
INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO (ESCOM)



# FUNDAMENTOS DE INTELIGENCIA ARTIFICIAL

## Práctica 7: K-NN

## Grupo: 4BM1


**Integrantes:**

Carmona Serrano Ian Carlo

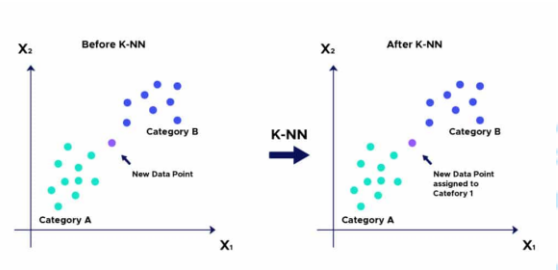
Rojas Alarcon Sergio Ulises

**Profesor:** Macario Hernández Cruz


Fecha de entrega: 01 de junio de 2023

# PRÁCTICA 7: K-NN

## Introducción




El algoritmo de las K vecinas más cercanas o K-nearest neighbors (kNN) es un algoritmo de Machine Learning que pertenece a los algoritmos de aprendizaje supervisado simples y fáciles de aplicar que pueden ser utilizados para resolver problemas de clasificación y de regresión.

Es un algoritmo no paramétrico. No paramétrico significa que el algoritmo no hace suposiciones sobre la distribución de probabilidad de los datos de la muestra.

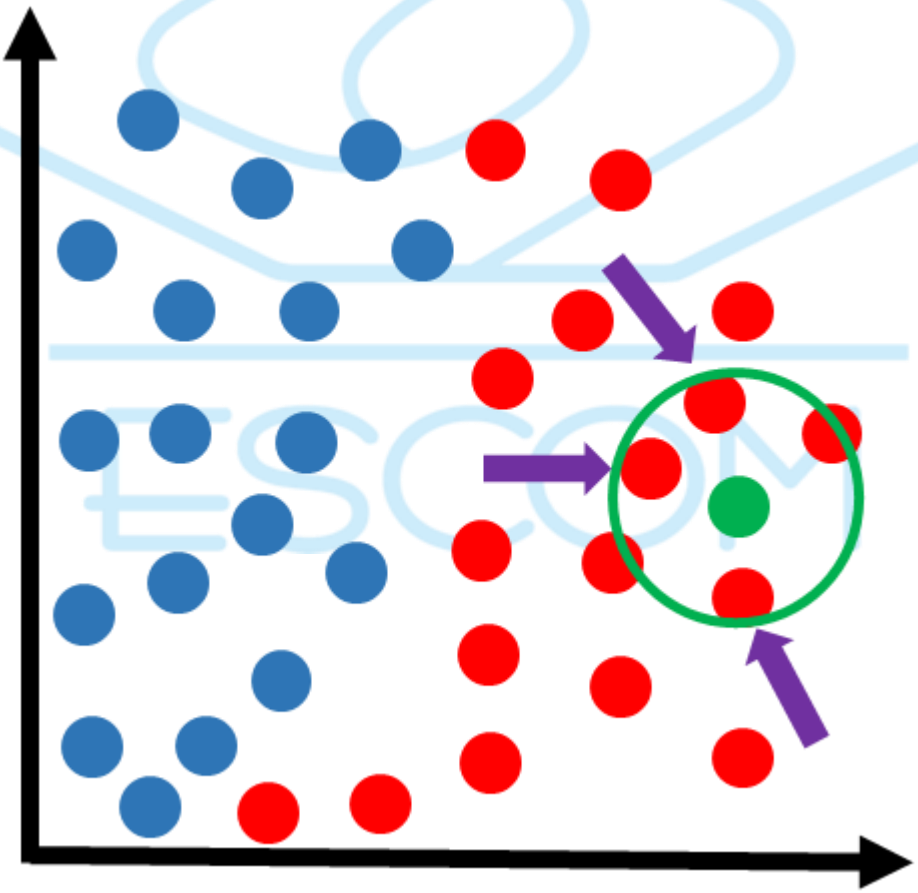
El algoritmo k-nn toma su nombre del hecho de que usa información sobre los k vecinos más cercanos de un ejemplo para clasificar ejemplos no etiquetados. La letra k es un término variable que implica que se podría usar cualquier número de vecinos más cercanos.

