INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

25 de noviembre del 2021

Jerez, Zac

Ingeniería en sistemas computacionales

Semestre: 9

Alumna: Leticia carrera venegas

Correo: Letycv25@gmail.com

No. control:S17070155

Actividad: K-means

Inteligencia Artificial.

Docente: MCC Vanessa Saldivar Quezada.

Introducción

El algoritmo trabaja iterativamente para asignar a cada “punto” (las filas de nuestro conjunto de entrada forman una coordenada) uno de los “K” grupos basado en sus características. Son agrupados en base a la similitud de sus features (las columnas). Como resultado de ejecutar el algoritmo tendremos; los “centroids” de cada grupo que serán unas “coordenadas” de cada uno de los K conjuntos que se utilizarán para poder etiquetar nuevas muestras. Y las etiquetas para el conjunto de datos de entrenamiento. Cada etiqueta perteneciente a uno de los K grupos formados.

Los grupos se van definiendo de manera “orgánica”, es decir que se va ajustando su posición en cada iteración del proceso, hasta que converge el algoritmo. Una vez hallados los centroids deberemos analizarlos para ver cuáles son sus características únicas, frente a la de los otros grupos. Estos grupos son las etiquetas que genera el algoritmo.

El algoritmo utiliza un proceso iterativo en el que se van ajustando los grupos para producir el resultado final. Para ejecutar el algoritmo deberemos pasar como entrada el conjunto de datos y un valor de K. El conjunto de datos serán las características para cada punto. Las posiciones iniciales de los K centroids serán asignadas de manera aleatoria de cualquier punto del conjunto de datos de entrada. Luego se itera en dos pasos, paso de asignación de datos y paso de actualización de Centroid.

Objetivo

Elaborar un programa el cual incluya un algoritmo de aprendizaje supervisado, en específico k-means, en donde se le solicite al usuario seleccionar el numero de clusters en un rango de 2 a 6, y posteriormente poder mostrar el resultado en una gráfica.

Planteamiento del programa

Elaborar un programa el cual incluya un algoritmo de aprendizaje supervisado, en específico k-means.

El algoritmo K-means es un algoritmo iterativo que intenta dividir el conjunto de datos en subgrupos (clústeres) distintos no superpuestos definidos previamente por K donde cada punto de datos pertenece a un solo grupo. Intenta hacer que los puntos de datos dentro del clúster sean lo más similares posible y, al mismo tiempo, mantiene los clústeres lo más diferentes (lejos) posible.

Asigna puntos de datos a un grupo de modo que la suma de la distancia al cuadrado entre los puntos de datos y el centroide del grupo (media aritmética de todos los puntos de datos que pertenecen a ese grupo) es mínima. Mientras menos variación tengamos dentro de los conglomerados, más homogéneos (similares) serán los puntos de datos dentro del mismo conglomerado.

Resultado

Primero se procede a especificar el número de conglomerados K (clases), las cuales se solicitan en la interfaz al usuario.

Inicializar los centroides barajando primero el conjunto de datos y luego seleccionando aleatoriamente K puntos de datos para los centroides sin reemplazo, y se sigue iterando hasta que no haya cambios en los centroides. es decir, la asignación de puntos de datos a los clústeres no está cambiando.

Calcular la suma de la distancia al cuadrado entre los puntos de datos y todos los centroides y asignar cada punto de datos al grupo más cercano (centroide) y, por último, calcular los centroides de los grupos tomando el promedio de todos los puntos de datos que pertenecen a cada grupo.

La base de datos seleccionada para este sistema es la base de datos Iris.

Código

ow.py × main.py ×

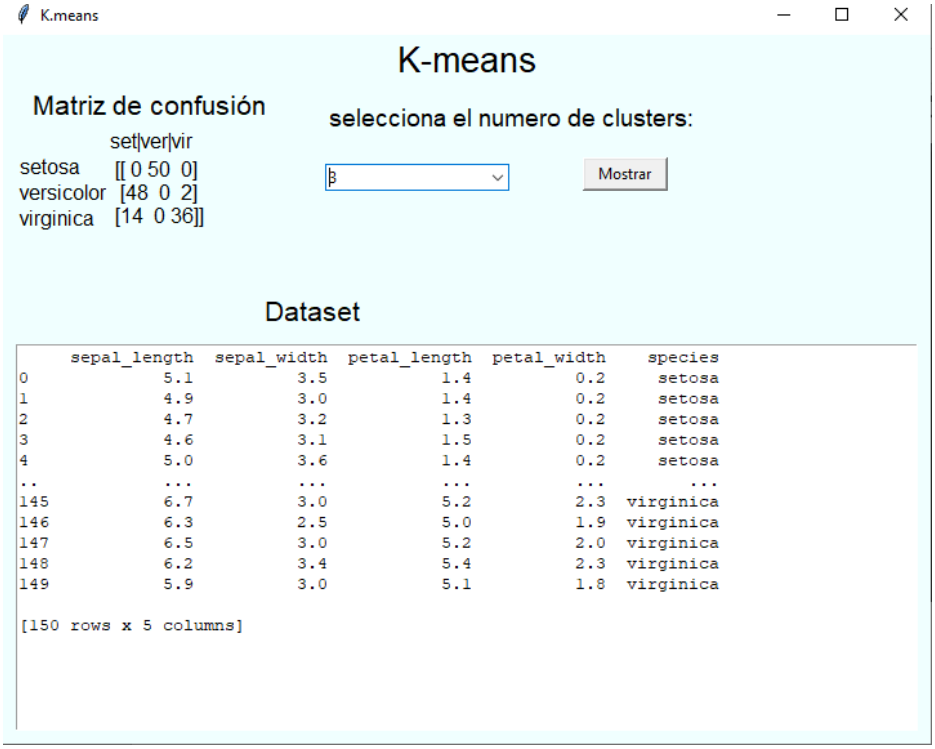
```python
from tkinter import ttk
import matplotlib.pyplot as plt
import pandas as pd
import tkinter as tk
from sklearn.metrics import confusion_matrix
from sklearn.cluster import KMeans

#Leer con pandas el dataset Iris
dataset = pd.read_csv('Iris.csv')
print(dataset)
x = dataset.iloc[:,:4].values
y = dataset['species'].values
#dataset por cada especie

ventana = tk.Tk()
ventana.title("K.means")
ventana.geometry("720x550")
ventana.configure(bg="azure")
```

```python
titulo = tk.Label(ventana, text="K-means", bg="azure", font=("Helvetica", 20))
titulo.pack(fill=tk.X)
ltit = tk.Label(ventana, text="Matriz de confusión", bg="azure", font=("Helvetica", 16))
ltit.place(x=20, y=40)
lbl = tk.Label(ventana, text="set|ver|vir", bg="azure", font=("Helvetica", 12))
lbl.place(x=80, y=70)
lbl2 = tk.Label(ventana, text="setosa", bg="azure", font=("Helvetica", 12))
lbl2.place(x=10, y=90)
lbl3 = tk.Label(ventana, text="versicolor", bg="azure", font=("Helvetica", 12))
lbl3.place(x=10, y=110)
lbl4 = tk.Label(ventana, text="virginica", bg="azure", font=("Helvetica", 12))
lbl4.place(x=10, y=130)



lbló=tk.Label(ventana, text="selecciona el numero de clusters:",bg="azure", font=("Helvetica", 14))
lbló.place(x=250,y=50)
combo_cluster=ttk.Combobox(ventana, values=["2","3","4","5","6"])
combo_cluster.set(3)
combo_cluster.place(x=250, y=100)
```

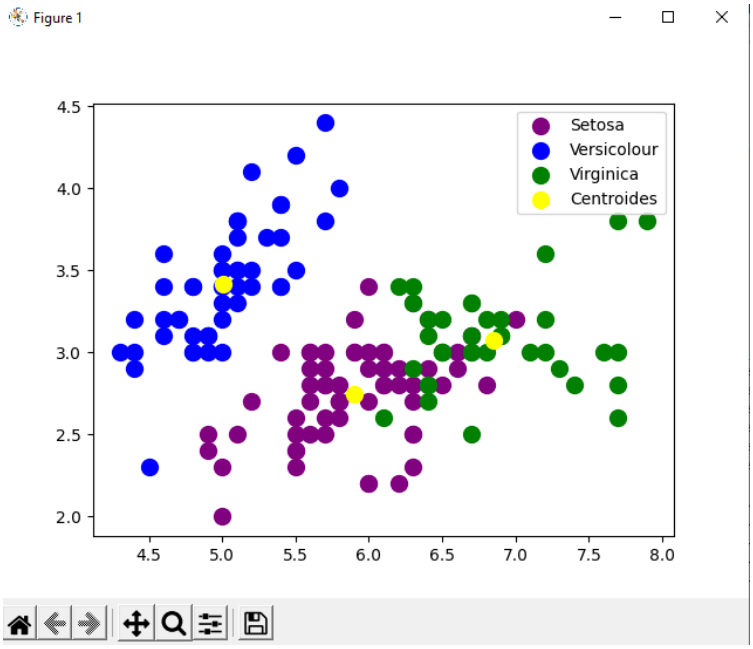
```
ltitd2 = tk.Label(ventana, text="Dataset", bg="azure", font=("Helvetica", 16))
ltitd2.place(x=200, y=200)
caja2 = tk.Text(ventana)
caja2.insert(tk.INSERT,dataset)
caja2.place(x=10, y=240, height=300, width=700)
def graf():
    # Visualising the clusters
    nu = int(combo_cluster.get())
    print(nu)
    kmeans = KMeans(n_clusters=nu, init='k-means++', max_iter=300, n_init=10, random_state=0)
    y_kmeans = kmeans.fit_predict(x)
    d = {"setosa": 0, "versicolor": 1, "virginica": 2}
    mc = confusion_matrix(dataset.species.map(d), y_kmeans)
    lbl5 = tk.Label(ventana, text=mc, bg="azure", font=("Helvetica", 12))
    lbl5.place(x=80, y=92)
    plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s=100, c='purple', label='Setosa')
    plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s=100, c='blue', label='Versicolour')
    plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s=100, c='green', label='Virginica')
    # Plotting the centroids of the clusters
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=100, c='yellow', label='Centroides')
    plt.legend()
    plt.show()
boton_1 = tk.Button(ventana, text="Mostrar", width=8, command=graf)
boton_1.place(x=450, y=95)
ventana.mainloop()
```

```
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#importing the Iris dataset with pandas
dataset = pd.read_csv('Iris.csv')
x = dataset.iloc[:,:4].values
# Finding the optimum number of clusters for k-means classification
from sklearn.cluster import KMeans

wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)

# Plotting the results onto a line graph, allowing us to observe 'The elbow'
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')  # within cluster sum of squares
plt.show()
```

Interfaz



Gráfica



Conclusión

El programa solicita al usuario que ingrese la cantidad de clusters, para posteriormente generar los centroides de manera aleatoria, debido a que ya cuenta con la base de datos Iris.

Una vez que se seleccionan los clusters y se selecciona el botón mostrar, se abre la interfaz de la gráfica que muestra los centroides y las clases de la base de datos.

Referencias

K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks. (20 de 11 de 2021). Obtenido de https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a