**INSTITUTO POLITÉCNICO NACIONAL**

**Unidad Profesional Interdiciplinaria**

**de Ingeniería Campus Tlaxcala**

**“Programa de algoritmo**

**de agrupación”**

**K Nearest Neighbors**

**FUNDAMENTOS DE PROGRAMACIÓN**

**DR. Esaú Eliezer Escobar Juárez**

**PRESENTA:**

**Miguel Alejandro Flores Sotelo**

**Juan Carlos Flroes Mora**

**Oscar Ayala Elizalde**

1. **INTRODUCCIÓN**

El programa KNN es un muy efectivo, esta herramienta es muy utilizada en el campo del aprendizaje automático que permite clasificar y predecir datos en función de la información de los puntos más cercanos. Básicamente el objetivo principal es encontrar los vecinos mas cercanos a un punto desconocido (centroides iniciales) y utilizar su información para determinar su clasificación y el valor.

Actualmente vivimos en un entorno en el que existe una cantidad inmensa de datos, de los cuales se pueden utilizar para aprovecharlos de una manera efectiva, y utilizarlos para potenciarlos como una ventaja competitiva. Por lo que el KNN, es una forma de ejemplificarlo.

1. **PLANTEMIENTO DEL PROBLEMA**

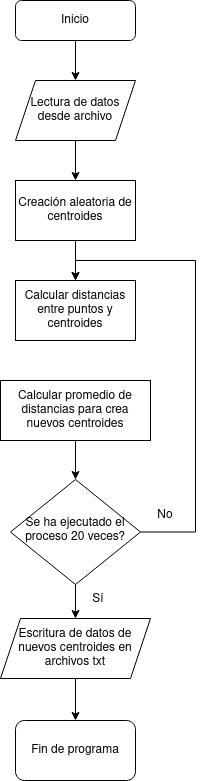
A continuación, se dará una basta cantidad de datos por los cueles mediante un programa utilizando el método KNN, clasificará los datos para determinarlos con su respectiva planta en este caso: Iris-setosa, Iris-versicolor y Iris-virginica.

1. **OBJETIVO**

El objetivo es lograr una clasificación precisa y confiable de las nuevas muestras de plantas utilizando el algoritmo KNN, tratando de tener un buen desempeño en términos de precisión y eficacia.

1. **Estructura del programa**
   1. **Planteamiento:**

Antes de comenzar a escribir el código que resuelva la clasificación de los datos, se han planteado los pasos a seguir para tener el objetivo más claro, estos pasos se representan en nuestro diagrama de flujo.

****

Cada proceso mostrado en nuestro diagrama es necesario para una correcta ejecución del programa y así obtener los resultados esperados, a continuación se procede a explicar cada paso a seguir.

* 1. **Lectura de datos:**

Ya que debemos trabajar con una gran cantidad de datos, se ha elegido que el programa lea la información desde el archivo que se nos fue asignado. Para cumplir con la lectura de datos, hemos escrito un bloque encargado de hacer esta función.



Ya que la información en el archivo a leer no está libre de caracteres no deseados, no se puede leer y copiar todo el contenido directamente al programa para trabajar, se escribireron algunas líneas especiales para retirar caracteres no deseados y así procesar libremente los valores numéricos.

* 1. **Creación de centroides aleatorios:**

Es necesario determinar las coordenadas para los centroides que distribuirán los datos en sus categorías correspondientes, para ello se escribió una función que haciendo uso de la instrucción “rand()”, se generan los valores pseudoaleatorios de las posiciones de los centroides.

void numeros\_rand(float c1[4],float c2[4],float c3[4]){

*for*(int k=0;k<4;k++){

c1[k]=((float)rand()/RAND\_MAX)\*9;

c2[k]=((float)rand()/RAND\_MAX)\*9;

c3[k]=((float)rand()/RAND\_MAX)\*9;

}

* 1. **Calcular distancias entre puntos y centroides:**

Una vez se han generado las posiciones de los centroides y tenemos la información en nuestro programa, podemos comenzar a hacer los cálculos sobre distancias, igualmente hemos escrito una función encargada de hacer estas operaciones, en donde se hace uso de la fórmula para cálculo de distancias en un plano cartesiano con cuatro dimensiones. Esta función se mandará a llamar para medir la distancia entre cada punto con los centroides para posteriormente declarar cuál es el más cercano y asignar los puntos a sus correspondientes grupos.

float distance(float datos[][4], float *c*[], int *i*){

float distancia;

distancia=sqrt((pow((datos[i][0]-c[0]),2))+(pow((datos[i][1]-c[1]),2))+(pow((datos[i][2]-c[2]),2))+(pow((datos[i][3]-c[3]),2)));

*return* distancia;

}

* 1. **Promediar valores y generar nuevos centroides:**

Después de agrupar nuestros puntos, se debe calcular el promedio de los grupos de puntos para generar nuevas cordenadas para nuestros centroides, usamos un contador llamado ‘con\_cn’, con el cual sabemos cuantos puntos corresponden a cada grupo y dividir el promedio entre el valor correcto.

