|  |  |  |
| --- | --- | --- |
| **Minería de datos**  **“Método OCR”** | | |
| **T3** | **#26. Gustavo Pimentel Zamora**  **#30. Nancy Pahola Rodríguez Parra** | **8A** |

**Introducción**

OCR (Optical Character Recognition) es una tecnología que permite transformar el contenido de una imagen en texto plano. Normalmente, el contenido de una imagen que suele transformarse es aquel asociado a cadenas de texto.

El uso de aplicaciones OCR permite extraer el contenido textual de documentos contenidos en formato de imagen, a fin de realizar diversas funciones de gestión documental, tales como: Búsqueda de documentos con cierta información, extracción de la información concreta de un documento, pasar de un documento escrito contenido en un archivo de imagen, a un archivo con formato de texto plano.

**Requerimientos**

* Equipo de computo
* Anaconda versión 3.6
* Imágenes segmentadas a partir del código en matlab

**Pasos del algoritmo OCR**

1. **Generar Dataset** 
   1. Partir de un conjunto de imágenes segmentadas (dataset del 0 al 9) obtenidas a partir del código en matlab con formato .png. Cada carpeta contiene 237 imágenes, dando un total de 2,369 imágenes.



* 1. En Spyder (python 3.5) recorrer las carpetas y extraer cada imagen del dataset, posteriormente analizar cada imagen y extraer sus 14 características:

**Característica 1:** Se obtiene la relación del número de pixeles blancos(1’s) de la imagen/ (ancho\*alto) de la imagen.



Pixeles Negros (0´s)

Alto

Pixeles blancos (1´s)

Ancho

 **Característica 2:** Se obtiene la relación del ancho/alto de la imagen.

Alto

Ancho

**Característica 3:** Relación de pixeles blancos (1´s) de la columna intermedia de la imagen/ (alto\*ancho) de la imagen.



**Característica 4:** Relación de pixeles blancos (1´s) de ¼ de columna de la imagen/ (alto\*ancho) de la imagen.

**Característica 5:** Relación de pixeles blancos (1´s) de ¾ de columna de la imagen/ (alto\*ancho) de la imagen.

**Característica 6:** Relación de pixeles blancos (1´s) de la fila intermedia de la imagen/ (alto\*ancho) de la imagen.

**Característica 7:** Relación de pixeles blancos (1´s) de ¼ de fila de la imagen/ (alto\*ancho) de la imagen.

**Característica 8:** Relación de pixeles blancos (1´s) de ¾ de fila de la imagen/ (alto\*ancho) de la imagen.

**Característica 9:** Relación del número de cortes entre pixeles blancos (1´s) y pixeles negros (0´s) de la columna intermedia de la imagen/ (alto\*ancho) de la imagen.

**Característica 10:** Relación del número de cortes entre pixeles blancos (1´s) y pixeles blancos (0´s) a ¼ de la columna de la imagen/ (alto\*ancho) de la imagen.

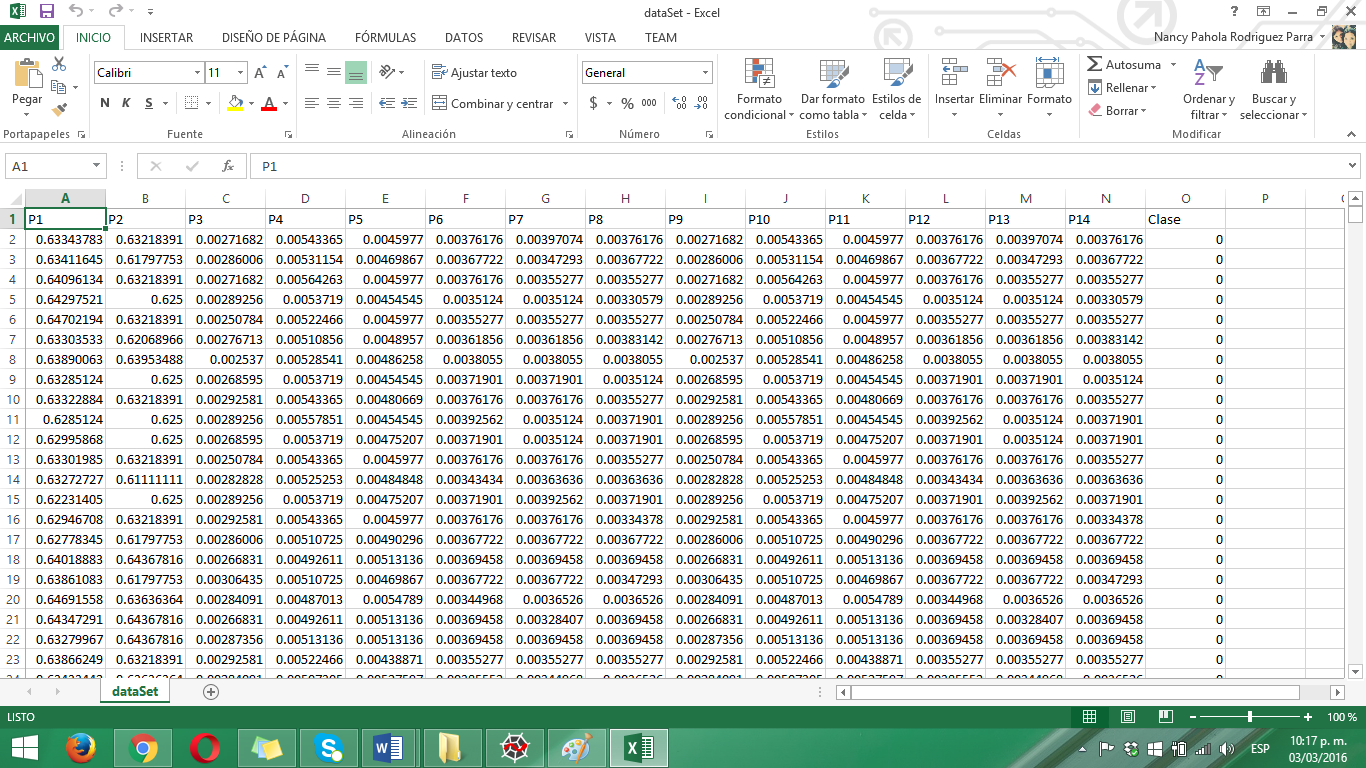
**Característica 11:** Relación del número de cortes entre pixeles blancos (1´s) y pixeles blancos (0´s) a ¾ de la columna de la imagen/ (alto\*ancho) de la imagen.

**Característica 12:** Relación del número de cortes entre pixeles blancos (1´s) y pixeles blancos (0´s) de la fila intermedia de la imagen/ (alto\*ancho) de la imagen.

**Característica 13:** Relación del número de cortes entre pixeles blancos (1´s) y pixeles blancos (0´s) de ¼ de fila de la imagen/ (alto\*ancho) de la imagen.

**Característica 14:** Relación del número de cortes entre pixeles blancos (1´s) y pixeles blancos (0´s) de ¾ de fila de la imagen/ (alto\*ancho) de la imagen.

Nota: Las características antes mencionadas se deben obtener de cada una de las imágenes del dataset, esto quiere decir que se obtienen 14 características de 2,369 imágenes que contiene el dataset, teniendo como resultado un total de 33,166 características.

* 1. Las características obtenidas se irán almacenando en una matriz en python, agregando la clase a la que pertenecen de tal manera que, si se lee una imagen con número 1, aparte de almacenar sus 14 características también almacenara que la imagen leída pertenece a la clase 1.
  2. Se crea un archivo con extensión .csv y se le pasan los datos almacenados en la matriz, con este paso queda generado el dataset, con las características que se obtuvieron en los pasos anteriores de las 2, 369 imágenes.

1. Función del programa:
   1. Al ejecutar el programa se muestra un mensaje que pide el nombre de la nueva instancia, esta instancia debe de ser una imagen segmentada a partir del código de matlab con formato .png.
   2. Una vez que se introdujo el nombre de la imagen, el programa pedirá que se introduzca el valor de K, es decir el número de vecinos.
   3. Para definir los vecinos más cercanos se hace lo siguiente:

* Una vez que se lee la nueva instancia se crearán dos matrices, una será la matriz que almacenará los datos de la imagen de entrada y la segunda será la matriz que almacenará los vecinos más cercanos.
* Se obtienen las filas (alto) y las columnas (ancho) de la nueva instancia.
* Posteriormente se lee el archivo dataSet.csv con el cual se llenará la primera matriz con las 14 propiedades de cada imagen y la clase a la que pertenece.
* Se obtienen las 14 propiedades de la nueva instancia, los datos resultantes de la obtención de estas propiedades se almacenarán en una segunda matriz.
* Una vez llenadas las matrices se calculará la distancia para encontrar los vecinos más cercanos. Para esto se utilizará la fórmula Euclidiana.
* Dependiendo el número de vecinos a calcular será el número de distancias mínimas que se deben considerar para definir la clase a la que pertenece la imagen ingresada anteriormente.
* Se define la clase a la que pertenece la nueva instancia considerando la clase que más se repite en los vecinos cercanos encontrados (valor de k).
  1. Se muestra en pantalla la salida del programa con los siguientes datos:

