

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

**UNIDAD DE APRENDIZAJE:**

**MACHINE LEARNING**

**EVIDENCIAS EXÁMEN 3|**

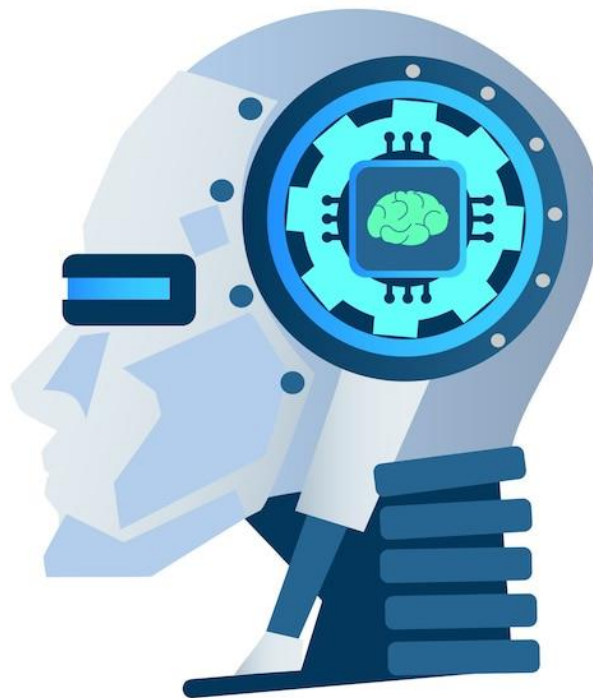
**INTEGRANTES:**

Hernández Hernández Roberto Isaac

Gonzalez Llamosas Noe Ramses

**PROFESOR:**

Ortiz Castillo Marco Antonio



**FECHA DE ENTREGA: 10/01/2025**

INSTITUTO POLITÉCNICO NACIONAL



## 1. Sea la siguiente matriz de características:

$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$
0	2	0	10	0	6	0
0	2	0	8	0	5	0
5	0	10	0	2	0	1
5	0	9	0	1	0	1.5
3	6	0	0	9	7	0
3	5	0	0	8	9	0

Realice un programa que:

