UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

INGENIERO EN COMPUTACIÓN

ANÁLISIS DE ALGORITMOS

MACIEL VARGAS OSWALDO DANIEL

GARCÍA SALDIVAR HUGO GABRIEL

Actividad en equipos #03

Análisis de Clustering con TMAP en base de datos

I. Introducción

Explicación sobre la importancia de la reducción de dimensionalidad

En la época actual de la tecnología los profesionales del area se enfrentan a con conjuntos de datos que son complejos, enormes y en ocasiones difíciles de interpretar con sistemas convencionales. Los datos en la actualidad llevan consigo cientos o miles de variables que pueden describir una enorme cantidad de funciones u objetivos como lo son una imagen, un perfil de usuario, un algoritmo, entre otros. A pesar de su potencial valor al contener múltiples variables en un solo grupo dato, esto plantea un reto crucial llamado "la maldición de la dimensionalidad". Este problema se basa en que la cantidad de datos necesarios para generalizar correctamente aumenta exponencialmente al añadir dimensiones (variables) a un modelo, haciendo que los datos se vuelvan dispersos y dificultando el análisis, la búsqueda de patrones y la validación de modelos de aprendizaje automático.

La dimensionalidad de un conjunto de datos se refiere simplemente al número de características o variables que se utilizan para describir cada observación. Una imagen de 28x28 píxeles, como las del dataset Fashion MNIST, tiene una dimensionalidad de 784 (una por cada píxel). Si bien más dimensiones pueden significar más información, también traen consigo problemas significativos:

- Dispersión de Datos: A medida que aumenta el número de dimensiones, el espacio se vuelve vasto y los puntos de datos se alejan cada vez más entre sí, volviéndose "solitarios". Esto hace que los algoritmos que dependen de la densidad, como el clustering, pierdan efectividad.
- Pérdida de Significado en las Distancias: En espacios de muy alta dimensión, las distancias entre los puntos tienden a volverse uniformes. Es decir, la distancia al vecino más cercano y al más lejano se vuelve casi la misma, haciendo que conceptos como "proximidad" o "similitud" pierdan su significado.
- Costo Computacional: El tiempo de procesamiento y la memoria requerida por los algoritmos de aprendizaje automático a menudo crecen exponencialmente con el número de dimensiones, haciendo que el análisis sea lento o inviable.

Por ello y para poder tener un mejor control sobre los datos con los que se estan trabajando es plantea utilizar métodos de reducción de dimensionalidad. Esta es técnica de procesamiento de datos que disminuye el número de características (dimensiones) en un conjunto de datos,

transformándolos a un espacio de menor dimensión sin perder información significativa ayudando a optimizar y facilitar el uso y evaluación de miles o millones de datos.

Para esta práctica estaremos utilizando TMAP que es una de las herramientas para visualizar y entender el análisis de clústeres de una forma sencilla e intuitiva.

El uso de TMAP en análisis de clústers

TMAP en comparación con técnicas clásicas como PCA (Análisis de Componentes Principales) permite un diseño especializado en la visualizaciones, entendimiento y exploraciones de datos de alta dimensión. TMAP se encarga de posicionar un grupo de puntos similares entre si entre un espacio cercano, pero además con esta lógica construye un grafo que revela todas las relaciones entre los elementos existentes, esta función permite analizar de una mejor manera los clústeres abordar y actuar conforme a la estructura de los datos.

El proceso de TMAP se puede resumir en los siguientes pasos:

1. Indexación Rápida (LSH): TMAP usa una técnica llamada Locality Sensitive Hashing (LSH) para encontrar rápidamente "vecinos aproximados" de cada punto.
2. Construcción del Grafo: Con la información de los vecinos, TMAP construye un grafo, una red donde los nodos son nuestros datos (las imágenes) y las aristas conectan los nodos que son similares entre sí.
3. Cálculo del Árbol de Expansión Mínima (MST): El algoritmo encuentra el "esqueleto" de ese grafo, un subgrafo que conecta todos los puntos sin formar ciclos y con el menor "costo" posible. Este árbol es el que genera las conexiones, ramas y hojas que son tan informativas visualmente.
4. Visualización 2D: Finalmente, TMAP utiliza un algoritmo de layout de grafos para dibujar esta estructura de árbol/grafo en un espacio 2D, creando el mapa final.

De esta forma TMAP nos permite visualizar los clústeres lejanos de un grupo de datos de una mejor manera y con un mejor entendimiento, además tener la información ya separada de esta manera es muy útil para aplicar sub-clusters en ella con el objetivo de encontrar relaciones específicas de cada grupo específico de datos.

Finalmente, TMAP nos permite visualizar de una manera muy agradable a la vista como es la transición entre clúster, mostrándonos como dos clústeres que en principio deberían de ser completamente diferentes tambien tienen ciertas similitudes entre ellos, pero no por ello se trata de

un mismo clúster.

II. Objetivos

El propósito de esta actividad es mediante la reducción de dimensionalidad TMAP, aplicar esta técnica a un conjunto de imágenes fashion-mnist en un archivo .csv para diferenciar los diferentes tipos de modas dentro de este conjunto de datos. Después de obtener el cluster general que engloba a todas las modas, se tomará un subcluster al que le aplicaremos de nuevo la técnica de reducción de dimensionalidad TMAP para ahora diferenciar los tipos de una moda en específico.

Al final mediante la biblioteca matplotlib mostraremos un conjunto de imágenes que representen diferentes subcluster en ese subcluster de la moda en específico para poder identificar las diferencias de forma visual.

III. Desarrollo

Explicación del procedimiento realizado en cada paso:

Para cumplir correctamente con esta actividad tomamos como base el código de Fashion MNIST de la página https://tmap.gdb.tools/ y respectivamente utilizaremos una base de datos “fashion-mnist_test.csv” está es un documento de excel la cual contiene una serie de valores que hacen referencia a los pixeles obtenidos a partir de distintas imágenes de ropas y prendas, esto con el objetivo de que tmap pueda encontrar ciertas similitudes entre estos valores dados y que cree gráficamente un archivo html para visualizar de una mejor manera los cluster existentes y ya definidos para esta base de datos.

Para este punto, nosotros ya ejecutamos el código y vimos gráficamente cómo crea y se comporta Tmap con la información importada, ahora lo que buscamos es explorar aún más alguno de los clusters que nos entregó el código con el objetivo de definir un grupo de sub-clusters del original, para este caso utilizaremos los bolsos (bag) porque nos pareció un accesorio cuyas cualidades específicas son más fáciles de identificar a simple vista (en comparación a una playera, por ejemplo) esto nos permitirá identificar de una mejor manera los sub-clusters existentes.

Código y visualizaciones obtenidas: para nuestro código, primeramente, haremos uso del cluster original por lo que será necesario crearlo a partir de nuestros datos, esto se hizo en una función generate_full_cluster que recibe como parámetro los datos de nuestro archivo de excel y

crea un archivo .html que contiene la gráfica de los cluster completos.

<codigo>

Ahora volveremos a usa esta misma lógica, pero para el cluster de los bolsos que sería el número 8, la función generate_bag_subcluster esta función igualmente recibirá como parámetro los datos de nuestro archivo de excel y entregará como resultado un archivo .html con el contenido gráfico del sub-cluster de bolsos.

Ahora lo que sigue es hacer “zoom” en el subcluster creado de las bolsas para lograr esto implementamos la siguiente función que encuentra relaciones con las imágenes y las agrupa, en este caso tuvimos que encontrar un valor óptimo para el número de subcluster pues si el valor era muy grande los sub-clusters a partir de las fotos seria muy únicos e independientes, pero si el valor era muy bajo los sub-cluster serían muy generales, por lo que intentando y viendo encontramos que 4 sub-clusters es lo óptimo.

kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)

Código de las funciones principales:

```
# --- FUNCIÓN PARA EL CLÚSTER ORIGINAL (TODAS LAS PRENDAS) ---
def generate_full_cluster(df):
    print("\n--- Iniciando la generación del clúster completo ---")
    LABELS = df['label'].values
    IMAGES = df.drop('label', axis=1).values

    IMAGE_LABELS = []

    dims = 1024
    enc = tm.Minhash(28 * 28, 42, dims)
    lf = tm.LSHForest(dims * 2, 128)

    print("Convirtiendo todas las imágenes...")
```

```
for image in IMAGES:
    img = Image.fromarray(np.uint8(np.split(np.array(image), 28)))
    buffered = BytesIO()
    img.save(buffered, format="JPEG")
    img_str = base64.b64encode(buffered.getvalue())
    IMAGE_LABELS.append(
        "" + str(img_str).replace("b'", "").replace("'", "")
    )

tmp = [tm.VectorFloat(image / 255) for image in IMAGES]

print("Ejecutando tmap en el dataset completo...")
start = timer()
lf.batch_add(enc.batch_from_weight_array(tmp))
lf.index()
x, y, s, t, _ = tm.layout_from_lsh_forest(lf, CFG)
print(f"tmap (completo) finalizado en: {timer() - start:.2f}s")

legend_labels = [
    (0, "T-shirt/top"), (1, "Trouser"), (2, "Pullover"), (3, "Dress"),
    (4, "Coat"), (5, "Sandal"), (6, "Shirt"), (7, "Sneaker"),
    (8, "Bag"), (9, "Ankle boot"),
]

faerun = Faerun(clear_color="#111111", view="front", coords=False)
faerun.add_scatter(
    "FMNIST",
    {"x": x, "y": y, "c": LABELS, "labels": IMAGE_LABELS},
    colormap="tab10", shader="smoothCircle", point_scale=2.5,
    max_point_size=10, has_legend=True, categorical=True,
    legend_labels=legend_labels,
```

```
    )
    faerun.add_tree(
        "FMNIST_tree", {"from": s, "to": t}, point_helper="FMNIST", color="#666666"
    )
    faerun.plot("fmnist", template="url_image")
    print(" Visualización 'fmnist.html' generada correctamente.")

# --- FUNCIÓN PARA EL SUBCLÚSTER DE BOLSOS ---
def generate_bag_subcluster(df):
    print("\n--- Iniciando la generación del subcluster de bolsos ---")
    bags_df = df[df['label'] == 8].copy()
    IMAGES = bags_df.drop('label', axis=1).values

    print(f"Se encontraron {len(IMAGES)} bolsos para el subcluster.")

    IMAGE_LABELS = []

    dims = 1024
    enc = tm.Minhash(28 * 28, 42, dims)
    lf = tm.LSHForest(dims * 2, 128)

    print("Convirtiendo las imágenes de los bolsos...")
    for image in IMAGES:
        img = Image.fromarray(np.uint8(np.split(np.array(image), 28)))
        buffered = BytesIO()
        img.save(buffered, format="JPEG")
        img_str = base64.b64encode(buffered.getvalue())
        IMAGE_LABELS.append(
            "" + str(img_str).replace("b'", "").replace("'", "")
        )
```

```
tmp = [tm.VectorFloat(image / 255) for image in IMAGES]

print("Ejecutando tmap en el subcluster de bolsos...")
start = timer()
lf.batch_add(enc.batch_from_weight_array(tmp))
lf.index()
x, y, s, t, _ = tm.layout_from_lsh_forest(lf, CFG)
print(f"tmap (bolsos) finalizado en: {timer() - start:.2f}s")

print("Creando 4 subcategorías de bolsos con KMeans...")
coords = np.vstack((x, y)).T
kmeans = KMeans(n_clusters=4, random_state=42, n_init=10)
new_labels = kmeans.fit_predict(coords)

# <--- 2. INICIO DEL NUEVO BLOQUE CON MATPLOTLIB ---
print("Generando visualización de imágenes aleatorias de cada clúster...")
n_clusters = 4
n_examples = 9

fig, axs = plt.subplots(n_clusters, n_examples, figsize=(10, 5))
fig.suptitle('Muestras Aleatorias de Cada Clúster de Bolsos', fontsize=16)

for i in range(n_clusters):
    indices = np.where(new_labels == i)[0]

    # Prevenir error si un clúster tiene menos de 9 imágenes
    num_to_sample = min(n_examples, len(indices))
    if num_to_sample > 0:
        random_indices = np.random.choice(indices, size=num_to_sample, replace=False)
    else:
        random_indices = []
```

```
    axs[i, 0].set_ylabel(f"Tipo {i+1}", rotation=0, size='large', labelpad=30)

    for j, idx in enumerate(random_indices):
        image = IMAGES[idx].reshape(28, 28)
        ax = axs[i, j]
        ax.imshow(image, cmap='gray')
        ax.axis('off')

    # Ocultar ejes de subplots no utilizados
    for j in range(num_to_sample, n_examples):
        axs[i, j].axis('off')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
# <--- FIN DEL NUEVO BLOQUE ---

legend_labels_bags = [
    (0, "Agarradera corta"), (1, "Maleta/Maletín"), (2, "Cartera"),
    (3, "Agarradera larga"),
]

faerun = Faerun(clear_color="#111111", view="front", coords=False)
faerun.add_scatter(
    "BolsosFMNIST",
    {"x": x, "y": y, "c": new_labels, "labels": IMAGE_LABELS},
    colormap="tab10", shader="smoothCircle", point_scale=2.5,
    max_point_size=10, has_legend=True, categorical=True,
    legend_labels=legend_labels_bags,
)
faerun.add_tree(
```

```
        "BolsosFMNIST_tree", {"from": s, "to": t}, point_helper="BolsosFMNIST", color="#666666"
    )
    faerun.plot("fmnist_bolsos", template="url_image")
    print("Visualización 'fmnist_bolsos.html' generada correctamente.")
```

Análisis de los clusters y sub-clusters identificados:

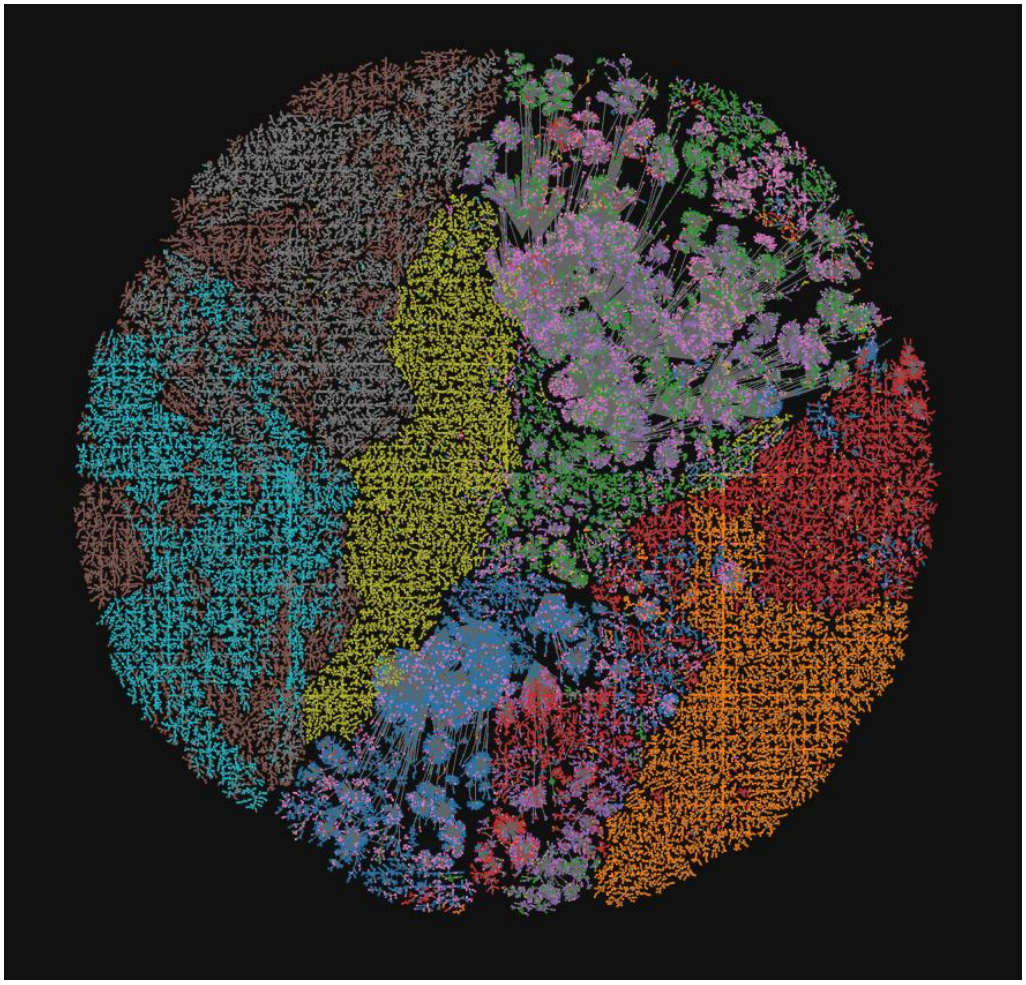
Como lo mencionamos previamente en el objetivo, aplicamos dos veces la técnica de reducción de dimensionalidad, primero a un conjunto de imágenes con distintos tipos de moda y de nuevo a otro conjunto de imágenes, pero con un solo tipo de moda.

En nuestro código, al ejecutarlo, obtenemos ambos clusters, así que empezaremos con el grande. Para estas pruebas usamos el archivo de datos “fashion-mnist_test.csv” y “fashion-mnist_train.csv”, la diferencia es que el de test contiene 10,000 imágenes y el de train contiene aproximadamente 60,000 imágenes, para este análisis tomaremos en cuenta del más grande para observar más detalles.

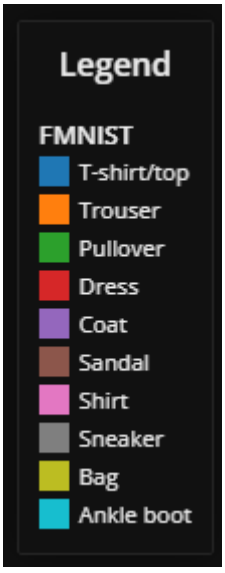
Nuestro programa primero procesa los datos para mostrar el primer cluster, por lo que al inicio es muy tardado, una vez procesado, obtenemos como resultado un archivo html correspondiente al primer cluster:

Gracias a este archivo podemos analizar de manera visual los cluster, esto pasa igual con el subcluster de la moda en específico la cual fue de bolsos.

Al abrir ese archivo nos muestra el cluster gigante, el de 60,000 imágenes:

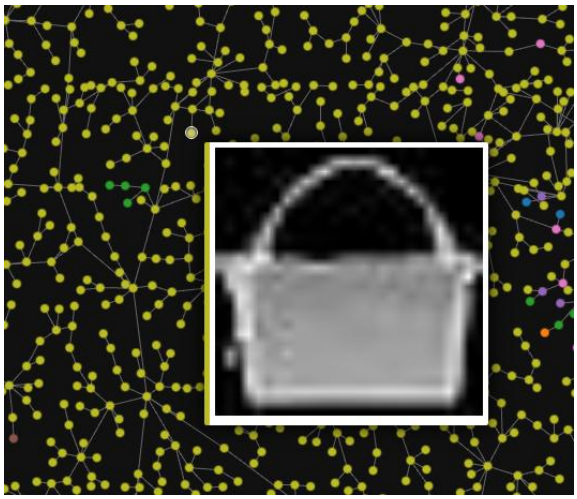


En este cluster, cada punto de color representa un tipo de moda en específico, la leyenda de colores y su respectiva moda se muestra a continuación:



Gracias a esta diferenciación podemos indagar más a fondo en nuestro cluster. Lo que observamos a simple vista es que hay 3 colores que predominan, es decir, estos colores raramente se traslapan con otros, estos son: amarillo, rojo y naranja, los cuales muestran respectivamente bolsa, vestido y pantalón.

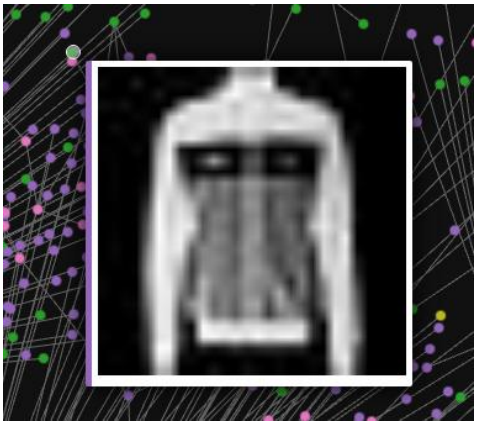
Aquí podemos observar un bolso en la región amarilla:

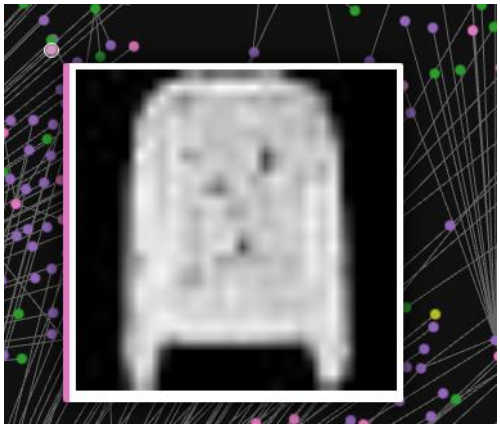


Gracias a esta imagen nos damos cuenta de una cosa curiosa, la reducción de dimensionalidad no es perfecta, hay puntos que corresponden a otra moda, lo cual quiere decir algo muy importante, y es que nuestro algoritmo como tal no tiene un sistema de inteligencia artificial el cual aprende a diferencias las imágenes, nuestro algoritmo simplemente hace comparaciones con números que representan pixeles, por lo tanto, podemos encontrar moda en un sitio el cual claramente no le corresponde.

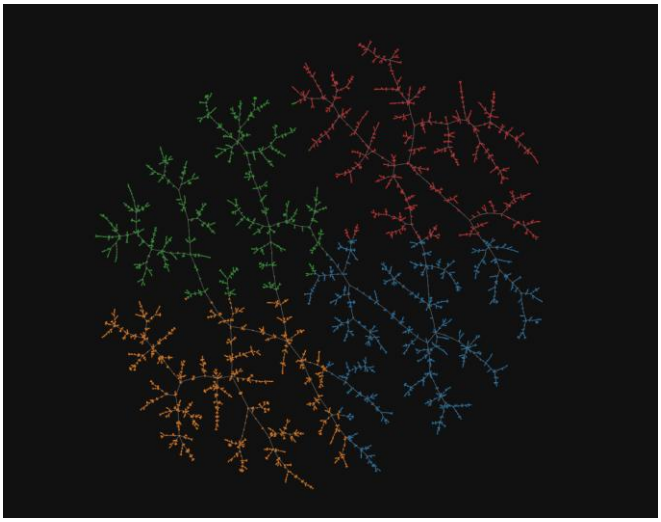
¿Por qué tiene un color diferente entonces? El color confunde, si distingue para poner otro color, por qué pondría una moda donde no le toca, eso es sencillo, el color lo asigna ya que en el archivo csv hay un label el cual nos indica con un número cada tipo de moda, pero el algoritmo solo ve una clasificación por números, no por cómo se ve, es por ello por lo que sí distingue el color.

Un ejemplo claro de lo anterior lo podemos ver en las regiones con verde, rosa y azul, ya que esas prendas para el torso visualmente son demasiado parecidas, incluso se parecen más dos prendas de diferente tipo que dos del mismo. Es por eso que hay una combinación exagerada de estos colores, para nuestro algoritmo estás prendas van juntas porque en lo que respecta a sus pixeles se parecen, pero se ven de diferente color porque así vienen indicado en el archivo csv con el label:





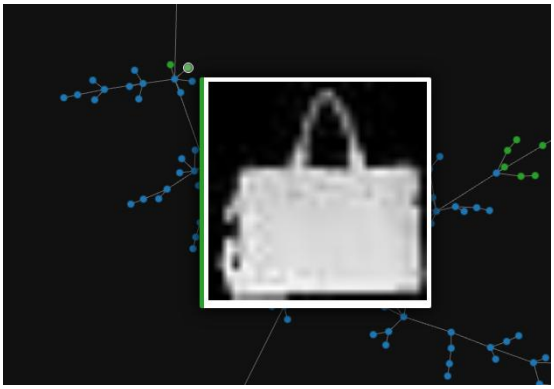
Ahora pasamos a nuestro subcluster elegido, el cual es el de los bolsos. En este subcluster obviamente la densidad de puntos será mucho menor pero aún aspi podemos ver muchos detalles, a continuación, se muestra este subcluster:

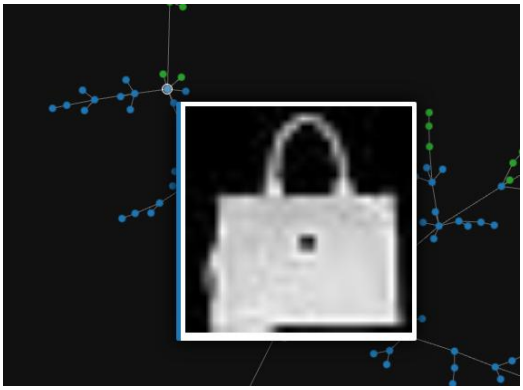


En este subcluster ya pasa algo interesante, como aquí no tenemos un label que nos indique cuales son los diferentes tipos de bolsos, obviamente nuestro programa tendrá más fallas.

La principal falla es que, en algunas regiones de un color, podemos encontrar ya ni siquiera otros colores, si no imágenes que no tienen absolutamente nada que ver, esto pasa porque nuestro programa esta coloreando mal los puntos por lo explicado anteriormente.

Realmente es dificil avanzar a partir de aquí, un siguiente paso seguría integrar tensorflow para poder identificar cada imagen de mejor manera. Gracias a esto nos dimos cuenta de una cosa muy importante, y es que el algoritmo es demasiado bueno relacionando unas imágenes con otroas, pero muy malo para colorear los puntos, ya que requiere que anteriormente se le específique que representa cada uno de ellos.





IV. Conclusión

Oswaldo Maciel:

En esta actividad hubo de todo, lo principal, en muchas ocasiones me sacó de mis casillas. Al principio empezamos bien, aprendiendo sobre tmap y el conjunto de datos, revisamos la actividad y empezamos a trabajar. Ya que necesitábamos entregar un avance, empezamos con un código de prueba, aquí empezaron los problemas. Al intentar usar los códigos tuvimos errores con tmap en Python, intentamos de todo, instalamos múltiples versiones, cambiamos de código, hicimos varios entornos virtuales, instalamos y desinstalamos cosas, etc. Al final dimos con el error, era cuestión de compatibilidad, pero pudimos solucionarlo a un archivo. yml para obtener las bibliotecas exactas. Implementar el código principio fue sencillo, ya teníamos las bases, para el subcluster tardamos un poco más, ya que tuvimos errores con la lógica hasta que hicimos uso de un dataframe para separar los datos que necesitábamos para el subcluster.

A pesar de que se complicó demasiado la instalación de la librería de tmap, el proceso para realizar la actividad fue demasiado entretenido, algo muy interesante es que previo a esta actividad, no sabíamos de entornos virtuales más que el nombre, pero debido a que intentamos de mil y un formas instalar tmap, aprendimos demasiado sobre como implementar estos entornos, como funcionan, etc. Fue muy gratificante como en búsqueda de una solución, aprendimos mucho de otros temas.

Hugo Garcia:

La realización de esta actividad fue muy interesante más que por la utilización de bibliotecas, por la pequeña experiencia que adquirimos sobre la reducción de dimensional y el análisis de datos, fue bastante interesante el visualizar los distintos sub-clusters que se creaban a partir del cluster de las bolsas, más que nada porque como equipo tuvimos que concluir un estándar para nuestra sub-clusterización de tal forma que los resultados de las imágenes fueran únicos, claros y significativos para pertenecer a su cluster.

Algo que no fue de mi agrado para esta actividad fueron las complicaciones que tuvimos en el camino al instalar la biblioteca tmap pues hicimos muchos intentos hasta cambiamos de sistema operativo, pero no funciona correctamente, al final la hicimos funcionar y el código se ejecutó correctamente.

V. Referencias

- Fashion MNIST. (2017, 7 diciembre). Kaggle. https://www.kaggle.com/datasets/zalando-research/fashionmnist
- Probst, D. (s. f.). tmap - Visualize big high-dimensional data. https://tmap.gdb.tools/
- Anaconda. (2025, 9 octubre). Download Anaconda Distribution | Anaconda. https://www.anaconda.com/download