



VERDAD, BELLEZA, PROBIDAD

UAT



Facultad de Ingeniería  
Arturo Narro Siller

# Practicas

Luis Enrique Borjas Mercado

Javier M. Vázquez García

Profesor: **Dr. García Ruiz Alejandro**

**Humberto**

Asignatura: **Diseño Electrónico Basado en**

**Sistemas**

**Embebidos**

**8vo. Semestre – Grupo “G”**

**2022-1**

## Índice

Índice .....	2
Práctica 1. KNN.....	3
Práctica 2. Genético.....	12
Práctica 3. ID3. ....	19
Práctica 4. Asociador Lineal.....	29
Práctica 5. Discretizador EWB .....	35
Práctica 6. Naive Bayes .....	37

## Práctica 1. KNN

**Descripción:** Realizar la aplicación KNN para el calculo del rendimiento de la técnica utilizando la instancia WINE.

### Introducción:

Dentro de los conceptos necesarios para el entendimiento de la practica realizada están los siguientes:

1. **MultiMode:** La función statistics. multimode devuelve una lista formada por los valores más frecuentes (la moda) de la data.
2. **Instancia Wine:** Dataset con datos con punto decimal.

### Desarrollo:

Código utilizado

```

1  def Euclidiana(A, B):
2      distancia = 0
3      for i in range(len(A)):
4          distancia += (A[i]-B[i])**2
5      distancia = distancia ** (1/2)
6      distancia = round(distancia, 2)
7      return distancia
8
9  def Diferencia(A,B):
10     distancia=0
11     for i in range(len(A)):
12         distancia += abs((A[i] - B[i]))
13     distancia=round(distancia/len(A), 2)
14     return distancia
15
16 def Canberra (A,B):
17     distancia=0
18     for i in range(len(A)):
19         distancia+=(abs(A[i]-B[i]))/abs((A[i]+B[i]))
20     return round(distancia, 2)
21
22 def Manhattan(A,B):
23     distancia=0
24     for i in range(len(A)):
25         distancia+=abs(A[i]-B[i])
26     return distancia
27
28 def Coseno(A,B):
29     distancia = 0
30     arriba = 0
31     abajox = 0
32     abajoy = 0
33     for i in range(len(A)):
34         arriba += A[i] + B[i]
35         abajox += A[i] ** 2
36         abajoy += B[i] ** 2
37     distancia = arriba/((abajox * abajoy) ** (1/2))
38     return round(distancia, 2)

```

```

36     ###CARGAR INSTANCIA DE ENTRENAMIENTO
37
38     archivo = open("wine_training98.0.csv", "r")
39     contenido = archivo.readlines()
40
41     #VISUALIZA EL CONTENIDO DEL ARCHIVO
42     print('\nArchivo Completo: ') #Impreso línea a línea
43     for l in contenido:
44         print(l, end="") #por el formato en que se lee el archivo se quita el terminador
45     print("\n\n")
46
47
48     lista = [linea.split(",") for linea in contenido]
49
50     #VISUALIZA LISTA PROCESADA
51     print("Lista de listas separadas por comas: ")
52     #Impreso línea a línea
53     for l in lista:
54         print(l)
55     print("\n\n")
56
57     #CONVIERTE LA LISTA DE LISTAS EN LA INSTANCIA NECESARIA PARA TRABAJAR CON EL KNN
58     instancia = [_list(map(float, x[:13])), x[13]] for x in lista] #iris
59
60     print("Total de datos de la instancia", len(instancia))
61
62     print("Instancia de entrenamiento:")
63     #VISUALIZA EL CONTENIDO DEL ARCHIVO
64     #Impreso línea a línea
65     for l in instancia:
66         print(l)
67     print("\n\n")
68
69
70     #####

```

```

70 #####
71 ###CARGAR INSTANCIA DE PRUEBA
72
73 archivo = open("wine_test90.0.csv","r") #ABRE EL ARCHIVO
74 contenido = archivo.readlines() #LEE TOFO EL CONTENIDO DEL ARCHIVO
75
76 #VISUALIZA EL CONTENIDO DEL ARCHIVO
77 print('\nArchivo Completo: ') #Impreso línea a línea
78 for l in contenido:
79     print(l, end="") #por el formato en que se lee el archivo se quita el termin
80 print("\n\n")
81
82 #CREA UNA LISTA EN LA QUE CADA ELEMENTO SEA UNA LINEA DEL ARCHIVO CONVERTIDA EN
83 lista = [linea.split(",") for linea in contenido]
84
85 #VISUALIZA LISTA PROCESADA
86 print("Lista de listas separadas por comas: ")
87 #Impreso línea a línea
88 for l in lista:
89     print(l)
90 print("\n\n")
91
92 #CONVIERTE LA LISTA DE LISTAS EN LA INSTANCIA NECESARIA PARA TRABAJAR CON EL KNN
93 prueba = [ [ list(map(float,x[:13])), x[13] ) for x in lista ] #iris
94
95 print("Total de datos de la Instancia",len(prueba))
96
97 #VISUALIZA EL CONTENIDO DEL ARCHIVO
98 print("Instancia de prueba:")
99 #Impreso línea a línea
100 for l in prueba:
101     print(l)
102 print("\n\n")
103
104 #####

```

```

105     ###DEFINIR EL VALOR DE "K" - Un número entre 1 y el total de registros de la instancia (entrenamiento)
106     contAciertos = 0
107     aux = 0
108     cantK = 0
109     cantAci = 0
110     for K in range(1, int(len(instancia)/2)):
111         contAciertos=0
112         for registroNC in prueba: # para recorrer a todos los registros de prueba y aplicar al algoritmo K-
113             print("Clasificación del registro: ")
114             print(registroNC) # registro de prueba procesado para su clasificacion
115
116             NC = registroNC[0] # vector de características del registro actual de prueba
117
118             estructuraDatos = {} # inicializacion de la estructura de datos
119
120             for NoCaso, i in enumerate(instancia): #por cada elemento/registro de la instancia
121                 distancia_NC_i = Canberra(NC, i[0]) #registro[0] = vector carac -- registro[1] = clase
122                 # print(distancia_NC_i)
123                 estructuraDatos[NoCaso] = distancia_NC_i
124
125             # print(estructuraDatos) # La distancia de los registros con el registroNC
126
127             ## 0 = NoCaso 1 = Distancia --> #retorna una lista de tuplas
128             ordenado = sorted(estructuraDatos.items(), key=lambda x: x[1]) #reverse=True # ordena los regis
129             # de menor a mayor de acuerdo con la distancia con el registroNC
130             # print(ordenado)
131
132             temporalK = []
133             for i in range(K):
134                 NoCaso = ordenado[i][0] ##0 = NumDeCaso
135                 # print(etiqueta)
136                 # registro correspondiente al indice (NoCaso) consultado
137                 registro = instancia[NoCaso]
138                 # print(registro)
139                 temporalK.append(registro[1]) # obtencion de la etiqueta

```

```

140
141     print("Clases de los vectores más cercanos al registro NC:")
142     print(temporalK) # los primeros K vectores
143     print("\n\n")
144
145     from statistics import \
146     | multimode # <<<- realizado unicamente para fines academicos, no se recomien
147
148     moda = multimode(temporalK)
149     respKnn = moda[0] # si existe más de una moda se queda con la primera de ellas
150
151     print("Clase asignada por el KNN: " + str(respKnn))
152     print("Clase Real: " + registroNC[1])
153
154     if str(respKnn) == registroNC[1]:
155         | contAciertos += 1
156
157     rendimiento = contAciertos / len(prueba) * 100
158
159     if rendimiento >= aux:
160         aux = rendimiento
161         cantK = K
162         | contAciertos
163
164     print("La mejor K: " + str(cantK))
165     print("Total de aciertos: " + str(cantAci))
166     print("Total de pruebas: " + str(len(prueba)))
167     print("Rendimiento: " + str(aux))
168
169     #K = 3
170     #####
171
172     #contAciertos = 0 #contador de aciertos obtenidos en la clasificación
173

```

### Explicación

1. Se definieron las medidas de similitud con las cuales se hicieron las pruebas del mejor rendimiento (líneas 1-34).
2. Se carga la instancia de entrenamiento (líneas 38-39).
3. Se visualiza el contenido del archivo (líneas 42-45).
4. Crea una lista en la que cada elemento sea una línea del archivo convertida en lista separada por coma (línea 48).
5. Visualiza lista procesada (líneas 51-55).
6. Convierte la lista de listas en la instancia necesaria para trabajar con el KNN (línea 58).
7. Visualiza el total de datos de la instancia (línea 60).
8. Visualiza el contenido del archivo línea a línea (líneas 65-67).
9. Carga la instancia de prueba (líneas 73-74).
10. Visualiza el contenido del archivo línea a línea (líneas 77-80).
11. Crea una lista en la que cada elemento sea una línea del archivo convertida en lista separada por coma (línea 83).
12. Visualiza la lista procesada línea a línea (líneas 86-90).

