```
#import all the necessary packages.

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 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 plotly.graph_objs import Scatter, Layout

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

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnir
  import pandas.util.testing as tm
```

```
from google.colab import drive
drive.mount('/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947

Enter your authorization code:
··········
Mounted at /gdrive
```

```
# we have give a json file which consists of all information about
# the products
# loading the data using pandas' read_json file.
data = pd.read_json('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/tops_fashion.json')
```

```
print ('Shape : ', data.shape)
```

```
Shape :  (183138, 19)
```

```
data.dtypes
```

```
sku                 object
asin                object
product_type_name   object
formatted_price     object
author              object
color               object
brand               object
publisher           object
availability        object
reviews             object
large_image_url     object
availability_type   object
small_image_url     object
editorial_review    object
title               object
model               object
medium_image_url    object
manufacturer        object
editorial_reivew    object
dtype: object
```

```
data.isnull().sum()
```

```
sku                 182775
asin                     0
product_type_name        0
formatted_price     154743
author              183137
color               118182
brand                  151
publisher           140239
availability        158606
reviews                  0
large_image_url          0
availability_type   158579
small_image_url          0
editorial_review    180380
title                    0
model               120768
medium_image_url         0
manufacturer        140239
editorial_reivew      2758
dtype: int64
```

```
# each product/item has 19 features in the raw dataset.
data.columns # prints column-names or feature-names.
```

```
Index(['sku', 'asin', 'product_type_name', 'formatted_price', 'author',
       'color', 'brand', 'publisher', 'availability', 'reviews',
```

```
data.head(3)
```

	sku	asin	product_type_name	formatted_price	author	color	brand	p
0	None	B016I2TS4W	SHIRT	None	None	None	FNC7C	
1	None	B01N49AI08	SHIRT	None	None	None	FIG Clothing	
2	None	B01JDPCOHO	SHIRT	None	None	None	FIG Clothing	

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

```
print ('Number of data points : ', data.shape[0], \
       'Number of features:', data.shape[1])
data.head() # prints the top rows in the table.
```

```
Number of data points :  183138 Number of features: 7
```

	asin	brand	color	medium_image_url	product_type_name	titl
0	B016I2TS4W	FNC7C	None	https://images-na.ssl-images-amazon.com/images...	SHIRT	Minion Com Superheroe Ironma Long Sleev R.
1	B01N49AI08	FIG Clothing	None	https://images-na.ssl-images-	SHIRT	FIG Clothin Womens Iz

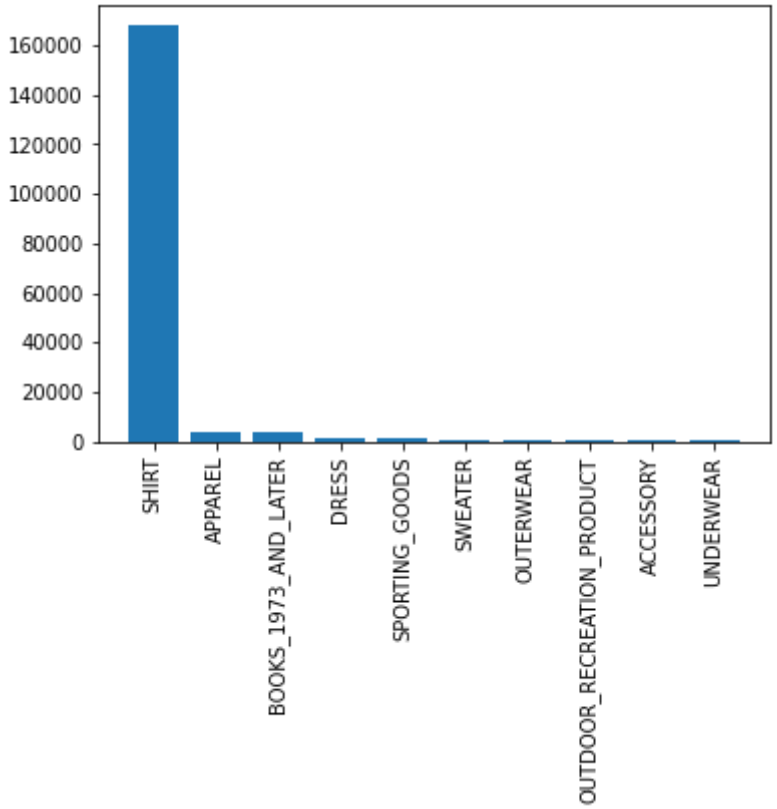
```
data['product_type_name'].describe()
```

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

```
# top 10 product type
X=data['product_type_name'].value_counts().sort_values(ascending=False).head(10)
```

```
plt.bar(X.index,X.values)
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], <a list of 10 Text major ticklabel objects>)
```



Top product type is a shirt

```
print(data['brand'].describe())
print(data['brand'].isnull().sum())
```

```
count     182987
unique     10577
top         Zago
freq          223
Name: brand, dtype: object
151
```

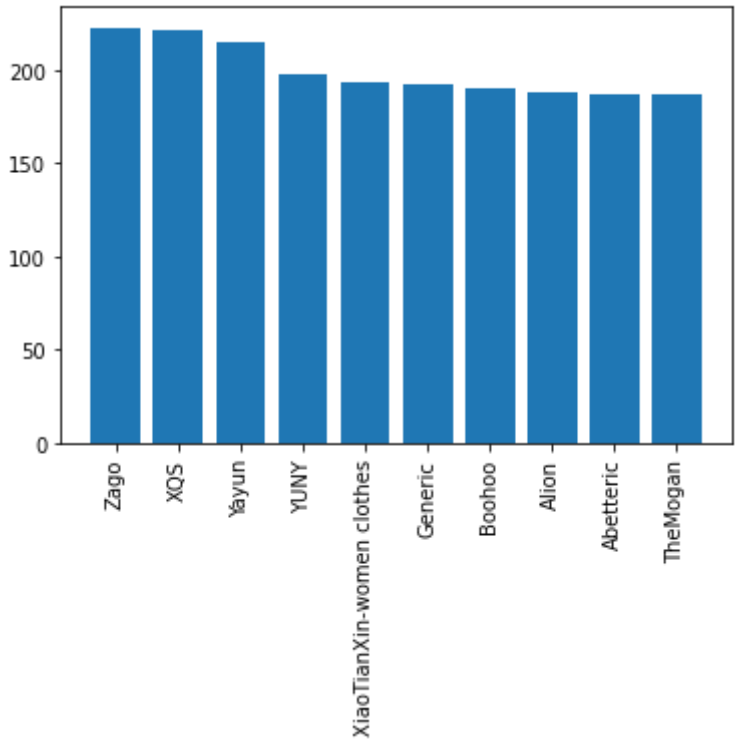
We have 151 missing values with 10577 unique values

```
# top 10 brand
X= data['brand'].value_counts().sort_values(ascending=False).head(10)
print(X)
```

```
Zago                          223
```

```
plt.bar(X.index,X.values)
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], <a list of 10 Text major ticklabel objects>)
```




```
print(data['color'].describe())
print(data['color'].isnull().sum())
```

```
count     64956
unique     7380
top       Black
freq      13207
Name: color, dtype: object
118182
```

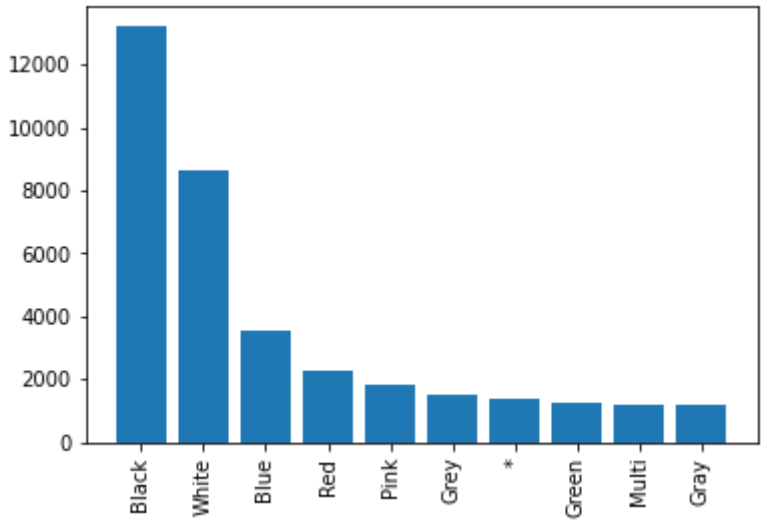
118K missing values in this category

```
# top 10 color
X= data['color'].value_counts().sort_values(ascending=False).head(10)
print(X)
```

```
Black    13207
White     8616
Blue      3570
Red       2289
Pink      1842
Grey      1499
*         1388
Green     1258
Multi     1203
Gray      1189
Name: color, dtype: int64
```

```
plt.bar(X.index,X.values)
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], <a list of 10 Text major ticklabel objects>)
```



We have one color value as "*" don't know what it means

```
print(data['formatted_price'].describe())
print(data['formatted_price'].isnull().sum())
```

```
count      28395
unique      3135
top       $19.99
freq          945
Name: formatted_price, dtype: object
154743
```

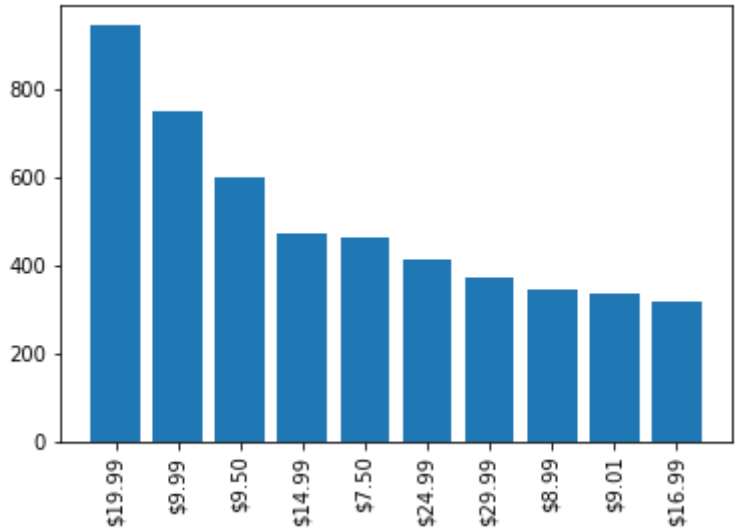
154K missing values

```
# top 10 color
X= data['formatted_price'].value_counts().sort_values(ascending=False).head(10)
print(X)
```

```
$19.99    945
$9.99     749
$9.50     601
$14.99    472
$7.50     463
$24.99    414
$29.99    370
$8.99     343
$9.01     336
$16.99    317
Name: formatted_price, dtype: int64
```

```
plt.bar(X.index,X.values)
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], <a list of 10 Text major ticklabel objects>)
```




```
print(data['title'].describe())
```

```
count                                                183138
unique                                               175985
top       Nakoda Cotton Self Print Straight Kurti For Women
freq                                                     77
Name: title, dtype: object
```

```
print(data['title'].isnull().sum())
```

```
0
```

Removing null values

```
# consider products which have price information
# data['formatted_price'].isnull() => gives the information
#about the dataframe row's which have null values price == None|Null
data = data.loc[~data['formatted_price'].isnull()]
print('Number of data points After eliminating price=NULL :', data.shape[0])
```

```
Number of data points After eliminating price=NULL : 28395
```

```
# consider products which have color information
# data['color'].isnull() => gives the information about the dataframe row's which have nul
data =data.loc[~data['color'].isnull()]
print('Number of data points After eliminating color=NULL :', data.shape[0])
```

```
Number of data points After eliminating color=NULL : 28385
```

```
# read data from pickle file from previous stage
data = pd.read_pickle('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/pickels/28k_apparel_

# find number of products that have duplicate titles.
print(sum(data.duplicated('title')))
```

```
# we have 2325 products which have same title but different color
```

```
2325
```

```
# Remove All products with very few words in title
data_sorted = data[data['title'].apply(lambda x: len(x.split())>4)]
print("After removal of products with short description:", data_sorted.shape[0])
```

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

```
# Sort the whole data based on title (alphabetical order of title)
data_sorted.sort_values('title',inplace=True, ascending=False)
data_sorted.head()
```

	asin	brand	color	medium_image_url	product_type_name	
61973	B06Y1KZ2WB	Éclair	Black/Pink	https://images-na.ssl-images-amazon.com/images...	SHIRT	W Thi
				https://images-na.ssl		xi W

```
Titles 1:
16. woman's place is in the house and the senate shirts for Womens XXL White
17. woman's place is in the house and the senate shirts for Womens M Grey

Title 2:
25. tokidoki The Queen of Diamonds Women's Shirt X-Large
26. tokidoki The Queen of Diamonds Women's Shirt Small
27. tokidoki The Queen of Diamonds Women's Shirt Large

Title 3:
61. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print Head
62. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print Head
63. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print Head
64. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print Head
```

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

len(indices)
```

```
27949
```

```
import itertools
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
    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',
        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
        # example: a =['a', 'b', 'c', 'd']
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', No
        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
        # if the number of words in which both strings differ are < 2 , we are considering
        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 in
            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
```

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

```
print('Number of data points : ', data.shape[0])
```

```
Number of data points :  17592
```

```
In the previous cell, we sorted whole data in alphabetical order of  titles.Then, we rem

But there are some products whose titles are not adjacent but very similar.

Examples:

Titles-1
86261.  UltraClub Women's Classic Wrinkle-Free Long Sleeve Oxford Shirt, Pink, XX-Large
115042. UltraClub Ladies Classic Wrinkle-Free Long-Sleeve Oxford Light Blue XXL

TItles-2
75004.  EVALY Women's Cool University Of UTAH 3/4 Sleeve Raglan Tee
109225. EVALY Women's Unique University Of UTAH 3/4 Sleeve Raglan Tees
120832. EVALY Women's New University Of UTAH 3/4-Sleeve Raglan Tshirt
```

```
data = pd.read_pickle('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/pickels/17k_apperal_
```

```
# This code snippet takes significant amount of time.
# O(n^2) time.
# Takes about an hour to run on a decent computer.

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

stage2_dedupe_asins = []
while len(indices)!=0:
    i = indices.pop()
    stage2_dedupe_asins.append(data['asin'].loc[i])
    # consider the first apperal's title
    a = data['title'].loc[i].split()
    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of
    for j in indices:

        b = data['title'].loc[j].split()
        # store the list of words of jth string in b, ex: b = ['tokidoki', 'The', 'Queen',

        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
        # example: a =['a', 'b', 'c', 'd']
```

```
        # b = ['a', 'b', 'd']
        # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c','d'), ('d', No
        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 < 3 , we are considering
        if (length - count) < 3:
            indices.remove(j)
```

```
# from whole previous products we will consider only
# the products that are found in previous cell
#data = data.loc[data['asin'].isin(stage2_dedupe_asins)]
```

```
#print('Number of data points after stage two of dedupe: ',data.shape[0])
# from 17k apperals we reduced to 16k apperals
```

```
Number of data points after stage two of dedupe:  43
```

Text Preprocessing

```
data = pd.read_pickle('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/pickels/16k_apperal_
```

Text Preprocessing process includes :

1. Stopword Removal
2. Removing HTML tags if any
3. Word Normalization like Stemming , Lemmitization etc
4. Removing Special Characters
5. Converting all in Lower-case

```
# we use the list of stop words that are downloaded from nltk lib.
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
print ('list of stop words:', stop_words)

def nlp_preprocessing(total_text, index, column):
    if type(total_text) is not int:
        string = ""
        for words in total_text.split():
            # remove the special chars in review like '"#$@!%^&*()_+-~?>< etc.
            word = ("".join(e for e in words if e.isalnum()))
            # Conver all letters to lower-case
            word = word.lower()
            # stop-word removal
            if not word in stop_words:
                string += word + " "
        data[column][index] = string
```

```
        data[column][index] = string
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
list of stop words: {'own', 'been', 'that', 'himself', "didn't", 'your', 'am', 'on',
```

```
start_time = time.clock()
# we take each title and we text-preprocess it.
for index, row in data.iterrows():
    nlp_preprocessing(row['title'], index, 'title')
# we print the time it took to preprocess whole titles
print(time.clock() - start_time, "seconds")
```

```
5.868034999999999 seconds
```

```
data.head(3)
```

	asin	brand	color	medium_image_url	product_type_name	title
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https://images-na.ssl-images-amazon.com/images...	SHIRT	featherlite ladies long sleeve stain

Stemming

```
from nltk.stem.porter import *
stemmer = PorterStemmer()
print(stemmer.stem('arguing'))
print(stemmer.stem('fishing'))
```

```
argu
fish
```

Arguing is converted in argu and fishing is converted to fish

```
# Utility Functions which we will use through the rest of the workshop.


#Display an image
def display_img(url,ax,fig):
    # we get the url of the apparel and download it
    response = requests.get(url)
    img = Image.open(BytesIO(response.content))
    # we will display it in notebook
    plt.imshow(img)

#plotting code to understand the algorithm's decision.
def plot_heatmap(keys, values, labels, url, text):
        # keys: list of words of recommended title
        # values: len(values) ==  len(keys), values(i) represents the occurence of the wor
```

```
        # values: len(values) == len(keys), values(i) represents the occurence of the wor
        # labels: len(labels) == len(keys), the values of labels depends on the model we a
                # 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 : apparel's url

        # we will devide the whole figure into two parts
        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 tit
        ax = plt.subplot(gs[0])
        # it displays a cell in white color if the word is intersection(lis of words of ti
        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()

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
    intersection = set(vec1.keys()) & set(vec2.keys())

    # we set the values of non intersecting words to zero, this is just to show the differ
    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): va
    values = [vec2[x] for x in vec2.keys()]
```

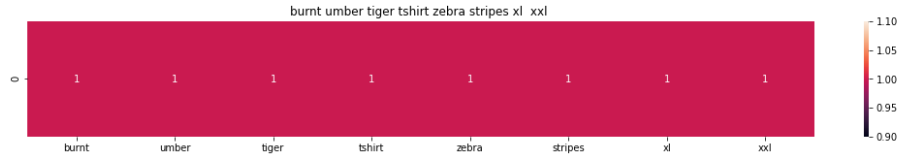
```
    # labels: len(labels) == len(keys), the values of labels depends on the model we are u
        # if model == 'bag of words': labels(i) = values(i)
        # if model == 'tfidf weighted bag of words':labels(i) = tfidf(keys(i))
        # if model == 'idf weighted bag of words':labels(i) = idf(keys(i))

    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 va
            if x in  tfidf_title_vectorizer.vocabulary_:
                labels.append(tfidf_title_features[doc_id, tfidf_title_vectorizer.vocabula
            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
            if x in  idf_title_vectorizer.vocabulary_:
                labels.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[
            else:
                labels.append(0)

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


# this function gets a list of wrods along with the frequency of each
# word given "text"
def text_to_vector(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()' t
    return Counter(words) # Counter counts the occurence of each word in list, it returns



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 of Product-Title

```
from sklearn.feature_extraction.text import CountVectorizer
title_vectorizer = CountVectorizer()
title_features   = title_vectorizer.fit_transform(data['title'])
title_features.get_shape()
```

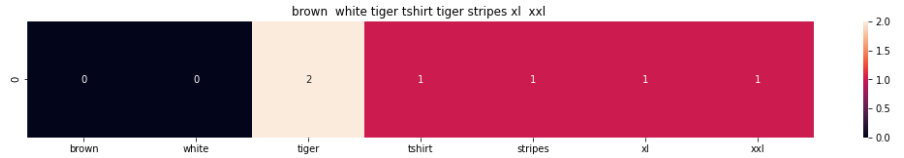
```
(16042, 12609)
```

```
def bag_of_words_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining appa
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(title_features,title_features[doc_id])

    # np.argsort will return indices of the smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])

    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3. title2, url, model
        get_result(indices[i],data['title'].loc[df_indices[0]], data['title'].loc[df_indic
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print ('Brand:', data['brand'].loc[df_indices[i]])
        print ('Title:', data['title'].loc[df_indices[i]])
        print ('Euclidean similarity with the query image :', pdists[i])
        print('='*60)

#call the bag-of-words model for a product to get similar products.
bag_of_words_model(12566, 20) # change the index if you want to.
# In the output heat map each value represents the count value
# of the label word, the color represents the intersection
# with inputs title.

#try 12566
#try 931
```



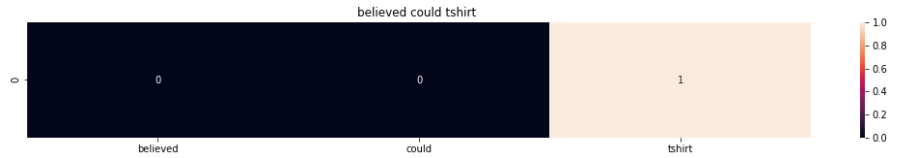
ASIN : B00JXQB5FQ
Brand: Si Row
Title: burnt umber tiger tshirt zebra stripes xl xxl
Euclidean similarity with the query image : 0.0
==========

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



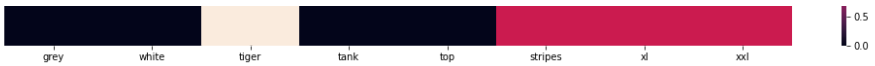
ASIN : B00JXQCWTO
Brand: Si Row
Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean similarity with the query image : 2.449489742783178
==========

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



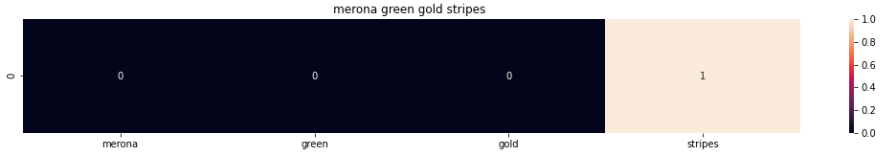
ASIN : B07568NZX4
Brand: Rustic Grace
Title: believed could tshirt
Euclidean similarity with the query image : 3.0
==========

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

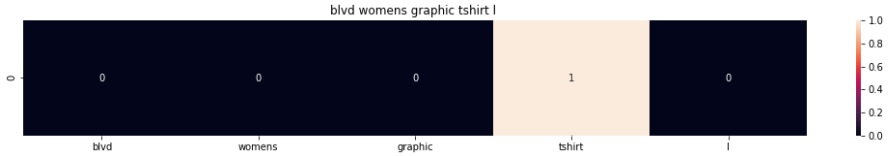


ASIN : B00JXQAFZ2
Brand: Si Row
Title: grey white tiger tank top tiger stripes xl xxl
Euclidean similarity with the query image : 3.0
==========

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

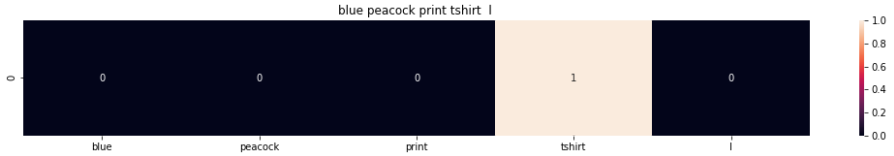


ASIN : B01KVZUB6G
Brand: Merona
Title: merona green gold stripes
Euclidean similarity with the query image : 3.1622776601683795
==========



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

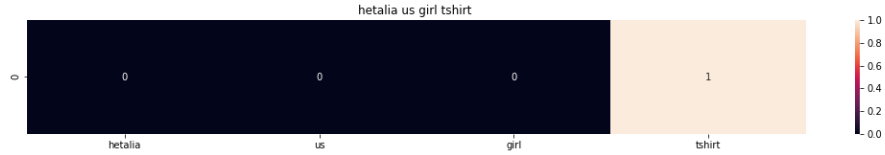
ASIN : B012VQLT6Y
Brand: KM T-shirt
Title: km tiger printed sleeveless vest tshirt
Euclidean similarity with the query image : 3.1622776601683795
==========



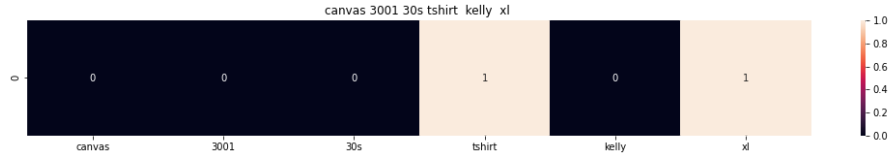
ASIN : B00JXQC8L6
Brand: Si Row
Title: blue peacock print tshirt l
Euclidean similarity with the query image : 3.1622776601683795
==========

ASIN : B06XC3CZF6
Brand: Fjallraven

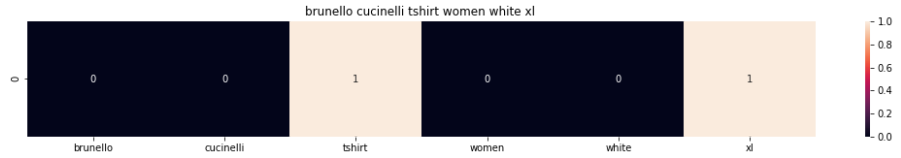
Brand: Fjallraven
Title: fjallraven womens ovik tshirt plum xxl
Euclidean similarity with the query image : 3.1622776601683795
==



ASIN : B005IT8OBA
Brand: Hetalia
Title: hetalia us girl tshirt
Euclidean similarity with the query image : 3.1622776601683795
==

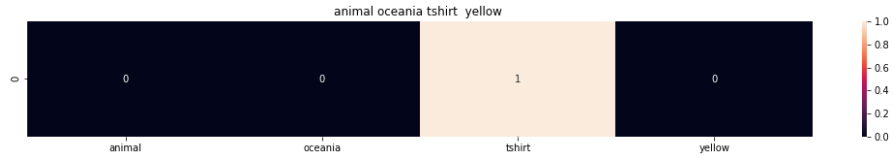


ASIN : B0088PN0LA
Brand: Red House
Title: canvas 3001 30s tshirt kelly xl
Euclidean similarity with the query image : 3.1622776601683795
==



ASIN : B06X99V6WC
Brand: Brunello Cucinelli
Title: brunello cucinelli tshirt women white xl
Euclidean similarity with the query image : 3.1622776601683795
==

ASIN : B06Y1JPW1Q
Brand: Xhilaration
Title: xhilaration womens lace tshirt salmon xxl
Euclidean similarity with the query image : 3.1622776601683795
==



ASIN : B06X6GX6WG
Brand: Animal
Title: animal oceania tshirt yellow
Euclidean similarity with the query image : 3.1622776601683795
==

ASIN : B017X8PW9U
Brand: Diesel
Title: diesel tserraf tshirt black
Euclidean similarity with the query image : 3.1622776601683795
==

ASIN : B00IAA4JIQ
Brand: I Love Lucy
Title: juniors love lucywaaaaahhhh tshirt size xl
Euclidean similarity with the query image : 3.1622776601683795
==========

TFIDF based Similartiy

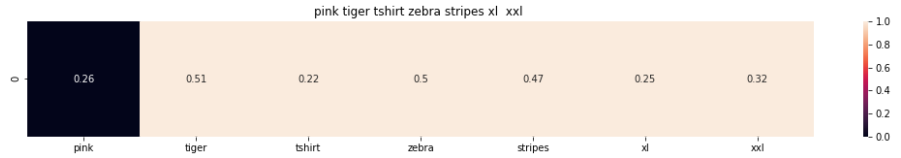
Title

```
tfidf_title_vectorizer = TfidfVectorizer(min_df = 0)
tfidf_title_features = tfidf_title_vectorizer.fit_transform(data['title'])
# tfidf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dimensions #dat
# tfidf_title_features[doc_id, index_of_word_in_corpus] = tfidf values of the word in give
```

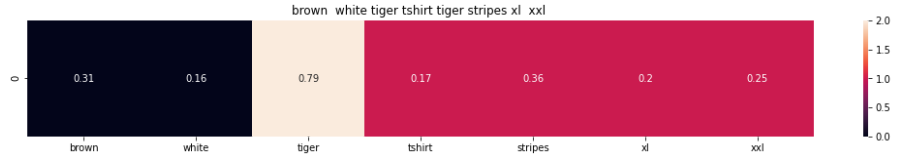
```
def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining appa
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(tfidf_title_features,tfidf_title_features[doc_id])
```

```
    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])

    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3. title2, url, model
        get_result(indices[i], data['title'].loc[df_indices[0]], data['title'].loc[df_indi
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('BRAND :',data['brand'].loc[df_indices[i]])
        print ('Eucliden distance from the given image :', pdists[i])
        print('='*125)
tfidf_model(12566, 20)
# in the output heat map each value represents the tfidf values of the label word, the col
```

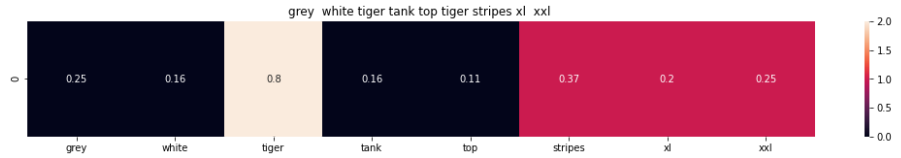
ASIN : B00JXQB5FQ
BRAND : Si Row
Eucliden distance from the given image : 0.0
==========



ASIN : B00JXQASS6
BRAND : Si Row
Eucliden distance from the given image : 0.7536331912451363
==========

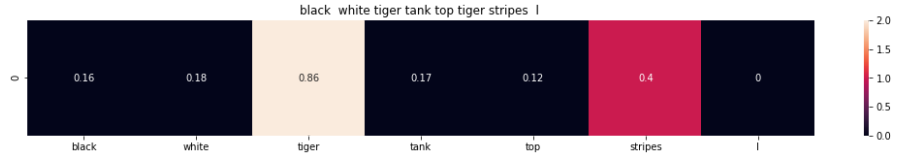


ASIN : B00JXQCWTO
BRAND : Si Row
Eucliden distance from the given image : 0.9357643943769647
==========

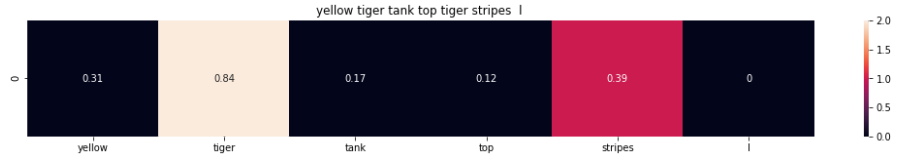


ASIN : B00JXQAFZ2
BRAND : Si Row
Eucliden distance from the given image : 0.9586153524200749
==========

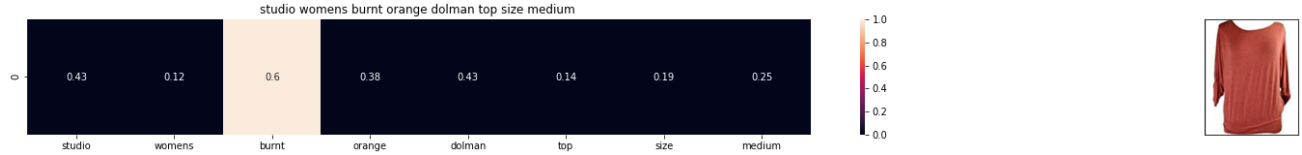
ASIN : B00JXQCUIC
BRAND : Si Row
Eucliden distance from the given image : 1.000074961446881
==========



ASIN : B00JXQAO94
BRAND : Si Row
Eucliden distance from the given image : 1.023215552457452
==========

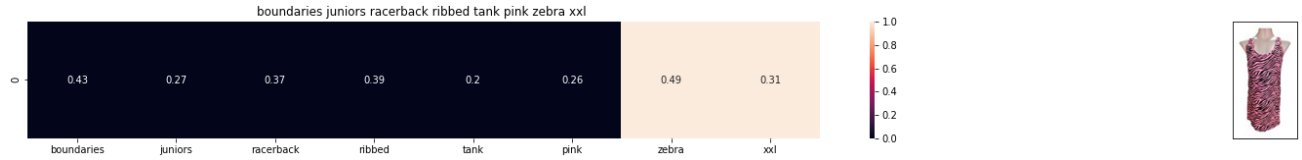


ASIN : B00JXQAUWA
BRAND : Si Row
Eucliden distance from the given image : 1.031991846303421
==========



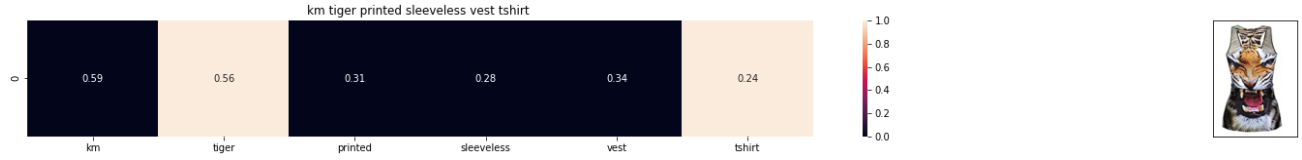
ASIN : B06XSCVFT5
BRAND : Studio M
Eucliden distance from the given image : 1.2106843670424716

==========



ASIN : B06Y2GTYPM
BRAND : No Boundaries
Eucliden distance from the given image : 1.2121683810720831

==========

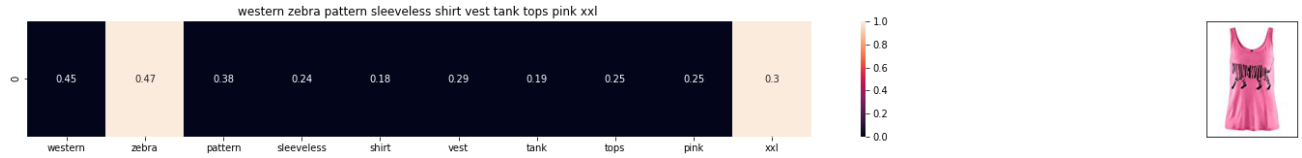


ASIN : B012VQLT6Y
BRAND : KM T-shirt
Eucliden distance from the given image : 1.219790640280982

==========

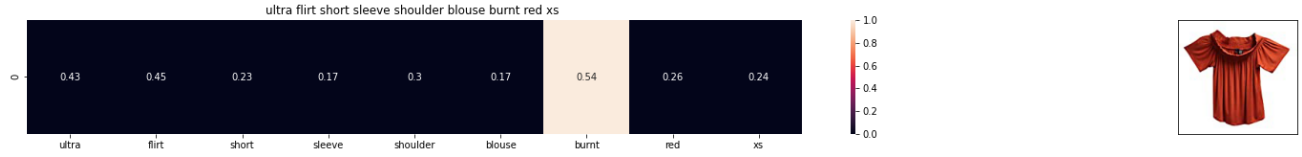
ASIN : B06Y1VN8WQ
BRAND : Black Swan
Eucliden distance from the given image : 1.2206849659998316

==========



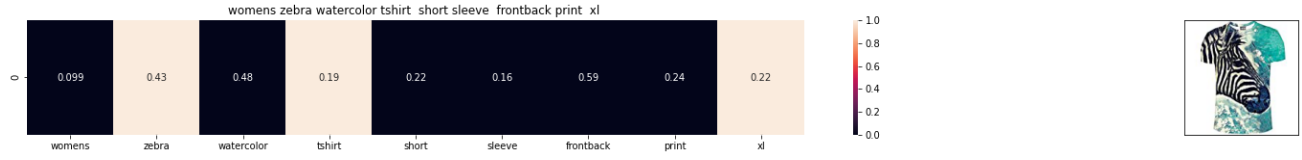
ASIN : B00Z6HEXWI
BRAND : Black Temptation
Eucliden distance from the given image : 1.221281392120943

==========



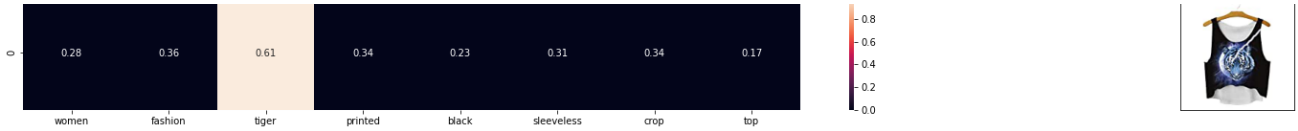
ASIN : B074TR12BH
BRAND : Ultra Flirt
Eucliden distance from the given image : 1.2313364094597743

==========



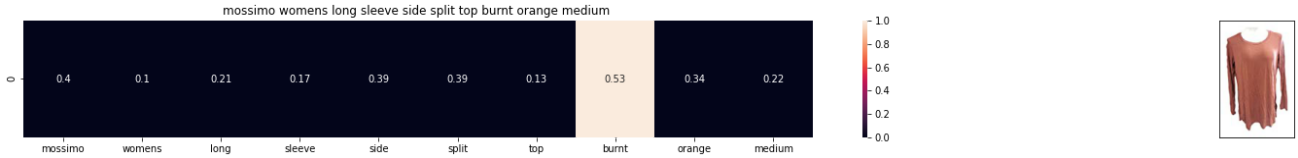
ASIN : B072R2JXKW
BRAND : WHAT ON EARTH
Eucliden distance from the given image : 1.2318451972624516

==========



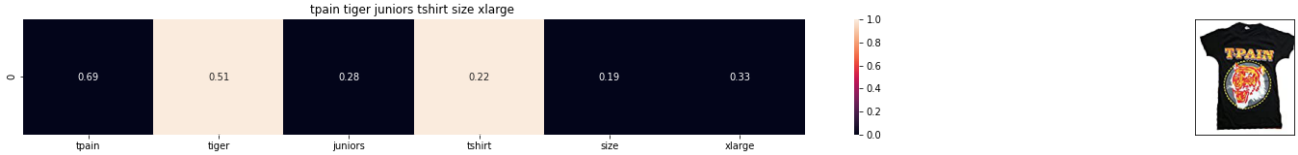
ASIN : B074T8ZYGX
BRAND : MKP Crop Top
Eucliden distance from the given image : 1.2340607457359425

==========



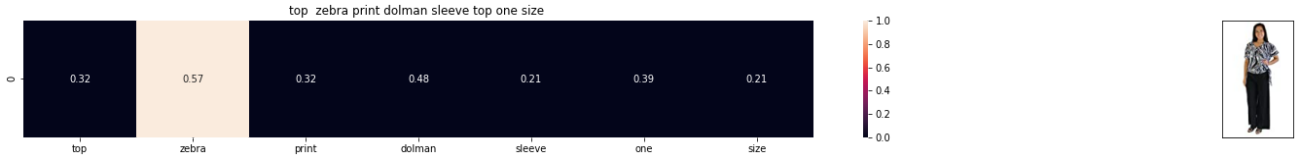
ASIN : B071ZDF6T2
BRAND : Mossimo
Eucliden distance from the given image : 1.2352785577664824

==========



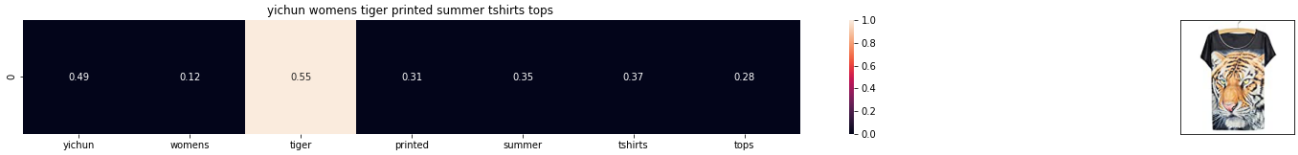
ASIN : B01K0H02OG
BRAND : Tultex
Eucliden distance from the given image : 1.236457298812782

==========



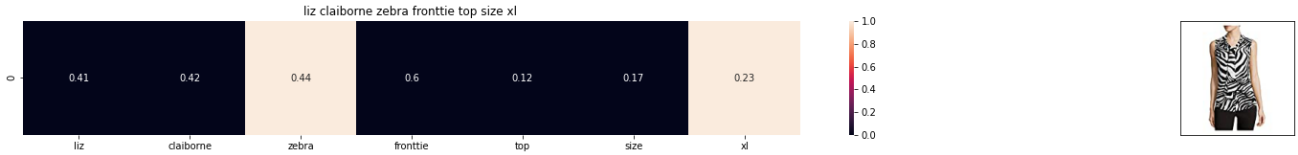
ASIN : B00H8A6ZLI
BRAND : Vivian's Fashions
Eucliden distance from the given image : 1.24996155052848

==========



ASIN : B010NN9RXO
BRAND : YICHUN
Eucliden distance from the given image : 1.2535461420856102

==========



ASIN : B06XBY5QXL
BRAND : Liz Claiborne
Eucliden distance from the given image : 1.2538832938357722

==========

IDF based Similarity

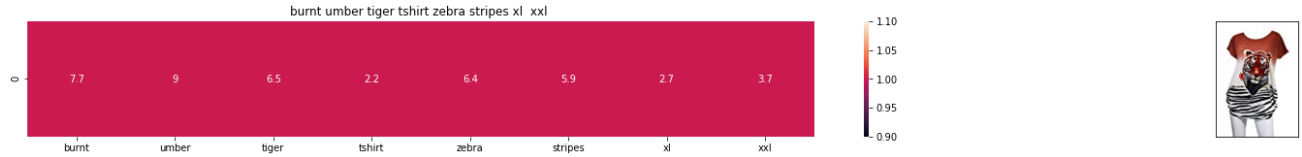
```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data['title'])

# idf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dimensions #dat
# idf_title_features[doc_id, index_of_word_in_corpus] = number of times the word occured i
```

```
# idf_title_features[doc_id, index_of_word_in_corpus] = number of times the word occured i
```

```
def n_containing(word):
    # return the number of documents which had the given word
    return sum(1 for blob in data['title'] if word in blob.split())

def idf(word):
    # idf = log(#number of docs / #number of docs which had the given word)
    return math.log(data.shape[0] / (n_containing(word)))
```

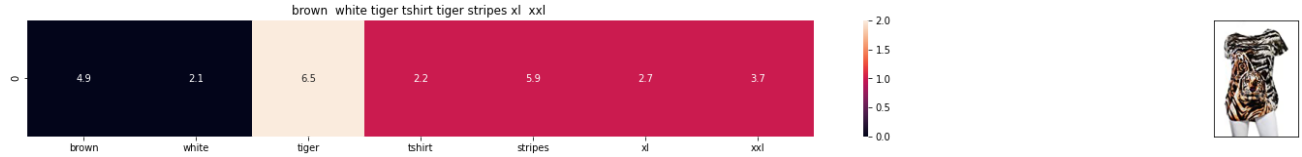
```
# we need to convert the values into float
idf_title_features  = idf_title_features.astype(np.float)

for i in idf_title_vectorizer.vocabulary_.keys():
    # for every word in whole corpus we will find its idf value
    idf_val = idf(i)

    # to calculate idf_title_features we need to replace the count values with the idf val
    # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0] will return
    for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:

        # we replace the count values of word i in document j with  idf_value of word i
        # idf_title_features[doc_id, index_of_word_in_courpus] = idf value of word
        idf_title_features[j,idf_title_vectorizer.vocabulary_[i]] = idf_val
```

```
def idf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining appa
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(idf_title_features,idf_title_features[doc_id])

    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    #data frame indices of the 9 smallest distace's
    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_indic
        print('ASIN :',data['asin'].loc[df_indices[i]])
        print('Brand :',data['brand'].loc[df_indices[i]])
        print ('euclidean distance from the given image :', pdists[i])
        print('='*125)



idf_model(12566,20)
# in the output heat map each value represents the idf values of the label word, the color
```

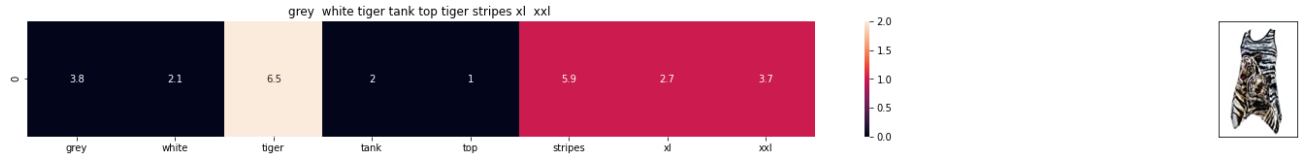


ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from the given image : 0.0
==========

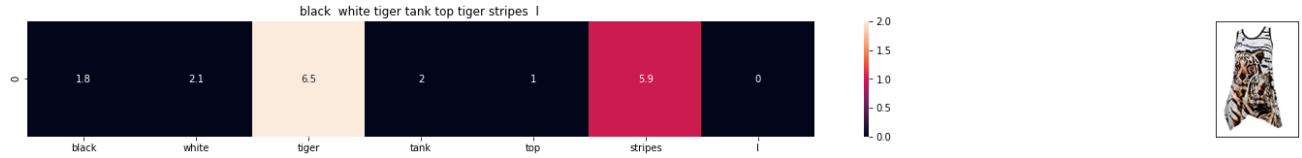
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from the given image : 12.20507131122177
==========



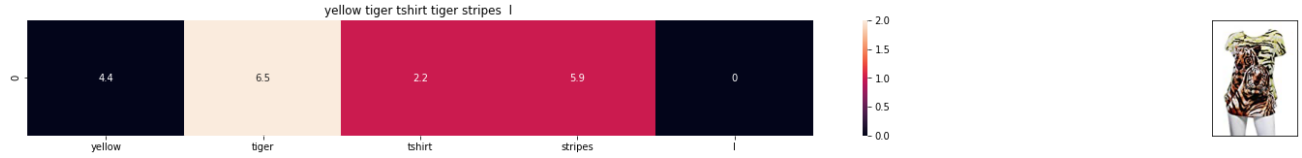
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from the given image : 14.468362685603465
==========



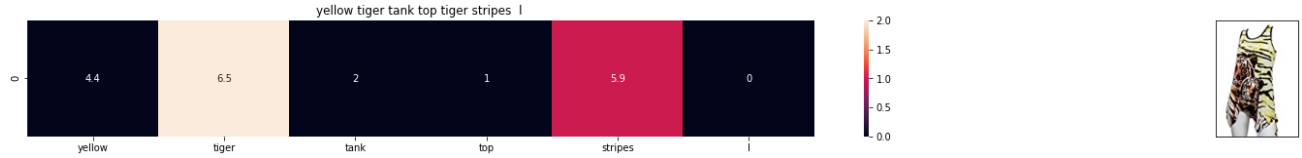
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from the given image : 14.486832924778964
==========



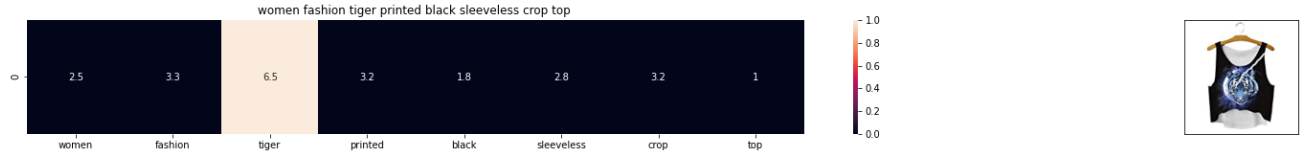
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from the given image : 14.833392966672909
==========



ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from the given image : 14.898744516719225
==========

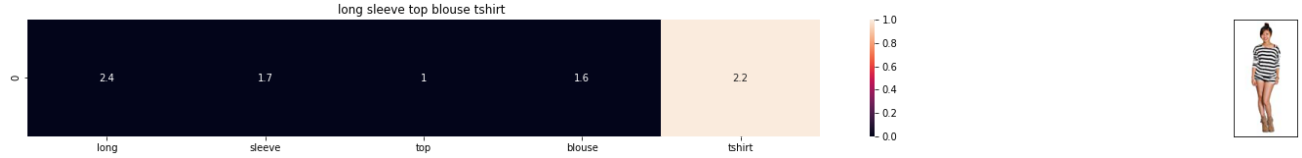


ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from the given image : 15.224458287343769
==========



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

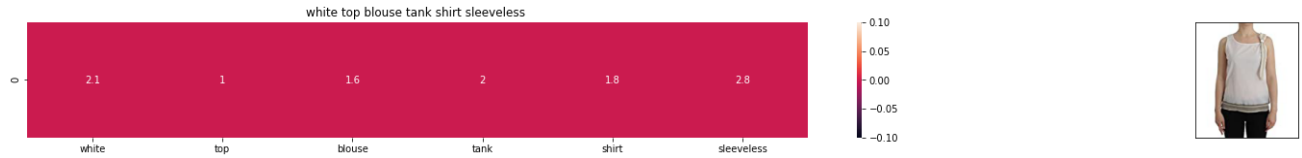
==========

ASIN : B074T9KG9Q
Brand : Rain
euclidean distance from the given image : 17.33671523874989

==========

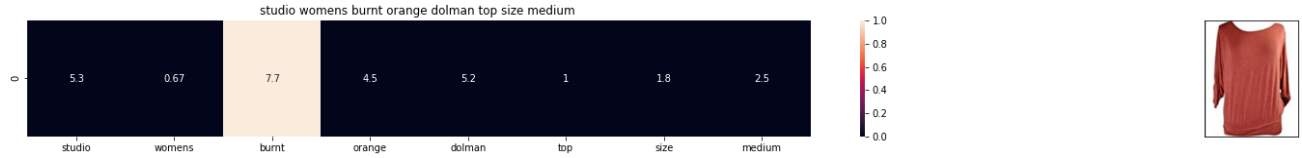
ASIN : B00H8A6ZLI
Brand : Vivian's Fashions
euclidean distance from the given image : 17.410075941001253

==========



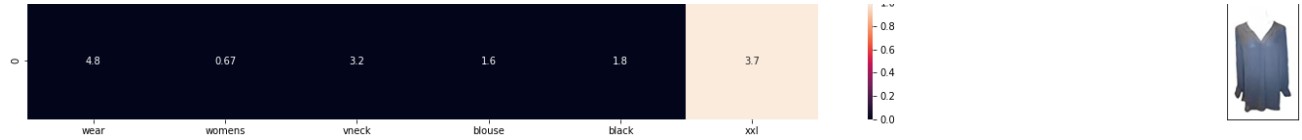
ASIN : B074G5G5RK
Brand : ERMANNO SCERVINO
euclidean distance from the given image : 17.539921335459557

==========

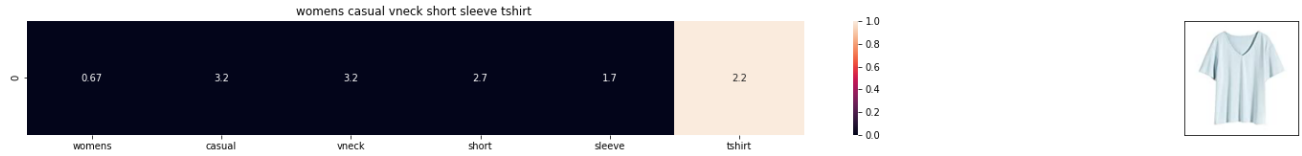


ASIN : B06XSCVFT5
Brand : Studio M
euclidean distance from the given image : 17.61275854366134

==========

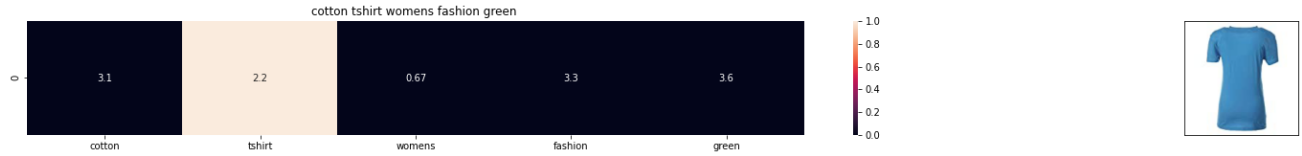


ASIN : B06Y6FH453
Brand : Who What Wear
euclidean distance from the given image : 17.623745282500135
==========

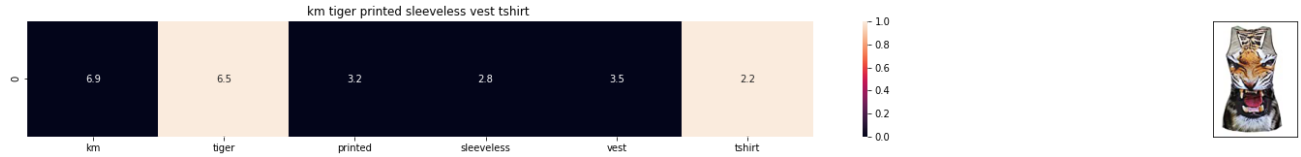


ASIN : B074V45DCX
Brand : Rain
euclidean distance from the given image : 17.634342496835046
==========

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

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
with open('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/word2vec_model', 'rb') as handle
    model = pickle.load(handle)
```

```
# Utility functions

def get_word_vec(sentence, doc_id, m_name):
    # sentence : title of the apparel
    # doc_id: document id in our corpus
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
    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]]
            elif m_name == 'avg':
                vec.append(model[i])
        else:
```

```
        else:
            # if the word in our courpus is not there in the google word2vec corpus, we ar
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 ) 300 = len(w
    # each row represents the word2vec representation of each word (weighted/avg) in given
    return  np.array(vec)

def get_distance(vec1, vec2):
    # vec1 = np.array(#number_of_words_title1 * 300), each row is a vector of length 300 c
    # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length 300 c

    final_dist = []
    # for each vector in vec1 we caluclate the distance(euclidean) to all vectors in vec2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result the euclidean distance between vectors i, j
            dist.append(np.linalg.norm(i-j))
        final_dist.append(np.array(dist))
    # final_dist = np.array(#number of words in title1 * #number of words in title2)
    # final_dist[i,j] = euclidean distance between vectors i, j
    return np.array(final_dist)


def heat_map_w2v(sentence1, sentence2, url, doc_id1, doc_id2, model):
    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended apparel
    # model: it can have two values, 1. avg 2. weighted

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg)
    s1_vec = get_word_vec(sentence1, doc_id1, model)
    #s2_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg)
    s2_vec = get_word_vec(sentence2, doc_id2, model)

    # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance between words i, j
    s1_s2_dist = get_distance(s1_vec, s2_vec)



    # devide whole figure into 2 parts 1st part displays heatmap 2nd part displays image o
    gs = gridspec.GridSpec(2, 2, width_ratios=[4,1],height_ratios=[2,1])
    fig = plt.figure(figsize=(15,15))

    ax = plt.subplot(gs[0])
    # ploting the heap map based on the pairwise distances
    ax = sns.heatmap(np.round(s1_s2_dist,4), annot=True)
    # set the x axis labels as recommended apparels title
    ax.set_xticklabels(sentence2.split())
    # set the y axis labels as input apparels title
    ax.set_yticklabels(sentence1.split())
    # set title as recommended apparels title
    ax.set_title(sentence2)
```

```
    ax.set_title(sentence2)

    ax = plt.subplot(gs[1])
    # we remove all grids and axis labels for image
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])
    display_img(url, ax, fig)

    plt.show()
```

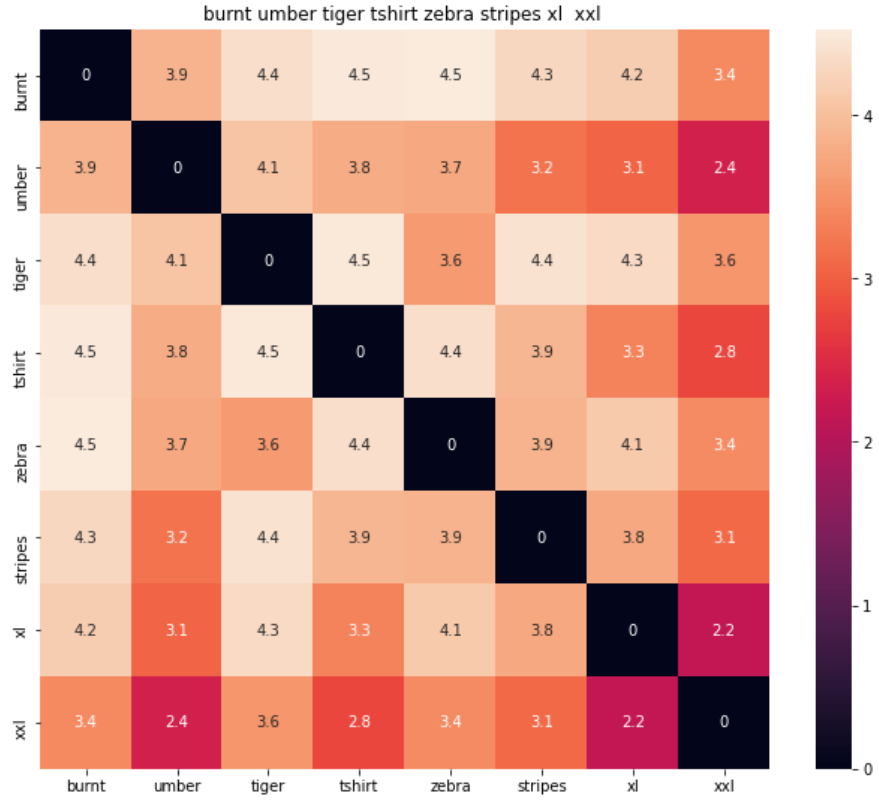
```
# vocab = stores all the words that are there in google w2v model
# vocab = model.wv.vocab.keys() # if you are using Google word2Vec

vocab = model.keys()
# this function will add the vectors of each word and returns the avg vector of given sent
def build_avg_vec(sentence, num_features, doc_id, m_name):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vector, its values = 300
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)

    featureVec = np.zeros((num_features,), dtype="float32")
    # we will intialize a vector of size 300 with all zeros
    # we add each word2vec(wordi) to this fetureVec
    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_vecto
            elif m_name == 'avg':
                featureVec = np.add(featureVec, model[word])
    if(nwords>0):
        featureVec = np.divide(featureVec, nwords)
    # returns the avg vector of given sentance, its of shape (1, 300)
    return featureVec
```

```
doc_id = 0
w2v_title = []
# for every title we build a avg vector representation
for i in data['title']:
    w2v_title.append(build_avg_vec(i, 300, doc_id,'avg'))
    doc_id += 1

# w2v_title = np.array(# number of doc in courpus * 300), each row corresponds to a doc
w2v_title = np.array(w2v_title)
```

```
w2v_title.shape
```

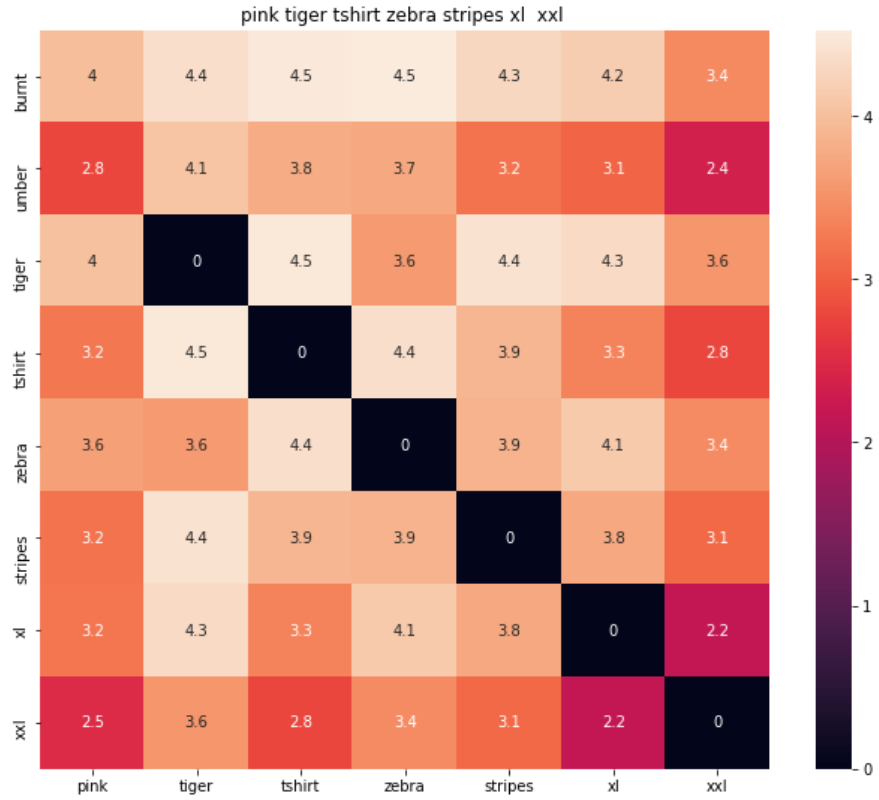
```
(16042, 300)
```

```
def avg_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # dist(x, y) = sqrt(dot(x, x) - 2 * dot(x, y) + dot(y, y))
    pairwise_dist = pairwise_distances(w2v_title, w2v_title[doc_id].reshape(1,-1))

    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]], da
        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)


avg_w2v_model(12566, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j
```



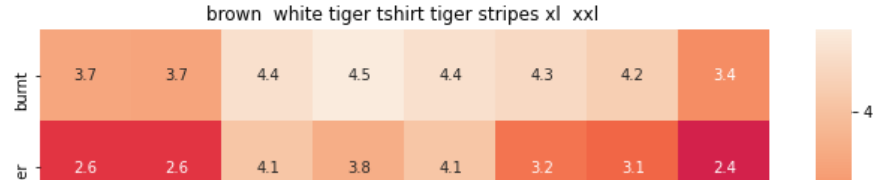
ASIN : B00JXQB5FQ
BRAND : Si Row
euclidean distance from given input image : 0.0

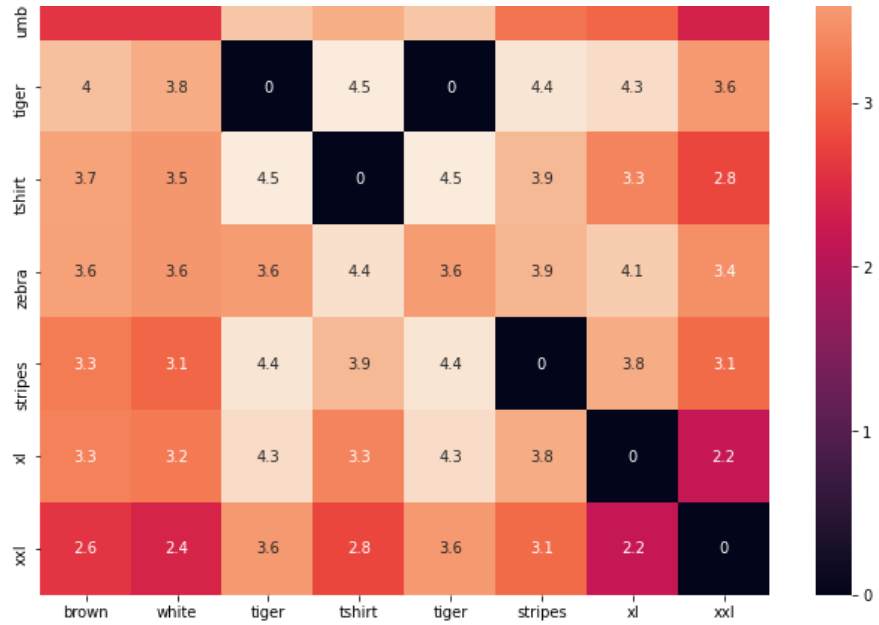
==========



ASIN : B00JXQASS6
BRAND : Si Row
euclidean distance from given input image : 0.5891926

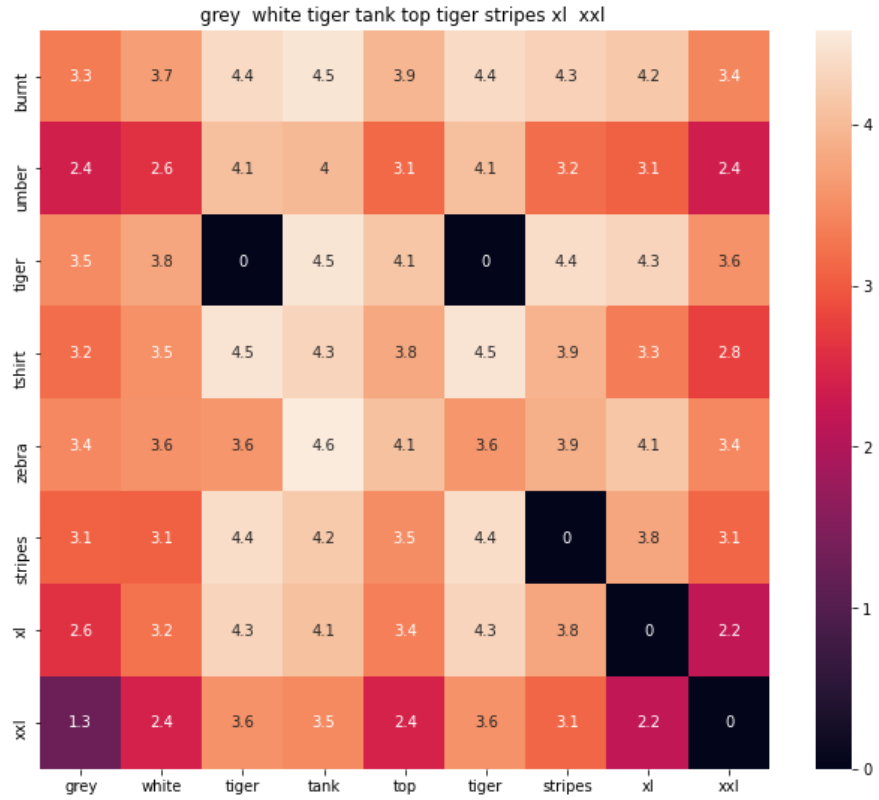
==========





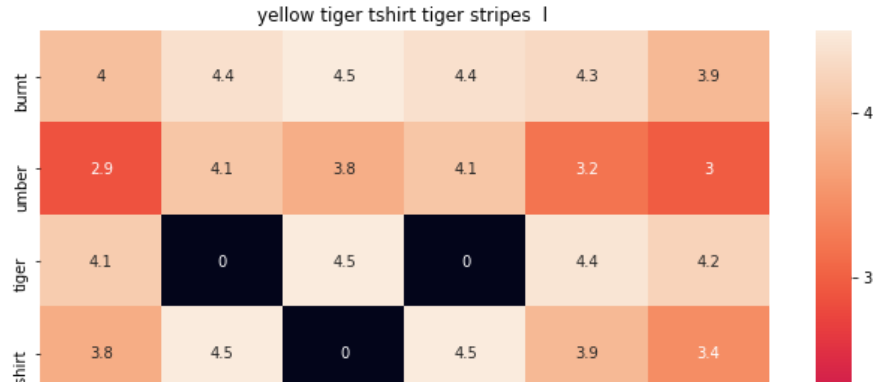
ASIN : B00JXQCWTO
BRAND : Si Row
euclidean distance from given input image : 0.7003438

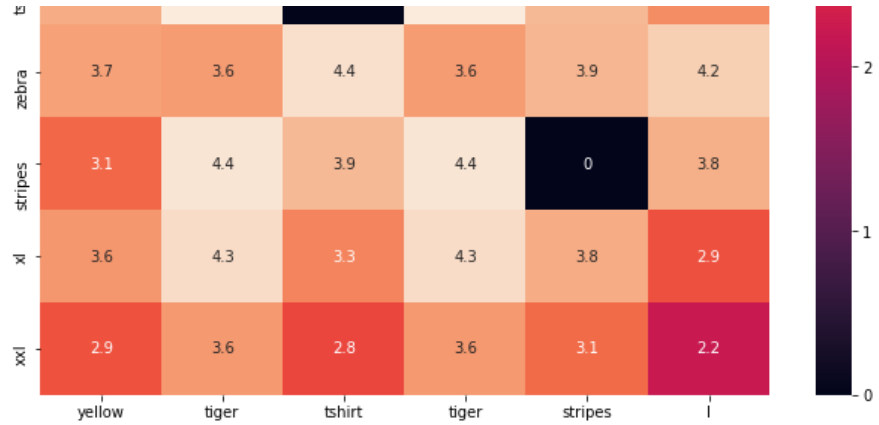
==========



ASIN : B00JXQAFZ2
BRAND : Si Row
euclidean distance from given input image : 0.89283955

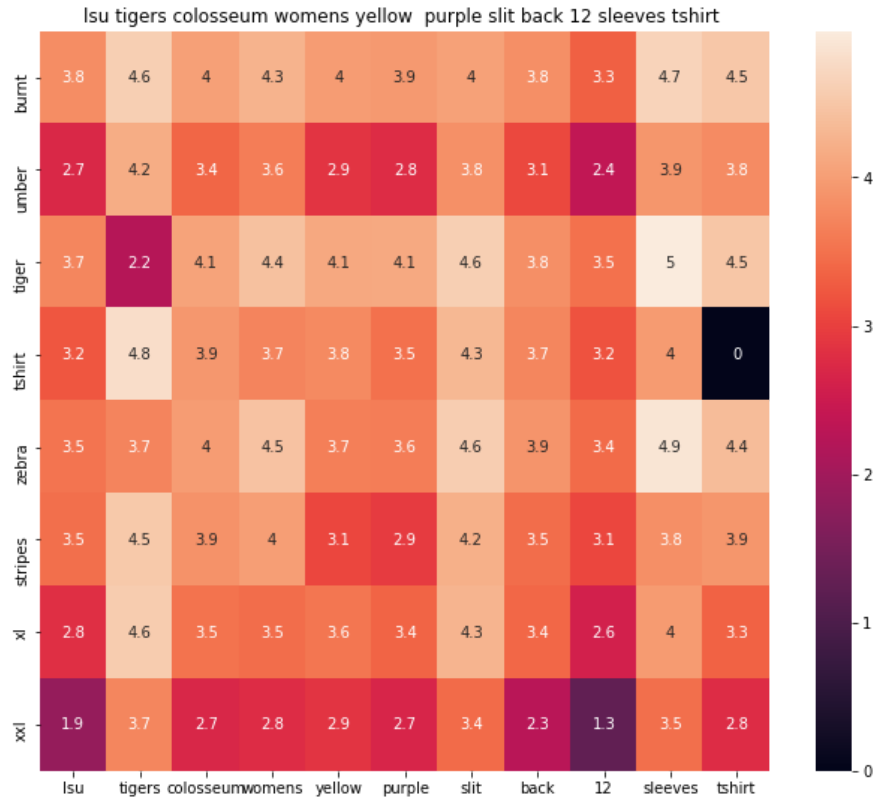
==========





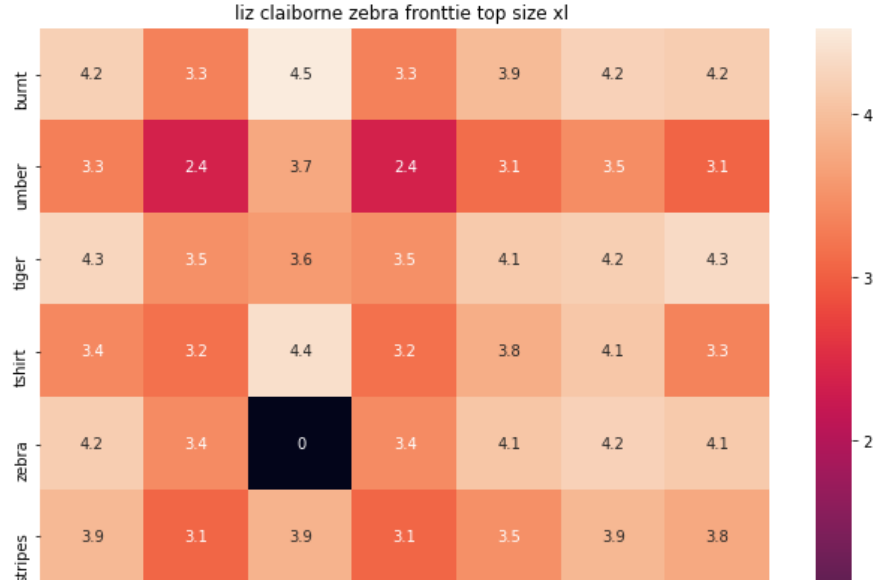
ASIN : B00JXQCUIC
BRAND : Si Row
euclidean distance from given input image : 0.95601255

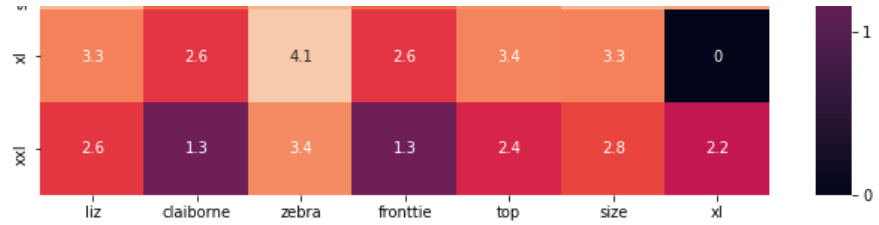
==========



ASIN : B073R5Q8HD
BRAND : Colosseum
euclidean distance from given input image : 1.022969

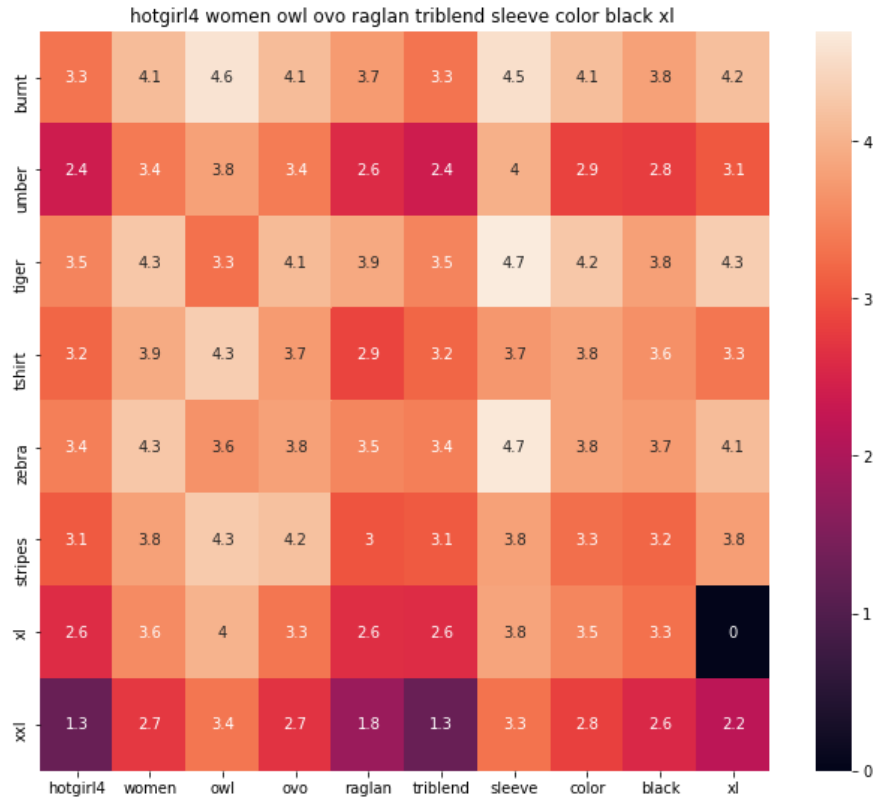
==========





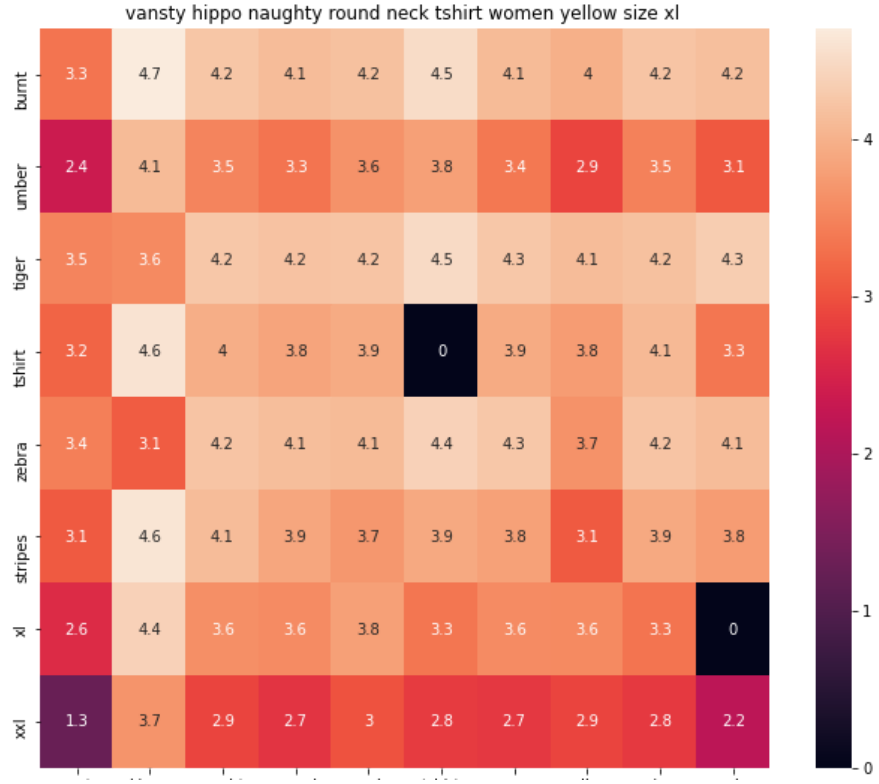
ASIN : B06XBY5QXL
BRAND : Liz Claiborne
euclidean distance from given input image : 1.0669324

==========



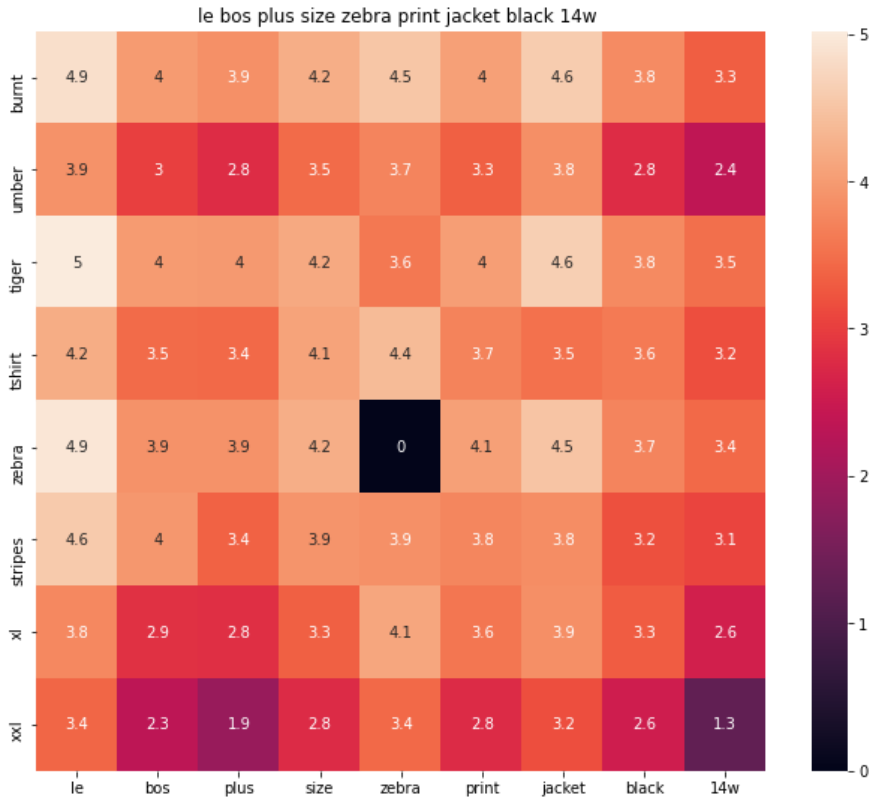
ASIN : B01L8L73M2
BRAND : Hotgirl4 Raglan Design
euclidean distance from given input image : 1.0731405

==========



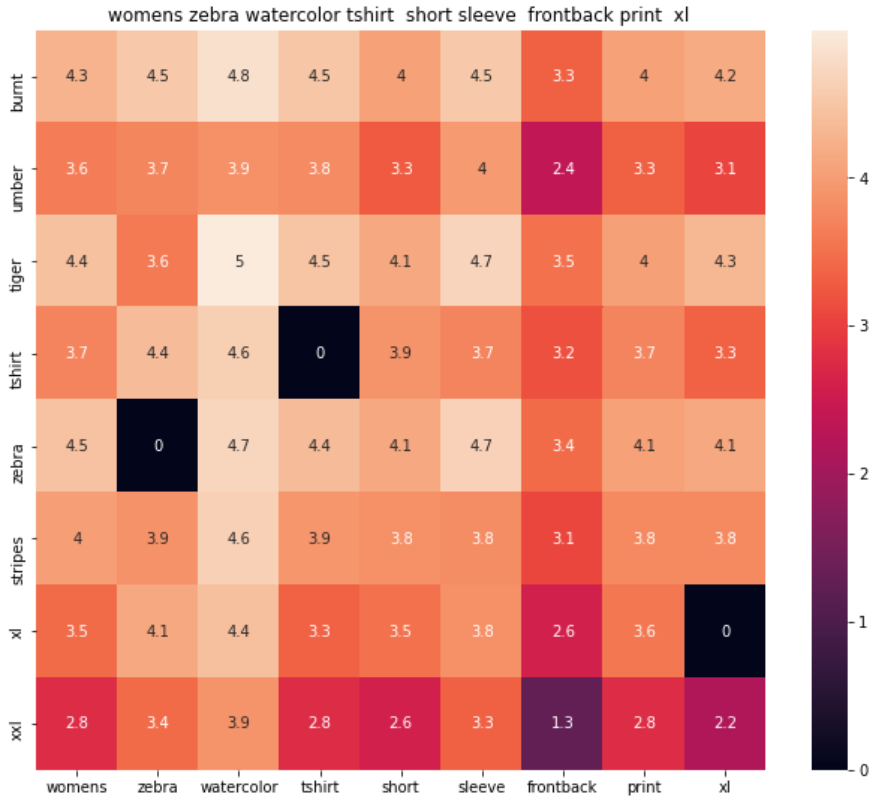
ASIN : B01EJS5H06
BRAND : Vansty
euclidean distance from given input image : 1.075719

==========



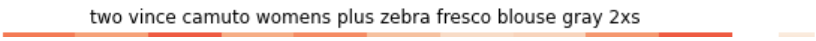
ASIN : B01BO1XRK8
BRAND : Le Bos
euclidean distance from given input image : 1.0839964

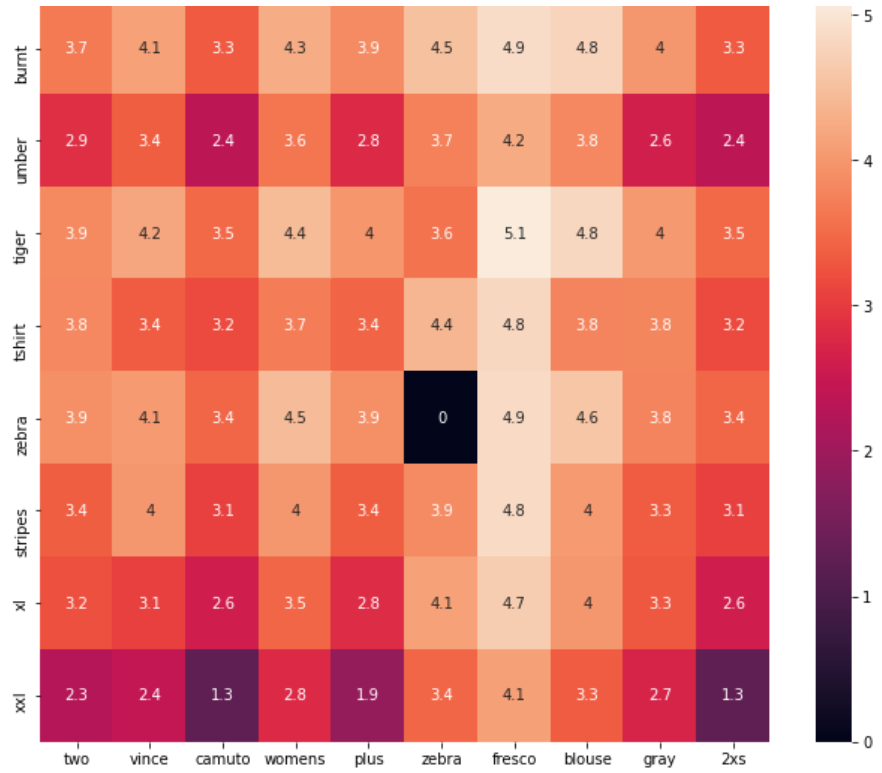
==========



ASIN : B072R2JXKW
BRAND : WHAT ON EARTH
euclidean distance from given input image : 1.0842218

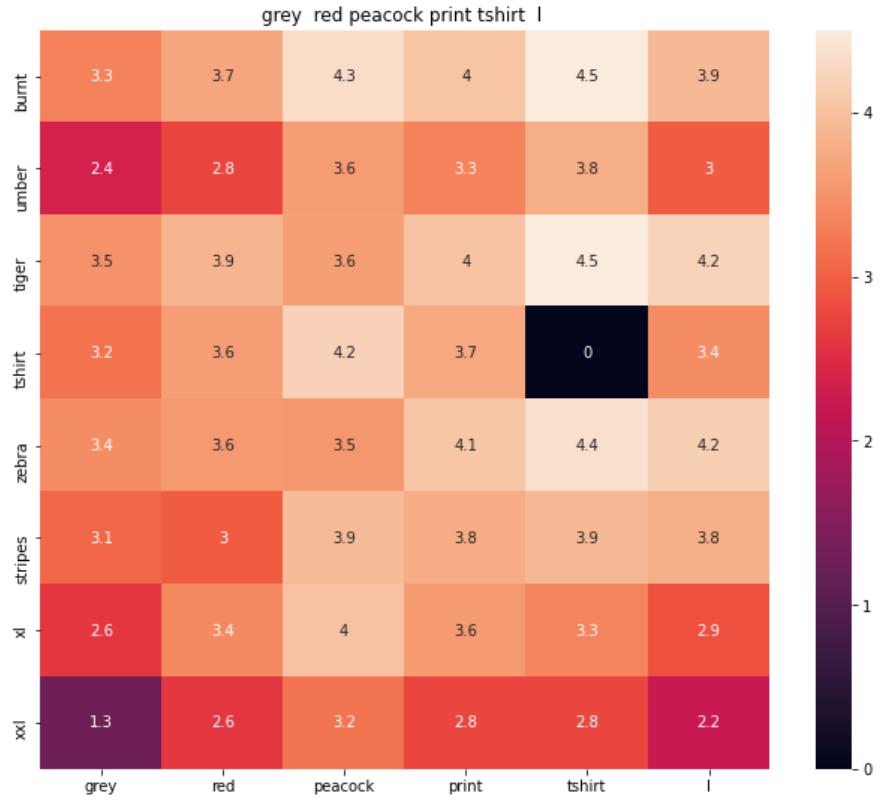
==========





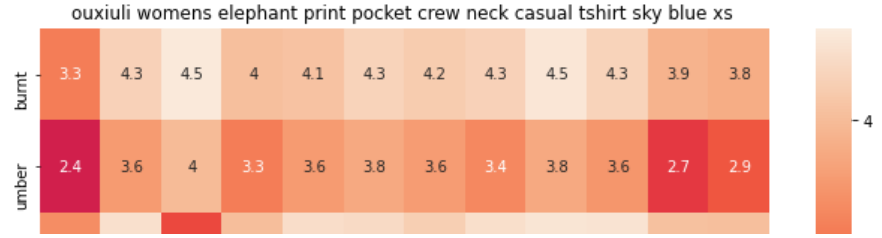
ASIN : B074MJRGW6
BRAND : Two by Vince Camuto
euclidean distance from given input image : 1.0895038

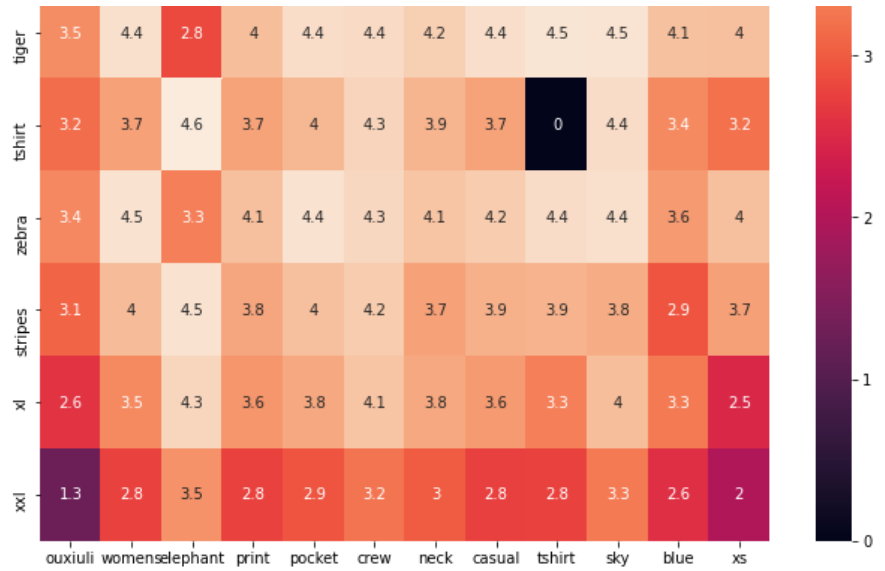
==========



ASIN : B00JXQCFRS
BRAND : Si Row
euclidean distance from given input image : 1.0900588

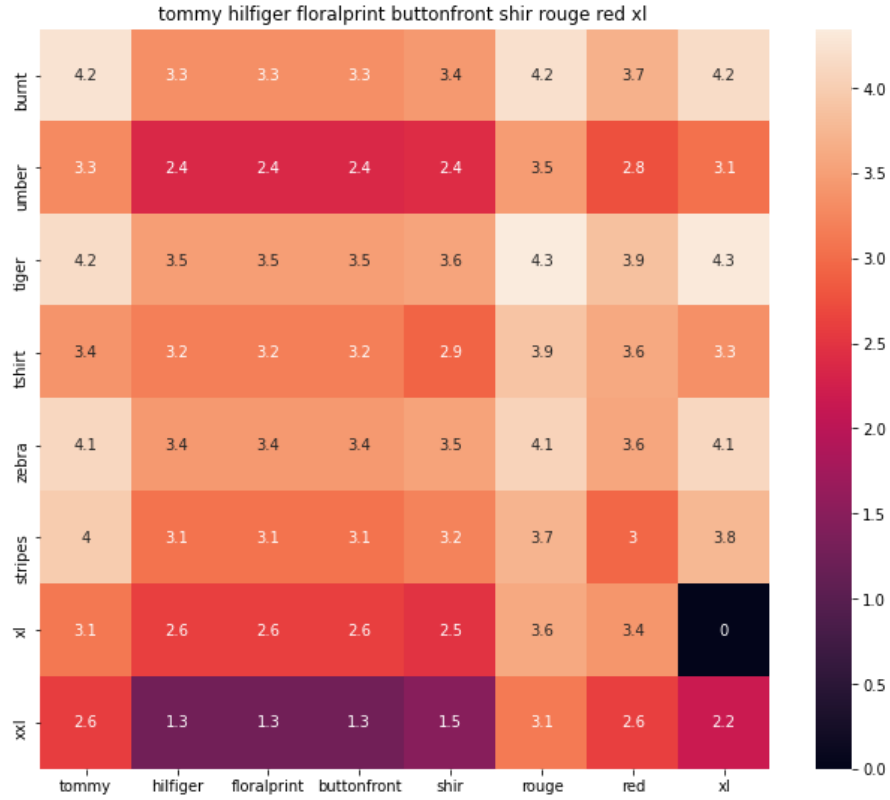
==========





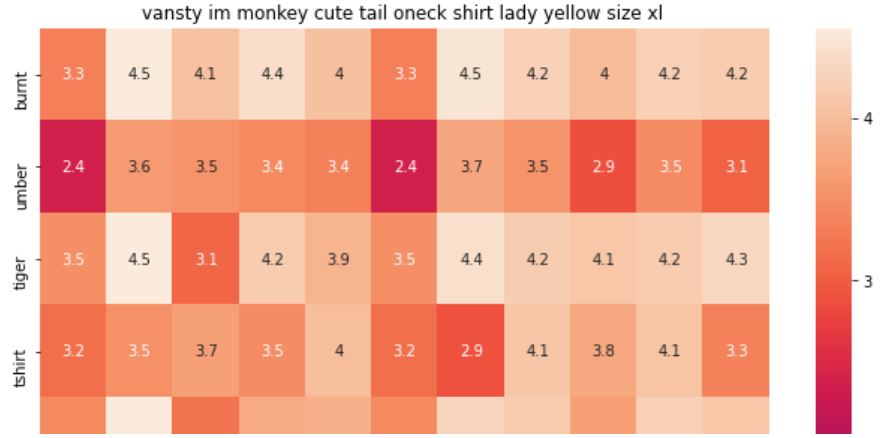
ASIN : B01I53HU6K
BRAND : ouxiuli
euclidean distance from given input image : 1.0920111

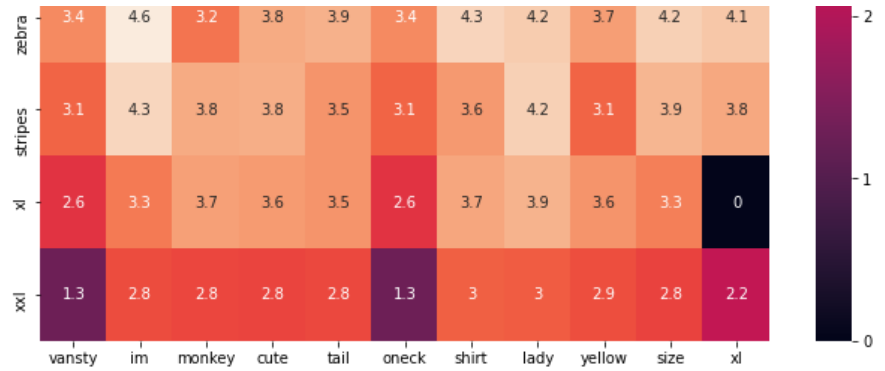
==========



ASIN : B0711NGTQM
BRAND : THILFIGER RTW
euclidean distance from given input image : 1.0923415

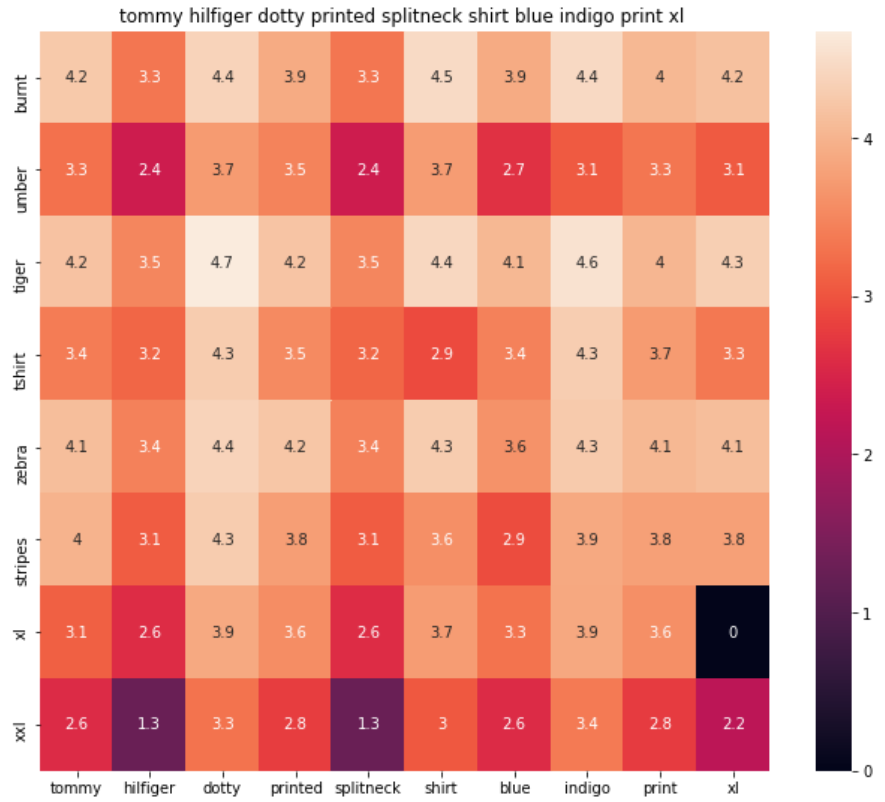
==========





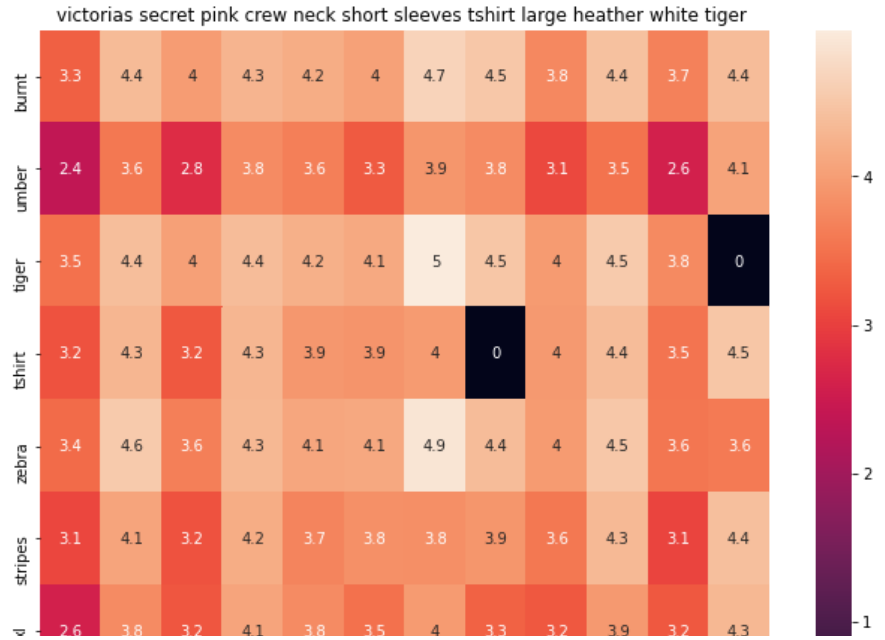
ASIN : B01EFSLO8Y
BRAND : Vansty
euclidean distance from given input image : 1.0934004

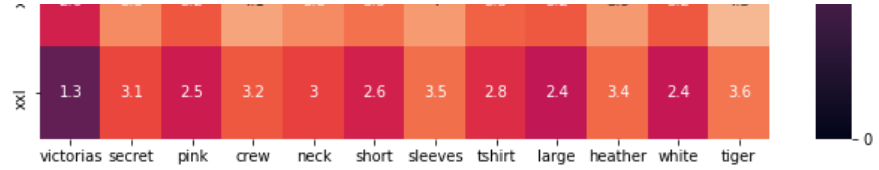
==========



ASIN : B0716TVWQ4
BRAND : THILFIGER RTW
euclidean distance from given input image : 1.0942024

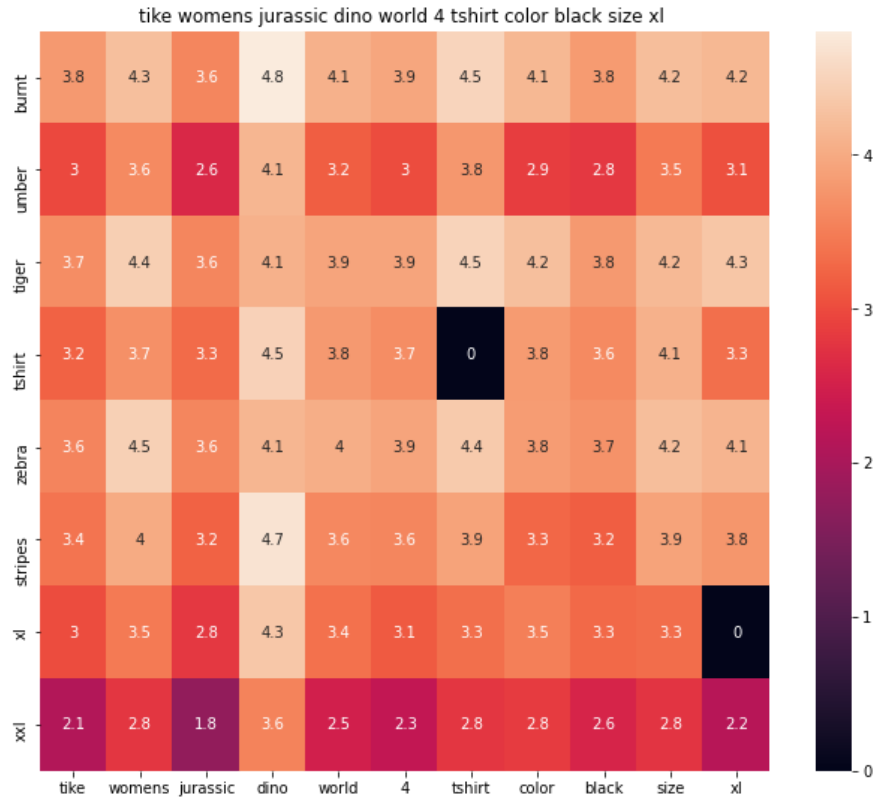
==========





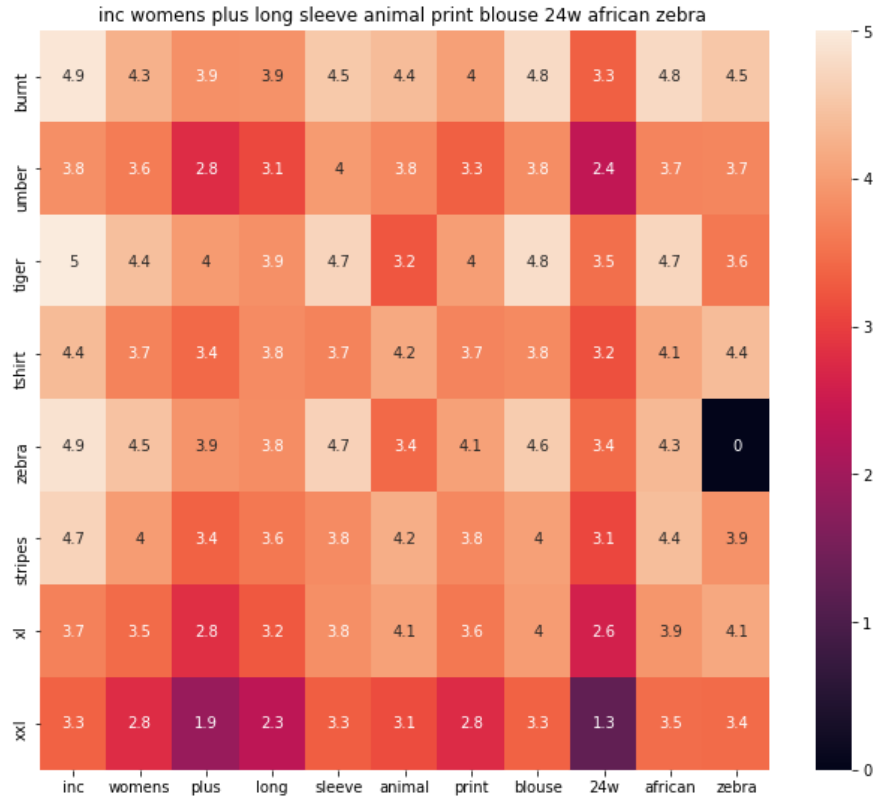
ASIN : B0716MVPGV
BRAND : V.Secret
euclidean distance from given input image : 1.0948304

==========



ASIN : B016OPN4OI
BRAND : TIKE Fashions
euclidean distance from given input image : 1.0951275

==========



ASIN : B018WDJCUA

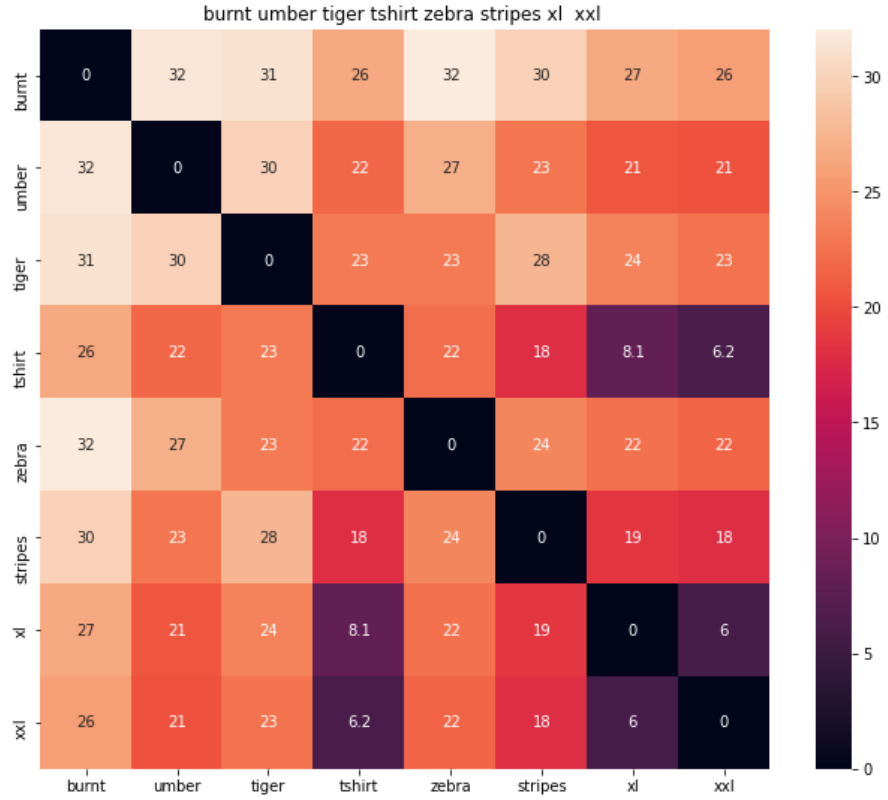
```
BRAND : INC - International Concepts Woman
euclidean distance from given input image : 1.0966892
==========================================================================================
```

```
doc_id = 0
w2v_title_weight = []
# for every title we build a weighted vector representation
for i in data['title']:
    w2v_title_weight.append(build_avg_vec(i, 300, doc_id,'weighted'))
    doc_id += 1
# w2v_title = np.array(# number of doc in courpus * 300), each row corresponds to a doc
w2v_title_weight = np.array(w2v_title_weight)
```

```
w2v_title_weight.shape
```

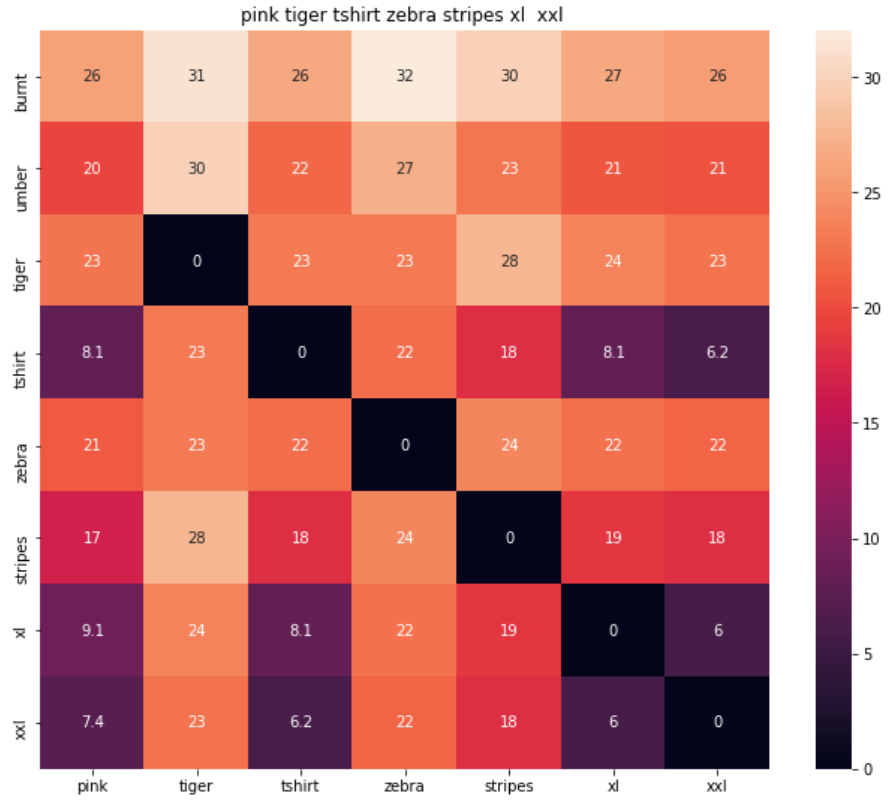
```
(16042, 300)
```

```
def weighted_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining appa
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    pairwise_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(

    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(data['title'].loc[df_indices[0]],data['title'].loc[df_indices[i]], da
        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)

weighted_w2v_model(12566, 20)
#931
#12566
# in the give heat map, each cell contains the euclidean distance between words i, j
```



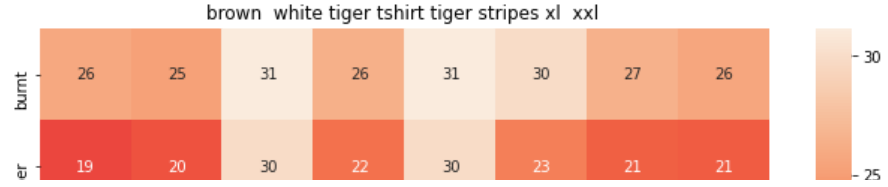
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0

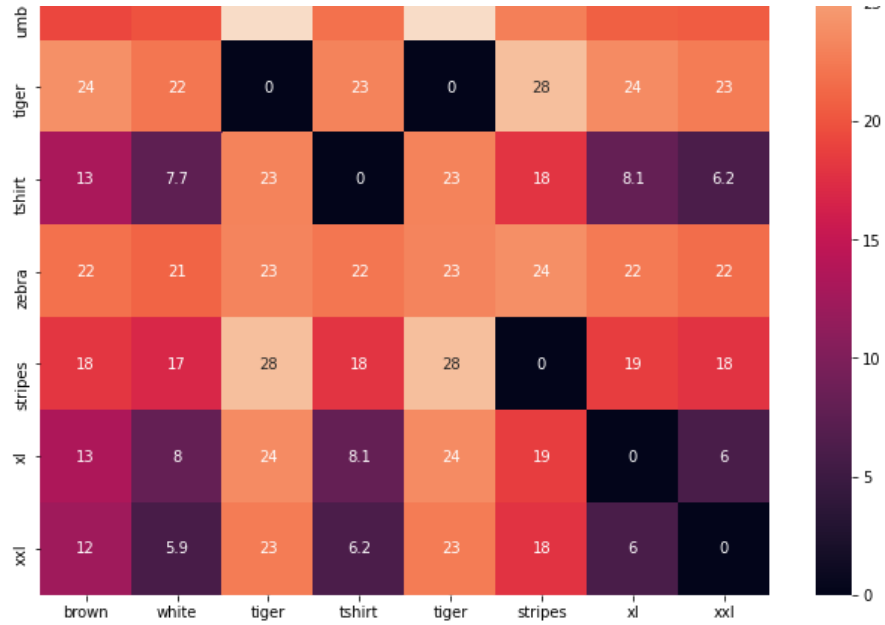
==========



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

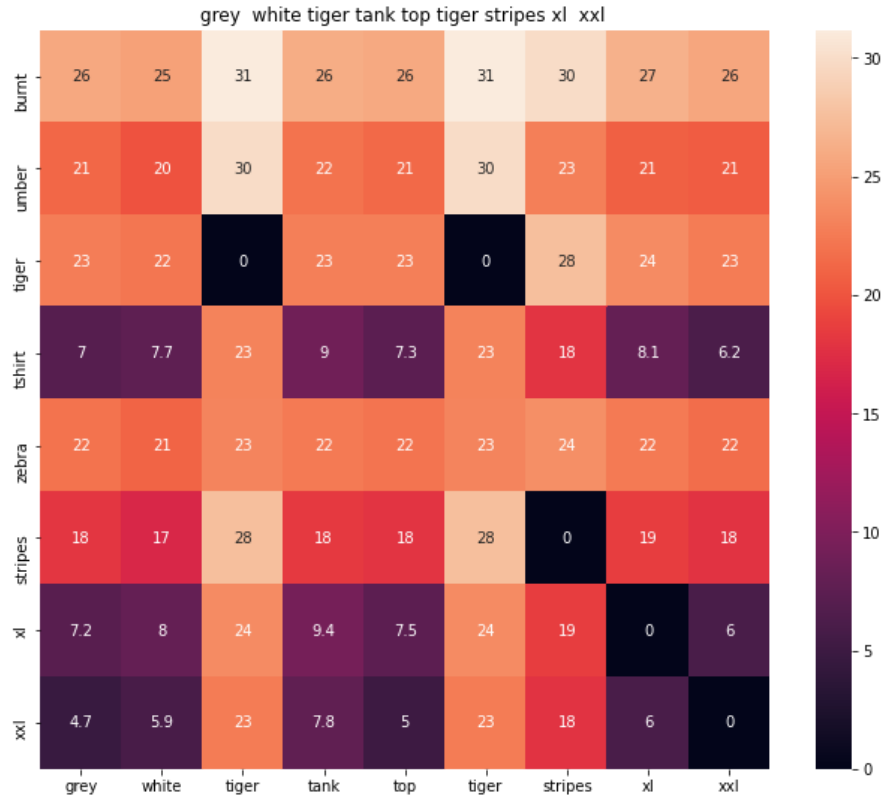
==========





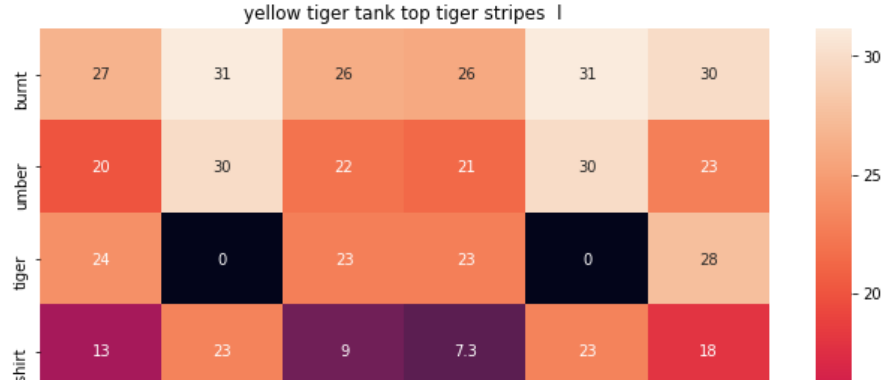
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 4.7709413

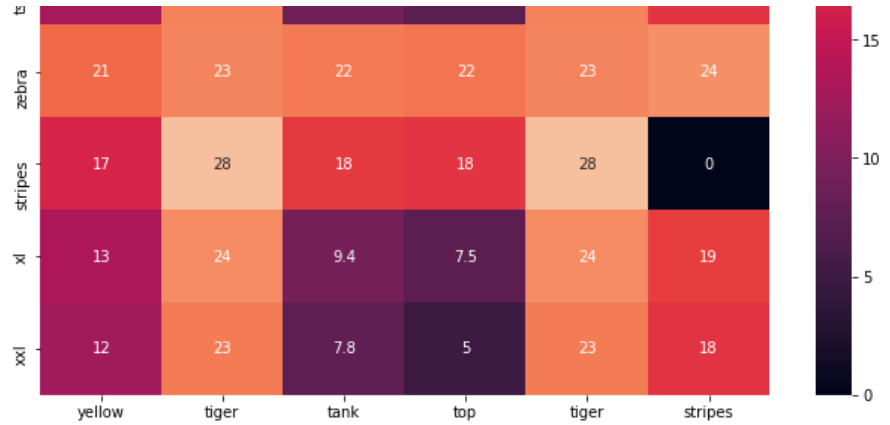
==========



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

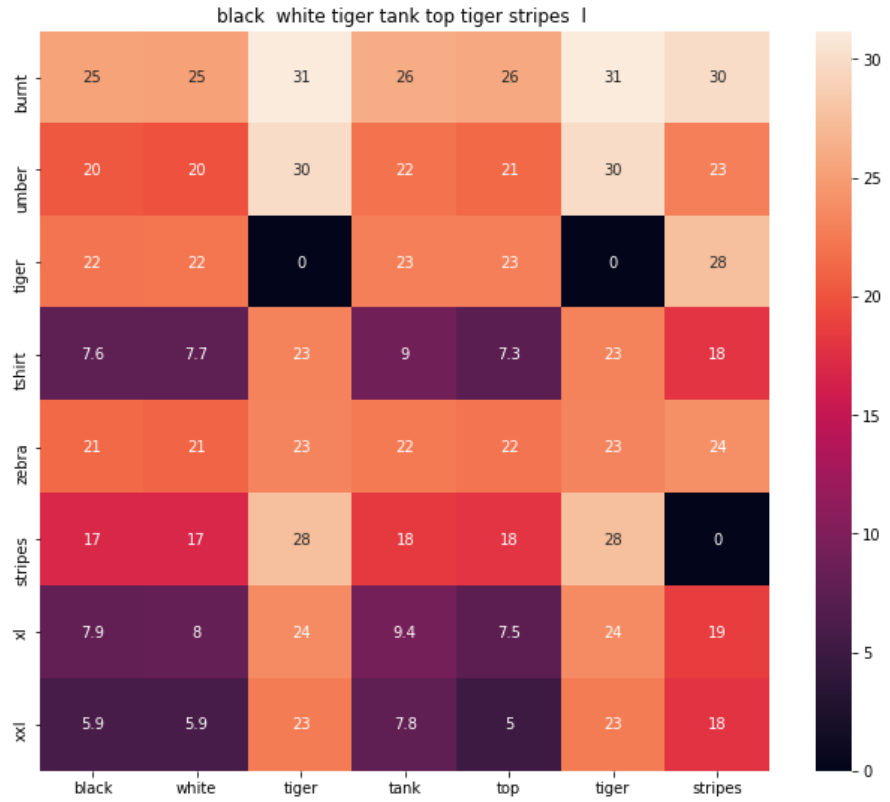
==========





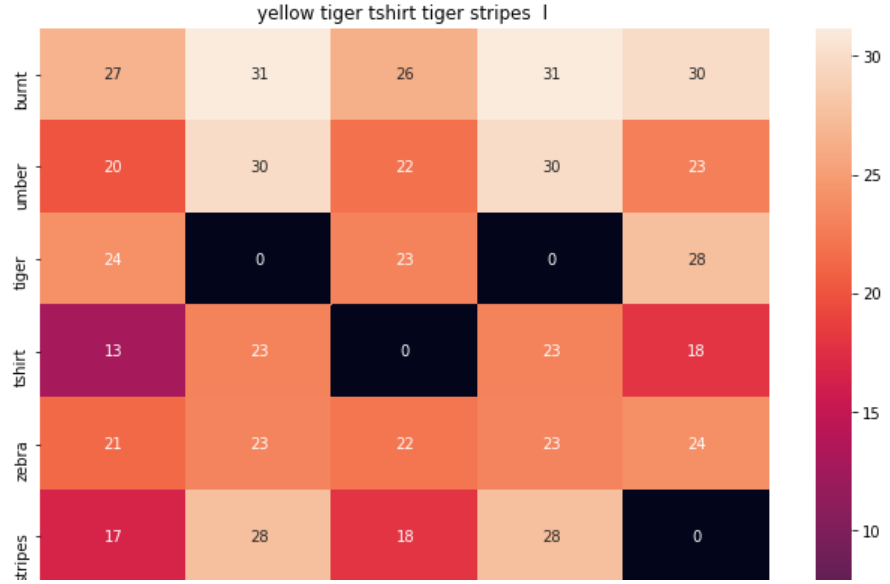
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 5.6895227

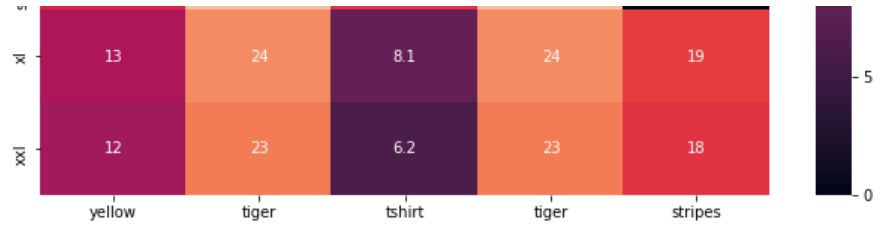
==========



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

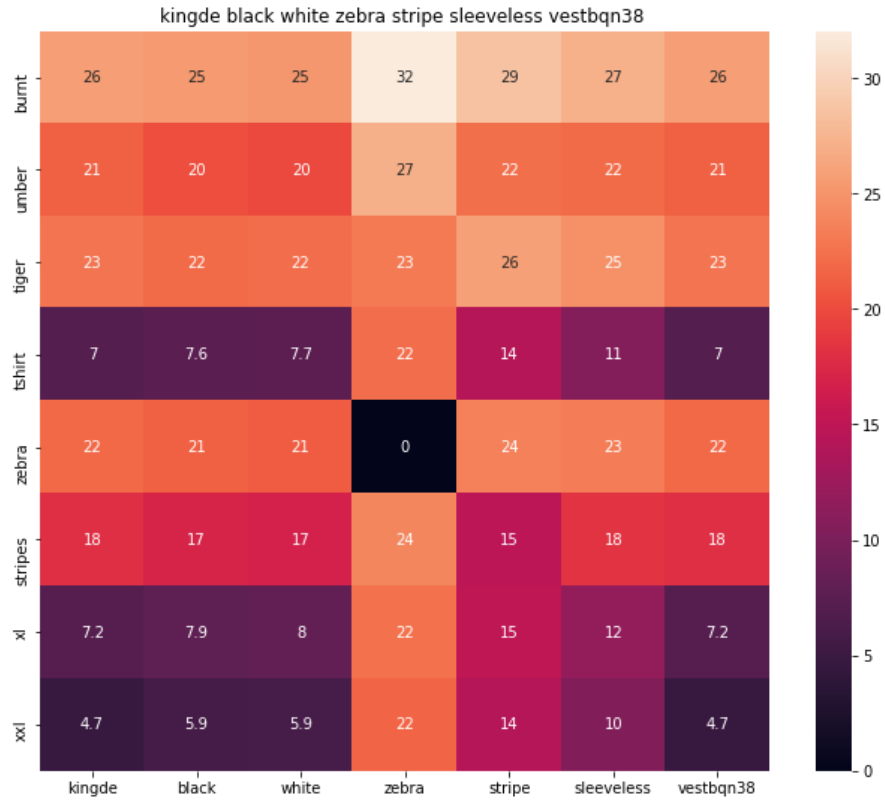
==========





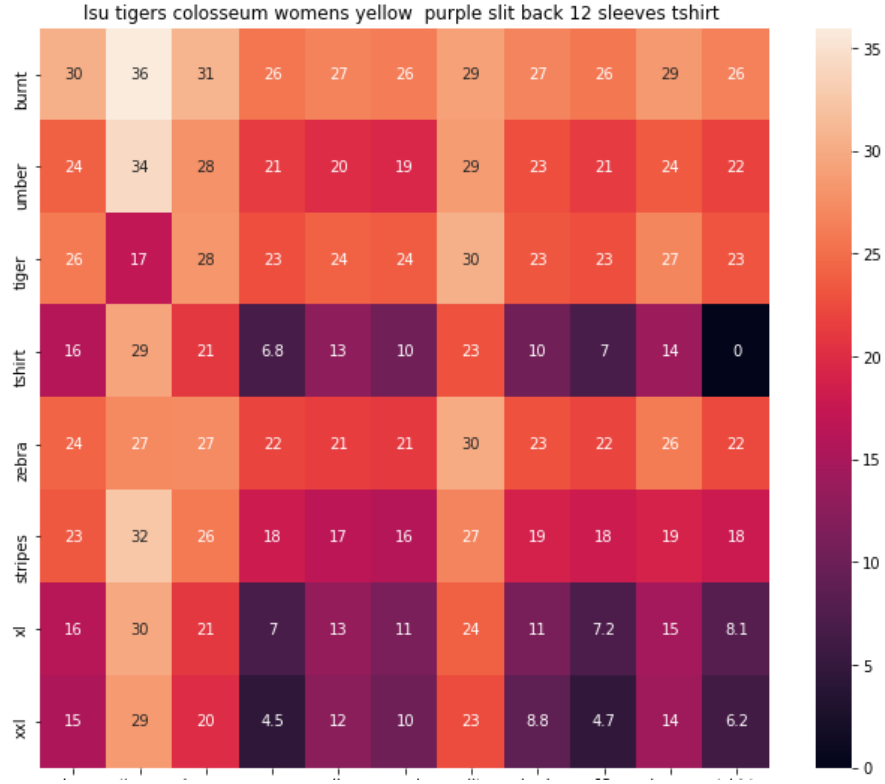
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 5.893442

==========



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

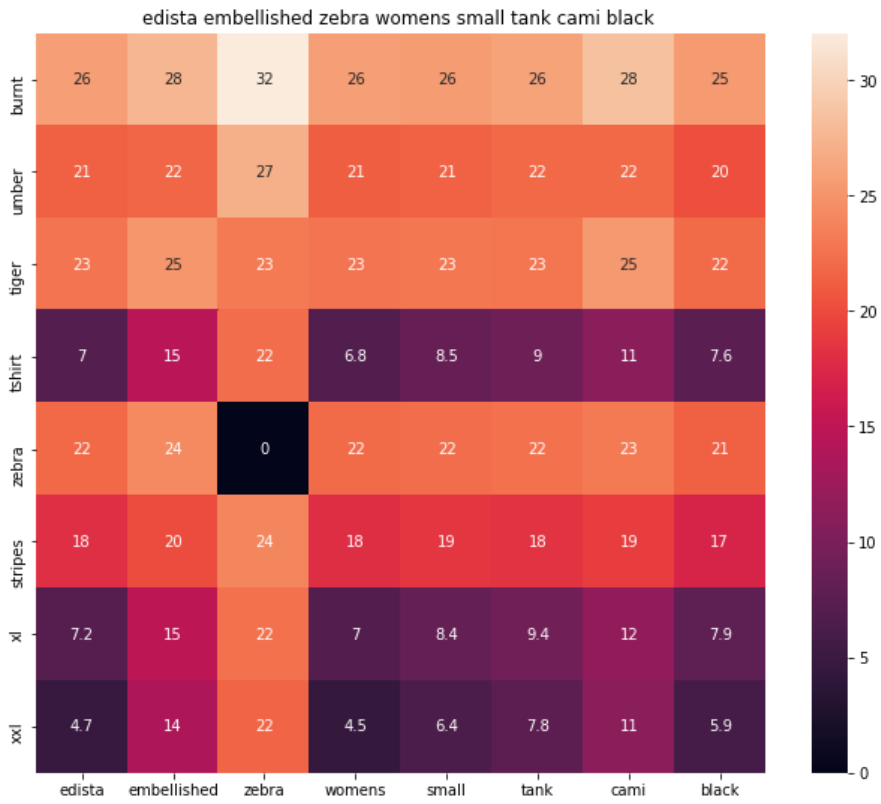
==========



lsu tigers colosseum womens yellow purple slit back 12 sleeves tshirt

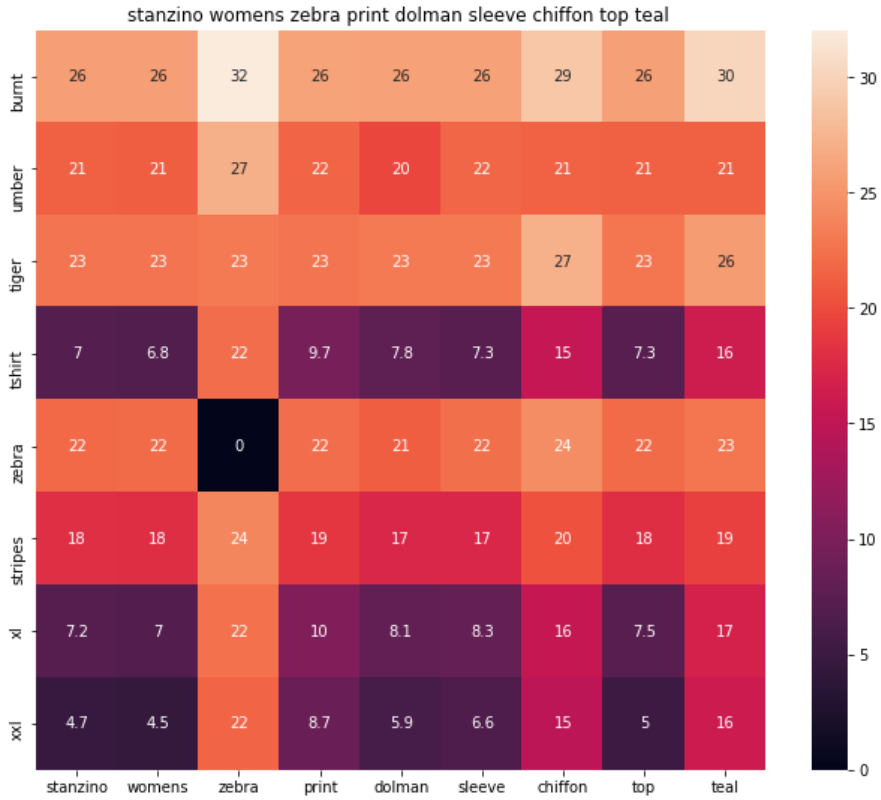
ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 6.2567053

==========



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

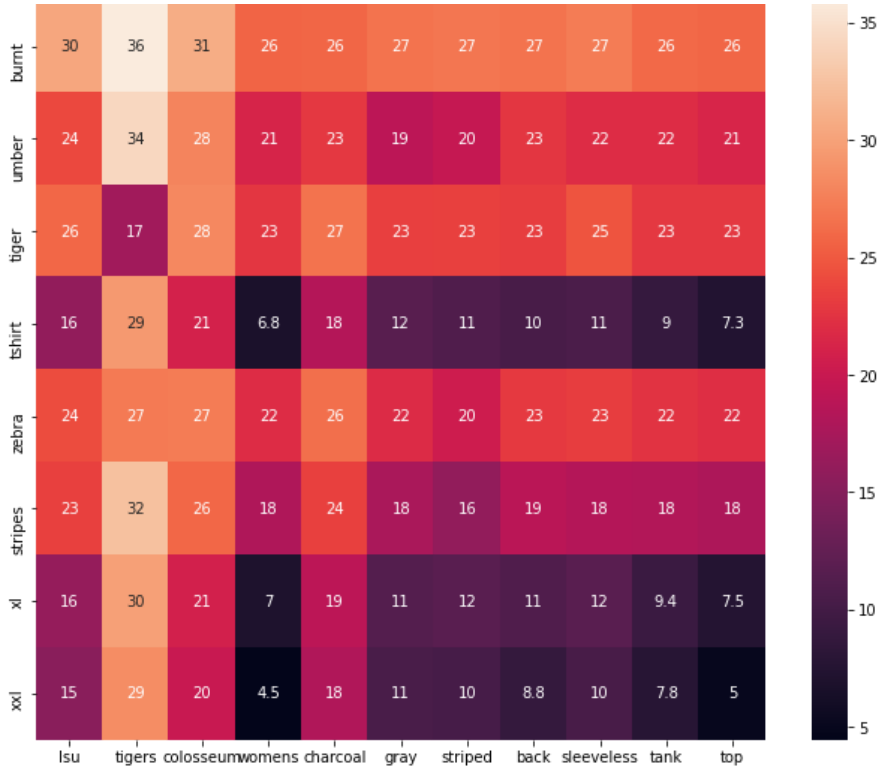
==========



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

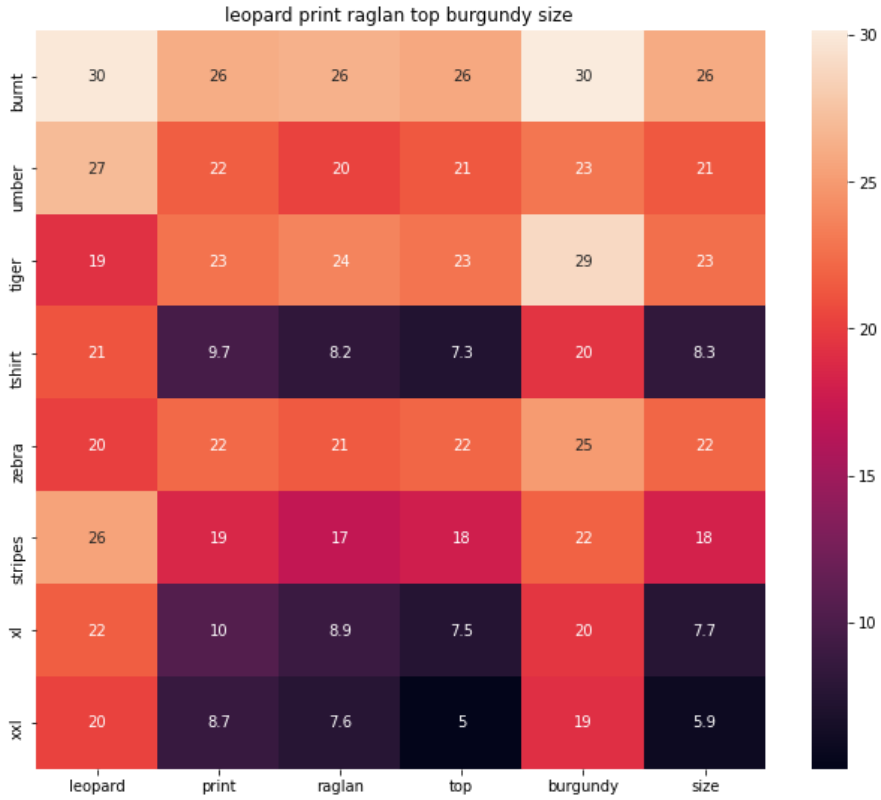
==========

lsu tigers colosseum womens charcoal gray striped back sleeveless tank top



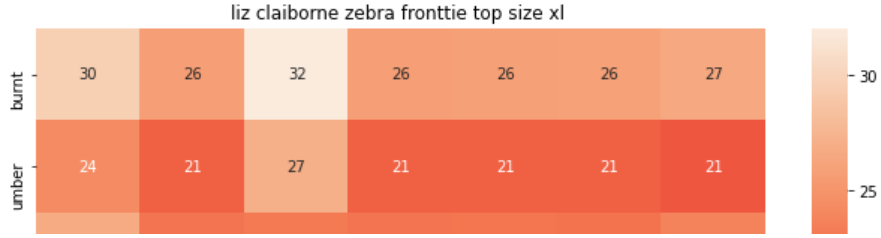
ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 6.450959

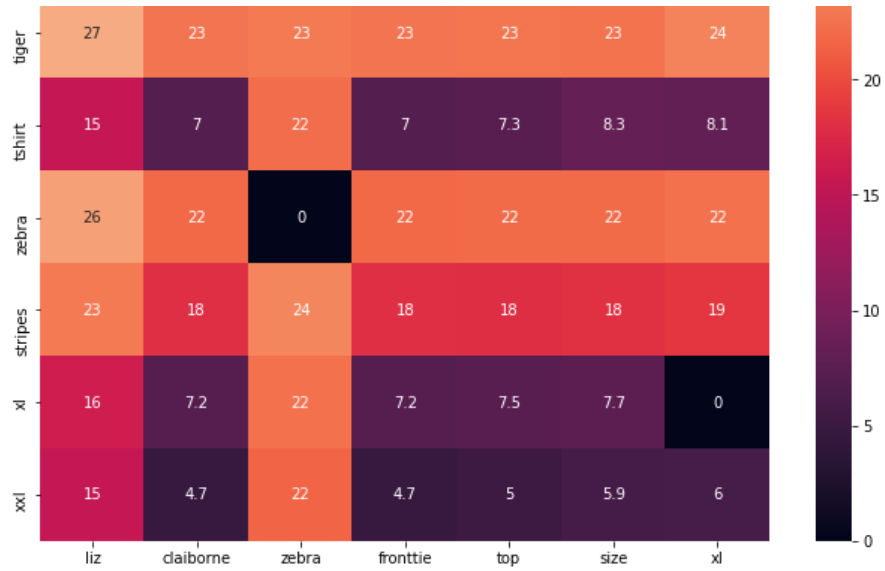
==========



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

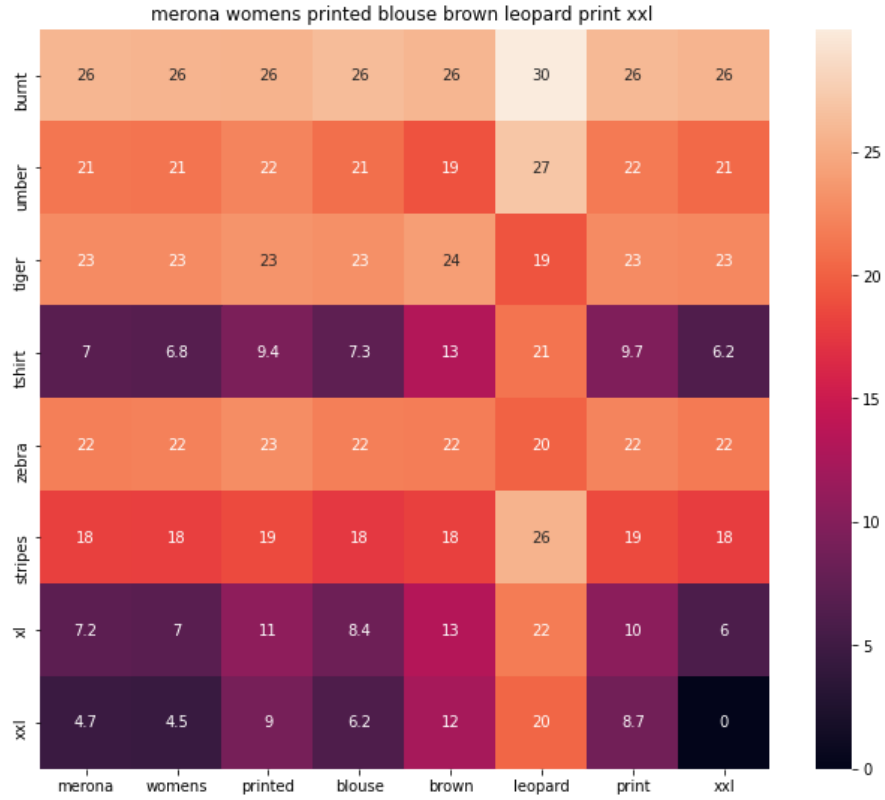
==========





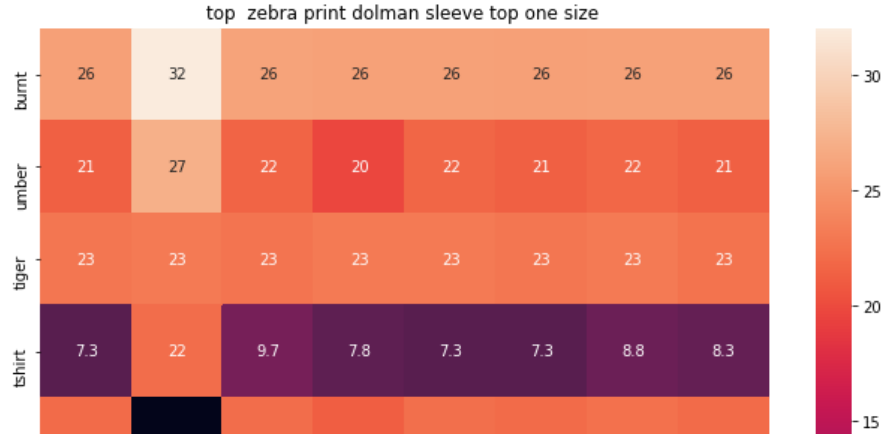
ASIN : B06XBY5QXL
Brand : Liz Claiborne
euclidean distance from input : 6.5392227

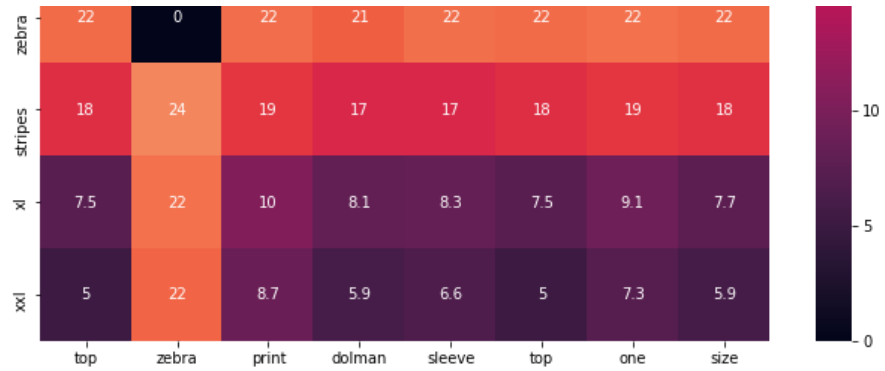
==========



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

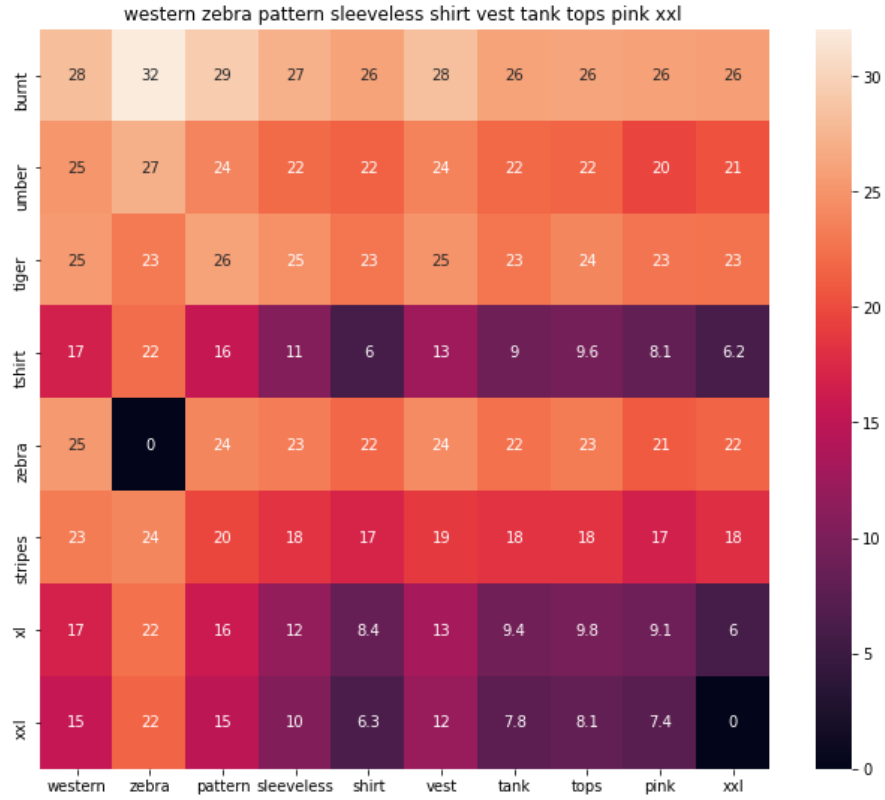
==========





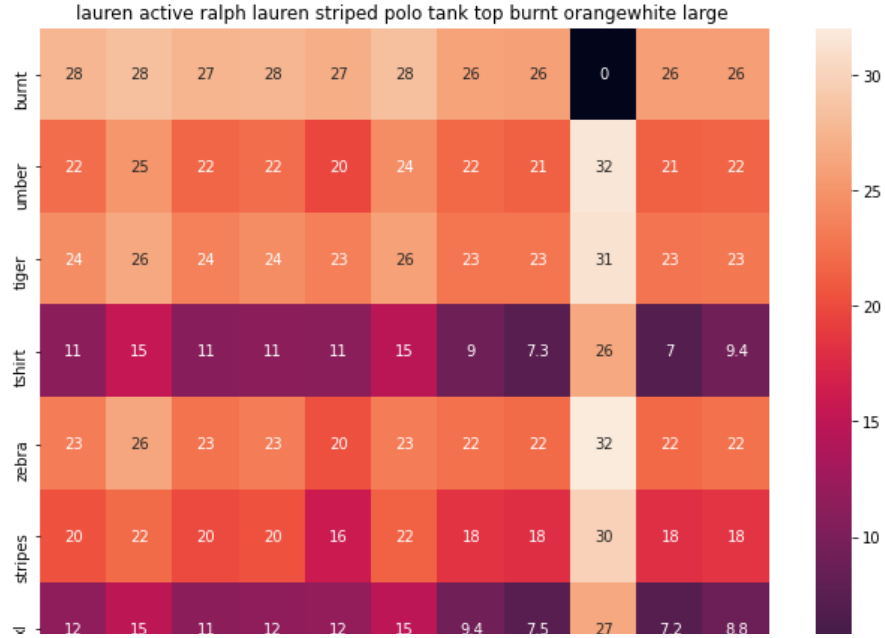
ASIN : B00H8A6ZLI
Brand : Vivian's Fashions
euclidean distance from input : 6.6382146

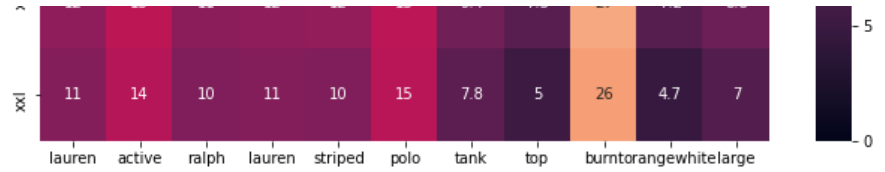
==========



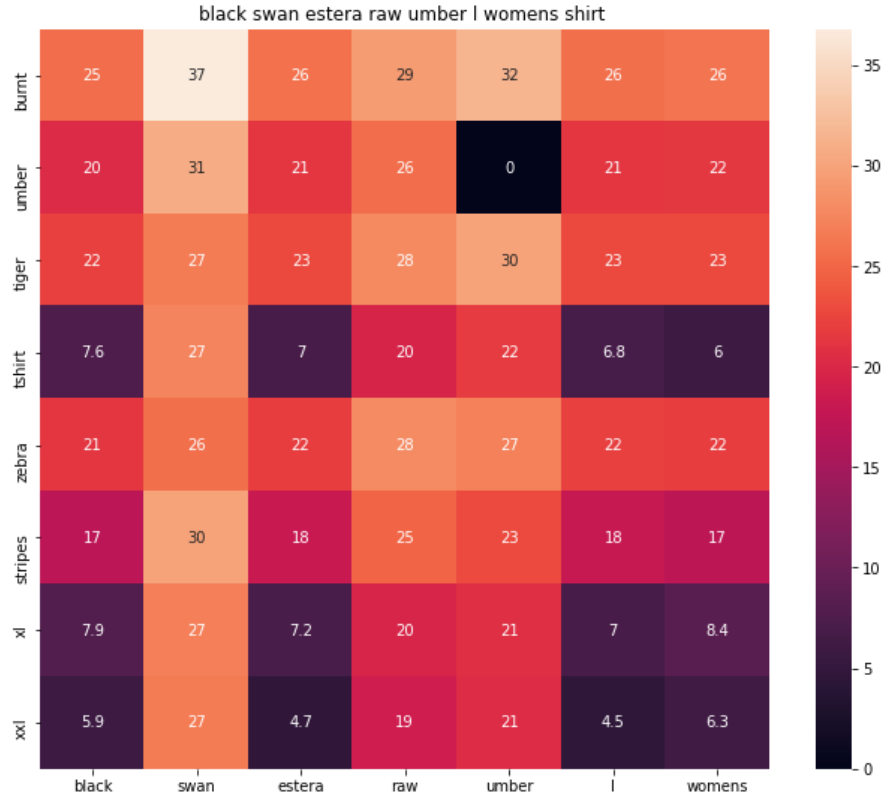
ASIN : B00Z6HEXWI
Brand : Black Temptation
euclidean distance from input : 6.6607366

==========

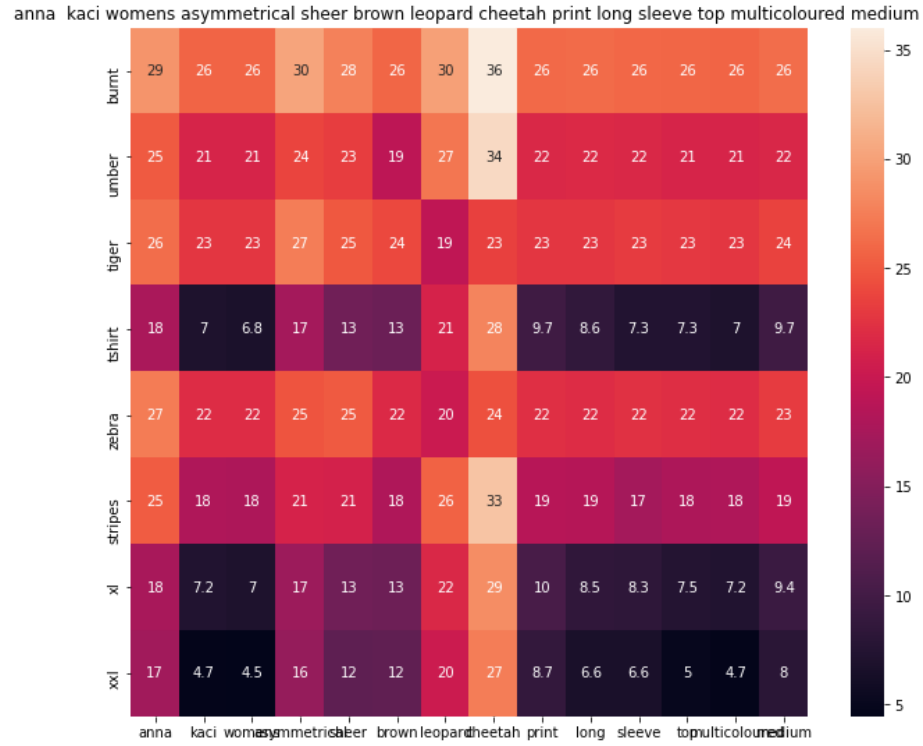




ASIN : B00ILGH5OY
Brand : Ralph Lauren Active
euclidean distance from input : 6.6839046
==========



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



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

```
Euclidean distance from input : 0.7661245
================================================================================
```

```
# some of the brand values are empty.
# Need to replace Null with string "NULL"
data['brand'].fillna(value="Not given", inplace=True )

# replace spaces with hypen
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()
```

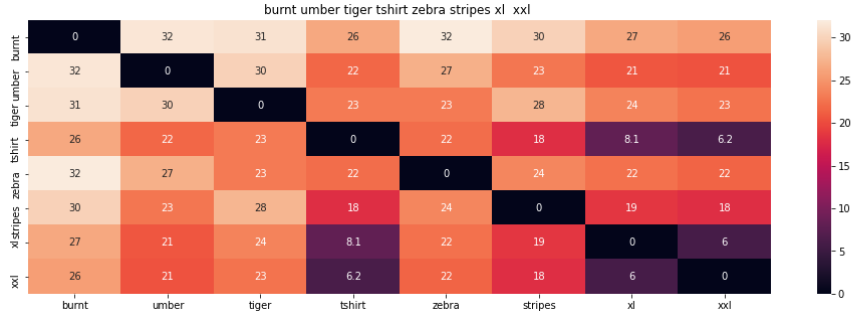
```
extra_features.shape
```

```
(16042, 5735)
```

```
def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, model)

    # sentance1 : title1, input apparel
    # sentance2 : title2, recommended apparel
    # url: apparel image url
    # doc_id1: document id of input apparel
    # doc_id2: document id of recommended apparel
    # df_id1: index of document1 in the data frame
    # df_id2: index of document2 in the data frame
    # model: it can have two values, 1. avg 2. weighted

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg)
    s1_vec = get_word_vec(sentance1, doc_id1, model)
    #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/avg)
    s2_vec = get_word_vec(sentance2, doc_id2, model)

    # s1_s2_dist = np.array(#number of words in title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance between words i, j
    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']] # to color the headings o

    # we create a table with the data_matrix
    table = ff.create_table(data_matrix, index=True, colorscale=colorscale)
    # plot it with plotly
    plotly.offline.iplot(table, filename='simple_table')

    # devide whole figure space into 25 * 1:10 grids
    gs = gridspec.GridSpec(25, 15)
    fig = plt.figure(figsize=(25,5))

    # in first 25*10 grids we plot heatmap
    ax1 = plt.subplot(gs[:, :-5])
    # ploting the heap map based on the pairwise distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), annot=True)
    # set the x axis labels as recommended apparels title
    ax1.set_xticklabels(sentance2.split())
    # set the y axis labels as input apparels title
    ax1.set_yticklabels(sentance1.split())
    # set title as recommended apparels title
    ax1.set_title(sentance2)

    # in last 25 * 10:15 grids we display image
    ax2 = plt.subplot(gs[:, 10:16])
    # we dont display grid lins and axis labels to images
    ax2.grid(False)
    ax2.set_xticks([])
    ax2.set_yticks([])
```

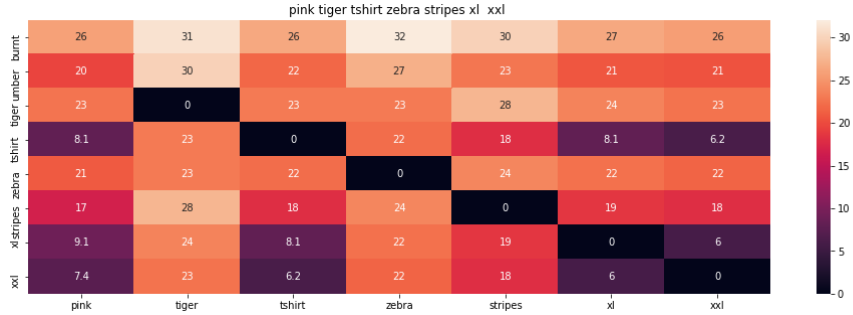
```
    # pass the url it display it
    display_img(url, ax2, fig)

    plt.show()
```

```
def idf_w2v_brand(doc_id, w1, w2, w3, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    # pairwise_dist will store the distance from given input apparel to all remaining appa
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(
    brand_feat_dist = pairwise_distances(brand_features, brand_features[doc_id])
    color_feat_dist = pairwise_distances(color_features, color_features[doc_id])

    pairwise_dist   = (w1 * idf_w2v_dist +  w2 * brand_feat_dist + w3*color_feat_dist)/flo

    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])


    for i in range(0, len(indices)):
        heat_map_w2v_brand(data['title'].loc[df_indices[0]],data['title'].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)

idf_w2v_brand(12566, 5, 3,2, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j
```



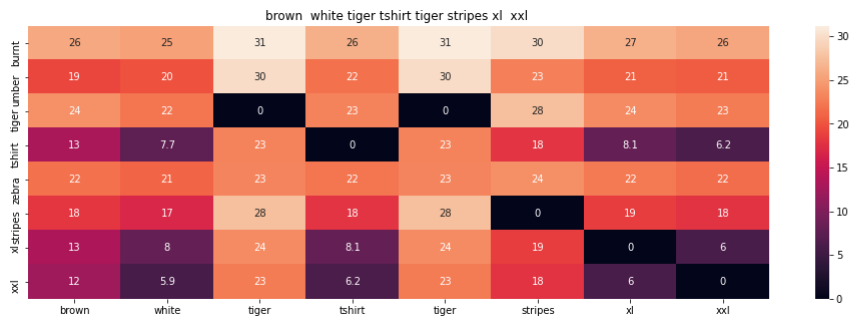
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0

==========

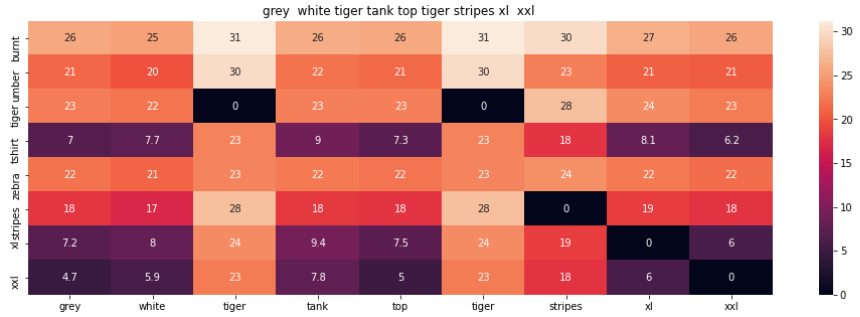


ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 2.3147860337026467

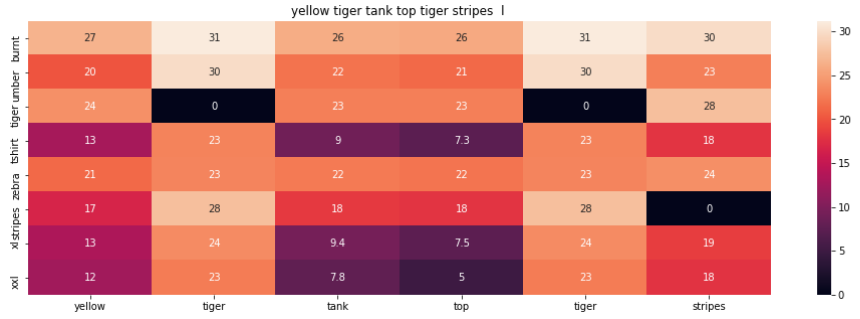
==========



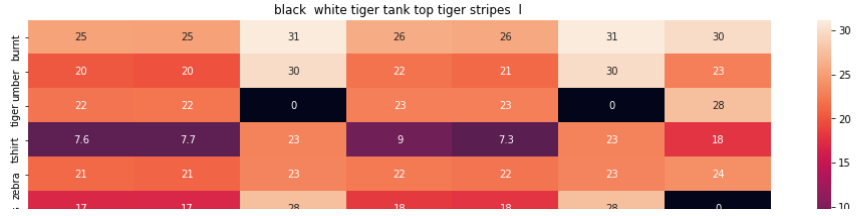
```
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 2.3854705810546877
==========================================================================================
```

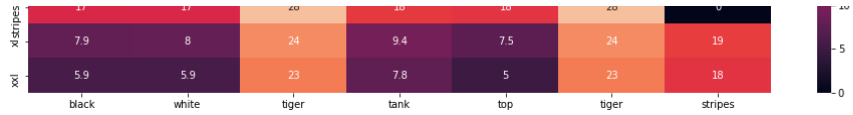


```
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 2.9629229355581153
==========================================================================================
```



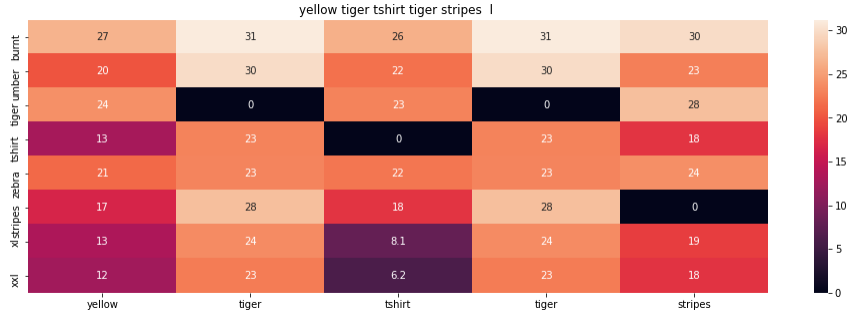
```
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 3.1276039887197364
==========================================================================================
```





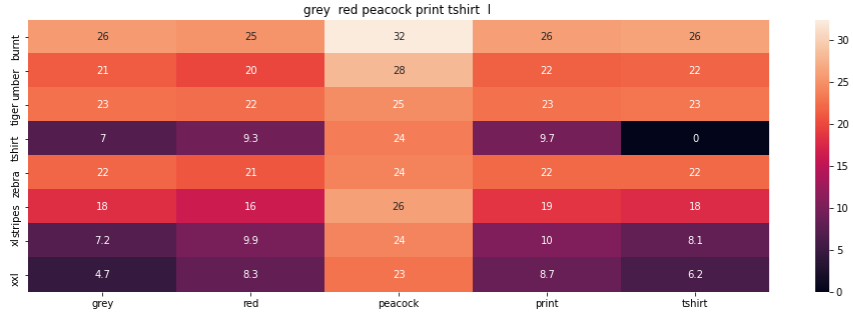
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 3.1293530274160255

==========



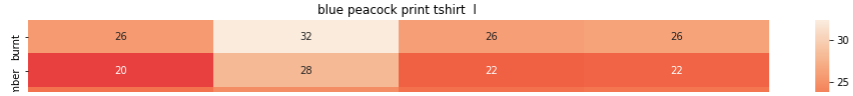
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 3.2295637894399514

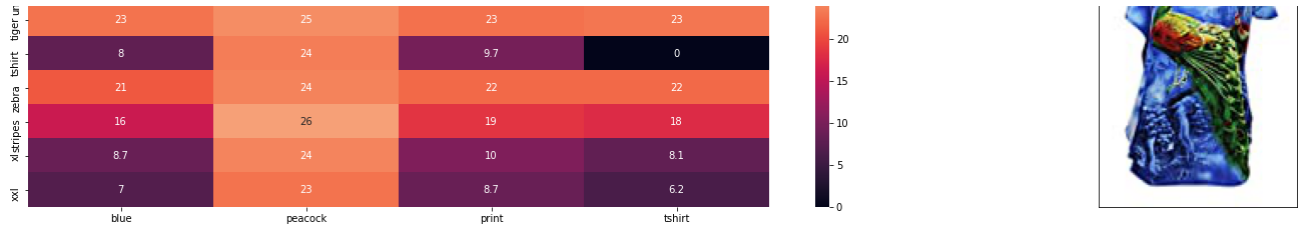
==========



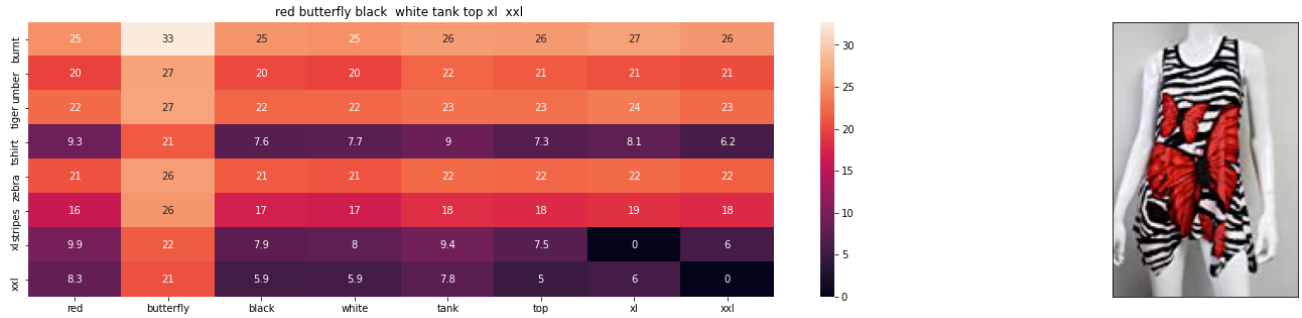
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 3.7045471955068456

==========

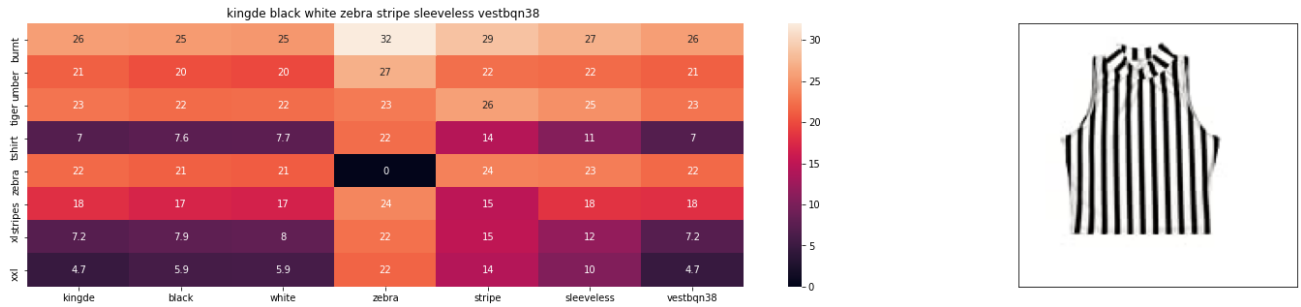




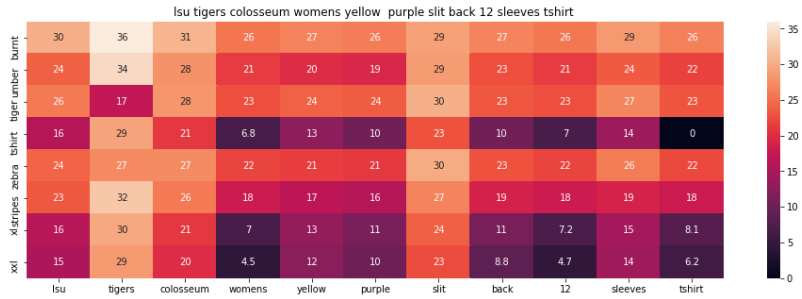
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 3.7796360779531346
==========



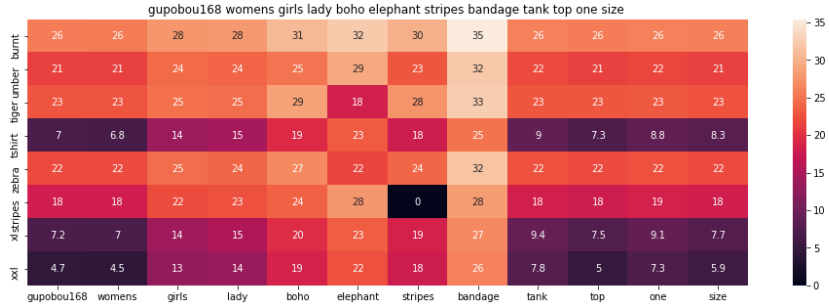
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 3.8623223114736427
==========



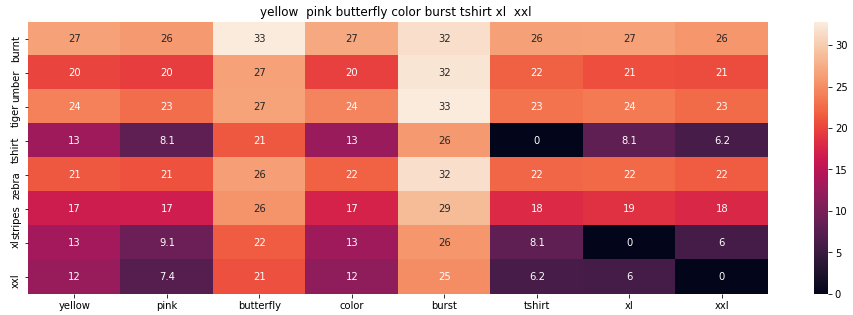
ASIN : B015H41F6G
Brand : KINGDE
euclidean distance from input : 3.8689527057218447
==========



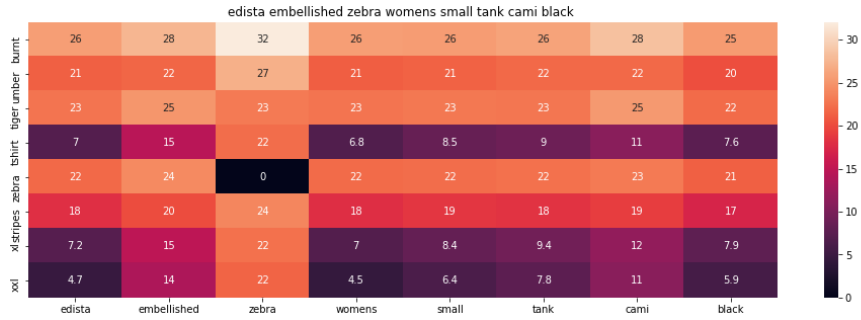
ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 3.9308106921720403
==========



ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 3.948397896079257
==========

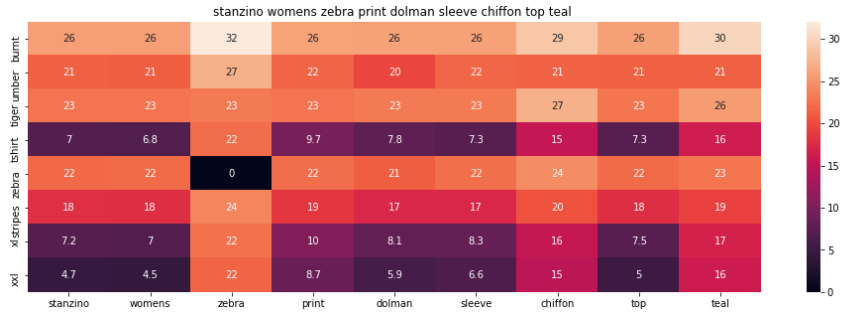


ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 3.9736458588369237
==========



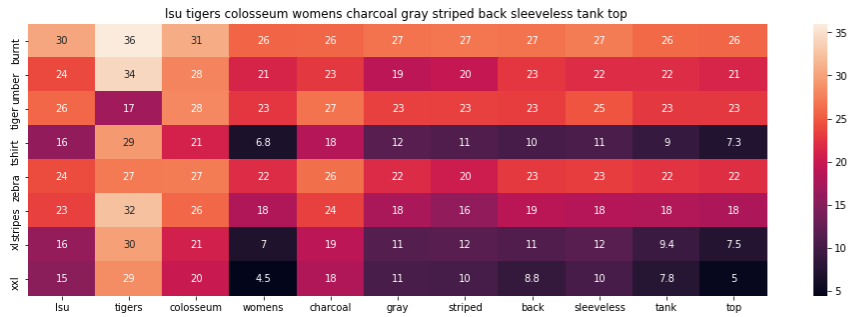
ASIN : B074P8MD22
Brand : Edista
euclidean distance from input : 3.9985597156095403

==========



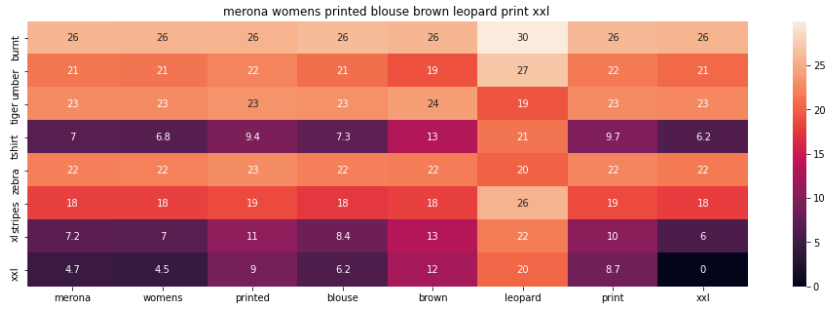
ASIN : B00C0I3U3E
Brand : Stanzino
euclidean distance from input : 4.009908058505048

==========



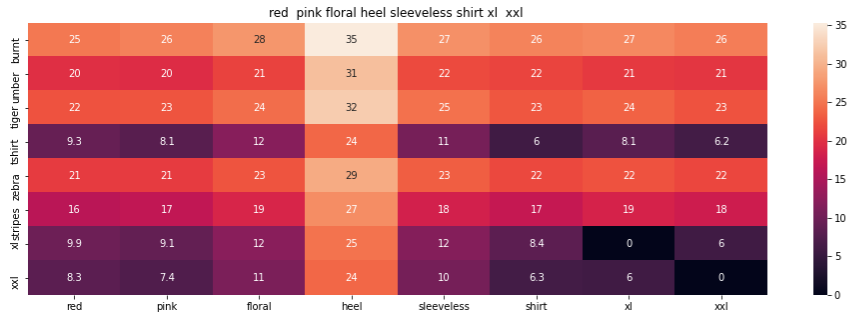
ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 4.027937462191572

==========



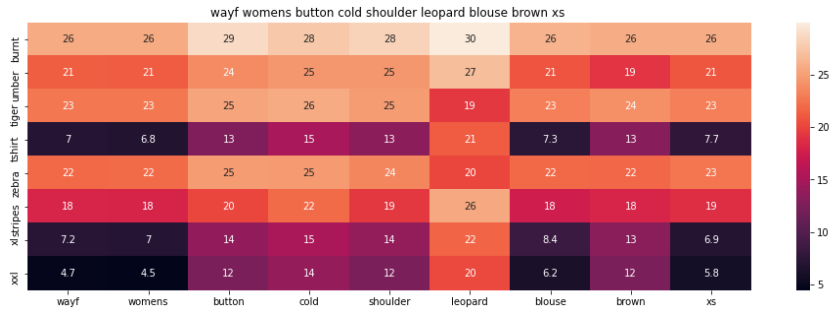
ASIN : B071YF3WDD
Brand : Merona
euclidean distance from input : 4.09020896182536

==========



ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 4.105110626292978

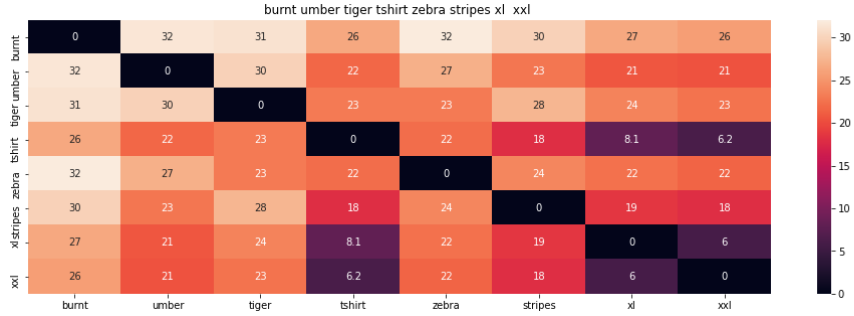
==========



ASIN : B01LZ7BQ4H
Brand : WAYF
euclidean distance from input : 4.106262275962069

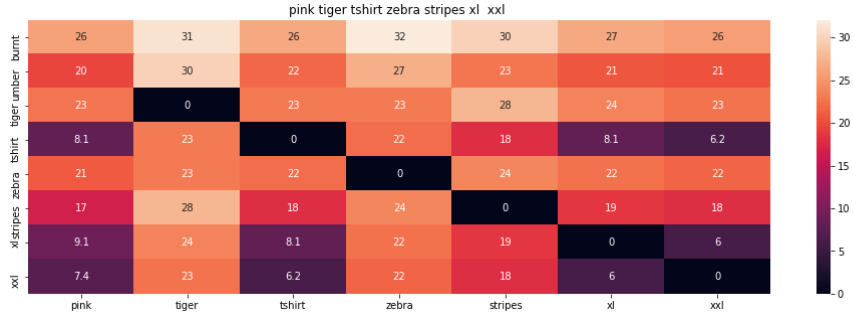
==========

```
# brand =5
# color = 2
# title vector weight = 5

idf_w2v_brand(12566, 5, 5, 2,20)
```



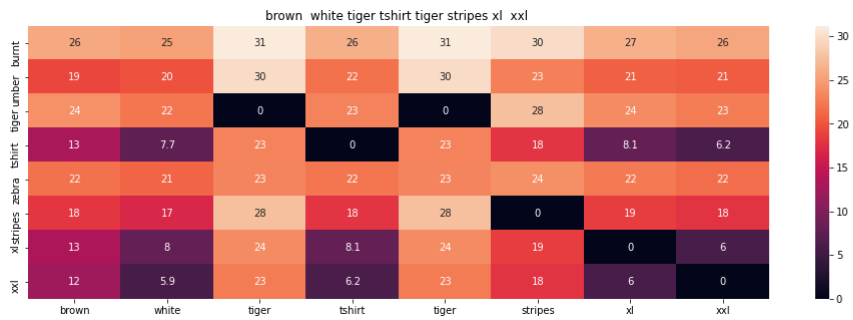
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0

==========

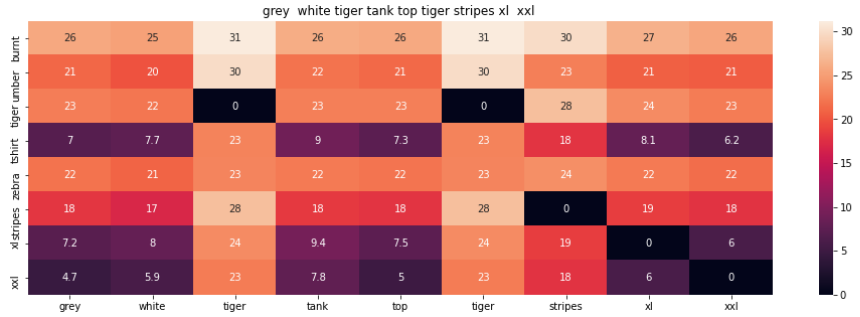


ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 1.928988361418872

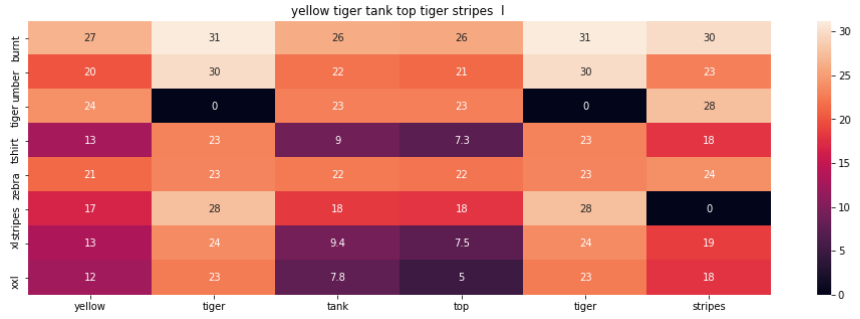
==========



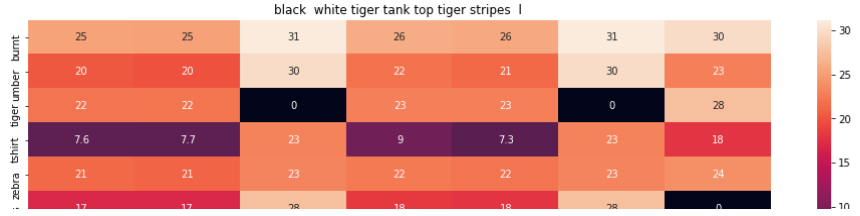
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 1.9878921508789062
==========

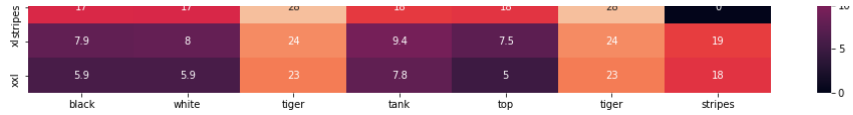


ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 2.4691024462984292
==========



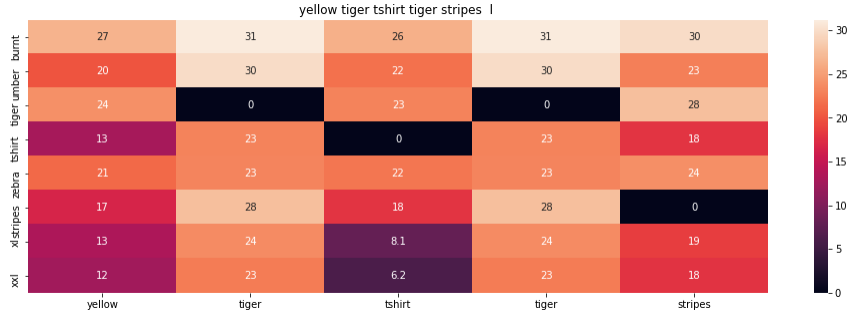
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 2.606336657266447
==========





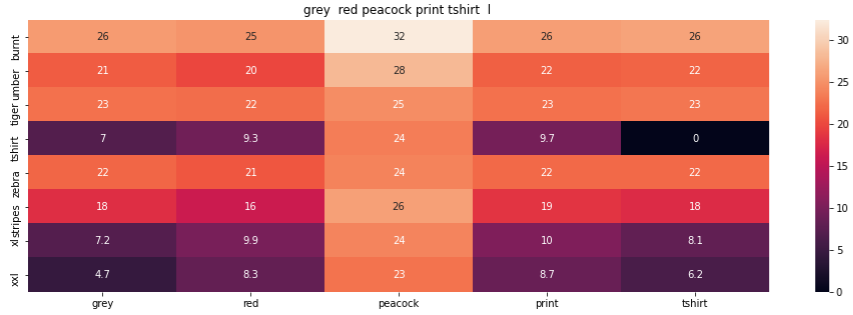
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 2.6077941895133545

==========



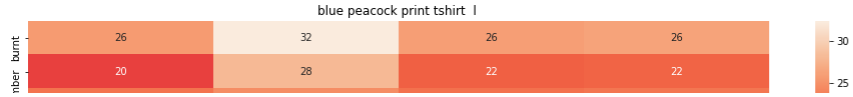
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 2.691303157866626

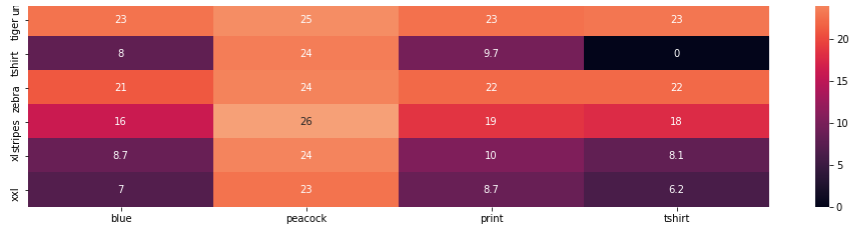
==========



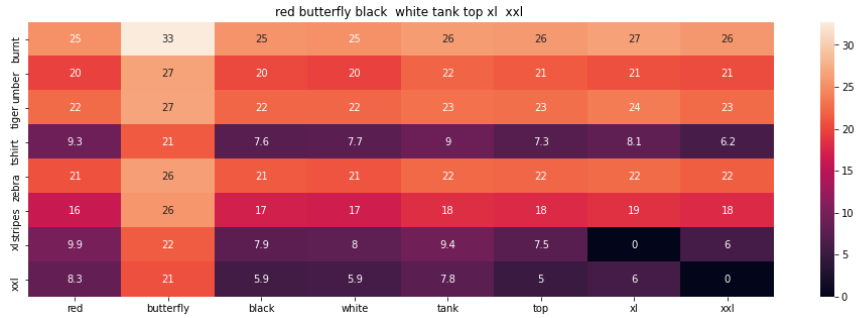
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 3.0871226629223716

==========

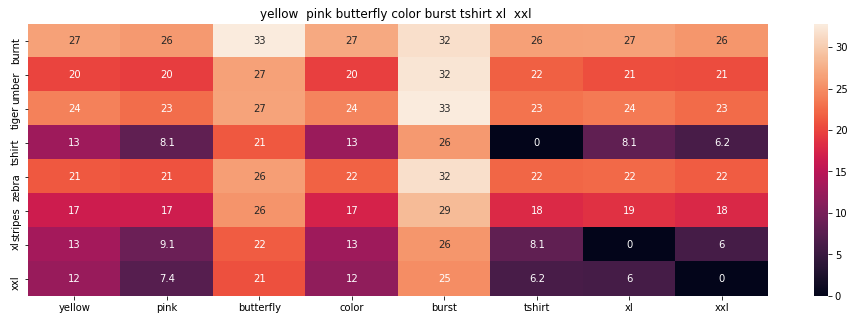




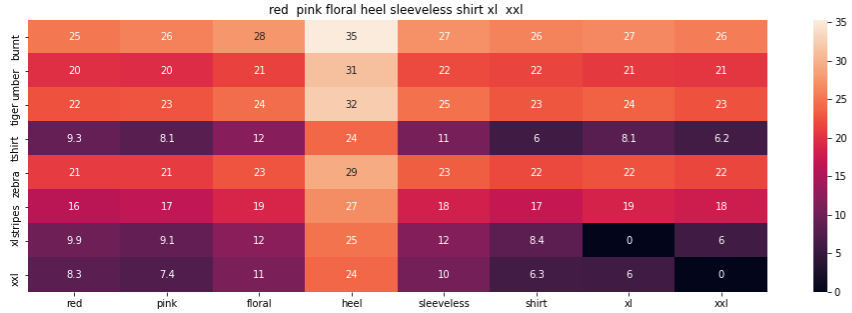
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 3.1496967316276123
==========



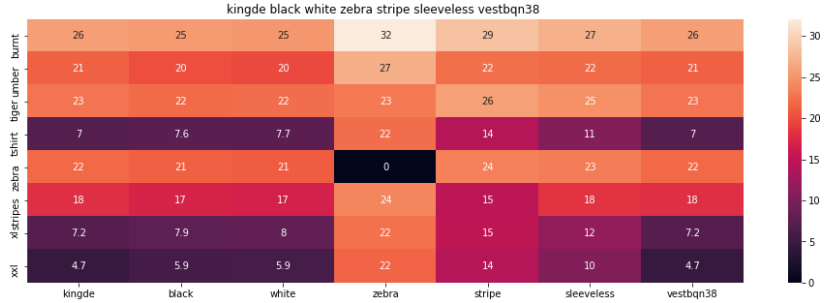
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 3.2186019262280356
==========



ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 3.31137154903077
==========

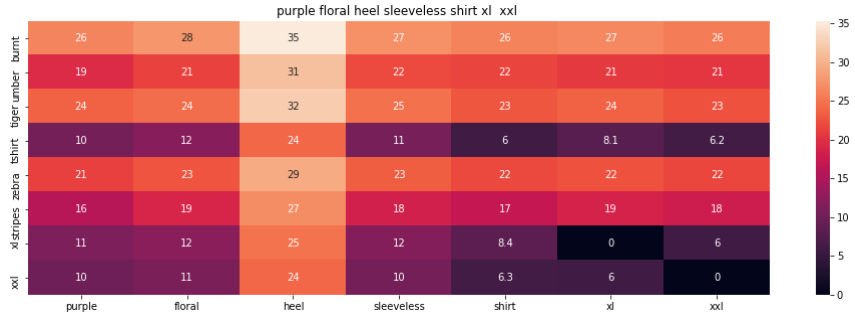


ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 3.4209255219108154
==========



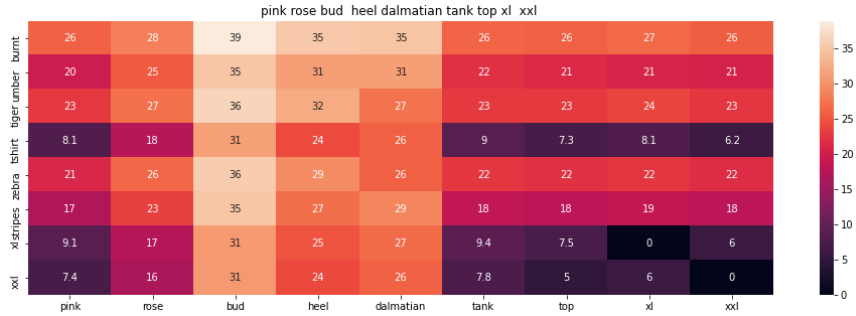
ASIN : B015H41F6G
Brand : KINGDE
euclidean distance from input : 3.5128023893630167
==========

ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 3.564350711404846
==========



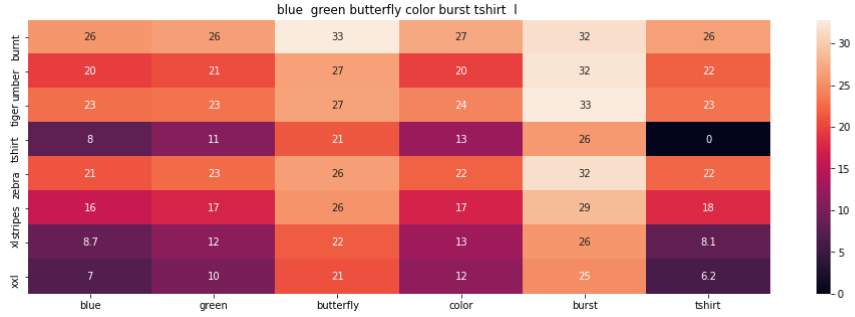
ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 3.565662765563159

==========



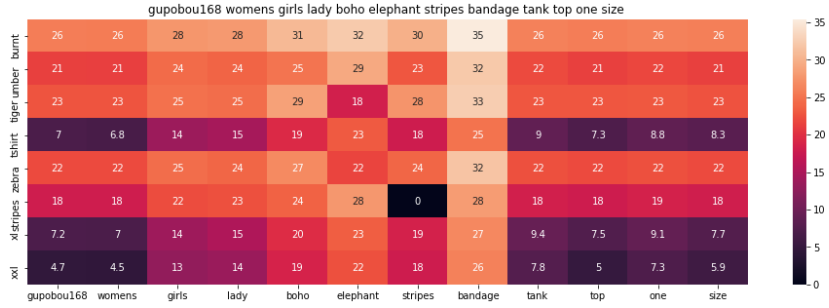
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 3.5710077922151773

==========



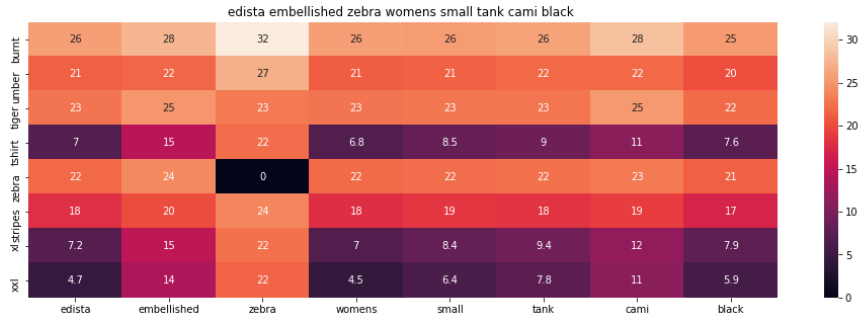
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 3.5734441122021567

==========



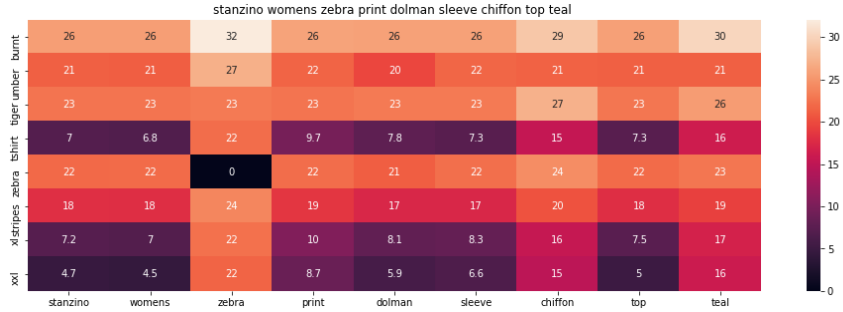
ASIN : B01ER18406
Brand : GuPoBoU168
euclidean distance from input : 3.57900671466086

==========



ASIN : B074P8MD22
Brand : Edista
euclidean distance from input : 3.620808230936096

==========



ASIN : B00C0I3U3E
Brand : Stanzino
euclidean distance from input : 3.630265183349019

.

==========

Keras and Tensorflow to extract features

```
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from sklearn.metrics import pairwise_distances
import matplotlib.pyplot as plt
import requests
from PIL import Image
import pandas as pd
import pickle
```

Visual features based product similarity.

```
#load the features and corresponding ASINS info.
bottleneck_features_train = np.load('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/16k_da
asins = np.load('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/16k_data_cnn_feature_asins
asins = list(asins)

# load the original 16K dataset
data = pd.read_pickle('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/pickels/16k_apperal_
df_asins = list(data['asin'])


from IPython.display import display, Image, SVG, Math, YouTubeVideo


#get similar products using CNN features (VGG-16)
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_trai

    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[0: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('Product Title: ', row['title'])
            print('Euclidean Distance from input image:', pdists[i])
            print('Amazon Url: www.amzon.com/dp/'+ asins[indices[i]])

get_similar_products_cnn(12566, 20)
```

Product Title: burnt umber tiger tshirt zebra stripes xl xxl
Euclidean Distance from input image: 6.32596e-06
Amazon Url: www.amzon.com/dp/B00JXQB5FQ

Product Title: pink tiger tshirt zebra stripes xl xxl
Euclidean Distance from input image: 30.05017
Amazon Url: www.amzon.com/dp/B00JXQASS6

Product Title: yellow tiger tshirt tiger stripes l
Euclidean Distance from input image: 41.261116
Amazon Url: www.amzon.com/dp/B00JXQCUIC

Product Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean Distance from input image: 44.000156
Amazon Url: www.amzon.com/dp/B00JXQCWTO

Product Title: kawaii pastel tops tees pink flower design
Euclidean Distance from input image: 47.38248

Amazon Url: www.amzon.com/dp/B071FCWD97

Product Title: womens thin style tops tees pastel watermelon print
Euclidean Distance from input image: 47.71842
Amazon Url: www.amzon.com/dp/B01JUNHBRM

Product Title: kawaii pastel tops tees baby blue flower design
Euclidean Distance from input image: 47.90206
Amazon Url: www.amzon.com/dp/B071SBCY9W

Product Title: edv cheetah run purple multi xl
Euclidean Distance from input image: 48.046482
Amazon Url: www.amzon.com/dp/B01CUPYBM0

Product Title: danskin womens vneck loose performance tee xsmall pink ombre
Euclidean Distance from input image: 48.101837
Amazon Url: www.amzon.com/dp/B01F7PHXY8

Product Title: summer alpaca 3d pastel casual loose tops tee design
Euclidean Distance from input image: 48.118866

Euclidean Distance from input image: 48.118866
Amazon Url: www.amzon.com/dp/B01I80A93G

Product Title: miss chievous juniors striped peplum tank top medium shadowpeach
Euclidean Distance from input image: 48.13122
Amazon Url: www.amzon.com/dp/B0177DM70S

Product Title: red pink floral heel sleeveless shirt xl xxl
Euclidean Distance from input image: 48.16945
Amazon Url: www.amzon.com/dp/B00JV63QQE

Product Title: moana logo adults hot v neck shirt black xxl
Euclidean Distance from input image: 48.256786
Amazon Url: www.amzon.com/dp/B01LX6H43D

Product Title: abaday multicolor cartoon cat print short sleeve longline shirt large
Euclidean Distance from input image: 48.265686
Amazon Url: www.amzon.com/dp/B01CR57YY0

Product Title: kawaii cotton pastel tops tees peach pink cactus design

Euclidean Distance from input image: 48.362602
Amazon Url: www.amzon.com/dp/B071WYLBZS

Product Title: chicago chicago 18 shirt women pink
Euclidean Distance from input image: 48.383606
Amazon Url: www.amzon.com/dp/B01GXAZTRY

Product Title: yichun womens tiger printed summer tshirts tops
Euclidean Distance from input image: 48.449356
Amazon Url: www.amzon.com/dp/B010NN9RXO

Product Title: nancy lopez whimsy short sleeve whiteblacklemon drop xs
Euclidean Distance from input image: 48.47889
Amazon Url: www.amzon.com/dp/B01MPX6IDX

Product Title: womens tops tees pastel peach ice cream cone print
Euclidean Distance from input image: 48.557957
Amazon Url: www.amzon.com/dp/B0734GRKZL

Product Title: uswomens many j blige without tshirts shirt

```
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from sklearn.metrics import pairwise_distances
import matplotlib.pyplot as plt
import requests
from PIL import Image
import pandas as pd
import pickle
```

```
bottleneck_features_train = np.load('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/16k_da
asins = np.load('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/16k_data_cnn_feature_asins
```

```
data = pd.read_pickle('/gdrive/My Drive/Applied_AI_Workshop_Code_Data/pickels/16k_apperal_
df_asins = list(data['asin'])
asins = list(asins)
```

```
bottleneck_features_train.shape
```

```
(16042, 25088)
```

Combining Different features with different weights

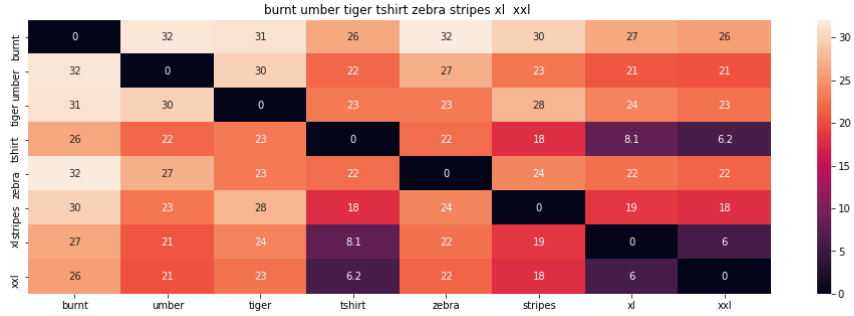
```
print(bottleneck_features_train.shape)
print(brand_features.shape)
print(color_features.shape)
print(w2v_title_weight.shape)
```

```
(16042, 25088)
(16042, 3835)
(16042, 1845)
(16042, 300)
```

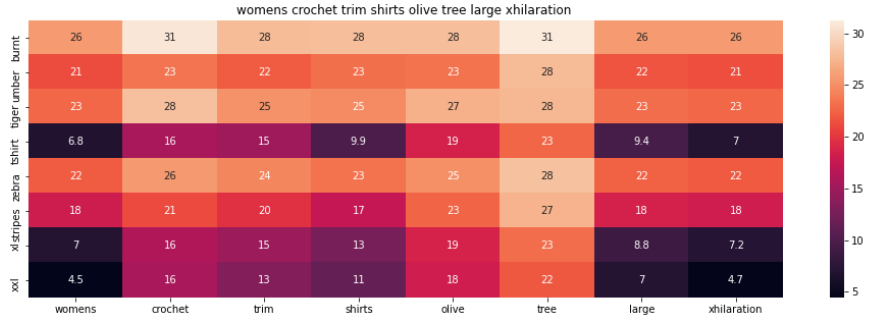
Equal Weightage

```
def combined(doc_id, w1, w2, w3, w4, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  title features
    # w2: weight for brand features
    # w3 : weight for color features
    # w4: weight for image feature

    # pairwise_dist will store the distance from given input apparel to all remaining appa
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <X, Y
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(
```

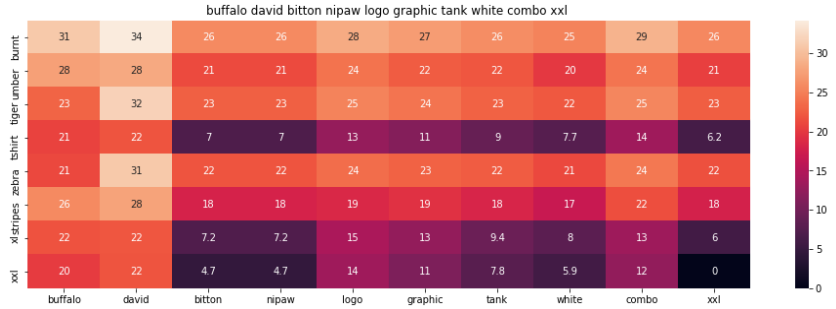
```
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(
    brand_feat_dist = pairwise_distances(brand_features, brand_features[doc_id])
    color_feat_dist = pairwise_distances(color_features, color_features[doc_id])
    image = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id

    pairwise_dist   = (w1 * idf_w2v_dist +  w2 * brand_feat_dist + w3*color_feat_dist + w4

    # np.argsort will return indices of 9 smallest distances
    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    #pdists will store the 9 smallest distances
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    #data frame indices of the 9 smallest distace's
    df_indices = list(data.index[indices])


    for i in range(0, len(indices)):
        heat_map_w2v_brand(data['title'].loc[df_indices[0]],data['title'].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)

combined(12566, 5, 5,5,5 ,20)
# in the give heat map, each cell contains the euclidean distance between words i, j
```



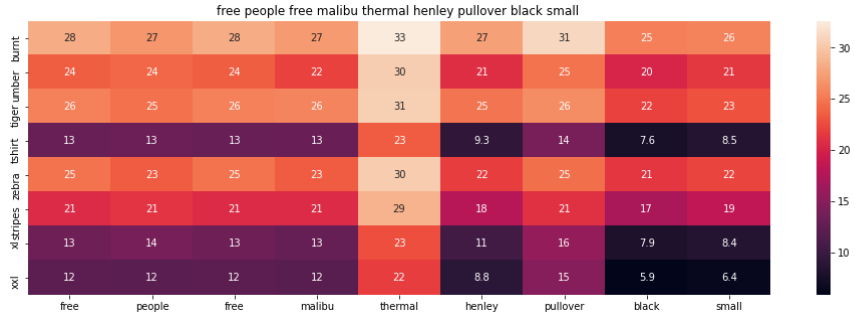
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 1.8773096599034034e-06
==========



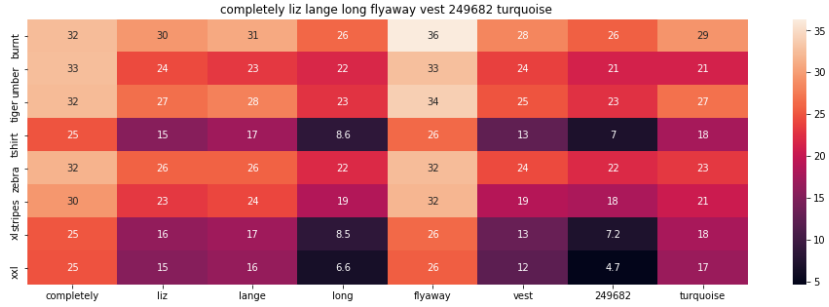
ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 12.14585423831813
==========



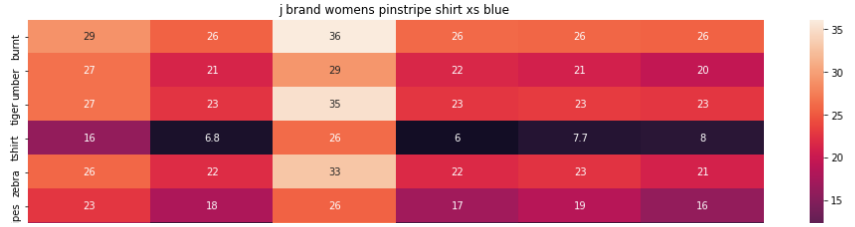
ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 12.611845449255629
==========

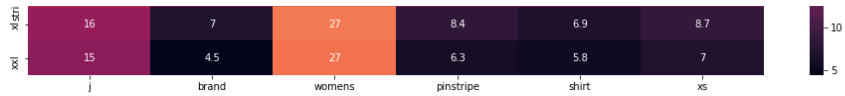


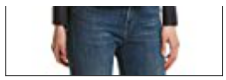
ASIN : B074MXY984
Brand : We The Free
euclidean distance from input : 12.86125774763912
==========



ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 12.864222908110364
==========

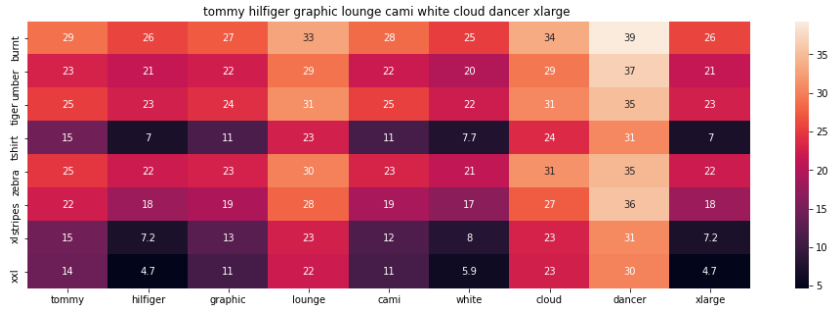




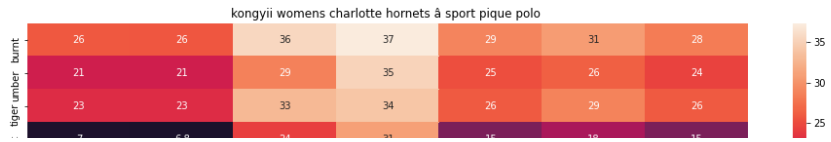


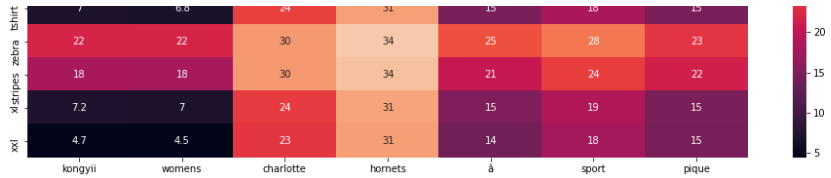
ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 12.926071929931641
==========

ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 12.94621880244643
==========

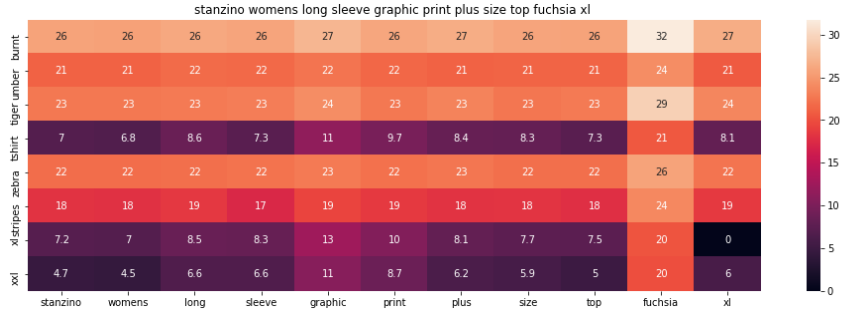


ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 13.170285606384278
==========

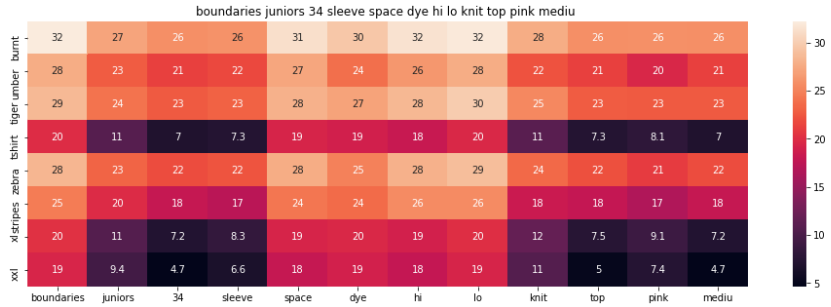




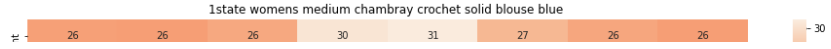
ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 13.288894392961566
==========

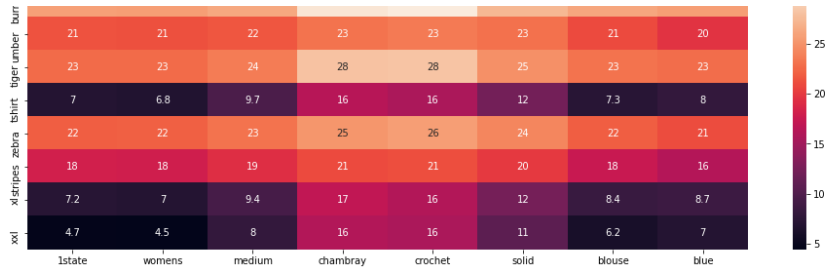


ASIN : B00DP4VHWI
Brand : Stanzino
euclidean distance from input : 13.314256026262347
==========

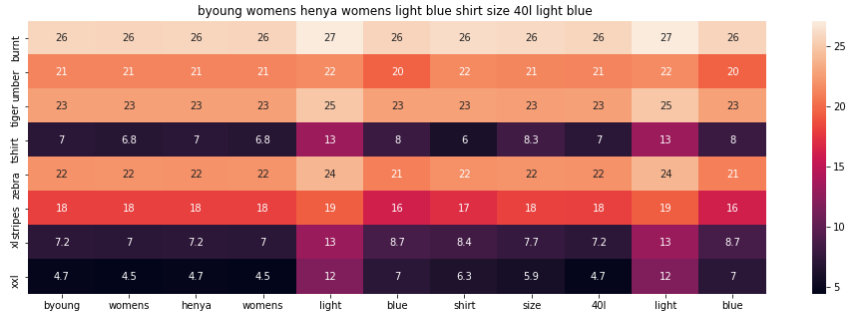


ASIN : B01EXXFS4M
Brand : No Boundaries
euclidean distance from input : 13.319862747282727
==========

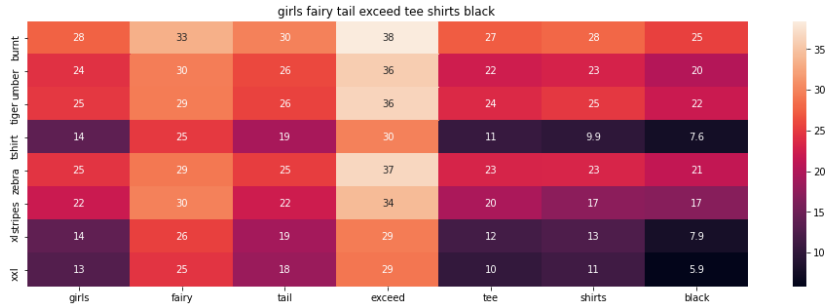




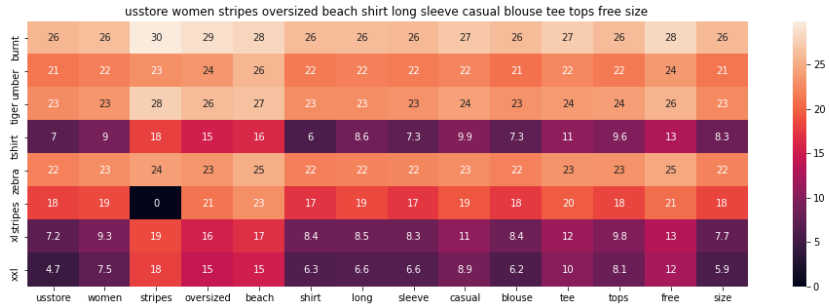
ASIN : B074MK6LV2
Brand : 1.State
euclidean distance from input : 13.32218315791518
==========



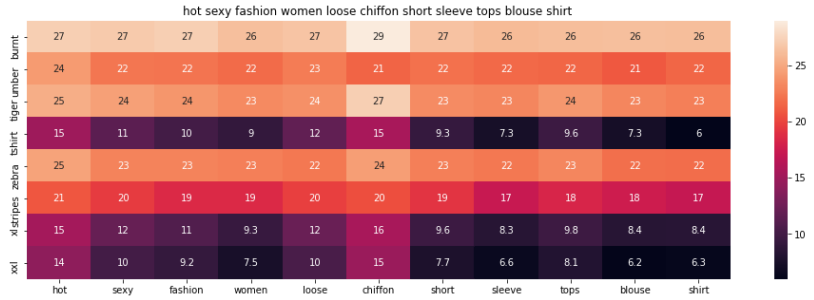
ASIN : B06Y41MRCH
Brand : Byoung
euclidean distance from input : 13.377500013159437
==========



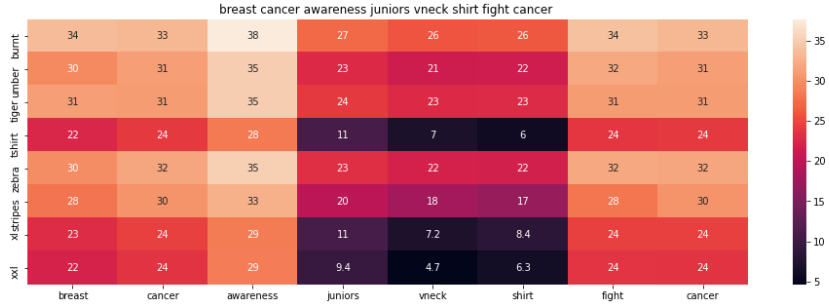
ASIN : B01L9F153U
Brand : ATYPEMX
euclidean distance from input : 13.39535210322768
==========



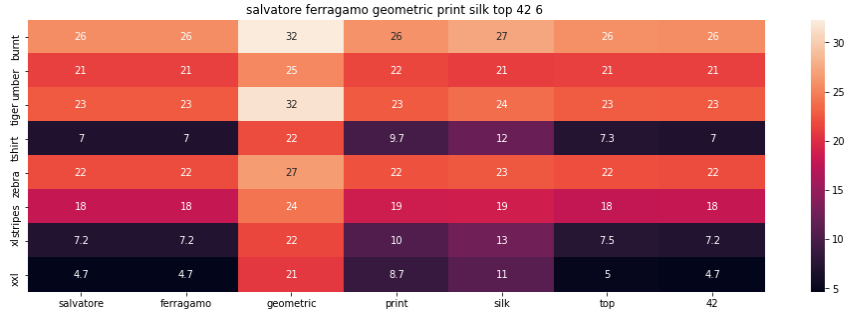
ASIN : B01DNNI1RO
Brand : Usstore
euclidean distance from input : 13.400609067724867
==========



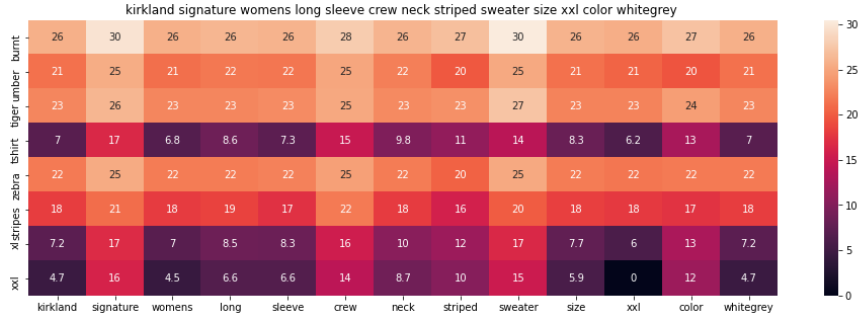
ASIN : B00JMAASRO
Brand : Wotefusi
euclidean distance from input : 13.404457022661273
==========



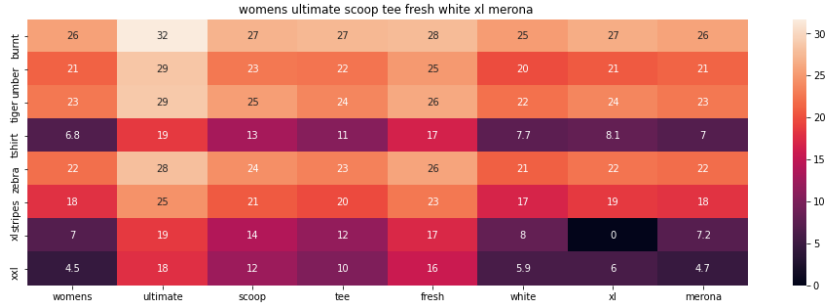
ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 13.40730164241225
==========



ASIN : B0756JTS1F
Brand : Salvatore Ferragamo
euclidean distance from input : 13.444146468066535
==========



ASIN : B06XTPC3FP
Brand : Kirkland Signature
euclidean distance from input : 13.461921691984875
==========

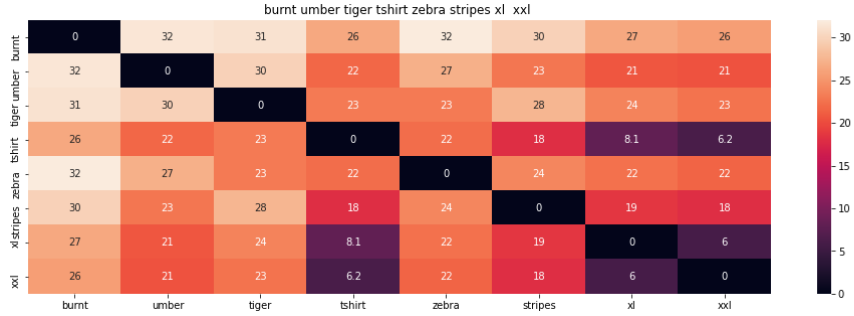


ASIN : B01G7XE50E
Brand : Merona
euclidean distance from input : 13.490835883134906
==========

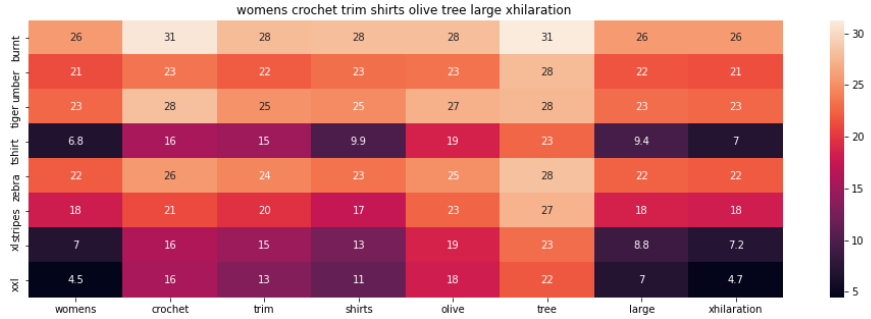
when equal weightage is given we can see that it finds the product which has some similarity in all the attributes of the query product

Image is given more weightage

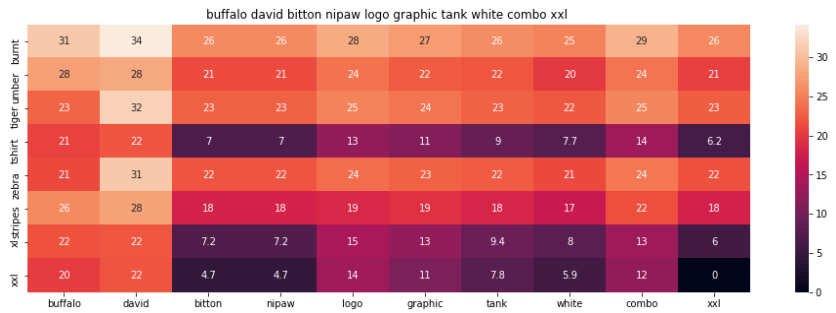
```
combined(12566, 7, 2,1,10 ,20)
```



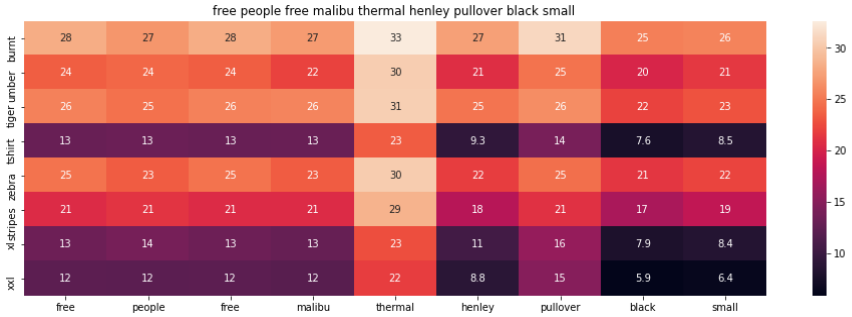
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 3.754619319806807e-06
==========



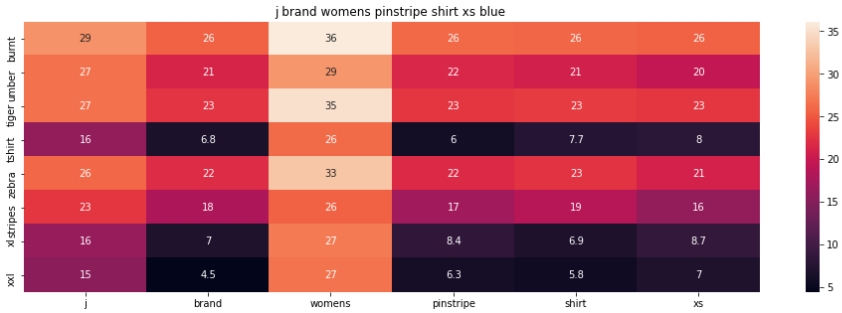
ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 21.629733692302324
==========



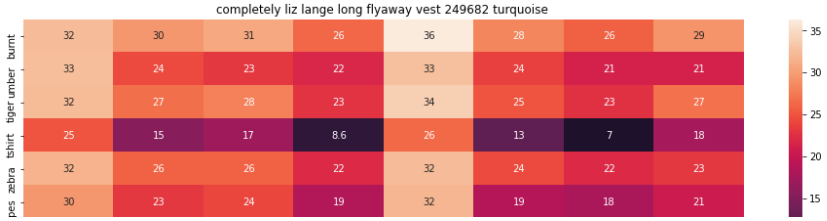
ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 22.638267551555252
==========

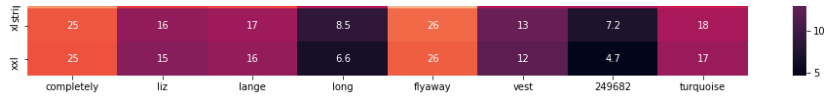


ASIN : B074MXY984
Brand : We The Free
euclidean distance from input : 22.927109185768536
==========



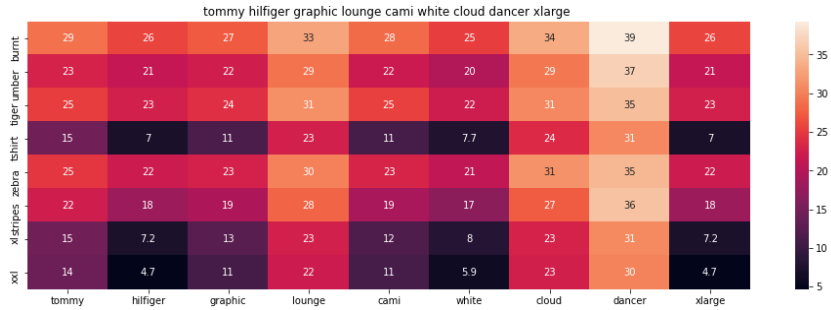
ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 22.93509159088135
==========





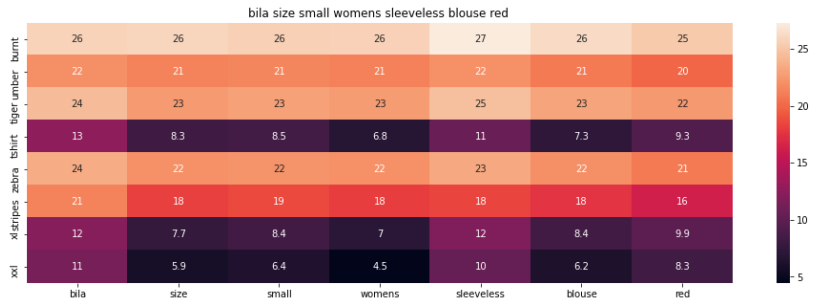
ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 22.990277404803226

==========



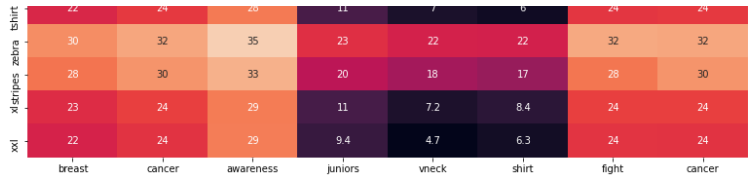
ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 23.298449897766112

==========

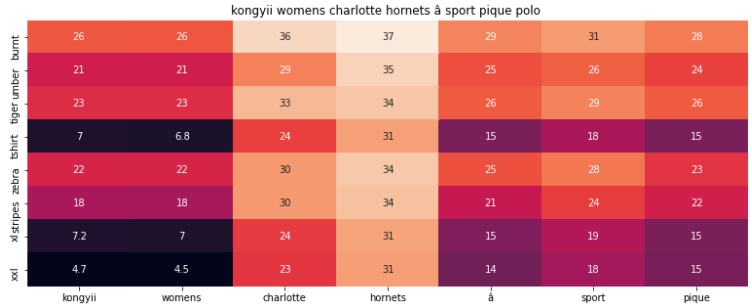


ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 23.422896395472222

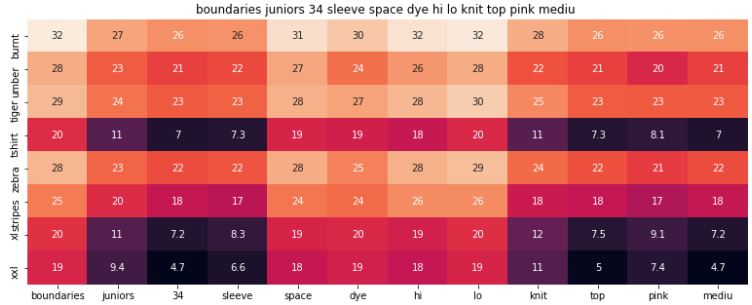
==========



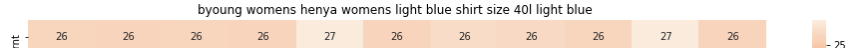
ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 23.65991383626812
==========

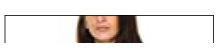


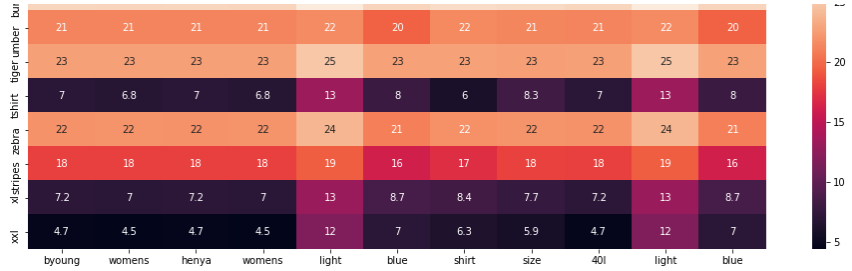
ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 23.85630971029156
==========



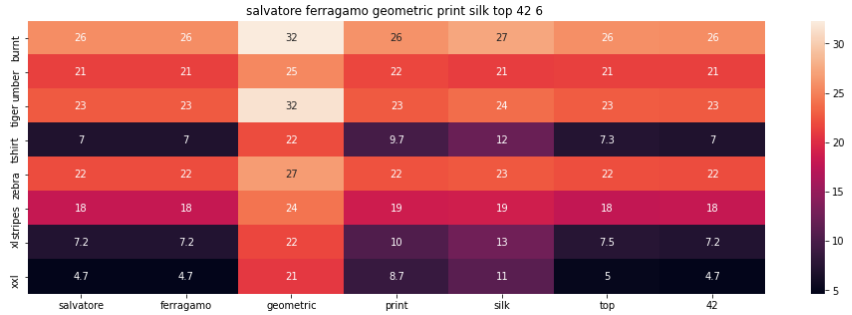
ASIN : B01EXXFS4M
Brand : No Boundaries
euclidean distance from input : 23.977068061846683
==========



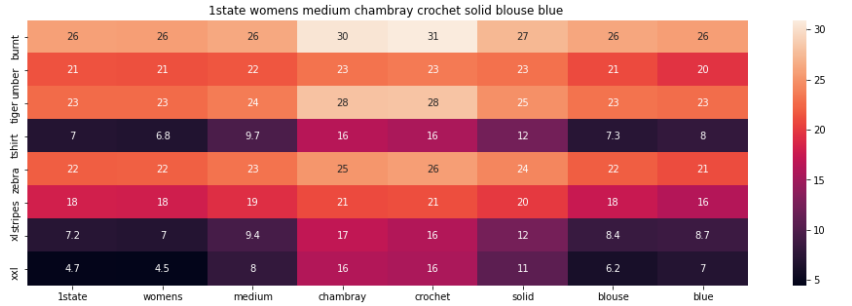




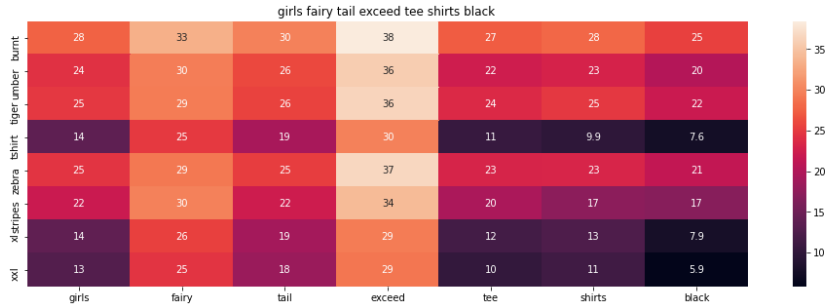
ASIN : B06Y41MRCH
Brand : Byoung
euclidean distance from input : 24.067939602031327
==========



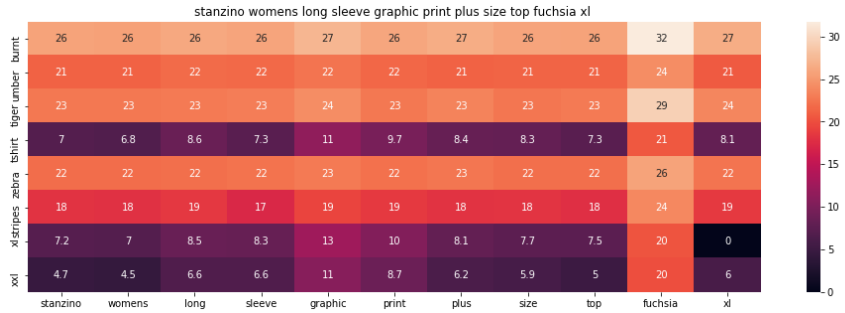
ASIN : B0756JTS1F
Brand : Salvatore Ferragamo
euclidean distance from input : 24.08281359231936
==========



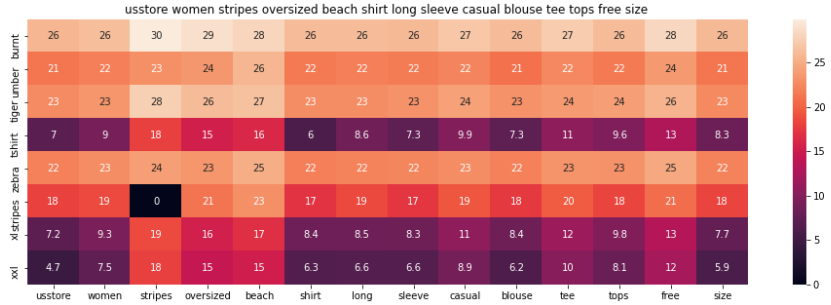
ASIN : B074MK6LV2
Brand : 1.State
euclidean distance from input : 24.101202402857474
==========



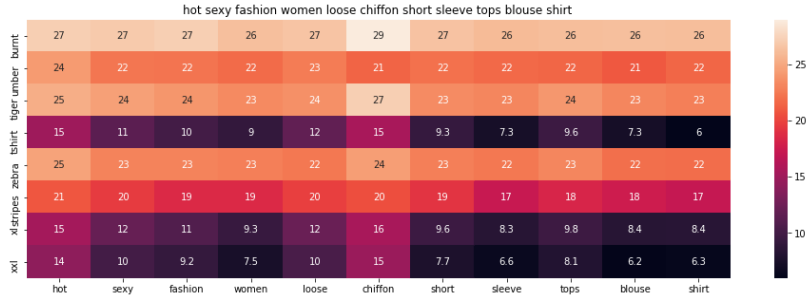
ASIN : B01L9F153U
Brand : ATYPEMX
euclidean distance from input : 24.134207163599662
==========



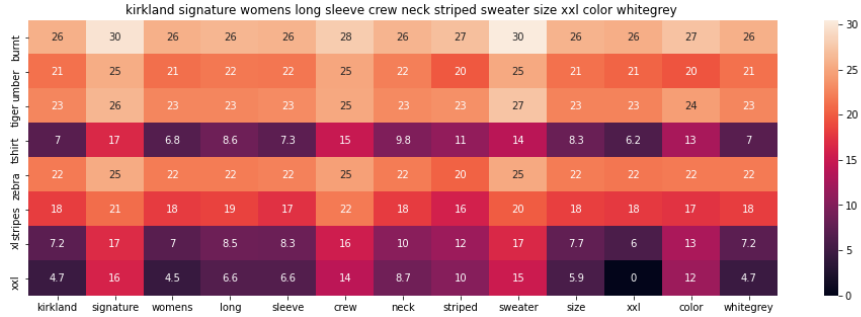
ASIN : B00DP4VHWI
Brand : Stanzino
euclidean distance from input : 24.13915558889263
==========



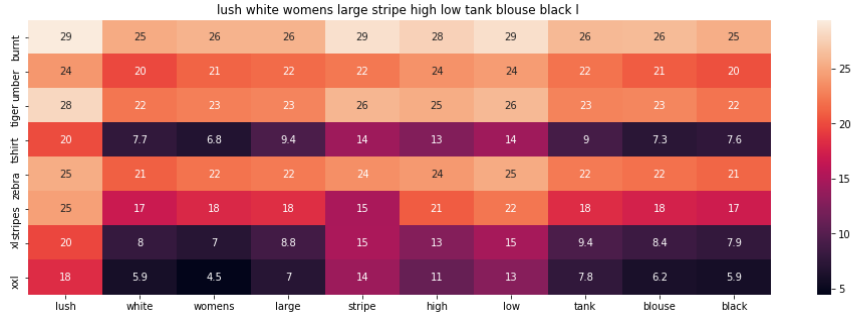
ASIN : B01DNNI1RO
Brand : Usstore
euclidean distance from input : 24.196374355449297
==========



ASIN : B00JMAASRO
Brand : Wotefusi
euclidean distance from input : 24.269813929346732
==========



ASIN : B06XTPC3FP
Brand : Kirkland Signature
euclidean distance from input : 24.35863811494727
==========

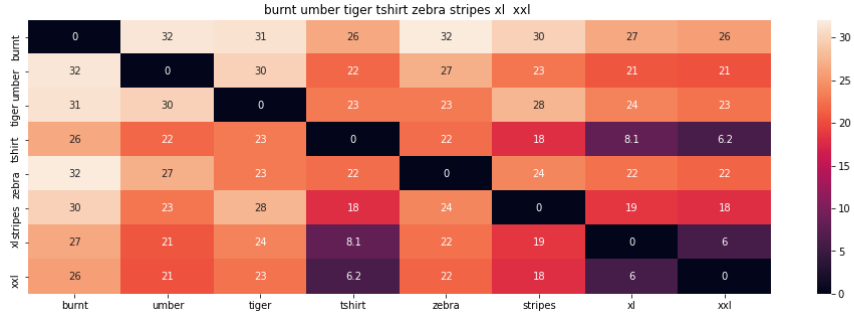


ASIN : B07285DZ7S
Brand : Lush Clothing
euclidean distance from input : 24.361934776324222
==========

Image and Brand are given more preference which can also be seen the products being recommended

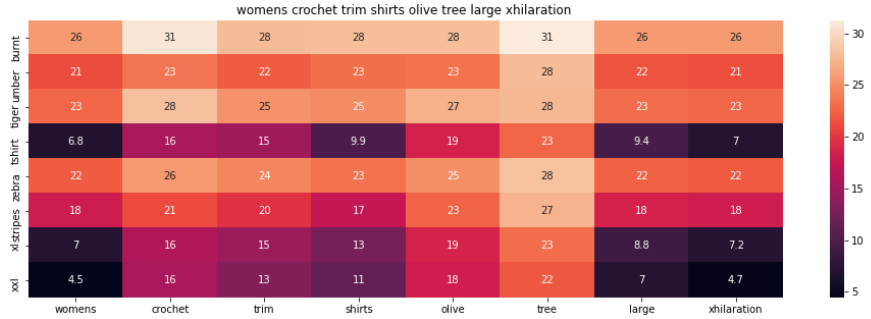
Image and title is given equal weightage, more than other feature

```
combined(12566, 10, 2,2 ,10, 20)
```



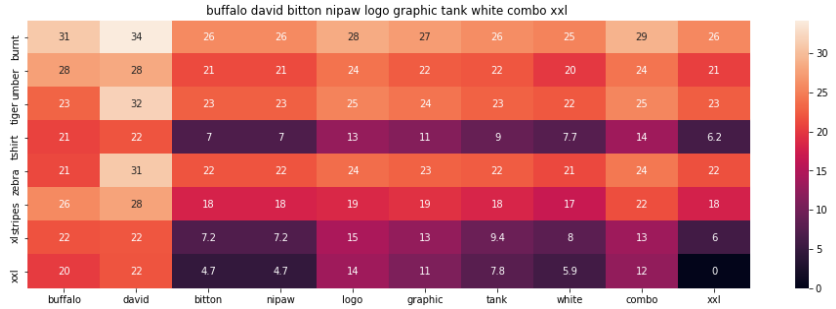
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 3.128849433172339e-06

==========

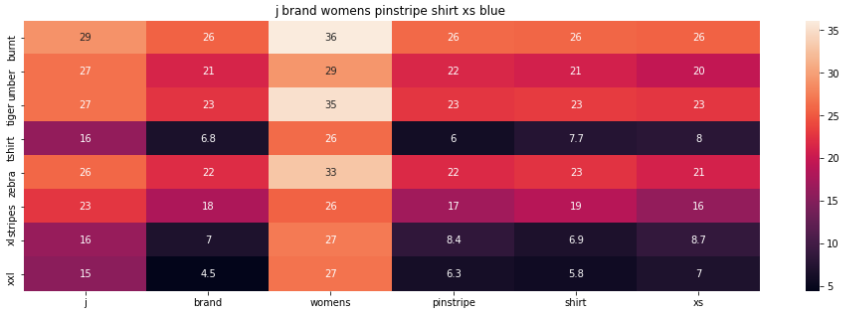


ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 19.08838985881763

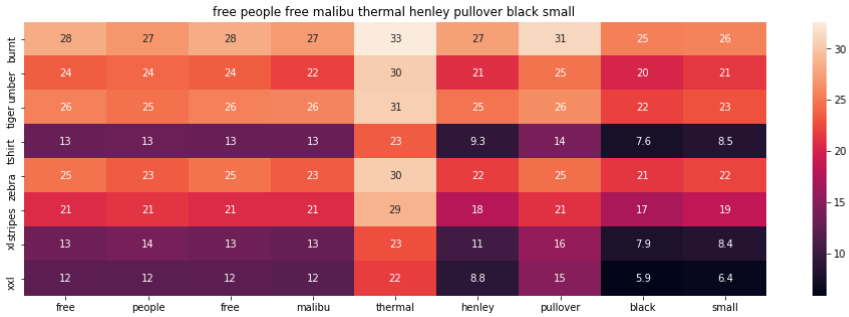
==========



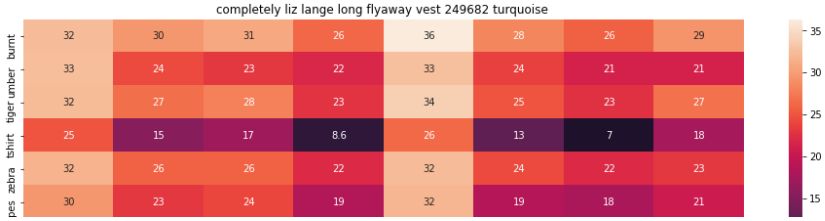
ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 19.865041877046796
==========

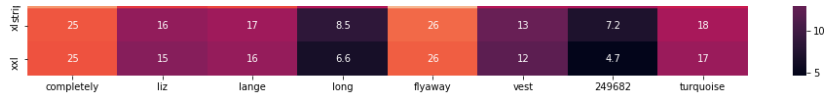


ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 20.210119883219402
==========



ASIN : B074MXY984
Brand : We The Free
euclidean distance from input : 20.21866906610757
==========



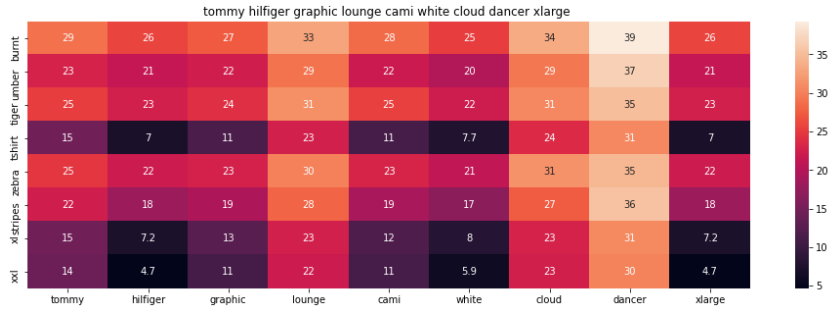


ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 20.302300326059576

==========

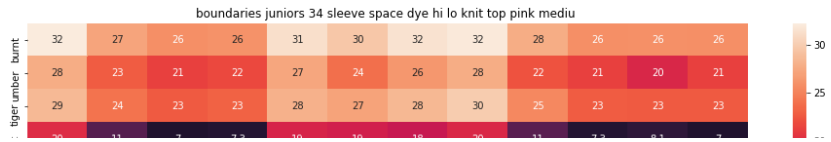
ASIN : B01L7ROZNC
Brand : Bila
euclidean distance from input : 20.52827654743006

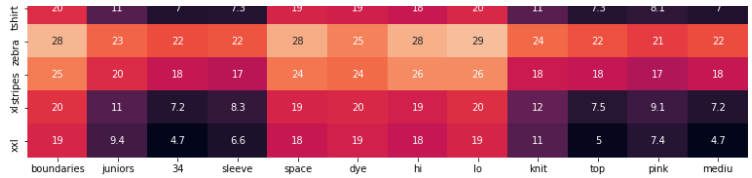
==========



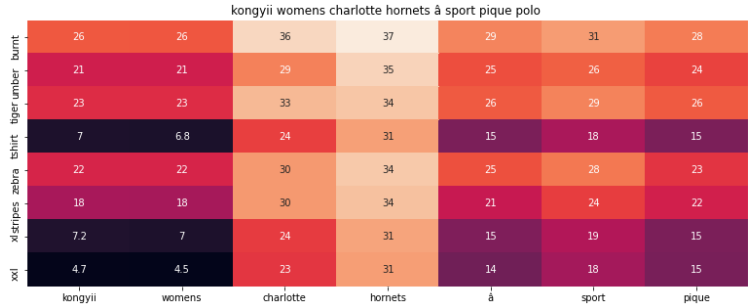
ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 20.61714267730713

==========

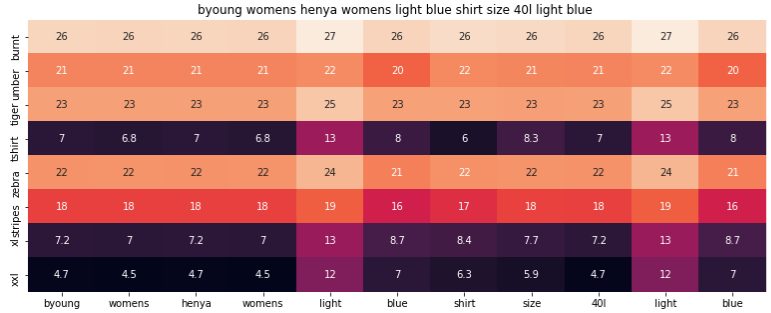




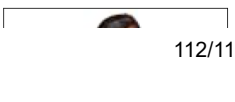
ASIN : B01EXXFS4M
Brand : No Boundaries
euclidean distance from input : 21.061700058013514
==========

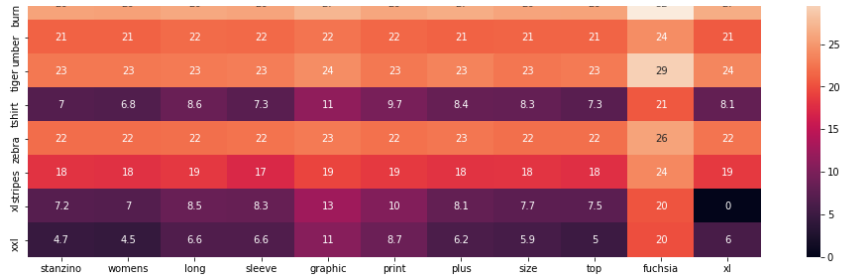


ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 21.099402531621955
==========

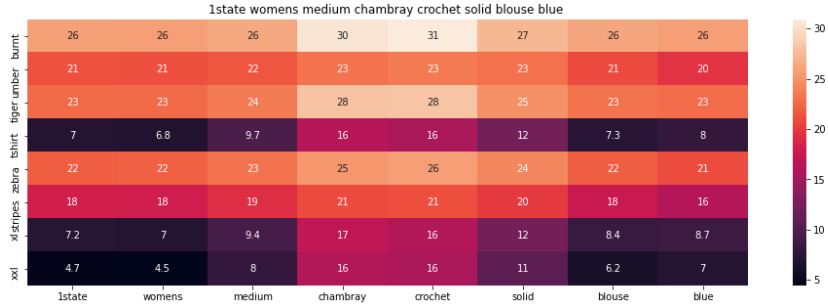


ASIN : B06Y41MRCH
Brand : Byoung
euclidean distance from input : 21.14113281688648
==========

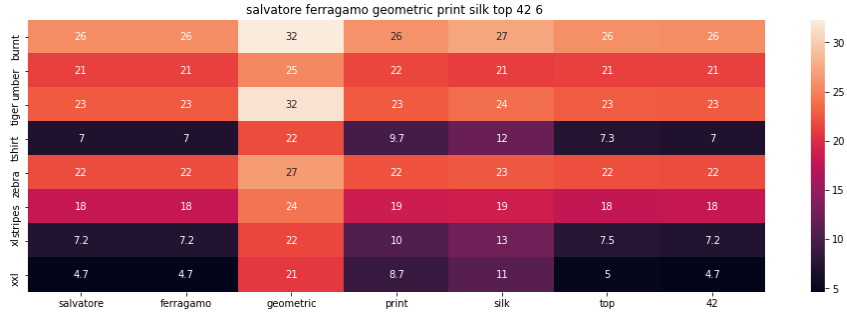




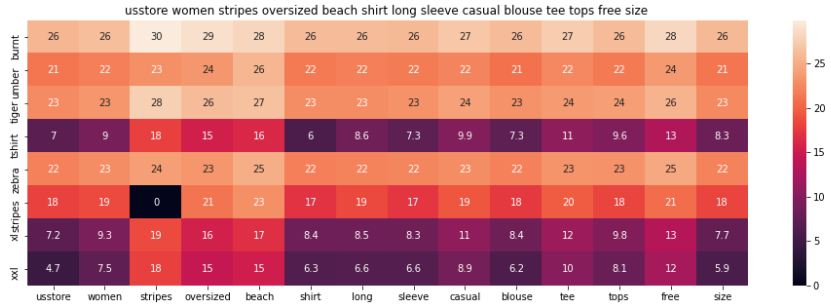
ASIN : B00DP4VHWI
Brand : Stanzino
euclidean distance from input : 21.14167192045659
==========



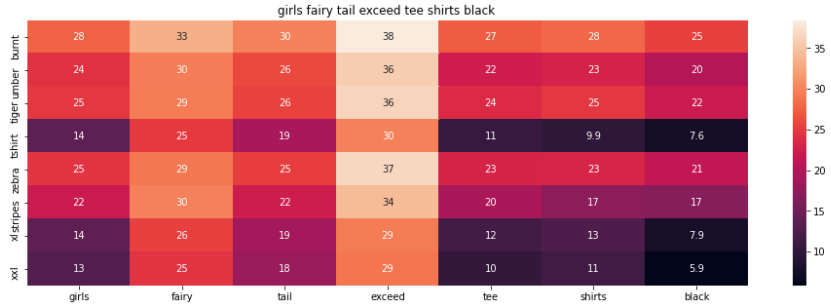
ASIN : B074MK6LV2
Brand : 1.State
euclidean distance from input : 21.154883806544643
==========



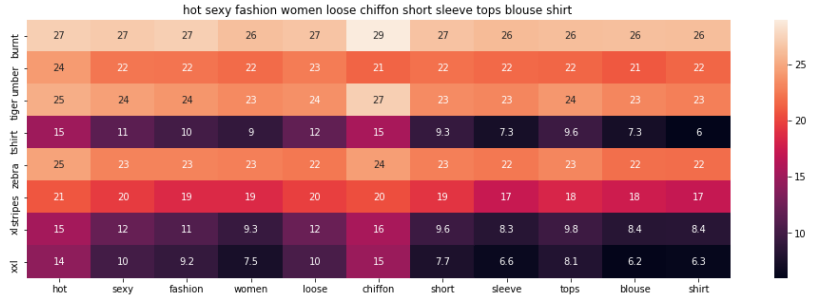
ASIN : B0756JTS1F
Brand : Salvatore Ferragamo
euclidean distance from input : 21.1628938442546
==========



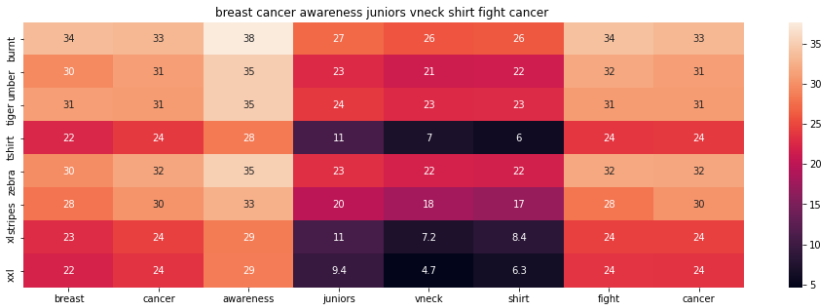
ASIN : B01DNNI1RO
Brand : Usstore
euclidean distance from input : 21.179647907828862
==========



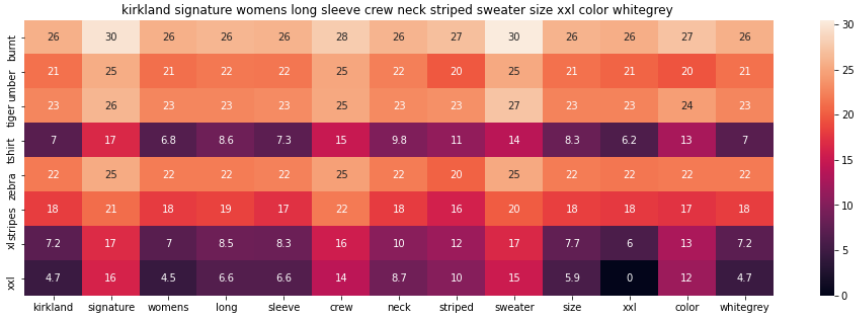
ASIN : B01L9F153U
Brand : ATYPEMX
euclidean distance from input : 21.276832048732143
==========



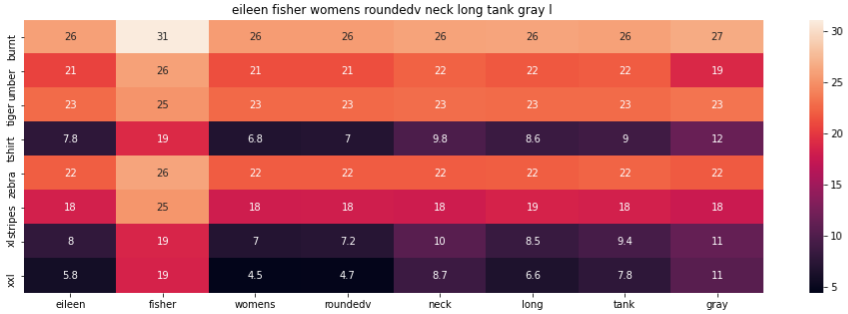
ASIN : B00JMAASRO
Brand : Wotefusi
euclidean distance from input : 21.2920069144548
==========



ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 21.296747947373095
==========



ASIN : B06XTPC3FP
Brand : Kirkland Signature
euclidean distance from input : 21.298464965850428
==========



ASIN : B01N2JSRDM
Brand : Eileen Fisher
euclidean distance from input : 21.358461252878424
==========

Brand and images are given equal weightage and in the recommended products we can see that the products of the same brand which same pattern are being recommended

Conclusion

We combided features and tried giving different weights.

1. title feature with weighted word2vec
2. other features like brand , color with one-hot encoding
3. image using VGG-16

We tried giving different weightage to different features and we got different product recommendations according to it.