000 2.01630547e-072]
 [2.47409574e-007 9.99999753e-001]
 ...
 [1.99238367e-186 1.00000000e+000]
 [1.00000000e+000 0.00000000e+000]
 [1.48863334e-048 1.00000000e+000]]
Accuracy: 0.9469320066334992
Numero de instancias predichas correctamente: 1142

Reporte de clasificación
      precision    recall  f1-score   support

         0        0.97        0.95        0.96         852
         1        0.89        0.94        0.91         354

   accuracy                    0.95         1206
  macro avg        0.93        0.94        0.94         1206
weighted avg        0.95        0.95        0.95         1206

Matriz de confusión
[[810  42]
 [ 22 332]]
[0.9453189726594863, 0.9378624689312345, 0.9469320066334992]
El promedio de exactitud de la predicción de los pliegues es: 0.9433711494080734

```

# Código Fuente

```
In [72]: 1 data = {'Dataset' : nombreDataset, 'Clasificador': clasificadores, 'Accuracy Promedio - 3 pliegues': promediosExactitudesPli  
2 modelos = pd.DataFrame(data)  
3 print(modelos)
```

	Dataset	Clasificador	Accuracy Promedio - 3 pliegues
0	emails.csv	Gauss	0.946686
1	emails.csv	Multinomial	0.943371

# Código Fuente

```
1  # Modelo multinomial aplicado a los conjuntos de entrenamiento
2  for modelo in tiposModelo:
3      if modelo == 'Gauss':
4          clf = GaussianNB()
5          print('\n-----Gauss NB-----')
6      elif modelo == 'Multinomial':
7          clf = MultinomialNB()
8          print('\n-----Multinomial NB-----')
9      elif modelo == 'Bernoulli':
10         clf = BernoulliNB()
11         print('\n-----Bernoulli NB-----')
12     else:
13         print("El modelo ingresado no esta disponible")
14         break
15     print(dataset)
16     clf.fit(X_train, y_train)
17     y_predict = clf.predict(X_train)
18     target_names = clf.classes_
19     print(target_names)
20     print('Resultados de la prediccion:')
21     print(y_predict) #Para
22     totalPredicciones = y_predict.shape[0]
23     print(totalPredicciones)
24     print('Probabilidades sin la funcion logaritmo')
25     print(f'Clases: {target_names}')
26     print(clf.predict_proba(X_train))
27     exactitud = accuracy_score(y_train, y_predict)
28     exactitudes.append(exactitud)
29     print(f'Accuracy: {exactitud}')
30     print(f'Numero de instancias predichas correctamente: {accuracy_score(y_train, y_predict, normalize=False)} de {totalPre')
31     print('\nReporte de clasificación')
32     print(classification_report(y_train, y_predict, target_names=target_names))
33     print('\nMatriz de confusión')
34     cm = confusion_matrix(y_train, y_predict, labels = clf.classes_)
35     print(cm)
36     print('\n')
```

## Código Fuente

```

-----Gauss NB-----
emails.csv
['0', '1']
Resultados de la prediccion:
[1 0 0 ... 1 0 1]
3620
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[0. 1.]
 [1. 0.]
 [1. 0.]
 ...
 [0. 1.]
 [1. 0.]
 [0. 1.]]
Accuracy: 0.9676795580110498
Numero de instancias predichas correctamente: 3503 de 3620

Reporte de clasificación

```

	precision	recall	f1-score	support
0	1.00	0.96	0.98	2561
1	0.90	1.00	0.95	1059
accuracy			0.97	3620
macro avg	0.95	0.98	0.96	3620
weighted avg	0.97	0.97	0.97	3620

```

Matriz de confusión
[[2447 114]
 [ 3 1056]]

```

```

-----Multinomial NB-----
emails.csv
['0', '1']
Resultados de la prediccion:
[1 0 0 ... 1 0 1]
3620
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[1.47344875e-025 1.00000000e+000]
 [1.00000000e+000 1.84863298e-024]
 [1.00000000e+000 4.53379644e-016]
 ...
 [9.16541422e-192 1.00000000e+000]
 [1.00000000e+000 0.00000000e+000]
 [1.52145778e-051 1.00000000e+000]]
Accuracy: 0.9488950276243094
Numero de instancias predichas correctamente: 3435 de 3620

