Título: Sistema de recomendación de imágenes utilizando CNN y KNeighbors
Integrantes: Alan Ignacio López Carrillo y Mauricio Escorza Cantú

Explicación del conjunto de datos:

a) ¿Cuál será el conjunto de datos que usarás?
   Utilizaremos imágenes de productos de ropa para niño, niña, hombre y mujer.
b) ¿Dónde lo conseguiste?
   En Kaggle.
c) Liga a la base de datos:
   https://www.kaggle.com/datasets/vikashrajluhaniwal/fashion-images

Preguntas a resolver sobre el conjunto de datos:

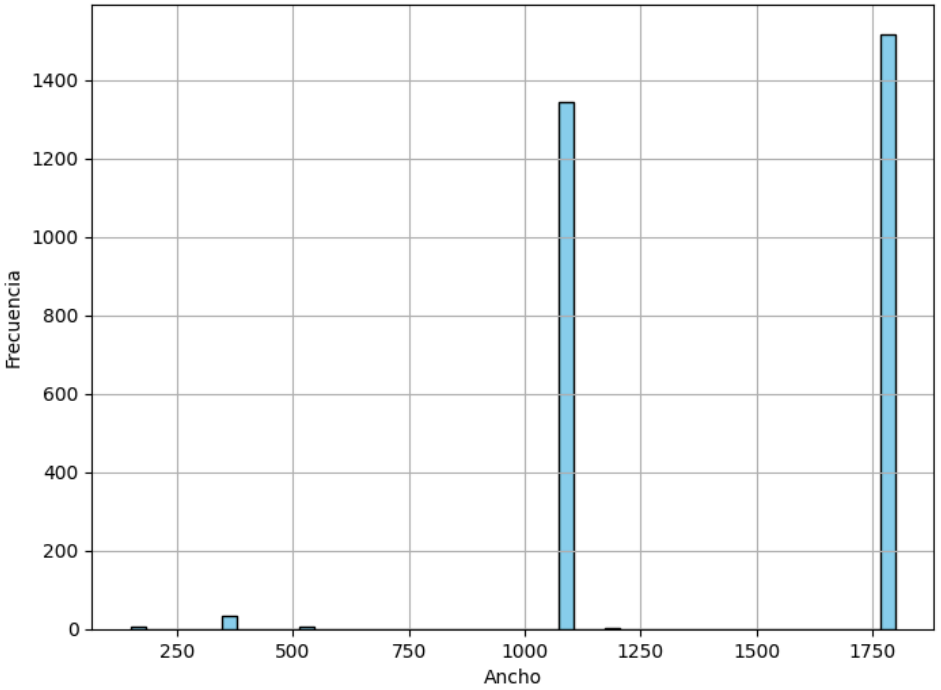
a) ¿Qué problema deseas resolver?
   El objetivo es desarrollar un sistema de recomendación robusto y eficaz. En numerosas ocasiones al considerar la compra de un producto en línea, las recomendaciones ofrecidas por las tiendas suelen ser irrelevantes o poco adecuadas para la intención de compra del cliente. Esta discrepancia conduce a una disminución en las conversiones de ventas, ya que la falta de precisión en las recomendaciones puede afectar negativamente la experiencia del usuario.
b) ¿Por qué es interesante resolver este problema usando redes neuronales?
   Este desafío se beneficia de la aplicación de técnicas avanzadas como las redes neuronales debido a la complejidad y la naturaleza visual de los datos. La visión por computadora, habilitada por redes neuronales convolucionales (CNN), permite extraer características significativas de las imágenes de productos y utilizar estas representaciones para agrupar y recomendar productos relevantes de manera más precisa.
c) Variables de entrada: imágenes de productos de ropa para niño, niña, hombre y mujer.
   Variables de salida: imágenes de productos similares basadas en las características de la imagen de entrada.

Etapa 4

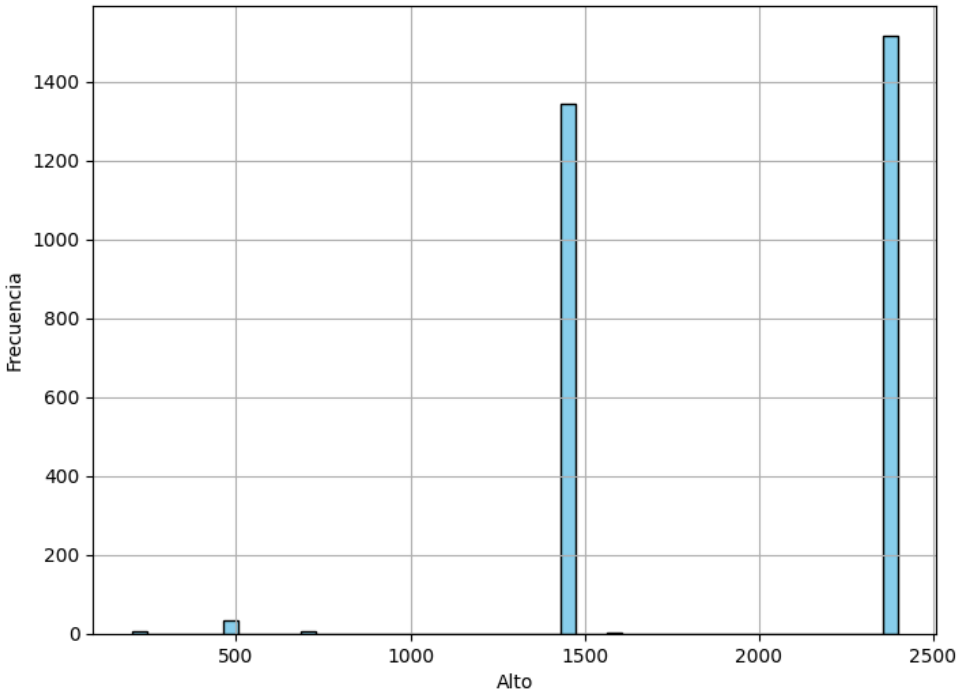
Dado que nuestra base de datos está compuesta por imágenes en lugar de datos tabulares, nuestro enfoque actual se centra en redimensionar las imágenes para facilitar el proceso de entrenamiento. En primer lugar, generaremos un histograma para visualizar la distribución de alturas y anchos de las imágenes en nuestra base de datos. Esto nos permitirá identificar las dimensiones más comunes y ver qué tanto afectará el redimensionamiento de estas imágenes.

Nuestra base de datos contiene un total de 2,906 imágenes a color, es decir, con 3 canales RGB. A continuación se presenta la distribución de anchos y alturas de las imágenes:

Distribución de la anchura de las imágenes



Distribución de la altura de las imágenes



Por último, es importante mencionar que como nuestro proyecto no es de clasificación, no adjuntamos una gráfica de barras de clases. Nuestro proyecto se enfocará en obtener vectores de características de las imágenes para agruparlas. Por lo mismo, no se realizó una división en conjuntos de entrenamiento, validación y

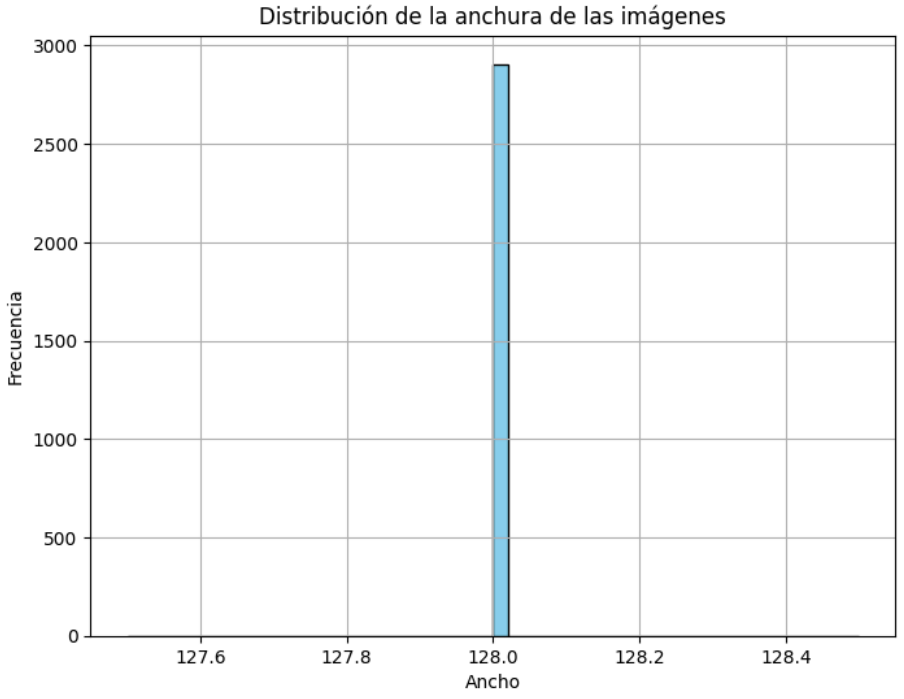
prueba. Utilizaremos un modelo (ResNet50), para la extracción de características, que ya ha sido entrenado en un conjunto grande y diverso de imágenes (ImageNet).

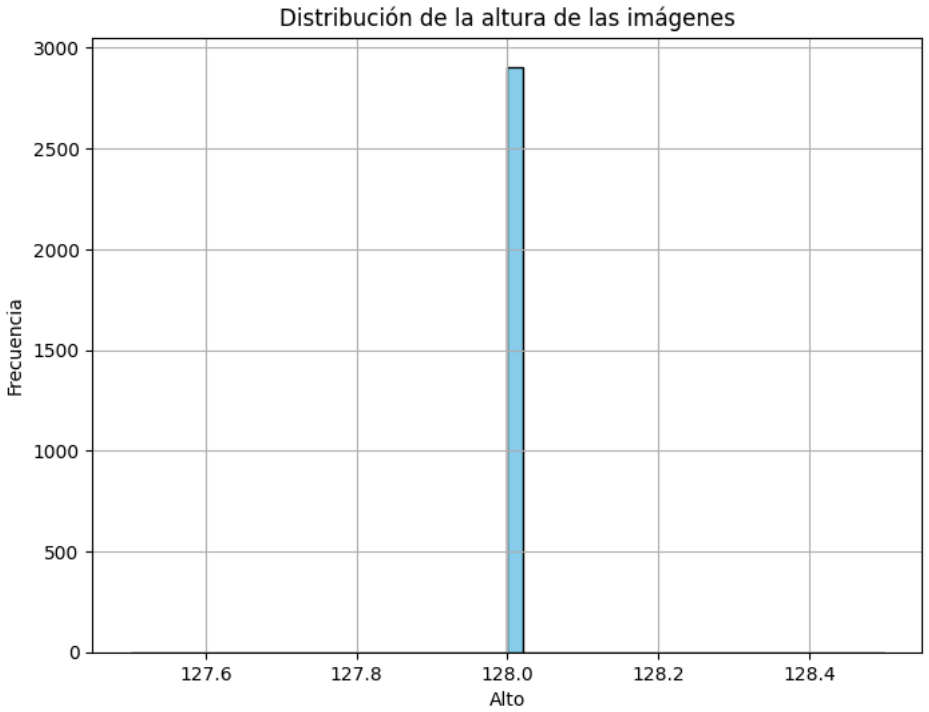
Etapa 5

Al revisar el dataset y analizar los histogramas, observamos que la mayoría de las imágenes tienen un tamaño de 1800 x 2400 píxeles. También encontramos una cantidad significativa de imágenes con dimensiones de 1100 x 1400 píxeles, mientras que algunas imágenes son más pequeñas, con dimensiones como 360 x 480 píxeles u otras.

Mantener las imágenes en estas dimensiones originales puede ser problemático, ya que requeriría una cantidad considerable de nodos y conexiones en nuestra red neuronal. Por esta razón, es necesario redimensionar las imágenes. Inicialmente, intentaremos redimensionarlas a 128 x 128 píxeles. Aunque esta dimensión sigue siendo relativamente grande para la entrada de la red, si encontramos problemas durante el entrenamiento, como tiempos de ejecución prolongados, consideraremos reducir aún más las dimensiones a 64 x 64 píxeles o incluso a 32 x 32 píxeles.

Ahora, todas las imágenes quedaron de la misma dimensión:





- Comentario de ayudante

Etapa 6:
En esta etapa, nos enfocaremos en el proceso de entrenamiento del modelo utilizando una red neuronal convolucional preentrenada y la implementación de un sistema de recomendación basado en vecinos más cercanos. A continuación, enumeramos cada fase de la etapa 6:

1. Arquitectura de la red
   ResNet50 es una red neuronal convolucional (CNN) profunda que se utiliza comúnmente para tareas de clasificación y extracción de características de imágenes. Esta red consta de 50 capas, incluyendo convolucionales, de activación, y de pooling. Vamos a eliminar la parte superior de la red y la reemplazaremos con una capa de Global Max Pooling. Esto permite transformar una imagen de entrada en un vector de características, manteniendo la información más relevante.

2. Definición del Modelo
   Para construir nuestro sistema de recomendación de imágenes, comenzamos con una red neuronal convolucional (CNN) pre-entrenada. El modelo pre-entrenado se carga con los pesos del conjunto de datos ImageNet, y se excluye la capa superior (`include_top=False`), ya que solo necesitamos las capas convolucionales para la extracción de características.

```
model = ResNet50(weights='imagenet', include_top=False,
input_shape=(128,128,3))
model.trainable = False
```

A continuación, encapsulamos el modelo dentro de una secuencia de Keras, añadiendo una capa de pooling global para reducir las dimensiones de salida y obtener un vector de características compacto.

```
model = tensorflow.keras.Sequential([
    model,
    GlobalMaxPooling2D()
])
```

3. Extracción de características
   Para cada imagen en nuestro conjunto de datos, necesitamos convertirla en un vector de características. Este vector representará la imagen en el espacio de características de alta dimensión, que es crucial para el agrupamiento y la recomendación de imágenes similares.

   Para ello, se utilizará la función `extract_feature` que se encarga de:
   I. Leer la imagen y convertirla en un array.
   II. Expandir la dimensión de la imagen para que sea compatible con la entrada del modelo.
   III. Preprocesar la imagen utilizando la función `preprocess_input` de ResNet50.
   IV. Obtener las características mediante el modelo y normalizar el resultado.

```
def extract_feature(img_path, model):
    img = cv2.imread(img_path)
    img = np.array(img)
    expand_img = np.expand_dims(img, axis=0)
    pre_img = preprocess_input(expand_img)
    result = model.predict(pre_img).flatten()
    normalized = result / norm(result)
    return normalized
```

Recorremos todas las imágenes en el directorio especificado, extrayendo y almacenando los vectores de características y los nombres de archivo correspondientes.

```
filename = []
feature_list = []
```

```
folder_path = "/content/drive/MyDrive/Redes
Neuronales/Proyecto/data"

for file in os.listdir(folder_path):
    filename.append(os.path.join(folder_path, file))

for file in tqdm(filename):
    feature_list.append(extract_feature(file, model))

pickle.dump(feature_list, open('featurevector.pk', 'wb'))
pickle.dump(filename, open('filenames.pkl', 'wb'))
```

4. Entrenamiento del Sistema de Recomendación
   Utilizaremos K-Nearest Neighbors para encontrar imágenes similares basadas en las características extraídas por la ResNet50. Cargamos los vectores de características y los nombres de los archivos guardados previamente.

   ```
   feature_list2 =
   np.array(pickle.load(open("/content/featurevector.pk",
   "rb")))
   filename2 = pickle.load(open("/content/filenames.pkl",
   "rb"))
   ```

   Entrenamos el modelo de KNeighbors con los vectores de características. Utilizamos la métrica euclidiana y el método brute para calcular las distancias entre los puntos en el espacio de características.

   ```
   neighbors = NearestNeighbors(n_neighbors=6,
   algorithm="brute", metric="euclidean")
   neighbors.fit(feature_list2)
   ```

5. Recomendación de imágenes
   Para realizar una recomendación, primero extraemos las características de la imagen de consulta de la misma manera que lo hicimos durante la extracción de características iniciales.

   ```
   img2 = cv2.imread("/content/drive/MyDrive/Redes
   Neuronales/Proyecto/data/img (1999).jpg")
   img2 = np.array(img2)
   expand_img2 = np.expand_dims(img2, axis=0)
   pre_img2 = preprocess_input(expand_img2)
   result2 = model.predict(pre_img2).flatten()
   normalized2 = result2 / norm(result2)
   ```

Usamos el modelo de KNeighbors para encontrar las imágenes más similares a la imagen de consulta.

```
distance, indices = neighbors.kneighbors([normalized2])
```

Finalmente, mostramos las imágenes recomendadas.

```
for file in indices[0][1:6]:
    img_name = cv2.imread(filename2[file])
    cv2_imshow(img_name)
    cv2.waitKey(0)
```

Ejemplos:

input

recomendaciones

input

recomendaciones

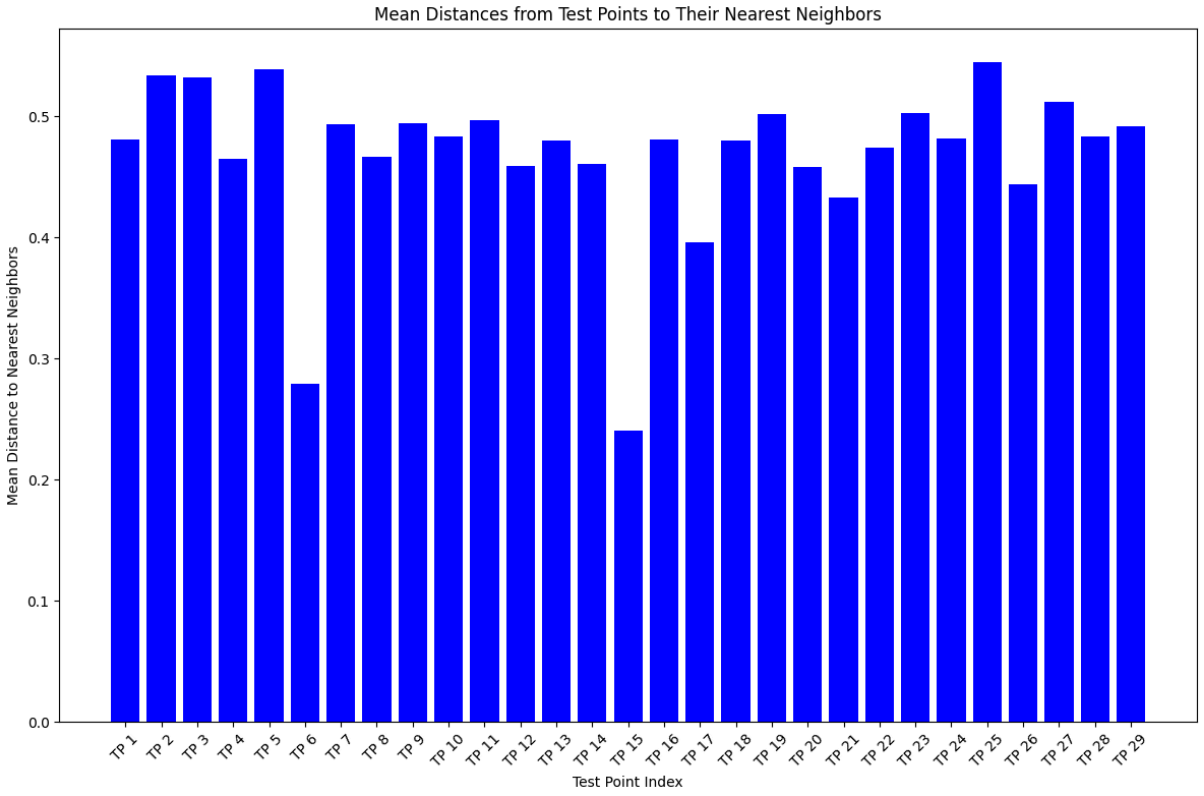
Etapa 7:
En esta última etapa, se analizan los resultados del entrenamiento del modelo y de múltiples recomendaciones para determinar la efectividad del mismo.

Utilizamos una métrica de similitud para visualizar la cercanía en distancias entre la imagen de entrada y las recomendaciones del modelo. Para esto, calculamos las **distancias promedio** de cada punto con los puntos de sus recomendaciones. Esta métrica es útil para evaluar cuán similares son, en promedio, los artículos recomendados al artículo de entrada. Una menor distancia promedio sugiere que las recomendaciones son más similares al artículo de entrada, lo cual es generalmente deseable en un sistema de recomendaciones basado en similitudes.

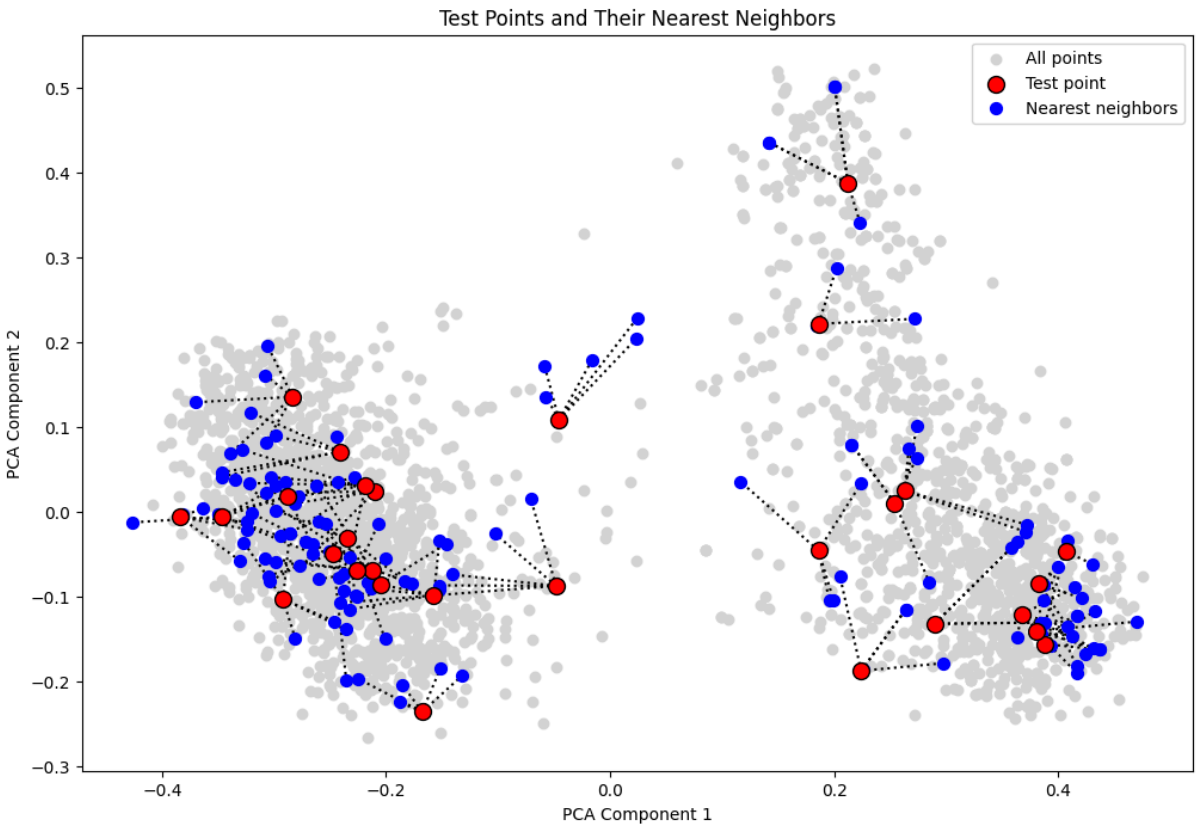
```
Test point 0: Mean distance to nearest neighbors = 0.48055142908021026
Test point 1: Mean distance to nearest neighbors = 0.533891961724943
Test point 2: Mean distance to nearest neighbors = 0.5322263947815316
Test point 3: Mean distance to nearest neighbors = 0.46474056983787154
Test point 4: Mean distance to nearest neighbors = 0.5390347462660837
Test point 5: Mean distance to nearest neighbors = 0.278928945444052
Test point 6: Mean distance to nearest neighbors = 0.4933976861976645
Test point 7: Mean distance to nearest neighbors = 0.466824300256103
Test point 8: Mean distance to nearest neighbors = 0.49378344205464336
Test point 9: Mean distance to nearest neighbors = 0.482940558705592
Test point 10: Mean distance to nearest neighbors = 0.49653599657555697
Test point 11: Mean distance to nearest neighbors = 0.45906490245517273
Test point 12: Mean distance to nearest neighbors = 0.4802901665344916
Test point 13: Mean distance to nearest neighbors = 0.4606818942865481
Test point 14: Mean distance to nearest neighbors = 0.2408729707753334
Test point 15: Mean distance to nearest neighbors = 0.48078222460484543
Test point 16: Mean distance to nearest neighbors = 0.39628547525341523
Test point 17: Mean distance to nearest neighbors = 0.479904783856332
Test point 18: Mean distance to nearest neighbors = 0.5016984063304775
Test point 19: Mean distance to nearest neighbors = 0.4578750508808637
Test point 20: Mean distance to nearest neighbors = 0.432724028497325555
Test point 21: Mean distance to nearest neighbors = 0.47433925401285926
Test point 22: Mean distance to nearest neighbors = 0.5024678741964632
Test point 23: Mean distance to nearest neighbors = 0.4818741109458768
Test point 24: Mean distance to nearest neighbors = 0.5447781245869007
Test point 25: Mean distance to nearest neighbors = 0.4437056815107776
Test point 26: Mean distance to nearest neighbors = 0.5117402926435809
Test point 27: Mean distance to nearest neighbors = 0.48298004343440676
Test point 28: Mean distance to nearest neighbors = 0.4918435894739499
```

```
Test point 0 (Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2472).jpg)
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2478).jpg, Distance: 0.5523077730496938
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2326).jpg, Distance: 0.5765120972723081
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2476).jpg, Distance: 0.5776680093836969
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2325).jpg, Distance: 0.5858497808281017
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2471).jpg, Distance: 0.5909709139474607

Test point 1 (Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2060).jpg)
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (77).jpg, Distance: 0.5463462453243074
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2071).jpg, Distance: 0.6541554767923164
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2022).jpg, Distance: 0.6660709525924438
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2894).jpg, Distance: 0.6679627267666347
    Neighbor Filename: /content/drive/MyDrive/Redes Neuronales (Vero)/Proyecto/data/img (2767).jpg, Distance: 0.6688163688739553
```



Posteriormente, utilizamos el algoritmo de PCA para reducir la cantidad de variables a 2 y así poder analizar la similitud de características de cada imagen visualmente en un diagrama bidimensional. En este diagrama, las líneas punteadas representan las recomendaciones para cada valor de entrada.



Una posible razón por la cual obtuvimos distancias promedio alrededor de 0.5 en nuestro sistema de recomendación podría ser la combinación de dos factores principales. Primero, al reducir las imágenes a 128x128 píxeles, es posible que hayamos afectado la extracción precisa de características visuales, lo cual repercute en comparaciones menos precisas entre las imágenes. Segundo, la diversidad limitada en ciertos tipos de prendas o estilos dentro de nuestro conjunto de datos también puede contribuir a distancias ligeramente mayores entre algunas recomendaciones. A pesar de estos desafíos, nuestros análisis indican que el resultado de recomendación sigue siendo aceptable y útil para la mayoría de los casos de uso previstos.