**Introducción**

El OCR (Optical Character Recognition) es una tecnología que trata de emular la capacidad del ojo humano para reconocer objetos. Concretamente es un software que permite el reconocimiento óptico de los caracteres contenidos en una imagen (documento escaneado o fotografía), de forma que estos se vuelven comprensibles o reconocibles para un ordenador, obteniendo como resultado final un archivo en un formato de texto editable. El formato del archivo de salida (txt, pdf, etc.) dependerá de las posibilidades que ofrezca el software.

Para reconocer los caracteres, el software inspecciona la imagen pixel a pixel, buscando formas que coincidan con los rasgos de los caracteres. En función del nivel de complejidad o grado de desarrollo del software, éste buscará coincidencias con los caracteres y fuentes disponibles en el programa, o tratará de identificar los caracteres a través del análisis de sus características, de forma que el reconocimiento de los mismos no se limite exclusivamente a un determinado número de fuentes.

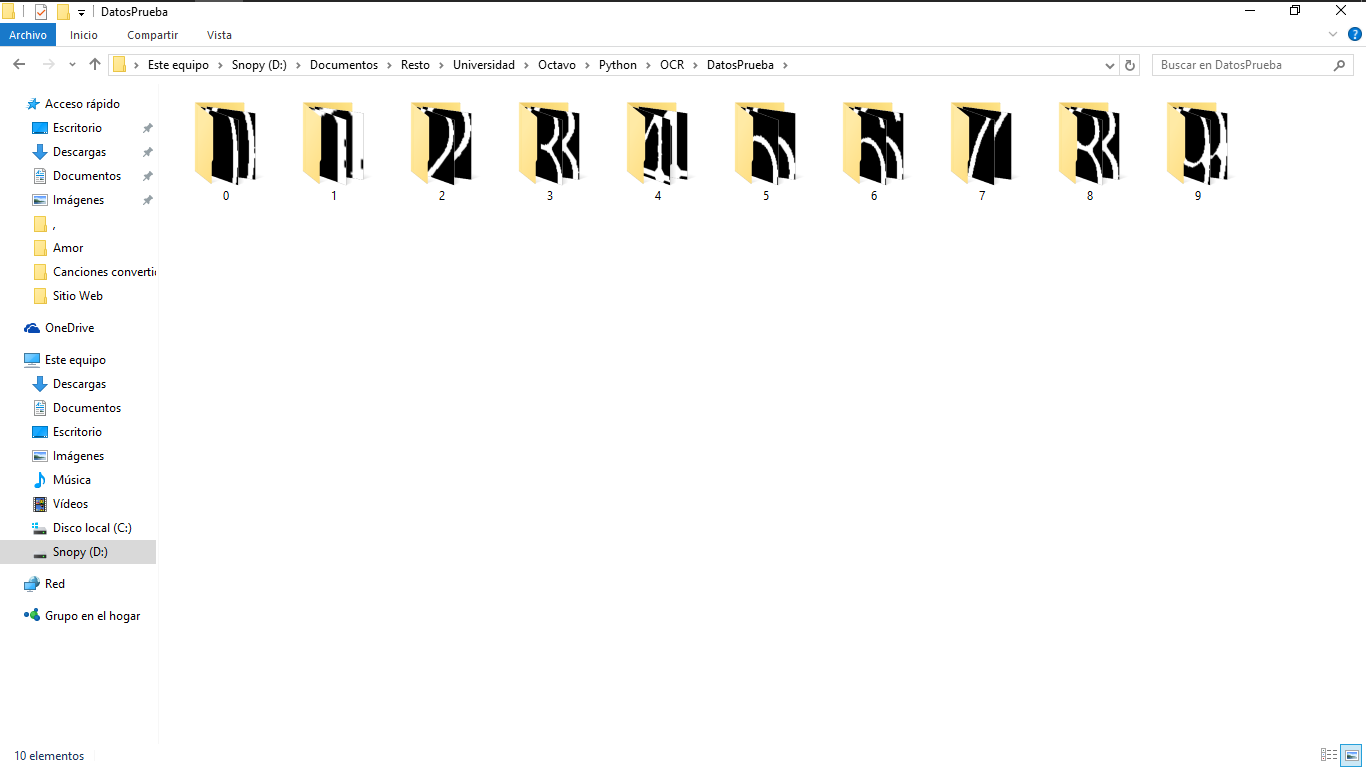
**Requerimientos**

* Equipo de cómputo.
* Anaconda versión 3.6.
* Imágenes segmentadas a partir del código en MatLab.