Reporte de clasificación

```

	precision	recall	f1-score	support
0	0.98	0.95	0.96	2561
1	0.89	0.94	0.92	1059
accuracy			0.95	3620
macro avg	0.93	0.95	0.94	3620
weighted avg	0.95	0.95	0.95	3620

```

Matriz de confusión
[[2435 126]
 [ 59 1000]]

```

# Código Fuente

```
1 for modelo in tiposModelo:
2     if modelo == 'Gauss':
3         clf = GaussianNB()
4         print('\n-----Gauss NB-----')
5     elif modelo == 'Multinomial':
6         clf = MultinomialNB()
7         print('\n-----Multinomial NB-----')
8     elif modelo == 'Bernoulli':
9         clf = BernoulliNB()
10        print('\n-----Bernoulli NB-----')
11    else:
12        print("El modelo ingresado no esta disponible")
13        break
14    clf.fit(X_train, y_train)
15    y_predict = clf.predict(X_test)
16    target_names = clf.classes_
17    print(target_names)
18    print('Resultados de la prediccion:')
19    print(y_predict)
20    totalPredicciones = y_predict.shape[0]
21    print(totalPredicciones)
22    print('Probabilidades sin la funcion logaritmo')
23    print(f'Clases: {target_names}')
24    print(clf.predict_proba(X_test))
25    exactitud = accuracy_score(y_test, y_predict)
26    exactitudesPrueba.append(exactitud)
27    print(f'Accuracy: {exactitud}')
28    print(f'Numero de instancias predichas correctamente: {accuracy_score(y_test, y_predict, normalize=False)} de {totalPred}
29    print('\nReporte de clasificación')
30    print(classification_report(y_test, y_predict, target_names=target_names))
31    print('\nMatriz de confusión')
32    print(confusion_matrix(y_test, y_predict, labels=clf.classes_))
```



## Código Fuente

```

-----Gauss NB-----
['0', '1']
Resultados de la prediccion:
[0 0 0 ... 0 1 0]
1552
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[1. 0.]
 [1. 0.]
 [1. 0.]
 ...
 [1. 0.]
 [0. 1.]
 [1. 0.]]
Accuracy: 0.9484536082474226
Numero de instancias predichas correctamente: 1472 de 1552

Reporte de clasificación

```

	precision	recall	f1-score	support
0	0.98	0.95	0.96	1111
1	0.88	0.95	0.91	441
accuracy			0.95	1552
macro avg	0.93	0.95	0.94	1552
weighted avg	0.95	0.95	0.95	1552

```

Matriz de confusión
[[1054  57]
 [ 23 418]]

```

```

-----Multinomial NB-----
['0', '1']
Resultados de la prediccion:
[0 0 0 ... 0 1 0]
1552
Probabilidades sin la funcion logaritmo
Clases: ['0', '1']
[[1.00000000e+00 5.83437844e-20]
 [1.00000000e+00 6.63692242e-34]
 [1.00000000e+00 5.28140751e-83]
 ...
 [1.00000000e+00 2.05633691e-86]
 [3.87943174e-19 1.00000000e+00]
 [1.00000000e+00 3.32394238e-20]]
Accuracy: 0.9413659793814433
Numero de instancias predichas correctamente: 1461 de 1552

Reporte de clasificación

```

	precision	recall	f1-score	support
0	0.98	0.94	0.96	1111
1	0.86	0.95	0.90	441
accuracy			0.94	1552
macro avg	0.92	0.94	0.93	1552
weighted avg	0.94	0.94	0.94	1552

```

Matriz de confusión
[[1043  68]
 [ 23 418]]

```

# Código Fuente

```
1 data = {'Dataset': nombreDataset, 'Clasificador': clasificadores, 'Accuracy Promedio - 3 pliegues': promediosExactitudesPli  
2 modelos = pd.DataFrame(data)  
3 print(modelos)
```

	Dataset	Clasificador	Accuracy Promedio - 3 pliegues	Accuracy Prueba
0	emails.csv	Gauss	0.946686	0.948454
1	emails.csv	Multinomial	0.943371	0.941366

# Código Fuente

## IRIS

```
1 df = pd.read_csv("iris.csv", sep=',', engine='python')
2 X = df.drop(df.columns[[len(df.columns)-1]],axis=1).values
3 y = df['species'].values