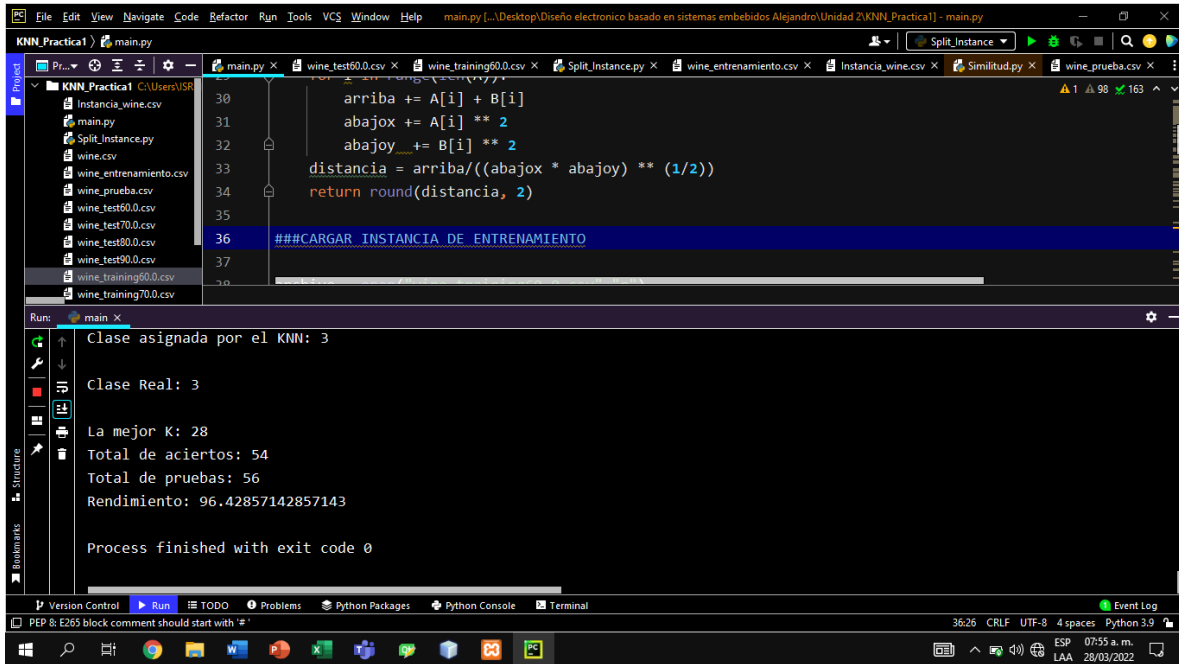


13. Convierte las listas de listas en la prueba necesaria para trabajar con el KNN (línea 93).
14. Visualiza el total de datos de la instancia de prueba (línea 95).
15. Visualiza el contenido del archivo línea a línea (líneas 98-102).
16. Se declaran los contadores de aciertos, un auxiliar que se utilizara más adelante, una cantidad de k's y cantidad de aciertos (líneas 106-109).
17. Se recorren las k (líneas 110-167).
18. Contador de aciertos obtenidos en la clasificación (línea 111).
19. Recorrerá a todos los registros de prueba y aplicar el KNN (líneas 112-155).
20. Visualiza el registro de prueba procesado para su clasificación (líneas 113-114).
21. Inicialización de un vector de características del registro actual de prueba (línea 116).
22. Inicialización de la estructura de datos (línea 118).
23. Se recorre por cada elemento/registro de la instancia (líneas 120-123).
24. Inicialización de la distancia Canberra (línea 121).
25. Inicialización de las distancias de los registros con el numero de caso (línea 123).
26. Ordenamiento de los registros de menor a mayor de acuerdo con las distancias con el registroNC (línea 128).
27. Se crea un temporal para k, para la obtención de las etiquetas (línea 132).
28. Recorrido de k para la obtención de etiqueta (líneas 133-139).
29. Registro correspondiente al índice consultado (línea 137).
30. Obtención de la etiqueta (línea 139).
31. Visualización de los primeros k vectores (línea 142).
32. Importación de multimodal para la obtención de la moda (líneas 145-146).
33. Obtención de la moda de temporalK (línea 148).
34. Si existe más de una moda se queda con la primera de ellas (línea 149).
35. Visualización de la clase asignada por el KNN y la clase real (líneas 151-152).
36. Si la clase asignada es igual a la clase real, el contador de aciertos aumenta en 1 (líneas 154-155).
37. Inicialización del rendimiento donde el contador de aciertos se divide por el producto de el tamaño de la prueba por cien (línea 157).
38. Si el rendimiento es mayor o igual a auxiliar, el auxiliar toma el valor del rendimiento, la cantidad de k toma el valor de k y la cantidad de aciertos tomara el valor del contador de aciertos (líneas 159-162).
39. Visualización de la mejor k, total de aciertos, total de pruebas y cual fue el rendimiento (líneas 164-167).

### **Resultados:**

Realizamos las pruebas pertinentes con las pruebas al 60%, 70%, 80% y al 90%.

### **Prueba al 60%:**



```

30     arriba += A[i] + B[i]
31     abajox += A[i] ** 2
32     abajoy += B[i] ** 2
33     distancia = arriba/((abajox * abajoy) ** (1/2))
34     return round(distancia, 2)
35
36 ###CARGAR INSTANCIA DE ENTRENAMIENTO
37

```

Run: main X

```

Clase asignada por el KNN: 3

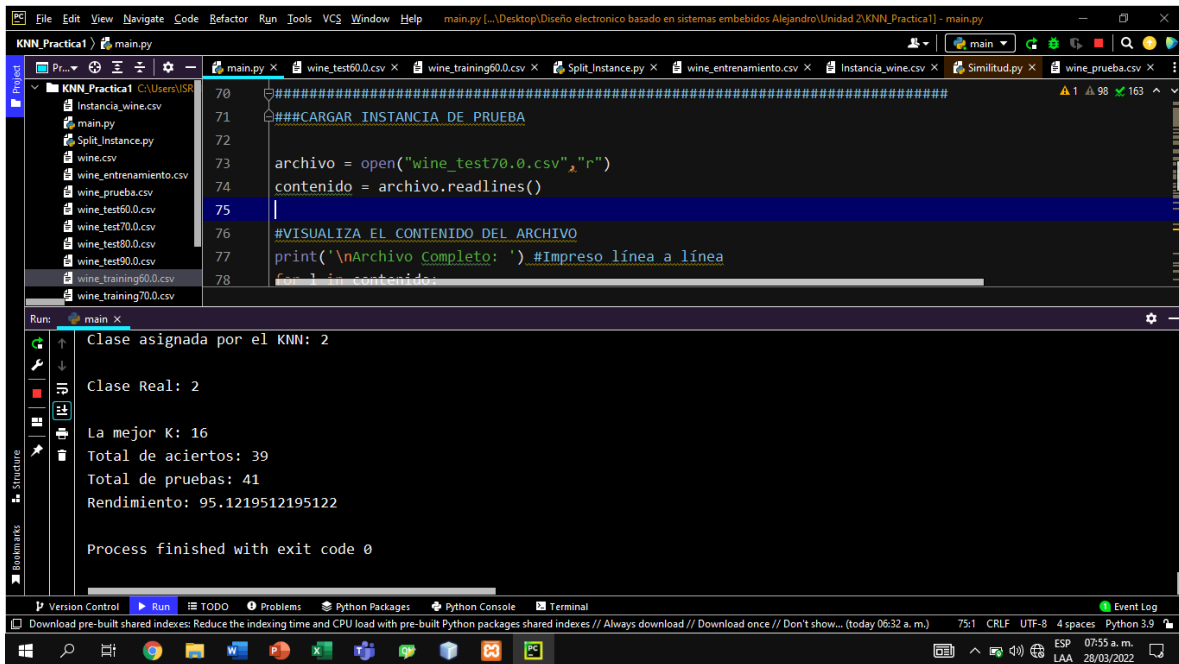
Clase Real: 3

La mejor K: 28
Total de aciertos: 54
Total de pruebas: 56
Rendimiento: 96.42857142857143

Process finished with exit code 0

```

## Prueba al 70%:



```

70 #####CARGAR INSTANCIA DE PRUEBA
71
72
73 archivo = open("wine_test70.0.csv", "r")
74 contenido = archivo.readlines()
75
76 #VISUALIZA EL CONTENIDO DEL ARCHIVO
77 print('\nArchivo Completo: ') #Impreso línea a línea
78 for i in contenido:

```

Run: main X

```

Clase asignada por el KNN: 2

Clase Real: 2

La mejor K: 16
Total de aciertos: 39
Total de pruebas: 41
Rendimiento: 95.1219512195122

Process finished with exit code 0

```

## Prueba al 80%:

```

70 #####
71 ###CARGAR INSTANCIA DE PRUEBA
72
73 archivo = open("wine_test70.0.csv", "r")
74 contenido = archivo.readlines()
75
76 #VISUALIZA EL CONTENIDO DEL ARCHIVO
77 print('\nArchivo Completo: ') #Impreso línea a línea
78 for l in contenido:

```

Run: main X

```

Clase asignada por el KNN: 2

Clase Real: 2

La mejor K: 16
Total de aciertos: 39
Total de pruebas: 41
Rendimiento: 95.1219512195122

Process finished with exit code 0

```

## Prueba al 90%:

```

70 #####
71 ###CARGAR INSTANCIA DE PRUEBA
72
73 archivo = open("wine_test70.0.csv", "r")
74 contenido = archivo.readlines()
75
76 #VISUALIZA EL CONTENIDO DEL ARCHIVO
77 print('\nArchivo Completo: ') #Impreso línea a línea
78 for l in contenido:

```

Run: main X

```

Clase asignada por el KNN: 2

Clase Real: 2

La mejor K: 16
Total de aciertos: 39
Total de pruebas: 41
Rendimiento: 95.1219512195122

Process finished with exit code 0

```

## Conclusiones:

La utilización de algoritmos de clasificación incremental partiendo de un conjunto de datos en este caso nuestro conjunto de datos de prueba y un conjunto de datos de entrenamiento, utilizando diferentes fórmulas de similitud para la obtención de la distancia, hemos obtenido un mayor rendimiento acorde a la formula de similitud de Canberra en las cuatro pruebas realizadas.

## Práctica 2. Genético.

**Descripción:** Desarrollo de un algoritmo genético para optimización de problemas de resolución.

**Introducción:**

Dentro de los conceptos necesarios para el entendimiento de la practica realizada están los siguientes:

**Desarrollo:**

Código utilizado

```
1
2  def calc_F0(indv):
3      sum = 0.0
4      for i in range(1, len(indv)):
5          sum += indv[i] ** 2
6      return indv[0] + pow(10, 6) * sum
7
8  ## numero de genes
9  tot_genes = 10
10
11  #n = numero de vectores
12  tot_individuos = 120 #numero de individuos
13
14  #Poblacion Inicial
15  import random as rnd
16  poblacion = []
17  for i in range(tot_individuos):
18      vector = [rnd.uniform(-10,10) for i in range(tot_genes)]
19      ##          vector , F0
20      poblacion.append([vector, calc_F0(vector)])
21
22  #print("Poblacion Inicial: ")
23  #for indv in poblacion:
24  #    print(indv)
25
26  it = 1
27  mejorActual = 1000000000000
28  while it<=100:
29      print("Iteracion : ", it)
30      it+=1
31
32      padres = []
33      tot_padres = 30
34
35      poblacion.sort(key= lambda x:x[1], reverse=False)
36      #sorted(poblacion, key= lambda
```

```

38     if poblacion[0][1] <= mejorActual:
39         mejorActual = poblacion[0][1]
40
41     #print("Poblacion Ordenada: ")
42     #for indiv in poblacion:
43         # print(indv)
44
45     #Me quedo con los MejoresPadres
46     poblacion = poblacion[0:tot_individuos-tot_padres]
47     #print("tot poblacion de mejores: ", len(poblacion))
48
49     ##Seleccion de los padres que seran cruzados
50     for i in range(tot_padres):
51         indexPadre1 = rnd.randint(0, tot_padres-1) #aleatorio entre 0 y n-1
52         indexPadre2 = rnd.randint(0, tot_padres-1) #aleatorio entre 0 y n-1
53         while(indexPadre1==indexPadre2):
54             indexPadre2 = rnd.randint(0, tot_padres - 1) # aleatorio entre 0 y n-1
55
56         tempPadre1 = poblacion[indexPadre1]
57         tempPadre2 = poblacion[indexPadre2]
58
59         #print(tempPadre1)
60         #print(tempPadre2)
61
62         if tempPadre1[1] <= tempPadre2[1]:
63             padres.append(tempPadre1[0].copy())
64         else:
65             padres.append(tempPadre2[0].copy())
66
67     #print("Padres para cruza: ")
68     #for index, padre in enumerate(padres):
69         # print(index,".-", padre)
70
71     hijos = []
72     for i in range(0,tot_padres, 2):

```

```

73     tempPadre1 = padres[i]
74     tempPadre2 = padres[i+1]
75
76     #Generar un aleatorio
77     puntoCruza = rnd.randint(0, tot_genes-1)
78
79     #La primera parte del padre 1, será la primera parte del hijo 1,
80     # la segunda parte del padre 1, será la segunda parte del hijo 2
81
82     # La primera parte del padre 2, será la primera parte del hijo 2,
83     # la segunda parte del padre 2, será la segunda parte del hijo 1
84
85     # Generar un aleatorio
86     puntoCruza += 1 # +1 -> puntoCruza incluyente
87     hijo1 = tempPadre1[:puntoCruza] + tempPadre2[puntoCruza:]
88     hijo2 = tempPadre2[:puntoCruza] + tempPadre1[puntoCruza:]
89
90     hijos.append([hijo1, 0])
91     hijos.append([hijo2, 0])
92
93
94     #Mutacion
95     probMuta = 0.9
96     for indexHijo in range(len(hijos)):
97         hijo = hijos[indexHijo][0]
98
99         for indexGen in range(len(hijo)):
100             r = rnd.random() # 0 - 1
101             if r >= probMuta:
102                 # se efectua la mutacion
103                 if rnd.random() >= 0.5:
104                     val = 0.5
105                 else:
106                     val = 1
107
108             # print(val)

```

```

109
110         # if hijo[indexGen] != val:
111         #     pass
112
113         hijo[indexGen] = hijo[indexGen] * val
114
115         hijos[indexHijo][1] = calc_F0(hijo)
116
117     ##poblacion completa
118     #mejores individuos + hijos
119     poblacion += hijos
120
121
122     #print("Nueva Poblacion: ")
123     #for indv in poblacion:
124     #    print(indv)
125
126     print("Mejor Solucion Actual:", mejorActual)
127

```

Explicación:

1. Se define la función objetivo, donde se recorren los individuos para obtener sus cuadrados (líneas 2-6).
2. Inicialización de total de genes con un valor a 10 (línea 9).
3. Inicialización del total de individuos con un valor de 120 (línea 12).
4. Inicialización de la lista de población (línea 16).
5. Recorrimiento del total de individuos (líneas 17-20).
6. Inicialización del vector donde se le asignan valores de entre un rango -10 a 10 de manera random (línea 18).
7. Asignación a la lista de población los valores del vector y la función objetivo (línea 20).
8. Inicialización de las iteraciones con un valor a 1 (línea 26).
9. Inicialización del mejor actual con un valor a 10000000000000 (línea 27).
10. Ciclo donde mientras la iteración sea menor o igual a 100 (líneas 28-126).
11. Visualización de las iteraciones (línea 29).
12. Inicialización de la lista de padres (línea 32).
13. Inicialización del total de padres con un valor de 30 (línea 33).
14. Ordenamiento de menor a mayor de la población (línea 35).
15. Si la población es menor o igual al mejor actual, entonces mejor actual tomara el valor de la población (líneas 38-39).
16. Tomando la lista de población se le asignara la diferencia del valor del total de individuos y el total de padres (línea 46).
17. Recorrimiento del total de padres (líneas 50-65).
18. Inicialización del índice del primer padre con un valor de manera aleatoria de entre 0 a total de padres menos uno (línea 51).
19. Inicialización del índice del segundo padre con un valor de manera aleatoria de entre 0 a total de padres menos uno (línea 52).

20. Siempre y cuando el índice del primer padre sea igual al índice del segundo padre, el índice del segundo padre tomara un valor de manera aleatoria en un rango de cero a el total de padres menos uno (líneas 53-54).
21. Inicialización del temporal del primer padre con el valor de la lista de población con el valor del índice del primer padre (línea 56).
22. Inicialización del temporal del segundo padre con el valor de la lista de población con el valor del índice del segundo padre (línea 57).
23. Si el temporal del primer padre es menor o igual al temporal del segundo padre, la lista de padres agregara la copia del temporal del primer padre, sino agregara el temporal del segundo padre (líneas 62-65).
24. Inicialización de la lista de hijos (línea 71).
25. Recorrimiento del total de padres partiendo de cero y será de dos en dos (líneas 72-74).
26. El temporal del primer padre tomara el valor de padre con el valor de posición del recorrimiento (línea 73).
27. El temporal del segundo padre tomara el valor de padre con el valor de posición del recorrimiento (línea 74).
28. Se genera el punto de cruza con un aleatorio con un rango de cero al total de genes menos uno (línea 77).
29. El punto de cruza ir incrementando en uno de manera incluyente (línea 86).
30. Inicialización de la primera parte del padre uno, será la primera parte del hijo uno y la segunda parte del padre uno será, la segunda parte del hijo dos (línea 87).
31. Inicialización de la segunda parte del padre dos, será la primera parte del hijo dos y la segunda parte del padre dos será, la segunda parte del hijo uno (línea 88).
32. La lista de hijos agregara a hijos uno y cero (línea 90).
33. La lista de hijos agregara a hijos dos y cero (línea 91).
34. Inicialización de la mutación con un valor de punto nueve (línea 95).
35. Recorrimiento del índice de hijos en un rango de la longitud de hijos (líneas 96-115).
36. Inicialización de la lista de hijo donde toma el valor de hijos en el cual estará el valor de la lista de índice de hijo en la primera lista (línea 97).
37. Recorrimiento del índice de genes en un rango de longitud de hijo (líneas 99-113).
38. Inicialización de r donde será random su valor con un rango de cero y uno (línea 100).
39. Si el random es mayor o igual a la prueba de mutación se efectúa la mutación (líneas 101-113).
40. Si el random es mayor o igual punto cinco, entonces el valor será igual a punto cinco, si no será igual a 1 (líneas 103-106).
41. La lista de hijo en el índice de genes será igual a la lista de hijo en el índice de genes por el valor acorde a la condición (línea 113).

