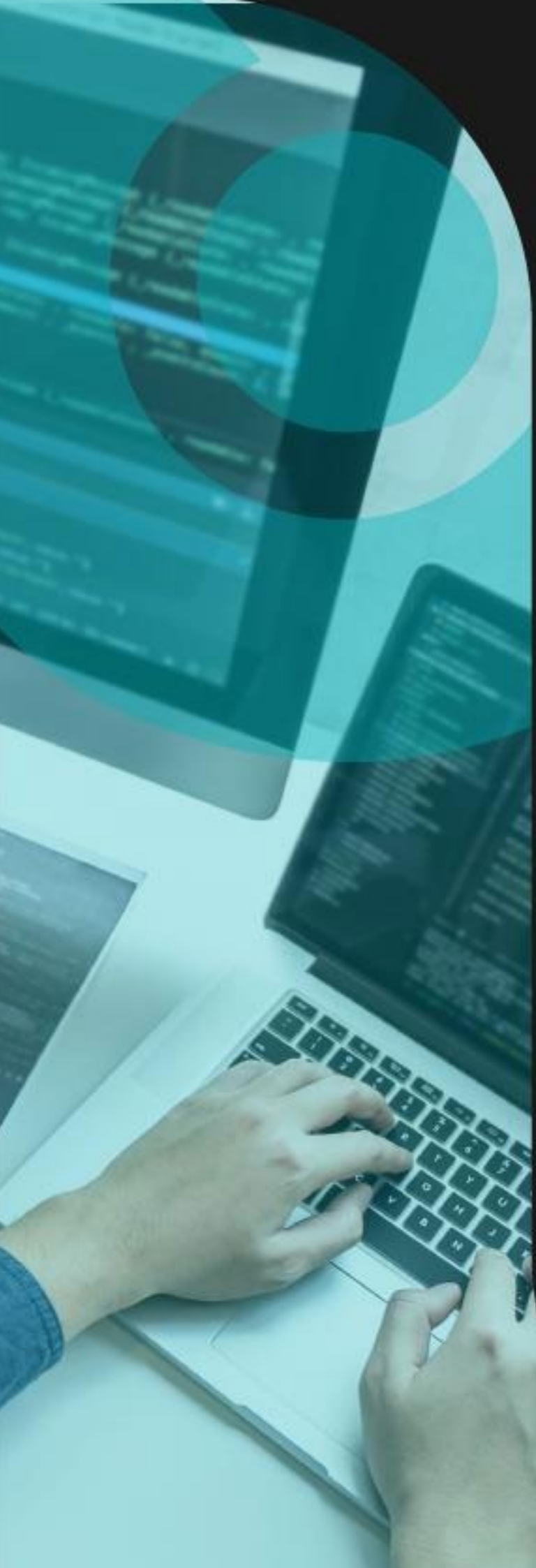


AWAKELAB

BASECAMP

Ciencia de Datos

Módulo: Aprendizaje de Máquina Supervisado

Aprendizaje Esperado

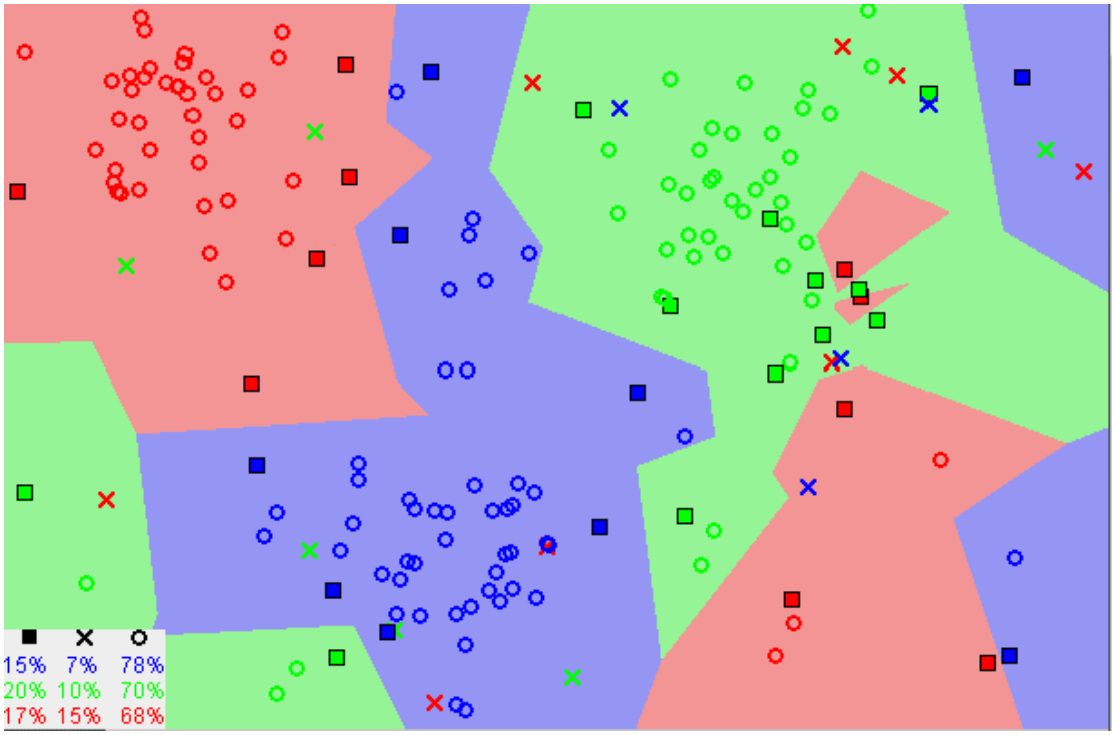
5. Elabora un modelo predictivo aplicando el algoritmo K-NN para resolver un problema de clasificación utilizando lenguaje Python.

K-NN

El algoritmo K - vecinos más cercanos (K-nearest neighbors - KNN) es un algoritmo de aprendizaje automático supervisado simple y fácil de implementar que se puede usar para resolver problemas de clasificación y regresión.

Un algoritmo **de aprendizaje automático supervisado** (a diferencia de un algoritmo de aprendizaje automático no supervisado) es uno que se basa en datos de entrada etiquetados para aprender una función que produce una salida adecuada cuando se le dan nuevos datos no etiquetados.

El algoritmo KNN asume que existen cosas similares muy cerca. En otras palabras, las cosas similares están cerca unas de otras.



Observe en la imagen de arriba que la mayoría de las veces, los puntos de datos similares están cerca uno del otro. El algoritmo KNN depende de que esta suposición sea lo suficientemente cierta para que el algoritmo sea útil. KNN captura la idea de similitud (a veces llamada distancia, proximidad o cercanía) con algunas matemáticas que podríamos haber aprendido en nuestra infancia: calcular la distancia entre puntos en un gráfico.

En qué consiste

Hay otras formas de calcular la distancia, y una forma podría ser preferible según el problema que estemos resolviendo. Sin embargo, la distancia en línea recta (también llamada distancia euclidiana) es una opción popular y familiar.

El algoritmo:

1. Cargar los datos.
2. Inicialice K a su número elegido de vecinos.
3. Para cada ejemplo en los datos:
    3.1 Calcular la distancia entre el ejemplo de consulta y el ejemplo actual a partir de los datos.

3.2 Añadir la distancia y el índice del ejemplo a una colección ordenada.

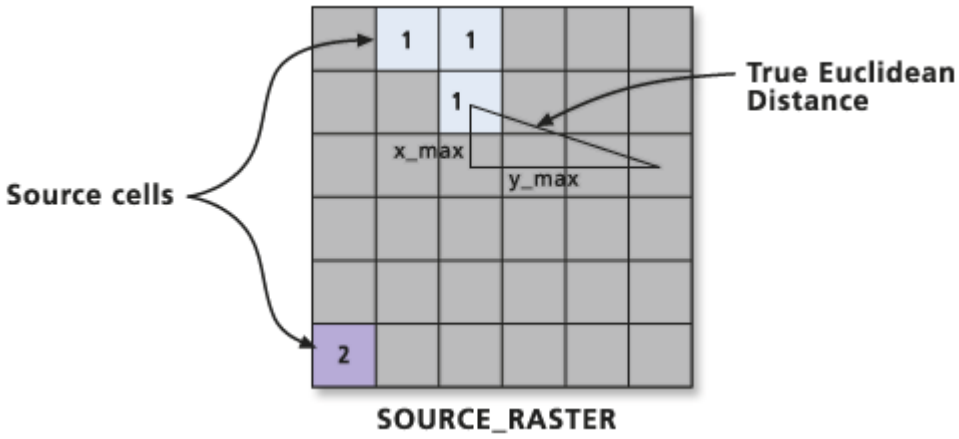
4. Ordene la colección ordenada de distancias e índices de menor a mayor (en orden ascendente) por las distancias.
5. Elija las primeras K entradas de la colección ordenada.
6. Obtener las etiquetas de las K entradas seleccionadas.
7. Si es una regresión, devuelva la media de las etiquetas K.
8. Si clasificación, devolver el modo de las etiquetas K.

Concepto de Escalamiento de Datos

Es una forma de preprocesamiento de datos numéricos, es decir, datos que consisten en números, a diferencia de categorías o cadenas, por ejemplo; centrar una variable es restar la media de la variable de cada punto de datos para que la media de la nueva variable sea 0; escalar una variable es multiplicar cada punto de datos por una constante para alterar el rango de los datos.

Concepto de Distancia Euclidiana

La distancia euclidiana se calcula desde el centro de la celda de origen hasta el centro de cada una de las celdas circundantes. La distancia euclidiana se calcula en cada una de las herramientas de distancia. Conceptualmente, el algoritmo euclidiano funciona del siguiente modo: para cada celda, la distancia a cada celda de origen se determina al calcular la hipotenusa con x_max y y_max como los otros dos lados del triángulo. Este cálculo deriva la verdadera distancia euclidiana, en vez de la distancia de la celda. Se determina la distancia más corta a un origen, y si es menor que la distancia máxima especificada, el valor se asigna a la ubicación de la celda en el ráster de salida.



Los valores de salida para el ráster de distancia euclidiana son los valores de distancia de puntos flotantes. Si la celda está a la misma distancia de dos o más orígenes, la celda se asigna al origen que se encontró primero en el proceso de escaneo. No se puede controlar este proceso de escaneo.

La descripción anterior solo es una representación conceptual de cómo se derivan los valores. El algoritmo real calcula la información mediante un proceso secuencial de dos escaneos. La velocidad de la herramienta en este proceso es independiente del número de celdas de origen, la distribución de celdas de origen y la distancia máxima especificada. El único factor que influye en la velocidad con la que se ejecuta la herramienta es el tamaño del ráster. El tiempo de cómputo es linealmente proporcional al número de celdas en la ventana Análisis.

Elegir el valor correcto para K

Para seleccionar la K adecuada para sus datos, ejecutamos el algoritmo KNN varias veces con diferentes valores de K y elegimos la K que reduce la cantidad de errores que encontramos mientras mantenemos la capacidad del algoritmo para hacer predicciones precisas cuando recibe datos que no tiene.

Aquí hay algunas cosas a tener en cuenta:

1. A medida que disminuimos el valor de K a 1, nuestras predicciones se vuelven menos estables. Solo piense por un minuto, imagine K=1 y tenemos un punto de consulta rodeado por varios rojos y uno verde (estoy pensando en la esquina superior izquierda de la gráfica

coloreada arriba), pero el verde es el único vecino más cercano. Razonablemente, pensaríamos que el punto de consulta probablemente sea rojo, pero debido a que K=1, KNN predice incorrectamente que el punto de consulta es verde.

2. Inversamente, a medida que aumentamos el valor de K, nuestras predicciones se vuelven más estables debido a la votación mayoritaria/promedio y, por lo tanto, es más probable que hagamos predicciones más precisas (hasta cierto punto). Eventualmente, comenzamos a presenciar un número creciente de errores. Es en este punto que sabemos que hemos llevado el valor de K demasiado lejos.

3. En los casos en los que estamos tomando una mayoría de votos (por ejemplo, eligiendo el modo en un problema de clasificación) entre las etiquetas, generalmente hacemos que K sea un número impar para tener un desempate.

Ventajas	**Desventajas**
1. El algoritmo es simple y fácil de implementar.	1. El algoritmo se vuelve significativamente más lento a medida que aumenta el número de ejemplos y/o predictores/variables independientes.
2. No hay necesidad de construir un modelo, ajustar varios parámetros o hacer suposiciones adicionales.	

3. El algoritmo es versátil. Puede usarse para clasificación, regresión y búsqueda (como veremos en la siguiente sección).	

```
# Implementación en Python

from collections import Counter

import math


def knn(data, query, k, distance_fn, choice_fn):

    neighbor_distances_and_indices = []


    # 3. Ejemplo

    for index, example in enumerate(data):

        # 3.1 Calcular la distancia entre el ejemplo
de la consulta y el actual

        # ejemplo de los datos.

        distance = distance_fn(example[:-1], query)


        # 3.2 Añade la distancia y el índice del
ejemplo a una colección ordenada


neighbor_distances_and_indices.append((distance,
index))
```

```
    # 4. Ordena la colección ordenada de distancias e
índices de

    # menor a mayor (en orden ascendente) por las
distancias

    sorted_neighbor_distances_and_indices =
sorted(neighbor_distances_and_indices)



    # 5. Elige las primeras K entradas de la
colección ordenada

    k_nearest_distances_and_indices =
sorted_neighbor_distances_and_indices[:k]



    # 6. Obtener las etiquetas de las entradas K
seleccionadas

    k_nearest_labels = [data[i][-1] for distance, i
in k_nearest_distances_and_indices]




    # 7. Si la regresión (elección_fn = media),
devuelve la media de las etiquetas K

    # 8. Si la clasificación (choice_fn = mode),
devuelve la moda de las K etiquetas

    return k_nearest_distances_and_indices ,
choice_fn(k_nearest_labels)
```

```
def mean(labels):
    return sum(labels) / len(labels)



def mode(labels):
    return Counter(labels).most_common(1)[0][0]



def euclidean_distance(point1, point2):
    sum_squared_distance = 0
    for i in range(len(point1)):
        sum_squared_distance += math.pow(point1[i] -
point2[i], 2)
    return math.sqrt(sum_squared_distance)



def main():
    '''
    # Regression Data
    #
    # Column 0: height (inches)
```

```
    # Column 1: weight (pounds)
    '''
    reg_data = [
       [65.75, 112.99],
       [71.52, 136.49],
       [69.40, 153.03],
       [68.22, 142.34],
       [67.79, 144.30],
       [68.70, 123.30],
       [69.80, 141.49],
       [70.01, 136.46],
       [67.90, 112.37],
       [66.49, 127.45],
    ]


    # pregunta:
    # Teniendo en cuenta los datos que tenemos, ¿cuál
es la mejor estimación del peso de una persona que
mide 60 pulgadas?
    reg_query = [60]
    reg_k_nearest_neighbors, reg_prediction = knn(
        reg_data, reg_query, k=3,
distance_fn=euclidean_distance, choice_fn=mean
```

```
    )


    '''
    # Classification Data
    #
    # Column 0: age
    # Column 1: likes pineapple
    '''
    clf_data = [
       [22, 1],
       [23, 1],
       [21, 1],
       [18, 1],
       [19, 1],
       [25, 0],
       [27, 0],
       [29, 0],
       [31, 0],
       [45, 0],
    ]
    # pregunta:
```

```
    # Teniendo en cuenta los datos que tenemos, ¿a
una persona de 33 años le gusta la piña en su pizza?

    clf_query = [33]

    clf_k_nearest_neighbors, clf_prediction = knn(

        clf_data, clf_query, k=3,
distance_fn=euclidean_distance, choice_fn=mode

    )



if __name__ == '__main__':

    main()
```

Resumen

El algoritmo k-vecinos más cercanos (KNN) es un algoritmo de aprendizaje automático simple y supervisado que se puede usar para resolver problemas de clasificación y regresión. Es fácil de implementar y comprender, pero tiene la gran desventaja de volverse significativamente más lento a medida que crece el tamaño de los datos en uso.

KNN funciona encontrando las distancias entre una consulta y todos los ejemplos en los datos, seleccionando los ejemplos numéricos especificados (K) más cercanos a la consulta, luego vota por la etiqueta más frecuente (en el caso de la clasificación) o promedia las etiquetas (en el caso de la regresión).

En el caso de la clasificación y la regresión, vimos que elegir la K correcta para nuestros datos se hace probando varias K y eligiendo la que mejor funciona.
Finalmente, vimos un ejemplo de cómo el algoritmo KNN podría usarse en sistemas de recomendación, una aplicación de búsqueda KNN.

Referencias

[1] KNN

https://aprendeia.com/k-vecinos-mas-cercanos-teoria-machine-learning/#:~:text=K%20vecinos%20m%C3%A1s%20cercanos%20es,y%20la%20detecci%C3%B3n%20de%20intrusos.

[2] Método de clásificación

https://medium.com/soldai/m%C3%A9todo-de-los-k-vecinos-m%C3%A1s-cercanos-f8231c28f7c7

[3] K nearest neighbors

https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Material Complementario

[1] Video – KNN

https://www.youtube.com/watch?v=FHHuo7xEeo4

[2] Teoria

https://www.youtube.com/watch?v=FHHuo7xEeo4&feature=youtu.be