1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle = True, random_state=0)

1 # DIVISION EN K-PLIEGUES
2
3 # Numero de pliegues
4 n_pliegues = 3
5
6
7 validation_sets_Iris = []
8 kf = KFold(n_pliegues)
9
10
11 for train_index, test_index in kf.split(X_train):
12     print("TRAIN:", train_index, "\n", "TEST:",
13         test_index)
14     X_train_, X_test_ = X_train[train_index], X_train[test_index]
15     y_train_, y_test_ = y_train[train_index], y_train[test_index]
16     print(X_train_.shape)
17     print(X_test_.shape)
18     # print(y_train_.shape)
19     print(y_test_.shape)
20
21
22     validation_sets_Iris.append(validation_set(X_train_, y_train_, X_test_, y_test_))
23
24
25
26 my_test_set_Iris = test_set(X_test, y_test)
27
28 my_data_set_Iris = data_set(validation_sets_Iris, my_test_set_Iris)
29
```

# Código Fuente

```
TRAIN: [ 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88
89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104]
TEST: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31 32 33 34]
(70, 4)
(35, 4)
(35,)
TRAIN: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 70
71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88
89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104]
TEST: [35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
59 60 61 62 63 64 65 66 67 68 69]
(70, 4)
(35, 4)
(35,)
TRAIN: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69]
TEST: [ 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104]
(70, 4)
(35, 4)
(35,)
```

# Código Fuente

```
1  # Modelo gaussiano aplicado a los tres pliegues
2  nPliegue = 0
3  exactitudes = []
4  dataset = 'iris.csv'
5  modelo = 'Gauss'
6  for pliegue in validation_sets_Iris:
7      nPliegue = nPliegue+1
8      clf = GaussianNB()
9      clf.fit(pliegue.X_train, pliegue.y_train)
10     y_predict = clf.predict(pliegue.X_test)
11     target_names = clf.classes_
12     print(target_names)
13     # target_names = target_names.tolist()
14     print ('-----Gaussian NB-----')
15     print(f'Pliegue: {nPliegue}')
16     print('Resultados de la predicción:')
17     print (y_predict)
18     print(y_predict.shape)
19     print('Probabilidades sin la función logaritmo')
20     print(f'Clases: {target_names}')
21     print(clf.predict_proba(pliegue.X_test))
22     exactitud = accuracy_score(pliegue.y_test, y_predict)
23     exactitudes.append(exactitud)
24     print(f'Accuracy: {exactitud}')
25     print(f'Número de instancias predichas correctamente: {accuracy_score(pliegue.y_test, y_predict, normalize=False)}')
26     print('\nReporte de clasificación')
27     print(classification_report(pliegue.y_test, y_predict, target_names=target_names))
28     print('\nMatriz de confusión')
29     print(confusion_matrix(pliegue.y_test, y_predict, labels=clf.classes_))
30     print(exactitudes)
31     promedioExactitud = sum(exactitudes) / len(exactitudes)
32     promediosExactitudesPliegues.append(promedioExactitud)
33     clasificadores.append(modelo)
34     nombreDataset.append(dataset)
35     print(f"El promedio de exactitud de la predicción de los pliegues es: {promedioExactitud}")
```

# Código Fuente

```
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
-----Gaussian NB-----
Pliegue: 1
Resultados de la prediccion:
['Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor']
(35,)
Probabilidades sin la funcion logaritmo
Clases: ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
[[2.07076061e-065 1.00000000e+000 2.68556589e-011]
 [2.49637793e-266 2.31970878e-003 9.97680291e-001]
 [0.00000000e+000 2.91922772e-010 1.00000000e+000]]
```

# Código Fuente

```
[6.27599305e-224 1.48834722e-001 8.51165278e-001]
[1.23728830e-045 1.00000000e+000 6.95143282e-011]
[1.00000000e+000 9.98627751e-018 1.22468584e-033]
[2.09581902e-157 9.99817246e-001 1.82753603e-004]]
Accuracy: 0.9142857142857143
Numero de instancias predichas correctamente: 32

Reporte de clasificación
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.82	1.00	0.90	14
Iris-virginica	1.00	0.79	0.88	14
accuracy			0.91	35
macro avg	0.94	0.93	0.93	35
weighted avg	0.93	0.91	0.91	35