- Clasifique cada uno de los datos mediante el método (visto en clase) que más adecuado sea.
- Clasifique  $[1, 0, 0, 7, 0, 3, 0]$

```
File Edit Selection View Go Run Terminal Help ← → Q Machine Learning
Exercise01.py X Exercise03.py U
Exam > Exercise01.py > ...
1 import matplotlib.pyplot as plt
2 import random
3 import numpy as np
4
5 def distancia_ecludiana(X, x_test):
6     x_test = np.array(x_test)
7     d = (X - x_test) ** 2
8     distancia = [np.sqrt(d[i][0] + d[i][1]) for i in range(len(d))]
9     return distancia
10
11 def inicializar_centroides(data, k):
12     data = np.array(data)
13     num_datos = len(data)
14
15     # Inicializar el array de centroides con tamaño k x num_caracteristicas
16     num_caracteristicas = data.shape[1]
17     centroides = np.zeros((k, num_caracteristicas))
18
19     # Seleccionar el primer centroide al azar
20     primer_centroide_idx = random.randint(0, num_datos - 1)
21     centroides[0] = data[primer_centroide_idx]
22
23     for i in range(1, k):
24         # Calcular las distancias minimas al conjunto actual de centroides
25         distancias_minimas = np.zeros(len(data))
26         for idx, punto in enumerate(data):
27             distancias = [np.linalg.norm(punto - centroides[j]) for j in range(i)]
28             distancias_minimas[idx] = min(distancias)
29
30         # Elegir el siguiente centroide con probabilidad proporcional al cuadrado de la distancia
31         distancias_cuadradas = distancias_minimas ** 2
32         probabilidades = distancias_cuadradas / distancias_cuadradas.sum()
33         siguiente_centroide_idx = np.random.choice(range(len(data)), p=probabilidades)
34         centroides[i] = data[siguiente_centroide_idx]
35
36     return centroides
37
```

```
File Edit Selection View Go Run Terminal Help Machine Learning
Exercise01.py U Exercise03.py U
Exam > Exercise01.py > ...
37
38 def Kmeans(data, k, epocas):
39     data = np.array(data)
40     num_datos, num_caracteristicas = data.shape
41
42     # Inicializar centroides usando K-Means++
43     centroides = inicializar_centroides(data, k)
44
45     # Iteraciones para calcular los clústeres y actualizar los centroides
46     for iteraciones in range(epocas):
47         clouster_asignados = [0] * num_datos
48         for i in range(num_datos):
49             distancias = [distancia_ecludiana([centroide], data[i])[0] for centroide in centroides]
50             clouster_asignados[i] = np.argmin(distancias)
51
52         # Actualizar centroides
53         Nuevos_Centroides = [[0] * num_caracteristicas for _ in range(k)]
54         puntos_por_clouster = [0] * k
55         for i in range(num_datos):
56             clouster = clouster_asignados[i]
57             puntos_por_clouster[clouster] += 1
58             for j in range(num_caracteristicas):
59                 Nuevos_Centroides[clouster][j] += data[i][j]
60
61         for clouster_index in range(k):
62             if puntos_por_clouster[clouster_index] > 0:
63                 for j in range(num_caracteristicas):
64                     Nuevos_Centroides[clouster_index][j] /= puntos_por_clouster[clouster_index]
65             else:
66                 for j in range(num_caracteristicas):
67                     Nuevos_Centroides[clouster_index][j] = centroides[clouster_index][j]
68
```

```
File Edit Selection View Go Run Terminal Help Machine Learning
Exercise01.py U Exercise03.py U
Exam > Exercise01.py > ...
38 def Kmeans(data, k, epocas):
68
69     # Verificar convergencia
70     variacion = True
71     epsilon = 1e-10
72     for i in range(k):
73         for j in range(num_caracteristicas):
74             if abs(Nuevos_Centroides[i][j] - centroides[i][j]) > epsilon:
75                 variacion = False
76                 break
77         if not variacion:
78             break
79     if variacion:
80         break
81
82     centroides = Nuevos_Centroides
83
84     # Agrupamiento final
85     clousters = [[] for _ in range(k)]
86     for i in range(num_datos):
87         clousters[clouster_asignados[i]] = clousters[clouster_asignados[i]] + [data[i]]
88
89     return centroides, clousters
90
91 def clasificar_punto(nuevo_punto, centroides):
92     distancias = [distancia_ecludiana([centroide], nuevo_punto)[0] for centroide in centroides]
93     return np.argmin(distancias)
94
95 # Datos de 7 dimensiones
96 data = [
97     [0, 2, 0, 10, 0, 6, 0],
98     [0, 2, 0, 8, 0, 5, 0],
99     [5, 0, 10, 0, 2, 0, 1],
100     [5, 0, 9, 0, 1, 0, 1.5],
101     [3, 6, 0, 0, 9, 7, 0],
102     [3, 5, 0, 0, 8, 9, 0],
103 ]
```

```
File Edit Selection View Go Run Terminal Help
Exercise01.py U X Exercise03.py U

Exam > Exercise01.py > ...
90
91 def clasificar_punto(nuevo_punto, centroides):
92     distancias = [distancia_ecludiana([centroide], nuevo_punto)[0] for centroide in centroides]
93     return np.argmin(distancias)
94
95 # Datos de 7 dimensiones
96 data = [
97     [0, 2, 0, 10, 0, 6, 0],
98     [0, 2, 0, 8, 0, 5, 0],
99     [5, 0, 10, 0, 2, 0, 1],
100    [5, 0, 9, 0, 1, 0, 1.5],
101    [3, 6, 0, 0, 9, 7, 0],
102    [3, 5, 0, 0, 8, 9, 0],
103 ]
104
105 k = 3
106 epocas = 100
107 centroides, clousters = Kmeans(data, k, epocas)
108
109 print("Centroides finales:")
110 for idx, centroide in enumerate(centroides):
111     print(f"Centroide {idx + 1}: {centroide}")
112
113 print("\nClústeres:")
114 for idx, cluster in enumerate(clousters):
115     print(f"Clúster {idx + 1}: {cluster}")
116
117 nuevo_punto = [1, 0, 0, 7, 0, 3, 0]
118 clase = clasificar_punto(nuevo_punto, centroides)
119 print(f"\nEl nuevo punto {nuevo_punto} pertenece al clúster {clase + 1}")
120
```

## Resultados en terminal

```
Exercise01.py U X
Exam > Exercise01.py > ...
4
5 def distancia_ecludiana(x, x_test):
6     x_test = np.array(x_test)
7     d = (x - x_test) ** 2
8     distancia = [np.sqrt(d[i][0] + d[i][1]) for i in range(len(d))]
9     return distancia
10
11 def inicializar_centroides(data, k):
12     data = np.array(data)
13     num_datos = len(data)
14
15     # Inicializar el array de centroides con tamaño k x num_caracteristicas

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS
● PS C:\Users\gonza\OneDrive\Documentos\Octavo Semestre\Machine Learning> cd Exam
● PS C:\Users\gonza\OneDrive\Documentos\Octavo Semestre\Machine Learning\Exam> python Exercise01.py
Centroides finales:
Centroide 1: [np.float64(3.0), np.float64(5.5), np.float64(0.0), np.float64(0.0), np.float64(8.5), np.float64(8.0), np.float64(0.0)]
Centroide 2: [np.float64(5.0), np.float64(0.0), np.float64(9.5), np.float64(0.0), np.float64(1.5), np.float64(0.0), np.float64(1.25)]
Centroide 3: [np.float64(0.0), np.float64(2.0), np.float64(0.0), np.float64(9.0), np.float64(0.0), np.float64(5.5), np.float64(0.0)]

Clústeres:
Clúster 1: [array([3., 6., 0., 0., 9., 7., 0.]), array([3., 5., 0., 0., 8., 9., 0.])]
Clúster 2: [array([ 5., 0., 10., 0., 2., 0., 1.]), array([5., 0., 9., 0., 1., 0., 1.5])]
Clúster 3: [array([ 0., 2., 0., 10., 0., 6., 0.]), array([0., 2., 0., 8., 0., 5., 0.])]

El nuevo punto [1, 0, 0, 7, 0, 3, 0] pertenece al clúster 3
❖ PS C:\Users\gonza\OneDrive\Documentos\Octavo Semestre\Machine Learning\Exam>
```

## **2. Diga con M.S.V. ¿Cómo sería el algoritmo para clasificar los datos mostrados? ¿Dónde y por qué se clasificaría el punto P?**

Dado que las muestras forman una circunferencia en capas, la implementación de una máquina de soporte vectorial (SVM) requiere un cambio de dimensión. Esto implica que, al manejar tres tipos de características distintas, podemos representar una característica en un espacio 2D, otra en un espacio 3D y la última en un espacio 4D. De esta manera, es posible clasificar los datos de manera efectiva utilizando este algoritmo.

El objetivo principal de las SVM es maximizar la distancia entre los márgenes y los vectores de soporte, lo cual se logra mediante la operación de un producto punto.

3. Considerando el perceptrón propuesto en clase, implementar el código que permite el entrenamiento de dicha neurona y la solución de alguna tabla de verdad.

```
File Edit Selection View Go Run Terminal Help
Exercise01.py U Exercise03.py U X
Exam > Exercise03.py > entrenamiento
9
10 # Función de entrenamiento
11 def entrenamiento():
12     X1 = np.array([0, 0, 1, 1])
13     X2 = np.array([0, 1, 0, 1])
14     Yd = np.array([1, 1, 1, 0])
15     l = 0.5 # Tasa de aprendizaje
16     epocas = 10
17     W = np.array([1, 0.2, 0.5]) # Pesos iniciales
18     X0 = 1 # Entrada de sesgo
19     bandera = 0
20
21     for i in range(epocas):
22         print(f"Época actual: {i + 1}")
23         print(f"Pesos iniciales: {W[0]}, {W[1]}, {W[2]}")
24
25         for j in range(len(X1)):
26             # Cálculo de la salida del perceptrón
27             z = X0 * W[0] + W[1] * X1[j] + W[2] * X2[j]
28             Yobt = activacion(z)
29
30             if Yobt == Yd[j]:
31                 bandera += 1
32                 print(f"Yd = {Yd[j]} y Yobt = {Yobt}")
33             else:
34                 # Actualización de los pesos
35                 W[0] = W[0] - l * (Yobt - Yd[j]) * X0 # Peso asociado al sesgo
36                 W[1] = W[1] - l * (Yobt - Yd[j]) * X1[j] # Peso asociado a X1
37                 W[2] = W[2] - l * (Yobt - Yd[j]) * X2[j] # Peso asociado a X2
38                 bandera = 0
39                 print(f"Yd = {Yd[j]} y Yobt = {Yobt}")
40                 print(f"Pesos actualizados: {W[0]}, {W[1]}, {W[2]}")
41
42     entrenamiento()
43
```

## Resultados en terminal

```
File Edit Selection View Go Run Terminal Help Machine Learning
Exercise01.py U Exercise03.py U X
Exam > Exercise03.py > entrenamiento
9
10 # Función de entrenamiento
11 def entrenamiento():
12     X1 = np.array([0, 0, 1, 1])
13     X2 = np.array([0, 1, 0, 1])
14     Yd = np.array([1, 1, 1, 0])
15     l = 0.5 # Tasa de aprendizaje
16     epocas = 10
17     W = np.array([1, 0.2, 0.5]) # Pesos iniciales
18     X0 = 1 # Entrada de sesgo
19     bandera = 0
20
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS COMMENTS
Yd = 1 y Yobt = 1
Yd = 0 y Yobt = 0
Época actual: 7
Pesos iniciales: 1.5, -0.8, -0.5
Yd = 1 y Yobt = 1
Yd = 1 y Yobt = 1
Yd = 1 y Yobt = 1
Yd = 0 y Yobt = 0
Época actual: 8
Pesos iniciales: 1.5, -0.8, -0.5
Yd = 1 y Yobt = 1
Yd = 1 y Yobt = 1
Yd = 1 y Yobt = 1
Yd = 0 y Yobt = 0
Época actual: 9
Pesos iniciales: 1.5, -0.8, -0.5
Yd = 1 y Yobt = 1
Yd = 1 y Yobt = 1
Yd = 1 y Yobt = 1
Yd = 0 y Yobt = 0
Época actual: 10
Pesos iniciales: 1.5, -0.8, -0.5
Yd = 1 y Yobt = 1
Yd = 1 y Yobt = 1
Yd = 1 y Yobt = 1
Yd = 0 y Yobt = 0
PS C:\Users\gonza\OneDrive\Documents\Octavo Semestre\Machine Learning\Exam>
```