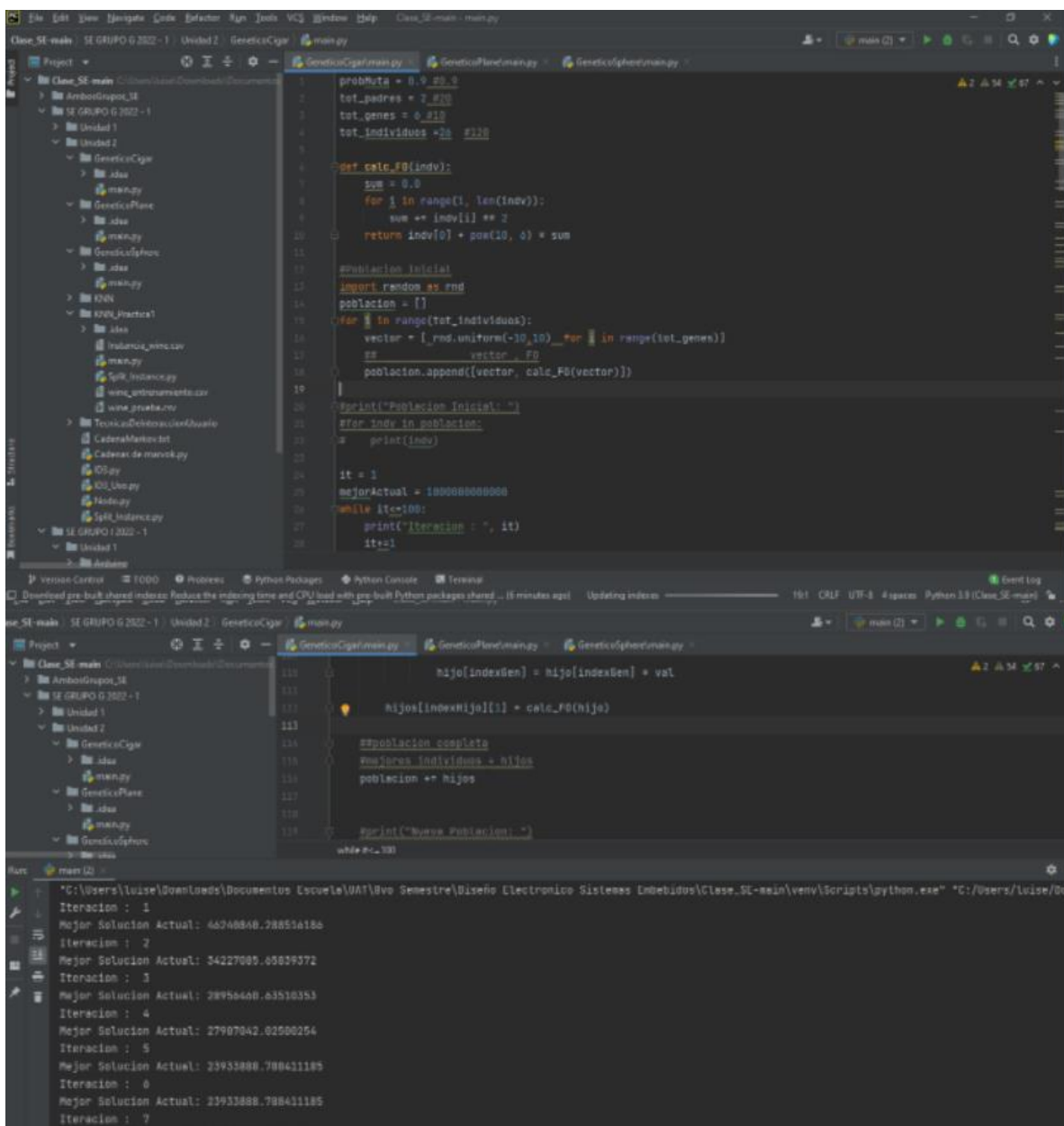


42. La lista hijos en el índice de hijos en la segunda lista será igual al cálculo de la función objetivo de hijo (línea 115).
43. La lista de población será igual a población más hijos (línea 119).
44. Visualización de la mejor solución actual (línea 126).

### Resultados:

Realizamos las pruebas pertinentes tratando de que se equivocara el algoritmo y después diera un resultado correcto.

### Código donde es un resultado erróneo.



```

1  probMuta = 0.9 #0.9
2  tot_padres = 2 #10
3  tot_genes = 0.010
4  tot_individuos = 20 #110
5
6  def calc_FB(indv):
7      sum = 0.0
8      for i in range(1, len(indv)):
9          sum += indv[i] ** 2
10     return indv[0] + pow(10, 4) * sum
11
12 #Poblacion inicial
13 import random as rnd
14 poblacion = []
15
16 for i in range(tot_individuos):
17     vector = [rnd.uniform(-10, 10) for i in range(tot_genes)]
18     ss = vector, FB
19     poblacion.append((vector, calc_FB(vector)))
20
21 #Print("Poblacion Inicial: ")
22 for indv in poblacion:
23     # print(indv)
24
25 it = 1
26 mejorActual = 1000000000000
27 while it < 100:
28     print("Iteracion : ", it)
29     it += 1
30
31     #Cálculo de la función objetivo de los hijos
32     hijos = []
33     for i in range(tot_padres):
34         #Cálculo de la función objetivo de los hijos
35         hijo = []
36         for j in range(tot_genes):
37             hijo.append(poblacion[i][0][j] * probMuta)
38         hijo.append(calc_FB(hijo))
39         hijos.append(hijo)
40
41     #Poblacion completa
42     #Hijos + Poblacion
43     poblacion += hijos
44
45     #Print("Nueva Poblacion: ")
46
47     #Mejor solución actual
48     mejorActual = min(poblacion, key=lambda x: x[1])[1]
49
50     #Print("Mejor Solucion Actual: ", mejorActual)
51
52     #Print("Iteracion : ", it)
53
54     #Print("Poblacion: ")
55     for indv in poblacion:
56         # print(indv)
57
58     #Print("Fin de la iteración")
59
60     #Print("Fin del algoritmo")
61
62     #Print("Mejor Solucion Actual: ", mejorActual)
63
64     #Print("Fin del algoritmo")
65
66     #Print("Fin del algoritmo")
67
68     #Print("Fin del algoritmo")
69
70     #Print("Fin del algoritmo")
71
72     #Print("Fin del algoritmo")
73
74     #Print("Fin del algoritmo")
75
76     #Print("Fin del algoritmo")
77
78     #Print("Fin del algoritmo")
79
80     #Print("Fin del algoritmo")
81
82     #Print("Fin del algoritmo")
83
84     #Print("Fin del algoritmo")
85
86     #Print("Fin del algoritmo")
87
88     #Print("Fin del algoritmo")
89
90     #Print("Fin del algoritmo")
91
92     #Print("Fin del algoritmo")
93
94     #Print("Fin del algoritmo")
95
96     #Print("Fin del algoritmo")
97
98     #Print("Fin del algoritmo")
99
100    #Print("Fin del algoritmo")
101
102    #Print("Fin del algoritmo")
103
104    #Print("Fin del algoritmo")
105
106    #Print("Fin del algoritmo")
107
108    #Print("Fin del algoritmo")
109
110    #Print("Fin del algoritmo")
111
112    #Print("Fin del algoritmo")
113
114    #Print("Fin del algoritmo")
115
116    #Print("Fin del algoritmo")
117
118    #Print("Fin del algoritmo")
119
120    #Print("Fin del algoritmo")
121
122    #Print("Fin del algoritmo")
123
124    #Print("Fin del algoritmo")
125
126    #Print("Fin del algoritmo")
127
128    #Print("Fin del algoritmo")
129
130    #Print("Fin del algoritmo")
131
132    #Print("Fin del algoritmo")
133
134    #Print("Fin del algoritmo")
135
136    #Print("Fin del algoritmo")
137
138    #Print("Fin del algoritmo")
139
140    #Print("Fin del algoritmo")
141
142    #Print("Fin del algoritmo")
143
144    #Print("Fin del algoritmo")
145
146    #Print("Fin del algoritmo")
147
148    #Print("Fin del algoritmo")
149
150    #Print("Fin del algoritmo")
151
152    #Print("Fin del algoritmo")
153
154    #Print("Fin del algoritmo")
155
156    #Print("Fin del algoritmo")
157
158    #Print("Fin del algoritmo")
159
160    #Print("Fin del algoritmo")
161
162    #Print("Fin del algoritmo")
163
164    #Print("Fin del algoritmo")
165
166    #Print("Fin del algoritmo")
167
168    #Print("Fin del algoritmo")
169
170    #Print("Fin del algoritmo")
171
172    #Print("Fin del algoritmo")
173
174    #Print("Fin del algoritmo")
175
176    #Print("Fin del algoritmo")
177
178    #Print("Fin del algoritmo")
179
180    #Print("Fin del algoritmo")
181
182    #Print("Fin del algoritmo")
183
184    #Print("Fin del algoritmo")
185
186    #Print("Fin del algoritmo")
187
188    #Print("Fin del algoritmo")
189
190    #Print("Fin del algoritmo")
191
192    #Print("Fin del algoritmo")
193
194    #Print("Fin del algoritmo")
195
196    #Print("Fin del algoritmo")
197
198    #Print("Fin del algoritmo")
199
200    #Print("Fin del algoritmo")
201
202    #Print("Fin del algoritmo")
203
204    #Print("Fin del algoritmo")
205
206    #Print("Fin del algoritmo")
207
208    #Print("Fin del algoritmo")
209
209

```

### Código donde el resultado es correcto.

```
main.py x Unidad 2\GeneticoCigar\main.py x Unidad 2\GeneticoSphere\main.py x
Download grammar and spelling checker for Spanish?

6   tot_genes = 10
7
8   #n = numero de vectores
9   tot_individuos = 120 #numero de individuos
10
11  #Poblacion Inicial
12  import random as rnd
13  poblacion = []
14  for i in range(tot_individuos):
15      vector = [_rnd.uniform(-10,10) for i in range(tot_genes)]
16      ##          vector , FO
17      poblacion.append([vector, calc_FO(vector), abs(calc_FO(vector))])
18
19  #print("Poblacion Inicial: ")
20  #for indv in poblacion:
21  #    print(indv)
22
23  it = 1
24  mejorActual = 11
25  while it<=100:

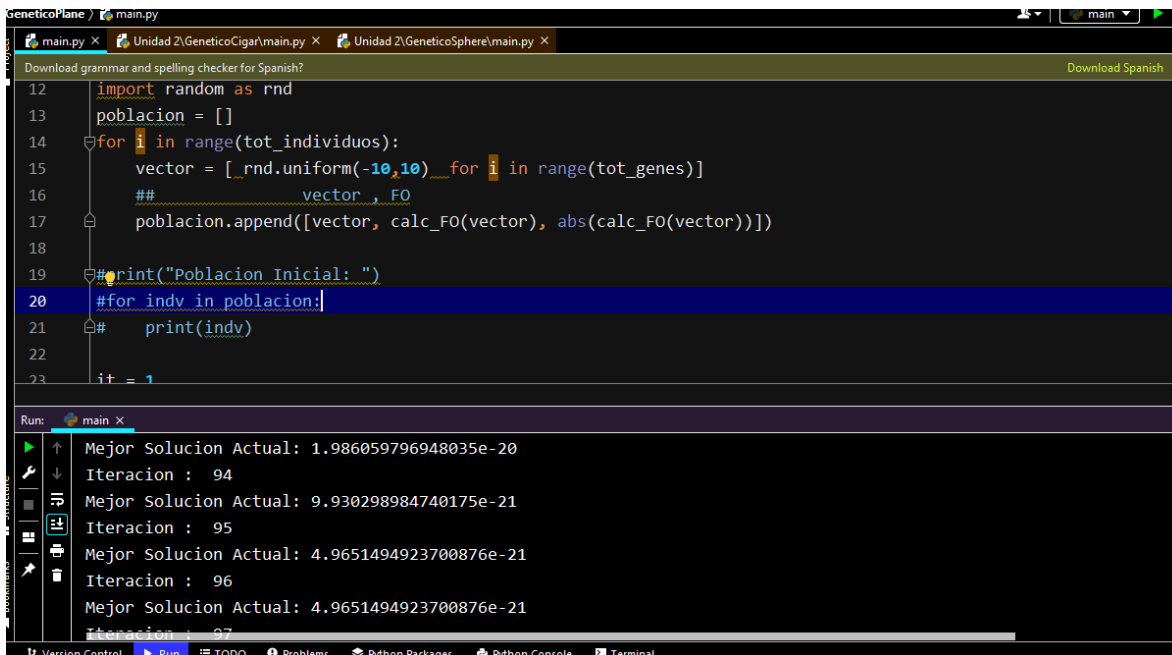
Version Control Run TODO Problems Python Packages Python Console Terminal

#print("Poblacion Inicial: ")
#for indv in poblacion:
#    print(indv)

it = 1
mejorActual = 11
while it<=100:
    print("Iteracion : ", it)
    it+=1

    padres = []
    tot_padres = 30

    poblacion.sort(key=_lambda x:x[2], reverse=False)
    #sorted(poblacion, key=_lambda
```



```

12 import random as rnd
13 poblacion = []
14 for i in range(tot_individuos):
15     vector = [rnd.uniform(-10,10) for i in range(tot_genes)]
16     ##          vector , F0
17     poblacion.append([vector, calc_F0(vector), abs(calc_F0(vector))])
18
19 #print("Poblacion Inicial: ")
20 #for indv in poblacion:
21     print(indv)
22
23 it = 1
  