# Código Fuente

```
Matriz de confusión
[[ 7  0  0]
 [ 0 14  0]
 [ 0  3 11]]
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
-----Gaussian NB-----
Pliegue: 2
Resultados de la prediccion:
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor']
```



# Código Fuente

```
Accuracy: 1.0
Numero de instancias predichas correctamente: 35

Reporte de clasificación
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	16
Iris-versicolor	1.00	1.00	1.00	5
Iris-virginica	1.00	1.00	1.00	14
accuracy			1.00	35
macro avg	1.00	1.00	1.00	35
weighted avg	1.00	1.00	1.00	35

# Código Fuente

Matriz de confusión

```
[[16  0  0]
```

```
 [ 0  5  0]
```

```
 [ 0  0 14]]
```

```
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
```

```
-----Gaussian NB-----
```

Pliegue: 3

Resultados de la prediccion:

```
['Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
```

```
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
```

```
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
```

```
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
```

```
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
```

```
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
```

```
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
```

```
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
```

# Código Fuente

```
Accuracy: 0.9428571428571428
```

```
Numero de instancias predichas correctamente: 33
```

```
Reporte de clasificación
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	0.85	0.92	13
Iris-virginica	0.85	1.00	0.92	11
accuracy			0.94	35
macro avg	0.95	0.95	0.94	35
weighted avg	0.95	0.94	0.94	35

# Código Fuente

```
Matriz de confusión
[[11  0  0]
 [ 0 11  2]
 [ 0  0 11]]
[0.9142857142857143, 1.0, 0.9428571428571428]
El promedio de exactitud de la predicción de los pliegues es: 0.9523809523809522
```

# Código Fuente

```
1 # Modelo multinomial
2 nPliegue = 0
3 exactitudes = []
4 dataset = 'iris.csv'
5 modelo = 'Multinomial'
6 for pliegue in validation_sets_Iris:
7     nPliegue = nPliegue+1
8     clf = MultinomialNB()
9     clf.fit(pliegue.X_train, pliegue.y_train)
10    y_predict = clf.predict(pliegue.X_test)
11    print(target_names)
12    print('-----Multinomial NB-----')
13    print(f'Pliegue: {nPliegue}')
14    print('Resultados de la prediccion:')
15    print(y_predict)
16    print(y_predict.shape)
17    print('Probabilidades sin la funcion logaritmo')
18    print(f'Clases:\n{target_names}')
19    print(clf.predict_log_proba(pliegue.X_test))
20    exactitud = accuracy_score(pliegue.y_test, y_predict)
21    exactitudes.append(exactitud)
22    print(f'Accuracy: {exactitud}')
23    print(f'Numero de instancias predichas correctamente: {accuracy_score(pliegue.y_test, y_predict, normalize=False)}')
24    print('\nReporte de clasificación')
25    print(classification_report(pliegue.y_test, y_predict, target_names=target_names))
26    print('\nMatriz de confusión')
27    print(confusion_matrix(pliegue.y_test, y_predict, labels=clf.classes_))
28 print(exactitudes)
29 promedioExactitud = sum(exactitudes) / len(exactitudes)
30 promediosExactitudesPliegues.append(promedioExactitud)
31 nombreDataset.append(dataset)
32 clasificadores.append(modelo)
33 print(f"El promedio de exactitud de la predicción de los pliegues es: {promedioExactitud}")
```

# Código Fuente

```
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']  
-----Multinomial NB-----  
Pliegue: 1  
Resultados de la prediccion:  
['Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'  
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'  
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'  
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'  
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'  
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'  
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'  
'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'  
'Iris-virginica' 'Iris-setosa' 'Iris-virginica']  
(35,)  
Probabilidades sin la funcion logaritmo  
Clases:  
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']  
[[-2.33227409 -0.92444731 -0.68087356]  
 [-3.96966866 -0.92259406 -0.5384817 ]
```

# Código Fuente

```
[-0.32181431 -1.88405416 -2.09399092]  
[-3.80262887 -0.93628498 -0.53510767]  
[-1.84266834 -0.95137123 -0.78659074]  
[-0.29195282 -1.96068551 -2.18538593]  
[-3.34874879 -0.90821776 -0.57691296]]
```

