Análisis de Medidas Disimilitud y Similitud

Practica 1.2

Alumnos: Karla Rivera Lima, Natali Meza Barranco, Didier García Toquiantzi, Juan Diego Espinosa Sandoval.

1. Introducción

En el análisis de medidas de similitud han sido utilizadas en la minería de datos y reconocimiento de patrones, en la que se intenta utilizar la medida más adecuada al tipo de análisis, la cual impactara en la obtención de los resultados, tiempo y el proceso.

La medida de similitud es una función cuyo valor es la semejanza entre dos datos, la cual es utilizada en medir las características que son similares. Estos valores determinaran la presencia o ausencia de características, estas medidas nos indicaran la relación por la cual están agrupadas.

En el estudio de dichas medidas es necesario tener ciertas restricciones que nos ayudaran a tener mejores resultados, lo que indica que la medida se encuentre entre un intervalo dado.

2. Metodología

En esta práctica las medidas a utilizar son: Distancia Minkowski, Mahalanobis y el conjunto de datos Iris, donde se analizará la similitud que existe entre los tipos de flor.

En la cual consiste en un enfoque para responder problemas mediante medidas de similitud y disimilitud. A continuación, se describen las actividades realizadas en las que se analizaron las muestras y exceptuando las fases de desarrollo para obtener la implementación de dicha práctica.

3. Desarrollo

3.1 Creación de las funciones que calculen la distancia de Minkowski y Mahalanobis utilizando lenguaje de programación Python con ayuda de colab.

```python
# Distancia euclidiana
def distancia_euclidiana(punto1, punto2):
    return np.linalg.norm(punto1 - punto2)

# Inicializar diccionario que guarde las distancias entre los puntos seleccionados y el resto de los puntos
distancias = {}

# Enumerate para darle seguimiento de la posición y valor al mismo tiempo
for i, punto1 in enumerate(puntos_seleccionados):
    distancias[i] = {}
    for j, punto2 in enumerate(X):
        if i != j:  # Evitar un posible error al calcular distancia en un mismo punto
            dist = distancia_euclidiana(punto1, punto2)
            distancias[i][j] = dist
```

```python
# Distancia mahalanobis
def distancia_mahalanobis(punto1, punto2):
    return np.linalg.norm(punto1 - punto2)

# Inicializar diccionario que guarde las distancias entre los puntos seleccionados y el resto de los puntos
distancias = {}

# Enumerate para darle seguimiento de la posición y valor al mismo tiempo
for i, punto1 in enumerate(puntos_seleccionados):
    distancias[i] = {}
    for j, punto2 in enumerate(X):
        if i != j:  # Evitar un posible error al calcular distancia en un mismo punto
            dist = distancia_mahalanobis(punto1, punto2)
            distancias[i][j] = dist
```

3.2 Obtención del conjunto de datos Iris e implementen el algoritmo de k vecinos cercanos.

```python
# obtencion de conjunto de datos iris
iris = datasets.load_iris()
```

```python
# K vecinos cercanos
k = 6

# Clasificar los puntos evaluados
etiquetas_estimadas = []
for i, dist_dict in distancias.items():
    # Obtener los k vecinos más cercanos
    k_vecinos = sorted(dist_dict.items(), key=lambda x: x[1])[:k]

    # Obtener las etiquetas de los k vecinos
    etiquetas_k_vecinos = [y[j] for j, _ in k_vecinos if j < len(y)]

    # Votar por la clase más común
    clase_estimada = max(set(etiquetas_k_vecinos), key=etiquetas_k_vecinos.count)
    etiquetas_estimadas.append(clase_estimada)
```

3.3 Cálculo de distancias y ordenamiento del menor al mayor

```
distancias
```

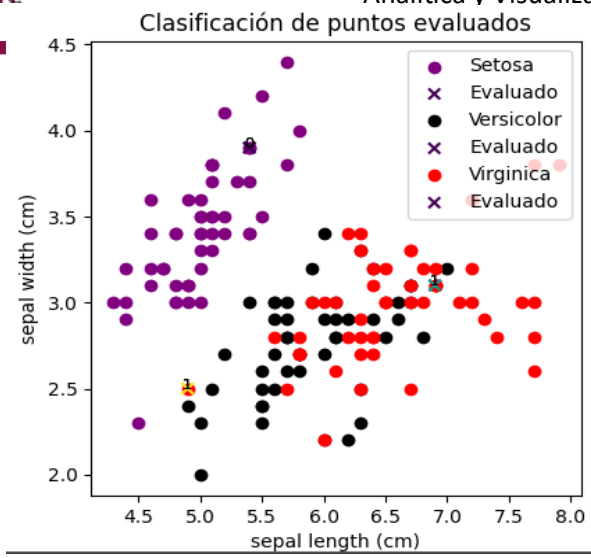
```
  92: 1.1489125293076052,
  93: 1.407124727947029,
  94: 0.883176086632784,
  95: 1.1090536506409414,
  96: 1.0246950765959595,
  97: 1.431782106327635,
  98: 1.6278820596099706,
  99: 1.0246950765959597,
 100: 2.343074902771996,
 101: 1.1180339887498942,
 102: 2.68514431641951,
 103: 1.827566688249706,
 104: 2.1794494717703365,
 105: 3.4799425282610623,
 106: 0.0,
 107: 3.0282007859453435,
 108: 2.2226110770892866,
 109: 3.1144823004794873,
 110: 1.8708286933869702,
 111: 1.7233687939614086,
 112: 2.2405356502408074,
 113: 0.9899494936611664,
 114: 1.3228756555322947,
```