```

Run: main x

```

Mejor Solucion Actual: 1.986059796948035e-20
Iteracion : 94
Mejor Solucion Actual: 9.930298984740175e-21
Iteracion : 95
Mejor Solucion Actual: 4.9651494923700876e-21
Iteracion : 96
Mejor Solucion Actual: 4.9651494923700876e-21
Iteracion : 97
  
```

## Conclusiones:

La utilización de un algoritmo genético para binarios puede llegar a ser muy útil cuando se necesitan afrontar problemas muy complejos sin tener que llegar a utilizar algún procedimiento muy complejo, y bueno este algoritmo al ser un proceso de manera aleatoria a dependido de los valores iniciales por lo cual no siempre se obtienen los mismos resultados con la misma cantidad de iteraciones.

## Práctica 3. ID3.

**Descripción:** Desarrollo de una versión simplificada de un árbol inductivo de decisión ID3.

### Introducción:

Dentro de los conceptos necesarios para el entendimiento de la practica realizada están los siguientes:

### Desarrollo:

Código utilizado

```

1  ###ESTA ES UNA VERSION SIMPLIFICADA DEL ARBOL INDUCTIVO DE DECISION ID3.
2  #
3  # NO SE REALIZA LA IMPRESIÓN DEL ÁRBOL. NO OBSTANTE PERMITE LA EVALUACIÓN D
4  #
5  # VERSIÓN ELABORADA EL 30 DE MARZO DE 2022
6  # DR. ALEJANDRO H. GARCÍA RUIZ
7  ###
8  import Valor
9  import Atributo
10 import Nodo
11 import math
12 contadorNodos = 0
13 #####
14 def generarVector(listaAtributos, vector):    #Vector = registro/caso/fila
15     aux = [];
16     for i in range(len(vector)):
17         aux.append(Valor.Valor(listaAtributos[i].nombre, vector[i]))
18     return aux
19 #####
20 def Log2(n):    #version generalizada.. sin embargo, math cuenta con log2
21     return math.log10(n) / math.log10(2)
22 #####
23 def cuantosPorClase(caso, clase):
24     total = 0;
25     for i in range(len(caso)): # por cada caso
26         if caso[i][len(caso[i]) - 1].etiqueta==clase:
27             total+=1
28     return total
29 #####
30 def obtenerMejorAtrib(listaAtributos):
31     maxGanancia = -99999
32     IndiceMax = 0 # si la ganancia es 0, entonces se escoge el primer nodo d
33     etropiaAtributo = 0;
34     inter = 0;

```

```

34     inter = 0;
35     # Entropia del conjunto / arbol
36     totalCasos = listaAtributos[len(listaAtributos) - 1].totalCasos
37     etropiaArbol = 0
38     for i in range(len(listaAtributos[len(listaAtributos) - 1].listaEtiquetas)):
39         aux = len(listaAtributos[len(listaAtributos) - 1].listaCasos[i]) / totalCasos
40         etropiaArbol += -1.0 * aux * Log2(aux)
41     # Entropia del atributo
42     for i in range(len(listaAtributos)-1): # por cada atributo
43         etropiaAtributo = 0;
44         for j in range(len(listaAtributos[i].listaEtiquetas)): # por cada etiqueta
45             inter = 0;
46             clase = listaAtributos[len(listaAtributos) - 1]
47             for k in range(len(clase.listaEtiquetas)): # por cada clase
48                 aux = cuantosPorClase(listaAtributos[i].listaCasos[j], clase.listaEtiquetas[k])
49                 if (aux != 0):
50                     aux /= len(listaAtributos[i].listaCasos[j]);
51                     inter += -1.0 * aux * Log2(aux);
52             etropiaAtributo += len(listaAtributos[i].listaCasos[j]) / listaAtributos[i].totalCasos * inter;
53     # Ganancia:
54     Ganancia = (etropiaArbol - etropiaAtributo) / etropiaArbol;
55     # ACTUALIZA al ATRIBUTO SELECCIONADO...
56     if (maxGanancia < Ganancia):
57         maxGanancia = Ganancia
58         IndiceMax = i
59     return listaAtributos[IndiceMax]
60     #####
61     def claseMayoritaria(listaCasos, listaEtiquetas):
62         claseMayor = ""
63         contadores = [0 for i in range(len(listaEtiquetas))]
64         indice = -1
65         for i in range(len(listaCasos)):
66             for j in range(len(listaCasos[i])):
67                 indice = listaEtiquetas.index(listaCasos[i][j][len(listaCasos[i][j]) - 1].etiqueta)

```

```

68         if (indice != -1):
69             contadores[indice]+=1
70     max = -9999;
71     for i in range(len(contadores)):
72         if (max < contadores[i]):
73             max = contadores[i]
74             claseMayor = listaEtiquetas[i]
75     return claseMayor;
76     #####
77 def removerAtributo(caso, nombre):
78     indiceColumna = -1
79     for j in range(len(caso)): # Por cada caso
80         if Valor.Valor(nombre) in caso[j]:
81             indiceColumna = caso[j].index(Valor.Valor(nombre))
82             del caso[j][indiceColumna]
83     return caso;
84     #####
85 def ID3(casos, listaAtributos, mayoritaria):
86     global contadorNodos
87     contadorNodos+=1
88     root = Nodo.Nodo(contadorNodos)
89     mismaClase = False
90     aux = -1
91     clase = listaAtributos[len(listaAtributos) - 1];
92     for k in range(len(clase.listaEtiquetas)): # por cada clase
93         aux = cuantosPorClase(casos, clase.listaEtiquetas[k])
94         if (aux == len(casos)): # Todos los atributos son de la misma clase
95             mismaClase = True
96             root.setEtiqueta(clase.listaEtiquetas[k]);
97     if (not mismaClase):
98         if (len(casos)==0):
99             root.setEtiqueta(mayoritaria)
100     else:

```

```

100     else:
101         # PROCESO DE LIMPIA DE LA LISTA DE CASOS DE CADA ETIQUETA
102         for i in range(len(listaAtributos)): # Por cada atributo
103             for j in range(len(listaAtributos[i].listaEtiquetas)): # por cada etiqueta
104                 listaAtributos[i].listaCasos[j] = []
105                 listaAtributos[i].totalCasos = 0
106         # Generar Lista de Atributos
107         indice=-1
108         for i in range(len(casos)):
109             for j in range(len(listaAtributos)): # por cada atributo.. El ultimo atributo es la clase...
110                 indice = listaAtributos[j].listaEtiquetas.index(casos[i][j].etiqueta)
111                 listaAtributos[j].listaCasos[indice].append(casos[i])
112                 listaAtributos[j].totalCasos+=1
113         A = obtenerMejorAtrib(listaAtributos)
114         root.setAtributo(A.nombre)
115         # ELIMINA ATRIBUTO
116         for i in range(len(listaAtributos)):
117             if listaAtributos[i].nombre == A.nombre:
118                 del listaAtributos[i]
119                 break
120         ##CONTINUA PROCEDIMIENTO
121         for i in range(len(A.listaEtiquetas)): # Para cada etiqueta (v) del atributo hacer
122             if (len(A.listaCasos)==0):
123                 root.setEtiqueta(mayoritaria)
124             else:
125                 mayoritaria = claseMayoritaria(A.listaCasos, listaAtributos[len(listaAtributos) - 1].listaEtiquetas)
126                 A.listaCasos[i] = removerAtributo(A.listaCasos[i], A.nombre)
127                 listaAtribAuxiliar = []
128                 listaAtribAuxiliar.extend(listaAtributos)
129                 # Solo se enviarán los casos de la etiqueta en particular...
130                 root.addRama(A.listaEtiquetas[i], ID3(A.listaCasos[i], listaAtribAuxiliar, mayoritaria))
131         return root;
132     #####

```

```

133 def generarVectorToEval(listaAtributos, vector):
134     aux = []
135     for i in range(len(vector)):
136         aux.append(Valor.Valor(listaAtributos[i].nombre, vector[i]))
137     return aux
138
139 #EJECUCIÓN PRINCIPAL
140 archivo = open("cancer_training.csv")
141 contenido = archivo.readlines()
142 #####QUITA \n
143 for i in range(len(contenido)):
144     contenido[i] = contenido[i].replace("\n", "")
145 #####
146 listaAtributos = []
147 aux = contenido[0].split(",")
148 #Lee el nombre de los atributos
149 for i in range(len(aux)):
150     listaAtributos.append(Atributo.Atributo(aux[i]))
151 #Para Evaluar nuevos casos...
152 nombresAtributos = []
153 nombresAtributos.extend(listaAtributos)
154 #####
155 for l in range(1, len(contenido)):
156     linea = contenido[l]
157     aux = linea.split(",")
158     #Nota por cada atributo.. El ultimo atributo es la clase...
159     for i in range(len(aux)):
160         # añade el caso a la etiqueta correspondiente
161         temporal = generarVector(listaAtributos, aux);
162         #////////////////////
163         if (aux[i] not in listaAtributos[i].listaEtiquetas):
164             # se añade la etiqueta y el caso al final
165             listaAtributos[i].listaEtiquetas.append(aux[i])

```



```

166         listaAtributos[i].listaCasos.append([])
167         # representa por cada etiqueta al valor que le sigue.. ya sea un atributo o
168         # listaAtributos[i].siguiente.append(None)
169         indice = listaAtributos[i].listaEtiquetas.index(aux[i])
170         listaAtributos[i].listaCasos[indice].append(temporal)
171         listaAtributos[i].totalCasos+=1
172 #####
173 #####
174 # COMIENZA ENTRENAMIENTO (GENERACION) DEL ARBOL ID3
175 #####
176 #####
177 raiz = obtenerMejorAtrib(listaAtributos) # Sea Raiz el MEJOR de los atributos
178 #####
179 # ELIMINA ATRIBUTO
180 for i in range(len(listaAtributos)):
181     if listaAtributos[i].nombre == raiz.nombre:
182         del listaAtributos[i]
183         break
184     listaAtribsAuxiliar = []
185     #####
186     contadorNodos = 0;
187     arbol = Nodo.Nodo(contadorNodos)
188     arbol.setAtributo(raiz.nombre)
189     r = None;
190     indiceColumna = -1;
191     for i in range(len(raiz.listaEtiquetas)): # Para cada etiqueta (v) del atributo hacer
192         contadorNodos+=1
193         nodo = Nodo.Nodo(contadorNodos)
194         arbol.addRama(raiz.listaEtiquetas[i], nodo)
195         #Sea Ejemplos(v) el subconjunto de ejemplos cuyo valor de atributo Raiz es v
196         # --- Lista de Casos
197         if (len(raiz.listaCasos)==0):
198             pass

```

```

198     pass
199     else:
200         #Sino Devolver Id3(Ejemplos(v), Atributo-objetivo, Atributos/{A})
201         mayoritaria = claseMayoritaria(raiz.listaCasos, listaAtributos[len(listaAtributos) - 1].listaEtiquetas);
202         raiz.listaCasos[i] = removerAtributo(raiz.listaCasos[i], raiz.nombre);
203         listaAtribsAuxiliar = []
204         listaAtribsAuxiliar.extend(listaAtributos)
205         #Solo se enviarán los casos de la etiqueta en particular...
206         arbol.addRama(raiz.listaEtiquetas[i], ID3(raiz.listaCasos[i], listaAtribsAuxiliar, mayoritaria));
207     print("Fin Creación del Arbol")
208     #####
209     #####
210     # COMIENZA EVALUACIÓN DEL ARBOL ID3
211     #####
212     #####
213     print("\n\n Evaluación de Casos: ...")
214     casos = []
215     try:
216         arch = open("cancer_test.csv")
217         contenido = arch.readlines()
218         #####QUITA \n
219         for i in range(len(contenido)):
220             contenido[i] = contenido[i].replace("\n", "")
221         #####
222         for linea in contenido:
223             casos.append(linea.split(","))
224     except Exception as ex:
225         print("Error")
226
227     totalCorrectos = 0;
228     for k in range(len(casos)): #Por cada caso
229         toEvaluar = generarVectorToEval(nombresAtributos, casos[k]);
230         temp = arbol
231         while (temp.getEtiqueta()==""):
232             for i in range(len(toEvaluar)):
233                 if temp.getAtributo() == toEvaluar[i].nombreAtributo:
234                     temp = temp.getRamas()[toEvaluar[i].etiqueta]
235             claseReal = casos[k][len(casos[k]) - 1]
236             print("Clase Asignada: " + temp.getEtiqueta() + " Clase Real: " + claseReal + " Evaluacion: " + str(claseReal == temp.getEtiqueta()));
237             if claseReal == temp.getEtiqueta():
238                 totalCorrectos+=1
239
240     print("\nRendimiento : " + str(round(totalCorrectos / len(casos) * 100, 4)));

```

### Explicación:

1. Inicialización del contador de nodos con un valor de cero (línea 12).
2. Definición del generador de vectores, donde se inicializa la lista auxiliar, recorriendo la longitud del vector para agregar el valor de del nombre de la lista de atributos y el vector en su posición acorde el recorrimiento, regresando un el valor del auxiliar (líneas 14-18).
3. Definición del logaritmo natural base 2 (líneas 20-21).
4. Definición de cuantos, por clase, donde se inicializa el total en cero y se recorre la longitud por cada caso, si la etiqueta del caso es igual a la clase, el total incrementara en uno (líneas 23-28).
5. Definición de la obtención del mejor atributo (líneas 30-59).
6. Inicialización de la ganancia máxima con un valor de -99999, de índice máximo con un valor de cero, si es cero la ganancia entonces se escoge el primer nodo de forma arbitraria, de la entropía del atributo un valor inicial de cero y por último un inter de valor inicial en cero (líneas 31-34).

7. Inicialización del total de casos donde tomara el valor en la lista de atributos en la posición de la longitud de la lista de atributos menos uno (línea 36).
8. Inicialización de la entropía del árbol (línea 37).
9. Recorrimiento de las listas de etiquetas, donde tomara el valor de la longitud de la lista de casos entre el total de casos, y la entropía del árbol incrementara con un valor derivado de el producto de menos uno con el auxiliar, producto con el logaritmo natural base dos del auxiliar (líneas 38-40).
10. Recorrimiento de cada atributo de la longitud de la lista de atributos menos uno (líneas 42-58).
11. Recorrimiento de la lista de etiquetas de la lista de atributos (líneas 44-52).
12. Inicialización de la clase (línea 46).
13. Recorrimiento de la lista de etiquetas por cada clase (líneas 47-51).
14. Inicialización de la entropía del atributo con valor de la división de la lista de casos de la lista de atributos entre el producto del total de casos con el inter (línea 52).
15. Inicialización de la ganancia con valor dado con la diferencia de la entropía del árbol y la entropía del atributo entre la entropía del árbol (línea 54).
16. Actualización del atributo seleccionado (líneas 56-59).
17. Definición de la clase mayoritaria (líneas 61-75).
18. Recorrimiento de la lista de casos (líneas 65-69).
19. Recorrimiento de las listas de casos, donde el índice será igual a la etiqueta del índice de la lista de etiquetas (líneas 66-69).
20. Si el índice es diferente a menos uno, los contadores en el índice incrementan de uno en uno (líneas 68-69).
21. Recorrimiento de los contadores (líneas 71-75).
22. Si el máximo es menor a los contadores, el auxiliar tomará el valor de los contadores en la posición conforme al recorrido y la clase mayor será igual a la lista de etiquetas (líneas 72-74).
23. Se define la función para remover el atributo (líneas 77-83).
24. Recorrimiento de la longitud de casos de cada caso (líneas 79-83).
25. Si el nombre que esta en el valor esta en el caso en la posición j, el índice de la columna será igual al índice del caso y elimina el caso (líneas 80-82).
26. Definiendo el ID3 (líneas 85-131).
27. Inicialización del contador de nodos de manera global (línea 86).
28. Inicialización del root el cual será igual al nodo del contador de nodos (línea 88).
29. Recorrimiento de la lista de etiquetas de cada clase (líneas 92-96).
30. El auxiliar tomara el valor de cuantos por clase el cual consiste en los casos y en la lista de etiquetas de la clase (línea 93).
31. Si todos los atributos son de la misma clase, misma clase será verdadera y el root dará la etiqueta (líneas 94-96).
32. Si no es la misma clase (líneas 97-130).
33. Si la longitud de los casos es igual a cero, el root dará la etiqueta mayoritaria (líneas 98-99).
34. Si no es igual a cero efectuara los procesos de limpia de casos de etiqueta, generara la lista de atributos, eliminara atributos y continuara con el procedimiento (líneas 100-130).

35. Definición de la función generar vector para evaluar, recorriendo la longitud del vector en el que el auxiliar tomara el valor del nombre de la lista de atributos (líneas 133-137).
36. Se efectúa la ejecución principal en el cual se lee el archivo de entrenamiento (líneas 140-141).
37. Recorrimiento de la longitud de la lista de contenido para reemplazar los saltos de línea por comas (143-144).
38. Se lee el nombre de los atributos (líneas 149-150).
39. Inicialización de los nombres de atributos para la evaluación de nuevos casos (líneas 152-153).
40. Recorrimiento de la longitud de la lista de contenido partiendo de uno (líneas 155-171).
41. Recorrimiento de la longitud de la lista de auxiliar donde añade el caso a la etiqueta correspondiente (líneas 159-171).
42. Si el auxiliar no esta en la lista de etiquetas de la lista de atributos entonces se añade la etiqueta y caso al final (líneas 163-166).
43. Representación por cada etiqueta el valor que le sigue ya sea un atributo o una clase (líneas 169-171).
44. Sea raíz el mejor de los atributos (línea 177).
45. Se elimina el atributo (líneas 180-184).
46. Inicialización del árbol con valor del nodo de contador de nodos donde el árbol da el nombre de la raíz (líneas 187-188).
47. Recorrimiento de la lista de etiquetas de la raíz, esto se hará para cada etiqueta (v) del atributo, el árbol añadirá una rama (líneas 191-206).
48. Evaluación del árbol ID3 (líneas 213-240).
49. Leerá el archivo de prueba, reemplazará los saltos de línea por comas, si ocurre un error se mandará un mensaje de error (líneas 215-225).
50. Recorrimiento de k por cada caso (líneas 228-238).
51. Inicialización para evaluar con un valor de la función generar vector para evaluar y temporal será igual al árbol (líneas 229-230).
52. Mientras que la etiqueta obtenida por el temporal este vacía, se recorrerá la longitud de la lista para evaluar, eso si el atributo obtenido por el temporal es igual al nombre del atributo de la lista para evaluar el temporal obtendrá la etiqueta de las ramas (líneas 231-238).
53. Visualización de la clase asignada, la clase real y la evaluación (línea 236).
54. Si la clase real es igual a la etiqueta obtenida por el temporal, el total de correctos incrementara de uno en uno (líneas 237-238).
55. Visualización del rendimiento (línea 240).

### **Resultados:**

Realizamos las pruebas pertinentes.

```
Evaluación de Casos: ...  
Clase Asignada: NO Clase Real: NO Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: NO Clase Real: NO Evaluacion: True  
Clase Asignada: NO Clase Real: NO Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: NO Clase Real: NO Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: NO Clase Real: NO Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
Clase Asignada: NO Clase Real: NO Evaluacion: True  
Clase Asignada: YES Clase Real: YES Evaluacion: True  
  
Rendimiento : 100.0  
  
Process finished with exit code 0
```

### Conclusiones:

La utilización de este algoritmo ID3 al ser un método de aprendizaje supervisado no parametrizado utilizado principalmente para su clasificación y regresión, cuyo propósito es construir un modelo el cual pueda predecir valores de la variable objetivo, el cual ira aprendiendo una regla de decisiones simples. Este algoritmo entre más información aporte más importante son las características.

### Práctica 4. Asociador Lineal

**Descripción:** Verificar si el asociador lineal es capaz de darnos una respuesta igual a la que se contiene en la instancia.

### Introducción:

Dentro de los conceptos necesarios para el entendimiento de la practica realizada están los siguientes:

### Desarrollo:

### Código utilizado

```
8 import numpy as n
9
0 archivo = open("instancia_claseExplicacion_training.txt")
1 contenido = archivo.readlines()
2
3 X = contenido[3:3+int(contenido[1])]
4 X = [i.split("\t") for i in X]
5 X = [list(map(int, i)) for i in X]
6
7 Y = contenido[3+int(contenido[1]):]
8 Y = [i.split("\t") for i in Y]
9 Y = [list(map(int, i)) for i in Y]
0
1 X = n.array(X)
2 Y = n.array(Y)
3
4 Paso1 = X.dot(X.T)
5 Paso2 = n.linalg.inv(Paso1)
6 Xpseudo = X.T.dot(Paso2)
7
8 W = Y.dot(Xpseudo)
9
0 print("X:")
1 print(X)
2
3 print("Y:")
4 print(Y)
5
6 print("W:")
```

```
36 print("W:")
37 print(W)
38
39 #####
40 #####
41 ### EVALUACION DE LOS CASOS DE PRUEBA
42 #####
43 #####
44 print("Prueba...")
45 archivo = open("instancia_claseExplicacion_test.txt")
46 contenido = archivo.readlines()
47
48 X = contenido[3:3+int(contenido[1])]
49 X = [i.split("\t") for i in X]
50 X = [list(map(int, i)) for i in X]
51
52 Y = contenido[3+int(contenido[1]):]
53 Y = [i.split("\t") for i in Y]
54 Y = [list(map(int, i)) for i in Y]
55
56 X = n.array(X)
57 Y = n.array(Y)
58
59 print("X:")
60 print(X)
61
62 print("Y:")
63 print(Y)
64
65 casosCorrectos = 0
66
67 #CLASE SALIDA1 SALIDA 2 SALIDA 3
```

```
68 Clases = ["ALTA", "MEDIA", "BAJA"]
69
70 for i in range(X.shape[1]): #para cada uno de los casos/registros de prueba
71     print("Prueba del Caso ", i + 1)
72     casoi = X[:,i]
73     print("Caso Analizado: ")
74     print(casoi)
75
76     Ycasoi = W.dot(casoi)
77     print("Salidas Generadas: ")
78     print(Ycasoi)
79
80     print("Salidas Real: ")
81     Yrealcasoi = Y[:,i]
82     print(Yrealcasoi)
83
84     IndexMaxYcasoi = list(Ycasoi).index(max(Ycasoi))
85     IndexMaxYrealcasoi = list(Yrealcasoi).index(max(Yrealcasoi))
86
87     if IndexMaxYcasoi == IndexMaxYrealcasoi:
88         casosCorrectos +=1
89
90     print("Clase Asignada: ", Clases[IndexMaxYcasoi])
91     print("Clase Real: ", Clases[IndexMaxYrealcasoi])
92     print()
93
94     print("Total de Casos Analizados: ", X.shape[1])
95     print("Total de Casos Correctos: ", casosCorrectos)
96
97     print("Eficiencia del Asociador Lineal: ", casosCorrectos/X.shape[1]*100.0)
98
```

### Explicación:

1. Se lee el archivo de entrenamiento (líneas 10-11).
2. Se obtiene el valor de x después de separar los valores por tabulador y convertirlos a enteros (líneas 13-15).
3. Se obtiene el valor de y después de separar los valores por tabulador y convertirlos a enteros (líneas 17-19).
4. Para poder hacer la multiplicación punto producto punto (líneas 21-22).
5. Se obtiene la pseudoinversa de x (líneas 24-26).
6. Se obtiene w al multiplicar y por el punto producto punto de la pseudoinversa de x (línea 28).
7. Visualiza X, Y, W (líneas 30-37).
8. Lee el archivo de prueba (líneas 45-46).



9. Se obtiene el valor de x después de separar los valores por tabulador y convertirlos a enteros (líneas 48-50).
10. Se obtiene el valor de y después de separar los valores por tabulador y convertirlos a enteros (líneas 52-54).
11. Para poder hacer la multiplicación punto producto punto (líneas 56-57).
12. Visualización de x, y además de la inicialización de casos correctos (líneas 59-65).
13. Inicialización de las clases (línea 68).
14. Recorrimiento para cada uno de los casos (líneas 70-92).
15. Visualización del numero de caso con el caso analizado con las salidas generadas y salidas reales (líneas 71-82).
16. Obtención del caso mayor en ambos en el analizado y en real (líneas 84-85).
17. Si el caso mayor analizado es igual al caso mayor correcto, casos correctos aumentara en uno y visualizara la clase real y la clase asignada (líneas 87-92).
18. Visualización del total de casos analizados y el total de casos correctos (líneas 94-95).

**Resultados:**

Realizamos las pruebas pertinentes.

```
Prueba del Caso  2
Caso Analizado:
[73 55 13  1 78 46]
Salidas Generadas:
[-0.15894139 -0.00643914  0.94601149]
Salidas Real:
[0 0 1]
Clase Asignada:  BAJA
Clase Real:  BAJA

Prueba del Caso  3
Caso Analizado:
[56 90 81 22 60 49]
Salidas Generadas:
[ 0.56759329 -0.29202906  0.50788898]
Salidas Real:
[1 0 0]

Prueba del Caso  7
Caso Analizado:
[32 71 41 84 16 82]
Salidas Generadas:
[ 0.46633884  0.7097332  -0.08011415]
Salidas Real:
[0 1 0]
Clase Asignada:  MEDIA
Clase Real:  MEDIA

Total de Casos Analizados:  7
Total de Casos Correctos:  6
Eficiencia del Asociador Lineal:  85.7142857

Process finished with exit code 0
```

**Conclusiones:**

Con la practica mediante los datos de entrenamiento, los datos de prueba y una clase asignada, buscando una eficiencia en el programa en el cual pueda obtener menos errores y más aciertos en su ejecución acorde los casos asignados y los casos correctos obtuvimos una eficiencia del ochenta y cinco por ciento en cual con siete casos analizados solamente se equivocó en uno, esto quiere decir que con cada ejecución el algoritmo va aprendiendo para poder solucionar problemas como las redes neurales.

## Práctica 5. Discretizador EWB

**Descripción:** Verificar si programa puede discretizar los datos continuos a datos discretos.

### Introducción:

Dentro de los conceptos necesarios para el entendimiento de la practica realizada están los siguientes:

### Desarrollo:

#### Código utilizado

```

2      #archivo = open("iris_completa.csv")
3      archivo = open("wine.csv")
4      contenido = archivo.readlines()
5      #####
6      instancia = []
7      for i in contenido:
8          instancia.append(i.split(","))
9      encabezados = instancia[0]
10     del instancia[0]
11     #####
12     ###GRUPOS A GENERAR
13     v_K = 5 # Pagina de referencia comparativa ->>> https://orange.readth
14     #####
15     intervalos = []
16     for index_atributo in range(len(instancia[0])-1):
17         auxiliar = []
18         for index_registro in range(len(instancia)):
19             auxiliar.append(float(instancia[index_registro][index_atributo]))
20         v_max = max(auxiliar)
21         v_min = min(auxiliar)
22         v_width = round((v_max-v_min)/v_K,4)
23         #####
24         print("Atributo analizado:", encabezados[index_atributo])
25         print("min: ", v_min)
26         print("max: ", v_max)
27         print("width: ", v_width)
28         #####
29         control = round(v_min+v_width,4)
30         temp = ["<" + str(control)]
31         for j in range(1,v_K-1):
32             s = "[" + str(control) + ", "

```

```
33         control = round(control+v_width,4)
34         s += str(control) + ","
35         temp.append(s)
36         temp.append(">=" + str(control))
37         intervalos.append(temp)
38         auxiliar.clear()
39         #####
40         for i in intervalos:
41             print(i)
```

### Explicación

1. Se lee el archivo a discretizar (líneas 2-4).
2. Se declara la instancia y se recorre el contenido para agregarlo a la instancia separado por comas y eliminando los encabezados (líneas 6-10).
3. Se generan los grupos (línea 13).
4. Se declaran los intervalos y se recorre la instancia menos la clase (líneas 16-38).
5. Se declara el auxiliar para almacenar los datos de cada atributo y se recorre toda la instancia para agregar los valores de cada atributo en un tipo flotante al que el valor mínimo, máximo y su ancho (líneas 18-22).
6. Visualización del atributo analizado, el valor mínimo, valor máximo y el valor del ancho (líneas 24-27).
7. Inicialización de control donde será igual al valor mínimo más el valor del ancho (línea 29).
8. Inicialización del temporal donde tomara los valores menor, mayor o menor y mayor o igual al valor de control, colocándolos de una manera ordenada y ya separada, agregando al temporal a la lista de intervalos para poder eliminar el auxiliar (líneas 29-38).
9. Recorre los intervalos para visualizarlos (líneas 40-41).

### **Resultados:**

Realizamos las pruebas pertinentes.

```
Atributo analizado: atr11
min:  0.48
max:  1.71
width: 0.246
Atributo analizado: atr12
min:  1.27
max:  4.0
width: 0.546
Atributo analizado: atr13
min:  278.0
max:  1680.0
width: 280.4
```

```
['<11.79', '[11.79, 12.55)', '[12.55, 13.31)', '[13.31, 14.07)', '>=14.07']
['<1.752', '[1.752, 2.764)', '[2.764, 3.776)', '[3.776, 4.788)', '>=4.788']
['<1.734', '[1.734, 2.108)', '[2.108, 2.482)', '[2.482, 2.856)', '>=2.856']
['<14.48', '[14.48, 18.36)', '[18.36, 22.24)', '[22.24, 26.12)', '>=26.12']
['<88.4', '[88.4, 106.8)', '[106.8, 125.2)', '[125.2, 143.6)', '>=143.6']
['<1.56', '[1.56, 2.14)', '[2.14, 2.72)', '[2.72, 3.3)', '>=3.3']
['<1.288', '[1.288, 2.236)', '[2.236, 3.184)', '[3.184, 4.132)', '>=4.132']
['<0.236', '[0.236, 0.342)', '[0.342, 0.448)', '[0.448, 0.554)', '>=0.554']
['<1.044', '[1.044, 1.678)', '[1.678, 2.312)', '[2.312, 2.946)', '>=2.946']
['<3.624', '[3.624, 5.968)', '[5.968, 8.312)', '[8.312, 10.656)', '>=10.656']
['<0.726', '[0.726, 0.972)', '[0.972, 1.218)', '[1.218, 1.464)', '>=1.464']
['<1.816', '[1.816, 2.362)', '[2.362, 2.908)', '[2.908, 3.454)', '>=3.454']
['<558.4', '[558.4, 838.8)', '[838.8, 1119.2)', '[1119.2, 1399.6)', '>=1399.6']
```

## Conclusiones:

Con la practica acorde a la simplicidad de esta, al momento de la ejecución con los valores continuos, analizando atributo por atributo a la conversión a números discretos únicamente queda con la condición el cual fue nuestro resultado serian nuestros valores, aquel número mayor al calculado será el valor uno y así con todas las condiciones.

## Práctica 6. Naive Bayes

**Descripción:** Verificar si programa puede decirnos de todos los atributos cual es el que tiene mayor probabilidad de ocurrencia.

## Introducción:

Dentro de los conceptos necesarios para el entendimiento de la practica realizada están los siguientes:

## Desarrollo:

### Código utilizado

```
3 file2read = open("Instancia_Explicacion_Golf.csv")
4 file_content = file2read.readlines()
5
6 #####
7 dataset = []
8 for i in file_content:
9     dataset.append((i.replace("\n", "").split(",")))
10 headers = dataset[0]
11 del dataset[0] #remove headers from dataset
12 #####
13 ##count registers per class
14 #####
15 probabilities = [] #se cuentan los atributos de cada
16 auxiliar = {}
17 for register_index in range(len(dataset)): #por cada
18     label = dataset[register_index][-1] #etiqueta
19     if label in auxiliar: #si esta la etiqueta se le
20         | auxiliar[label] += 1
21     else:
22         | auxiliar[label] = 1
23 probabilities.append(auxiliar)
```

```
25     ##count registers per attribute
26     #####
27     #por cada indice del atributo voy a recorrer a todos los registros para recorrer h
28     for attribute_index in range(len(dataset[0])-1): #cuantos registros hay por atribu
29         auxiliar = {} #se armara una tupla con la etiqueta y la clase
30         for register_index in range(len(dataset)):
31             v_label = dataset[register_index][attribute_index] #se asigna un label par
32             v_class = dataset[register_index][-1] # se asigna la clase
33             if (v_label,v_class) in auxiliar: #si la tupla se encuentra en el diccionari
34                 auxiliar[(v_label,v_class)] += 1 #se le agrega unnn al valor que ya ten
35             else:
36                 auxiliar[(v_label,v_class)] = 1 #si no se creara
37                 #print(v_label, " ", v_class)
38         probabilities.append(auxiliar)
39     #####
40     ##calculate probabilities per attribute
41     #####
42     for index in range(1, len(probabilities)):
43         for c in probabilities[index]: #per attribute
44             #print(probabilities[index][c]) # se obtienen las probabilidades por cada
45             #print(probabilities[0][c[1]])
46             probabilities[index][c] = probabilities[index][c]/probabilities[0][c[1]]
47             #print(probabilities[0][c])
48     #####
49     ##calculate probabilities per class
```

```
51 for c in probabilities[0]: #por cada clase de las probabilidades de c
52     probabilities[0][c] = probabilities[0][c]/len(dataset) #cuantos n
53     #print(probabilities[0][c])
54     #####
55     #####
56     #####
57     ## classify a register
58     #####
59
60 #register = ["Soleado", "Frío", "Alta", "Fuerte"] #test1
61 register = ["Nublado", "Frío", "Normal", "Fuerte"] #test2
62
63
64 sum = 0
65 probabilities_per_class = {}
66 for c in probabilities[0]: #per class por cada clase
67     #print(c)
68     auxiliar = probabilities[0][c] #es el resultado de haber echo tod
69     for index in range(1, len(probabilities)):
70         print(register[index-1], " ", c)
71         if (register[index-1], c) in probabilities[index]: #si exist
72             auxiliar *= probabilities[index][(register[index-1], c)]#
73             #al aux se le mul la probabilidad de dicho caso en cuesti
74         else:
75             auxiliar = 0 #nullify the product
76     sum += auxiliar
77     probabilities_per_class[c] = auxiliar
78
79 max = -9999 #cual es el mas grande despues de calculr la probabilidad
80 c_toAssign = ""
81 for p in probabilities_per_class: #para cada p sera que la probabilidad de ese
82     probabilities_per_class[p] = probabilities_per_class[p]/sum #a la probabili
83     if probabilities_per_class[p] > max: #si la probabiidad es mayor a la que te
84         max = probabilities_per_class[p]
85         c_toAssign = p # clase asignada
86     #####
87     #da una seguridad aparente que ccon la informacion que tiene el esta seguro que
88     print("Assigned Class: ", c_toAssign, " Probability: ", round(max*100,4), "%")
89
```

## Explicación

1. Se lee el archivo (líneas 3-4).



2. Se crea un dataset en donde se almacenará los datos eliminando el salto de línea y separando por coma además de eliminar el encabezado (líneas 7-11).
3. Es cuentan los atributos de cada clase y se crea el auxiliar, donde por cada registro nos interesa, label será igual a la etiqueta y si la etiqueta esta en el diccionario pues lo incrementa en uno si no la crea, lo agrega a probabilities (líneas 15-23).
4. Por cada índice del atributo se recorrerá a todos los registros para recorrer hacia abajo, además de cuantos registros hay por atributo por cada clase (líneas 28-29).
5. Recorrimiento de los datos por cada índice de registro, donde se asigna un label por cada atributo, se asigna la clase y si la tupla se encuentra en el diccionario se le agrega uno al valor ya que ya tenia y si no se creara (líneas 30-36).
6. Se calculan las probabilidades por atributo (líneas 42-46).
7. Se calculan las probabilidades por clase (líneas 51-52).
8. Inicialización de registro de prueba (línea 61).
9. Inicialización de la sumatoria de las probabilidades de cada atributo (línea 64).
10. Inicialización de las probabilidades por clase donde se recorrerá por cada clase, en el cual el auxiliar obtiene el resultado de haber echo todas las multiplicaciones (líneas 65-68).
11. Recorrimiento de las probabilidades por cada atributo donde si existe una probabilidad para tal caso el auxiliar será igual a su producto con el atributo y clase, sino será un valor nulo (líneas 69-75).
12. La sumatoria se adhiere al auxiliar y los valores por clase en la clase toman el valor de auxiliar (líneas 76-77).
13. Inicialización del máximo para después calcular la probabilidad y se inicializa la clase asignada (líneas 79-80).
14. Recorrimiento de las probabilidades por clase por cada probabilidad será igual a la probabilidad que tengo más la sumatoria y si la probabilidad es mayor a la que tengo se asigna al máximo y la clase asignada toma el valor de la clase (líneas 81-85).
15. Visualización de la clase asignada y la probabilidad (línea 88).

### Resultados:

Realizamos las pruebas pertinentes.

```
Soleado    No
Frío       No
Alta       No
Fuerte     No
Soleado    Si
Frío       Si
Alta       Si
Fuerte     Si
Assigned Class:  No  Probability:  79.5417 %
```

```
Process finished with exit code 0
```

```
Lluvioso  No
Ambiente   No
Alta       No
Débil      No
Lluvioso  Si
Ambiente   Si
Alta       Si
Débil      Si
Assigned Class:  Si  Probability:  53.6481 %
```

```
Process finished with exit code 0
```

### Conclusiones:

Con la utilización de este algoritmo Naive Bayes para la toma de decisiones acorde a un conjunto de datos y el calculo de las probabilidades correspondientes a sus clases y atributos nos da una seguridad aparente que con la información que tiene el algoritmo esta seguro de que la clase que debe asignar en el primer caso fue un No y en el segundo es cao fue un Si no muy seguro.