UNIVERSIDAD NACIONAL DE AGUSTÍN

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

Kd Tree - KNN

Alumno:
HERMOZA LOAYZA, MIGUEL ÁNGEL
HERRERA COOPER, MIGUEL ALEXANDER

Docente:
Machaca Arceda VICENTE

Arequipa - Perú

Índice

Índice de figuras

1. Resumen

En este documento presentamos la implementación función Range Query y haremos uso del Algoritmo Knn para clasificar imagenes.

Presentamos un algoritmo K-D Tree es una estructura de datos espacial común en muchos métodos de análisis de datos,se pueden ejecutar cuando se ha indexado todo el conjunto de datos, es decir, no están en línea. Incluso las pocas implementaciones en línea no son progresivas en el sentido de que el tiempo para indexar los datos entrantes no está limitado y puede exceder la latencia requerida por los sistemas progresivos. Esta latencia restringe significativamente la interactividad de los sistemas de visualización, especialmente cuando se trata de datos a gran escala. Mejoramos los k-dtrees tradicionales para el vecino progresivo más cercano al más cercano, lo que permite consultas KNN rápidas e indexa continuamente nuevos lotes de datos cuando es necesario. Siguiendo el paradigma de computación progresiva, nuestro k-dtree progresivo está limitado en el tiempo, lo que permite a los analistas acceder a resultados continuos dentro de una latencia interactiva. También presentamos puntos de referencia de rendimiento para comparar en línea y progresivo-dtrees.

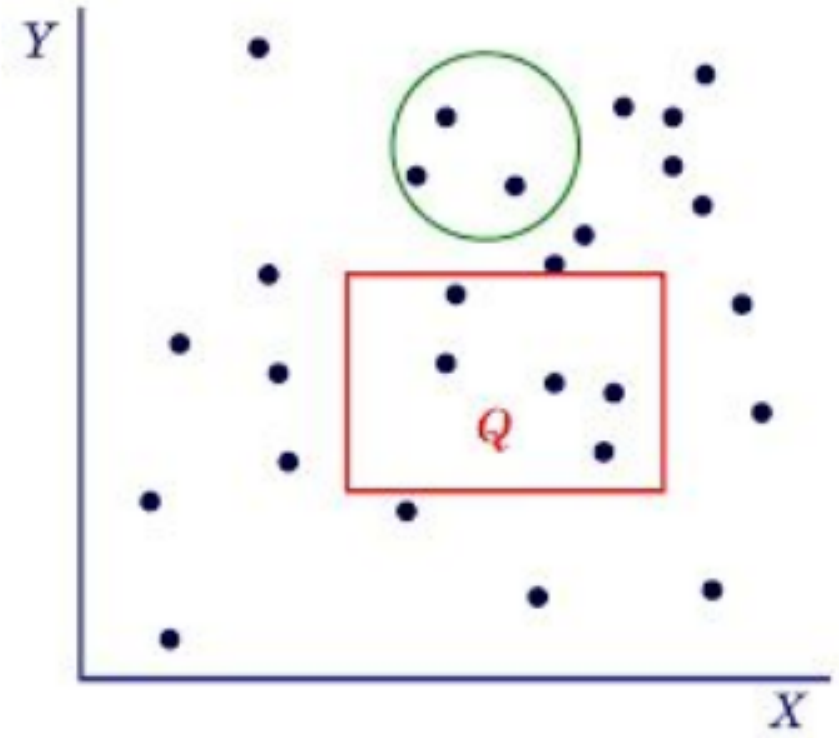
2. Introducción

La idea es realmente sencilla: el algoritmo clasifica cada dato nuevo en el grupo que corresponda, según tenga k vecinos más cerca de un grupo o de otro. Es decir, calcula la distancia del elemento nuevo a cada uno de los existentes, y ordena dichas distancias de menor a mayor para ir seleccionando el grupo al que pertenecer. Este grupo será, por tanto, el de mayor frecuencia con menores distancias.

El K-NN es un algoritmo de aprendizaje supervisado, es decir, que a partir de un juego de datos inicial su objetivo será el de clasificar correctamente todas las instancias nuevas. El juego de datos típico de este tipo de algoritmos está formado por varios atributos descriptivos y un solo atributo objetivo (también llamado clase).

3. Range Query

- La función RangeQuery , devuelve un conjunto de puntos a partir de una área de búsqueda, por ejemplo en la Figura de abajo, el círculo y el rectángulo representan el rango de búsqueda. La estructura debe ser capaz de retornar el conjunto de puntos dentro de dicho rango sin consultar todos los puntos.