Accuracy: 0.6

Numero de instancias predichas correctamente: 21

Reporte de clasificación

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.00	0.00	0.00	14
Iris-virginica	0.50	1.00	0.67	14
accuracy			0.60	35
macro avg	0.50	0.67	0.56	35
weighted avg	0.40	0.60	0.47	35

# Código Fuente

```
Matriz de confusión
[[ 7  0  0]
 [ 0  0 14]
 [ 0  0 14]]
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
-----Multinomial NB-----
Pliegue: 2
Resultados de la prediccion:
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
'Iris-setosa' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
'Iris-versicolor' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica']
```



# Código Fuente

```
Accuracy: 0.9428571428571428
Numero de instancias predichas correctamente: 33

Reporte de clasificación
      precision    recall  f1-score   support

   Iris-setosa      1.00      1.00      1.00        16
  Iris-versicolor  0.71      1.00      0.83         5
   Iris-virginica   1.00      0.86      0.92        14

   accuracy          0.94          35
  macro avg          0.90          35
 weighted avg          0.96          35

Matriz de confusión
[[16  0  0]
 [ 0  5  0]
 [ 0  2 12]]
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
```

# Código Fuente

```
-----Multinomial NB-----  
Pliegue: 3  
Resultados de la prediccion:  
['Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'  
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'  
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'  
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'  
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'  
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'  
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'  
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'  
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa']  
(35,)  
Probabilidades sin la funcion logaritmo  
Clases:  
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']  
[[-2.16107639 -0.95092602 -0.69632044]  
 [-4.23924677 -0.9096364 -0.53972086]  
 [-2.66765444 -0.8844638 -0.65845668]  
 ]
```

# Código Fuente

```
Accuracy: 0.6285714285714286
Numero de instancias predichas correctamente: 22

Reporte de clasificación
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	0.00	0.00	0.00	13
Iris-virginica	0.46	1.00	0.63	11
accuracy			0.63	35
macro avg	0.49	0.67	0.54	35
weighted avg	0.46	0.63	0.51	35

```
Matriz de confusión
[[11  0  0]
 [ 0  0 13]
 [ 0  0 11]]
[0.6, 0.9428571428571428, 0.6285714285714286]
El promedio de exactitud de la predicción de los pliegues es: 0.7238095238095238
```

# Código Fuente

```
1 data = {'Dataset' : nombreDataset, 'Clasificador': clasificadores, 'Accuracy Promedio - 3 pliegues': promediosExactitudesPli  
2 modelos = pd.DataFrame(data)  
3 print(modelos)
```

	Dataset	Clasificador	Accuracy Promedio - 3 pliegues
0	emails.csv	Gauss	0.946686
1	emails.csv	Multinomial	0.943371
2	iris.csv	Gauss	0.952381
3	iris.csv	Multinomial	0.723810

# Código Fuente

```
1 # Modelo multinomial aplicado a los conjuntos de entrenamiento
2 for modelo in tiposModelo:
3     if modelo == 'Gauss':
4         clf = GaussianNB()
5         print('\n-----Gauss NB-----')
6     elif modelo == 'Multinomial':
7         clf = MultinomialNB()
8         print('\n-----Multinomial NB-----')
9     elif modelo == 'Bernoulli':
10        clf = BernoulliNB()
11        print('\n-----Bernoulli NB-----')
12    else:
13        print("El modelo ingresado no esta disponible")
14        break
15    print(dataset)
16    clf.fit(X_train, y_train)
17    y_predict = clf.predict(X_train)
18    target_names = clf.classes_
19    print(target_names)
20    print('Resultados de la prediccion:')
21    print(y_predict)
22    totalPredicciones = y_predict.shape[0]
23    print(totalPredicciones)
24    print('Probabilidades sin la funcion logaritmo')
25    print(f'Clases: {target_names}')
26    print(clf.predict_proba(X_train))
27    exactitud = accuracy_score(y_train, y_predict)
28    exactitudes.append(exactitud)
29    print(f'Accuracy: {exactitud}')
30    print(f'Numero de instancias predichas correctamente: {accuracy_score(y_train, y_predict, normalize=False)} de {totalPre')
31    print('\nReporte de clasificación')
32    print(classification_report(y_train, y_predict, target_names=target_names))
33    print('\nMatriz de confusión')
34    cm = confusion_matrix(y_train, y_predict, labels = clf.classes_)
35    print(cm)
36    print('\n')
```

## Código Fuente