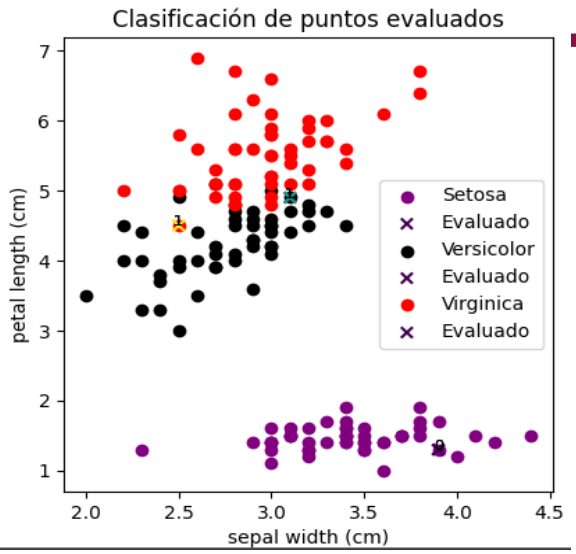
3.4 Se seleccionaron los k elementos del arreglo ordenado.

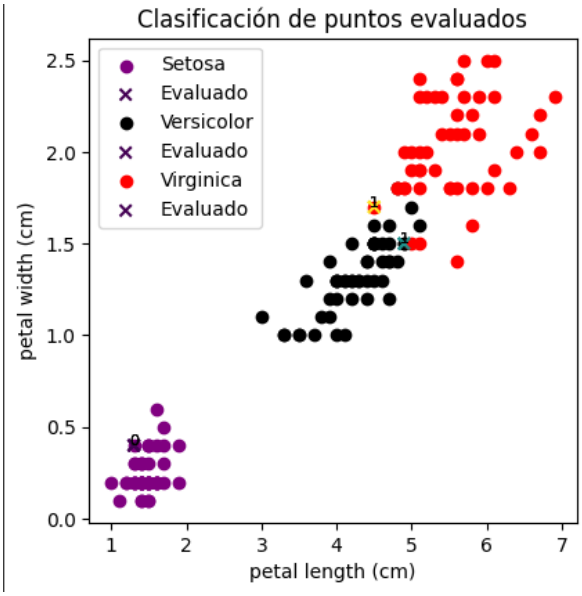
```python
# K vecinos cercanos
k = 6

# Clasificar los puntos evaluados
etiquetas_estimadas = []
for i, dist_dict in distancias.items():
    # Obtener los k vecinos más cercanos
    k_vecinos = sorted(dist_dict.items(), key=lambda x: x[1])[:k]

    # Obtener las etiquetas de los k vecinos
    etiquetas_k_vecinos = [y[j] for j, _ in k_vecinos if j < len(y)]

    # Votar por la clase más común
    clase_estimada = max(set(etiquetas_k_vecinos), key=etiquetas_k_vecinos.count)
    etiquetas_estimadas.append(clase_estimada)
```