*//CALCULA EL PROMEDIO PARA LOS NUEVOS CENTROIDES*

*for*(int k=0;k<4;k++){

*if*(cont\_c1!=0){

nuevoc1[k]/=cont\_c1;

}*else*{

nuevoc1[k]=c1[k];

}

*if*(cont\_c2!=0){

nuevoc2[k]/=cont\_c2;

}*else*{

nuevoc2[k]=c2[k];

}

*if*(cont\_c3!=0){

nuevoc3[k]/=cont\_c3;

}*else*{

nuevoc3[k]=c3[k];

}

}

*//ASIGNA LOS CENTROIDES NUEVOS AL ORIGINAL*

*for*(int k=0;k<4;k++){

c1[k]=nuevoc1[k];

c2[k]=nuevoc2[k];

c3[k]=nuevoc3[k];

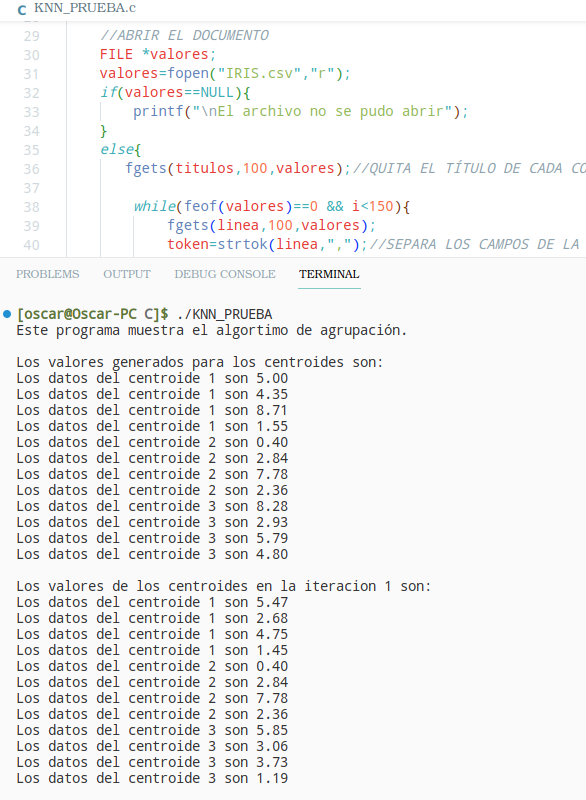
}

El proceso de generar los centroides debe generarse al menos 20 veces para una mejor presición en los resultados, por lo que se repite el cálculo de distancias, agrupación y cálculo de promedio para seguir reubicando los nuevos centroides.

* 1. **Mostrar y graficar reesultados:**

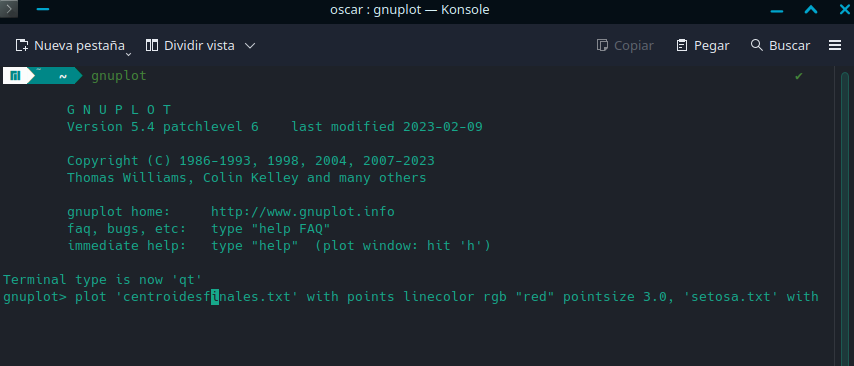
Después de haber repetido nuestro proceso las veces necesarias, podemos recopilar los resultados y escribirlos en archivos formato ‘txt’ para graficarlos dentro de una aplicación llamada GNUplot, la cual nos permite tener una mejor presepción del comportamiento de nuestras operaciones, los resultados se muestran a continuación:

En esta imagen podemos apreciar los resultados después de compilar y ejecutar el programa, se muestran los valores resultantes para cada centroide antes y después de las operaciones.

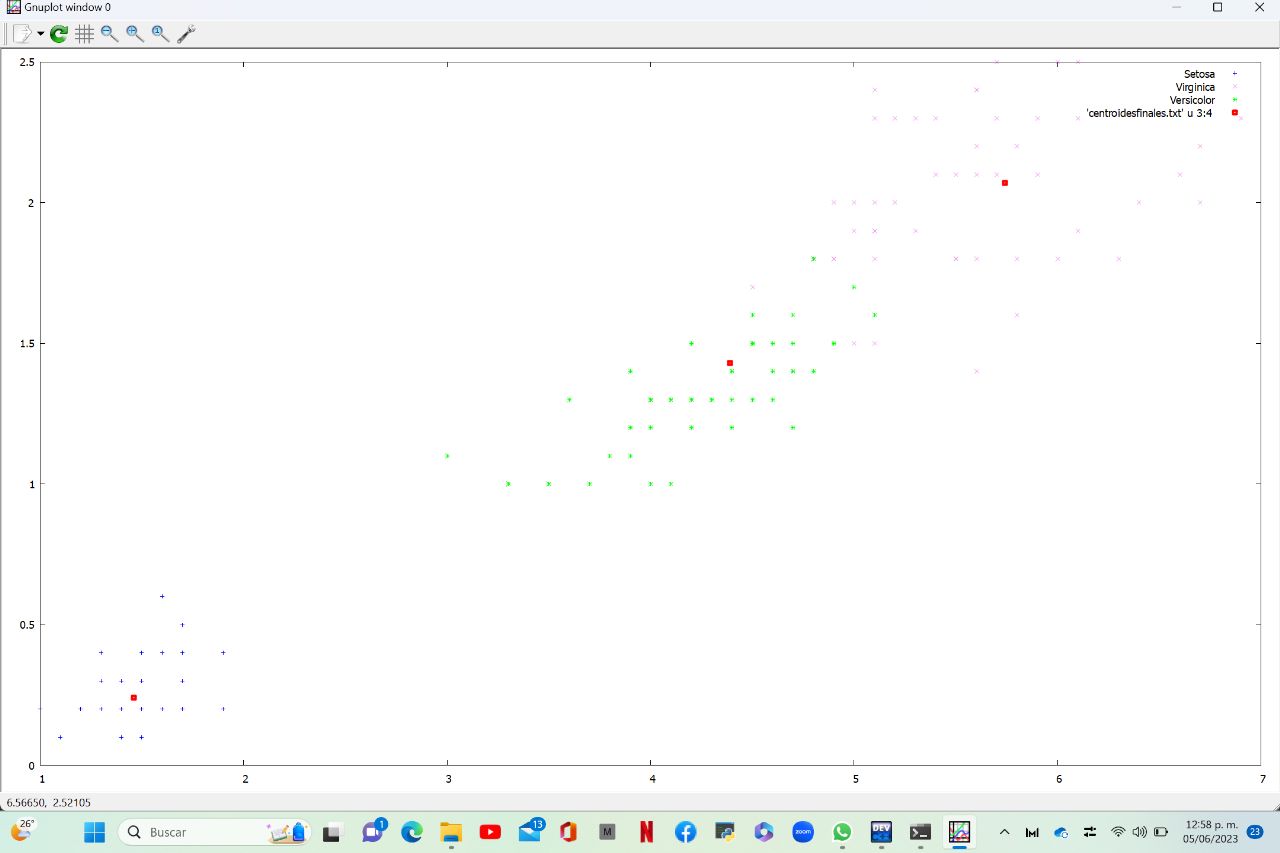


Como se ha mencionado antes, hicimos uso de GNUplot para graficar los resultados, esto se hace desde una terminal en la que por medio de códigos, podemos cargar los archivos que generamos en el programa.

Aquí se muestra la terminal que ejecuta los comandos para graficar:



Aquí se muestran los resultados graficados de los nuevos centroides con los puntos después de ejecutar todo el programa y hacer los cálculos de distancias 20 veces:



1. **Conclusión:**

En conclusión, el desarrollo de esta práctica nos permite tener un mejor entendimiento de las bases del aprendizaje computacional, pues no solo hemos escrito un programa que hace los cálculos que se le asignen, si no que, hicimos uso de nuevas herramientas para representar gráficamente los resultados generados. Además este fue un trabajo en equipo por lo que también afrontamos el reto mantenernos en el mismo flujo de trabajo para encontrar el método más eficiente, en donde hubo lluvia de ideas, opiniones sobre estilos de programación y confusión al armar el programa con bloques escritos por diferentes personas.Nuestro aprendizaje en este proceso de resolver problemas nos permitió mejorar nuestra lógica, orden, comunicación y perspicacia,