**Descripción del algoritmo OCR**

***Generación del Dataset***

* Se parte de un conjunto de 1,726 imágenes segmentadas (dataset del 0 al 9) obtenidas a partir del código en matlab con formato png.



* En Spyder (python 3.5), se analizará cada imagen del dataset para poder extraer sus 14 características respectivamente. Esas características son las siguientes:

1. Se deberá obtener la proporción del número de 1’s que se encuentran en la imagen en relación con el ancho\*alto de la misma imagen.



Total de pixels blancos / ancho\*alto

1. Se deberá obtener la proporción de la relación del ancho/alto de la imagen.
2. Se obtendrá la proporción de la relación de 1´s que se encuentran a la mitad de la imagen de forma vertical con respecto al alto de la imagen.



1. Se obtendrá la proporción de la relación de 1´s que se encuentran a 1/4 de la imagen de forma vertical con respecto al alto de la imagen.



1. Se obtendrá la proporción de la relación de 1´s que se encuentran a 3/4 de la imagen de forma vertical con respecto al alto de la imagen.



1. Se obtendrá la proporción de la relación de 1´s que se encuentran a 1/2 de la imagen, pero ahora de forma horizontal con respecto al ancho de la imagen.



1. Se obtendrá la proporción de la relación de 1´s que se encuentran a 1/4 de la imagen, pero ahora de forma horizontal con respecto al ancho de la imagen.



1. Se obtendrá la proporción de la relación de 1´s que se encuentran a 3/4 de la imagen, pero ahora de forma horizontal con respecto al ancho de la imagen.



1. Se obtendrá la proporción de la relación de cortes que se encuentran a 1/2 de la imagen de forma vertical, es decir, los cambios que se hacen entre 1´s y 0´s de la misma imagen.



1. Se obtendrá la proporción de la relación de cortes que se encuentran a 1/4 de la imagen de forma vertical, es decir, los cambios que se hacen entre 1´s y 0´s de la misma imagen.



1. Se obtendrá la proporción de la relación de cortes que se encuentran a 3/4 de la imagen de forma vertical, es decir, los cambios que se hacen entre 1´s y 0´s de la misma imagen.



1. Se obtendrá la proporción de la relación de cortes que se encuentran a 1/2 de la imagen, pero ahora de forma horizontal, es decir, los cambios que se hacen entre 1´s y 0´s de la imagen.