```

-----Gauss NB-----
iris.csv
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
Resultados de la prediccion:
['Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa']
105
Probabilidades sin la funcion logaritmo
Clases: ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']

```

```

[[4.68089903e-059 9.99999927e-001 7.25939137e-008]
 [1.01942708e-238 1.44292097e-003 9.98557079e-001]
 [1.00147106e-320 3.62956541e-010 1.00000000e+000]
 [3.91897132e-176 9.51750980e-001 4.82490196e-002]
 [1.95197123e-267 3.70855956e-004 9.99629144e-001]
 [1.53860406e-083 9.9996515e-001 3.48465545e-006]
 [0.00000000e+000 1.27697470e-010 1.00000000e+000]
 [7.18562227e-160 6.28151283e-001 3.71848717e-001]
 [3.31854219e-078 9.99998299e-001 1.70082182e-006]
 [3.11848647e-189 1.03072844e-001 8.96927156e-001]
 [4.76922202e-184 6.77225466e-001 3.22774534e-001]
 [1.48165634e-150 9.72113779e-001 2.78862210e-002]
 [8.24988714e-205 2.01245229e-002 9.79875477e-001]
 [1.61287326e-151 8.04952598e-001 1.95047402e-001]
 [1.86441183e-229 3.18171164e-004 9.99681829e-001]
 [5.55240634e-145 8.68818276e-001 1.31181724e-001]
 [1.00000000e+000 4.22757910e-015 4.04004296e-025]
 [2.37927935e-210 2.05890362e-002 9.79410964e-001]
 [1.51361596e-109 9.99869044e-001 1.30955887e-004]
 [3.44184751e-077 9.99983866e-001 1.61340656e-005]
 [3.59931244e-098 9.99966784e-001 3.32160626e-005]
 [4.44953654e-141 9.89545638e-001 1.04543623e-002]
 [4.77330165e-289 6.07888656e-006 9.99993921e-001]

```

# Código Fuente

```
[1.10794773e-124 9.99466633e-001 5.33367463e-004]
[1.00000000e+000 2.61112768e-016 6.43756862e-026]
[1.00000000e+000 4.67231531e-014 2.25625438e-023]
[1.00000000e+000 1.65488627e-016 1.36357941e-026]
[1.20011385e-246 1.53922992e-003 9.98460770e-001]
[1.16400109e-090 9.99993207e-001 6.79310091e-006]
[0.00000000e+000 2.60997905e-012 1.00000000e+000]
[1.00000000e+000 1.08119904e-016 8.91816309e-027]]
Accuracy: 0.9428571428571428
Numero de instancias predichas correctamente: 99 de 105

Reporte de clasificación
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	34
Iris-versicolor	0.91	0.91	0.91	32
Iris-virginica	0.92	0.92	0.92	39
accuracy			0.94	105
macro avg	0.94	0.94	0.94	105
weighted avg	0.94	0.94	0.94	105

```
Matriz de confusión
[[34  0  0]
 [ 0 29  3]
 [ 0  3 36]]
```

# Código Fuente

```

-----Multinomial NB-----
iris.csv
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
Resultados de la prediccion:
['Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa']
105
Probabilidades sin la funcion logaritmo
Clases: ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
[[0.07343555 0.43453702 0.49202743]
 [0.0138303 0.42933321 0.55683649]
 [0.00548736 0.40085413 0.59365851]
 [0.01709838 0.42196131 0.55894031]
 [0.00852363 0.41473995 0.57673642]
 [0.07239751 0.44557955 0.48202295]
 [0.00558533 0.41482048 0.58039419]]

```

```

[0.00665064 0.4286539 0.56469546]
[0.69093452 0.1823274 0.12673808]]
Accuracy: 0.7047619047619048
Numero de instancias predichas correctamente: 74 de 105

Reporte de clasificación

```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	34
Iris-versicolor	1.00	0.03	0.06	32
Iris-virginica	0.56	1.00	0.72	39
accuracy			0.70	105
macro avg	0.85	0.68	0.59	105
weighted avg	0.84	0.70	0.61	105

```

Matriz de confusión
[[34 0 0]
 [ 0 1 31]
 [ 0 0 39]]