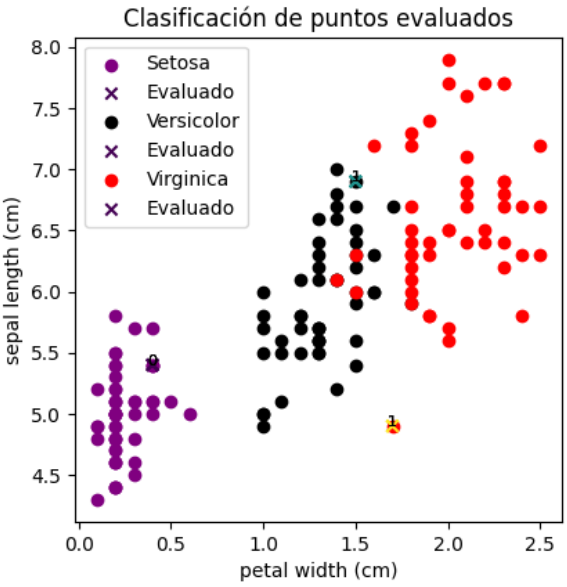
3.5 Graficas de todos los puntos, coloreados cada punto con cada clase, los no evaluados y para cada punto evaluado, elijan un color más y agreguen con texto qué clase se estimó.

```
# Graficar todos los puntos
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
axs = axs.ravel()

# Colores y etiquetas para cada clase
colors = ['purple', 'black', 'red']
labels = ['Setosa', 'Versicolor', 'Virginica']

for i in range(4):
    for j, (color, label) in enumerate(zip(colors, labels)):
        axs[i].scatter(X[y==j, i], X[y==j, (i+1)%4], c=color, cmap=
        'viridis', label=label)
        axs[i].scatter(puntos_seleccionados[:, i], puntos_seleccionados[:,
        (i+1)%4], c=etiquetas_seleccionadas, cmap='viridis', marker='x',
        label='Evaluado')
        for k, punto in enumerate(puntos_seleccionados):
            axs[i].text(punto[i], punto[(i+1)%4], f'{etiquetas_estimadas[k]
            }', fontsize=8, ha='center')
        axs[i].set_xlabel(iris.feature_names[i])
        axs[i].set_ylabel(iris.feature_names[(i+1)%4])
        axs[i].set_title('Clasificación de puntos evaluados')
        axs[i].legend()

plt.show()
```

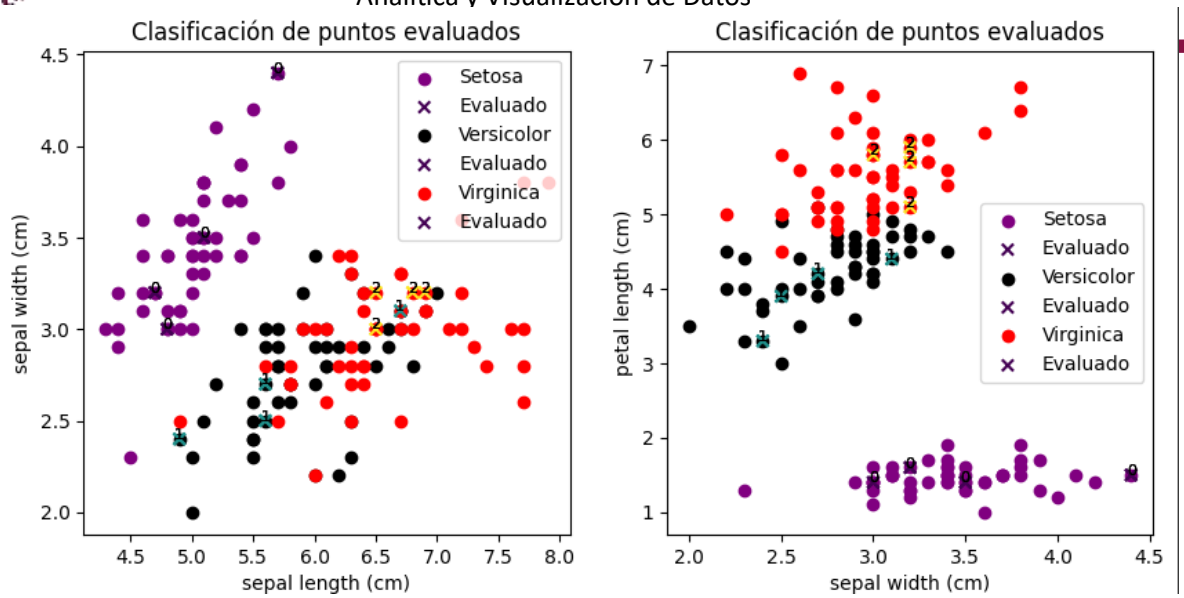








Evaluamos con diferentes valores para p y k, donde las graficas nos muestran las diferentes distancias entre los puntos.



Se utilizo un p=4 y K= 4, las x muestran el valor evaluado y la similitud que se encuentra entre las diferentes especies de iris.

4. Conclusión

En esta práctica se presenta un análisis en relación a medidas de similitud populares en el conjunto de datos iris, utilizando las medidas correspondientes y reconocimiento de datos, este análisis permite ver que estos datos pueden ser agrupados en diferentes clases y existe una gran semejanza. Se puede mencionar que es muy importante el uso de la distancia euclidiana que nos ayuda a identificar la distancia entre dos puntos, por ello el uso de esta medida en tareas de clasificación y reconocimiento de patrones. Se debe mencionar que los datos analizados tienen características en común que nos pondrían a cuestionar si es una especie diferente, en conclusión, podemos mencionar que las medidas de similitud y disimilitud, son muy importantes en el análisis de información y en las que nos podemos apoyar para poder clasificar mejor los datos, y con ellos obtener mejores resultados.