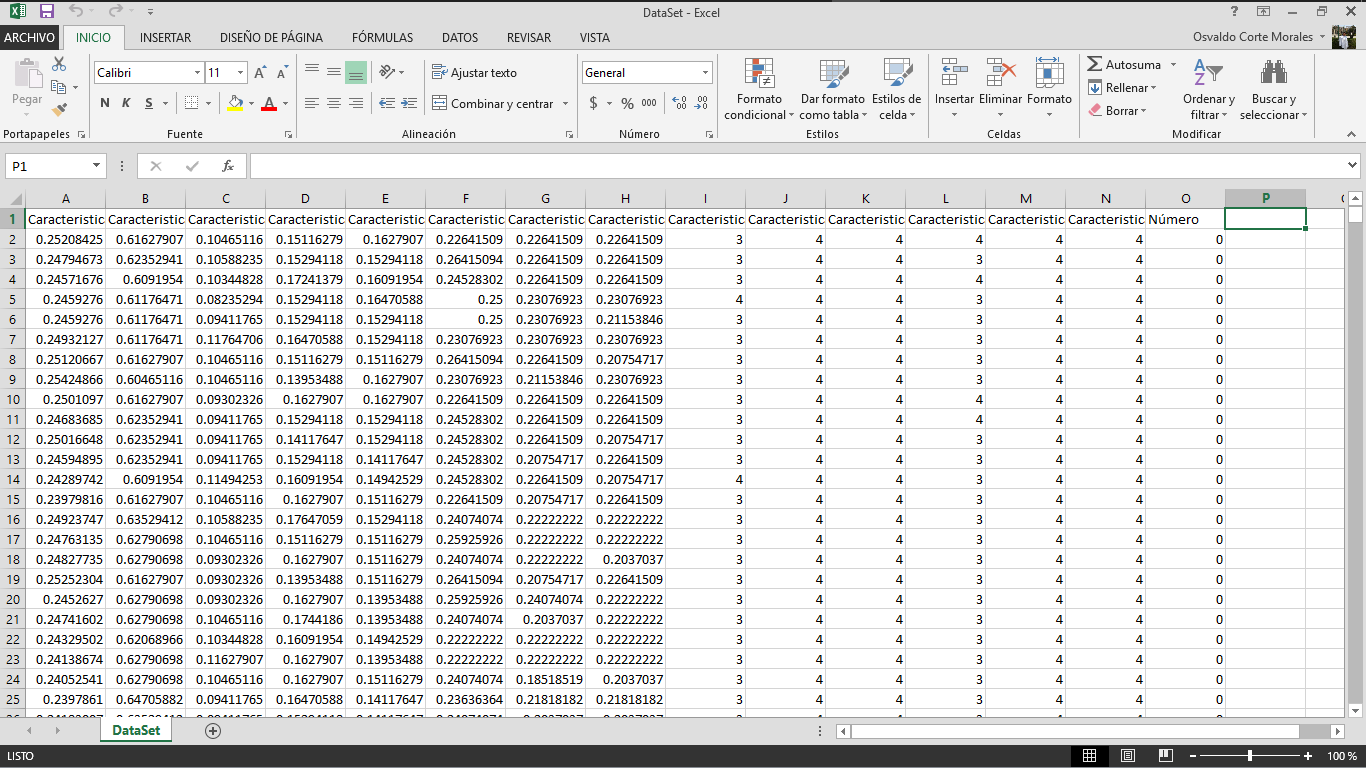
1. Se obtendrá la proporción de la relación de cortes que se encuentran a 1/4 de la imagen, pero ahora de forma horizontal, es decir, los cambios que se hacen entre 1´s y 0´s de la imagen.



1. Se obtendrá la proporción de la relación de cortes que se encuentran a 3/4 de la imagen, pero ahora de forma horizontal, es decir, los cambios que se hacen entre 1´s y 0´s de la imagen.



Es importante mencionar que las características anteriormente mencionadas, deben obtenerse de cada una de las imágenes del dataset, es decir, que se obtienen 14 características de 1,726 imágenes contenidas en el dataset, teniendo como resultado un total de 24,164 características.



Las características obtenidas se almacenarán en una matriz en python, la cual deberá tener la posición, la distancia y la clase a la que pertenecen. Esto se refiere a que si se lee una imagen con número 5, aparte de almacenar sus 14 características también almacenará que la imagen leída pertenece a la clase 5.

* Finalmente 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 1,726 imágenes.

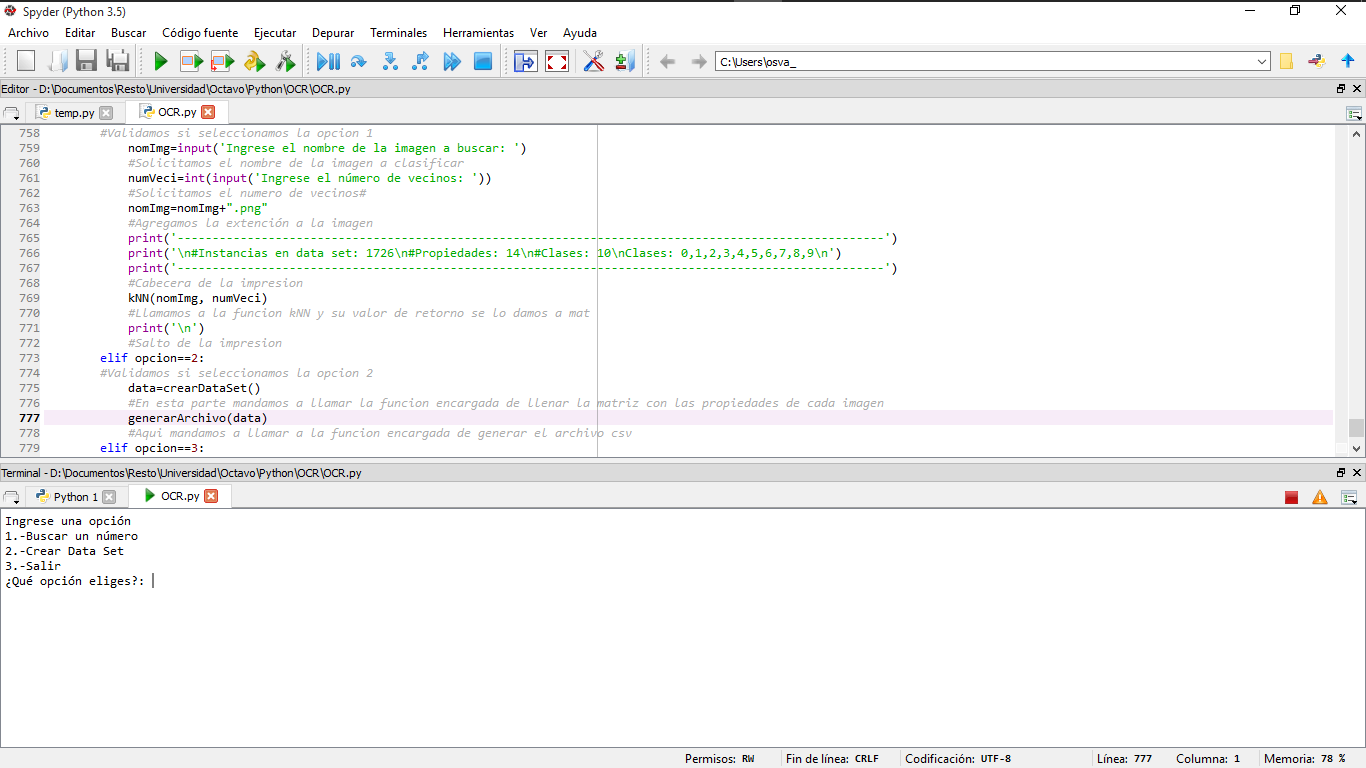
***KNN vecinos más cercanos***

Una vez obtenido el dataSet.csv, se utilizará el método KNN que nos permitirá obtener los vecinos más cercanos de cualquier imagen segmentada. La función de esté método es la siguiente:

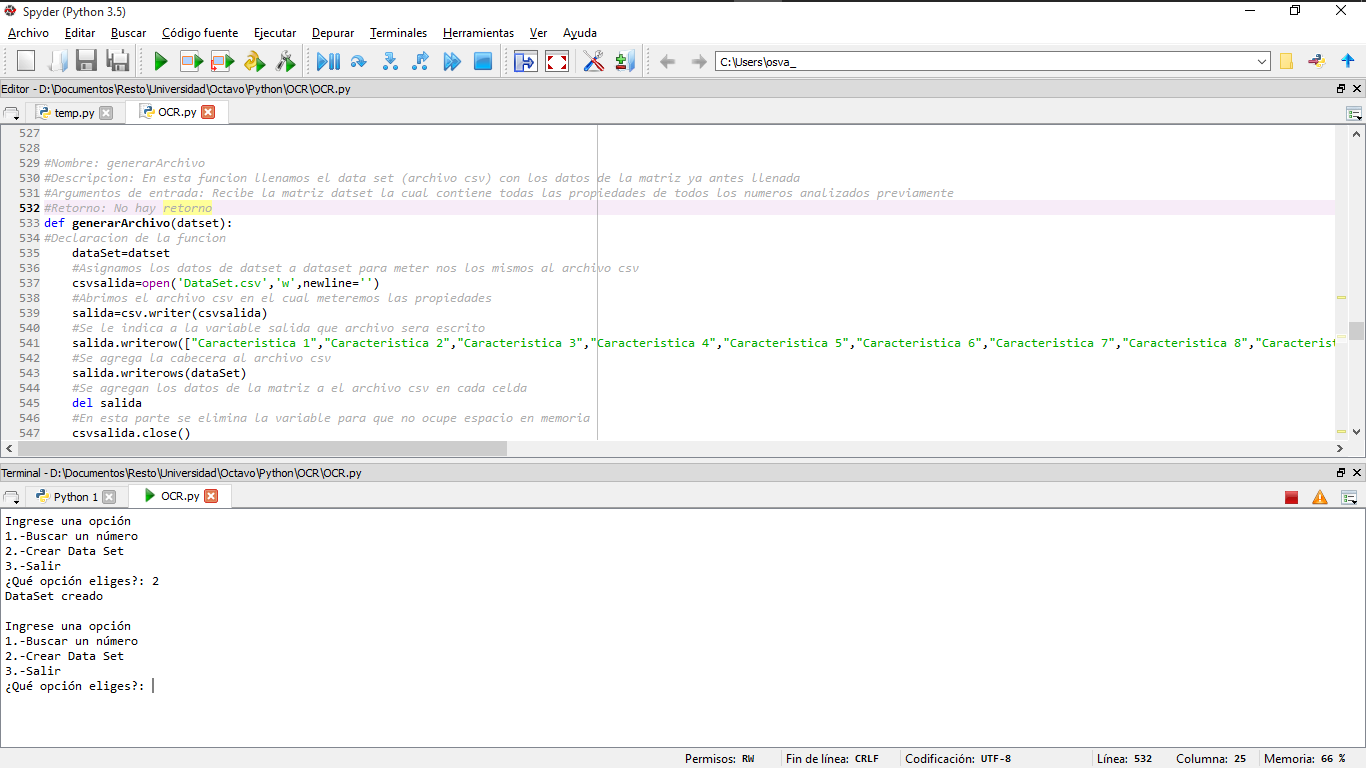
* Se seleccionará la imagen.
* Se obtienen las 14 caracteristicas de la nueva imagen con el proceso utilizado anteriormente para generar los datos del dataset, solo que esta vez se aplicara solo a la nueva imagen.
* Se declararán dos variables que serán nuestras matrices, una de ellas será la que almacenará los datos del datasetde entrada y la segunda la que almacenará los vecinos más cercanos.
* Una vez declaradas la matrices, se obtendrán las filas y las columnas de la imagen.
* La matriz que contendrá los datos de la imagen se llenará con comillas simples ‘ ‘, hasta llenar las 15 columnas.
* De igual manera se llenará la matriz knn con comillas simples ‘ ‘ para que almacene la posición, la distancia y la clase de la imagen.
* En seguida se leerá el archivo dataSet.csv. Esos datos ahora se almacenarán la primera matriz con las 14 propiedades de cada imagen.
* La segunda matriz se llenará con la nueva instancia obtenida del dataSet.csv.
* Una vez llenadas las matrices se calculará la distancia para encontrar los vecinos más cercanos. Para esto se utilizará la fórmula de la Euclídiana que es (el dato de la primer característica del dataset - el dato de la nueva instancia) ^ 2 + la segunda, la tercera, hasta sumar la 14 características. Estos datos serán almacenados en la matriz knn.
* Finalmente se retorna el elemento encontrado para que sea mostrada la posición, la distancia y la clase a la que pertenece dicha imagen.

**Corrida**

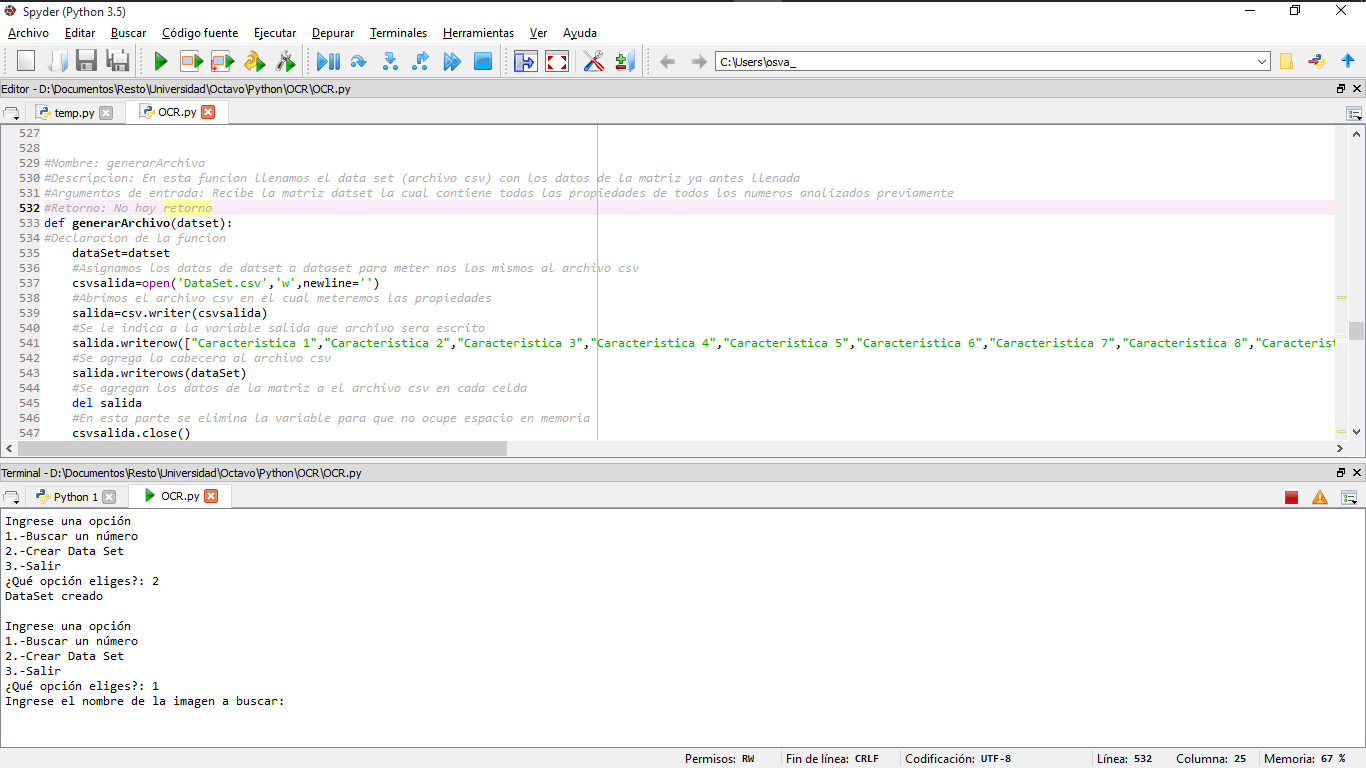
Al ejecutar el programa esta será la ventana que veremós, que es un menu de opciones entre las que eligiremos la deseada.