Si bien se puede usar para problemas de regresión o clasificación, generalmente se usa como un algoritmo de clasificación, partiendo de la suposición de que se pueden encontrar puntos similares cerca uno del otro.

Al igual que cualquier algoritmo de machine learning, k-NN tiene sus puntos fuertes y débiles. Dependiendo del proyecto y la aplicación, puede o no ser la elección correcta.

La lógica detrás del algoritmo de las K vecinas más cercanas es una de las más sencillas de todos los algoritmos de Machine Learning supervisados:

- Etapa 1: Seleccionar el número de K vecinas
- Etapa 2: Calcular la distancia Desde un punto no clasificado a otros puntos:
- Etapa 3: tomar las K vecinas más cercanas según la distancia calculada
- Etapa 4: entre las K vecinas, contar el número de puntos en cada categoría.
- Etapa 5: atribuir un nuevo punto a la categoría más presente entre las K vecinas
- Etapa 6: El modelo está listo.



Tipos de distancia

- Distancia Euclidiana

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

- Distancia Manhattan

$$\sum_{i=1}^{k}|x_i - y_i|$$

- Distancia Minkowski

$$(\sum_{i=1}^{k}(|x_i - y_i|)^q)^{\frac{1}{q}}$$

Ventajas:

- El algoritmo es simple y fácil de aplicar.
- No es necesario crear un modelo, configurar varios parámetros o formular hipótesis suplementarias.
- El algoritmo es polivalente. Puede ser utilizado para la clasificación o la regresión.
- No hace ninguna suposición sobre la distribución de los datos.
- La fase de entrenamiento es rápida.

Desventajas:

- El algoritmo se vuelve más lento a medida que el número de observaciones aumenta y las variables independientes aumentan.
- Requiere la selección de un k apropiado
- Variables cualitativas y missing data requieren un procesamiento adicional.
- Propenso al sobreajuste



**Desarrollo**

Primero vamos a importar las librerías numpy (cálculos númericos), matplotlib.pyplot (crear la gráfica), pandas (para abrir el archivo csv y manipular los datos) y sklearn.neighbors (algoritmo KNN).

```
1   import numpy as np
2   import matplotlib.pyplot as plot
3   import pandas as pd
4   from sklearn.neighbors import KNeighborsClassifier
```

Luego, se abre el archivo .csv (comma separated values) y especificamos cuáles son los datos junto con sus etiquetas

```
6   # Recupera datos
7   df = pd.read_csv('datos-estaturas.csv')
8
9   # Especifica los datos y sus etiquetas
10  X = df[['Estatura', 'Peso']]
11  y = df['Clase']
```

Las siguientes líneas de código son para crear y visualizar la gráfica de dispersión. También se especifican las características (Estatura y Peso) y las etiquetas (Clase, que incluye a “Hombre” y a “Mujer”).

```
13    # Para visualizar gráfica de dispersión
14    ax = plot.axes()
15    ax.scatter(df.loc[df['Clase'] == 'Hombre', 'Estatura'],
16               df.loc[df['Clase'] == 'Hombre', 'Peso'],
17               c="purple",
18               label="Hombre")
19    ax.scatter(df.loc[df['Clase'] == 'Mujer', 'Estatura'],
20               df.loc[df['Clase'] == 'Mujer', 'Peso'],
21               c="pink",
22               label="Mujer")
23
24    plot.xlabel("Height x Gender")
25    plot.ylabel("Weight")
26    ax.legend()
```

Ahora, se calcula el valor de k. En este caso, se calcula como la raíz cuadrada del número de muestras en los datos. Se procura que k sea impar para evitar empates.

Crea una instancia del clasificador KNeighborsClassifier con el valor de k y lo ajusta a los datos de entrenamiento (X e y) utilizando el método fit

```
28    # Calcular k
29    k = int(np.sqrt(X.shape[0]))
30
31    if k % 2 == 0:  # Hacer que K sea impar para evitar empates
32        k += 1
33
34    print(k)
35    knn = KNeighborsClassifier(n_neighbors=k)
36
37    knn.fit(X, y)
```

Aquí vamos a definir distintos colores para las 10 pruebas que vamos a realizar.

```
39    # Definir colores para las pruebas
40    colores = ['red', 'green', 'blue', 'orange', 'yellow', 'cyan', 'magenta', 'brown', 'gray', 'black']
```

Se crea un ciclo for para poder realizar las 10 pruebas. Se recopilan 10 datos de estatura y peso, después se crea un DataFrame. Después se realiza la predicción utilizando el clasificador KNN entrenado. Finalmente se muestran la predicción en consola

```
# Realizar 10 pruebas
for i in range(10):
    print(f'\nPrueba {i + 1}')
    estatura = float(input("Indica la estatura en metros: "))
    peso = float(input("Indica el peso en kilos: "))

    dfp = pd.DataFrame()
    dfp['Estatura'] = [estatura]
    dfp['Peso'] = [peso]

    ax.scatter(dfp['Estatura'],
               dfp['Peso'],
               c=colores[i],
               marker="o")

    prediccion = knn.predict(dfp)

    ax.text(dfp['Estatura'][0],
            dfp['Peso'][0],
            str(i + 1),
            fontsize=9,
            color='black',
            ha='center')

    print('Con los datos:')
    print(dfp)
    print('La categoría predicha es:')
    print(prediccion)
```

Finalmente, mostramos la gráfica de dispersión.

```
# Muestra gráfica de dispersión con distintivos de las pruebas
ax.legend()
plot.show()
```

## Demostración de Resultados

| | Estatura | Peso | Clase |
|---|---|---|---|
| 2 | 1.62 | 66 | Mujer |
| 3 | 1.88 | 72 | Hombre |
| 4 | 1.49 | 49 | Mujer |
| 5 | 1.65 | 70 | Hombre |
| 6 | 1.56 | 60 | Mujer |
| 7 | 1.72 | 58 | Hombre |
| 8 | 1.61 | 62 | Mujer |
| 9 | 1.65 | 63 | Mujer |
| 10 | 1.57 | 57 | Mujer |
| 11 | 1.9 | 75 | Hombre |
| 12 | 1.7 | 72 | Hombre |
| 13 | 1.6 | 47 | Mujer |
| 14 | 1.83 | 76 | Hombre |
| 15 | 1.53 | 57 | Mujer |
| 16 | 1.73 | 68 | Hombre |
| 17 | 1.71 | 71 | Mujer |
| 18 | 1.94 | 85 | Hombre |
| 19 | 1.84 | 82 | Hombre |
| 20 | 1.59 | 56 | Mujer |
| 21 | 1.74 | 68 | Hombre |
| 22 | 1.62 | 55 | Mujer |
| 23 | 1.85 | 80 | Hombre |
| 24 | 1.47 | 50 | Mujer |
| 25 | 1.73 | 68 | Mujer |
| 26 | 1.56 | 53 | Mujer |
| 27 | 1.62 | 65 | Hombre |
| 28 | 1.71 | 70 | Mujer |
| 29 | 1.89 | 88 | Hombre |
| 30 | 1.76 | 74 | Hombre |
| 31 | 1.68 | 55 | Mujer |
| 32 | 1.63 | 64 | Mujer |
| 33 | 1.79 | 71 | Hombre |
| 34 | 1.54 | 57 | Mujer |
| 35 | 1.67 | 65 | Hombre |
| 36 | 1.67 | 74 | Mujer |
| 37 | 1.71 | 73 | Hombre |
| 38 | 1.64 | 60 | Mujer |
| 39 | 1.5 | 52 | Mujer |
| 40 | 1.75 | 75 | Hombre |
| 41 | 1.7 | 64 | Mujer |

**Ilustración 1**. Los 40 datos para entrenar al algoritmo

| 1 | Estatura | Peso | Clase |
|---|---|---|---|
| 2 | 1.75 | 68 | Hombre |
| 3 | 1.81 | 69 | Hombre |
| 4 | 1.58 | 60 | Mujer |
| 5 | 1.49 | 52 | Mujer |
| 6 | 1.74 | 63 | Hombre |
| 7 | 1.61 | 58 | Mujer |
| 8 | 1.96 | 89 | Hombre |
| 9 | 1.58 | 52 | Hombre |
| 10 | 1.51 | 56 | Mujer |
| 11 | 1.71 | 65 | Mujer |

**Ilustración 2**. Los 10 datos de prueba

```
Prueba 1
Indica la estatura en metros: 1.75
Indica el peso en kilos: 68
Con los datos:
   Estatura  Peso
0      1.75  68.0
La categoría predicha es:
['Mujer']

Prueba 2
Indica la estatura en metros: 1.81
Indica el peso en kilos: 69
Con los datos:
   Estatura  Peso
0      1.81  69.0
La categoría predicha es:
['Hombre']
```

```
Prueba 3
Indica la estatura en metros: 1.58
Indica el peso en kilos: 60
Con los datos:
   Estatura  Peso
0      1.58  60.0
La categoría predicha es:
['Mujer']

Prueba 4
Indica la estatura en metros: 1.49
Indica el peso en kilos: 52
Con los datos:
   Estatura  Peso
0      1.49  52.0
La categoría predicha es:
['Mujer']
```

```
Prueba 5
Indica la estatura en metros: 1.74
Indica el peso en kilos: 63
Con los datos:
   Estatura  Peso
0      1.74  63.0
La categoría predicha es:
['Mujer']

Prueba 6
Indica la estatura en metros: 1.61
Indica el peso en kilos: 58
Con los datos:
   Estatura  Peso
0      1.61  58.0
La categoría predicha es:
['Mujer']
```

```
Prueba 7
Indica la estatura en metros: 1.96
Indica el peso en kilos: 89
Con los datos:
   Estatura  Peso
0      1.96  89.0
La categoría predicha es:
['Hombre']

Prueba 8
Indica la estatura en metros: 1.58
Indica el peso en kilos: 52
Con los datos:
   Estatura  Peso
0      1.58  52.0
La categoría predicha es:
['Mujer']
```

```
Prueba 9
Indica la estatura en metros: 1.51
Indica el peso en kilos: 56
Con los datos:
   Estatura  Peso
0      1.51  56.0
La categoría predicha es:
['Mujer']

Prueba 10
Indica la estatura en metros: 1.71
Indica el peso en kilos: 65
Con los datos:
   Estatura  Peso
0      1.71  65.0
La categoría predicha es:
['Mujer']
```

**Ilustración 3-7.** Ingresando los 10 datos de prueba. Para cada una se realiza la predicción.

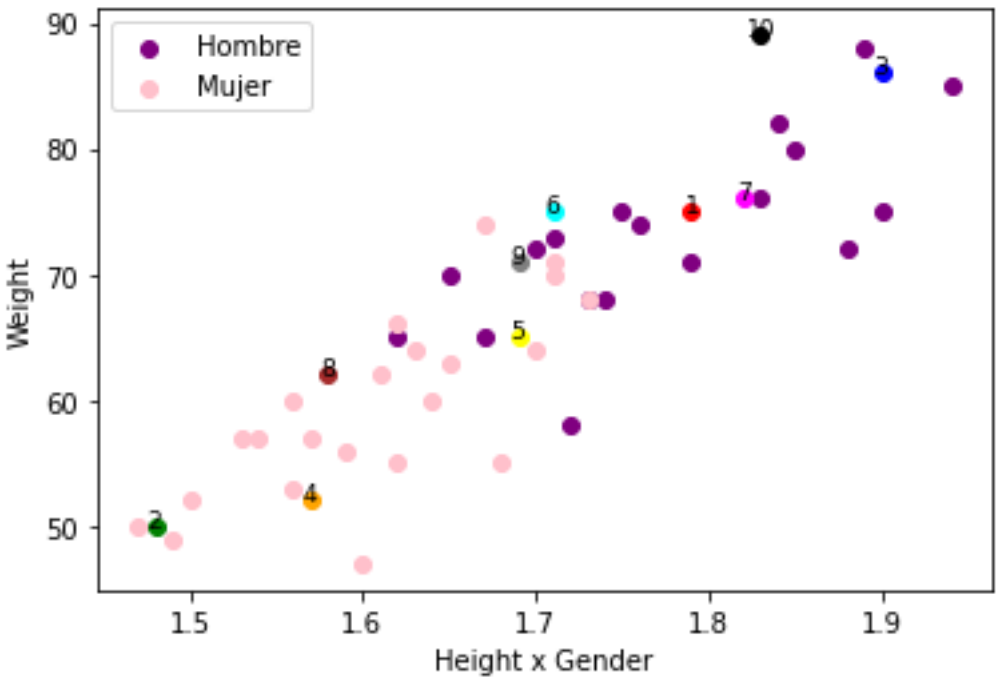



**Ilustración 8.** Gráfica de dispersión, cada muestra está numerada y tienen distintos colores para distinguirlas.

## Discusión de Resultados

La falta de datos suficientes afecta la precisión del algoritmo de clasificación. Al tener pocos datos, el algoritmo no puede aprender correctamente y, como resultado, clasifica erróneamente algunas clases. En este caso, el algoritmo clasificó correctamente 7 clases y falló en otras 3 (prueba 1, prueba 5 y prueba 8), lo que significa que tiene una eficiencia del 70%. Con más datos y un entrenamiento adecuado, el algoritmo podría mejorar su precisión.

## Conclusiones

Al tratarse de uno de los algoritmos más simples de Machine Learning, es muy implementado por los desarrolladores de sistemas basados en el aprendizaje, intuitivos e inteligentes que pueden efectuar y tomar pequeñas decisiones solos.

El algoritmo K-NN tiene múltiples aplicaciones, principalmente dentro de la clasificación, como el preprocesamiento de datos, los motores de recomendación, finanzas, cuidado de la salud y reconocimiento de patrones.

Para efectos de esta práctica, fue necesario recabar 50 datos de varias personas (estatura, peso y género). De esos 50, íbamos a usar 40 datos para entrenar al algoritmo K-NN y los 10 datos restantes servirían para hacer las pruebas.

Respecto al código que se nos proporcionó, solamente fue necesario agregarle un ciclo for para poder recabar 10 muestras (en lugar de 1, como era originalmente) y le agregamos unas cuantas líneas de código para distinguir cada una de las muestras de prueba (con colores).

Esto hace que sea aún más práctico para el aprendizaje y el desarrollo y puede servir para toda industria que utilice sistemas, soluciones o servicios inteligentes.

**Referencias**


- DataScientest. (2022, 28 diciembre). ¿Qué es el algoritmo KNN? Recuperado 1 de junio de 2023, de https://datascientest.com/es/que-es-el-algoritmo-knn
- Díaz, R. (s. f.). Algoritmo KNN – cómo funciona y ejemplos en Python. The Machine Learners. Recuperado 1 de junio de 2023, de https://www.themachinelearners.com/algoritmo-knn/
- El algoritmo K-NN y su importancia en el modelado de datos. (2020, 1 septiembre). Merkle. Recuperado 1 de junio de 2023, de https://www.merkle.com/es/es/blog/algoritmo-knn-modelado-datos
- Gómez Servan, W. J. (2022, 5 febrero). K-Vecinos mas cercanos(KNN). RPubs. Recuperado 1 de junio de 2023, de https://rpubs.com/WaldoGomez10/algoritmo_knn_modulo3
- IBM. (s. f.). ¿Qué es KNN? Recuperado 1 de junio de 2023, de https://www.ibm.com/mx-es/topics/knn
- Na8. (2018, 10 julio). Clasificar con K-Nearest-Neighbor ejemplo en Python. Aprende Machine Learning. Recuperado 1 de junio de 2023, de https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/