assignment-9

October 30, 2024

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[3]:
```
df = pd.read_csv('visual-taxonomy\\train.csv', na_values='nan')
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70213 entries, 0 to 70212
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   id        70213 non-null  int64 
 1   Category  70213 non-null  object
 2   len       70213 non-null  int64 
 3   attr_1    51867 non-null  object
 4   attr_2    55192 non-null  object
 5   attr_3    54698 non-null  object
 6   attr_4    59888 non-null  object
 7   attr_5    56493 non-null  object
 8   attr_6    38116 non-null  object
 9   attr_7    41415 non-null  object
 10  attr_8    37474 non-null  object
 11  attr_9    33565 non-null  object
 12  attr_10   24999 non-null  object
dtypes: int64(2), object(11)
memory usage: 7.0+ MB
```

[5]:
```
df.head()
```

[5]:

	id	Category	len	attr_1	attr_2	attr_3	attr_4	attr_5	\
0	0	Men Tshirts	5	default	round	printed	default	short sleeves	
1	1	Men Tshirts	5	multicolor	polo	solid	solid	short sleeves	
2	2	Men Tshirts	5	default	polo	solid	solid	short sleeves	
3	3	Men Tshirts	5	multicolor	polo	solid	solid	short sleeves	
4	4	Men Tshirts	5	multicolor	polo	solid	solid	short sleeves	

```
  attr_6 attr_7 attr_8 attr_9 attr_10
0    NaN    NaN    NaN    NaN     NaN
1    NaN    NaN    NaN    NaN     NaN
2    NaN    NaN    NaN    NaN     NaN
3    NaN    NaN    NaN    NaN     NaN
4    NaN    NaN    NaN    NaN     NaN
```

[6]:
```
df['Category'].unique()
```

[6]:
```
array(['Men Tshirts', 'Sarees', 'Kurtis', 'Women Tshirts',
       'Women Tops & Tunics'], dtype=object)
```

1 Task 1

I have used .info() to find the attributes with less number of nan values and have used them as main attributes for the basket and created baskets for the same

[7]:
```
menTshirts = df[df['Category'] == 'Men Tshirts']
menTshirts.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 7267 entries, 0 to 7266
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        7267 non-null   int64
 1   Category  7267 non-null   object
 2   len       7267 non-null   int64
 3   attr_1    6010 non-null   object
 4   attr_2    6144 non-null   object
 5   attr_3    5791 non-null   object
 6   attr_4    5949 non-null   object
 7   attr_5    5977 non-null   object
 8   attr_6    0 non-null      object
 9   attr_7    0 non-null      object
 10  attr_8    0 non-null      object
 11  attr_9    0 non-null      object
 12  attr_10   0 non-null      object
dtypes: int64(2), object(11)
memory usage: 794.8+ KB
```

[36]:
```
menTBasket1 = pd.DataFrame(menTshirts, columns=['Category', 'attr_1', 'id']).
 ↪sample(100, random_state=42).reset_index(drop=True)
menTBasket2 = pd.DataFrame(menTshirts, columns=['Category', 'attr_3', 'id']).
 ↪sample(100, random_state=42).reset_index(drop=True)
```

[9]:
```
sarees = df[df['Category'] == 'Sarees']
sarees.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18346 entries, 7267 to 25612
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        18346 non-null  int64
 1   Category  18346 non-null  object
 2   len       18346 non-null  int64
 3   attr_1    7885 non-null   object
 4   attr_2    17679 non-null  object
 5   attr_3    15861 non-null  object
 6   attr_4    17896 non-null  object
 7   attr_5    17649 non-null  object
 8   attr_6    5010 non-null   object
 9   attr_7    8896 non-null   object
 10  attr_8    16465 non-null  object
 11  attr_9    14303 non-null  object
 12  attr_10   17818 non-null  object
dtypes: int64(2), object(11)
memory usage: 2.0+ MB
```

[10]:
```
sareeBasket1 = pd.DataFrame(sarees, columns=['Category', 'attr_4', 'id']).
 ↪sample(100, random_state=42).reset_index(drop=True)
sareeBasket2 = pd.DataFrame(sarees, columns=['Category', 'attr_5', 'id']).
 ↪sample(100, random_state=42).reset_index(drop=True)
```

[11]:
```
kurtis = df[df['Category'] == 'Kurtis']
kurtis.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6822 entries, 25613 to 32434
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        6822 non-null   int64
 1   Category  6822 non-null   object
 2   len       6822 non-null   int64
 3   attr_1    6629 non-null   object
 4   attr_2    3231 non-null   object
 5   attr_3    3400 non-null   object
 6   attr_4    6431 non-null   object
 7   attr_5    3266 non-null   object
 8   attr_6    3848 non-null   object
 9   attr_7    3843 non-null   object
```

```
 10  attr_8    6702 non-null   object
 11  attr_9    6691 non-null   object
 12  attr_10   0 non-null      object
dtypes: int64(2), object(11)
memory usage: 746.2+ KB
```

[12]:
```
kurtiBasket1 = pd.DataFrame(kurtis, columns=['Category', 'attr_1', 'id']).
 ↪sample(100, random_state=42).reset_index(drop=True)
kurtiBasket2 = pd.DataFrame(kurtis, columns=['Category', 'attr_9', 'id']).
 ↪sample(100, random_state=42).reset_index(drop=True)
```

[13]:
```
womenT = df[df['Category'] == 'Women Tshirts']
womenT.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18774 entries, 32435 to 51208
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        18774 non-null  int64
 1   Category  18774 non-null  object
 2   len       18774 non-null  int64
 3   attr_1    17285 non-null  object
 4   attr_2    14801 non-null  object
 5   attr_3    16580 non-null  object
 6   attr_4    16161 non-null  object
 7   attr_5    17034 non-null  object
 8   attr_6    16041 non-null  object
 9   attr_7    15460 non-null  object
 10  attr_8    510 non-null    object
 11  attr_9    0 non-null      object
 12  attr_10   0 non-null      object
dtypes: int64(2), object(11)
memory usage: 2.0+ MB
```

[14]:
```
womenTBasket1 = pd.DataFrame(womenT, columns=['Category', 'attr_1', 'id']).
 ↪sample(100, random_state=42).reset_index(drop=True)
womenTBasket2 = pd.DataFrame(womenT, columns=['Category', 'attr_3', 'id']).
 ↪sample(100, random_state=42).reset_index(drop=True)
```

[15]:
```
topsNTunics = df[df['Category'] == 'Women Tops & Tunics']
topsNTunics.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 19004 entries, 51209 to 70212
Data columns (total 13 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
```

```
 0   id          19004 non-null  int64
 1   Category    19004 non-null  object
 2   len         19004 non-null  int64
 3   attr_1      14058 non-null  object
 4   attr_2      13337 non-null  object
 5   attr_3      13066 non-null  object
 6   attr_4      13451 non-null  object
 7   attr_5      12567 non-null  object
 8   attr_6      13217 non-null  object
 9   attr_7      13216 non-null  object
 10  attr_8      13797 non-null  object
 11  attr_9      12571 non-null  object
 12  attr_10     7181 non-null   object
dtypes: int64(2), object(11)
memory usage: 2.0+ MB
```

[16]:
```
topsNTunicsBasket1 = pd.DataFrame(topsNTunics, columns=['Category', 'attr_1','id']).sample(100).reset_index(drop=True)
topsNTunicsBasket2 = pd.DataFrame(topsNTunics, columns=['Category', 'attr_6','id']).sample(100).reset_index(drop=True)
```

2 Task 2

[17]:
```
from PIL import Image
```

[18]:
```
data = []
for id in menTBasket1['id']:
    img = Image.open('visual-taxonomy\\train_images\\' + str(id).zfill(6) + '.jpg').resize((128,128))
    pixels = np.array(img)
    data.append(pixels)

data = np.array(data)
data.shape
num_images, height, width, channels = data.shape
data_reshaped = data.reshape(num_images, height * width * channels)
data_reshaped.shape
```

[18]: (100, 49152)

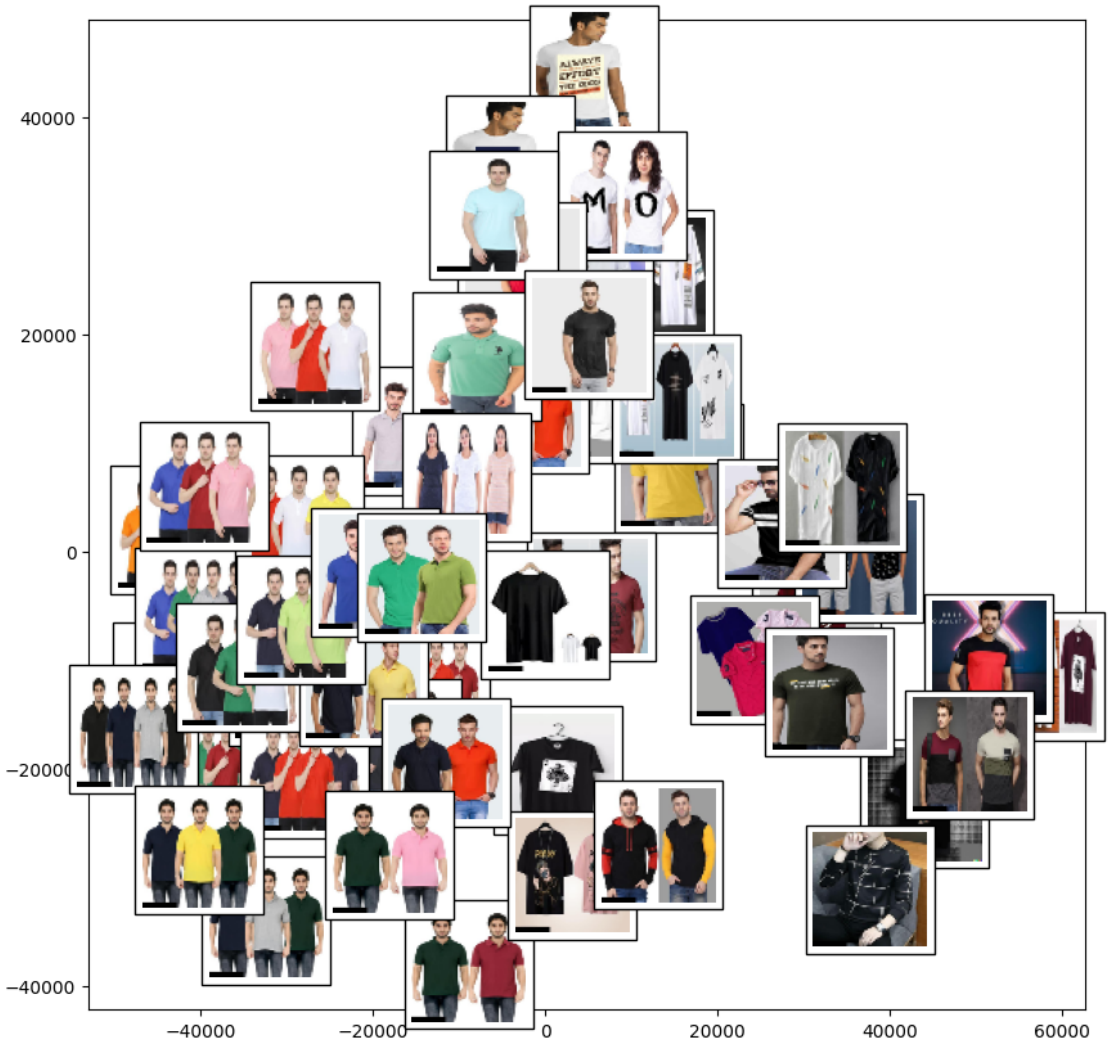
[19]:
```
from sklearn.manifold import Isomap
model = Isomap(n_components=2)
proj = model.fit_transform(data_reshaped)
proj.shape
```

[19]: (100, 2)

[20]:
```
import matplotlib.pyplot as plt
from matplotlib import offsetbox

def plot_components(data, model, images=None, ax=None,
                    thumb_frac=0.05, cmap='gray'):
    ax = ax or plt.gca()

    proj = model.fit_transform(data)
    ax.plot(proj[:, 0], proj[:, 1], '.k')

    if images is not None:
        min_dist_2 = (thumb_frac * max(proj.max(0) - proj.min(0))) ** 2
        shown_images = np.array([2 * proj.max(0)])
        for i in range(data.shape[0]):
            dist = np.sum((proj[i] - shown_images) ** 2, 1)
            if np.min(dist) < min_dist_2:
                continue
            shown_images = np.vstack([shown_images, proj[i]])
            imagebox = offsetbox.AnnotationBbox(
                offsetbox.OffsetImage(images[i], cmap=cmap),
                proj[i]
            )
            ax.add_artist(imagebox)
```

Plot the visualization

[21]:
```
downsampled_images = data[:, ::2, ::2, :]

fig, ax = plt.subplots(figsize=(10, 10))
plot_components(data.reshape(data.shape[0], -1),
                model=Isomap(n_components=2),
                images=downsampled_images)
plt.show()
```



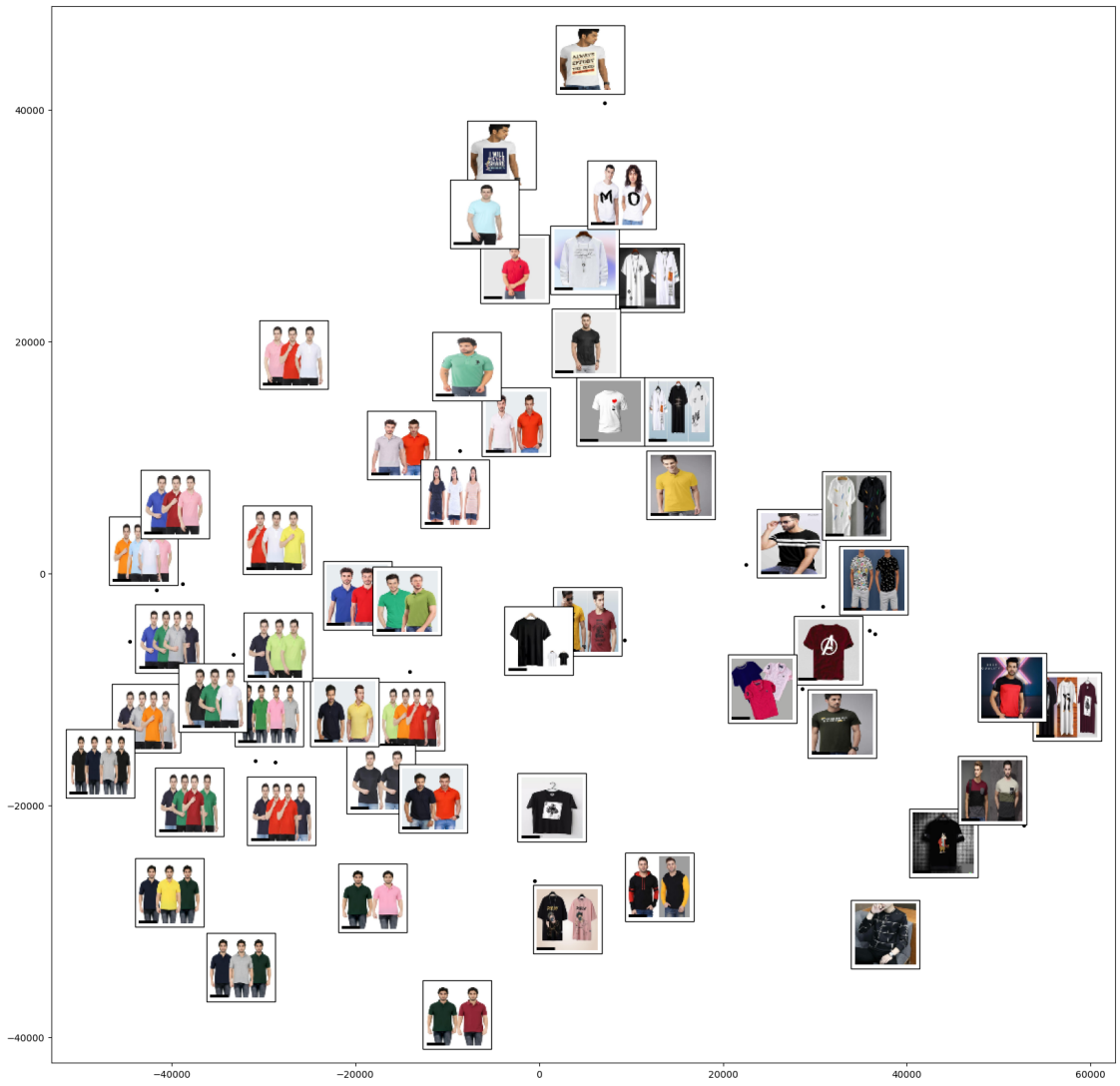
Similarly for all combinations

2.0.1 Isomap

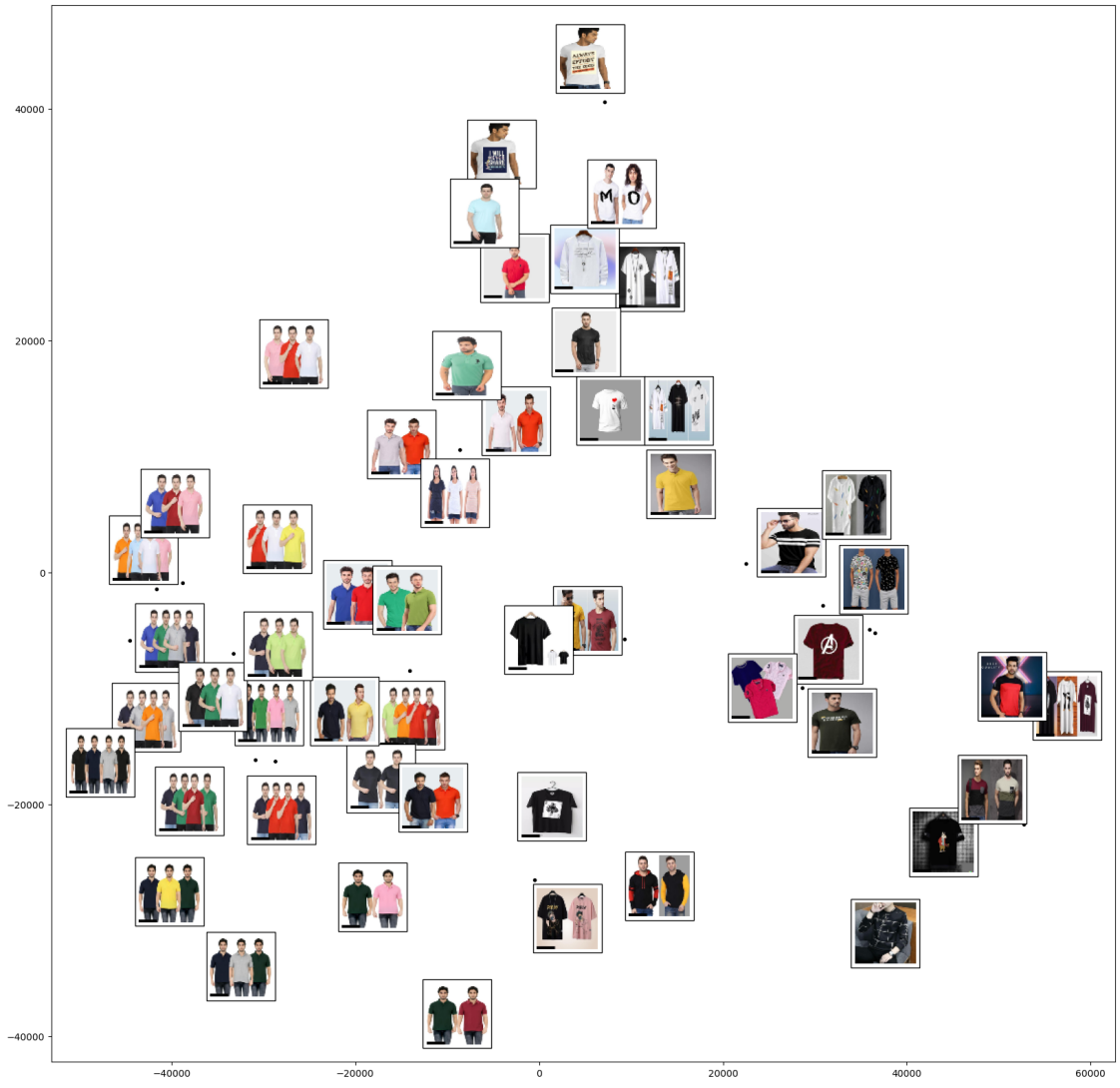
[37]:
```
baskets = ['menTBasket1', 'menTBasket2', 'sareeBasket1', 'sareeBasket2',
           'kurtiBasket1', 'kurtiBasket2', 'womenTBasket1', 'womenTBasket2',
           'topsNTunicsBasket1', 'topsNTunicsBasket2']

for basket in baskets:
    data = []
    for id in globals()[basket]['id']:
        img = Image.open('visual-taxonomy/train_images/' + str(id).zfill(6) + '.
 ↪jpg').resize((128, 128))
        pixels = np.array(img)
        data.append(pixels)
```

```
    data = np.array(data)
    num_images, height, width, channels = data.shape
    data_reshaped = data.reshape(num_images, height * width * channels)

    model = Isomap(n_components=2, n_neighbors=5)
    proj = model.fit_transform(data_reshaped)
    downsampled_images = data[:, ::2, ::2, :]

    fig, ax = plt.subplots(figsize=(20, 20))
    plot_components(data_reshaped, model=model, images=downsampled_images,␣
↪ax=ax)

    print(f"Projection for {basket}")
    plt.show()
```

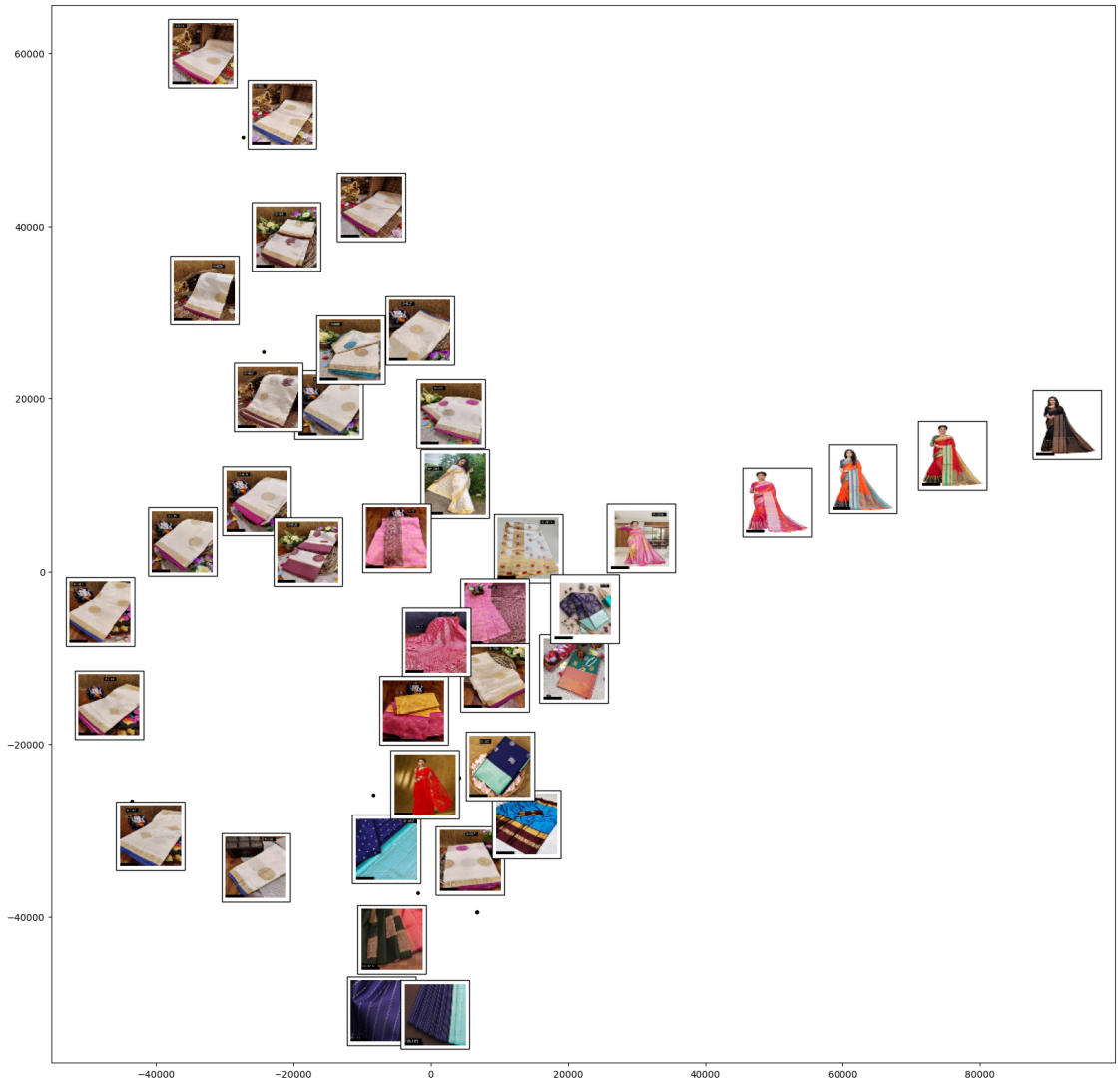
```
Projection for menTBasket1
```



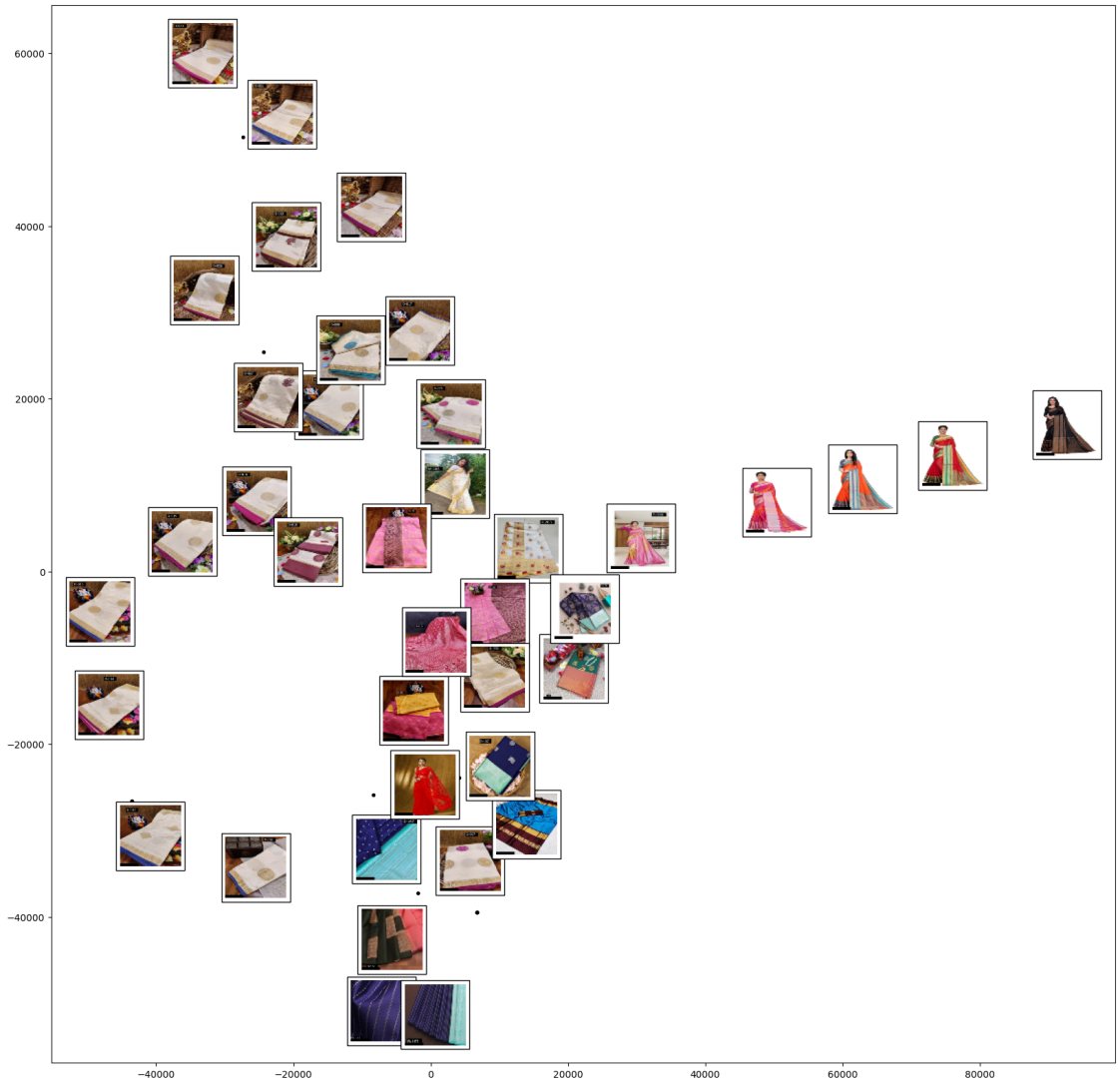
Projection for menTBasket2



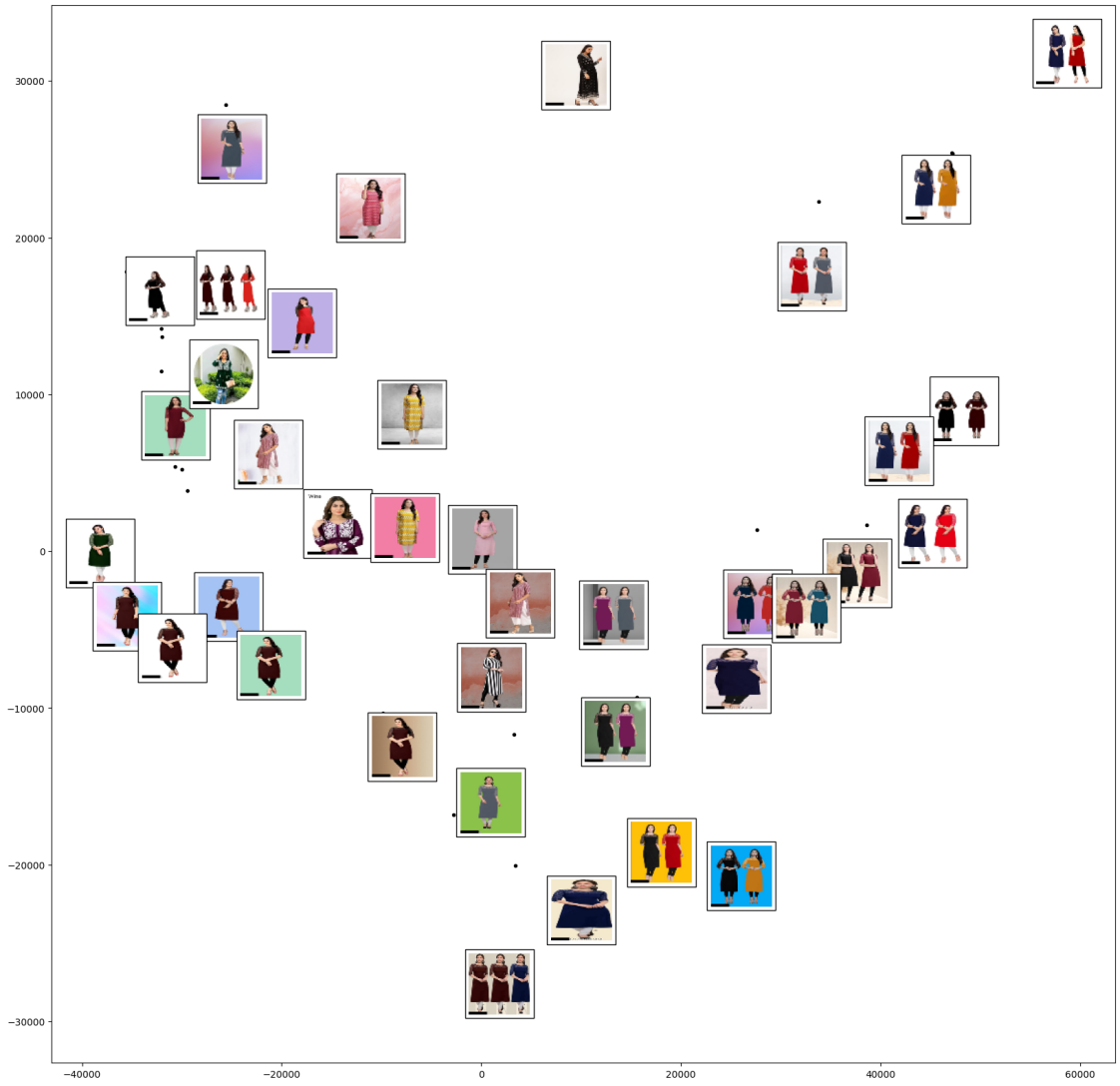
Projection for sareeBasket1



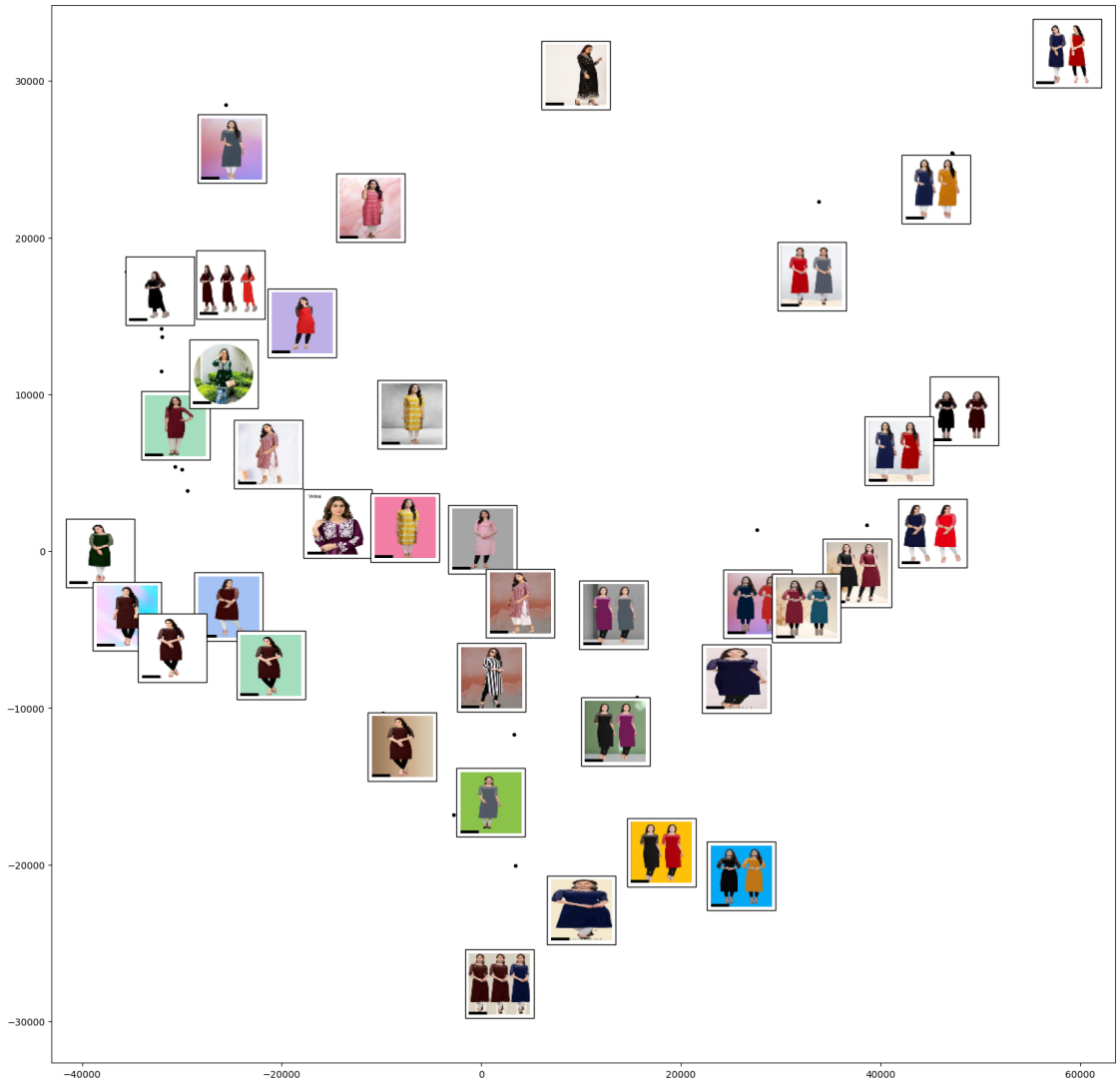
Projection for sareeBasket2



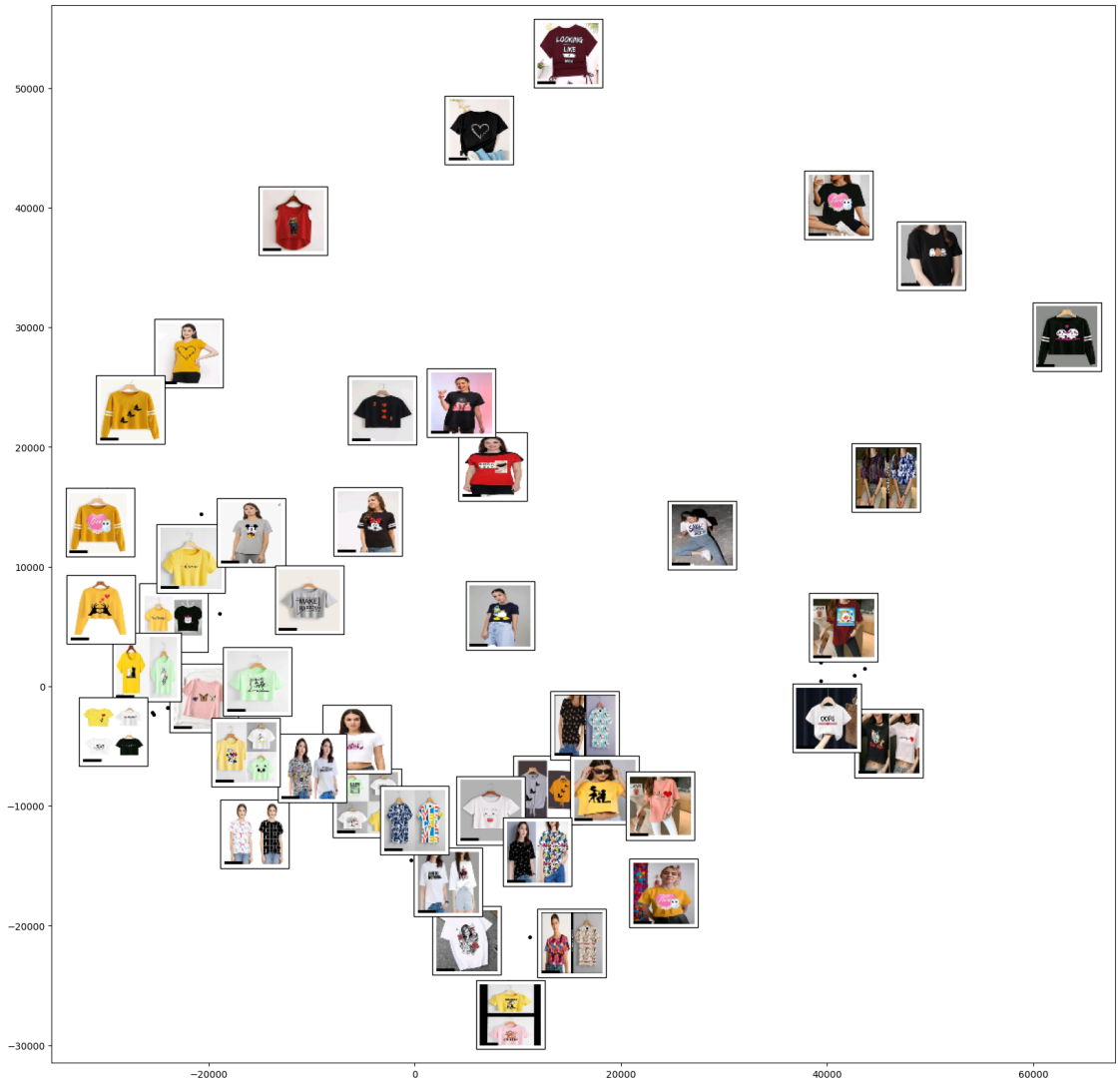
Projection for kurtiBasket1



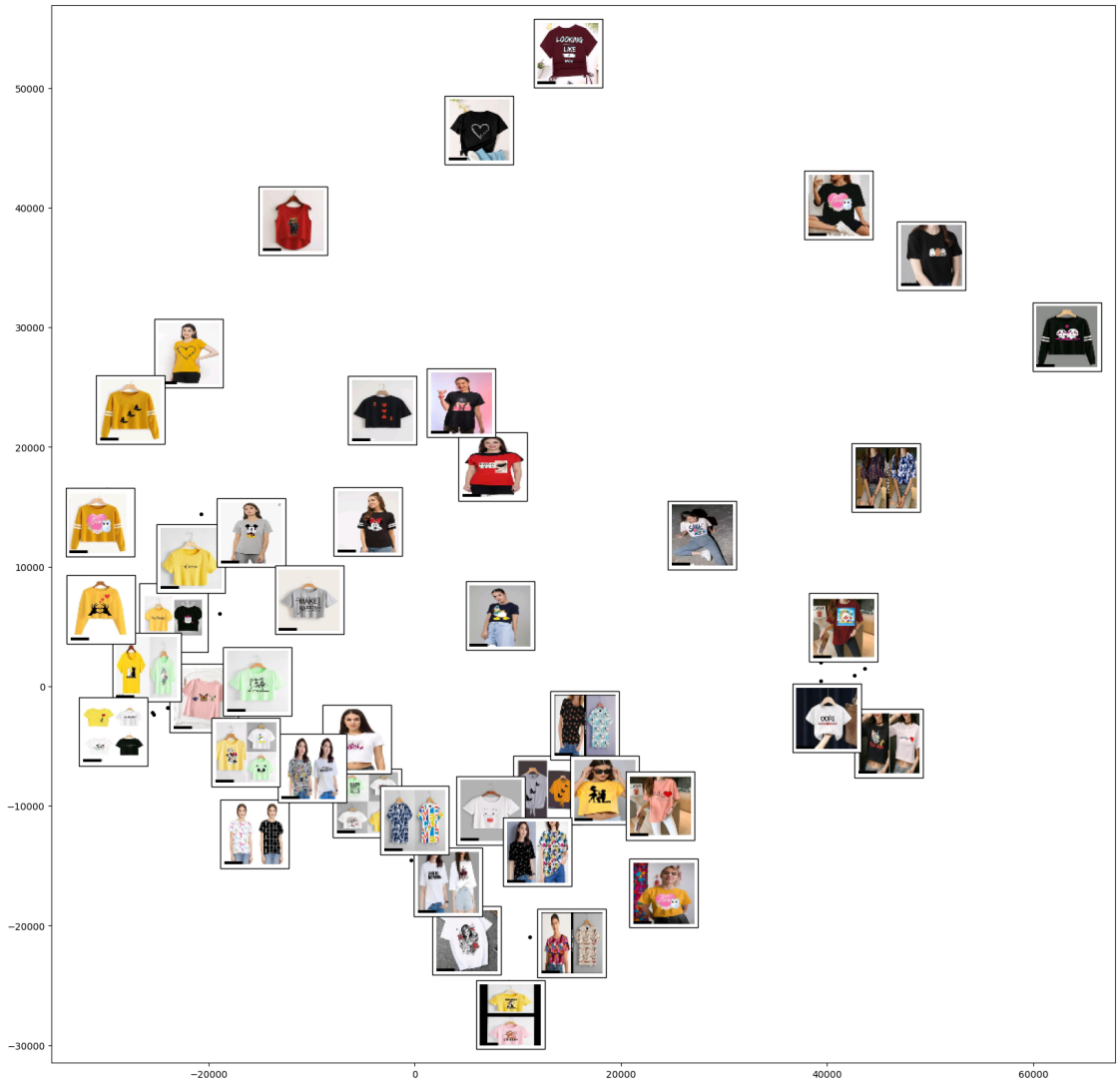
Projection for kurtiBasket2



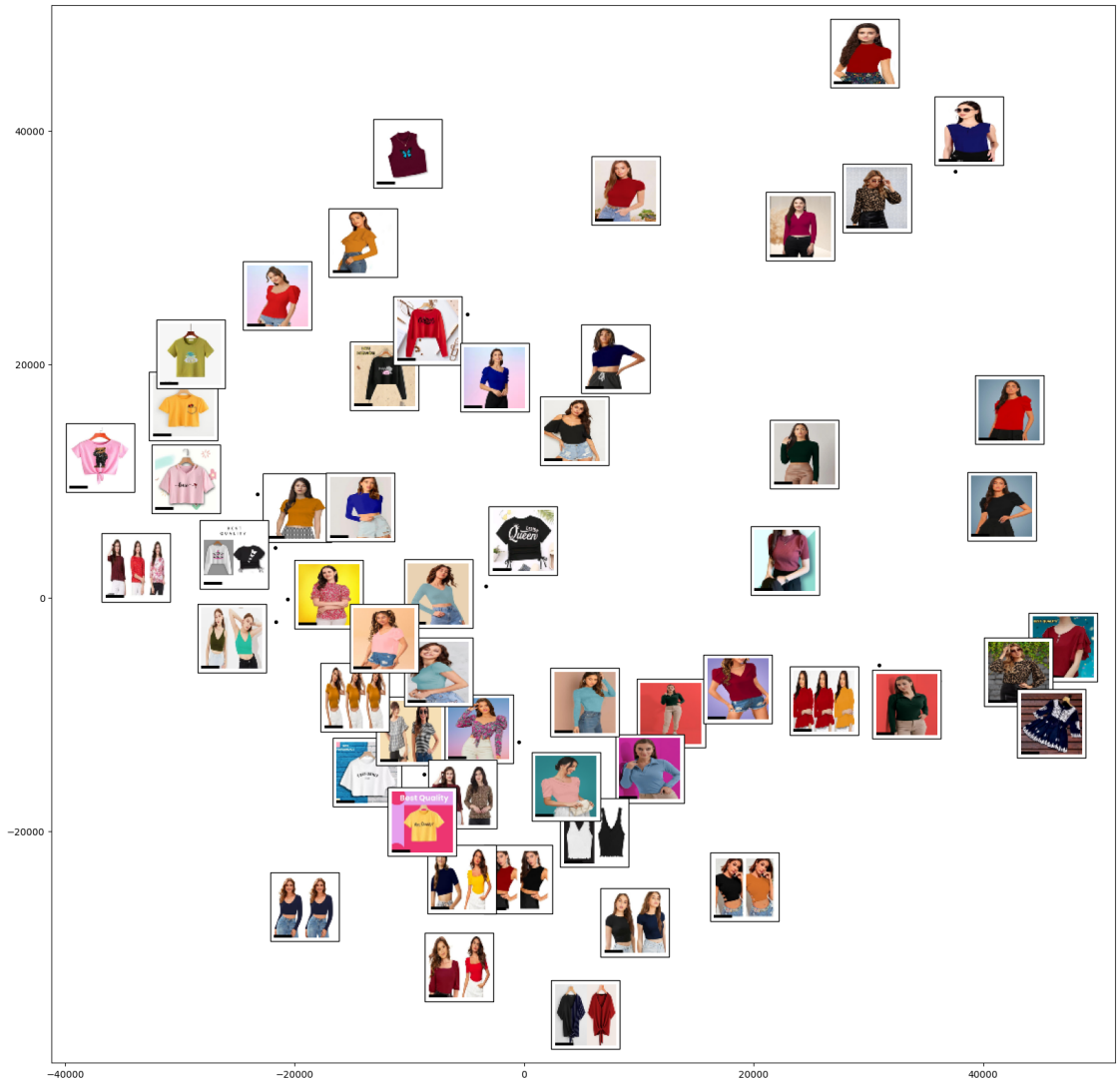
Projection for womenTBasket1



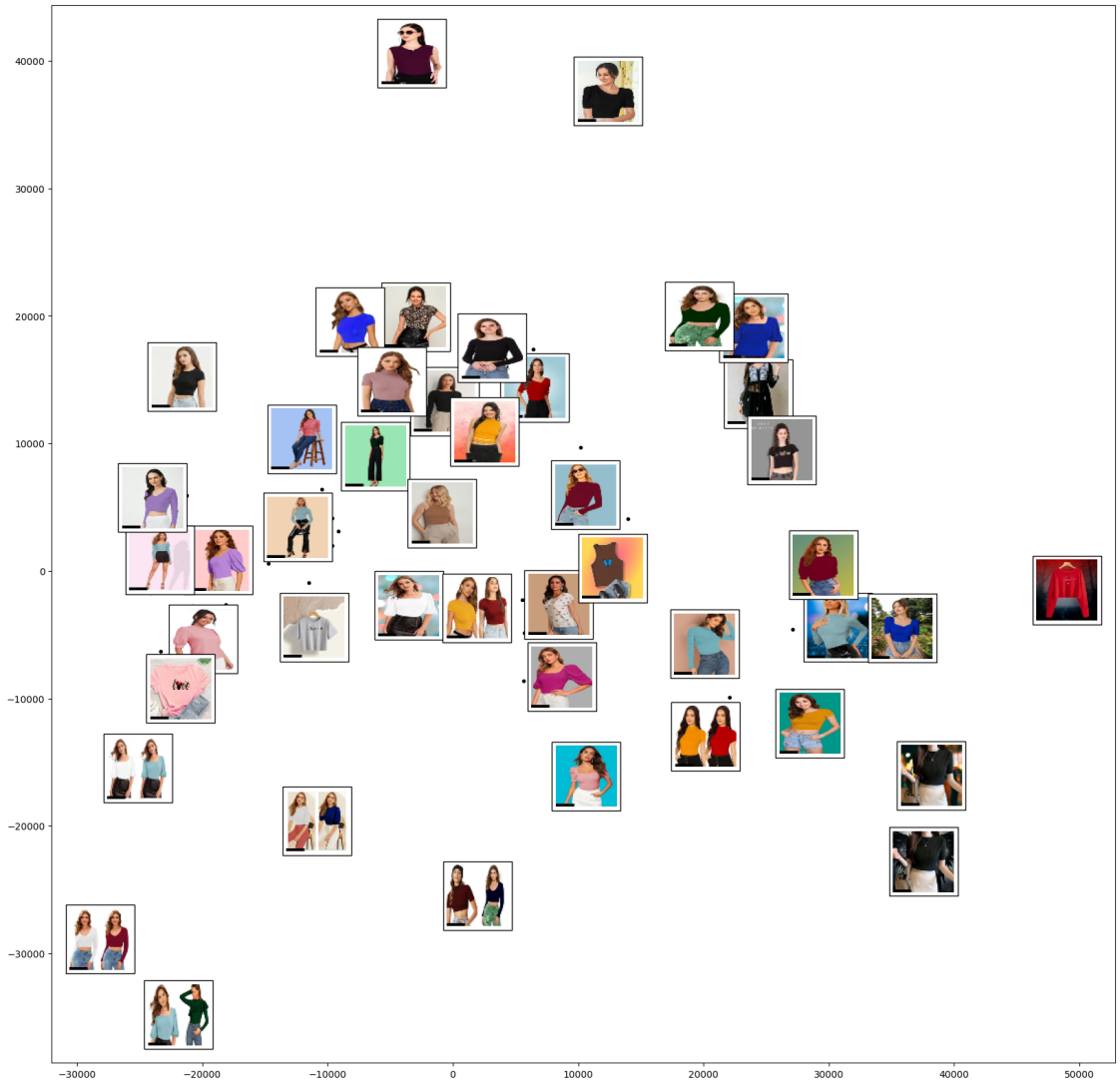
Projection for womenTBasket2



Projection for topsNTunicsBasket1



Projection for topsNTunicsBasket2

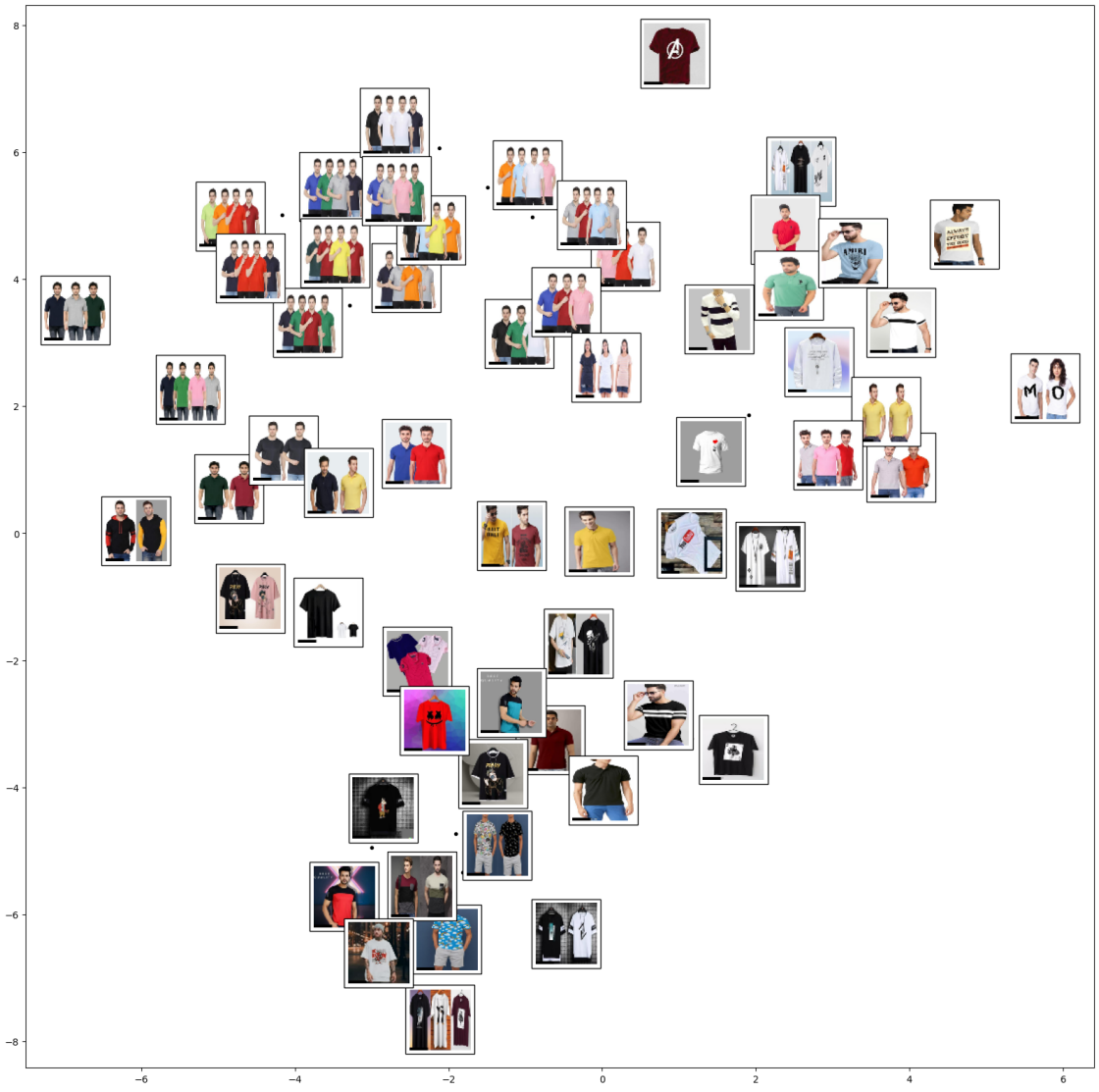


2.0.2 t-SNE

[25]:
```
from sklearn.manifold import TSNE

baskets = ['menTBasket1', 'menTBasket2', 'sareeBasket1', 'sareeBasket2',
           'kurtiBasket1', 'kurtiBasket2', 'womenTBasket1', 'womenTBasket2',
           'topsNTunicsBasket1', 'topsNTunicsBasket2']

for basket in baskets:
    data = []
    for id in globals()[basket]['id']:
        img = Image.open('visual-taxonomy/train_images/' + str(id).zfill(6) + '.
 ↪jpg').resize((128, 128))
        pixels = np.array(img)
```

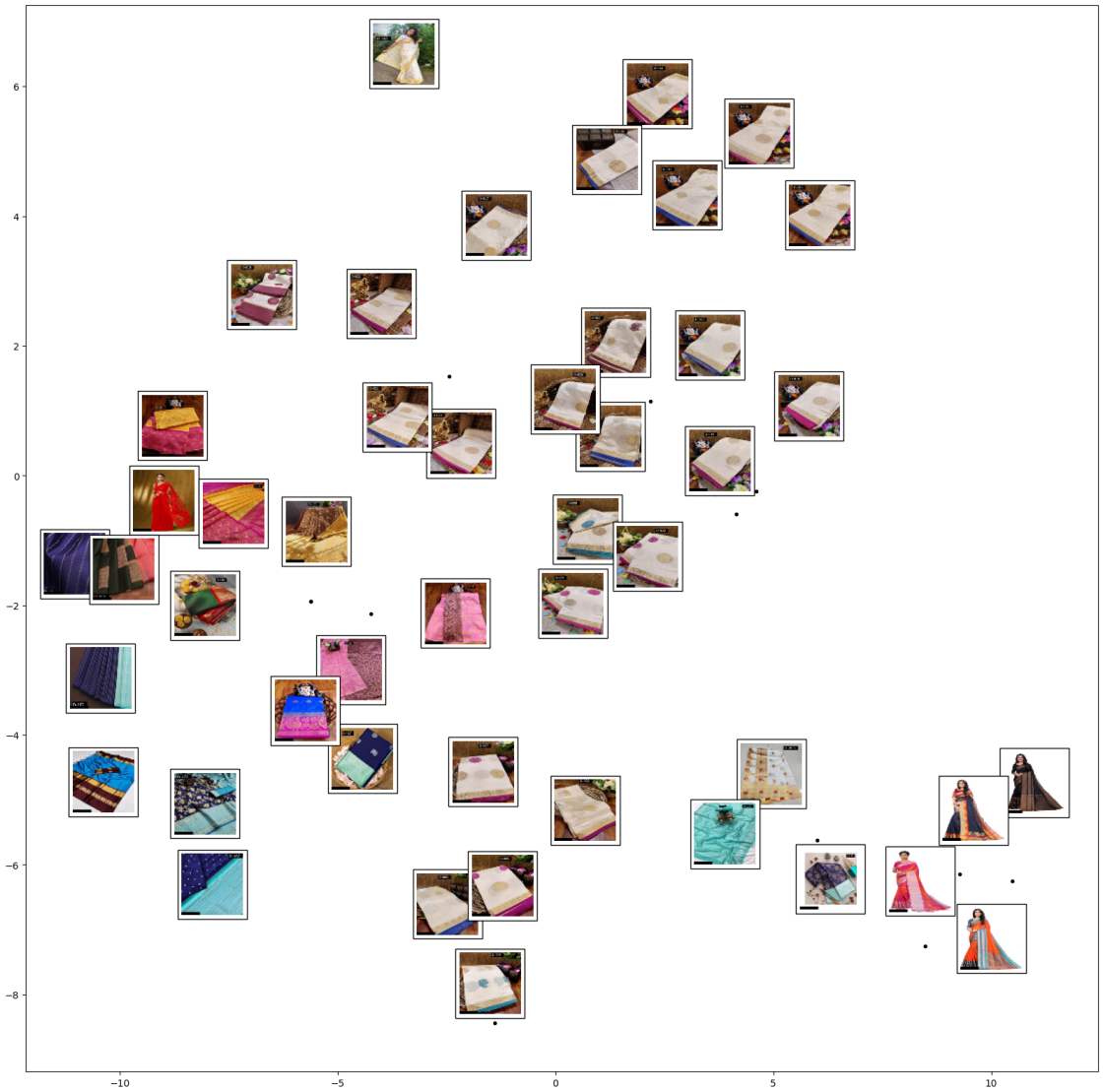
```
        data.append(pixels)
    data = np.array(data)
    num_images, height, width, channels = data.shape
    data_reshaped = data.reshape(num_images, height * width * channels)

    model = TSNE(n_components=2)
    proj = model.fit_transform(data_reshaped)
    downsampled_images = data[:, ::2, ::2, :]

    fig, ax = plt.subplots(figsize=(20, 20))
    plot_components(data_reshaped, model=model, images=downsampled_images,␣
↪ax=ax)

    print(f"Projection for {basket}")
    plt.show()
```

```
Projection for menTBasket1
```

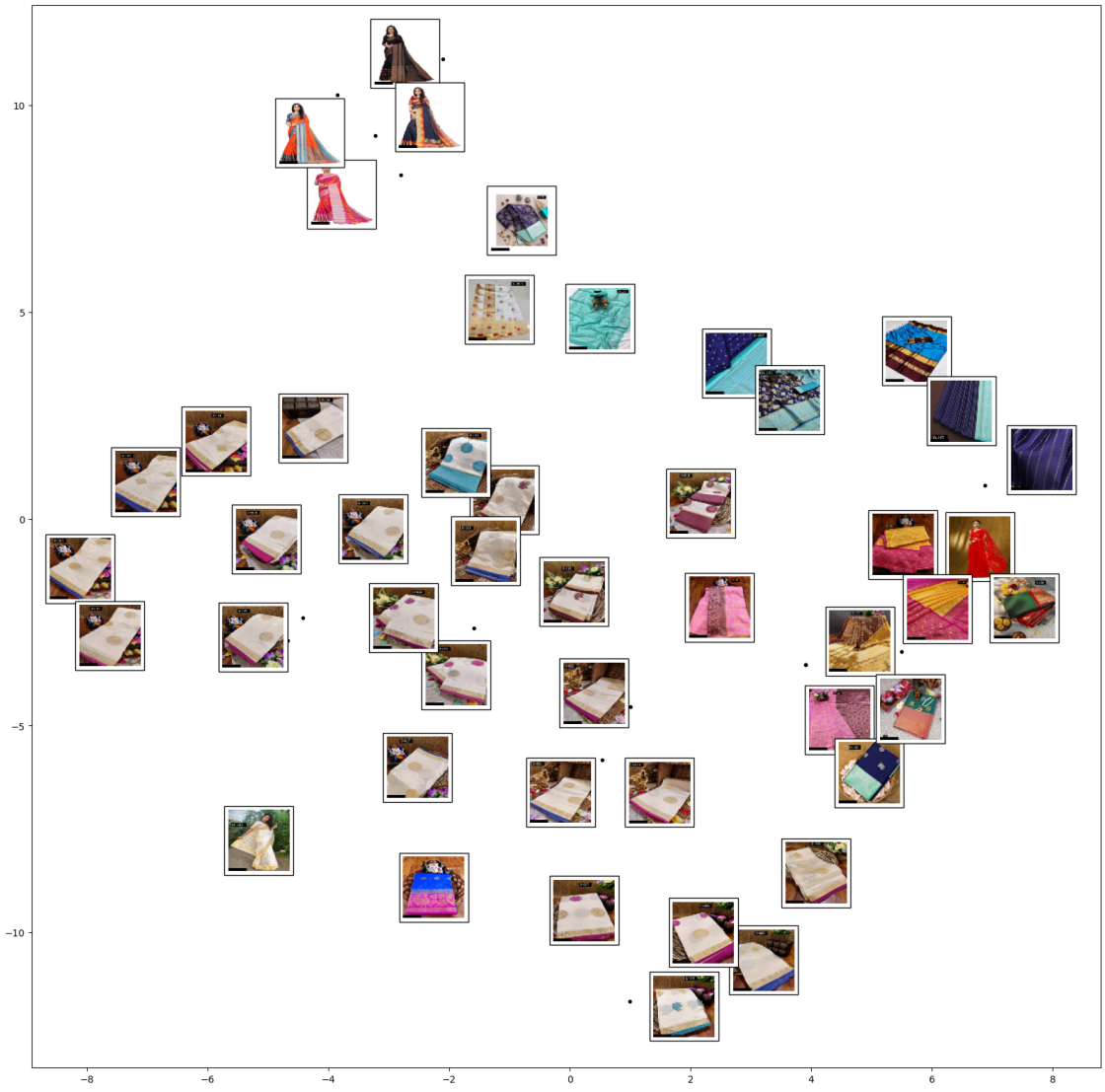


Projection for menTBasket2

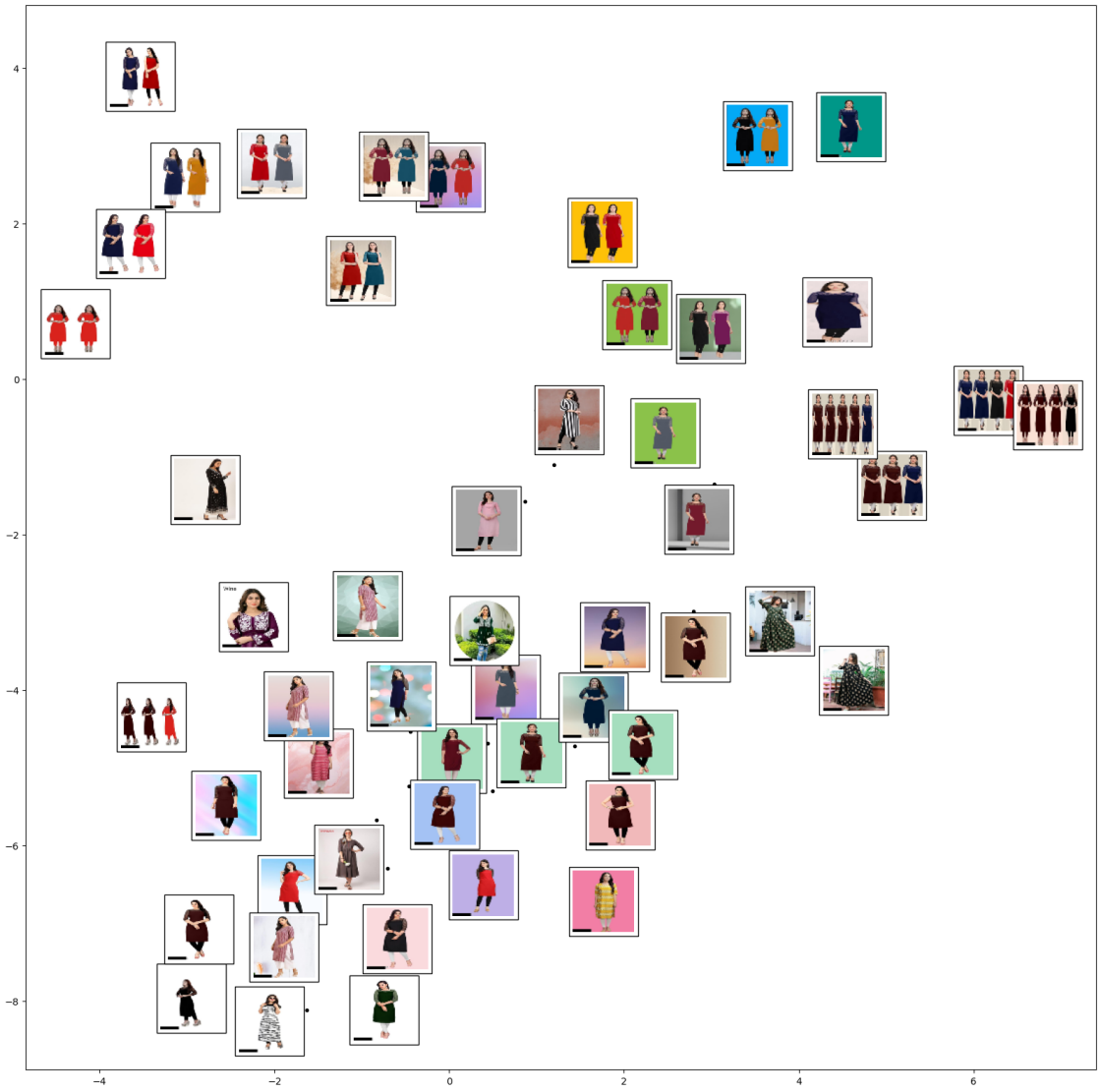
Projection for sareeBasket1



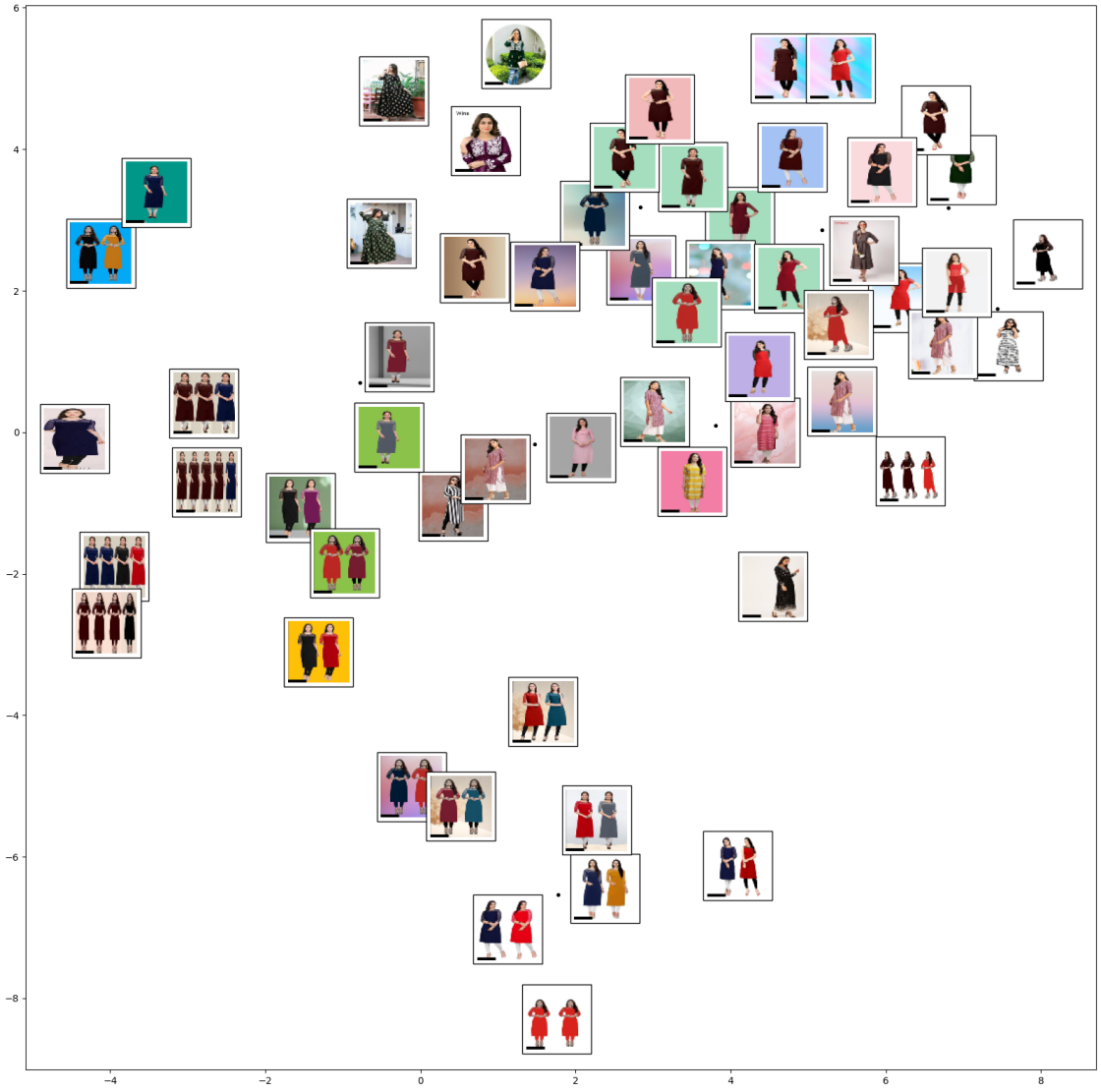
Projection for sareeBasket2



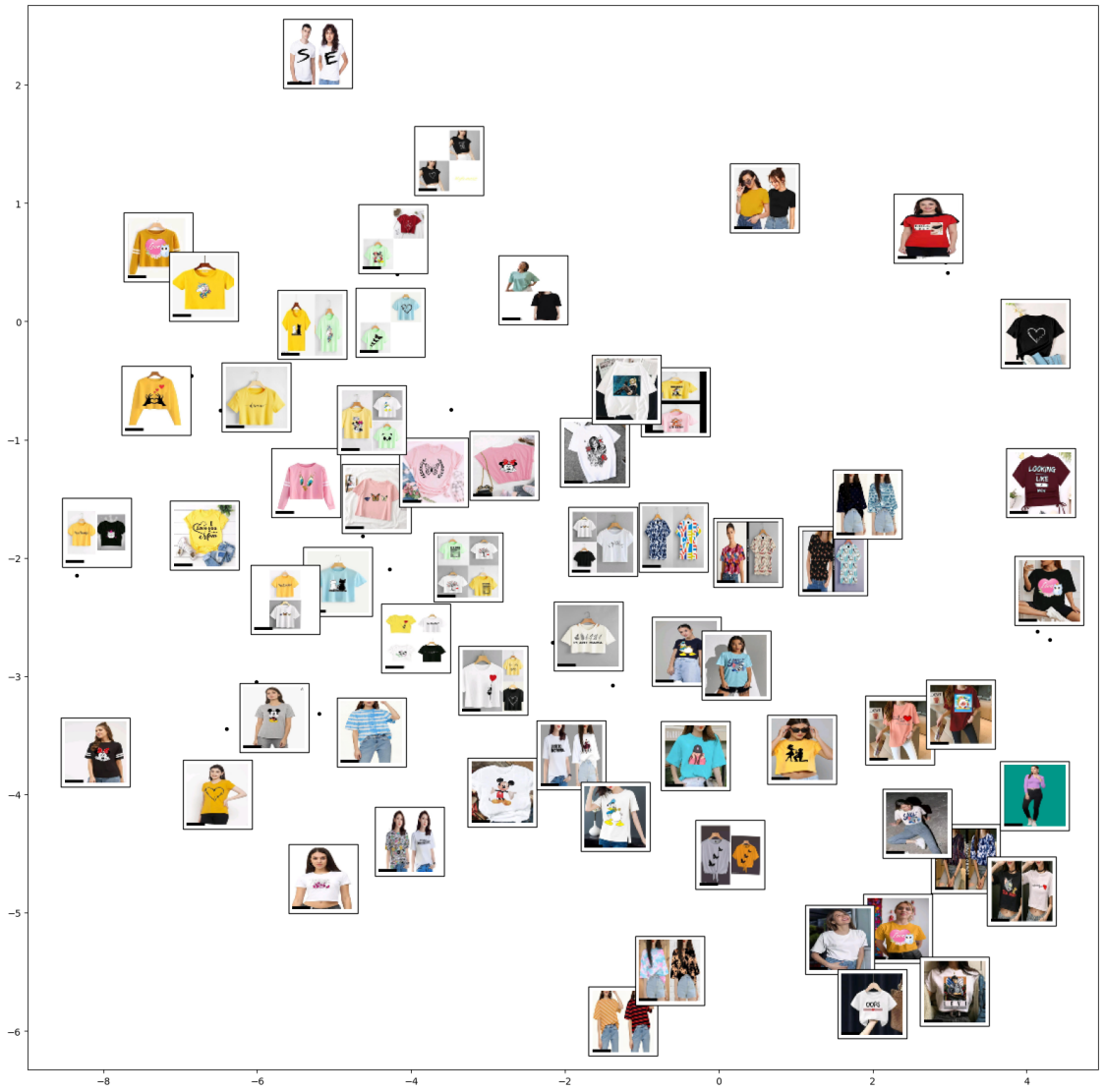
Projection for kurtiBasket1



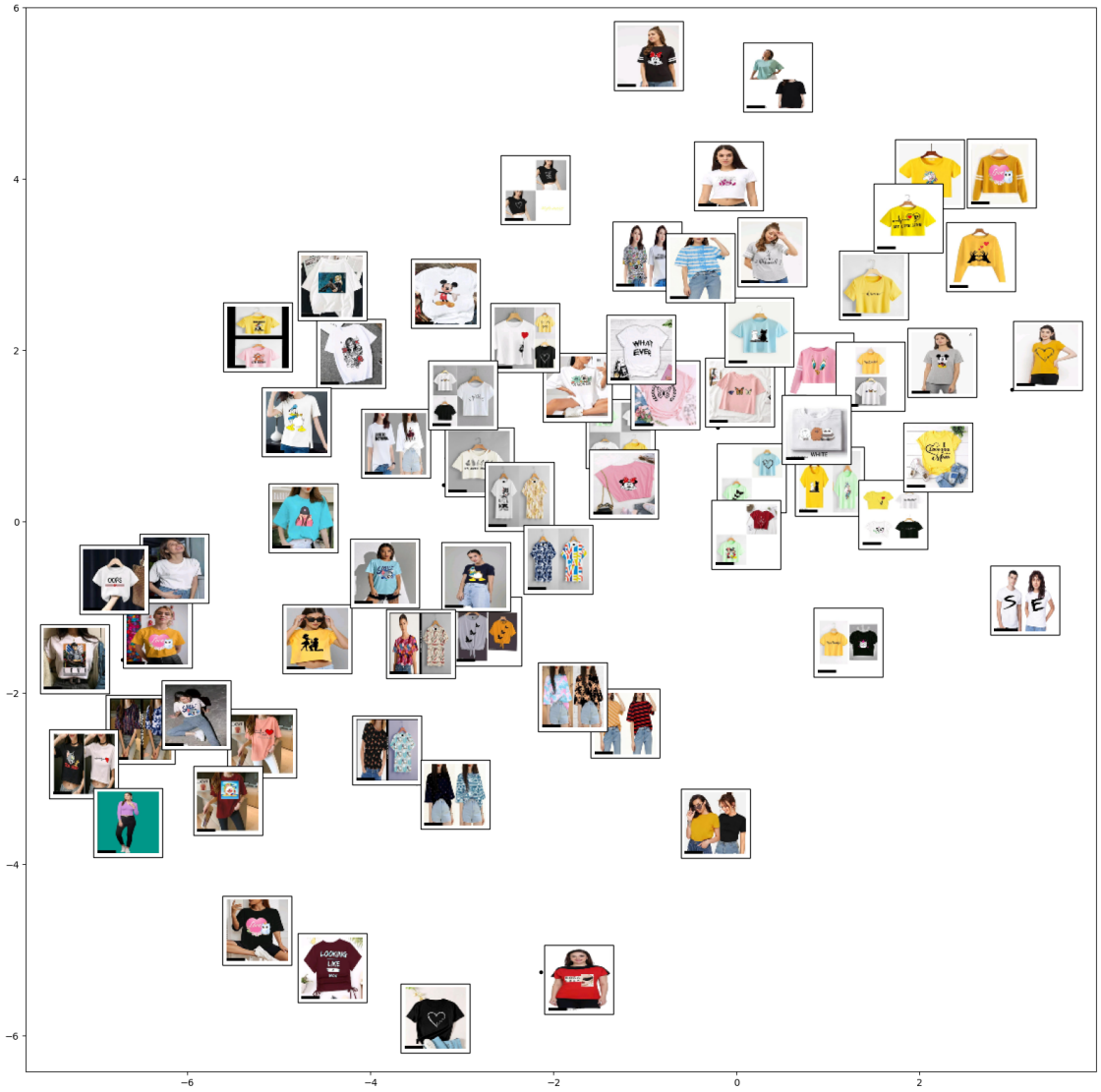
Projection for kurtiBasket2



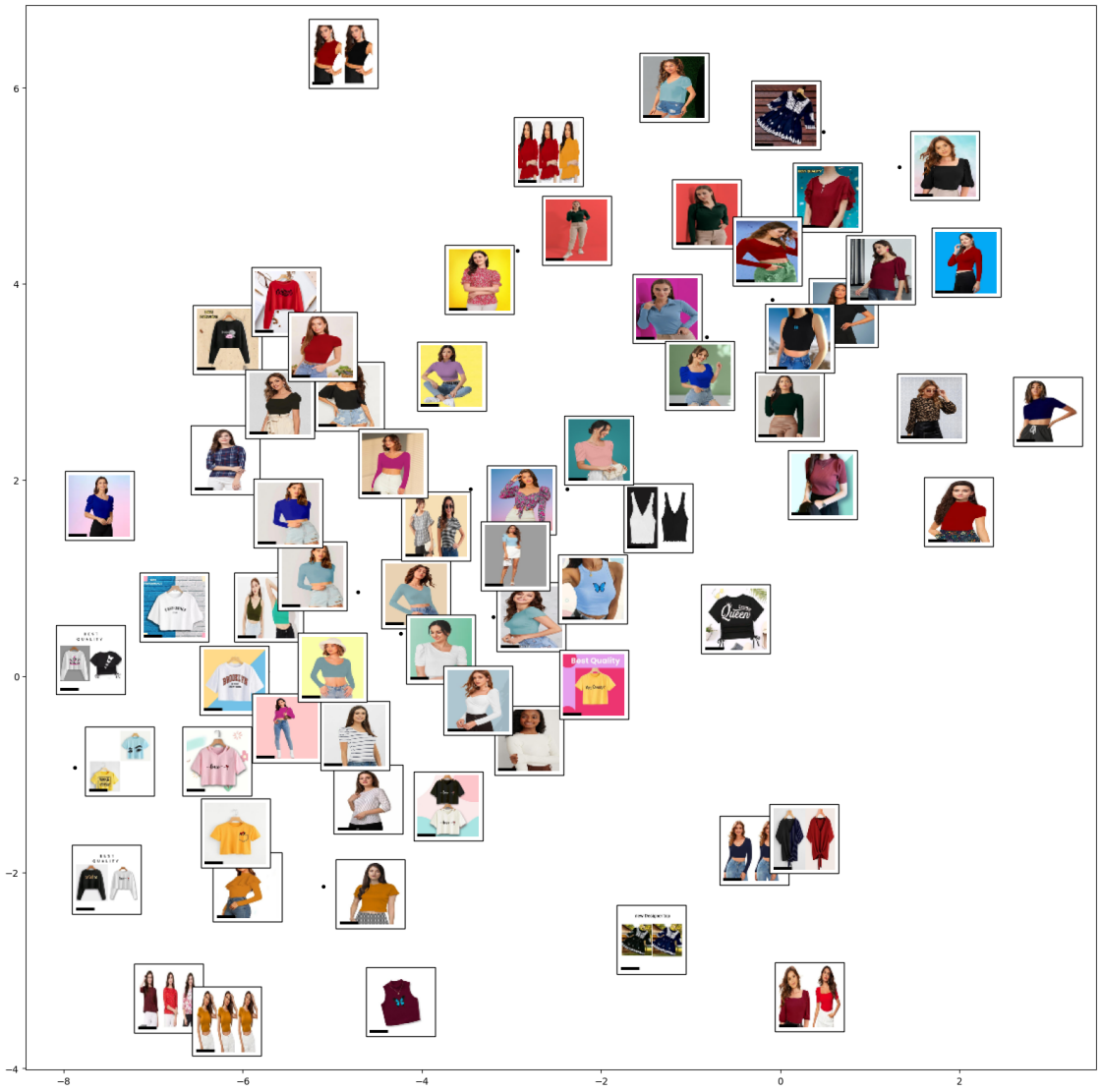
Projection for womenTBasket1



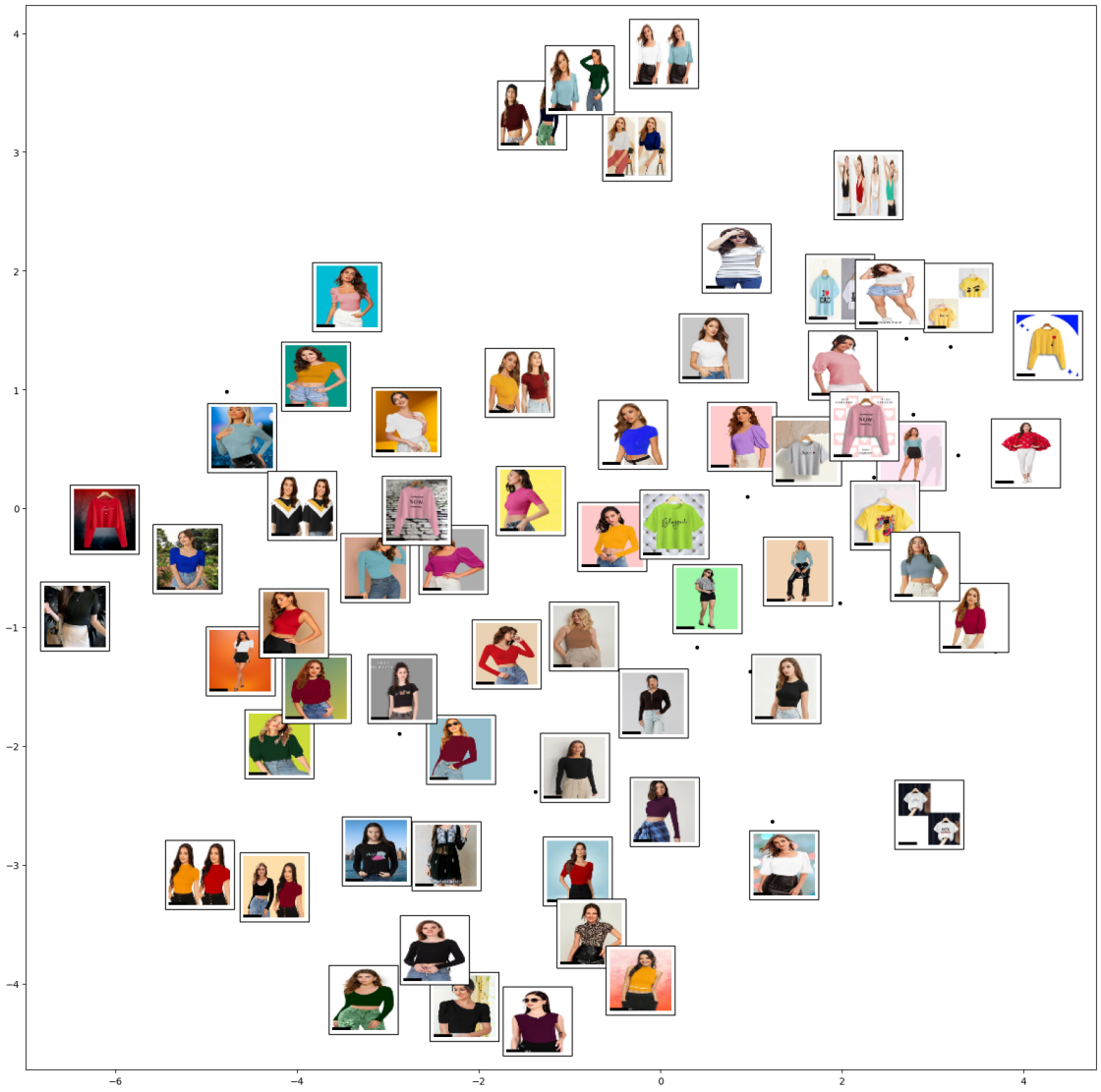
Projection for womenTBasket2



Projection for topsNTunicsBasket1



Projection for topsNTunicsBasket2



3 Task 3

3.1 ISOMAP

3.1.1 menTBasket1

X - Axis = Number of people (More people on the left to less on the right)

Y - Axis = Color (Light on the top to dark shades on the bottom)

3.1.2 menTBasket2

X - Axis = White background on the left vs Dark Background on the right

Y - Axis = Color (Light on the top to dark shades on the bottom)

3.1.3 kurtiBasket1

X - Axis = Number of people (Two people on the right side vs one or three people on the left side)

Y - Axis = Sleeve (Long sleeve on the top vs short sleeve on the bottom)

3.1.4 kurtiBasket2

X - Axis = Number of people (Two people on the right side vs one or three people on the left side)

Y - Axis = Sleeve (Long sleeve on the top vs short sleeve on the bottom)

3.1.5 sareeBasket1

X - Axis = People (Only saree on the left side vs Saree worn by models on the right)

Y - Axis = Color of the saree (white on the top to other colors on the bottom)

3.1.6 sareeTBasket2

X - Axis = Background color (white on the right vs natural backgrounds on the left regions)

Y - Axis = Color of the saree (white on the top to other colors on the bottom)

3.1.7 womenTBasket1

X - Axis = color (lighter colors on the left vs darker on the right)

Y - Axis = Number of people (single dress on the top sides vs multiple dresses on the bottom)

3.1.8 womenTBasket2

X - Axis = color (lighter colors on the left vs darker on the right)

Y - Axis = Number of people (single dress on the top sides vs multiple dresses on the bottom)

3.1.9 topsNTunicsBasket1

X - Axis = color/brightness (Lighter on the left and darker on the right)

Y - Axis = Number of people (single dress on the top sides vs multiple dresses on the bottom)

3.1.10 topsNTunicsBasket2

X - Axis = Background (Studio Solid plain backgrounds on the left vs Natural outdoor background on the right)

Y - Axis = Number of people (single dress on the top sides vs multiple dresses on the bottom)

3.2 TSNE

3.2.1 menTBasket1

X - Axis = Shirt type (Polos vs Round neck)

Y - Axis = Number of people (many on the top v less on the bottom)

3.2.2 menTBasket2

X - Axis = Color of the dress (Dark shades on the left to milder tones on the right)

Y - Axis = Shirt style (plain in the bottom vs printed/styled on the top)

3.2.3 kurtiBasket1

X - Axis = Background color (Colorful background on the center whereas plain background in the surrounding)

Y - Axis = Number of people present (more on the top and less on the bottom)

3.2.4 kurtiBasket2

X - Axis = Scaling of the image (In right the scaling is correct wheras in left the picture is scaled to fit the size)

Y - Axis = Number of people (more on the bottom and less on the top)

3.2.5 sareeBasket1

X - Axis = Variation in the color of saree

Y - Axis = Placement of saree (left facing on the top to right facing in the bottom)

3.2.6 sareeTBasket2

X - Axis = Color of the saree (white on the left to colorful toward the right)

Y - Axis = People wearing sarees vs only sarees on the bottom

3.2.7 womenTBasket1

X - Axis = Color of dress (Light color on the left to dark color on the right)

Y - Axis = Background color (Multicolor on the left to White background on the right)

3.2.8 womenTBasket2

X - Axis = Quality of image (Less quality images to clearer image on the right)

Y - Axis = Color of dress (Light color on the left to dark color on the right)

3.2.9 topsNTunicsBasket1

X - Axis = Color of the dress (varying from light to dark shades)

Y - Axis = Brightness (Color background to White background)

3.2.10 topsNTunicsBasket2

X - Axis = Long sleeves to short sleeves

Y - Axis = Color of the dress(Dark to light)

4 Observations

1. We can see that t-SNE shows different visualisations for different baskets of the same category. This is because t-SNE by principle focuses on preserving local similarities, making it more sensitive to specific attribute changes and resulting in more variation across visualizations.
2. In contrast, The visualizations when using isomap shows very less/ no difference. This might be happening due to the fact that it preserves global geometric structure and optimizes for preserving pairwise distances within a low-dimensional space. So even with different attribute sets, it may still produce similar global layouts if the data's underlying structure doesn't change significantly