```



# Código Fuente

```
1 for modelo in tiposModelo:
2     if modelo == 'Gauss':
3         clf = GaussianNB()
4         print ('\n-----Gauss NB-----')
5     elif modelo == 'Multinomial':
6         clf = MultinomialNB()
7         print ('\n-----Multinomial NB-----')
8     elif modelo == 'Bernoulli':
9         clf = BernoulliNB()
10        print ('\n-----Bernoulli NB-----')
11    else:
12        print("El modelo ingresado no esta disponible")
13        break
14    clf.fit(X_train, y_train) #Entrenamiento
15    y_predict = clf.predict(X_test)
16    target_names = clf.classes_
17    print(target_names)
18    print('Resultados de la prediccion:')
19    print(y_predict)
20    totalPredicciones = y_predict.shape[0]
21    print(totalPredicciones)
22    print('Probabilidades sin la funcion logaritmo')
23    print(f'Clases: {target_names}')
24    print(clf.predict_proba(X_test))
25    exactitud = accuracy_score(y_test, y_predict)
26    exactitudesPrueba.append(exactitud)
27    print(f'Accuracy: {exactitud}')
28    print(f'Numero de instancias predichas correctamente: {accuracy_score(y_test, y_predict, normalize=False)} de {totalPred}
29    print('\nReporte de clasificación')
30    print(classification_report(y_test, y_predict, target_names=target_names))
31    print('\nMatriz de confusión')
32    print (confusion_matrix(y_test, y_predict, labels=clf.classes_))
```

## Código Fuente

```

-----Gauss NB-----
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
Resultados de la prediccion:
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'
'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'
'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'
'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa']
45
Probabilidades sin la funcion logaritmo
Clases: ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
[[1.62002265e-255 1.75690913e-006 9.99998243e-001]
 [2.06951720e-086 9.99997509e-001 2.49062165e-006]
 [1.00000000e+000 5.94637347e-018 8.27382232e-027]
 [2.82654956e-320 1.10744521e-006 9.99998893e-001]
 [1.00000000e+000 3.21629260e-016 5.39931785e-026]
 [0.00000000e+000 1.46019193e-010 1.00000000e+000]
 [1.00000000e+000 4.47326904e-016 7.93379918e-026]

```

```

[1.00000000e+000 3.05868307e-016 1.38434323e-025]]
Accuracy: 1.0
Numero de instancias predichas correctamente: 45 de 45

Reporte de clasificación

```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	16
Iris-versicolor	1.00	1.00	1.00	18
Iris-virginica	1.00	1.00	1.00	11
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

```

Matriz de confusión
[[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]

```

# Código Fuente

```
-----Multinomial NB-----
['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
Resultados de la prediccion:
['Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-setosa'
'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
'Iris-setosa']
45
Probabilidades sin la funcion logaritmo
Clases: ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
[[0.00605335 0.38845624 0.60549041]
 [0.07045758 0.449108 0.48043443]
 [0.82830341 0.10892771 0.06276888]
 [0.00847099 0.42650739 0.56502162]]
```

```
[0.76608036 0.14428317 0.08963647]]
Accuracy: 0.6
Numero de instancias predichas correctamente: 27 de 45

Reporte de clasificación
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	16
Iris-versicolor	0.00	0.00	0.00	18
Iris-virginica	0.38	1.00	0.55	11
accuracy			0.60	45
macro avg	0.46	0.67	0.52	45
weighted avg	0.45	0.60	0.49	45

```
Matriz de confusión
[[16 0 0]
 [ 0 0 18]
 [ 0 0 11]]
```

# Código Fuente

```
1 data = {'Dataset' : nombreDataset, 'Clasificador': clasificadores, 'Accuracy Promedio - 3 pliegues': promediosExactitudesPli  
2 modelos = pd.DataFrame(data)  
3 print(modelos)
```

	Dataset	Clasificador	Accuracy Promedio - 3 pliegues	Accuracy Prueba
0	emails.csv	Gauss	0.946686	0.948454
1	emails.csv	Multinomial	0.943371	0.941366
2	iris.csv	Gauss	0.952381	1.000000
3	iris.csv	Multinomial	0.723810	0.600000