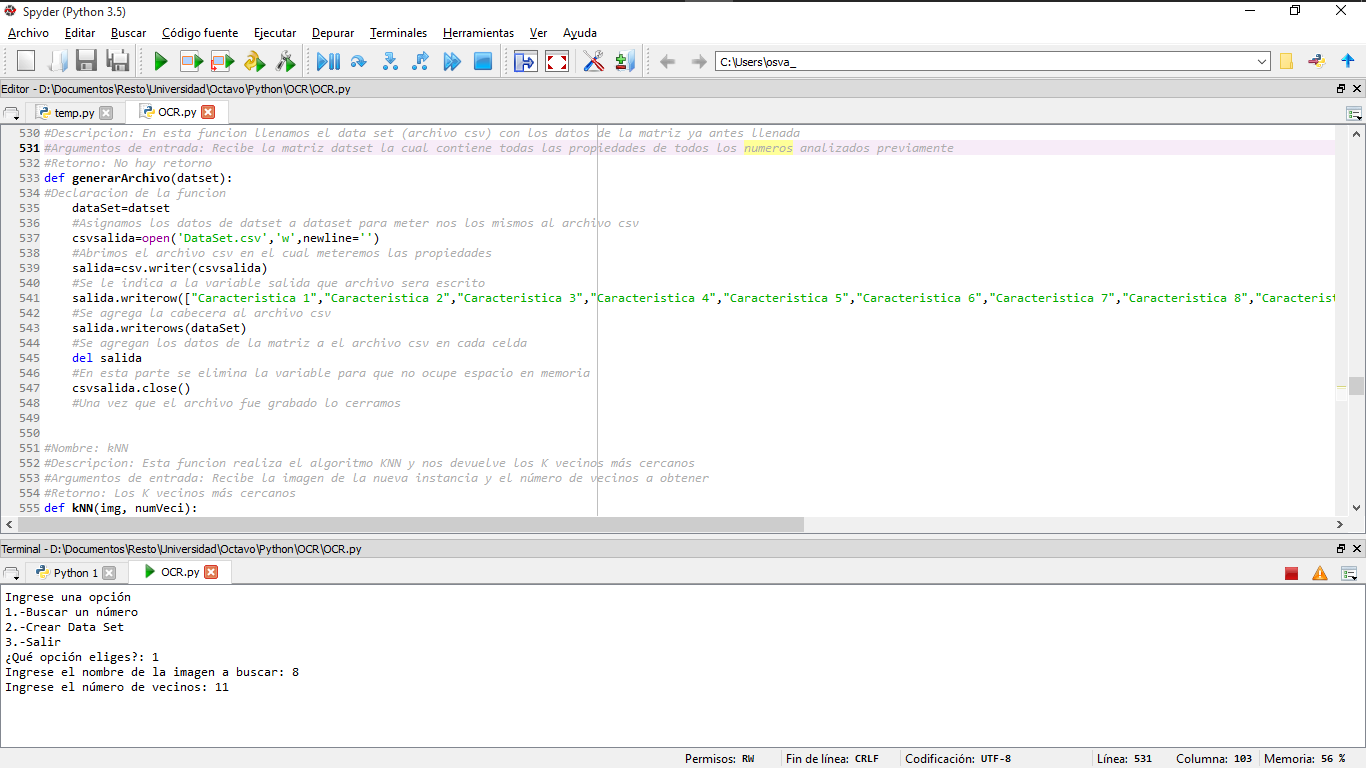
Lo primero que haremós será elegir la opcion 2, pues primero deberemos crear el Dataset para poder realizar el reconocimiento.



Una vez que termine de crear el Dataset nos mostrara el mensaje de “DataSet creado”, y podremos continuar y a continuación seleccionaremos la opción 1 para realizar el reconocimiento.



Ahora ingresaremos el nombre de la imagen que deseamos reconocer, en este caso solo pondremos el nombre, ejemplo “8”, puesto que la extención la agrega automaticamente.



Al final nos mostrará la impresión con el número de instancias del dataset, el número de propuedades que se consideraron, el número de clases y los nombres de cada clase, además mostrara la lista de los vecinos que se eligieron junto con su posición en el DataSet, su distancia Euclidiana y la clase a la que pertenece, después de ello nos mstrara el número de instancias en cada clase, el nombre de la clase y los vecinos que hay de esa clase, por último nos mostrara la clase a la que pertenece la imagen seleccionada.