3.1. Ejercicio 1

Implemente la función rangeQueryCircle en su kdtree:

```
function range_query_circle ( node , center , radio , queue ,
    depth = 0 ){
  if (node==null) return null;

  var axis = node.axis ;
  var nb = null;
  var ob = null;

  if (center[axis] < node.point[axis]){
    nb=node.left;
    ob=node.right;
  } else {
    nb=node.right;
    ob=node.left;
  }

  var best=closer_point(center,node,range_query_circle(nb,center
    ,radio,queue,depth+1));

  if(Math.abs(center[axis]-node.point[axis]) <= radio ||
    distanceSquared(center,best.point) > Math.abs(center[axis]-
    node.point[axis])){

    if(distanceSquared(center,node.point) <= radio){

      queue.push(node.point);
    }

    best=closer_point(center,best,range_query_circle(ob,center,
    radio,queue,depth+1));
  }

  return best ;
}
```

circleQ.js

3.2. Ejercicio 2

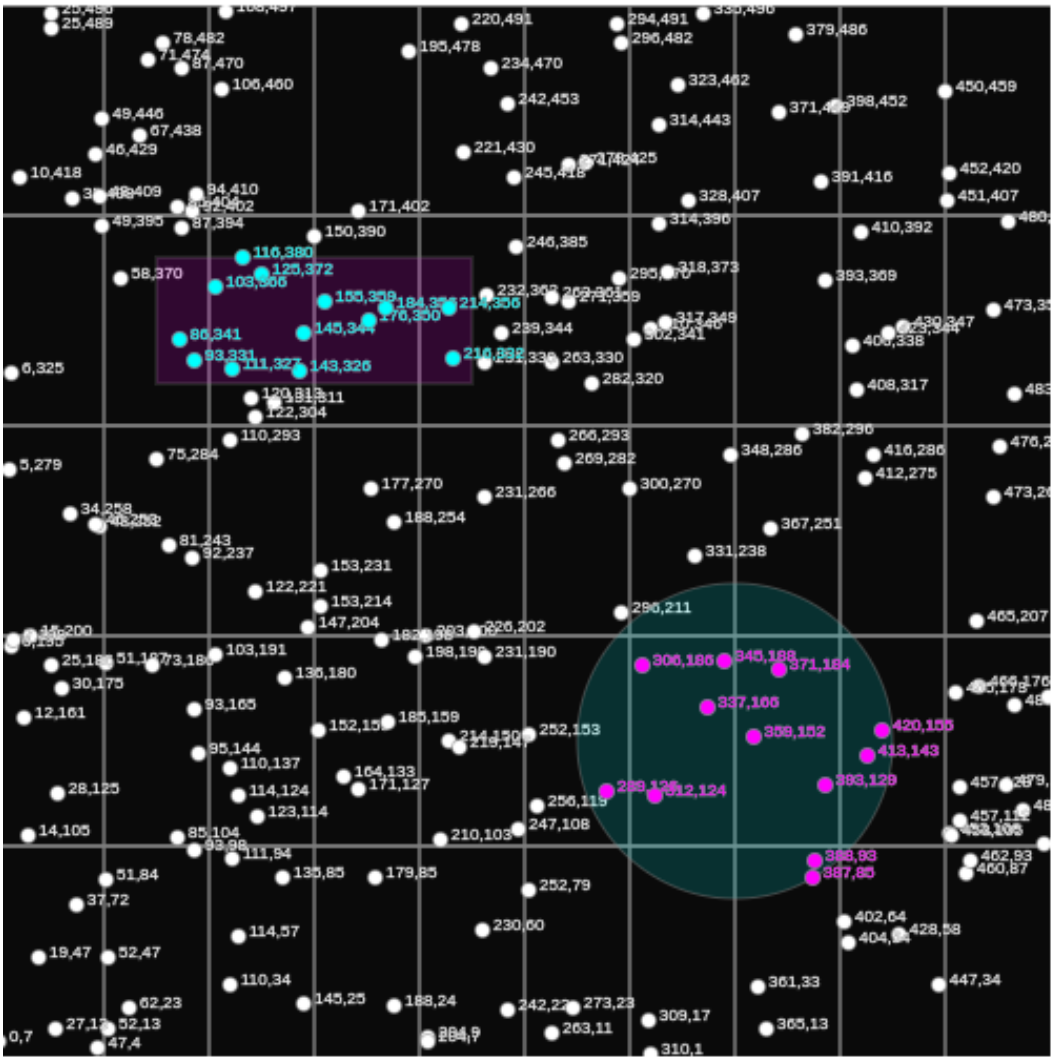
Implemente la función rangeQueryRectangle en su kdtree:

```
function range_query_rect ( node , center , hug , queue , depth
    = 0 ){
  if (node==null) return null;

  var axis = node.axis ;
  var nb = null;
  var ob = null;

  if (center[axis]<node.point[axis]){
    nb=node.left;
    ob=node.right;
  } else {
    nb=node.right;
    ob=node.left;
  }
  var best=closer_point(center,node,range_query_rect(nb,center,
   hug,queue,depth+1));

  if(Math.abs(center[axis]-node.point[axis])<=hug[axis]*2 ||
   distanceSquared(center,best.point)>Math.abs(center[axis]-node
   .point[axis])){

    if(Math.abs(center[0]-node.point[0])<=hug[0] && Math.abs(
   center[1]-node.point[1])<=hug[1]){

      queue.push(node.point);
    }
    best=closer_point(center,best,range_query_rect(ob,center,hug
   ,queue,depth+1));
  }

  return best ;
}
```

rectangleQ.js

3.3. Resultados

- Usando la función rangeQueryCircle y rangeQueryRectangle obtenemos los siguientes resultados determinando el conjunto de puntos dentro del rango de búsqueda de cada estructura.



4. Knn

4.1. Definición

La idea básica sobre la que se fundamenta este paradigma es que un nuevo caso se va a clasificar en la clase mas frecuente a la que pertenecen sus K vecinos mas cercanos. El paradigma se fundamenta por tanto en una idea muy simple e intuitiva, lo que unido a su fácil implementación hace que sea un paradigma clasificatorio muy extendido.

Después de introducir el algoritmo K-NN básico y presentar algunas variantes del mismo, en este tema se estudian métodos para la selección de prototipos

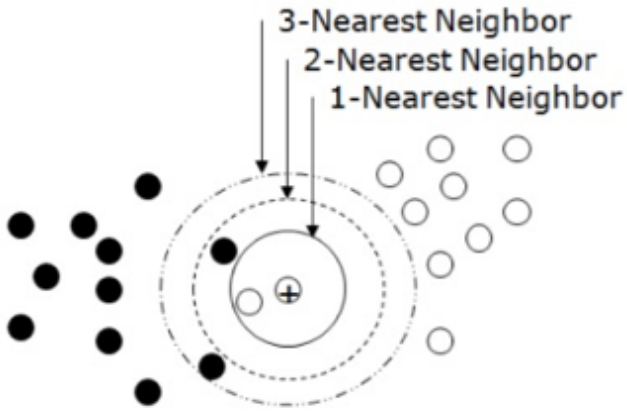
4.2. ¿Cómo funciona?

En contraste con otros algoritmos de aprendizaje supervisado, K-NN no genera un modelo fruto del aprendizaje con datos de entrenamiento, sino que el aprendizaje sucede en el mismo momento en el que se prueban los datos de test. A este tipo de algoritmos se les llama lazy learning methods.

Destacar que K-NN es muy sensible a:

- La variable k, de modo que con valores distintos de k podemos obtener resultados también muy distintos. Este valor suele fijarse tras un proceso de pruebas con varias instancias.
- La métrica de similitud utilizada, puesto que esta influirá, fuertemente, en las relaciones de cercanía que se irán estableciendo en el proceso de construcción del algoritmo. La métrica de distancia puede llegar a contener pesos que nos ayudarán a calibrar el algoritmo de clasificación, convirtiéndola, de hecho, en una métrica personalizada.

Vemos cómo influye el valor de k:



- Para k = 1 el algoritmo clasificará la bola con signo + como blanca
- Para k = 2 el algoritmo no tiene criterio para clasificar la bola con signo +
- Para k ¿= 3 el algoritmo clasificará la bola con signo + como negra

Su mayor debilidad es la lentitud en el proceso de clasificación puesto que su objetivo no es obtener un modelo optimizado, sino que cada instancia de prueba es comparada contra todo el juego de datos de entrenamiento y, será la bondad de los resultados lo que determinará el ajuste de aspectos del algoritmo como el propio valor k, el criterio de selección de instancias para formar parte del juego de datos "D" de entrenamiento o la propia métrica de medida de similitud.

Finalmente, conoceremos la precisión del algoritmo a través de una evaluación de la matriz de confusión. En ella, se valorarán los verdaderos positivos, los falsos positivos, los positivos incorrectos, etc.

		Clase verdadera		
		C1	C2	
Clase predicha	C1	a	b	p_1
	C2	c	d	p_2
		π_1	π_2	

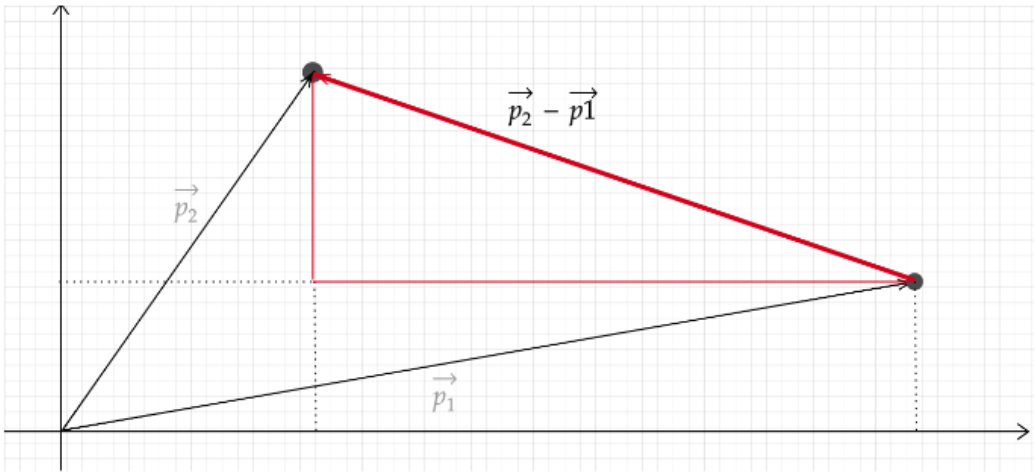
5. Imagen intuición de clasificación con KNN

Cada punto en el ejemplo de espacio 2D de KNN se puede representar como un vector (por ahora, una lista de dos números). Todos esos vectores apilados verticalmente formarán una matriz que representa todos los puntos en el plano 2D.

$$\begin{array}{cccc} P & C_x & C_y & color \\ \overrightarrow{p_1} & [10 & 20] & greeen \\ \overrightarrow{p_2} & [11 & 30] & orange \\ \dots & \dots & \dots & \dots \\ \overrightarrow{p_n} & [90 & 45] & green \end{array} \rightarrowtail \begin{array}{cc} X & Y \\ 10 & 20 \\ 11 & 30 \\ \vdots & \vdots \\ 90 & 40 \end{array} \begin{array}{c} Y \\ green \\ orange \\ \vdots \\ green \end{array} \rightarrowtail \begin{array}{c} X \\ \overrightarrow{p_1} \\ \overrightarrow{p_2} \\ \vdots \\ \overrightarrow{p_n} \end{array} \begin{array}{c} Y \\ green \\ orange \\ \vdots \\ green \end{array}$$

$$C_x = x\ cordinate\ of\ a\ point$$
$$C_y = y\ coordinate\ of\ a\ point$$

En un plano 2D, si cada punto es un vector, la distancia euclidiana (escalar) se puede derivar del siguiente diagrama:



Esto, esencialmente, es una representación geométrica de la ecuación de la Distancia Euclidiana:

$$d(p_1,\ p_2)\ =\ \sqrt{(x_1\ -\ x_2)^2\ +(y_1\ -\ y_2)^2}\ where\ p_1 = (x_1,\ y_1)\ and\ p_2\ =(x_2,\ y_2)$$

$$d(p_1,\ p_2)\ =\ \sqrt{\left(p_1^1\ -p_2^1\right)^2\ +\ \left(p_1^2\ -\ p_2^2\right)^2}$$

Ahora piense en una imagen de 32 x 32 . Tenemos un total de 32 x 32 = 1024 píxeles.
Entonces, si tenemos que representar una imagen usando una estructura que la computadora pueda entender, tendremos un vector grande de 1024 elementos de largo.

Pero, ¿cuál será el valor de los números?
Por supuesto, el color que tiene el píxel.

El color de un píxel se puede representar utilizando 3 valores de color (supongamos que el canal de color RGB) por lo que un píxel contiene una lista de 3 valores (1 valor para r, 1 valor para gy 1 valor para b del canal RGB). Codificamos esto en una estructura similar, pero creamos 1024 valores de r, seguidos de 1024 valores de g, seguidos de 1024 valores de b. Eso es un total de 3072 valores en un vector.

Si no tenemos más de dos objetos, entonces tenemos una longitud de vector de 3072 apilados verticalmente. La matriz se vería así:

$$\begin{bmatrix} px_r^{1,1} & px_r^{1,2} & \dots & px_r^{32,32} & px_g^{1,1} & px_g^{1,2} & \dots & px_g^{32,32} & px_b^{1,1} & px_b^{1,2} & \dots & px_b^{32,32} \\ px_r^{1,1} & px_r^{1,2} & \dots & px_r^{32,32} & px_g^{1,1} & px_g^{1,2} & \dots & px_g^{32,32} & px_b^{1,1} & px_b^{1,2} & \dots & px_b^{32,32} \end{bmatrix}$$

5.1. ¿Cómo puede una función asociar una etiqueta con una forma?

Pasamos una forma codificada (gato, perro, etc.) y le decimos al programa que esto es 'gato', esto es 'perro'. El programa representa imágenes como puntos en el espacio dimensional superior y les asigna una etiqueta. Esto se llama entrenar al modelo .

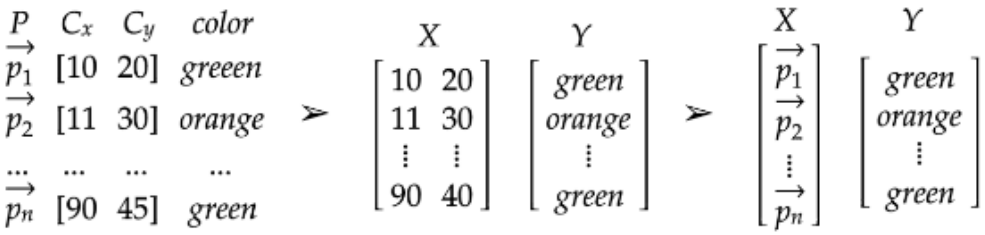
Una vez que se completa el entrenamiento, pasamos los datos de prueba al modelo para predecir el animal, al igual que le dimos un nuevo punto en el espacio 2D para determinar su color.

5.2. Preparando los datos

La depuración de un modelo con entrenamiento real y datos de prueba suele ser tedioso. Por supuesto, crear mis propios datos no hará que este programa esté libre de errores, pero este enfoque garantiza que los errores rudimentarios se puedan encontrar de manera rápida y oportuna.

La idea es simple: crear centros de clúster en función del número de clases que desea crear. Genere aleatoriamente más puntos alrededor del centro del clúster, escalados por una desviación estándar predefinida.

El valor de retorno de esta función es (X, y). X contiene vectores apilados verticalmente e y es el color del punto. Cada artículo en la pila es un punto en el plano XY.



$$C_x = x\ cordinate\ of\ a\ point$$
$$C_y = y\ coordinate\ of\ a\ point$$

5.3. Algoritmo

Usamos matrices numpy para realizar el cálculo de distancia. Cuando un programa no es entendible, debemos ejecutar cada línea en Python REPL para ver cuál es la entrada y cuál es la salida. Esto ayuda a entender por qué se hizo este método en particular.

```
import networkx as nx
import matplotlib.pyplot as plt
import cv2
import math



k=3072

class Node():

    def __init__(self, point):
        self.point = point
        self.axis = None
        self.left = None
        self.right = None
        self.tipo = None

class Cola_prioridad():

  def __init__(self, cant):
    self.cantidad = cant
    self.vector=[]

  def insertar(self, distancia,x):
    for i in range(len(self.vector)):
      if(self.vector[i][1]==x):
        return

    if(len(self.vector)==self.cantidad):
      if(self.vector[self.cantidad-1][0]>distancia):
        self.vector[self.cantidad-1]=[distancia,x]
        self.vector.sort(key=lambda tup: tup[0])
    else:
      self.vector.append([distancia,x])
      self.vector.sort(key=lambda tup: tup[0])

  def top(self):
    return self.vector[len(self.vector)-1][0]

  def capacidad(self):
    if(len(self.vector)==self.cantidad):
```

```
43        return True
44      else :
45        return False
46
47
48  def build_kdtree(points, depth=0):
49    if not points:
50      return None
51
52    axis = depth %k
53
54    points.sort(key=lambda tup: tup.point[axis])
55
56    median = len(points)//2
57
58    node = points[median]
59    node.axis=axis
60
61    node.left=build_kdtree(points[:median],depth+1)
62    node.right=build_kdtree(points[median+1:],depth+1)
63
64    return node
65
66  def distanceSquared(a, b):
67    distance = 0
68    for i in range(k):
69      distance = distance + pow((a[i]-b[i]),2)
70    return math.sqrt(distance)
71
72  def closest_point(node, point, depth , cola):
73    if(node == None):
74      return
75    axis = depth %k
76    nb = None
77    ob = None
78    cola.insertar(distanceSquared(point, node.point),node)
79    if(point[axis] < node.point[axis]):
80      nb = node.left
81      ob = node.right
82    else:
83      nb = node.right
84      ob = node.left
85
86    closest_point(nb, point, depth+1, cola)
87
88    if( (cola.capacidad()==False) or (cola.top() > abs(point[axis
       ]-node.point[axis]))):
89      closest_point(ob, point, depth+1, cola)
90
```

```
91 def nearest(root, point, count):
92   ColaP = Cola_prioridad(count)
93   closest_point(root, point, 0, ColaP)
94   return ColaP.vector
95
96
97 def image_to_feature_vector(image, size=(32, 32)):
98     return cv2.resize(image, size).flatten()
99
100
101 def main():
102
103     puntos=[]
104     normal=0
105     trampa=0
106
107     for i in range(1,11):
108         direccion1="/home/maik/Escritorio/eda/kdtree/img/normal"
    +str(i)+".png"
109         direccion1="/home/maik/Escritorio/eda/kdtree/img/trampa"
    +str(i)+".png"
110         image1 = cv2.imread(direccion1)
111         image1 = cv2.imread(direccion1)
112         pixels1 = image_to_feature_vector(image1)
113         pixels1 = image_to_feature_vector(image1)
114         nodo1 = Node(pixels1)
115         nodo1 = Node(pixels1)
116         nodo1.tipo="normal"
117         nodo1.tipo="trampa"
118         puntos.append(nodo1)
119         puntos.append(nodo1)
120
121
122
123     root=build_kdtree(puntos)
124     image = cv2.imread("/home/maik/Escritorio/eda/kdtree/img/
    test.png")
125     pixels = image_to_feature_vector(image)
126     resul = nearest(root, pixels,7)
127
128     for i in resul:
129       if(i[1].tipo=="normal"):
130         normal=normal+1
131       else:
132         trampa=trampa+1
133
134     if(trampa>normal):
135       print("Es carta Trampa")
136     else:
```

```
137         print("Es carta Monstruo")
138
139
140 main()
```

imagenes.py

5.4. Conjunto de entrenamiento, conjunto de validación, conjunto de prueba

Usamos una peueña base de datos de imagenes, que es un pequeño conjunto de datos de 10 clases. Este conjunto de datos ya está segregado en conjunto de entrenamiento y conjunto de prueba. Sin embargo, deberíamos crear un conjunto de validación a partir del conjunto de entrenamiento usando la relación de corte 1: 9 (si el conjunto de entrenamiento tenía 1k filas después del corte, 100 filas van en el conjunto de validación 900 filas van en el conjunto de entrenamiento).

Y necesitamos este conjunto de validación para ajustar nuestro hiperparámetro (parámetro que no se deriva del proceso de aprendizaje, pero es parte del proceso de aprendizaje) - K.

Tenga en cuenta que he usado el conjunto de entrenamiento y el conjunto de prueba solo con fines ilustrativos en la base de código.

6. Aplicaciones

- Este modelo se utiliza en data mining (minería de datos), por lo que su uso en aprendizaje automático (Machine Learning) es una de sus aplicaciones clave para, por ejemplo, sistemas de recomendación (plataformas de contenido digital, procesos de venta cruzada en eCommerce, etc.).
- El concepto de los vecinos más cercanos es un importante factor en la física de estado sólido, ya que las propiedades de la materia se modifican en función de la estructura de las redes y las distancias entre los átomos. De hecho, es habitual hacer aproximaciones en los modelos físicos para despreciar aquellas partículas que no forman parte de los K vecinos más próximos.

7. Conclusión

La precisión del algoritmo no es muy eficiente. Lo interesante es que incluso la implementación de vanguardia del algoritmo produce resultados similares; claramente, todavía tenemos un largo camino por recorrer, una vez que nuestras bases están cubiertas.

8. Bibliografía

- https://www.analiticaweb.es/algoritmo-knn-modelado-datos/
- https://es.wikipedia.org/wiki/K$_v ecinos_m$
- https://medium.com/@YearsOfNoLight/intro-to-image-classification-with-knn-987bc112f0c2
- https://www.pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification/