da24c026

October 30, 2024

DA24C026 - Assignment 9

[2]:
```
import numpy as np
import pandas as pd
import random
import os
from PIL import Image
from sklearn.decomposition import PCA
from sklearn.manifold import Isomap, TSNE
import matplotlib.pyplot as plt
from matplotlib import offsetbox
```

[3]:
```
data = pd.read_csv('/kaggle/input/visual-taxonomy/train.csv')
```

[4]:
```
data.head()
```

[4]:

	id	Category	len	attr_1	attr_2	attr_3	attr_4	attr_5	attr_6	attr_7	attr_8	attr_9	attr_10
0	0	Men Tshirts	5	default	round	printed	default	short sleeves	NaN	NaN	NaN	NaN	NaN
1	1	Men Tshirts	5	multicolor	polo	solid	solid	short sleeves	NaN	NaN	NaN	NaN	NaN
2	2	Men Tshirts	5	default	polo	solid	solid	short sleeves	NaN	NaN	NaN	NaN	NaN
3	3	Men Tshirts	5	multicolor	polo	solid	solid	short sleeves	NaN	NaN	NaN	NaN	NaN
4	4	Men Tshirts	5	multicolor	polo	solid	solid	short sleeves	NaN	NaN	NaN	NaN	NaN

[143]:
```
data.describe(include = 'all')
```

[143]:

	id	Category	len	attr_1	attr_2
count	70213.000000	70213	70213.000000	51867	55192
unique	NaN	5	NaN	18	14
top	NaN	Women Tops & Tunics	NaN	default	regular
freq	NaN	19004	NaN	9268	17222
mean	35254.985872	NaN	8.850569	NaN	NaN

```
std     20295.174166                 NaN      1.559819      NaN      NaN
min         0.000000                 NaN      5.000000      NaN      NaN
25%     17718.000000                 NaN      8.000000      NaN      NaN
50%     35272.000000                 NaN     10.000000      NaN      NaN
75%     52825.000000                 NaN     10.000000      NaN      NaN
max     70378.000000                 NaN     10.000000      NaN      NaN

         attr_3   attr_4  attr_5         attr_6           attr_7      attr_8  \
count     54698    59888   56493          38116            41415       37474
unique       10       19      14              7               11          10
top     regular  printed  casual  short sleeves  regular sleeves  zari woven
freq      15546    14285   12358          13703            14661       13533
mean        NaN      NaN     NaN            NaN              NaN         NaN
std         NaN      NaN     NaN            NaN              NaN         NaN
min         NaN      NaN     NaN            NaN              NaN         NaN
25%         NaN      NaN     NaN            NaN              NaN         NaN
50%         NaN      NaN     NaN            NaN              NaN         NaN
75%         NaN      NaN     NaN            NaN              NaN         NaN
max         NaN      NaN     NaN            NaN              NaN         NaN

                attr_9 attr_10
count            33565   24999
unique              13       8
top     regular sleeves      no
freq              7480   14835
mean               NaN     NaN
std                NaN     NaN
min                NaN     NaN
25%                NaN     NaN
50%                NaN     NaN
75%                NaN     NaN
max                NaN     NaN
```

Task 1

[144]:
```
categories = ["Men Tshirts", "Sarees", "Kurtis", "Women Tshirts", "Women Tops &␣
 ↪Tunics"]
samples = {}
random.seed(41)
for category in categories:
    attributes = ["attr_1", "attr_2", "attr_3", "attr_4", "attr_5", "attr_6",␣
 ↪"attr_7", "attr_8", "attr_9", "attr_10"]
    i = 0
    while i<2:
        attr = random.choice(attributes)
        basket = data[(data['Category'] == category) & (data[attr].notna())]
        if len(basket) >= 100:
```

```
            samples[f'{category}_{attr}'] = basket['id'].sample(100).tolist()
            i = i+1
            attributes.remove(attr)
```

[145]:
```
print(samples.items())
```

```
dict_items([('Men Tshirts_attr_4', [301, 1649, 626, 860, 2075, 2562, 2156, 4265,
5461, 5777, 3807, 2437, 6559, 700, 739, 4066, 2556, 4375, 4931, 3896, 4906,
3322, 5421, 5714, 4740, 3913, 1131, 2983, 2580, 3225, 1471, 2550, 1497, 624,
805, 3854, 3969, 6721, 1888, 990, 6463, 3274, 4377, 243, 1655, 6893, 3825, 2114,
2289, 2092, 778, 6369, 4928, 239, 2509, 1173, 668, 96, 5154, 4666, 6179, 5170,
4490, 1125, 533, 3714, 919, 956, 2112, 1287, 238, 1084, 555, 4604, 2104, 3945,
158, 6034, 5978, 5542, 2364, 5116, 2048, 3261, 5965, 2280, 1221, 6906, 857,
1339, 4318, 4825, 460, 5122, 2994, 4785, 2678, 4978, 8, 4575]), ('Men
Tshirts_attr_3', [4937, 481, 5599, 4504, 2499, 1849, 4770, 5524, 1678, 932,
5690, 2392, 2166, 6264, 2856, 1318, 3886, 7344, 3359, 2493, 6097, 2743, 635,
2127, 1594, 4781, 5479, 4102, 516, 4422, 7231, 34, 626, 1634, 6029, 3185, 6438,
6794, 3984, 1942, 542, 180, 2866, 1644, 4296, 2334, 1923, 4555, 2750, 4077,
6171, 3201, 866, 1164, 4431, 4388, 2855, 2749, 1577, 5023, 4346, 4008, 3104,
5704, 1776, 1002, 3304, 4900, 2138, 4854, 3682, 6043, 5181, 2101, 2300, 4434,
4999, 2984, 1787, 381, 6596, 3421, 4876, 2945, 2428, 1235, 3757, 6230, 3755,
4627, 4538, 5183, 4107, 5880, 2981, 3890, 671, 2547, 5121, 5161]),
('Sarees_attr_7', [25082, 7574, 25620, 10814, 16086, 9725, 17751, 8847, 20329,
11546, 24675, 14294, 22208, 16596, 23555, 19835, 18460, 17525, 10416, 12572,
15375, 16617, 17447, 19723, 17079, 18940, 8085, 22672, 15717, 14008, 15917,
11417, 21660, 24412, 14533, 17849, 17277, 14439, 7787, 20615, 17008, 8032,
17263, 15666, 15454, 17544, 25040, 20754, 22298, 22692, 11209, 20696, 12635,
18603, 8513, 13212, 13829, 12586, 13605, 20999, 10626, 17050, 21164, 19009,
12929, 11443, 18479, 21982, 12202, 10786, 20522, 11665, 21538, 21170, 21953,
21192, 11254, 10962, 19703, 23113, 8261, 20975, 15816, 11844, 11565, 20684,
16987, 25456, 8191, 14368, 11567, 13903, 14084, 8634, 9468, 11629, 22537, 24782,
18903, 20302]), ('Sarees_attr_5', [25375, 10333, 21232, 24940, 16677, 12128,
11725, 14834, 11707, 20395, 10414, 18598, 24985, 23439, 18275, 20196, 24284,
20423, 21465, 10307, 23654, 22434, 16868, 24122, 24246, 15337, 15606, 25651,
9307, 18042, 10835, 18941, 20684, 20984, 16001, 14624, 9932, 8351, 19670, 13928,
25155, 22534, 13683, 20963, 14489, 15934, 25707, 18110, 13969, 16104, 14972,
23841, 21762, 12672, 11730, 20804, 24888, 8715, 17607, 25759, 22423, 19536,
22672, 20356, 17190, 13758, 11211, 8396, 23208, 11565, 18834, 15409, 20283,
16305, 9085, 13607, 19264, 7552, 22049, 8571, 21524, 18827, 24620, 13120, 16128,
10110, 17162, 13876, 15415, 14998, 13245, 8995, 9986, 17395, 13118, 18587,
25469, 14319, 12151, 12681]), ('Kurtis_attr_9', [27180, 30133, 32579, 28787,
30699, 29539, 32168, 29229, 29654, 31898, 27533, 26709, 29886, 29113, 30522,
28776, 29785, 30914, 29939, 27378, 32408, 26213, 28276, 29919, 27122, 31849,
28703, 30796, 25940, 29209, 28143, 31468, 32004, 29588, 30712, 29139, 29052,
30883, 31490, 26442, 29786, 28626, 26259, 31758, 26428, 29046, 30671, 28432,
30722, 29506, 32330, 31478, 26832, 26665, 31472, 28504, 29394, 28354, 30627,
31279, 29527, 30218, 31632, 32472, 28209, 30840, 30496, 28374, 31044, 27763,
```

29488, 32599, 28684, 26470, 31755, 28627, 26190, 27098, 30697, 29689, 26624,
25885, 27274, 26335, 27613, 30128, 27165, 29271, 30426, 26662, 27251, 32425,
27563, 32028, 27788, 26773, 26481, 32186, 29124, 29544]), ('Kurtis_attr_5',
[27803, 27165, 31606, 27583, 26072, 27719, 28678, 26873, 29517, 29051, 29463,
27037, 27336, 26729, 29043, 29034, 29061, 25916, 30468, 27654, 26343, 30867,
26730, 30292, 26931, 30718, 27365, 26699, 25978, 27251, 28805, 26160, 27163,
26601, 27098, 29634, 29402, 26335, 26187, 28068, 25901, 31356, 26932, 26899,
27032, 29288, 26977, 27491, 27485, 29314, 26920, 28561, 26804, 28507, 26340,
29548, 28544, 26686, 26659, 28980, 28691, 26324, 27006, 29728, 26646, 28361,
26812, 28636, 26458, 27680, 27015, 27225, 27707, 29286, 25885, 28357, 28869,
28608, 25818, 27095, 27533, 27119, 31154, 27302, 26182, 29218, 27207, 27712,
26987, 28722, 26978, 28822, 27070, 30779, 31293, 28342, 26831, 28801, 29904,
26003]), ('Women Tshirts_attr_7', [33787, 46925, 41011, 43840, 34837, 32874,
41411, 38963, 36810, 40498, 39898, 45612, 46320, 33795, 49998, 46459, 38821,
43993, 42256, 33484, 50678, 40479, 38250, 50373, 40708, 43871, 41901, 43281,
34602, 40469, 33806, 36763, 37998, 33116, 49635, 35599, 43213, 39999, 49209,
38256, 41118, 44387, 38474, 40137, 44948, 39086, 41333, 34204, 36576, 34598,
45880, 33203, 50692, 39666, 42143, 33183, 44242, 49527, 45820, 35603, 47952,
43326, 46839, 43964, 41902, 49456, 49612, 35784, 41747, 36398, 46826, 38772,
44544, 35941, 47752, 39868, 46395, 41674, 50839, 49478, 38557, 39090, 39656,
43393, 49470, 33756, 38037, 36224, 49119, 37692, 38476, 38745, 42592, 35855,
38396, 44685, 37508, 44558, 43414, 37777]), ('Women Tshirts_attr_1', [32915,
44825, 32762, 38913, 40740, 50585, 49154, 37995, 43500, 36951, 45193, 34729,
33348, 50492, 40881, 42394, 48100, 50435, 33465, 40652, 45339, 42681, 37339,
43132, 32779, 37663, 49816, 38506, 51117, 37160, 47984, 36408, 48754, 46227,
38754, 41846, 40056, 42729, 47571, 37299, 39667, 34013, 34762, 48432, 35270,
41880, 39047, 37369, 32897, 47953, 40238, 39824, 45356, 36232, 46228, 46491,
47024, 42865, 40144, 44519, 44048, 35410, 34903, 37162, 40968, 40755, 35101,
50420, 46162, 51258, 33966, 42787, 37174, 38374, 49005, 41412, 33606, 46360,
43011, 40553, 38518, 35736, 39909, 48165, 38226, 38895, 50562, 42722, 36654,
47586, 39706, 48020, 46691, 36121, 39557, 35569, 43732, 34887, 41102, 45458]),
('Women Tops & Tunics_attr_4', [64635, 57093, 52851, 57329, 52997, 59200, 52503,
60284, 66473, 65243, 67405, 63917, 64014, 61762, 65811, 66243, 68994, 54747,
60529, 55434, 54035, 64089, 65456, 62748, 67984, 55826, 65257, 55765, 54534,
55803, 58801, 65026, 55767, 56880, 57083, 59876, 60432, 58907, 66712, 59869,
51417, 55561, 69971, 66639, 66483, 62340, 55609, 69183, 70345, 68729, 53270,
53971, 58468, 54033, 56961, 60615, 62092, 69804, 66787, 69729, 51608, 61640,
56277, 58052, 62124, 53110, 61314, 61909, 69159, 66286, 64018, 61034, 56953,
66229, 66072, 68392, 65578, 59439, 59564, 64396, 69883, 65742, 70280, 65080,
68507, 60461, 67549, 69603, 61692, 54521, 57433, 63618, 52113, 63295, 56401,
61293, 65599, 55073, 67518, 63165]), ('Women Tops & Tunics_attr_1', [64434,
61205, 61053, 54811, 57782, 59289, 66269, 52032, 69887, 66818, 63886, 61934,
67572, 62757, 54759, 59832, 57916, 54042, 65520, 66300, 53341, 59340, 52876,
62821, 67897, 58469, 61144, 55162, 64969, 65732, 69198, 58517, 54744, 65198,
65654, 54728, 55321, 66719, 54112, 56835, 65438, 57626, 58452, 63193, 57288,
69175, 54692, 55093, 51935, 51434, 64284, 59689, 59140, 55854, 63168, 57496,
69567, 61554, 62639, 63465, 67156, 54749, 68464, 63337, 53409, 60499, 65953,
65573, 56695, 61384, 64398, 60910, 56790, 63274, 61388, 61977, 69467, 56794,

```
57358, 65844, 65564, 57430, 56449, 51602, 59373, 55236, 53872, 53050, 62807,
51460, 66053, 56927, 59555, 65728, 66883, 63948, 56403, 56553, 64111, 65842])])
```

Created 10 baskets for all 5 different categories with random attributes.

Task 2 and Task 3

Plotting TSNE and Isomaps for each category - attribute pairs and identifying and interpreting pairs.

[146]:
```
# Helper function
def load_and_preprocess_images(ids):
    images = []
    actual_images=[]
    for product_id in ids:
        formatted_id = str(product_id).zfill(6)
        img_path = f"/kaggle/input/visual-taxonomy/train_images/{formatted_id}.
 ↪jpg"
        img = Image.open(img_path)
        img = img.resize((64, 64))
        images.append(np.array(img).flatten())
        actual_images.append(img)
    return np.array(images), actual_images
```

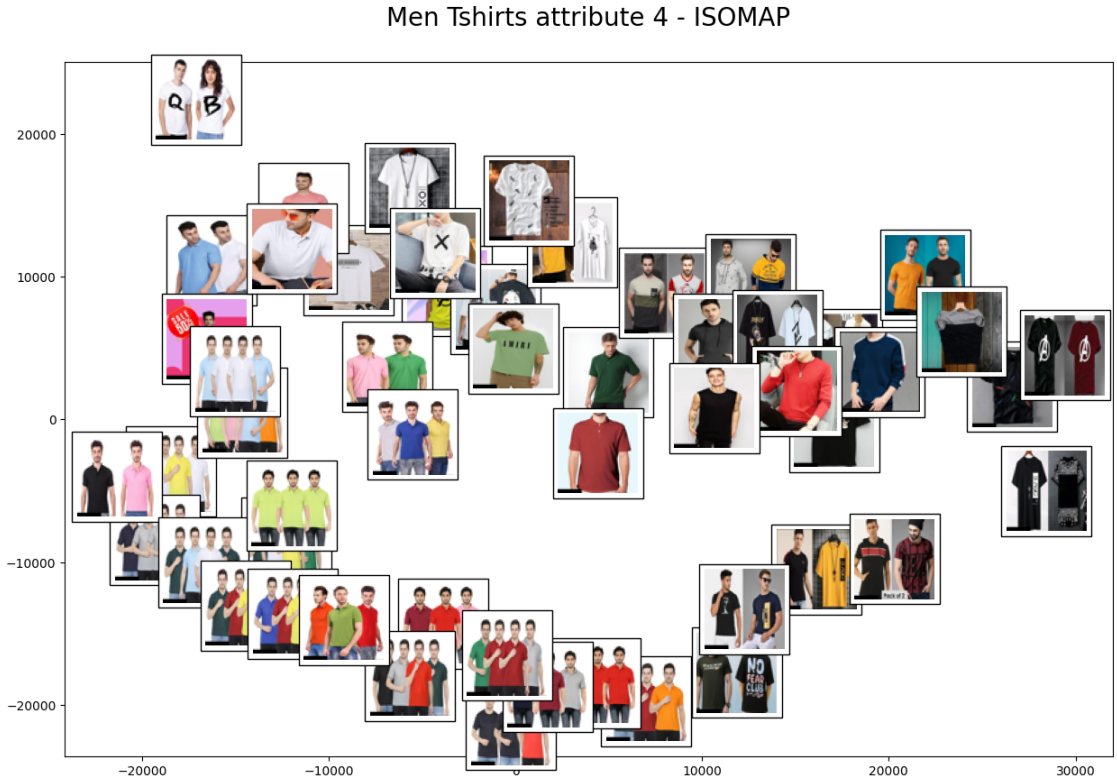
[147]:
```
# Helper function
def plot_components(data, model, images=None, ax=None,
                    thumb_frac=0.05, cmap='gray'):
    ax = ax or plt.gca()

    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:
                # don't show points that are too close
                continue
            shown_images = np.vstack([shown_images, proj[i]])
            imagebox = offsetbox.AnnotationBbox(
                offsetbox.OffsetImage(images[i], cmap=cmap),
                                      proj[i])
            ax.add_artist(imagebox)
```

[148]:
```
# Projection Models initialization
isomap = Isomap(n_neighbors=5, n_components=2)
tsne = TSNE(n_components=2, random_state=42)
```

a.) **Men Tshirts**

[149]:
```
images, actual_images = load_and_preprocess_images(samples['Men␣
↪Tshirts_attr_4'])
```

[150]:
```
plot_components(images,isomap, actual_images)
plt.title("Men Tshirts attribute 4 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



Results - Interpretation

1. X-axis: Component name - Color Intensity & Formality

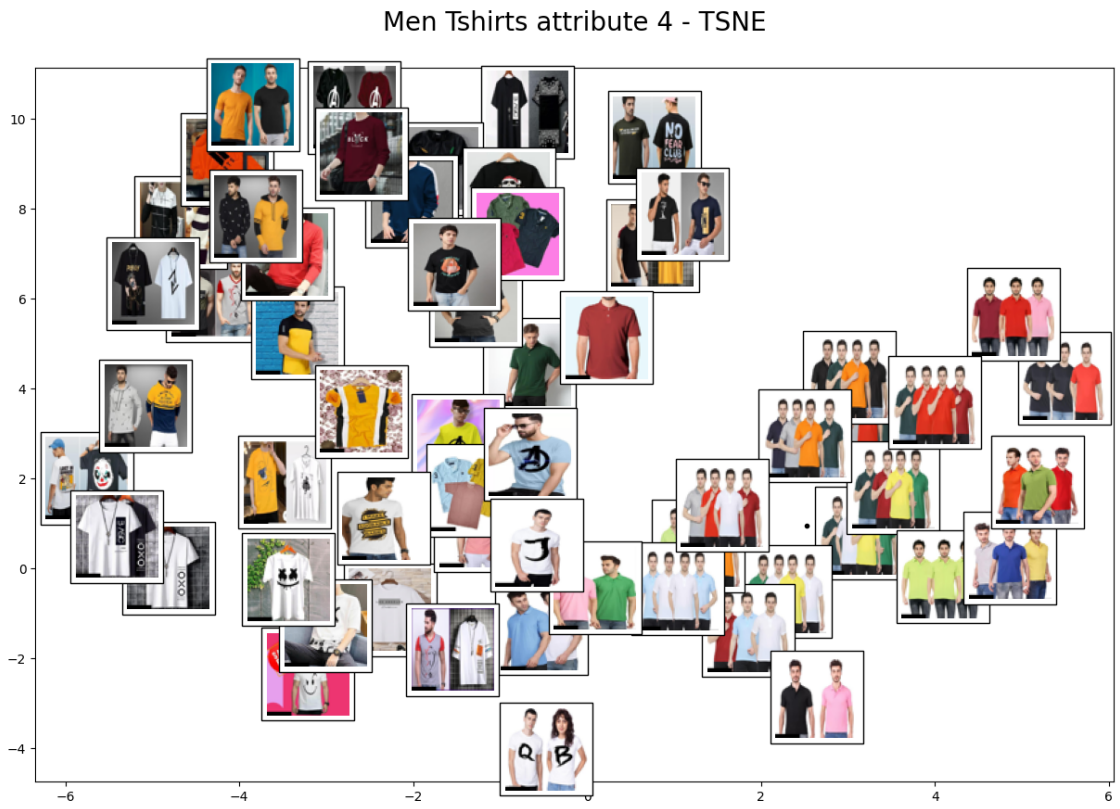
T-shirts on the left side of the plot appear to have more solid colors, often in neutral or pastel shades, typically associated with a classic or formal look. Moving towards the right, we see more vibrant colors and graphic designs, which are usually associated with casual or trendy attire.

2. Y-axis: Component name - Design Complexity

The bottom half of the plot mostly includes simple T-shirts, which may lack intricate designs or have minimal text, suggesting a more minimalist aesthetic. The upper half, however, features shirts with graphic prints, brand logos, or layered designs.

[151]:
```
plot_components(images,tsne, actual_images)
plt.title("Men Tshirts attribute 4 - TSNE\n", size = 20)
```

```
plt.gcf().set_size_inches(15, 10)
```



Results - Interpretation

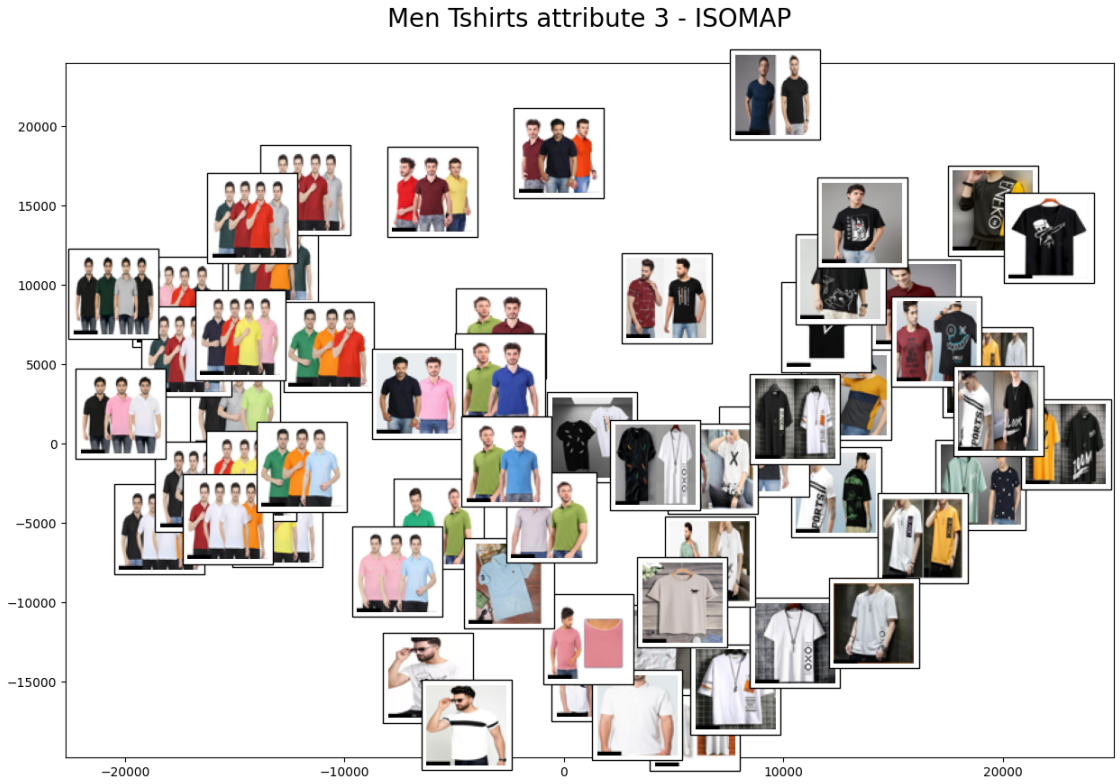
Similar to ISOMAP, in TSNE projection also two key dimensions emerge:

1.) Formality Spectrum (X-axis): Ranges from basic, solid-colored T-shirts (formal/classic) to vibrant, graphic-heavy designs (casual/trendy).

2.) Design Complexity (Y-axis): Moves from minimalist shirts to those with intricate graphics, logos, or branding.

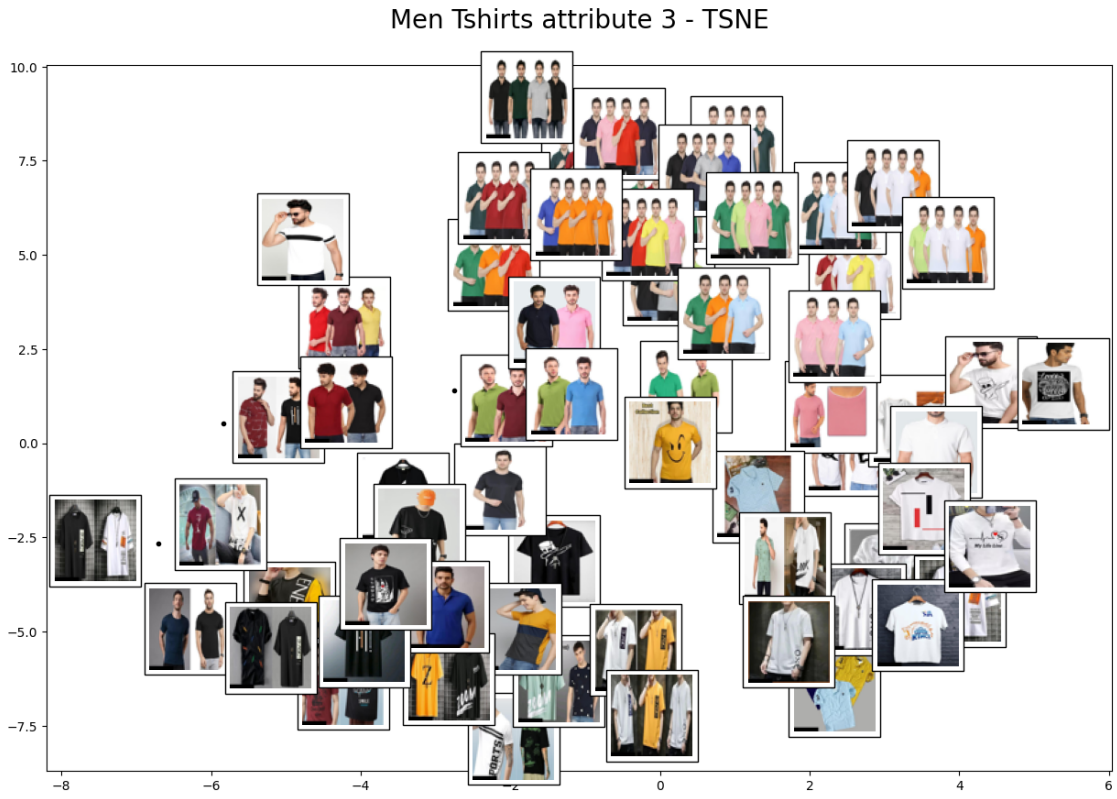
[152]:
```
images, actual_images = load_and_preprocess_images(samples['Men␣
↪Tshirts_attr_3'])
```

[154]:
```
plot_components(images,isomap, actual_images)
plt.title("Men Tshirts attribute 3 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



X - axis -> Number of Tshirts in decreasing order y - axis -> Color shade (light to dark)

[155]:
```
plot_components(images,tsne, actual_images)
plt.title("Men Tshirts attribute 3 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



X - axis -> color shade (dark to light)

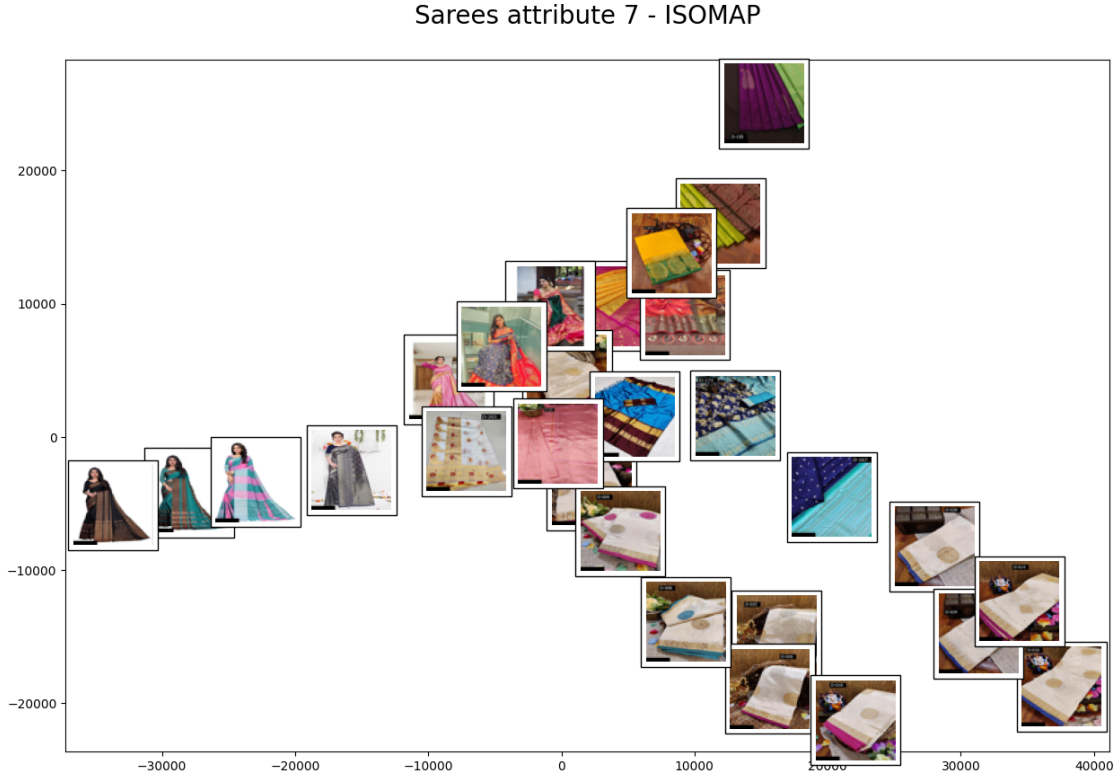
y - axis -> Design complexity (complex to simple)

b.) **SAREES**

[157]:
```
images, actual_images = load_and_preprocess_images(samples['Sarees_attr_7'])
```

[158]:
```
plot_components(images,isomap, actual_images)
plt.title("Sarees attribute 7 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```

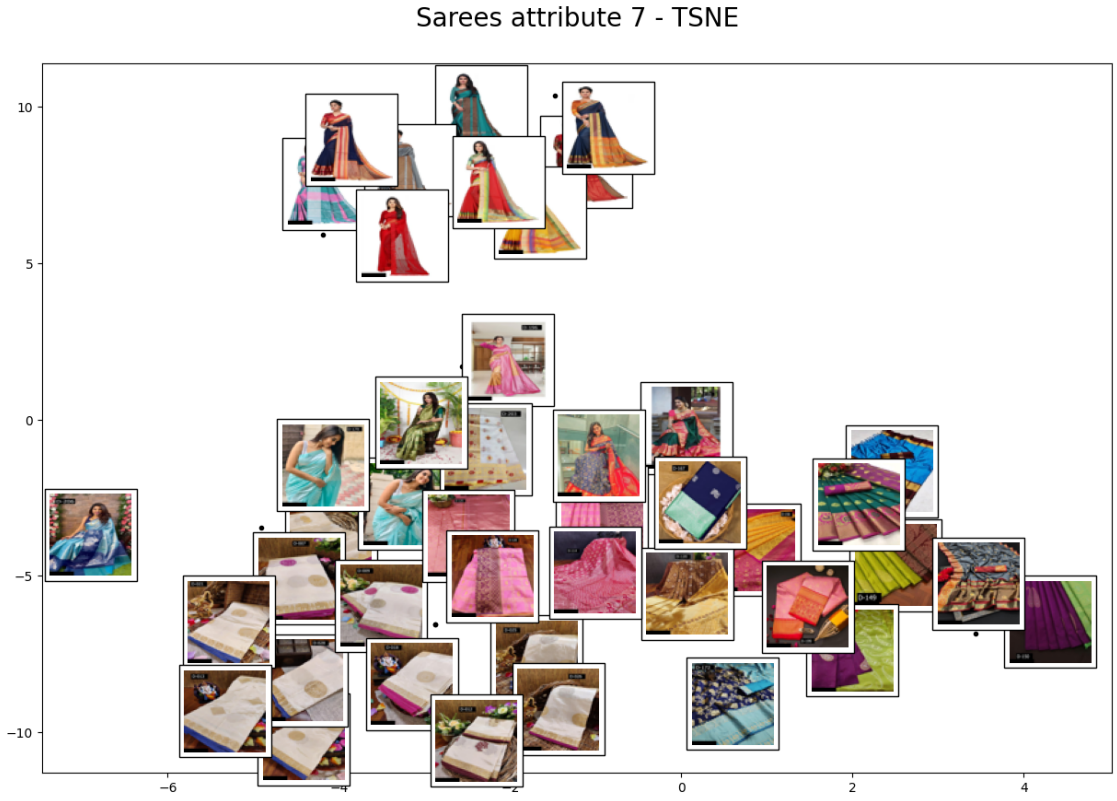
```
/opt/conda/lib/python3.10/site-packages/sklearn/manifold/_isomap.py:373:
UserWarning: The number of connected components of the neighbors graph is 2 > 1.
Completing the graph to fit Isomap might be slow. Increase the number of
neighbors to avoid this issue.
  self._fit_transform(X)
/opt/conda/lib/python3.10/site-packages/scipy/sparse/_index.py:108:
SparseEfficiencyWarning: Changing the sparsity structure of a csr_matrix is
expensive. lil and dok are more efficient.
  self._set_intXint(row, col, x.flat[0])
```



x - axis -> design styles (angle of lines)

y - axis -> color shade (light to dark)

[159]:
```
plot_components(images,tsne, actual_images)
plt.title("Sarees attribute 7 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```

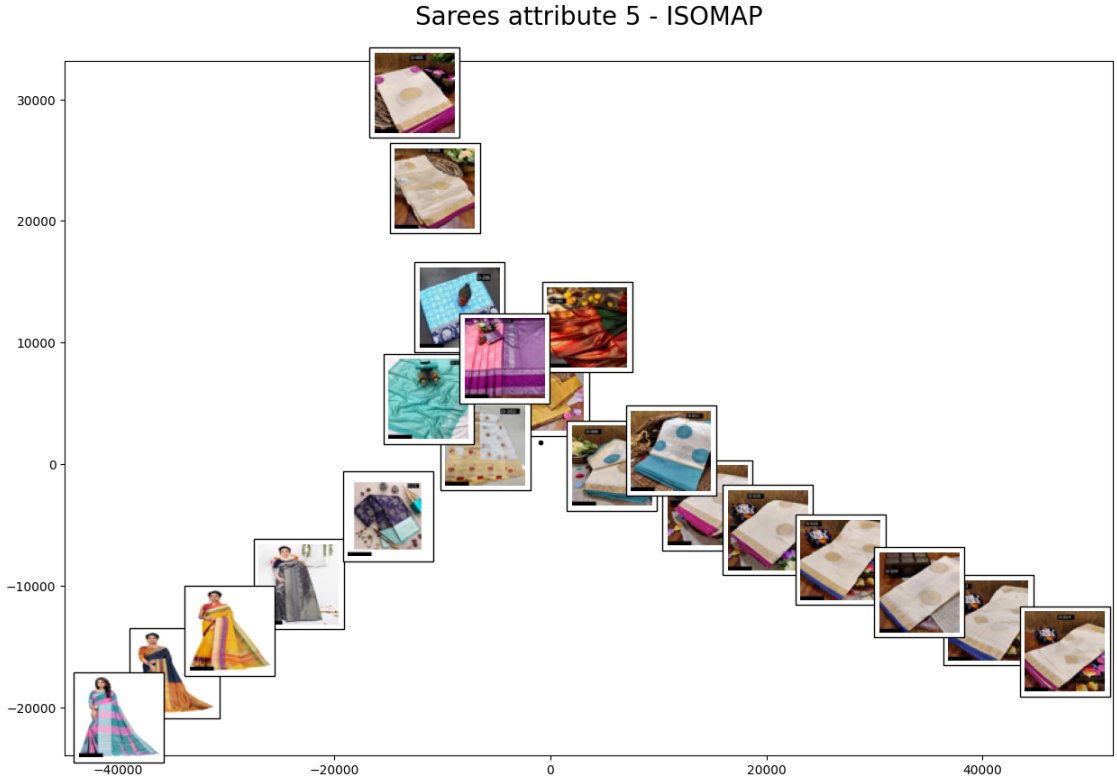


2 clusters bottom (without models) and top (with models)

y - axis -> color shade (light to dark)

[187]:
```
images, actual_images = load_and_preprocess_images(samples['Sarees_attr_5'])
```

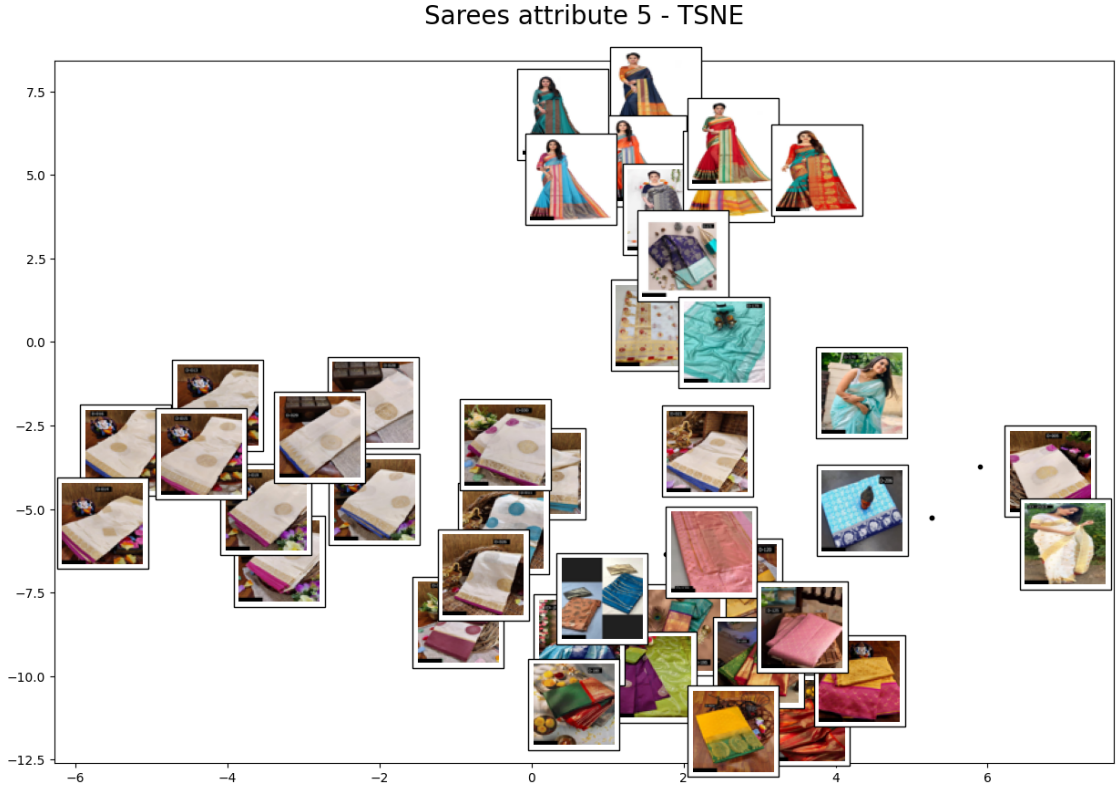
[161]:
```
plot_components(images,isomap, actual_images)
plt.title("Sarees attribute 5 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



X - axis -> color (colorful to plain sarees)

y - axis -> pattern (strips to plain)

[188]:
```
plot_components(images,tsne, actual_images)
plt.title("Sarees attribute 5 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



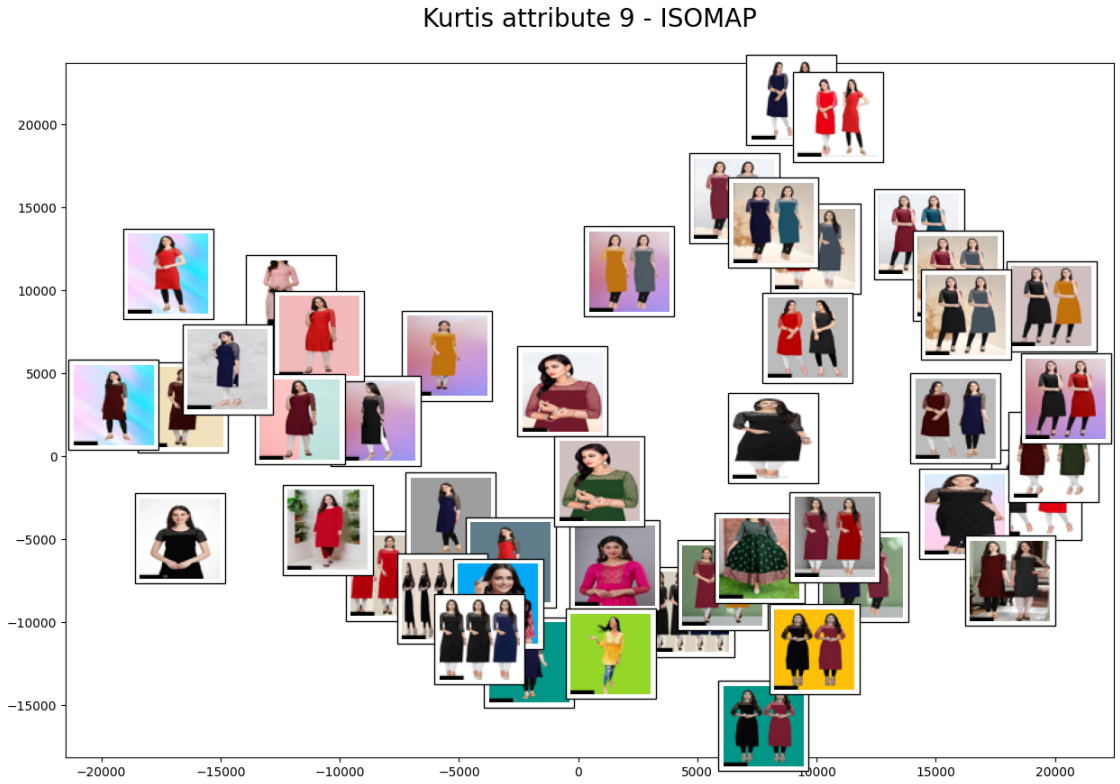
x - axis -> Color (plain to colorful)

y - axis -> model (not present to present)

c.) **Kurtis**

[163]:
```
images, actual_images = load_and_preprocess_images(samples['Kurtis_attr_9'])
```

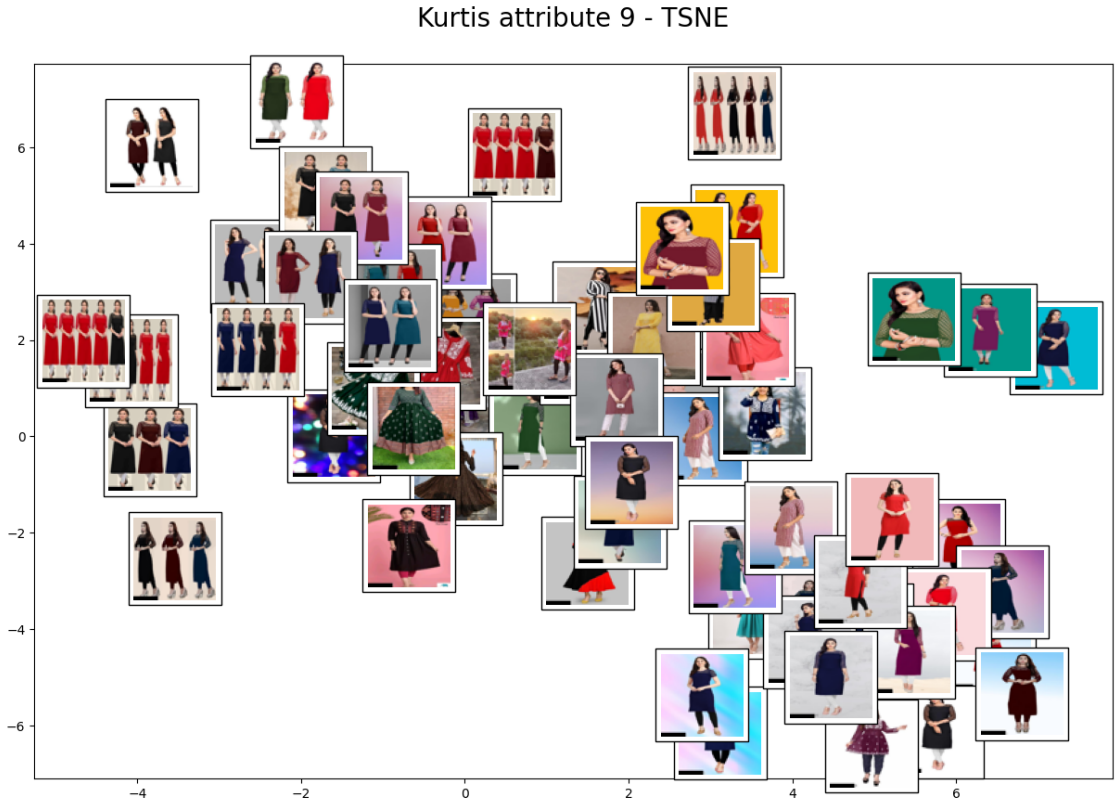
[164]:
```
plot_components(images,isomap, actual_images)
plt.title("Kurtis attribute 9 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



x - axis -> number of models

y - axis -> intensity of background color

[170]:
```
plot_components(images,tsne, actual_images)
plt.title("Kurtis attribute 9 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```

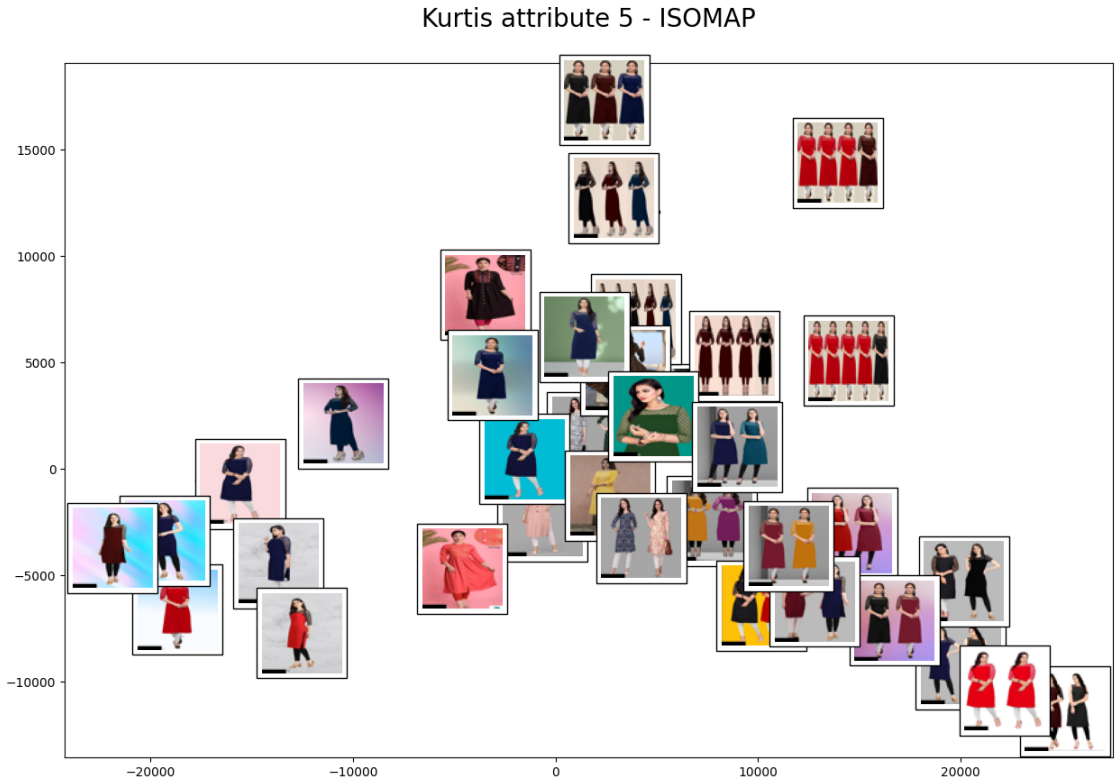


x - axis -> No of models (high to low)

y - axis -> background color difference

[166]:
```
images, actual_images = load_and_preprocess_images(samples['Kurtis_attr_5'])
```

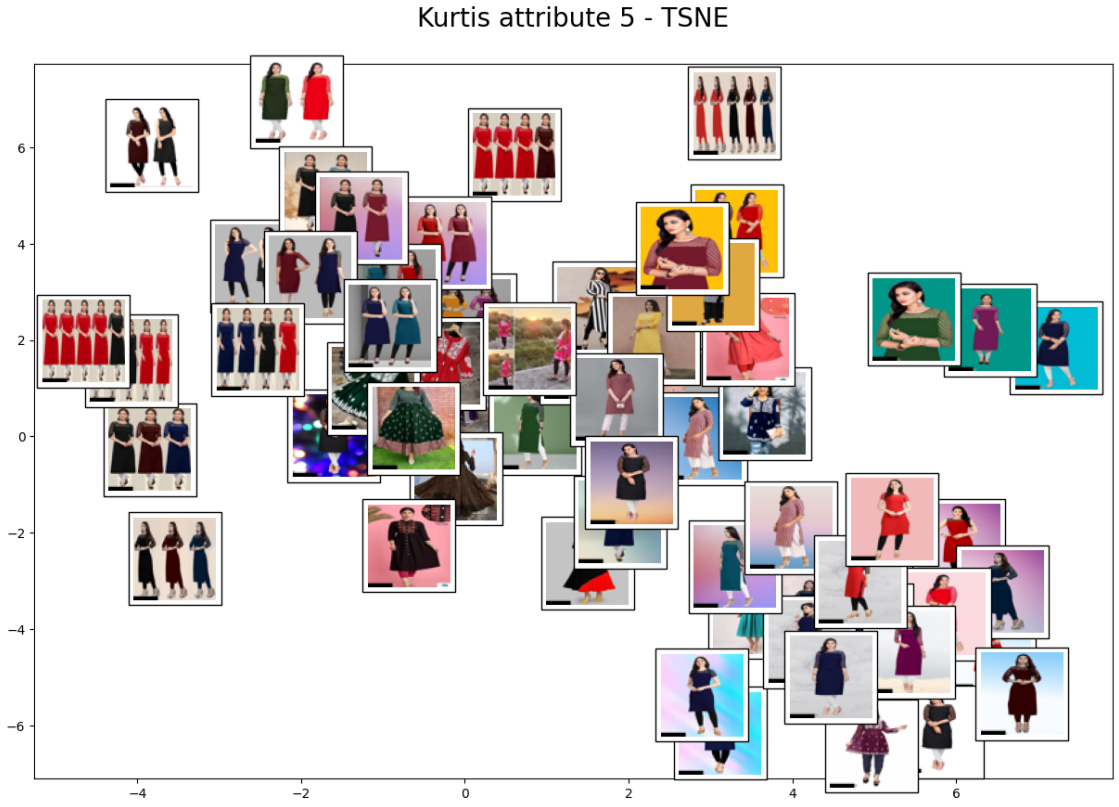
[168]:
```
plot_components(images,isomap, actual_images)
plt.title("Kurtis attribute 5 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



x - axis -> Background effect (complex to plain)

y - axis -> color intensity (bright to dark) and no of models

[169]:
```
plot_components(images,tsne, actual_images)
plt.title("Kurtis attribute 5 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



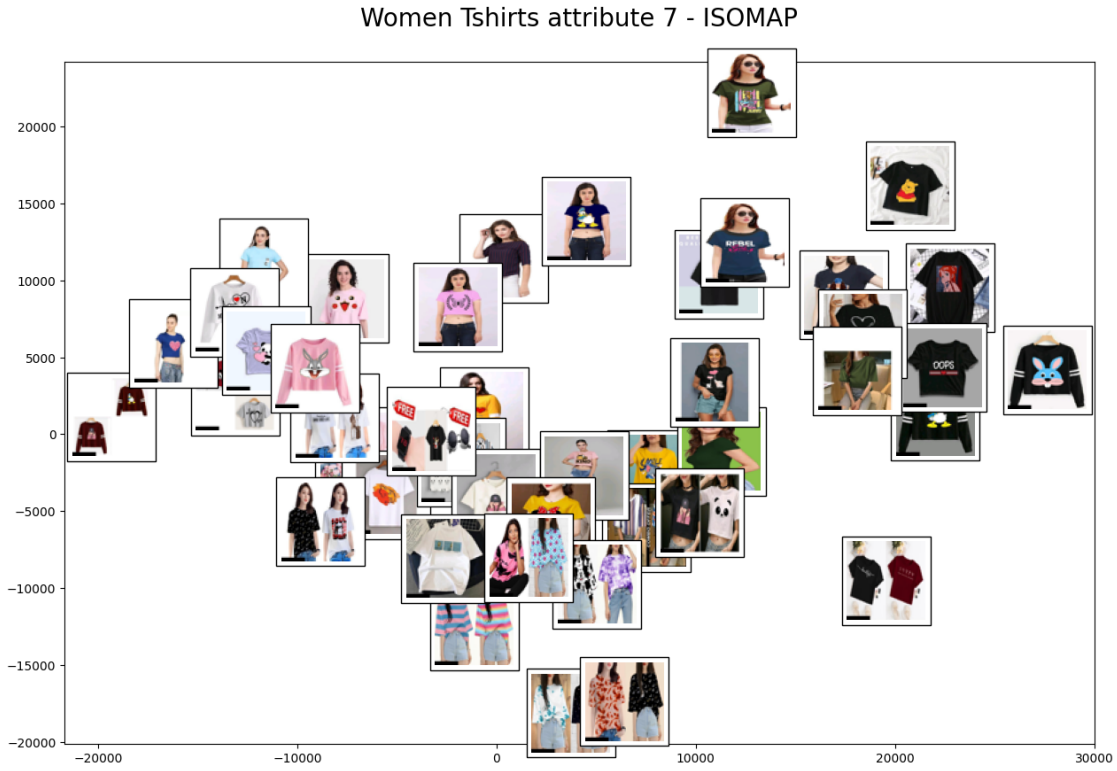
x axis -> No of models (high to low)

y axis -> No of models (low to high)

d.) **Women Tshirts**

[172]:
```
images, actual_images = load_and_preprocess_images(samples['Women␣
↪Tshirts_attr_7'])
```

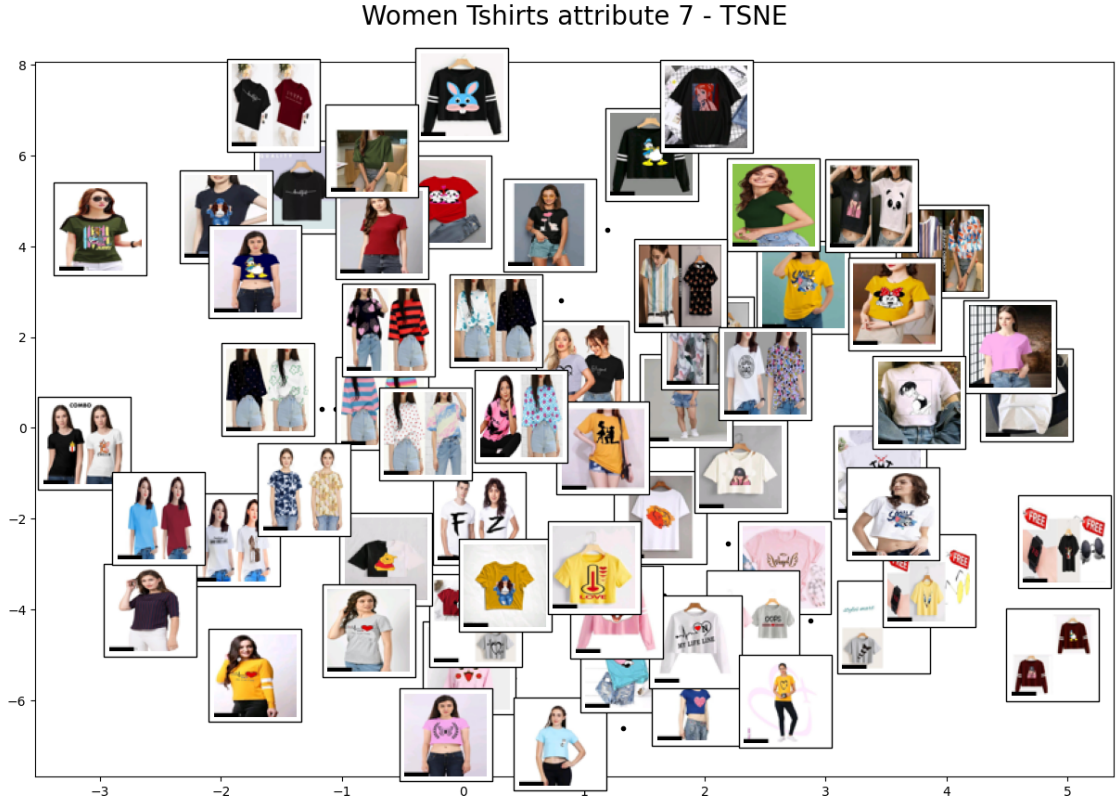
[173]:
```
plot_components(images,isomap, actual_images)
plt.title("Women Tshirts attribute 7 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



x axis -> color intensity (white to black)

y axis -> no of models

[175]:
```
plot_components(images,tsne, actual_images)
plt.title("Women Tshirts attribute 7 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```

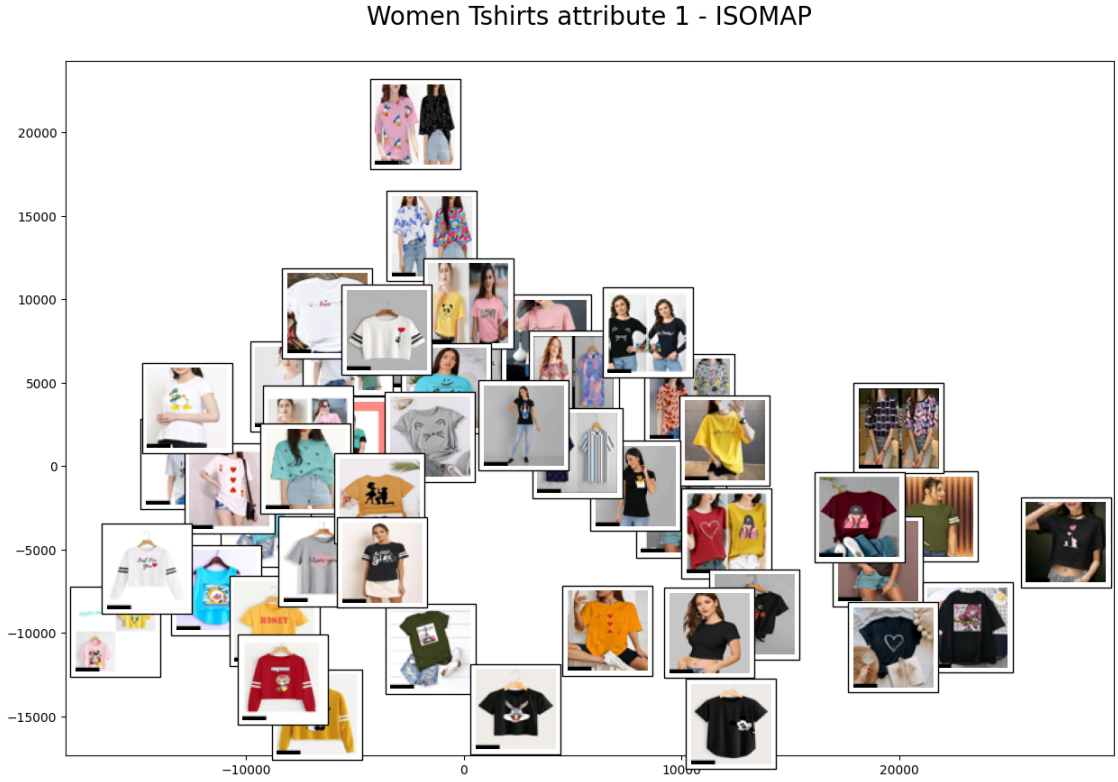


x axis -> length (long to short)

y axis -> color shade (lighter to darker)

[176]:
```
images, actual_images = load_and_preprocess_images(samples['Women␣
↪Tshirts_attr_1'])
```

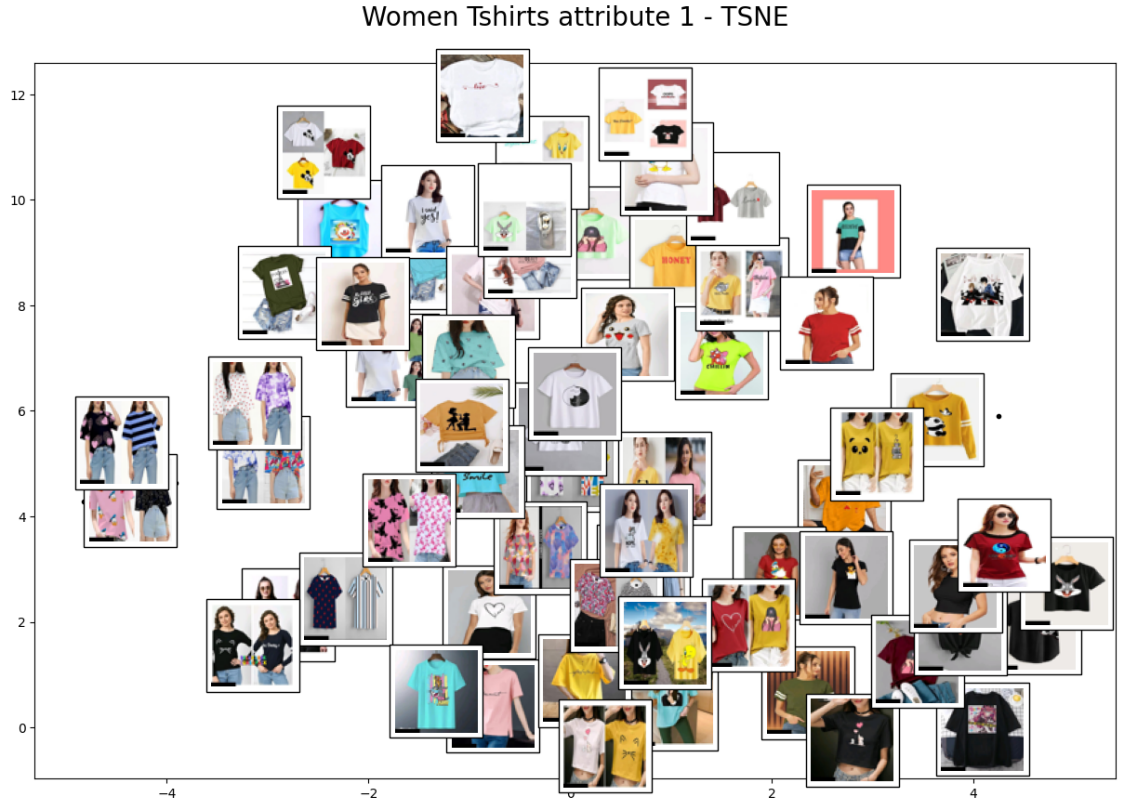
[177]:
```
plot_components(images,isomap, actual_images)
plt.title("Women Tshirts attribute 1 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



x axis -> color shade (lighter to darker) and model presence and backgroud intensity

y axis -> no of Tshirts (low to high)

[178]:
```
plot_components(images,tsne, actual_images)
plt.title("Women Tshirts attribute 1 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



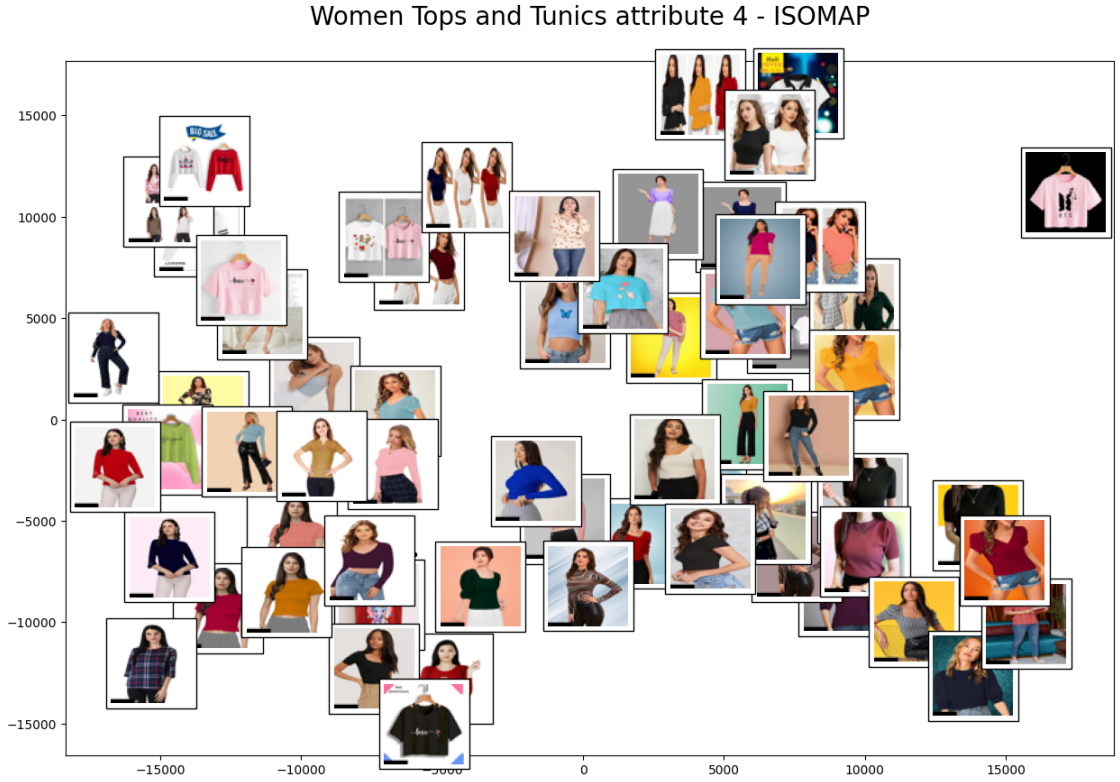
x axis -> no of t ahirts

y axis -> intensity of background

e.) **Women Tops & Tunics**

[179]:
```
images, actual_images = load_and_preprocess_images(samples['Women Tops &␣
↪Tunics_attr_4'])
```

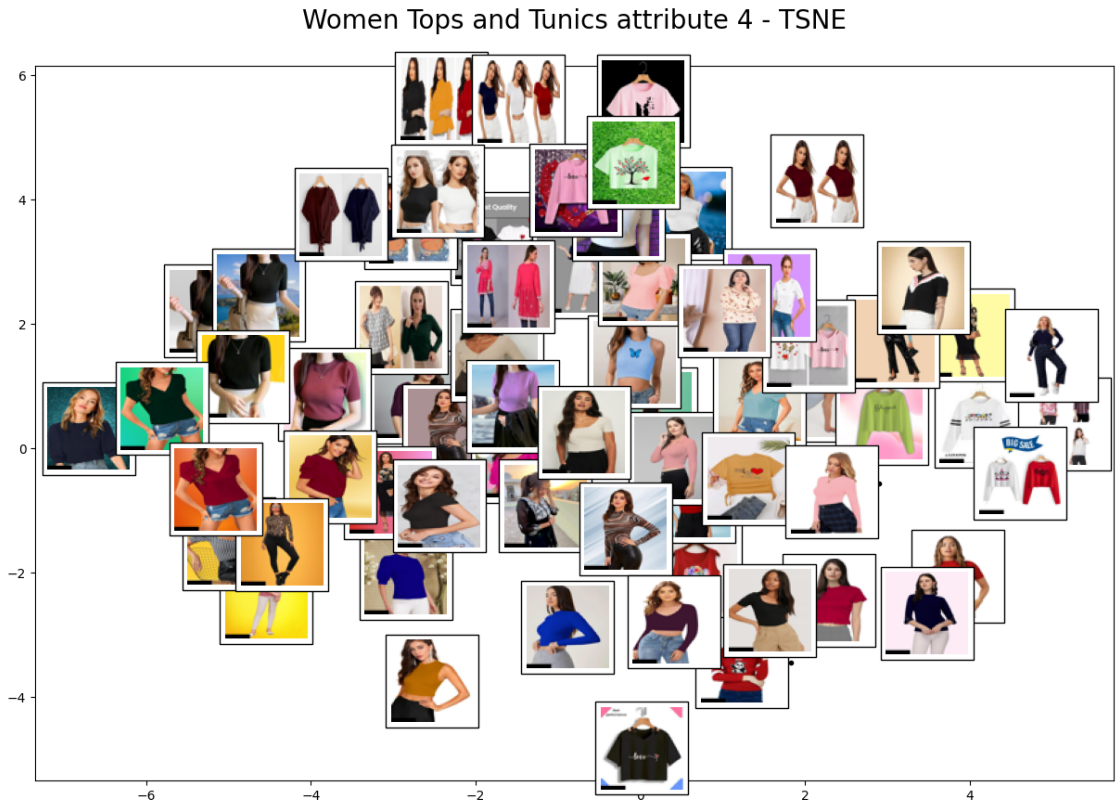
[180]:
```
plot_components(images,isomap, actual_images)
plt.title("Women Tops and Tunics attribute 4 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



x axis -> sleeve length (long to short)

y axis -> no of models/Tops

[181]:
```
plot_components(images,tsne, actual_images)
plt.title("Women Tops and Tunics attribute 4 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```

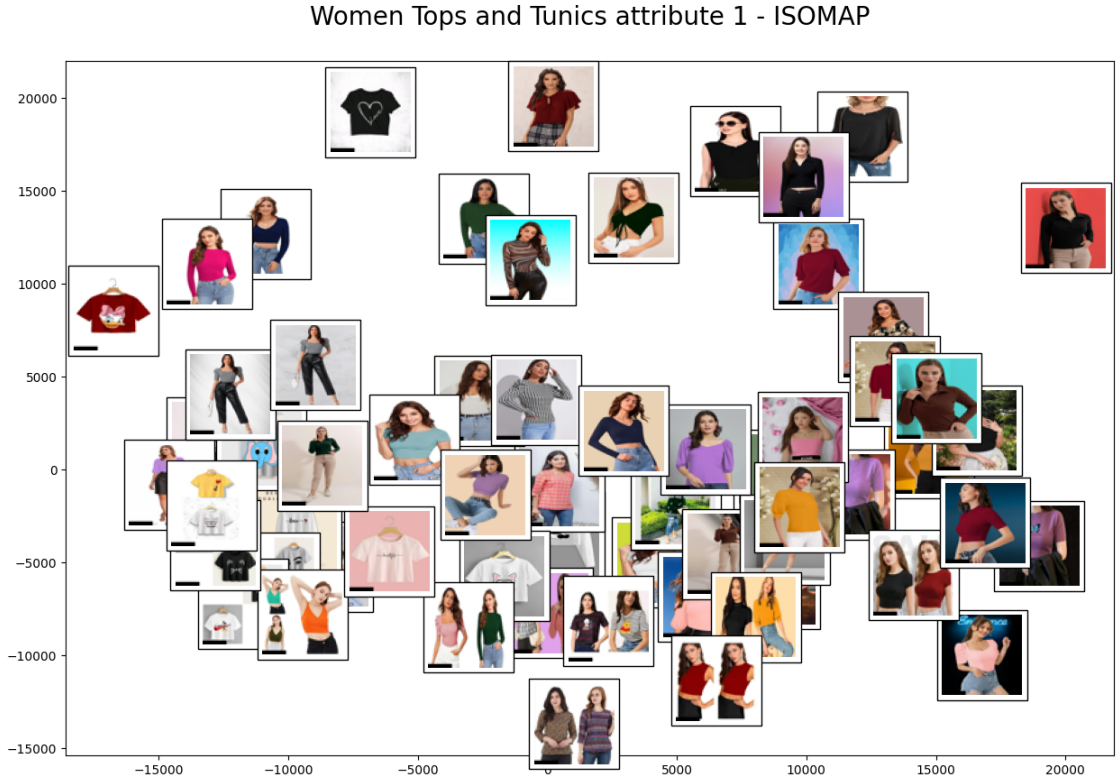


x axis -> sleeve length (short to long)

y axis -> no of tops

[182]:
```
images, actual_images = load_and_preprocess_images(samples['Women Tops &␣
↪Tunics_attr_1'])
```

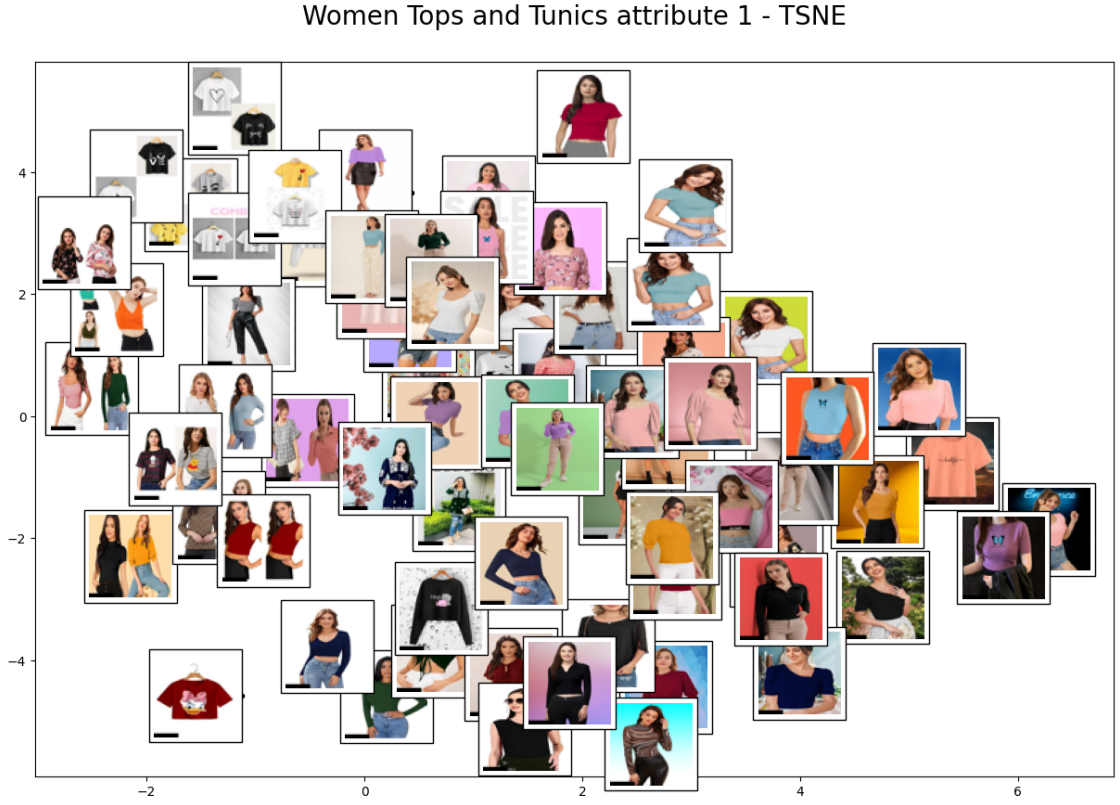
[184]:
```
plot_components(images,isomap, actual_images)
plt.title("Women Tops and Tunics attribute 1 - ISOMAP\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



x axis -> Background color intensity

y axis -> No of models (high to low)

[185]:
```
plot_components(images,tsne, actual_images)
plt.title("Women Tops and Tunics attribute 1 - TSNE\n", size = 20)
plt.gcf().set_size_inches(15, 10)
```



x axis -> No of tops/models and background color intensity

y axis -> Sleeve length (long to short)