Importing Packages

In [ ]:

```
from PIL import Image
import requests
from io import BytesIO
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import warnings
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
import math
import time
import re
import os
import itertools
import seaborn as sns
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import pairwise_distances
from matplotlib import gridspec
from scipy.sparse import hstack
import plotly
import plotly.figure_factory as ff
from tqdm import tqdm
from plotly.graph_objs import Scatter, Layout
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
from IPython.display import display, Image, SVG, Math, YouTubeVideo

plotly.offline.init_notebook_mode(connected=True)
warnings.filterwarnings("ignore")
```

Analysis of Data

In [ ]:

```
data = pd.read_json("/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/Applied_AI_Workshop_Code_Data/tops_fash
```

In [ ]:

```
print(f"Number of Data Points: {data.shape[0]}, Number of Features/Variables: {data.shape[1]}")
```

```
Number of Data Points: 183138, Number of Features/Variables: 19
```

In [ ]:

```
print(f"Columns of Dataset are: {list(data.columns)}")
```

```
Columns of Dataset are: ['sku', 'asin', 'product_type_name', 'formatted_price', 'author', 'color', 'brand', 'publi
```

Columns used in this Case Study are:

- Amazon Standard Identification Number (asin)
- Brand item belongs to (brand)
- Color (color)
- Product Type Name (product_type_name)
- Image URL (medium_image_url)
- Title of the Product (title)
- Price of the Product (formatted_price)

In [ ]:

```
data = data[['asin', 'brand', 'color', 'medium_image_url', 'product_type_name', 'title', 'formatted_price']]
```

In [ ]:

```
print(f"Number of Data Points: {data.shape[0]}\n Number of Features: {data.shape[1]}")
data.head()
```

```
Number of Data Points: 183138
 Number of Features: 7
```

Out[ ]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
0	B016I2TS4W	FNC7C	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Minions Como Superheroes Ironman Long Sleeve R...	None
1	B01N49AI08	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Izo Tunic	None
2	B01JDPCOHO	FIG Clothing	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	FIG Clothing Womens Won Top	None
3	B01N19U5H5	Focal18	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Focal18 Sailor Collar Bubble Sleeve Blouse Shi...	None
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherlite Ladies' Long Sleeve Stain Resistan...	$26.26

Univariate Analysis

Product Type

In [ ]:

```
print(data['product_type_name'].describe())
print(f"\n{((data['product_type_name'].describe()['freq']/data['product_type_name'].describe()['count'])*100):.2f}% of the data belor
```

```
count     183138
unique        72
top        SHIRT
freq      167794
Name: product_type_name, dtype: object

91.62% of the data belongs to the type SHIRT
```

In [ ]:

```
print(f"The Unique types of Products are: {data['product_type_name'].unique()}")
```

```
The Unique types of Products are: ['SHIRT' 'SWEATER' 'APPAREL' 'OUTDOOR_RECREATION_PRODUCT'
 'BOOKS_1973_AND_LATER' 'PANTS' 'HAT' 'SPORTING_GOODS' 'DRESS' 'UNDERWEAR'
 'SKIRT' 'OUTERWEAR' 'BRA' 'ACCESSORY' 'ART_SUPPLIES' 'SLEEPWEAR'
 'ORCA_SHIRT' 'HANDBAG' 'PET_SUPPLIES' 'SHOES' 'KITCHEN' 'ADULT_COSTUME'
 'HOME_BED_AND_BATH' 'MISC_OTHER' 'BLAZER' 'HEALTH_PERSONAL_CARE'
 'TOYS_AND_GAMES' 'SWIMWEAR' 'CONSUMER_ELECTRONICS' 'SHORTS' 'HOME'
 'AUTO_PART' 'OFFICE_PRODUCTS' 'ETHNIC_WEAR' 'BEAUTY'
 'INSTRUMENT_PARTS_AND_ACCESSORIES' 'POWERSPORTS_PROTECTIVE_GEAR' 'SHIRTS'
 'ABIS_APPAREL' 'AUTO_ACCESSORY' 'NONAPPARELMISC' 'TOOLS' 'BABY_PRODUCT'
 'SOCKSHOSIERY' 'POWERSPORTS_RIDING_SHIRT' 'EYEWEAR' 'SUIT'
 'OUTDOOR_LIVING' 'POWERSPORTS_RIDING_JACKET' 'HARDWARE' 'SAFETY_SUPPLY'
 'ABIS_DVD' 'VIDEO_DVD' 'GOLF_CLUB' 'MUSIC_POPULAR_VINYL'
 'HOME_FURNITURE_AND_DECOR' 'TABLET_COMPUTER' 'GUILD_ACCESSORIES'
 'ABIS_SPORTS' 'ART_AND_CRAFT_SUPPLY' 'BAG' 'MECHANICAL_COMPONENTS'
 'SOUND_AND_RECORDING_EQUIPMENT' 'COMPUTER_COMPONENT' 'JEWELRY'
 'BUILDING_MATERIAL' 'LUGGAGE' 'BABY_COSTUME' 'POWERSPORTS_VEHICLE_PART'
 'PROFESSIONAL_HEALTHCARE' 'SEEDS_AND_PLANTS' 'WIRELESS_ACCESSORY']
```

In [ ]:

```
product_type_count = Counter(list(data['product_type_name']))
print(f"The 10 most frequent product types are: \n{product_type_count.most_common(10)}")
```

```
The 10 most frequent product types are:
[('SHIRT', 167794), ('APPAREL', 3549), ('BOOKS_1973_AND_LATER', 3336), ('DRESS', 1584), ('SPORTING_GOODS', 1281),
```

Color

In [ ]:

```
print(data['color'].describe())
print(f"\n{((data['color'].describe()['freq']/data['color'].describe()['count'])*100):.2f}% of the data are {data['color'].describe(
```

```
count     64956
unique     7380
top       Black
freq      13207
Name: color, dtype: object

20.33% of the data are Black in color
```

In [ ]:

```
color_count = Counter(list(data['color']))
print(f"The top 10 most frequent colors are: \n{color_count.most_common(10)}")
```

```
The top 10 most frequent colors are:
[(None, 118182), ('Black', 13207), ('White', 8616), ('Blue', 3570), ('Red', 2289), ('Pink', 1842), ('Grey', 1499),
```

Price

In [ ]:

```
print(f"{data['formatted_price'].describe()}\nOnly {(data['formatted_price'].describe()['count'])/(data['formatted_price'].shape[0])*
```

```
count      28395
unique      3135
top       $19.99
freq         945
Name: formatted_price, dtype: object
Only 15.50% of the data have price information.
```

Title

In [ ]:

```
data['title'].head()
```

Out[ ]:

```
0    Minions Como Superheroes Ironman Long Sleeve R...
1                        FIG Clothing Womens Izo Tunic
2                          FIG Clothing Womens Won Top
3    Focal18 Sailor Collar Bubble Sleeve Blouse Shi...
4    Featherlite Ladies' Long Sleeve Stain Resistan...
Name: title, dtype: object
```

The total number of datapoints in the dataset is around 180000. Not all of them have price and color information. Omitting these datapoints to reduce the size of the dataset.

In [ ]:

```
data = data.loc[~data['formatted_price'].isnull()]
print(f"Number of datapoints after removing datapoints without price feature: {data.shape[0]}")
```

```
Number of datapoints after removing datapoints without price feature: 28395
```

In [ ]:

```
data = data.loc[~data['color'].isnull()]
print(f"Number of datapoints after removing datapoints without color feature: {data.shape[0]}")
```

```
Number of datapoints after removing datapoints without color feature: 28385
```

Removing Duplicate Elements

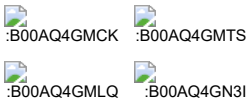
Understanding how duplicates are formed

In [ ]:

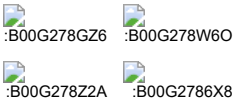
```
print(f"Number of products with the same title: {sum(data.duplicated('title'))}")
```

```
Number of products with the same title: 2325
```

These shirts have the same title, but different sizes.



These shirts have the same title, but different colors.



The dataset contains many such producest which need to be de-duped

Removing Duplicates

In [ ]:

```
data.head()
```

Out[ ]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	Featherlite Ladies' Long Sleeve Stain Resistan...	$26.26
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	Women's Unique 100% Cotton T - Special Olympic...	$9.99
11	B001LOUGE4	Fitness Etc.	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	Ladies Cotton Tank 2x1 Ribbed Tank Top	$11.99
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	FeatherLite Ladies' Moisture Free Mesh Sport S...	$20.54
21	B014ICEDNA	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	Supernatural Chibis Sam Dean And Castiel Short...	$7.50

In [ ]:

```
data_sorted = data[data['title'].apply(lambda x: len(x.split()) > 4)]
print(f"After removal of products with short description: {data_sorted.shape[0]}")
```

```
After removal of products with short description: 27949
```

In [ ]:

```
data_sorted.sort_values('title', inplace = True, ascending = False)
data_sorted.head()
```

Out[ ]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	Éclair Women's Printed Thin Strap Blouse Black...	$24.99
133820	B010RV33VE	xiaoming	Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Womens Sleeveless Loose Long T-shirts...	$18.19
81461	B01DDSDLNS	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Women's White Long Sleeve Single Brea...	$21.58
75995	B00X5LYO9Y	xiaoming	Red Anchors	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Stripes Tank Patch/Bear Sleeve Anchor...	$15.91
151570	B00WPJG35K	xiaoming	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	xiaoming Sleeve Sheer Loose Tassel Kimono Woma...	$14.32

Some titles have very similar sentences apart from a few words. We can use a threshold based system to elimnate very similar titles/datapoints.

In [ ]:

```
data.to_pickle('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/28k_apperal_data')
```

In [ ]:

```
data = pd.read_pickle('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/28k_apperal_data')
```

In [ ]:

```
indices = []
for i, row in data_sorted.iterrows():
  indices.append(i)
```

- Start from the first row
- For each row, compare current row with previous row.
- If the number of different words is <= 2, store the first index in a list and move to the next row.
- Repeat until title with > 2 number of words.
- Update flag values

In [ ]:

```
# non_duplicate_indices = []
# i = 0
# j = 0
# size = data_sorted.shape[0]
# while i < size and j < size:
#   prev_i = i
#   a = data['title'].loc[indices[i]].split()
#   j = i + 1
#   while j < size:
#     b = data['title'].loc[indices[i]].split()
#     length = max(len(a), len(b))
#     count = 0
#     for k in itertools.zip_longest(a, b):
#       if(k[0] == k[1]):
#         count += 1
#     if (length - count) > 2:
#       non_duplicate_indices.append(data_sorted['asin'].loc[indices[i]])
#       i = j
#       break
#     else:
#       j += 1
#   if prev_i == i:
#     break
```

In [ ]:

```
stage1_dedupe_asins = []
i = 0
j = 0
num_data_points = data_sorted.shape[0]
while i < num_data_points and j < num_data_points:
    previous_i = i
    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large
    a = data['title'].loc[indices[i]].split()
    # search for the similar products sequentially
    j = i+1
    while j < num_data_points:
        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'Sma
        b = data['title'].loc[indices[j]].split()
        # store the maximum length of two strings
        length = max(len(a), len(b))
        # count is used to store the number of words that are matched in both strings
        count  = 0
        # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will appened None in case of unequal strir
        # example: a =['a', 'b', 'c', 'd']
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', None)]
        for k in itertools.zip_longest(a,b):
            if (k[0] == k[1]):
                count += 1
        # if the number of words in which both strings differ are > 2 , we are considering it as those two apperals are different
        # if the number of words in which both strings differ are < 2 , we are considering it as those two apperals are same, hence w
        if (length - count) > 2: # number of words in which both sensences differ
            # if both strings are differ by more than 2 words we include the 1st string index
            stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[i]])
            # start searching for similar apperals corresponds 2nd string
            i = j
            break
        else:
            j += 1
    if previous_i == i:
        break
```

In [ ]:

```
data = data.loc[data['asin'].isin(stage1_dedupe_asins)]
```

In [ ]:

```
print(f"Number of datapoints: {data.shape[0]}")
```

```
Number of datapoints: 17592
```

In [ ]:

```
data.to_pickle('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/17k_apparel_data')
```

Some titles have similar sentences but there are some different words as a result of which they are not alphabetically adjacent to each other. To remove such duplicates, we can run a Brute Force algorithm that compares each title to every other title for similarity. An optimal approach could be to use Inverted Indices.

In [ ]:

```
data = pd.read_pickle('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/17k_apparel_data')
```

In [ ]:

```
def get_similar_title_indices(data):
  indices = []
  for i, row in data.iterrows():
    indices.append(i)

  non_duplicate_indices = []
  while len(indices) != 0:
    i = indices.pop()
    non_duplicate_indices.append(data['asin'].loc[i])
    a = data['title'].loc[i].split()
    for j in indices:
      b = data['title'].loc[j].split()
      length = max(len(a), len(b))
      count = 0
      for k in itertoools.zip_longest(a, b):
        if(k[0] == k[i]):
          count += 1
      if length - count < 3:
        indices.remove(j)
  return non_duplicate_indices
```

In [ ]:

```
"""
non_duplicate_indices = get_similar_title_indices(data)
data = data.loc[data['asin'].isin(non_duplicate_indices)]
print(f"Number of datapoints after removing similar titles: {data.shape[0]}")
data.to_pickle('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/16k_apparel_data')
"""
```

Out[ ]:

```
'\nnon_duplicate_indices = get_similar_title_indices(data)\ndata = data.loc[data[\'asin\'].isin(non_duplicate_indi
```

Text Pre Processing

In [ ]:

```
data = pd.read_pickle('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/16k_apperal_data')
```

In [ ]:

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[ ]:

```
True
```

In [ ]:

```
stop_words = set(stopwords.words('english'))
print(f"List of Stop Words: {stop_words}")
```

```
List of Stop Words: {"should've", 'from', 'again', 'while', 'it', 'they', 'him', 'should', "you'll", 'both', 'not'
```

In [ ]:

```
def nlp_preprocessing(total_text, index, column):
  if type(total_text) is not int:
    string = ""
    for words in total_text.split():
      word = ("".join(e for e in words if e.isalnum()))
      word = word.lower()
      if not word in stop_words:
        string += word + " "
    data[column][index] = string
```

In [ ]:

```
start_time = time.clock()
for index, row in tqdm(data.iterrows()):
  nlp_preprocessing(row['title'], index, 'title')
print(f"\nTime taken to process text: {time.clock() - start_time} seconds")
```

```
16042it [00:06, 2608.82it/s]

Time taken to process text: 6.123265999999999 seconds
```

In [ ]:

```
data.head()
```

Out[ ]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies long sleeve stain resistant...	$26.26
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	womens unique 100 cotton special olympics wor...	$9.99
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies moisture free mesh sport sh...	$20.54
27	B014ICEJ1Q	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	supernatural chibis sam dean castiel neck tshi...	$7.39
46	B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth degree womens gold foil graphic tees jun...	$6.95

In [ ]:

```
data.to_pickle("/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/preprocessed_16k")
```

Not using Stemming as it showed poor results.

Text Based Product Similarity

In [ ]:

```
data = pd.read_pickle("/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/preprocessed_16k")
data.head()
```

Out[ ]:

	asin	brand	color	medium_image_url	product_type_name	title	formatted_price
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies long sleeve stain resistant...	$26.26
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	womens unique 100 cotton special olympics wor...	$9.99
15	B003BSRPB0	FeatherLite	White	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies moisture free mesh sport sh...	$20.54
27	B014ICEJ1Q	FNC7C	Purple	https://images-na.ssl-images-amazon.com/images...	SHIRT	supernatural chibis sam dean castiel neck tshi...	$7.39
46	B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images...	SHIRT	fifth degree womens gold foil graphic tees jun...	$6.95

In [ ]:

```
def display_img(url, ax, fig):
  response = requests.get(url)
  img = Image.open(BytesIO(response.content))
  plt.imshow(img)
```

In [ ]:

```
def plot_heatmap(keys, values, labels, url, text):
  """
  keys   : list of words of recommended product's title
  values : len(values) == len(keys), values(i) represents the occurence of the word keys(i)
  labels : len(labels) == len(keys), the values of labels depend on the model being used
           if model == bag of words, labels(i) = values(i)
           if model == tfidf weighted bag of words, labels(i) = tfidf(keys(i))
           if model == idf weighted bag of words, labels(i) = idf(keys(i))
  url    : product image's url
  """
  gs = gridspec.GridSpec(2, 2, width_ratios=[4,1], height_ratios=[4,1])
  fig = plt.figure(figsize=(25,3))

  # 1st, ploting heat map that represents the count of commonly ocurred words in title2
  ax = plt.subplot(gs[0])
  # it displays a cell in white color if the word is intersection(lis of words of title1 and list of words of title2), in black if no
  ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
  ax.set_xticklabels(keys) # set that axis labels as the words of title
  ax.set_title(text) # apparel title

  # 2nd, plotting image of the the apparel
  ax = plt.subplot(gs[1])
  # we don't want any grid lines for image and no labels on x-axis and y-axis
  ax.grid(False)
  ax.set_xticks([])
  ax.set_yticks([])

  # we call dispaly_img based with paramete url
  display_img(url, ax, fig)

  # displays combine figure ( heat map and image together)
  plt.show()
```

In [ ]:

```
def plot_heatmap_image(doc_id, vec1, vec2, url, text, model):
  # doc_id : index of the title1
  # vec1 : input apparels's vector, it is of a dict type {word:count}
  # vec2 : recommended apparels's vector, it is of a dict type {word:count}
  # url : apparels image url
  # text: title of recomonded apparel (used to keep title of image)
  # model, it can be any of the models,
      # 1. bag_of_words
      # 2. tfidf
      # 3. idf

  # we find the common words in both titles, because these only words contribute to the distance between two title vec's
  intersection = set(vec1.keys()) & set(vec2.keys())

  # we set the values of non intersecting words to zero, this is just to show the difference in heatmap
  for i in vec2:
      if i not in intersection:
          vec2[i]=0

  # for labeling heatmap, keys contains list of all words in title2
  keys = list(vec2.keys())
  #  if ith word in intersection(lis of words of title1 and list of words of title2): values(i)=count of that word in title2 else val
  values = [vec2[x] for x in vec2.keys()]

  # labels: len(labels) == len(keys), the values of labels depends on the model we are using
      # if model == 'bag of words': labels(i) = values(i)
      # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
      # if model == 'idf weighted bag of words':labels(i) = idf(keys(i))

  if model == 'bag_of_words':
      labels = values
  elif model == 'tfidf':
      labels = []
      for x in vec2.keys():
          # tfidf_title_vectorizer.vocabulary_ it contains all the words in the corpus
          # tfidf_title_features[doc_id, index_of_word_in_corpus] will give the tfidf value of word in given document (doc_id)
          if x in  tfidf_title_vectorizer.vocabulary_:
              labels.append(tfidf_title_features[doc_id, tfidf_title_vectorizer.vocabulary_[x]])
          else:
              labels.append(0)
  elif model == 'idf':
      labels = []
      for x in vec2.keys():
          # idf_title_vectorizer.vocabulary_ it contains all the words in the corpus
          # idf_title_features[doc_id, index_of_word_in_corpus] will give the idf value of word in given document (doc_id)
          if x in  idf_title_vectorizer.vocabulary_:
              labels.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[x]])
          else:
              labels.append(0)

  plot_heatmap(keys, values, labels, url, text)
```

In [ ]:

```
def text_to_vector(text):
  """
  This function gets a list of words along with the frequency of each word given some text
  """
  word = re.compile(r'\w+')
  words = word.findall(text)
  # words stores list of all words in given string, you can try 'words = text.split()' this will also gives same result
  return Counter(words) # Counter counts the occurence of each word in list, it returns dict type object {word1:count}
```

In [ ]:

```
def get_result(doc_id, content_a, content_b, url, model):
    text1 = content_a
    text2 = content_b

    # vector1 = dict{word11:#count, word12:#count, etc.}
    vector1 = text_to_vector(text1)

    # vector1 = dict{word21:#count, word22:#count, etc.}
    vector2 = text_to_vector(text2)

    plot_heatmap_image(doc_id, vector1, vector2, url, text2, model)
```

Bag of Words (BoW) on Product Titles

In [ ]:

```
title_vectorizer = CountVectorizer()
title_features = title_vectorizer.fit_transform(data['title'])
title_features.get_shape()
```

Out[ ]:

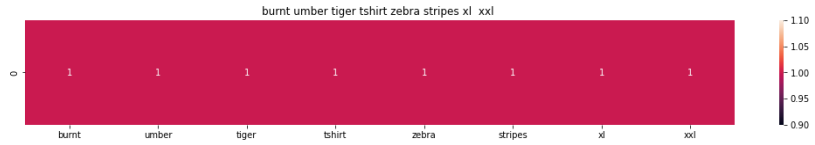
```
(16042, 12609)
```

In [ ]:

```
def bag_of_words_model(doc_id, num_results):
  pairwise_dist = pairwise_distances(title_features, title_features[doc_id])
  indices = np.argsort(pairwise_dist.flatten())[:num_results]
  pdists = np.sort(pairwise_dist.flatten())[0:num_results]
  df_indices = list(data.index[indices])
  for i in range(0, len(indices)):
    get_result(indices[i],data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices
    print(f"ASIN {data['asin'].loc[df_indices[i]]}")
    print(f"Brand {data['brand'].loc[df_indices[i]]}")
    print(f"Title {data['title'].loc[df_indices[i]]}")
    print(f"Euclidean similarity with the query image {pdists[i]}")
    print('='*60)
```

In [ ]:

```
bag_of_words_model(12566, 10)
```

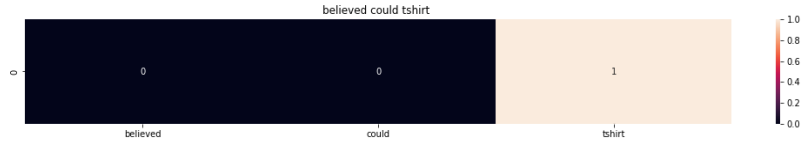


ASIN B00JXQB5FQ
Brand Si Row
Title burnt umber tiger tshirt zebra stripes xl xxl
Euclidean similarity with the query image 0.0
==

ASIN B00JXQASS6
Brand Si Row
Title pink tiger tshirt zebra stripes xl xxl
Euclidean similarity with the query image 1.7320508075688772
==

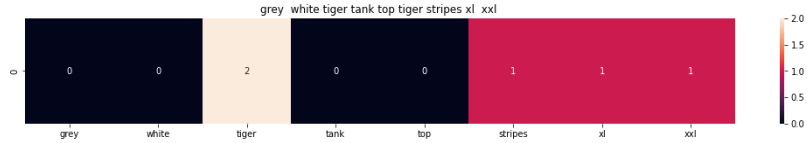
ASIN B00JXQCWTO
Brand Si Row
Title brown white tiger tshirt tiger stripes xl xxl
Euclidean similarity with the query image 2.449489742783178
==

ASIN B00JXQCUIC
Brand Si Row
Title yellow tiger tshirt tiger stripes l
Euclidean similarity with the query image 2.6457513110645907
==



ASIN B07568NZX4
Brand Rustic Grace
Title believed could tshirt
Euclidean similarity with the query image 3.0
==

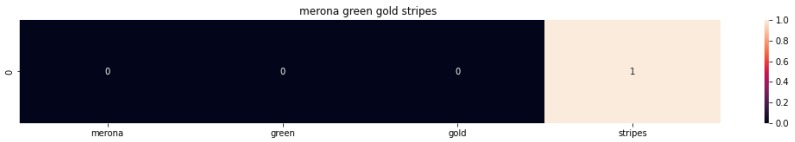
ASIN B01NB0NKRO
Brand Ideology
Title ideology graphic tshirt xl white
Euclidean similarity with the query image 3.0
==



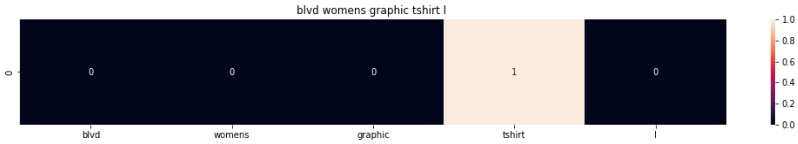
ASIN B00JXQAFZ2
Brand Si Row
Title grey white tiger tank top tiger stripes xl xxl

```
Euclidean similarity with the query image 3.0
============================================================
```

```
ASIN B01CLS8LMW
Brand Awake
Title morning person tshirt  troll picture  xl
Euclidean similarity with the query image 3.1622776601683795
============================================================
```



```
ASIN B01KVZUB6G
Brand Merona
Title merona green gold stripes
Euclidean similarity with the query image 3.1622776601683795
============================================================
```



```
ASIN B0733R2CJK
Brand BLVD
Title blvd womens graphic tshirt l
Euclidean similarity with the query image 3.1622776601683795
============================================================
```

TFIDF Based Product Similarity

In [ ]:

```
tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
```

In [ ]:

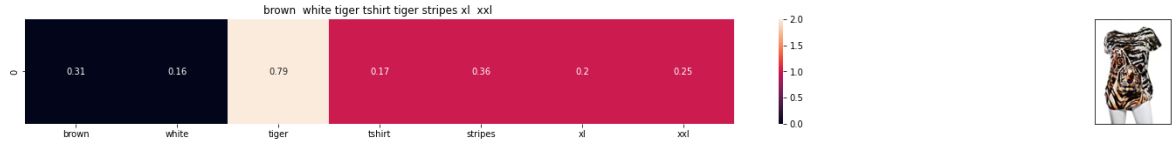
```
def tfidf_model(doc_id, num_results):
  pairwise_dist = pairwise_distances(tfidf_title_features, tfidf_title_features[doc_id])
  indices = np.argsort(pairwise_dist.flatten())[0:num_results]
  pdists = np.sort(pairwise_dist.flatten())[0:num_results]
  df_indices = list(data.index[indices])
  for i in range(len(indices)):
    get_result(indices[i], data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indice
    print(f"ASIN: {data['asin'].loc[df_indices[i]]}")
    print(f"Brand: {data['brand'].loc[df_indices[i]]}")
    print(f"Euclidean Distance from the given image: {pdists[i]}")
    print('='*100)
```

In [ ]:

```
tfidf_model(12566, 20)
```

ASIN: B00JXQB5FQ
Brand: Si Row
Euclidean Distance from the given image: 0.0
==========

ASIN: B00JXQASS6
Brand: Si Row
Euclidean Distance from the given image: 0.7536331912451363
==========

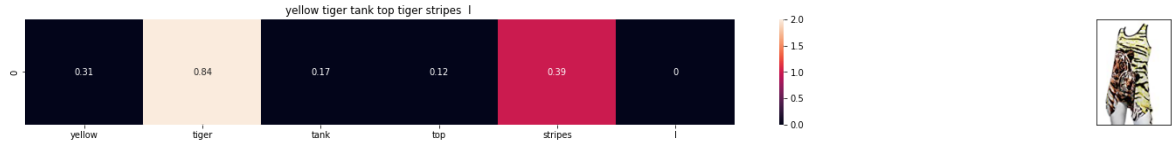


ASIN: B00JXQCWTO
Brand: Si Row
Euclidean Distance from the given image: 0.9357643943769647
==========

ASIN: B00JXQAFZ2
Brand: Si Row
Euclidean Distance from the given image: 0.9586153524200749
==========

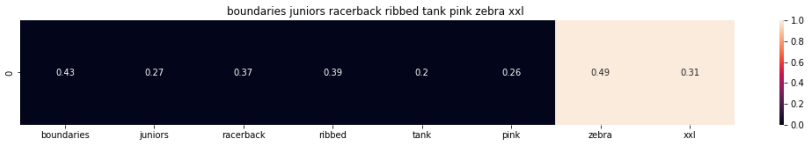
ASIN: B00JXQCUIC
Brand: Si Row
Euclidean Distance from the given image: 1.000074961446881
==========

ASIN: B00JXQAO94
Brand: Si Row
Euclidean Distance from the given image: 1.023215552457452
==========



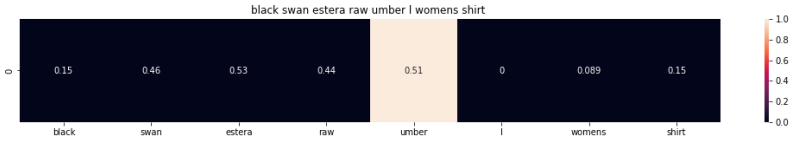
ASIN: B00JXQAUWA
Brand: Si Row
Euclidean Distance from the given image: 1.031991846303421
==========

ASIN: B06XSCVFT5
Brand: Studio M
Euclidean Distance from the given image: 1.2106843670424716
==========



ASIN: B06Y2GTYPM
Brand: No Boundaries
Euclidean Distance from the given image: 1.2121683810720831
==========

ASIN: B012VQLT6Y
Brand: KM T-shirt
Euclidean Distance from the given image: 1.219790640280982
==========



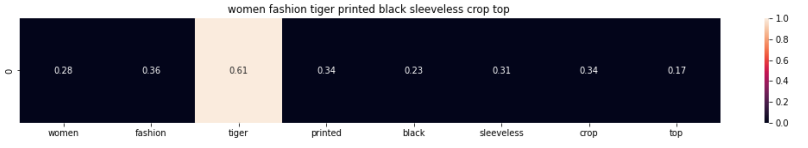
ASIN: B06Y1VN8WQ
Brand: Black Swan
Euclidean Distance from the given image: 1.2206849659998316
==========

ASIN: B00Z6HEXWI
Brand: Black Temptation
Euclidean Distance from the given image: 1.221281392120943
==========

ASIN: B074TR12BH
Brand: Ultra Flirt
Euclidean Distance from the given image: 1.2313364094597743
==========

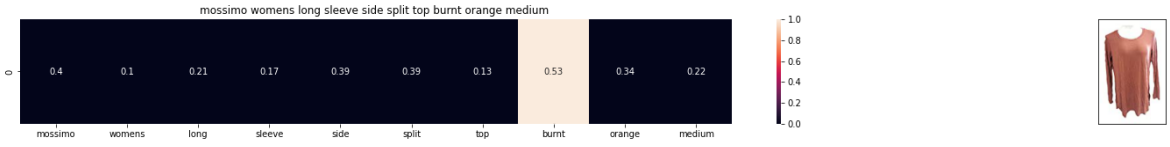
ASIN: B072R2JXKW
Brand: WHAT ON EARTH

Euclidean Distance from the given image: 1.2318451972624516
==========

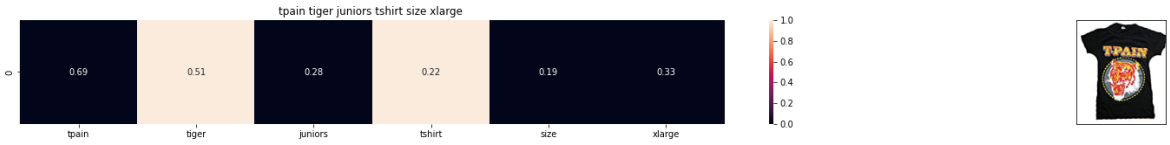


ASIN: B074T8ZYGX
Brand: MKP Crop Top
Euclidean Distance from the given image: 1.2340607457359425

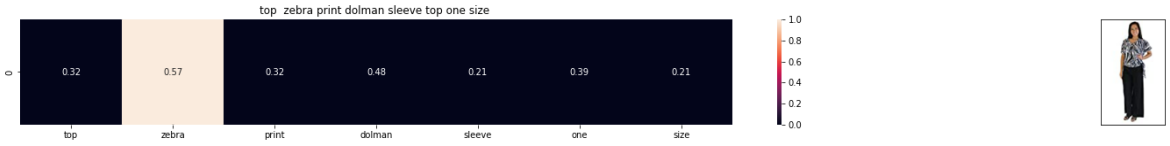
==========



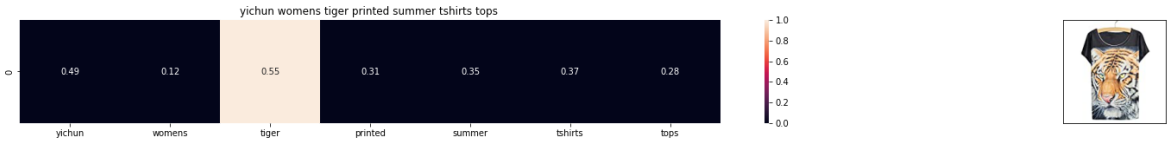
ASIN: B071ZDF6T2
Brand: Mossimo
Euclidean Distance from the given image: 1.2352785577664824
==========



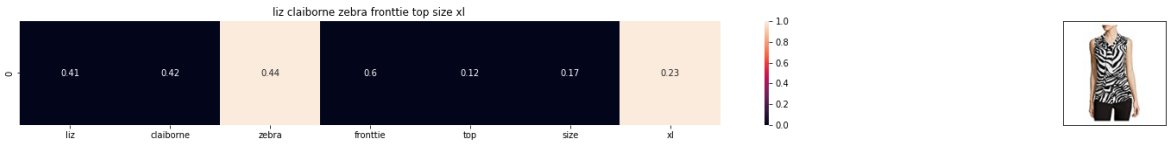
ASIN: B01K0H02OG
Brand: Tultex
Euclidean Distance from the given image: 1.236457298812782
==========



ASIN: B00H8A6ZLI
Brand: Vivian's Fashions
Euclidean Distance from the given image: 1.24996155052848
==========



ASIN: B010NN9RXO
Brand: YICHUN
Euclidean Distance from the given image: 1.2535461420856102
==========



ASIN: B06XBY5QXL
Brand: Liz Claiborne
Euclidean Distance from the given image: 1.2538832938357722
==========

IDF Based Product Similarity

In [ ]:

```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data['title'])
```

In [ ]:

```
def n_containing(word):
  """
  This function returns the number of documents(titles) containing the word
  """
  return sum(1 for blob in data['title'] if word in blob.split())
```

In [ ]:

```
def idf(word):
  return math.log(data.shape[0]/(n_containing(word)))
```

In [ ]:

```
idf_title_features = idf_title_features.astype(np.float)
```

In [ ]:

```
for i in tqdm(idf_title_vectorizer.vocabulary_.keys()):
  idf_val = idf(i)
  for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:
    idf_title_features[j, idf_title_vectorizer.vocabulary_[i]] = idf_val
```

```
100%|██████████| 12609/12609 [03:48<00:00, 55.29it/s]
```

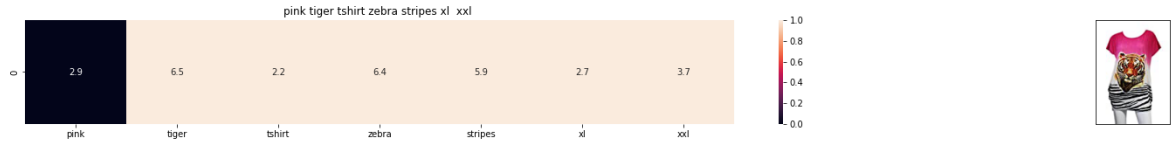
In [ ]:

```
def idf_model(doc_id, num_results):
  pairwise_dist = pairwise_distances(idf_title_features, idf_title_features[doc_id])
  indices = np.argsort(pairwise_dist.flatten())[:num_results]
  pdists = np.sort(pairwise_dist.flatten())[:num_results]
  df_indices = list(data.index[indices])
  for i in range(len(indices)):
    get_result(indices[i], data['title'].loc[df_indices[0]], data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indice
    print(f"ASIN: {data['asin'].loc[df_indices[i]]}")
    print(f"Brand: {data['brand'].loc[df_indices[i]]}")
    print(f"Euclidean Distance from the given image: {pdists[i]}")
    print('='*100)
```

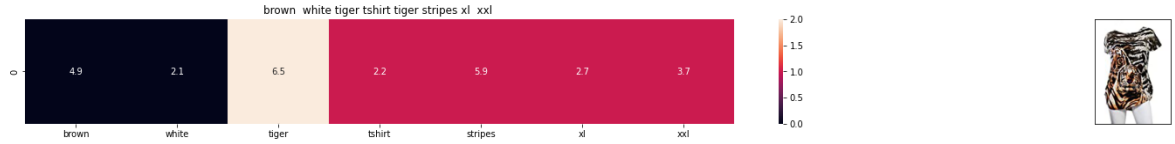
In [ ]:

```
idf_model(12566, 20)
```

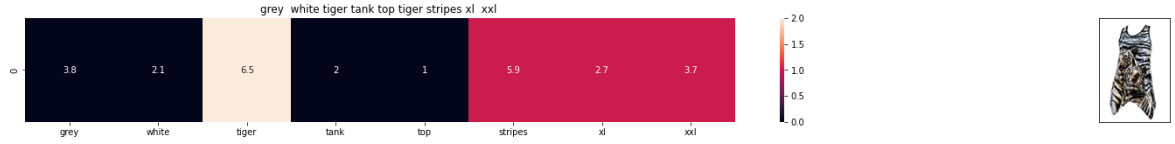
ASIN: B00JXQB5FQ
Brand: Si Row
Euclidean Distance from the given image: 0.0
==========



ASIN: B00JXQASS6
Brand: Si Row
Euclidean Distance from the given image: 12.20507131122177
==========

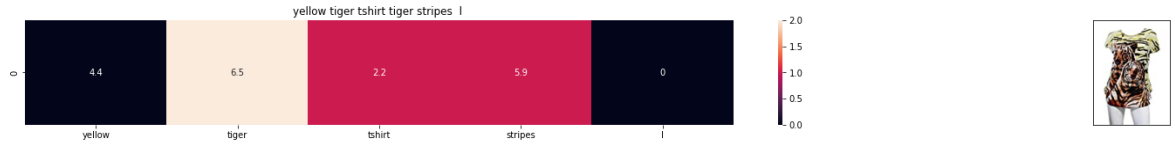


ASIN: B00JXQCWTO
Brand: Si Row
Euclidean Distance from the given image: 14.468362685603465
==========

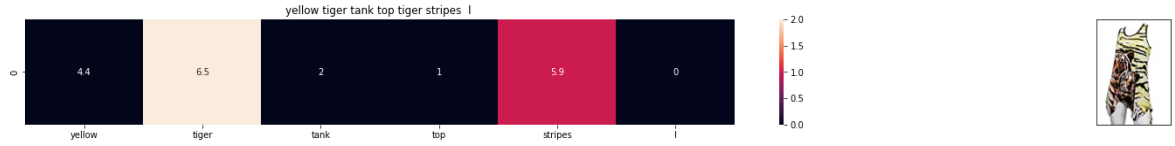


ASIN: B00JXQAFZ2
Brand: Si Row
Euclidean Distance from the given image: 14.486832924778964
==========

ASIN: B00JXQAO94
Brand: Si Row
Euclidean Distance from the given image: 14.833392966672909
==========



ASIN: B00JXQCUIC
Brand: Si Row
Euclidean Distance from the given image: 14.898744516719225
==========



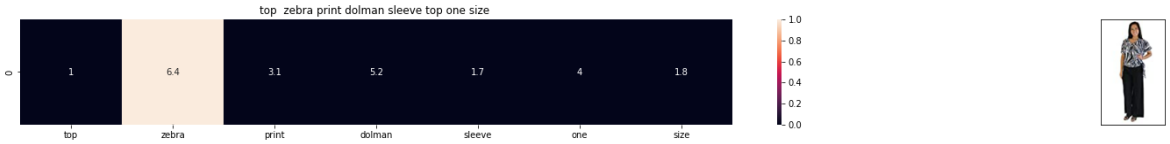
ASIN: B00JXQAUWA
Brand: Si Row
Euclidean Distance from the given image: 15.224458287343769
==========

ASIN: B074T8ZYGX
Brand: MKP Crop Top
Euclidean Distance from the given image: 17.080812955631995
==========

ASIN: B00KF2N5PU
Brand: Vietsbay
Euclidean Distance from the given image: 17.090168125645416
==========

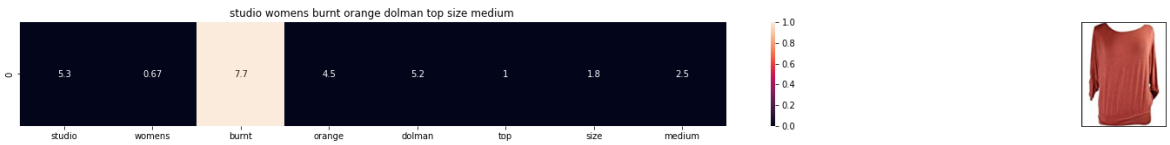
ASIN: B00JPOZ9GM
Brand: Sofra
Euclidean Distance from the given image: 17.153215337562703
==========

ASIN: B074T9KG9Q
Brand: Rain
Euclidean Distance from the given image: 17.33671523874989
==========



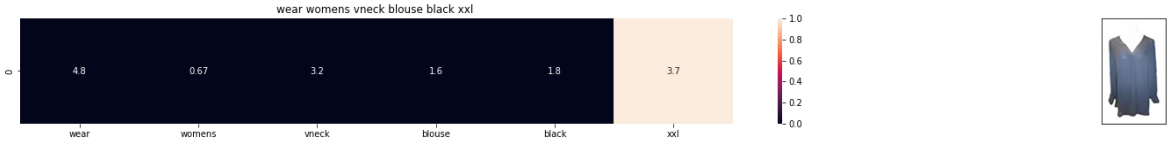
ASIN: B00H8A6ZLI
Brand: Vivian's Fashions
Euclidean Distance from the given image: 17.410075941001253
==========

ASIN: B074G5G5RK
Brand: ERMANNO SCERVINO
Euclidean Distance from the given image: 17.539921335459557
==========



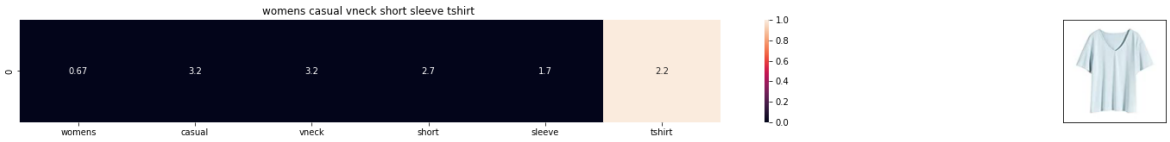
ASIN: B06XSCVFT5
Brand: Studio M

Euclidean Distance from the given image: 17.61275854366134
==========



ASIN: B06Y6FH453
Brand: Who What Wear
Euclidean Distance from the given image: 17.623745282500135

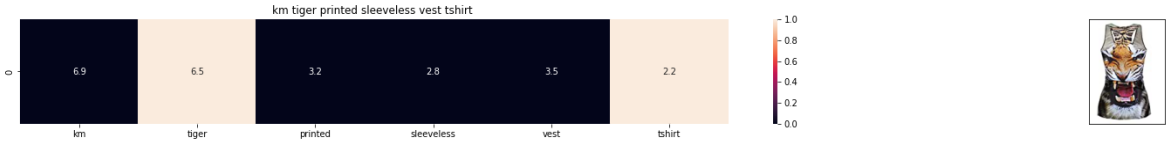
==========



ASIN: B074V45DCX
Brand: Rain
Euclidean Distance from the given image: 17.634342496835046
==========

ASIN: B07583CQFT
Brand: Very J
Euclidean Distance from the given image: 17.63753712743611
==========

ASIN: B073GJGVBN
Brand: Ivan Levi
Euclidean Distance from the given image: 17.7230738913371
==========



ASIN: B012VQLT6Y
Brand: KM T-shirt
Euclidean Distance from the given image: 17.762588561202364
==========

ASIN: B00ZZMYBRG
Brand: HP-LEISURE
Euclidean Distance from the given image: 17.779536864674238
==========

Text Semantics Based Product Similarity

In [ ]:

```
with open('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/Applied_AI_Workshop_Code_Data/word2vec_model', 'r
  model = pickle.load(handle)
```

In [ ]:

```
def get_word_vec(sentence, doc_id, m_name):
  vec = []
  for i in sentence.split():
    if i in vocab:
      if m_name == "weighted" and i in idf_title_vectorizer.vocabulary_:
        vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[i]] * model[i])
      elif m_name == 'avg':
        vec.append(model[i])
    else:
      vec.append(np.zeros(shape=(300,)))
  return np.array(vec)
```

In [ ]:

```
def get_distance(vec1, vec2):
  final_dist = []
  for i in vec1:
    dist = []
    for j in vec2:
      dist.append(np.linalg.norm(i-j))
    final_dist.append(np.array(dist))
  return np.array(final_dist)
```

In [ ]:

```
def heat_map_w2v(sentence1, sentence2, url, doc_id1, doc_id2, model):
  s1_vec = get_word_vec(sentence1, doc_id1, model)
  s2_vec = get_word_vec(sentence2, doc_id2, model)
  s1_s2_dist = get_distance(s1_vec, s2_vec)
  gs = gridspec.GridSpec(2,2, width_ratios = [4,1], height_ratios = [2,1])
  fig = plt.figure(figsize = (15,15))
  ax = plt.subplot(gs[0])
  ax = sns.heatmap(np.round(s1_s2_dist, 4), annot = True)
  ax.set_xticklabels(sentence2.split())
  ax.set_yticklabels(sentence1.split())
  ax.set_title(sentence2)
  ax = plt.subplot(gs[1])
  ax.grid(False)
  ax.set_xticks([])
  ax.set_yticks([])
  display_img(url, ax, fig)
  plt.show()
```

In [ ]:

```
vocab = model.keys()

def build_avg_vec(sentence, num_features, doc_id, m_name):
  featureVec = np.zeros((num_features,), dtype="float32")
  nwords = 0
  for word in sentence.split():
    nwords += 1
    if word in vocab:
      if m_name == "weighted" and word in idf_title_vectorizer.vocabulary_:
        featureVec = np.add(featureVec, idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[word]] * model[word])
      elif m_name == "avg":
        featureVec = np.add(featureVec, model[word])
  if nwords > 0:
    featureVec = np.divide(featureVec, nwords)
  return featureVec
```

Average Word2Vec

In [ ]:

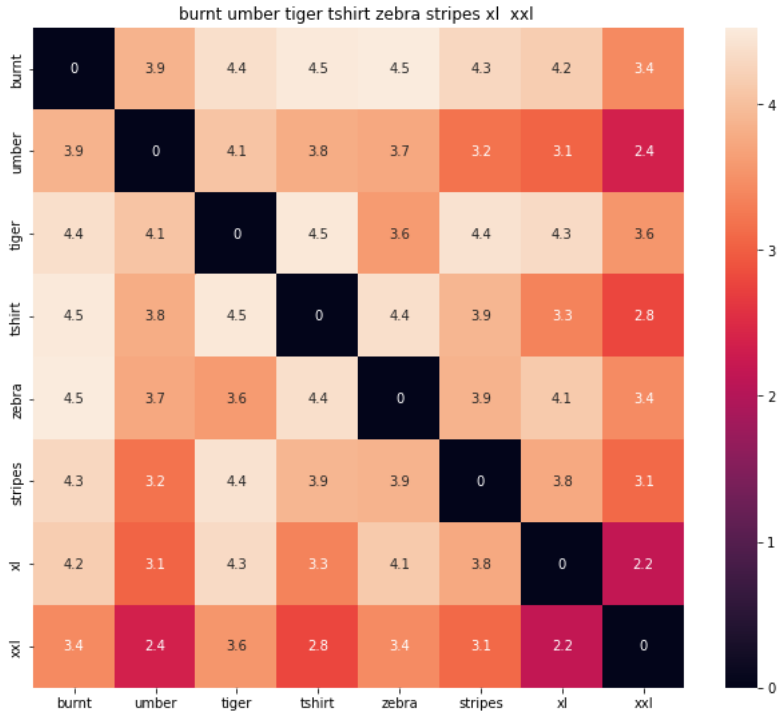
```
doc_id = 0
w2v_title = []

for i in data['title']:
  w2v_title.append(build_avg_vec(i, 300, doc_id, "avg"))
  doc_id += 1

w2v_title = np.array(w2v_title)
```

In [ ]:

```
def avg_w2v_model(doc_id, num_results):
  pairwise_dist = pairwise_distances(w2v_title, w2v_title[doc_id].reshape(1, -1))
  indices = np.argsort(pairwise_dist.flatten())[:num_results]
  pdists  = np.sort(pairwise_dist.flatten())[0:num_results]
  df_indices = list(data.index[indices])
  for i in range(len(indices)):
    heat_map_w2v(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indi
    print('ASIN :',data['asin'].loc[df_indices[i]])
    print('Brand :',data['brand'].loc[df_indices[i]])
    print ('Euclidean Distance from given input image :', pdists[i])
    print('='*125)
```

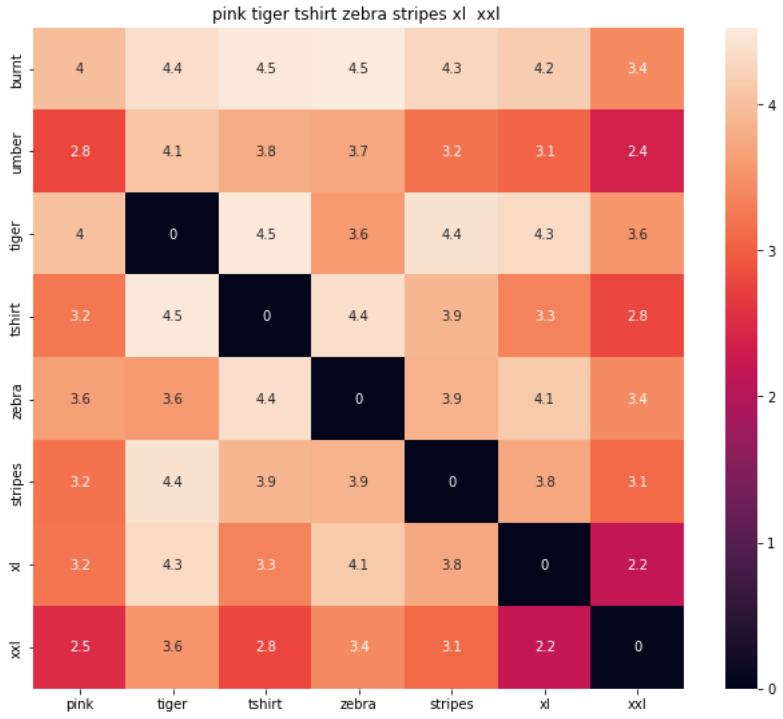
In [ ]:

```
avg_w2v_model(12566, 20)
```



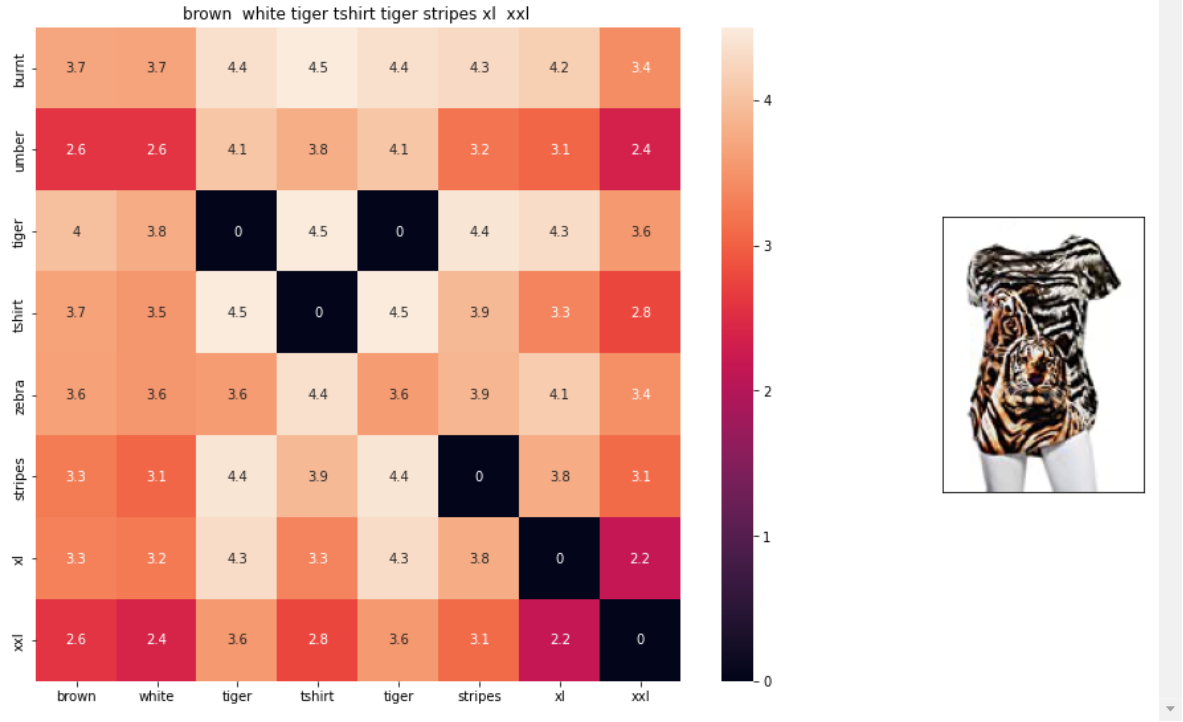
ASIN : B00JXQB5FQ
Brand : Si Row
Euclidean Distance from given input image : 0.0

==========



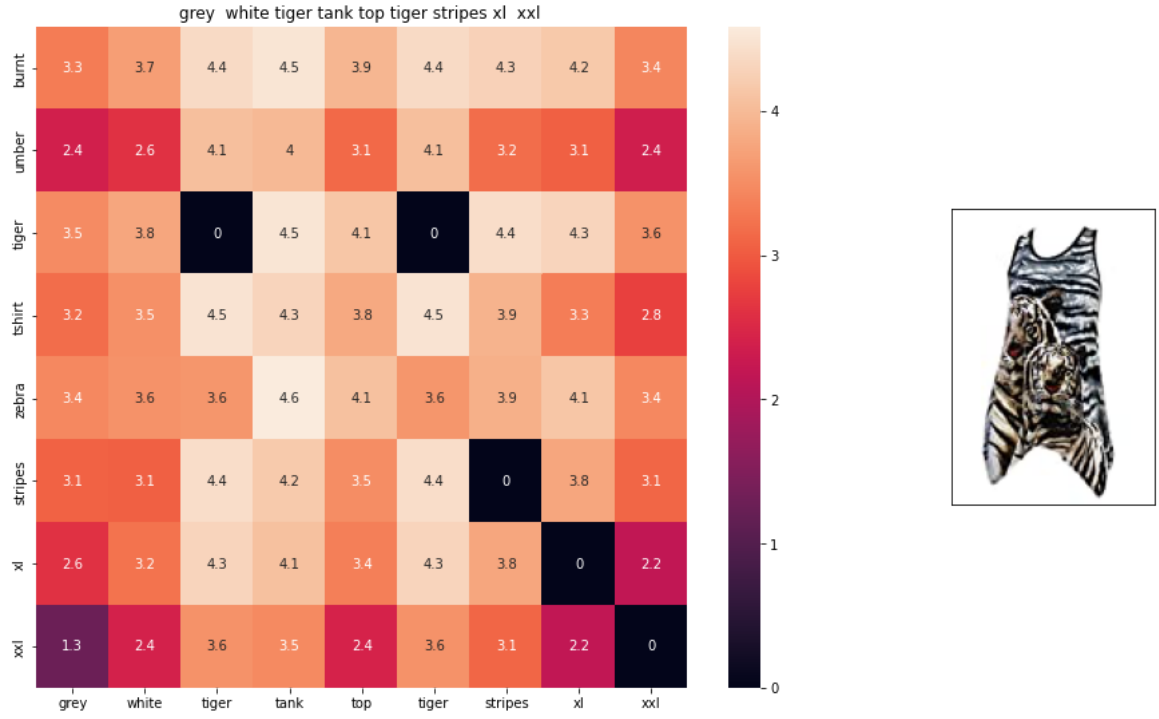
ASIN : B00JXQASS6
Brand : Si Row
Euclidean Distance from given input image : 0.5891926

==========



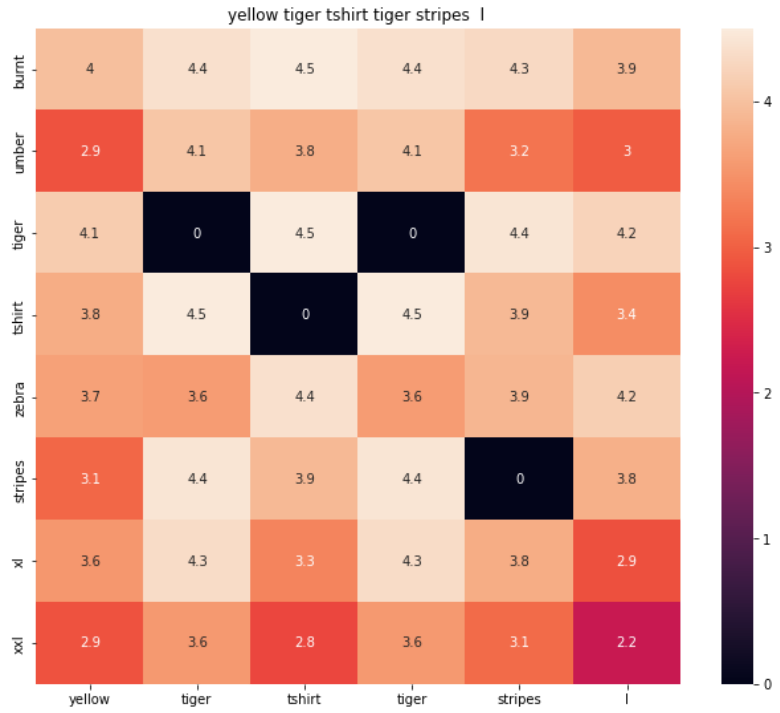
ASIN : B00JXQCWTO
Brand : Si Row
Euclidean Distance from given input image : 0.7003438

==========

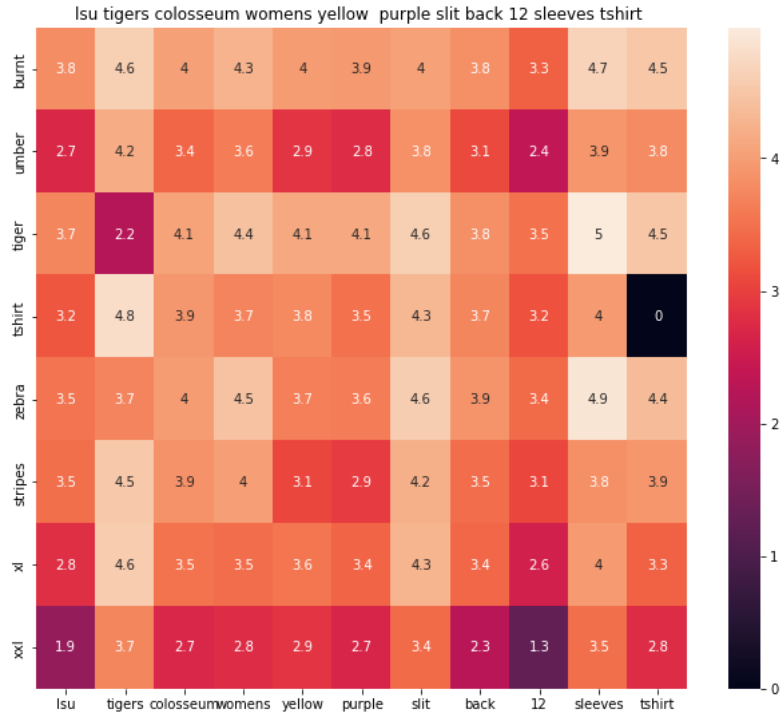


ASIN : B00JXQAFZ2
Brand : Si Row
Euclidean Distance from given input image : 0.89283955

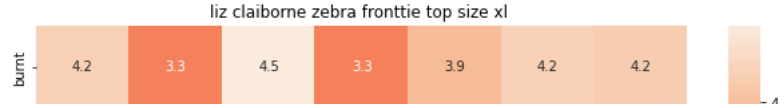
==========

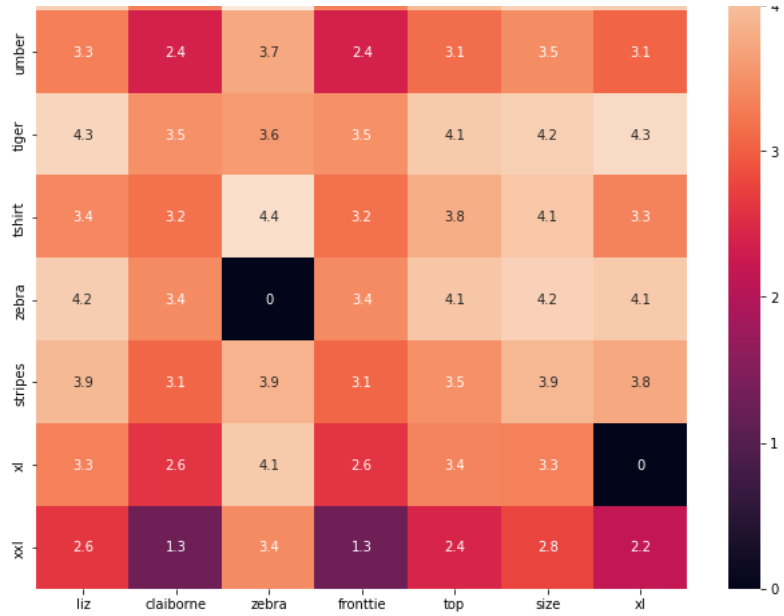


ASIN : B00JXQCUIC
Brand : Si Row
Euclidean Distance from given input image : 0.95601255



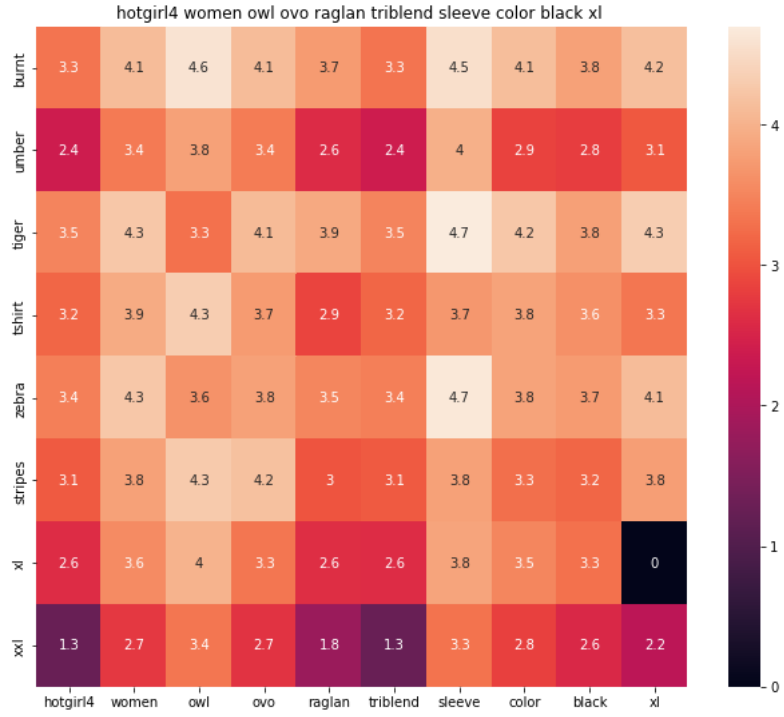
ASIN : B073R5Q8HD
Brand : Colosseum
Euclidean Distance from given input image : 1.022969





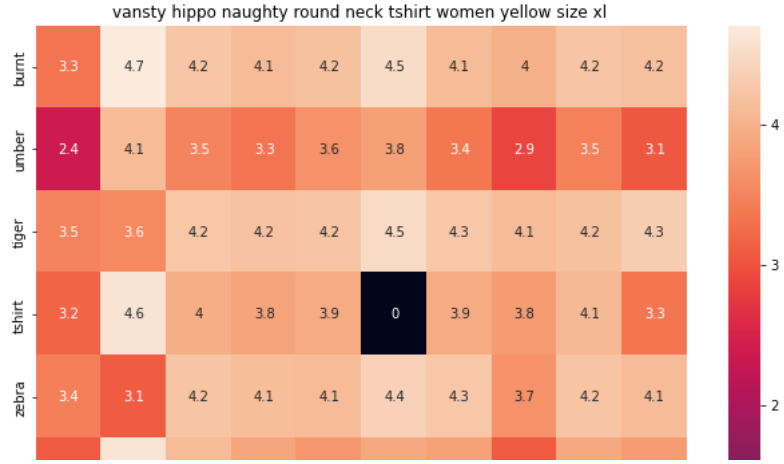
ASIN : B06XBY5QXL
Brand : Liz Claiborne
Euclidean Distance from given input image : 1.0669324

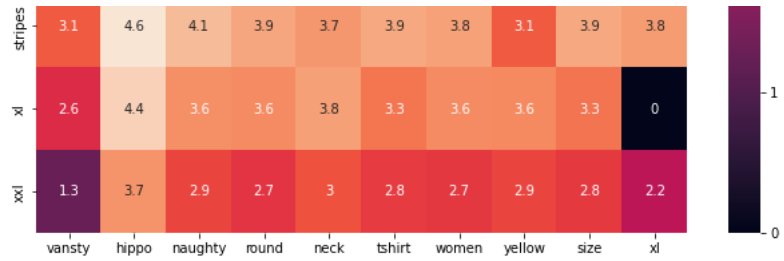
==========



ASIN : B01L8L73M2
Brand : Hotgirl4 Raglan Design
Euclidean Distance from given input image : 1.0731405

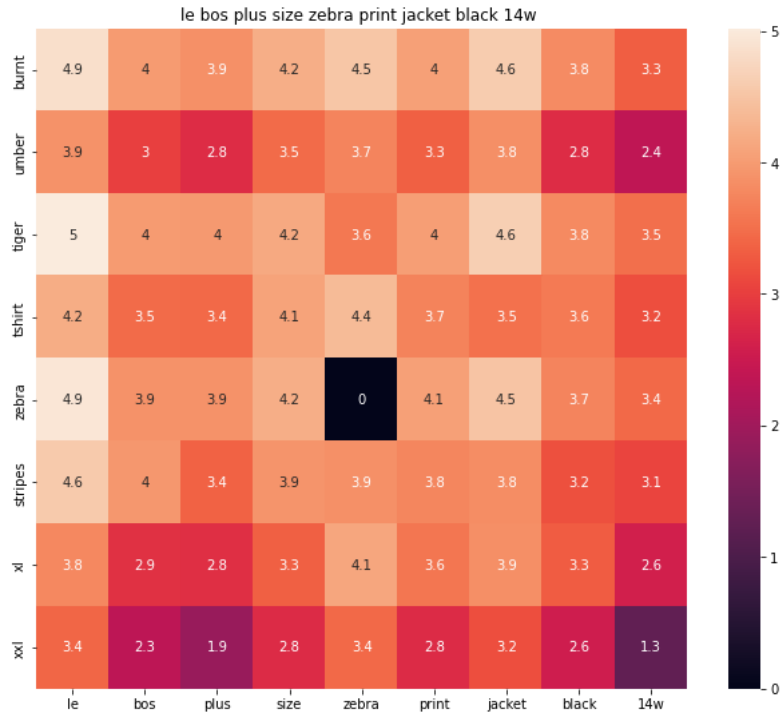
==========





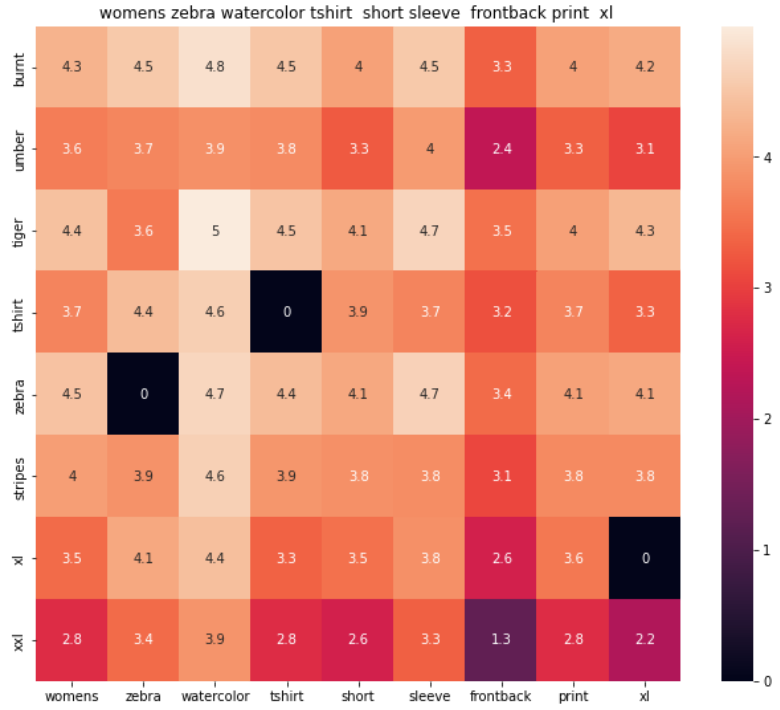
ASIN : B01EJS5H06
Brand : Vansty
Euclidean Distance from given input image : 1.075719

==========

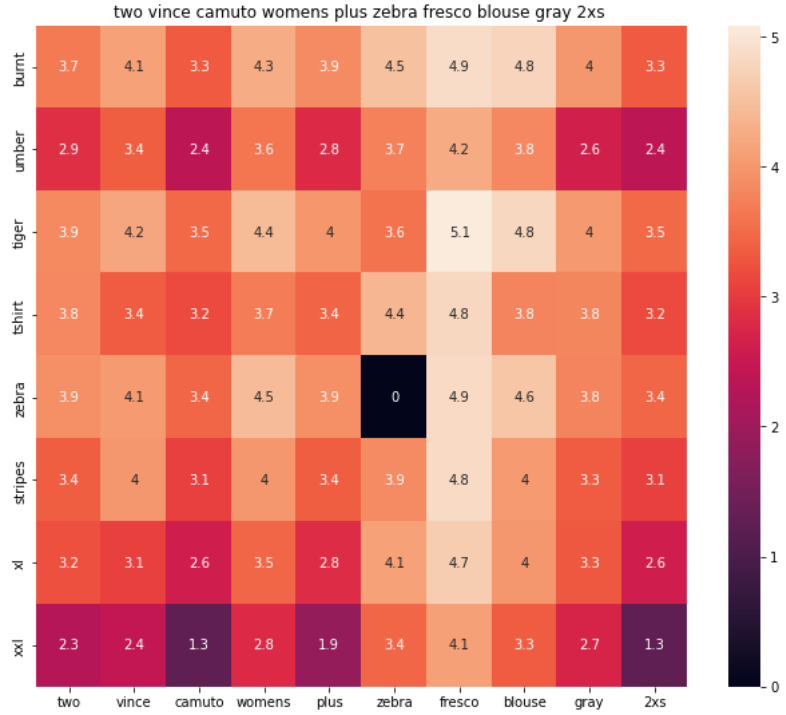


ASIN : B01BO1XRK8
Brand : Le Bos
Euclidean Distance from given input image : 1.0839964

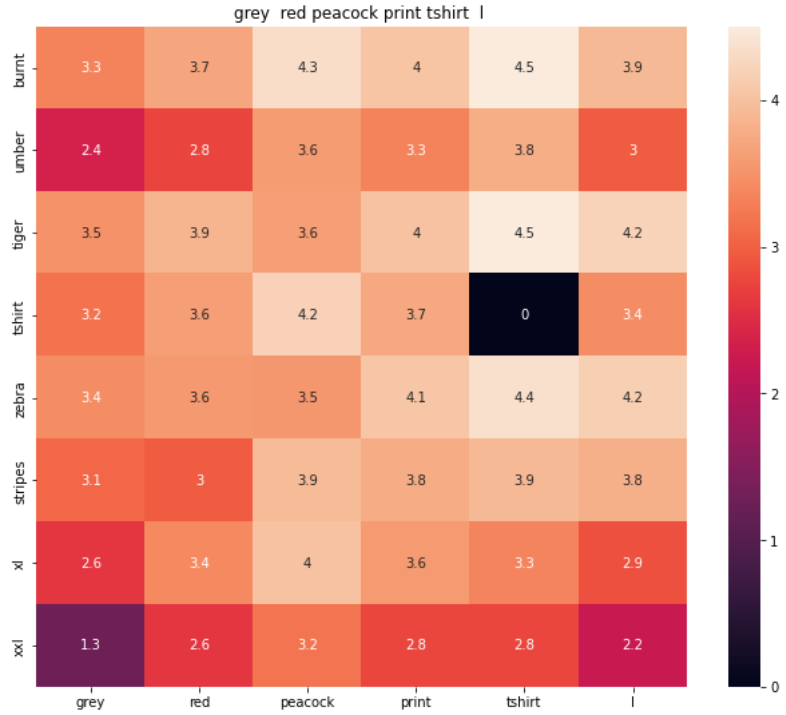
==========



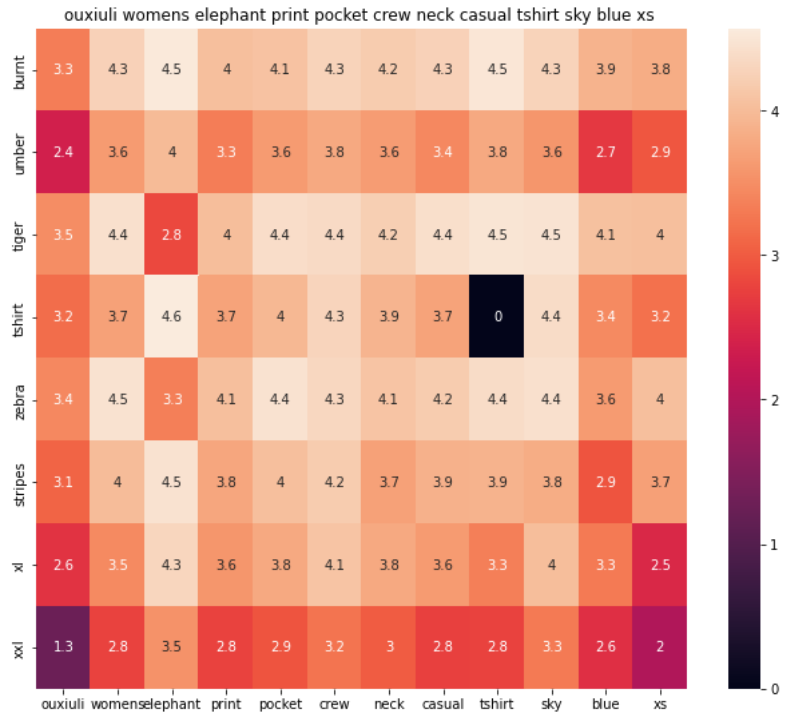
ASIN : B072R2JXKW
Brand : WHAT ON EARTH
Euclidean Distance from given input image : 1.0842218



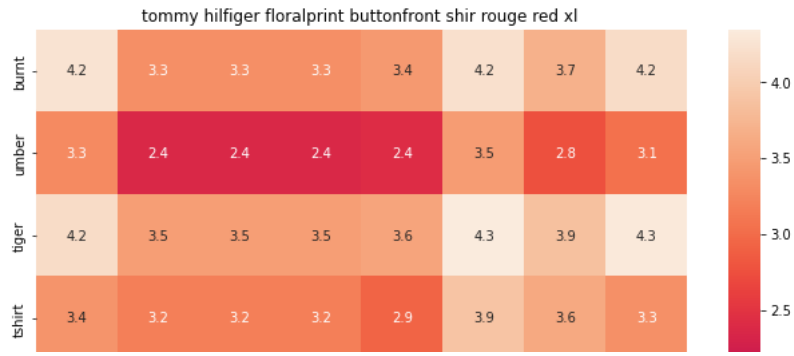
ASIN : B074MJRGW6
Brand : Two by Vince Camuto
Euclidean Distance from given input image : 1.0895038

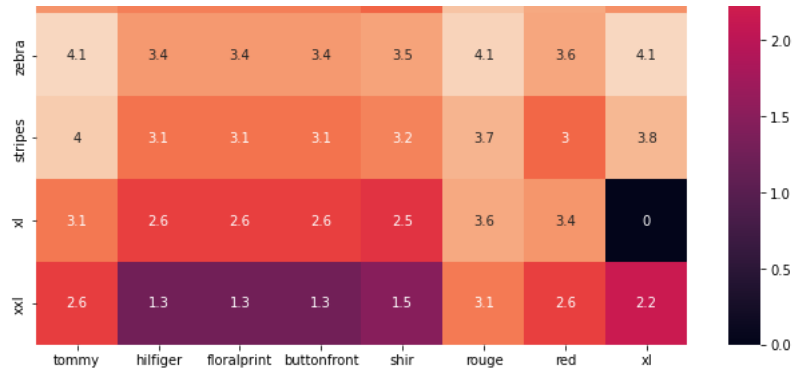


ASIN : B00JXQCFRS
Brand : Si Row
Euclidean Distance from given input image : 1.0900588



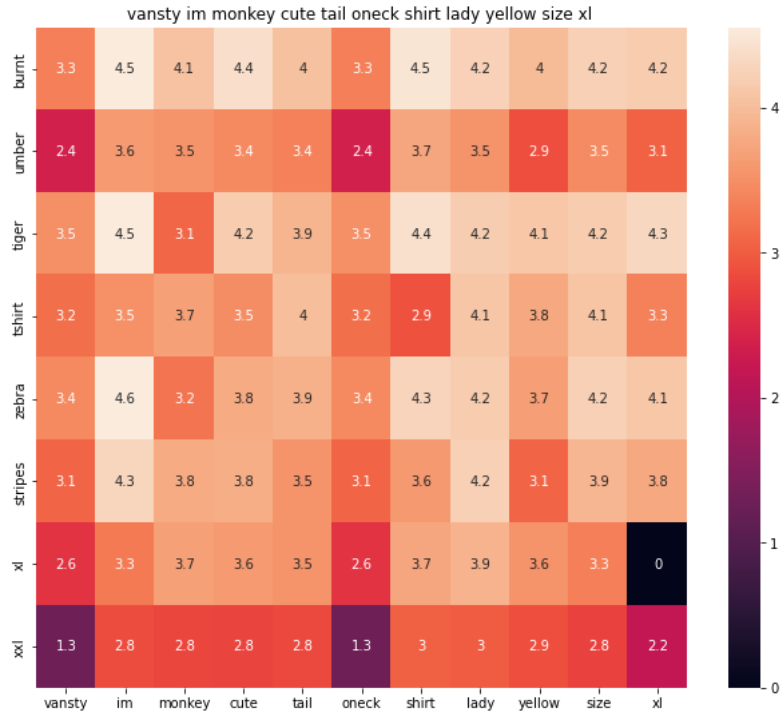
ASIN : B01I53HU6K
Brand : ouxiuli
Euclidean Distance from given input image : 1.0920111





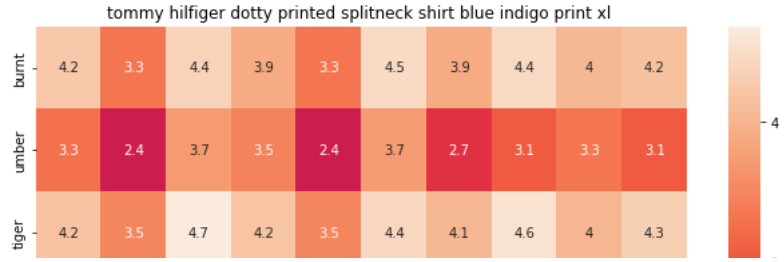
ASIN : B0711NGTQM
Brand : THILFIGER RTW
Euclidean Distance from given input image : 1.0923415

==========



ASIN : B01EFSLO8Y
Brand : Vansty
Euclidean Distance from given input image : 1.0934004

==========

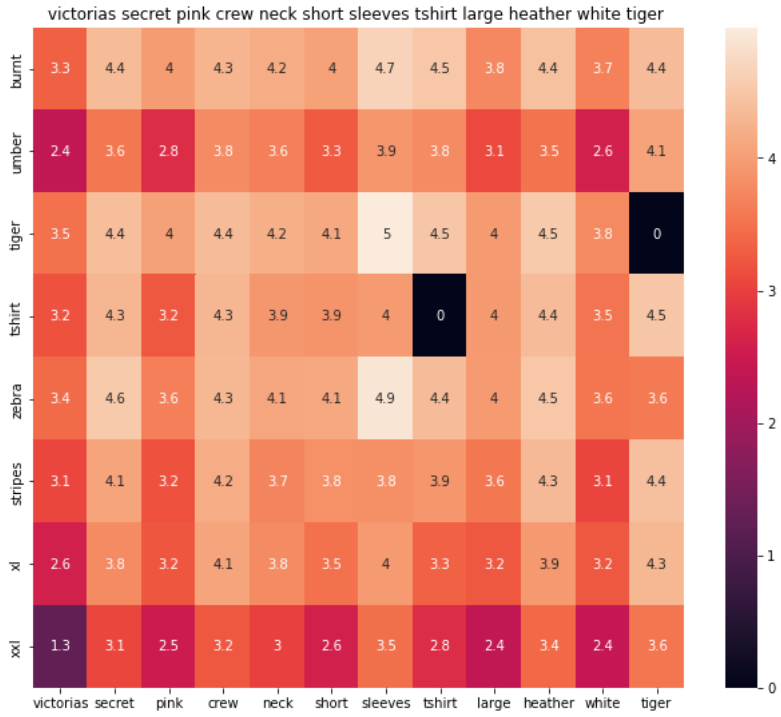


ASIN : B0716TVWQ4

Brand : THILFIGER RTW

Euclidean Distance from given input image : 1.0942024

==========

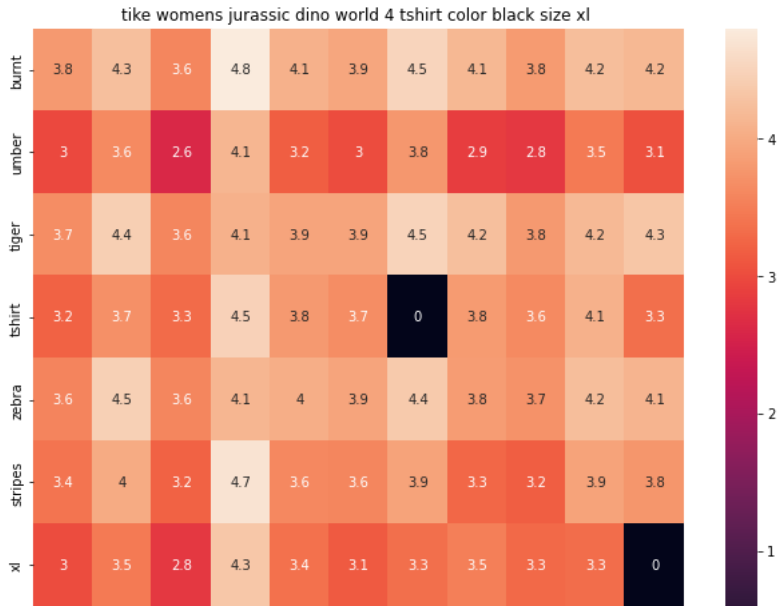


ASIN : B0716MVPGV

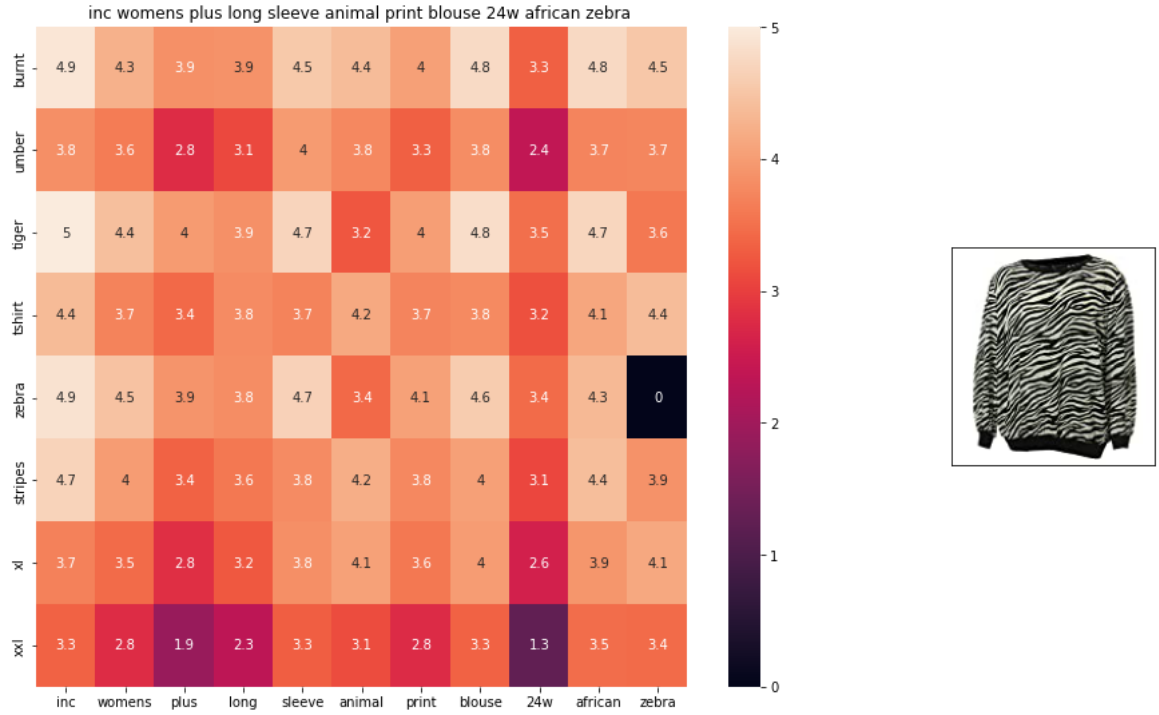
Brand : V.Secret

Euclidean Distance from given input image : 1.0948304

==========



ASIN : B016OPN4OI
Brand : TIKE Fashions
Euclidean Distance from given input image : 1.0951275



ASIN : B018WDJCUA
Brand : INC - International Concepts Woman
Euclidean Distance from given input image : 1.0966892

IDF Weighted Word2Vec

In [ ]:

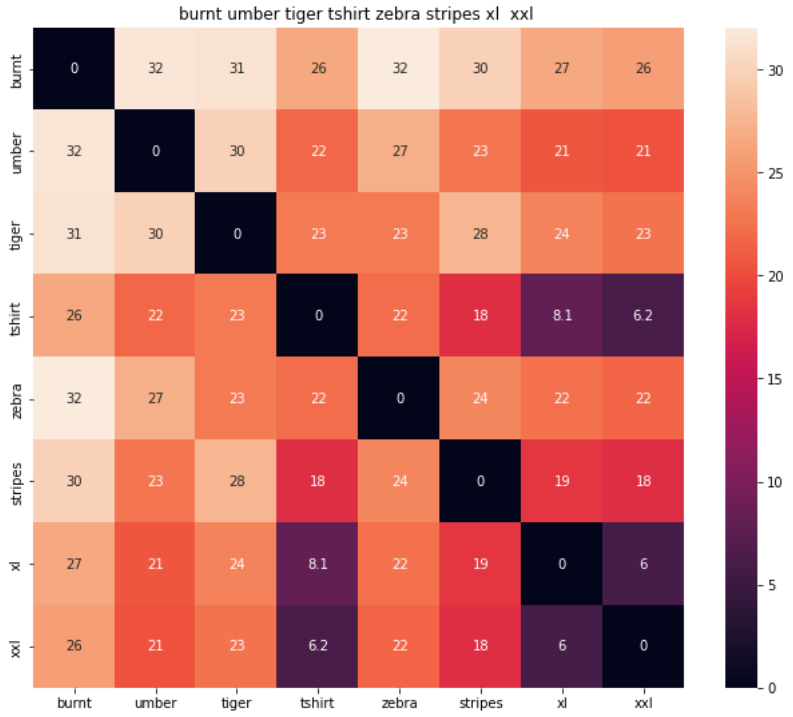
```
doc_id = 0
w2v_title_weight = []
for i in data['title']:
  w2v_title_weight.append(build_avg_vec(i, 300, doc_id, 'weighted'))
  doc_id += 1
w2v_title_weight = np.array(w2v_title_weight)
```

In [ ]:

```
def weighted_w2v_model(doc_id, num_results):
  pairwise_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
  indices = np.argsort(pairwise_dist.flatten())[:num_results]
  pdists = np.sort(pairwise_dist.flatten())[:num_results]
  df_indices = list(data.index[indices])
  for i in range(len(indices)):
    heat_map_w2v(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indi
    print('ASIN :',data['asin'].loc[df_indices[i]])
    print('Brand :',data['brand'].loc[df_indices[i]])
    print('euclidean distance from input :', pdists[i])
    print('='*125)
```

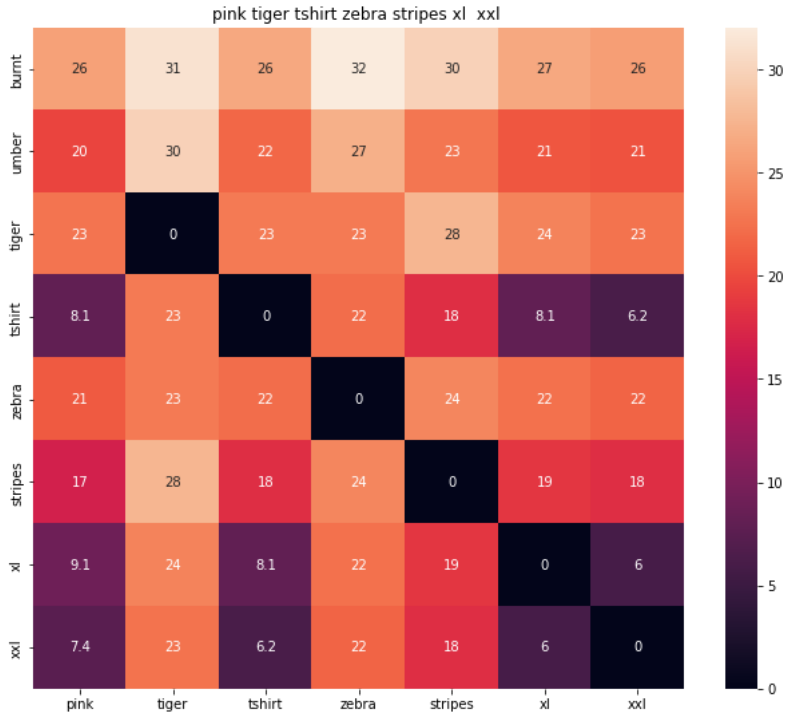
In [ ]:

```
weighted_w2v_model(12566, 20)
```



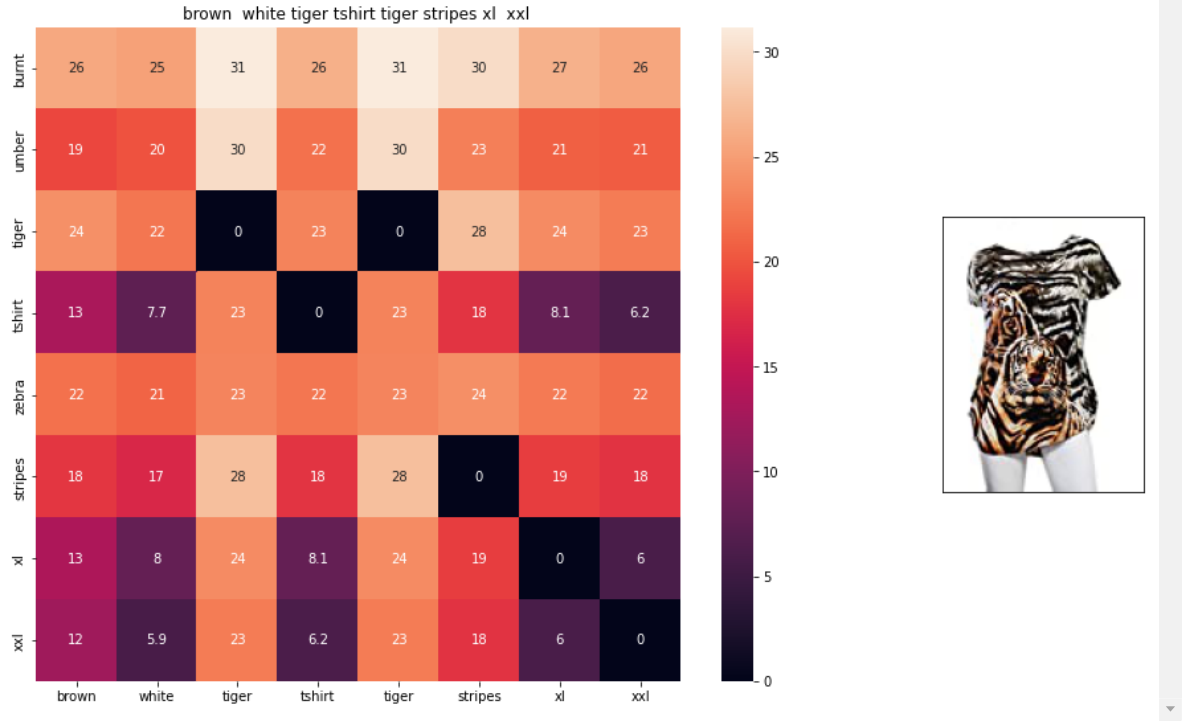
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0

==========



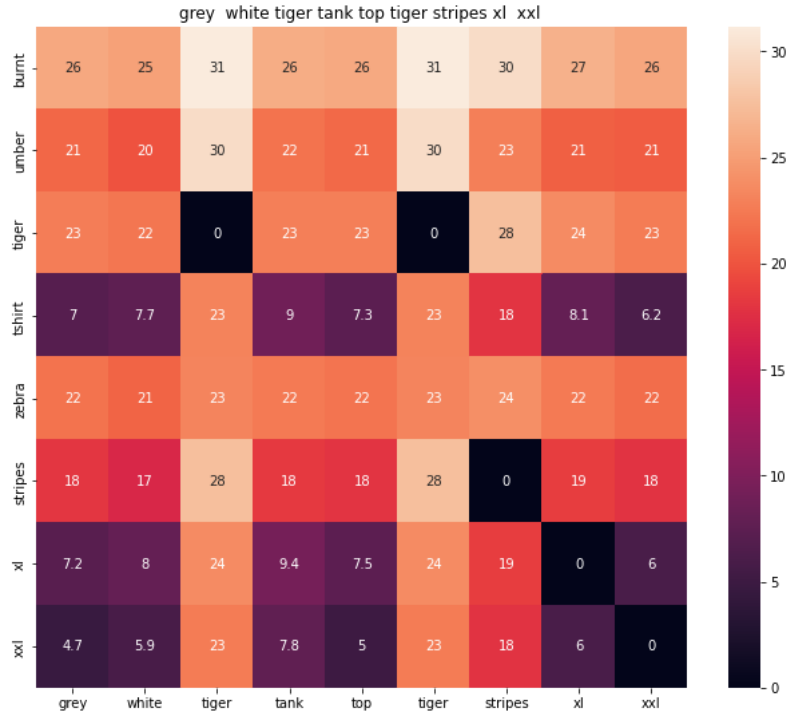
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 4.0638866

==========



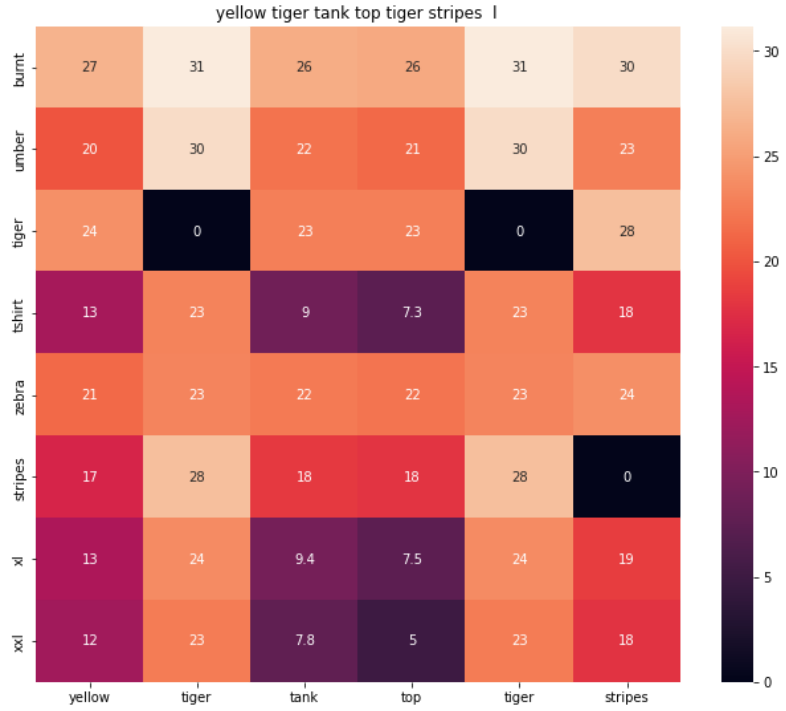
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 4.7709413

==========

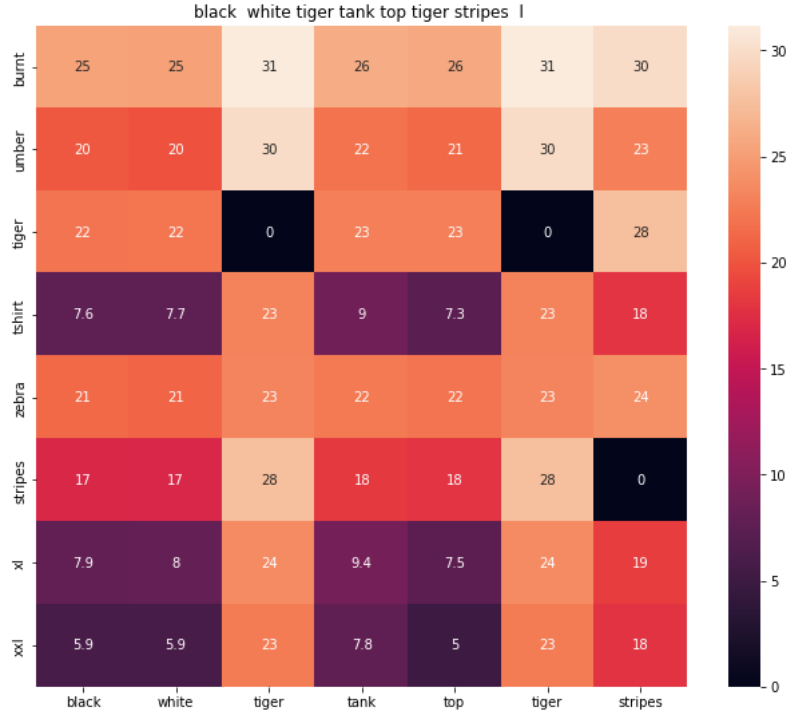


ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 5.3601604

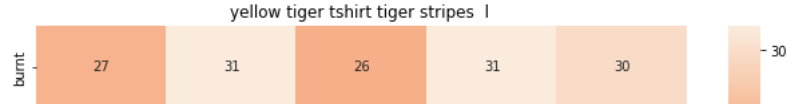
==========

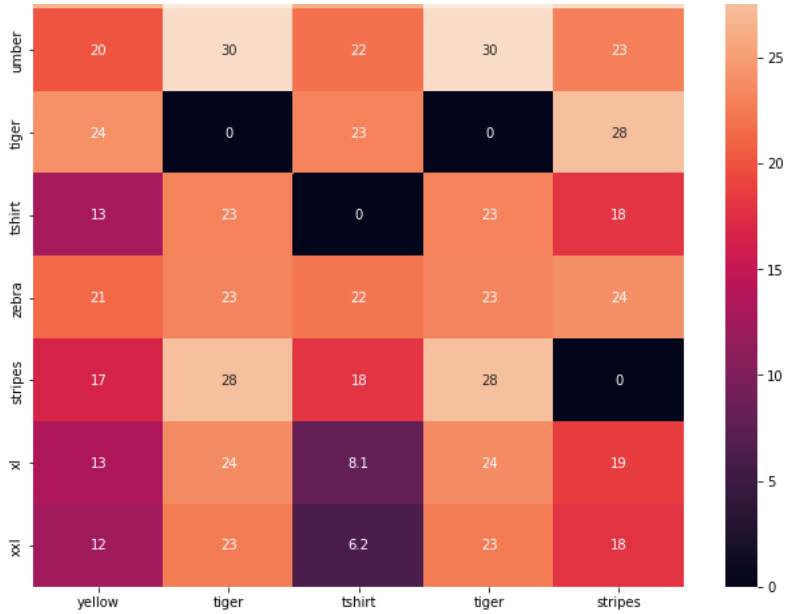


ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 5.6895227

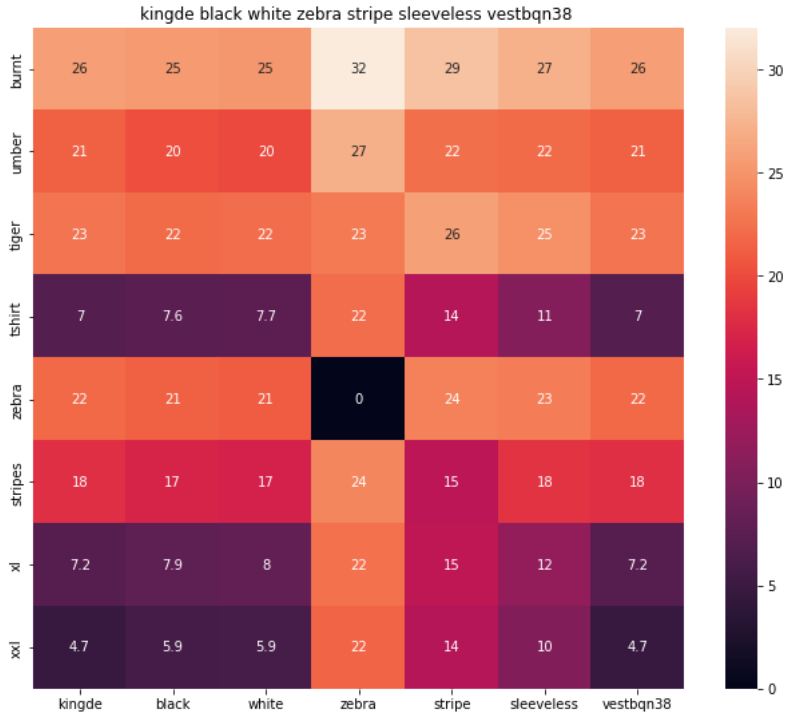


ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 5.693021

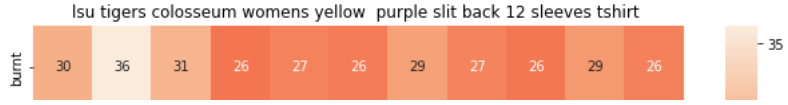


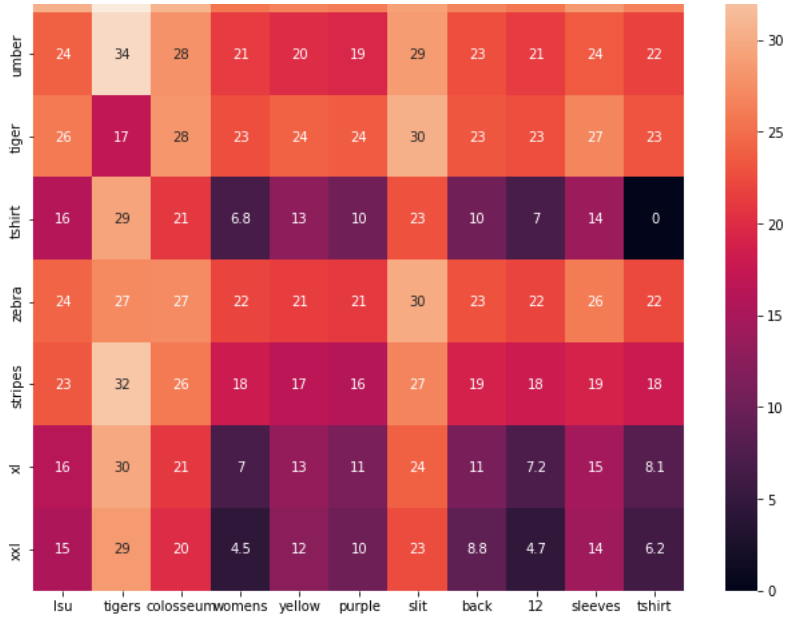


ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 5.893442

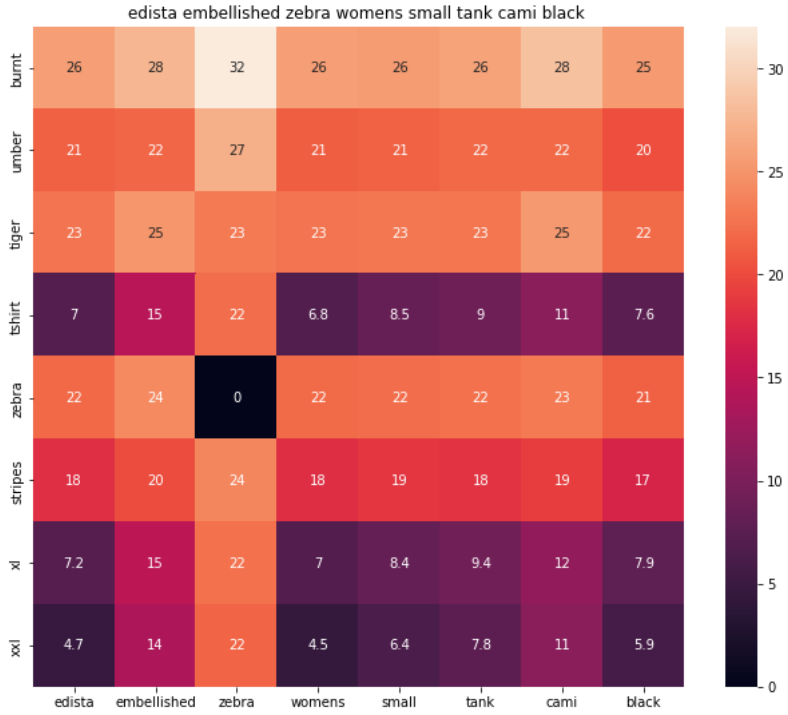


ASIN : B015H41F6G
Brand : KINGDE
euclidean distance from input : 6.1329894

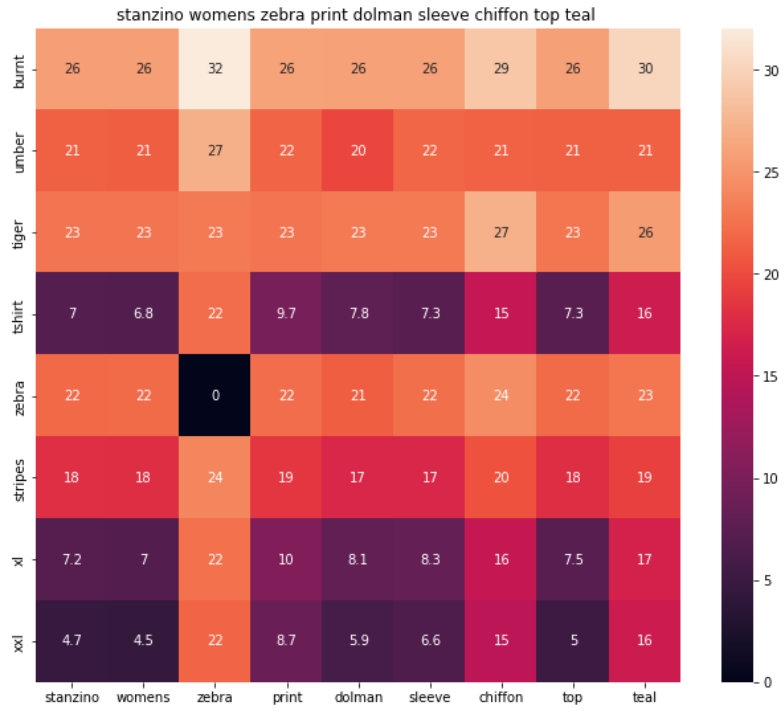




ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 6.2567053
==========

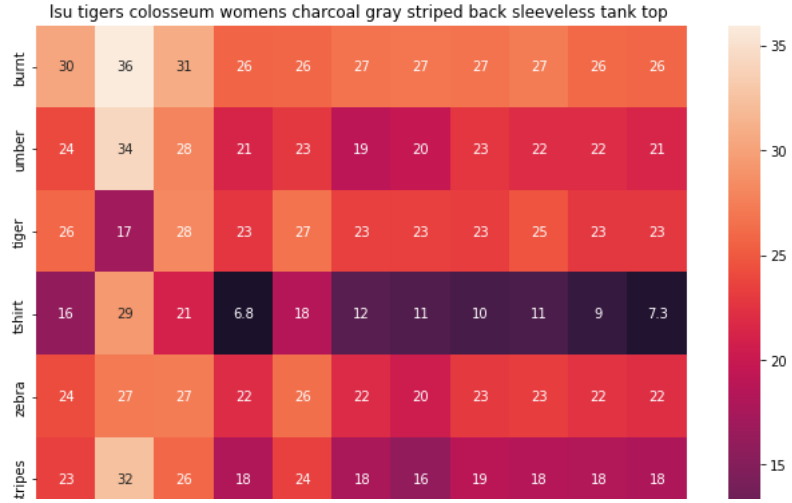


ASIN : B074P8MD22
Brand : Edista
euclidean distance from input : 6.3922033
==========

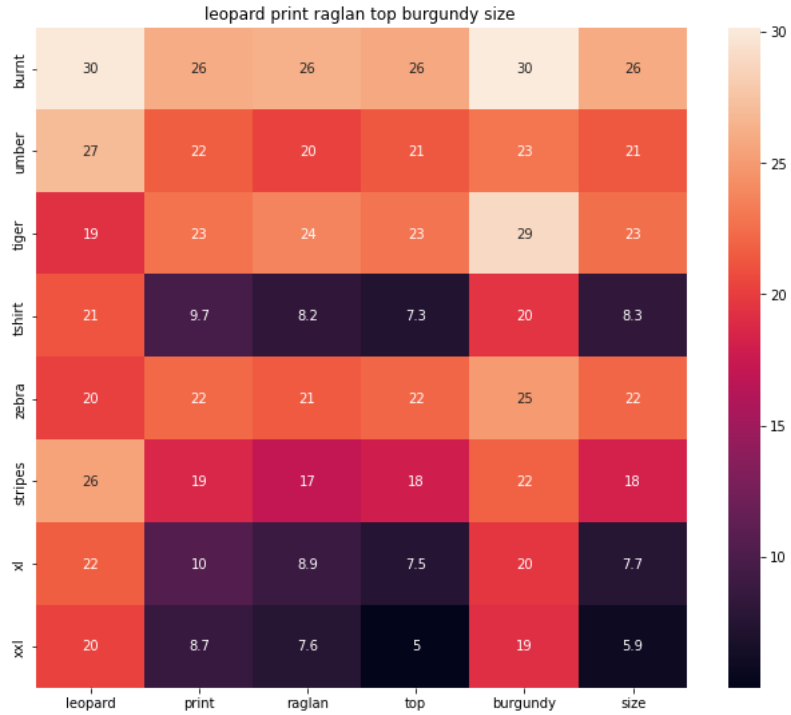


ASIN : B00C0I3U3E
Brand : Stanzino
euclidean distance from input : 6.4149003

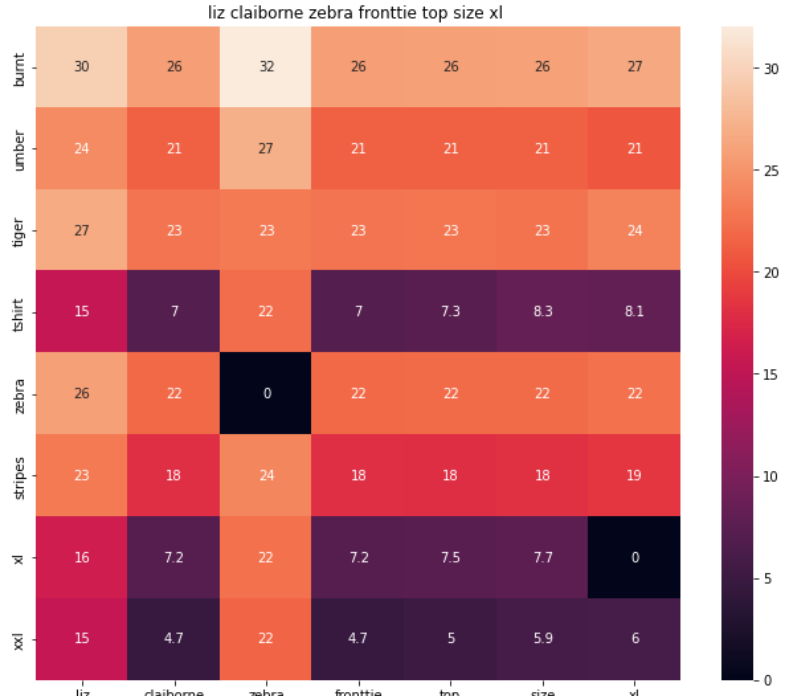
==========



ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 6.450959

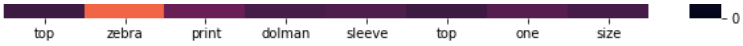


ASIN : B01C6ORLDQ
Brand : 1 Mad Fit
euclidean distance from input : 6.463409

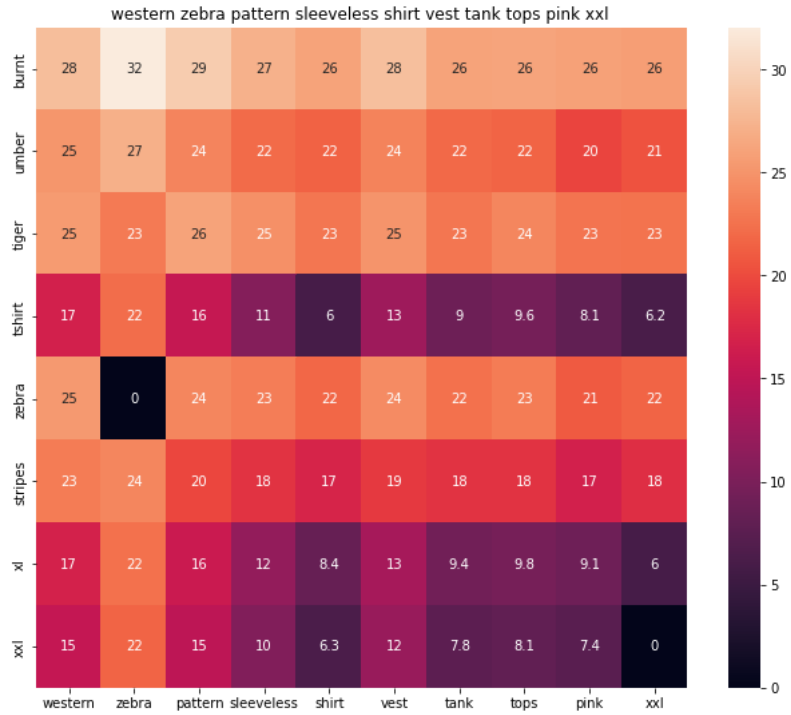


ASIN : B06XBY5QXL
Brand : Liz Claiborne
euclidean distance from input : 6.5392227

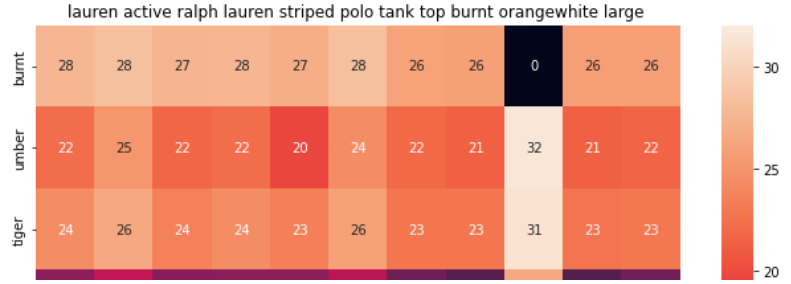
ASIN : B071YF3WDD
Brand : Merona
euclidean distance from input : 6.5755024

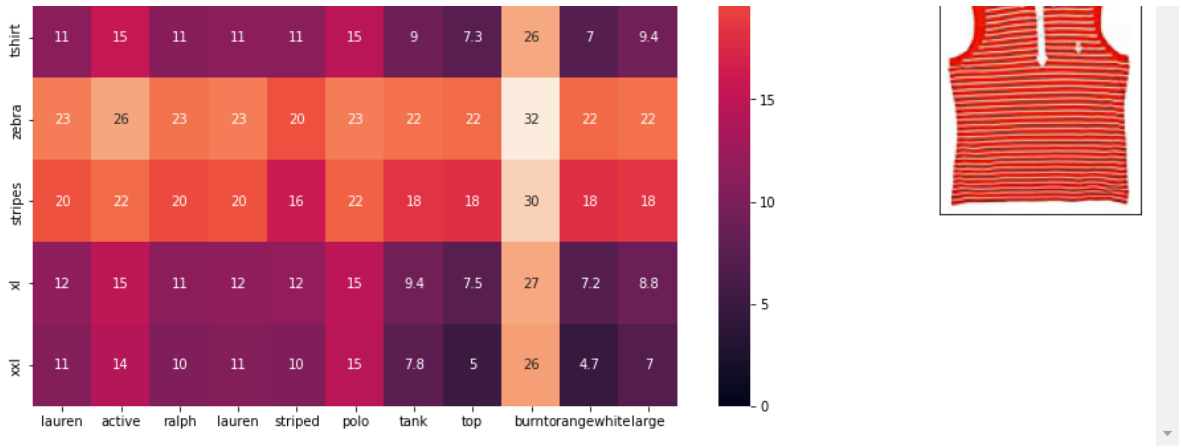


ASIN : B00H8A6ZLI
Brand : Vivian's Fashions
euclidean distance from input : 6.6382146

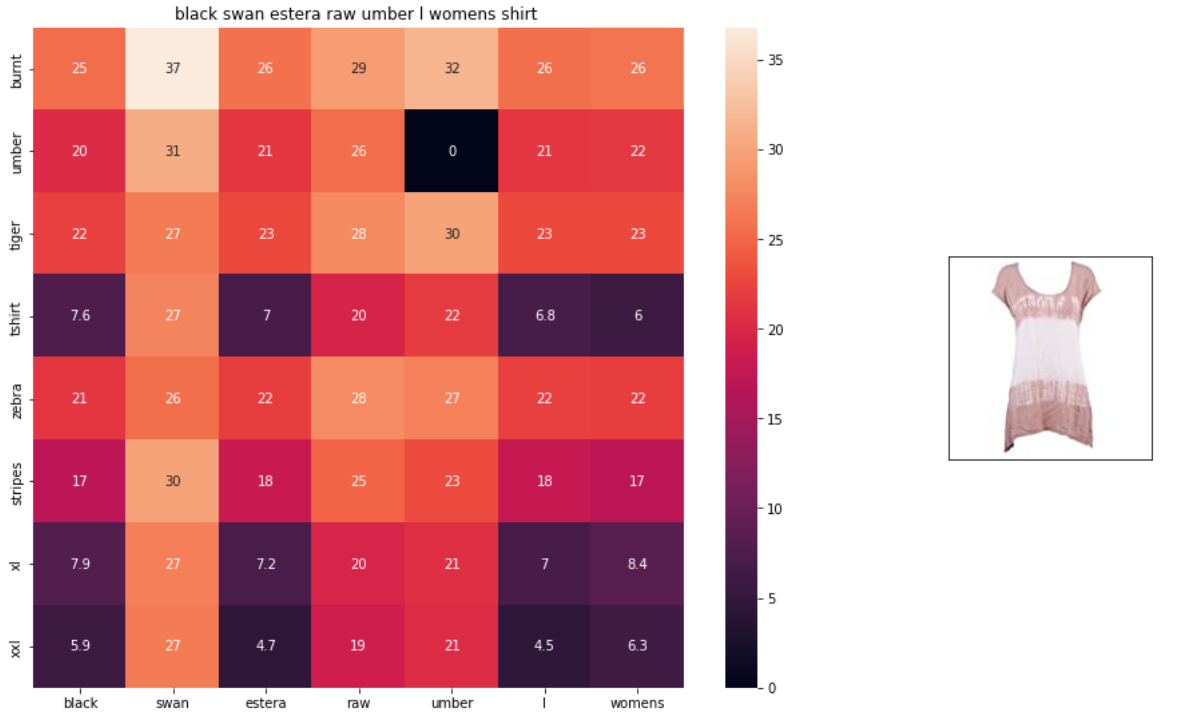


ASIN : B00Z6HEXWI
Brand : Black Temptation
euclidean distance from input : 6.6607366

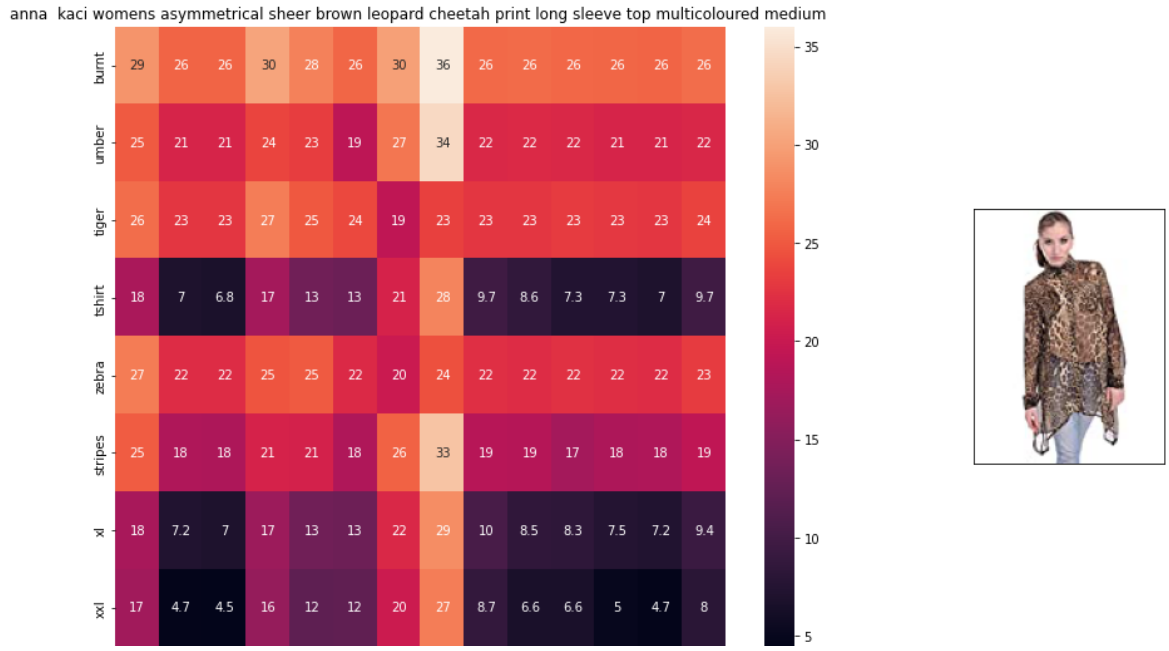


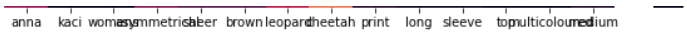


ASIN : B00ILGH5OY
Brand : Ralph Lauren Active
euclidean distance from input : 6.6839046



ASIN : B06Y1VN8WQ
Brand : Black Swan
euclidean distance from input : 6.705763






```
ASIN : B00KSNTY7Y
Brand : Anna-Kaci
euclidean distance from input : 6.7061243
```

```
=========================================================================================================================
```

Weighted Similarity Using Title, Brand and Color

In [ ]:

```
data['brand'].fillna(value = "Not Given", inplace =True)

brands = [x.replace(" ", "-") for x in data['brand'].values]
types = [x.replace(" ","-") for x in data['product_type_name'].values]
colors = [x.replace(" ","-") for x in data['color'].values]

brand_vectorizer = CountVectorizer()
brand_features = brand_vectorizer.fit_transform(brands)

type_vectorizer = CountVectorizer()
type_features = type_vectorizer.fit_transform(types)

color_vectorizer = CountVectorizer()
color_features = color_vectorizer.fit_transform(colors)

extra_features = hstack((brand_features, type_features, color_features)).tocsr()
```

In [ ]:

```
def heatmap_w2v_brand(sentence1, sentence2, url, doc_id1, doc_id2, df_id1, df_id2, model):
  s1_vec = get_word_vec(sentence1, doc_id1, model)
  s2_vec = get_word_vec(sentence2, doc_id2, model)
  s1_s2_dist = get_distance(s1_vec, s2_vec)
  data_matrix = [["ASIN", "Brand", "Color", "Product Type"],
                 [data['asin'].loc[df_id1], brands[doc_id1], colors[doc_id1], types[doc_id1]],
                 [data['asin'].loc[df_id2], brands[doc_id2], colors[doc_id2], types[doc_id2]]]
  colorscale = [[0, "#1D004D"], [.5, "#F2E5FF"], [1, "#F2E5D1"]]
  table = ff.create_table(data_matrix, index=True, colorscale = colorscale)
  plotly.offline.iplot(table, filename = "simple_table")
  gs = gridspec.GridSpec(25,15)
  fig = plt.figure(figsize = (25,5))
  ax1 = plt.subplot(gs[:, :-5])
  ax1 = sns.heatmap(np.round(s1_s2_dist, 6), annot = True)
  ax1.set_xticklabels(sentence2.split())
  ax1.set_yticklabels(sentence1.split())
  ax1.set_title(sentence2)
  ax2 = plt.subplot(gs[:,10:16])
  ax2.grid(False)
  ax2.set_xticks([])
  ax2.set_yticks([])
  display_img(url, ax2, fig)
  plt.show()
```

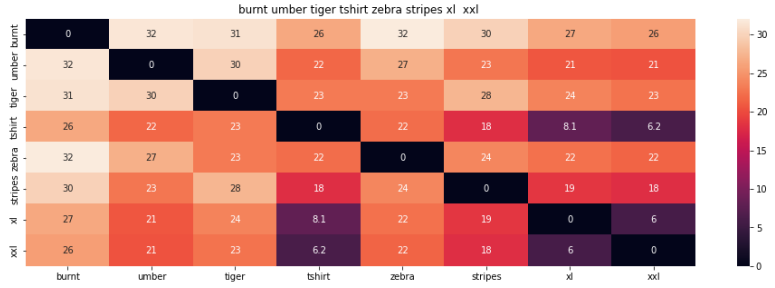
In [ ]:

```
def idf_w2v_brand(doc_id, w1, w2, num_results):
  idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
  ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
  pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist)/float(w1 + w2)
  indices = np.argsort(pairwise_dist.flatten())[0:num_results]
  pdists  = np.sort(pairwise_dist.flatten())[0:num_results]
  df_indices = list(data.index[indices])
  for i in range(0, len(indices)):
      heatmap_w2v_brand(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]], data['medium_image_url'].loc[df_indices[i]
      print('ASIN :',data['asin'].loc[df_indices[i]])
      print('Brand :',data['brand'].loc[df_indices[i]])
      print('euclidean distance from input :', pdists[i])
      print('='*125)
```

In [ ]:

```
idf_w2v_brand(12566, 5, 5, 20)
```

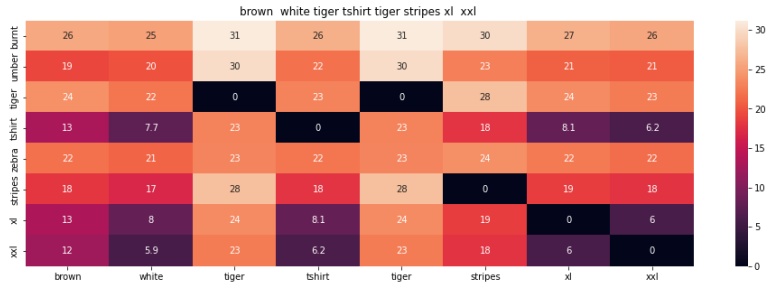
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES



```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0
```

```
==========================================================================================================================
```

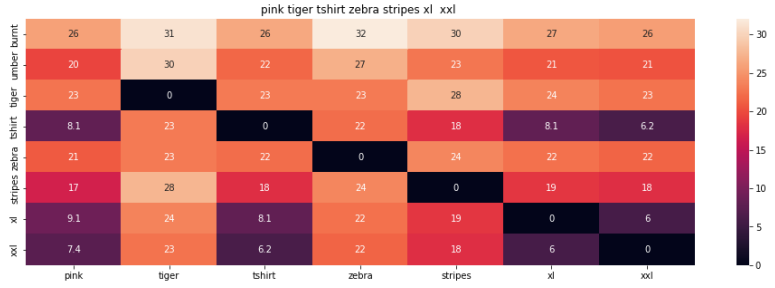
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCWTO	Si-Row	Brown	TOYS_AND_GAMES



```
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 2.3854705810546877
```

```
==========================================================================================================================
```

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQASS6	Si-Row	Pink	TOYS_AND_GAMES

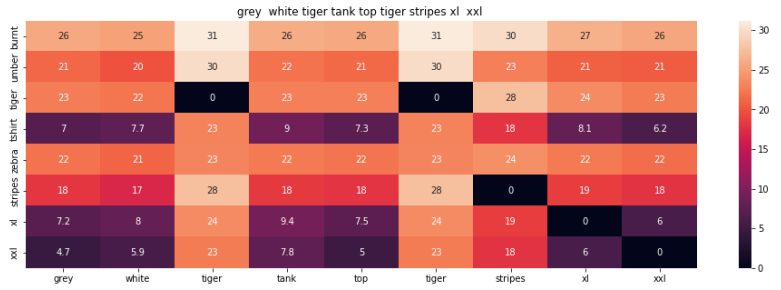


```
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 2.739050102414575
```

```
==========================================================================================================================
```

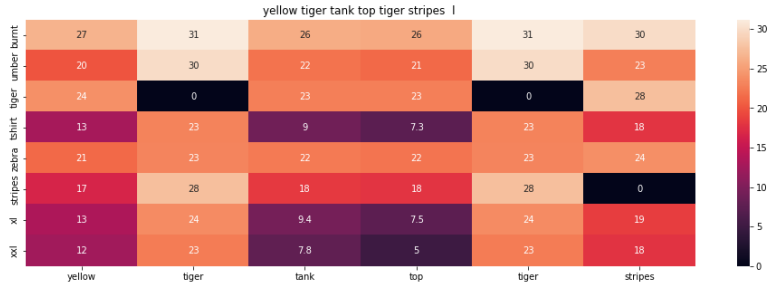
ASIN	Brand	Color	Product Type

B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAFZ2	Si-Row	Grey	TOYS_AND_GAMES



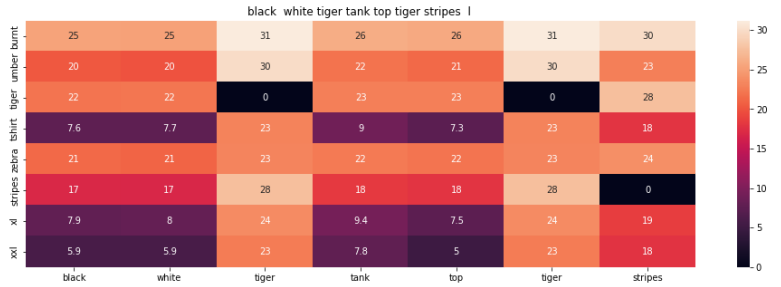
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 3.387187004270044

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAUWA	Si-Row	Yellow	TOYS_AND_GAMES



ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 3.5518680574316646

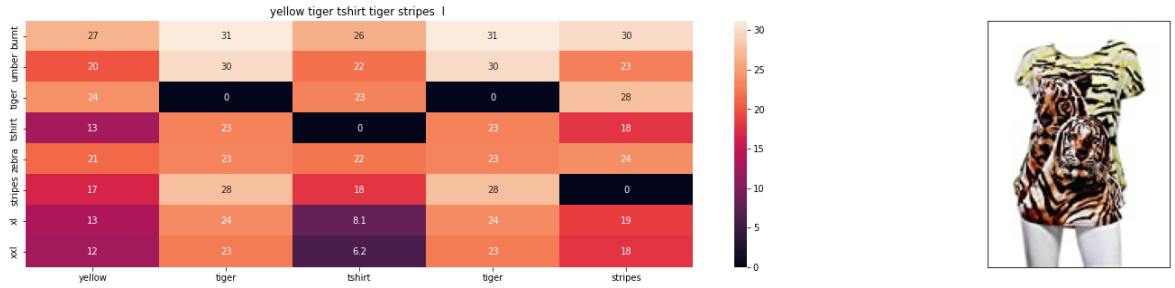
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAO94	Si-Row	White	TOYS_AND_GAMES



ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 3.5536170961279536

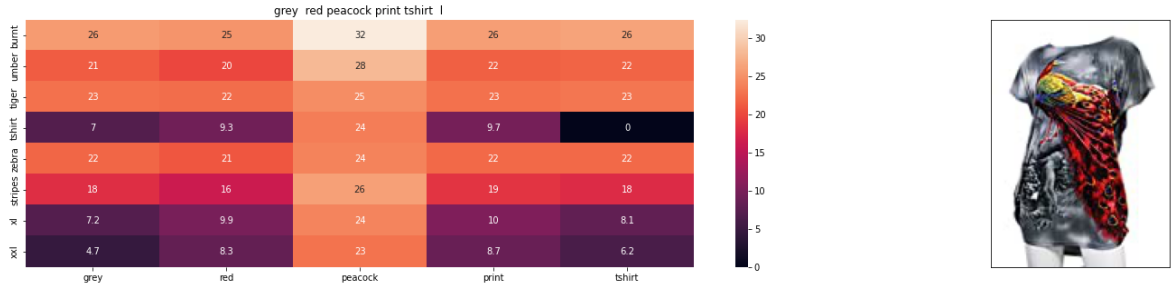
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES

B00JXQCUIC	Si-Row	Yellow	TOYS_AND_GAMES



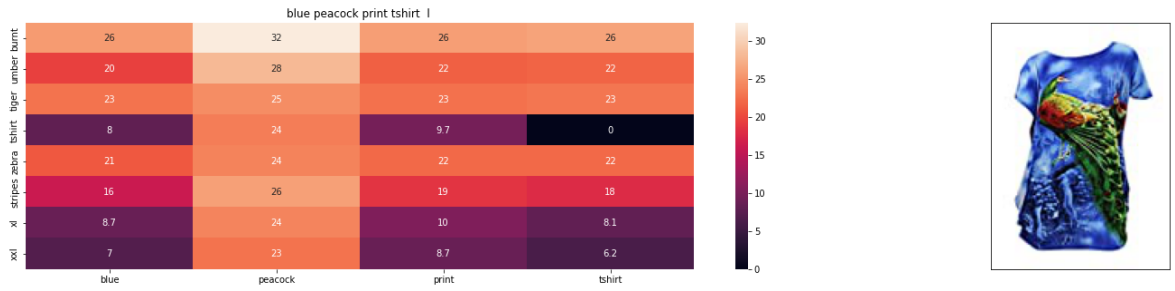
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 3.6538278581518795
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCFRS	Si-Row	Grey	TOYS_AND_GAMES



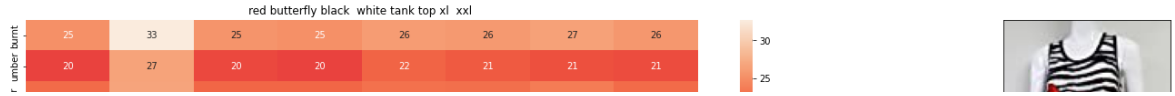
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 4.128811264218774
==========

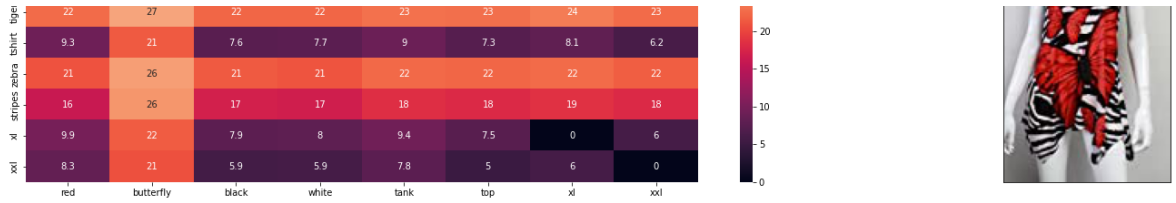
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQC8L6	Si-Row	Blue	TOYS_AND_GAMES



ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 4.203900146665063
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63CW2	Si-Row	Red	TOYS_AND_GAMES

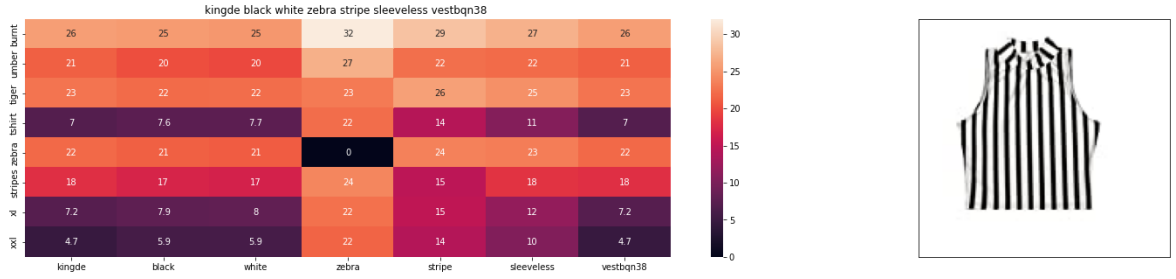




ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 4.286586380185571

==========

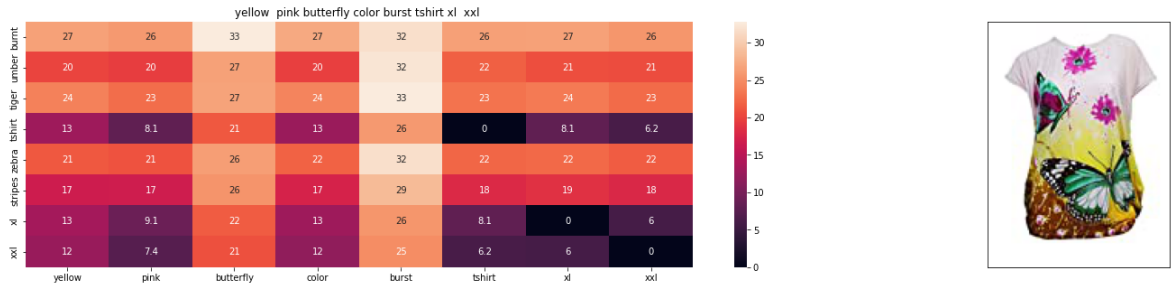
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B015H41F6G	KINGDE	White	SHIRT



ASIN : B015H41F6G
Brand : KINGDE
euclidean distance from input : 4.389370406508858

==========

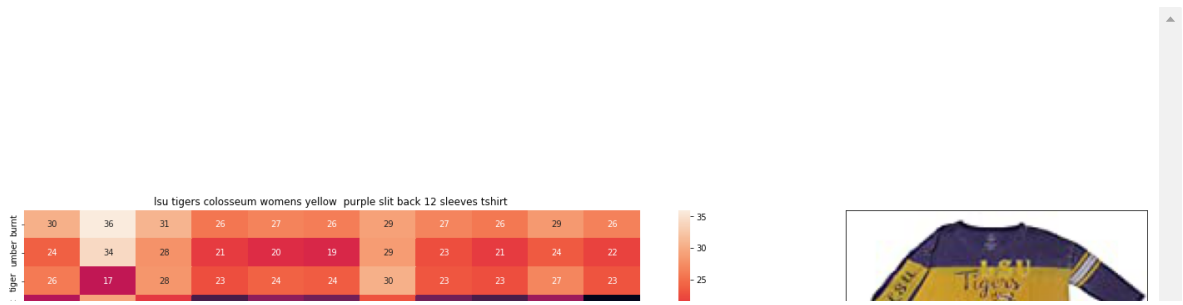
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQBBMI	Si-Row	Yellow	TOYS_AND_GAMES

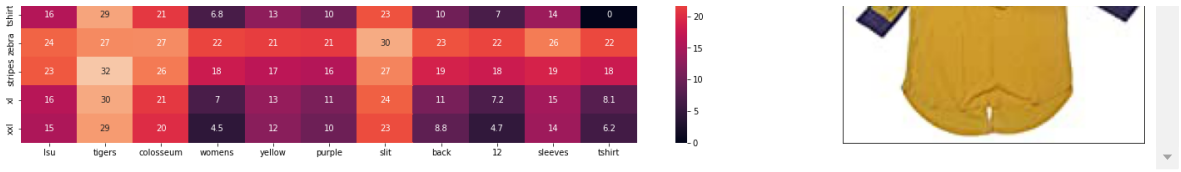


ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 4.397909927548852

==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B073R5Q8HD	Colosseum	Yellow	SPORTING_GOODS

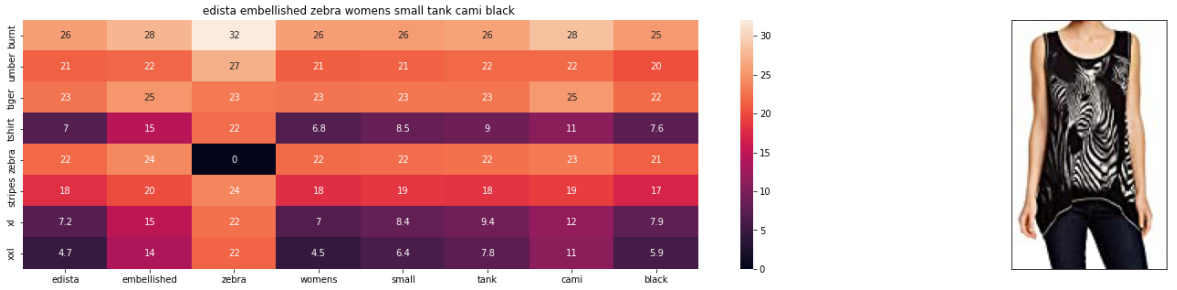




ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 4.451228392959053

==========

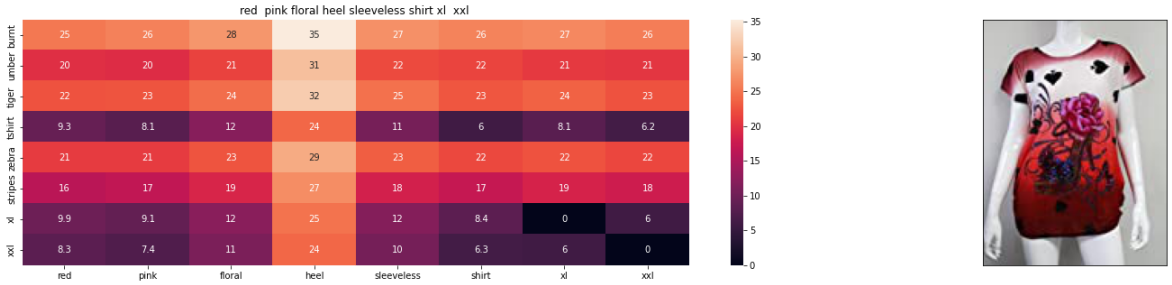
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B074P8MD22	Edista	Black	SHIRT



ASIN : B074P8MD22
Brand : Edista
euclidean distance from input : 4.518977416396553

==========

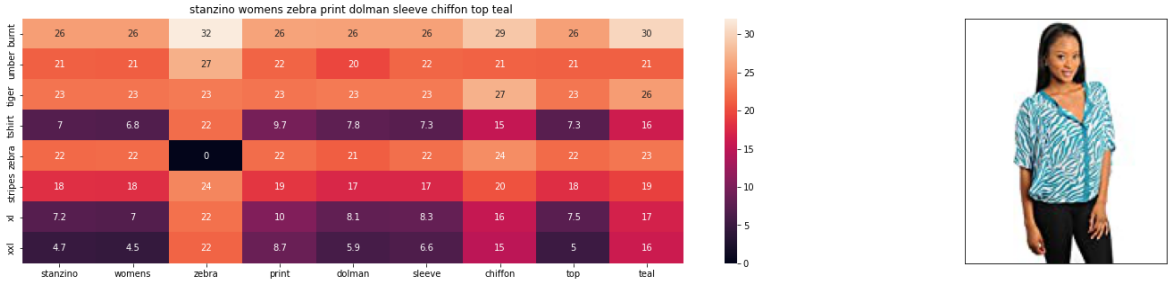
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63QQE	Si-Row	Red	TOYS_AND_GAMES



ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 4.529374695004907

==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00C0I3U3E	Stanzino	Teal	SHIRT



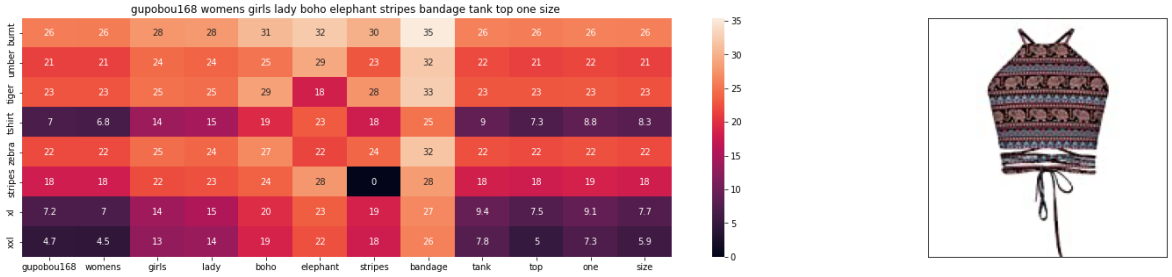
ASIN : B00C0I3U3E

Brand : Stanzino

euclidean distance from input : 4.530325759292061

==========

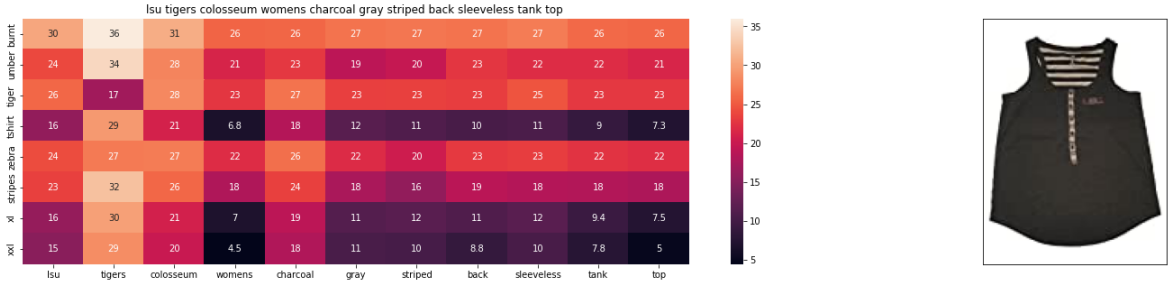
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01ER184O6	GuPoBoU168	Brown	SKIRT



ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 4.546816642558488

==========

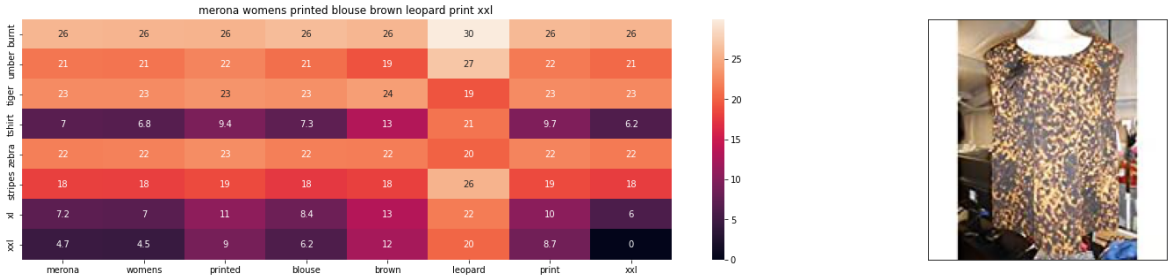
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B073R4ZM7Y	Colosseum	Gray	SPORTING_GOODS



ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 4.548355162978584

==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B071YF3WDD	Merona	,-Brown-Leopard-Print	SHIRT

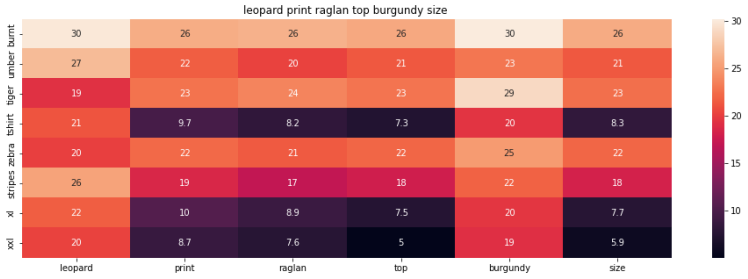


ASIN : B071YF3WDD
Brand : Merona
euclidean distance from input : 4.610626662612374

==========

ASIN	Brand	Color	Product Type

B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01C6ORLDQ	1-Mad-Fit	Burgundy	SHIRT

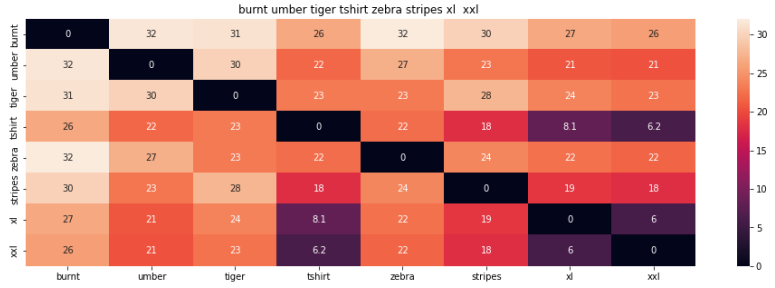


ASIN : B01C6ORLDQ
Brand : 1 Mad Fit
euclidean distance from input : 4.645917892817431
==========

In [ ]:

```
idf_w2v_brand(12566, 5, 50, 20)
```

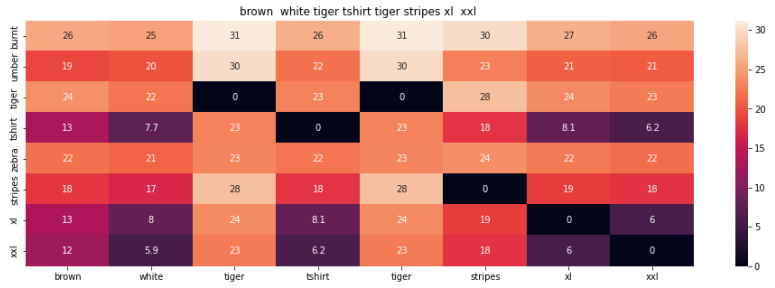
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES



```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0
```

```
==========================================================================================================================
```

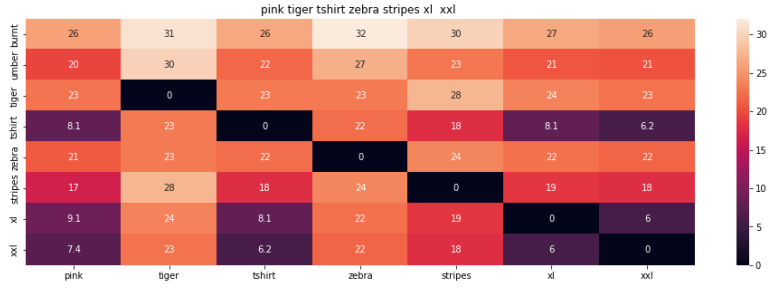
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCWTO	Si-Row	Brown	TOYS_AND_GAMES



```
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.433721923828125
```

```
==========================================================================================================================
```

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQASS6	Si-Row	Pink	TOYS_AND_GAMES

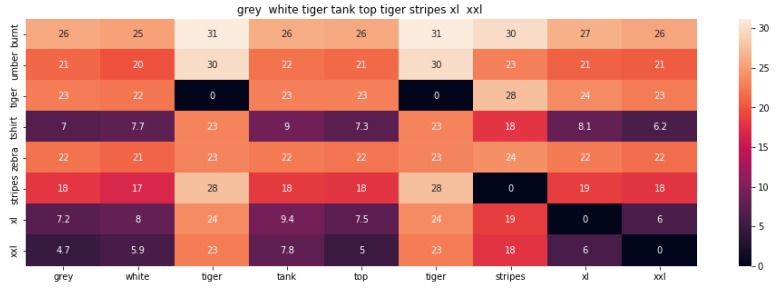


```
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 1.6550929332897277
```

```
==========================================================================================================================
```

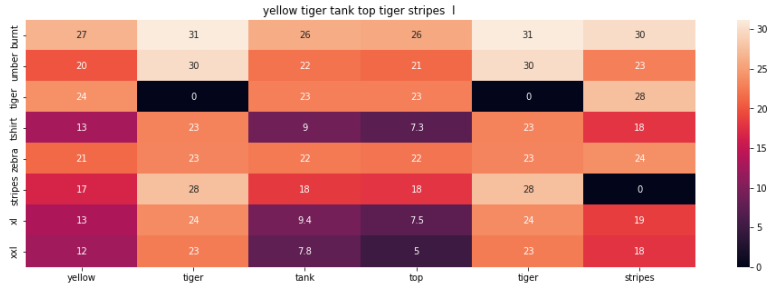
ASIN	Brand	Color	Product Type

B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAFZ2	Si-Row	Grey	TOYS_AND_GAMES



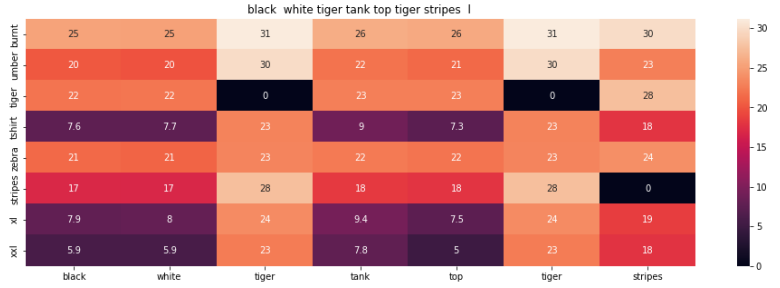
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 1.7729360063543584
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAUWA	Si-Row	Yellow	TOYS_AND_GAMES



ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 1.8028780160201077
==========

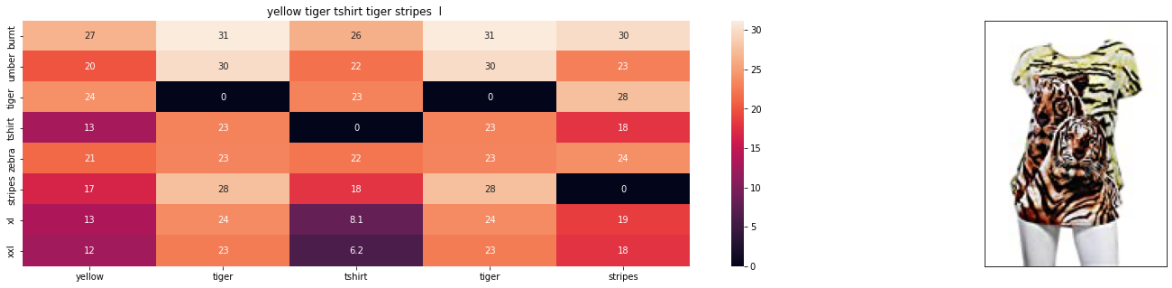
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAO94	Si-Row	White	TOYS_AND_GAMES



ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 1.8031960230557966
==========

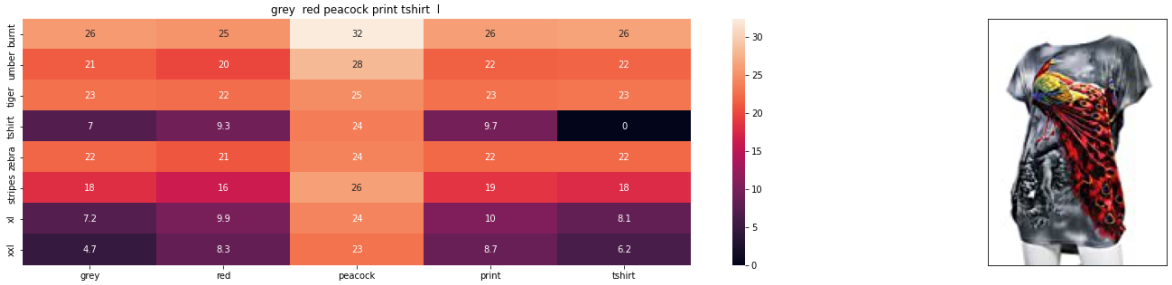
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES

B00JXQCUIC	Si-Row	Yellow	TOYS_AND_GAMES



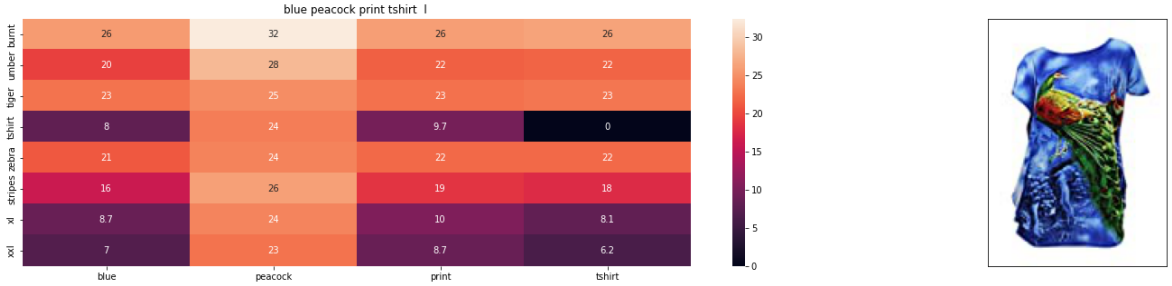
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 1.8214161616056013
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQCFRS	Si-Row	Grey	TOYS_AND_GAMES



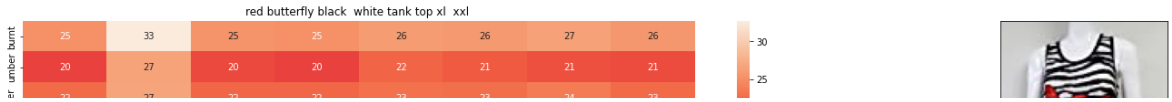
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.9077767808904913
==========

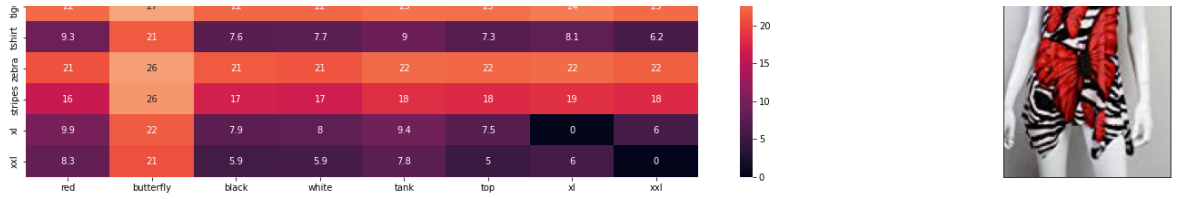
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQC8L6	Si-Row	Blue	TOYS_AND_GAMES



ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 1.9214293049716347
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63CW2	Si-Row	Red	TOYS_AND_GAMES

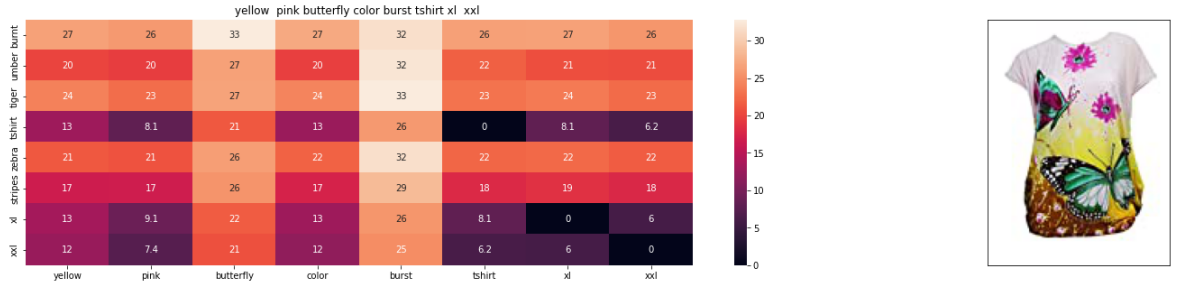




ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 1.936463165611727

==========

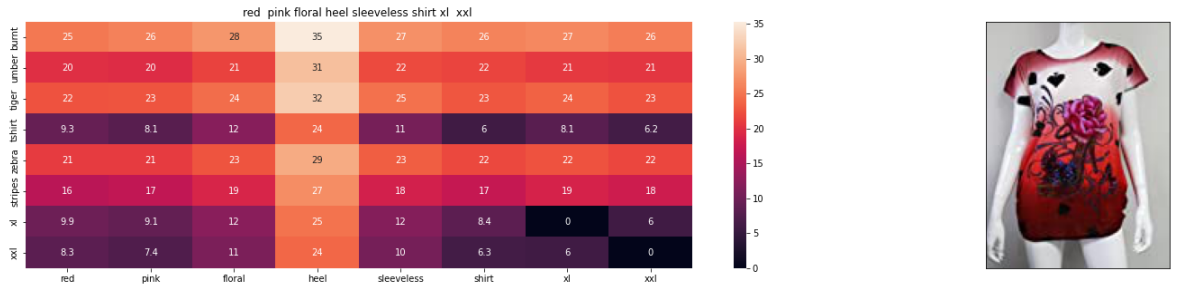
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQBBMI	Si-Row	Yellow	TOYS_AND_GAMES



ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.956703810586869

==========

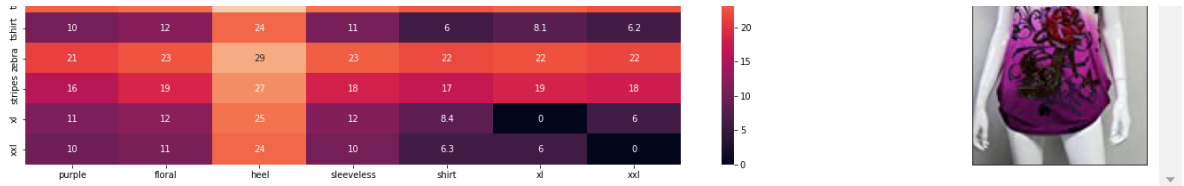
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63QQE	Si-Row	Red	TOYS_AND_GAMES



ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 1.9806064955788791

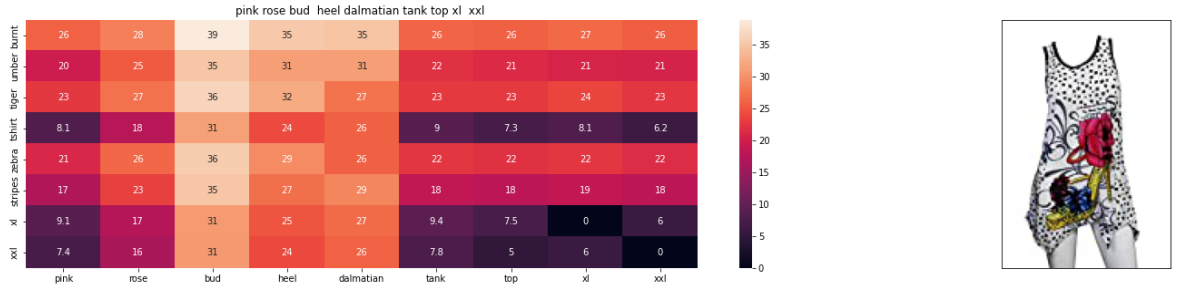
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JV63VC8	Si-Row	Purple	TOYS_AND_GAMES



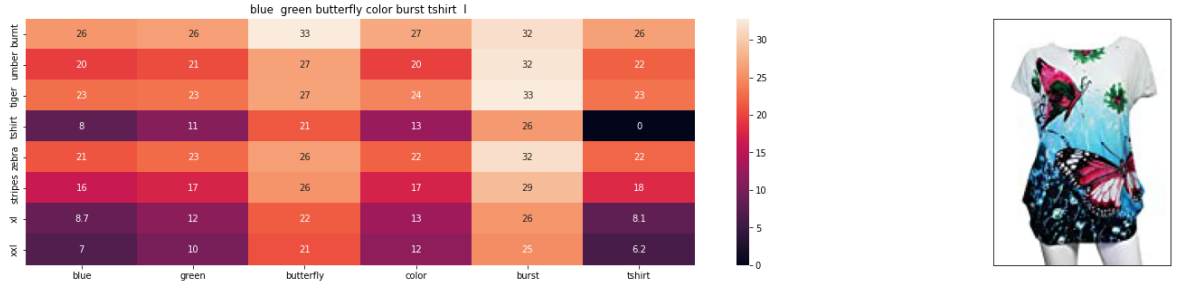
ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 2.012185530557572
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQAX2C	Si-Row	Pink	TOYS_AND_GAMES



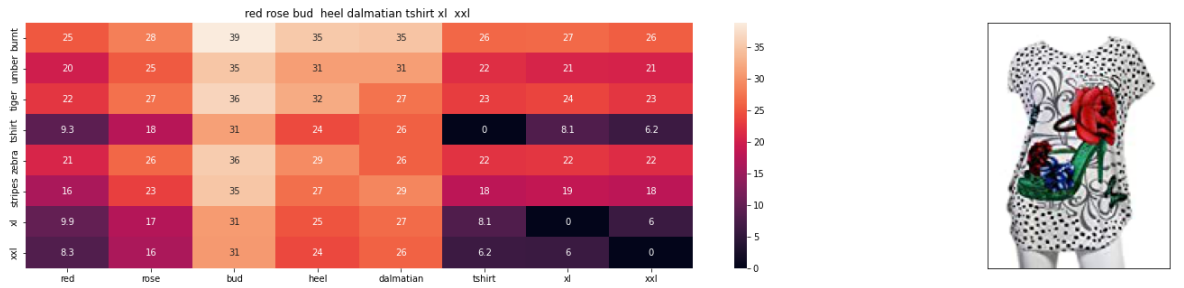
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 2.01335171819074
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQC0C8	Si-Row	Blue	TOYS_AND_GAMES



ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 2.0138832789151717
==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B00JXQABB0	Si-Row	Red	TOYS_AND_GAMES

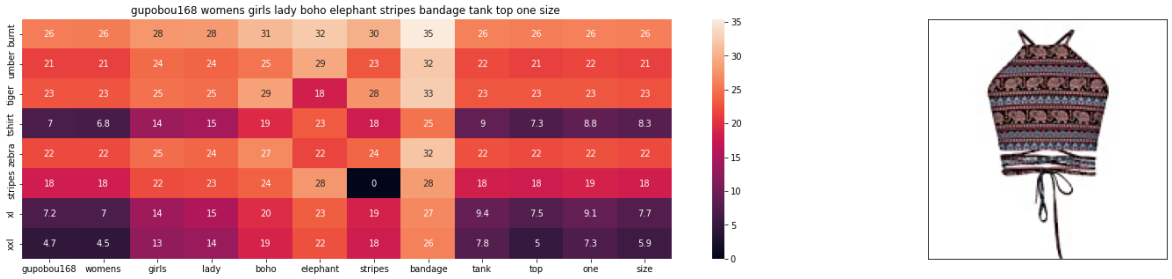


ASIN : B00JXQABB0
Brand : Si Row

euclidean distance from input : 2.0367257554998663

==========

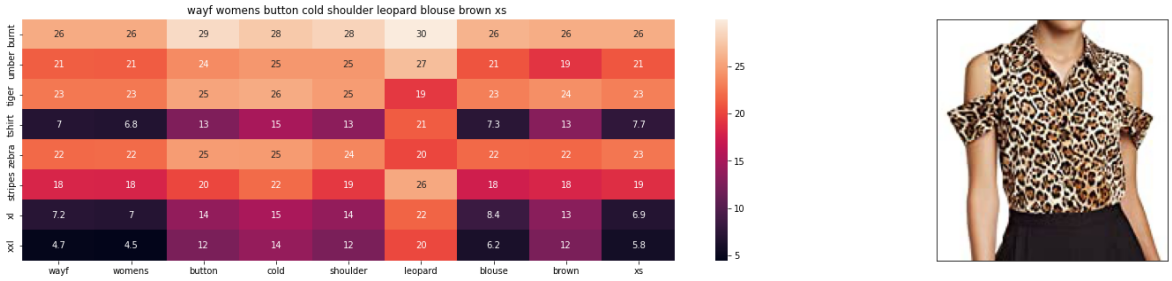
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01ER184O6	GuPoBoU168	Brown	SKIRT



ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 2.656204098419553

==========

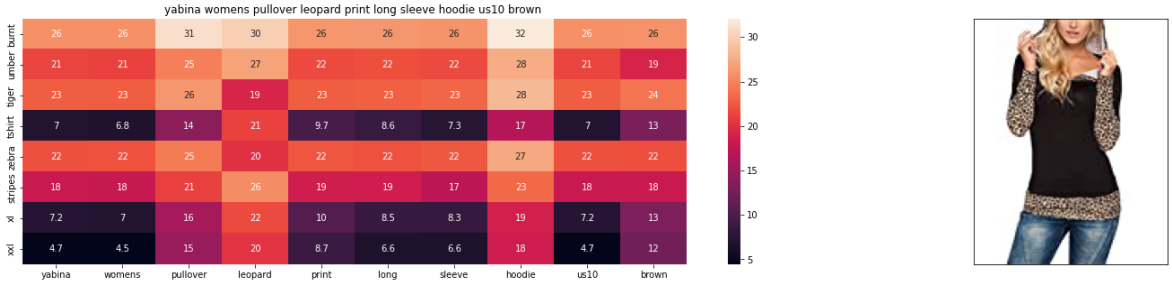
ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01LZ7BQ4H	WAYF	Brown	SHIRT



ASIN : B01LZ7BQ4H
Brand : WAYF
euclidean distance from input : 2.6849067129437008

==========

ASIN	Brand	Color	Product Type
B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01KJUM6JI	YABINA	Brown	BOOKS_1973_AND_LATER

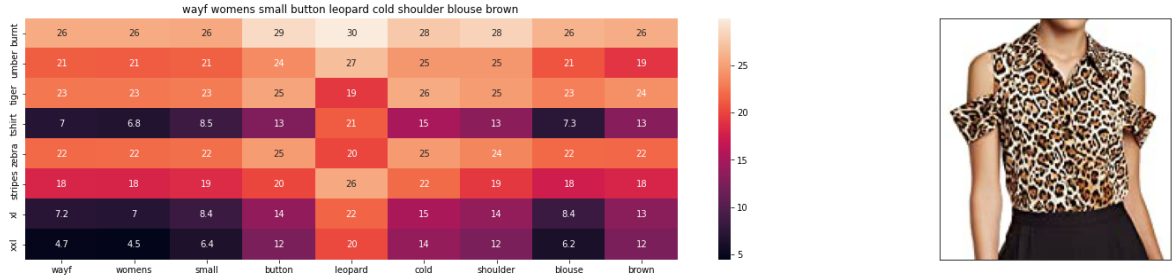


ASIN : B01KJUM6JI
Brand : YABINA
euclidean distance from input : 2.6858381232997024

==========

ASIN	Brand	Color	Product Type

B00JXQB5FQ	Si-Row	Brown	TOYS_AND_GAMES
B01M06V4X1	WAYF	Brown	SHIRT




```
ASIN : B01M06V4X1
Brand : WAYF
euclidean distance from input : 2.694761948650377
```

Deep Learning Solution

Using vgg-16 to get the dense vectors for each image.

In [ ]:

```
img_width, img_height = 224, 224

top_model_weights_path = 'bottleneck_fc_model.h5'
train_data_dir = 'images2/'
nb_train_samples = 16042
epochs = 50
batch_size = 1


def save_bottlebeck_features():

    #Function to compute VGG-16 CNN for image feature extraction.

    asins = []
    datagen = ImageDataGenerator(rescale=1. / 255)

    # build the VGG16 network
    model = applications.VGG16(include_top=False, weights='imagenet')
    generator = datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode=None,
        shuffle=False)

    for i in generator.filenames:
        asins.append(i[2:-5])

    bottleneck_features_train = model.predict_generator(generator, nb_train_samples // batch_size)
    bottleneck_features_train = bottleneck_features_train.reshape((16042,25088))

    np.save(open('16k_data_cnn_features.npy', 'wb'), bottleneck_features_train)
    np.save(open('16k_data_cnn_feature_asins.npy', 'wb'), np.array(asins))


save_bottlebeck_features()
```

In [ ]:

```
bottleneck_features_train = np.load('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/Applied_AI_Workshop_Coc
asins = np.load('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/Applied_AI_Workshop_Code_Data/16k_data_cnn_
asins = list(asins)
```

In [ ]:

```
data = pd.read_pickle('/content/drive/MyDrive/AAIC/Case Studies/Amazon Fashion Discovery Engine/pickles/16k_apperal_data_preprocessec
df_asins = list(data['asin'])
```

In [ ]:

```
def get_similar_products_cnn(doc_id, num_results):
  doc_id = asins.index(df_asins[doc_id])
  pairwise_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,-1))
  indices = np.argsort(pairwise_dist.flatten())[:num_results]
  pdists = np.sort(pairwise_dist.flatten())[:num_results]
  for i in range(len(indices)):
    rows = data[['medium_image_url', 'title']].loc[data['asin'] == asins[indices[i]]]
    for indx, row in rows.iterrows():
      display(Image(url=row['medium_image_url'], embed = True))
      print(f"Product Title: {row['title']}")
      print(f"Euclidean Distance from Input Image: {pdists[i]}")
      print(f"Amazon URL: www.amazon.com/dp/{asins[indices[i]]}")
```

In [ ]:

```
get_similar_products_cnn(12566, 20)
```

Product Title: burnt umber tiger tshirt zebra stripes xl xxl
Euclidean Distance from Input Image: 6.325959930109093e-06
Amazon URL: www.amazon.com/dp/B00JXQB5FQ

Product Title: pink tiger tshirt zebra stripes xl xxl
Euclidean Distance from Input Image: 30.0501708984375
Amazon URL: www.amazon.com/dp/B00JXQASS6

Product Title: yellow tiger tshirt tiger stripes l
Euclidean Distance from Input Image: 41.26111602783203
Amazon URL: www.amazon.com/dp/B00JXQCUIC

Product Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean Distance from Input Image: 44.00015640258789
Amazon URL: www.amazon.com/dp/B00JXQCWTO

Product Title: kawaii pastel tops tees pink flower design
Euclidean Distance from Input Image: 47.38248062133789
Amazon URL: www.amazon.com/dp/B071FCWD97

Product Title: womens thin style tops tees pastel watermelon print
Euclidean Distance from Input Image: 47.71841812133789
Amazon URL: www.amazon.com/dp/B01JUNHBRM

Product Title: kawaii pastel tops tees baby blue flower design
Euclidean Distance from Input Image: 47.902061462402344
Amazon URL: www.amazon.com/dp/B071SBCY9W

Product Title: edv cheetah run purple multi xl
Euclidean Distance from Input Image: 48.04648208618164
Amazon URL: www.amazon.com/dp/B01CUPYBM0

Product Title: danskin womens vneck loose performance tee xsmall pink ombre
Euclidean Distance from Input Image: 48.101837158203125
Amazon URL: www.amazon.com/dp/B01F7PHXY8

Product Title: summer alpaca 3d pastel casual loose tops tee design
Euclidean Distance from Input Image: 48.118865966796875
Amazon URL: www.amazon.com/dp/B01I80A93G

Product Title: miss chievous juniors striped peplum tank top medium shadowpeach
Euclidean Distance from Input Image: 48.131221771240234
Amazon URL: www.amazon.com/dp/B0177DM70S

Product Title: red pink floral heel sleeveless shirt xl xxl
Euclidean Distance from Input Image: 48.16944885253906
Amazon URL: www.amazon.com/dp/B00JV63QQE

Product Title: moana logo adults hot v neck shirt black xxl
Euclidean Distance from Input Image: 48.25678634643555
Amazon URL: www.amazon.com/dp/B01LX6H43D

Product Title: abaday multicolor cartoon cat print short sleeve longline shirt large
Euclidean Distance from Input Image: 48.26568603515625
Amazon URL: www.amazon.com/dp/B01CR57YY0

Product Title: kawaii cotton pastel tops tees peach pink cactus design
Euclidean Distance from Input Image: 48.36260223388672
Amazon URL: www.amazon.com/dp/B071WYLBZS

Product Title: chicago chicago 18 shirt women pink
Euclidean Distance from Input Image: 48.38360595703125
Amazon URL: www.amazon.com/dp/B01GXAZTRY

Product Title: yichun womens tiger printed summer tshirts tops
Euclidean Distance from Input Image: 48.44935607910156
Amazon URL: www.amazon.com/dp/B010NN9RXO

Product Title: nancy lopez whimsy short sleeve whiteblacklemon drop xs
Euclidean Distance from Input Image: 48.47888946533203
Amazon URL: www.amazon.com/dp/B01MPX6IDX

Product Title: womens tops tees pastel peach ice cream cone print
Euclidean Distance from Input Image: 48.55795669555664
Amazon URL: www.amazon.com/dp/B0734GRKZL

Product Title: uswomens mary j blige without tshirts shirt
Euclidean Distance from Input Image: 48.61437225341797
Amazon URL: www.amazon.com/dp/B01M0XXFKK

In [ ]: