Amazon Apparel Recommendations

Loading all required libraries

In [0]:

```
#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")
```

In [0]:

```
data = pd.read_json('tops_fashion.json')
```

In [0]:

```
print ('Number of data points : ', data.shape[0], \
       'Number of features/variables:', data.shape[1])
```

```
Number of data points :  183138 Number of features/variables: 19
```

Of these 19 features, we will be using only 6 features in this workshop.

1. asin (Amazon standard identification number)
2. brand (brand to which the product belongs to)
3. color (Color information of apparel, it can contain many colors as a value e
x: red and black stripes)
4. product_type_name (type of the apperal, ex: SHIRT/TSHIRT)
5. medium_image_url (url of the image)
6. title (title of the product.)
7. formatted_price (price of the product)

In [0]:

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

In [0]:

```
print ('Number of data points : ', data.shape[0], \
       'Number of features:', data.shape[1])
data.head()
```

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

Out[37]:

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

Basic stats for the feature: product_type_name

In [0]:

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

# 91.62% (167794/183138) of the products are shirts.
```

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

In [0]:

```
# names of different product types
print(data['product_type_name'].unique())
```

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

In [0]:

```
# 10 most frequent product_type_names.
product_type_count = Counter(list(data['product_type_name']))
product_type_count.most_common(10)
```

Out[40]:

```
[('SHIRT', 167794),
 ('APPAREL', 3549),
 ('BOOKS_1973_AND_LATER', 3336),
 ('DRESS', 1584),
 ('SPORTING_GOODS', 1281),
 ('SWEATER', 837),
 ('OUTERWEAR', 796),
 ('OUTDOOR_RECREATION_PRODUCT', 729),
 ('ACCESSORY', 636),
 ('UNDERWEAR', 425)]
```

Basic stats for the feature: brand

In [0]:

```
# there are 10577 unique brands
print(data['brand'].describe())

# 183138 - 182987 = 151 missing values.
```

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

In [0]:

```
brand_count = Counter(list(data['brand']))
brand_count.most_common(10)
```

Out[42]:

```
[('Zago', 223),
 ('XQS', 222),
 ('Yayun', 215),
 ('YUNY', 198),
 ('XiaoTianXin-women clothes', 193),
 ('Generic', 192),
 ('Boohoo', 190),
 ('Alion', 188),
 ('Abetteric', 187),
 ('TheMogan', 187)]
```

Basic stats for the feature: color

In [0]:

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

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

In [ ]:

```
# we have 7380 unique colors
# 7.2% of products are black in color
# 64956 of 183138 products have brand information. That's approx 35.4%.
```

In [0]:

```
color_count = Counter(list(data['color']))
color_count.most_common(10)
```

Out[44]:

```
[(None, 118182),
 ('Black', 13207),
 ('White', 8616),
 ('Blue', 3570),
 ('Red', 2289),
 ('Pink', 1842),
 ('Grey', 1499),
 ('*', 1388),
 ('Green', 1258),
 ('Multi', 1203)]
```

Basic stats for the feature: formatted_price

In [0]:

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

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

In [ ]:

```
# Only 28,395 (15.5% of whole data) products with price information
```

In [0]:

```
price_count = Counter(list(data['formatted_price']))
price_count.most_common(10)
```

Out[46]:

```
[(None, 154743),
 ('$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)]
```

Basic stats for the feature: title

In [0]:

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

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

In [ ]:

```
# All of the products have a title.
```

In [0]:

```
# consider products which have price information
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
```

In [0]:

```
# consider products which have color information
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
```

Remove near duplicate items

Understand about duplicates.

In [0]:

```
data = pd.read_pickle('pickels/28k_apparel_data')

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

```

```
2325
```

In [ ]:

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

Removing duplicates - 1

In [0]:

```
data = pd.read_pickle('pickels/28k_apparel_data')
```

In [0]:

```
1 data.head()
```

Out[103]:

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

In [0]:

```
1 # Remove All products with very few words in title
2
3 data_sorted = data[data['title'].apply(lambda x: len(x.split())>4)]
4 print("After removal of products with short description:", data_sorted.shape[0])
```

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

In [0]:

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

Out[105]:

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

Some examples of dupliacte titles that differ only in the last few words.

```
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 Shirt for woman Neon Wolf t-shirt
62. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print
Head Shirt for woman Neon Wolf t-shirt
63. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print
Head Shirt for woman Neon Wolf t-shirt
64. psychedelic colorful Howling Galaxy Wolf T-shirt/Colorful Rainbow Animal Print
Head Shirt for woman Neon Wolf t-shirt
```

In [ ]:

```
# Code to remove the duplicates which differ only at the end.
```

In [0]:

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

In [0]:

```
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

    a = data['title'].loc[indices[i]].split()


    j = i+1
    while j < num_data_points:

        b = data['title'].loc[indices[j]].split()
        length = max(len(a), len(b))
        count  = 0
        for k in itertools.zip_longest(a,b):
            if (k[0] == k[1]):
                count += 1

        them
        if (length - count) > 2:
            stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[i]])
            i = j
            break
        else:
            j += 1
    if previous_i == i:
        break
```

In [0]:

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

In [0]:

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

```
Number of data points :  17593
```

In [0]:

```
data.to_pickle('pickels/17k_apperal_data')
```

Remove duplicates - 2

In [ ]:

```
# Removing products with similar Title
```

In [0]:

```
data = pd.read_pickle('pickels/17k_apperal_data')
```

In [0]:

```
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])

    a = data['title'].loc[i].split()

    for j in indices:

        b = data['title'].loc[j].split()

        length = max(len(a),len(b))
        count  = 0
        for k in itertools.zip_longest(a,b):
            if (k[0]==k[1]):
                count += 1
        if (length - count) < 3:
            indices.remove(j)
```

In [0]:

```
data = data.loc[data['asin'].isin(stage2_dedupe_asins)]
```

In [0]:

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

```
Number of data points after stage two of dedupe:  16042
```

Text pre-processing

In [0]:

```
# we use the list of stop words that are downloaded from nltk lib.
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():

            word = ("".join(e for e in words if e.isalnum()))
            word = word.lower()
            if not word in stop_words:
                string += word + " "
        data[column][index] = string
```

```
list of stop words: {'such', 'and', 'hers', 'up', 'she', 'd', 'further', 'al
l', 'than', 'under', 'is', 'off', 'both', 'most', 'few', 'should', 're', 've
ry', 'just', 'then', 'didn', 'myself', 'in', 'too', 's', 'shouldn', 'hersel
f', 'because', 'how', 'itself', 'what', 'shan', 'weren', 'doing', 'them', 'c
ouldn', 'their', 'so', 'ain', 'haven', 'yourself', 'now', 'll', 'isn', 'abou
t', 'over', 'into', 'before', 'during', 'on', 'as', 'aren', 'against', 'abov
e', 'down', 'they', 'below', 'me', 'again', 'for', 'why', 'been', 'yourselve
s', 'more', 'her', 'that', 'can', 'am', 'was', 'themselves', 'mightn', 'doe
s', 'those', 'only', 'hasn', 'any', 'ma', 'are', 'nor', 'out', 'you', 'ourse
lves', 'the', 'an', 'has', 'where', 'i', 'while', 'ours', 'its', 'your', 'ha
d', 'were', 'being', 'no', 'or', 'needn', 've', 'y', 'a', 'each', 'have', 't
hrough', 'when', 'mustn', 'by', 'won', 'from', 'own', 'will', 'there', 't',
'him', 'these', 'doesn', 'theirs', 'my', 'did', 'of', 'who', 'until', 'would
n', 'we', 'do', 'having', 'yours', 'other', 'wasn', 'it', 'with', 'once', 'h
ere', 'don', 'o', 'whom', 'this', 'if', 'but', 'hadn', 'our', 'some', 'm',
'not', 'between', 'himself', 'same', 'at', 'be', 'he', 'after', 'which', 't
o', 'his'}
```

In [0]:

```
start_time = time.clock()

for index, row in data.iterrows():
    nlp_preprocessing(row['title'], index, 'title')

print(time.clock() - start_time, "seconds")
```

```
3.5727220000000006 seconds
```

In [0]:

```
1 data.head()
```

Out[6]:

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

Stemming

In [0]:

```
1 from nltk.stem.porter import *
2 stemmer = PorterStemmer()
3 print(stemmer.stem('arguing'))
4 print(stemmer.stem('fishing'))
5
6
7 # Stemming on titles did not work very well.
8
```

```
argu
fish
```

Text based product similarity

In [0]:

```
data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
data.head()
```

Out[10]:

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

In [0]:

```
def display_img(url,ax,fig):

    response = requests.get(url)
    img = Image.open(BytesIO(response.content))

    plt.imshow(img)

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


        gs = gridspec.GridSpec(2, 2, width_ratios=[4,1], height_ratios=[4,1])
        fig = plt.figure(figsize=(25,3))


        ax = plt.subplot(gs[0])

        ax = sns.heatmap(np.array([values]), annot=np.array([labels]))
        ax.set_xticklabels(keys)
        ax.set_title(text)
        ax = plt.subplot(gs[1])

        ax.grid(False)
        ax.set_xticks([])
        ax.set_yticks([])

        display_img(url, ax, fig)

        plt.show()

def plot_heatmap_image(doc_id, vec1, vec2, url, text, model):

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

    for i in vec2:
        if i not in intersection:
            vec2[i]=0


    keys = list(vec2.keys())

    values = [vec2[x] for x in vec2.keys()]

    if model == 'bag_of_words':
        labels = values
    elif model == 'tfidf':
        labels = []
        for x in vec2.keys():

            if x in  tfidf_title_vectorizer.vocabulary_:
                labels.append(tfidf_title_features[doc_id, tfidf_title_vectorizer.vocab
            else:
                labels.append(0)
    elif model == 'idf':
        labels = []
        for x in vec2.keys():

            if x in  idf_title_vectorizer.vocabulary_:
                labels.append(idf_title_features[doc_id, idf_title_vectorizer.vocabular
```

```
            else:
                labels.append(0)

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

def text_to_vector(text):
    word = re.compile(r'\w+')
    words = word.findall(text)

    return Counter(words)



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


    vector1 = text_to_vector(text1)

    vector2 = text_to_vector(text2)

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

Bag of Words (BoW) on product titles.

In [0]:

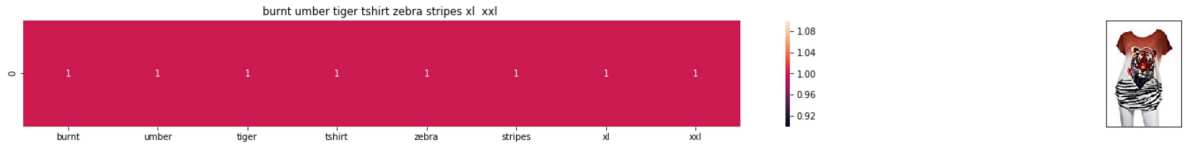
```
from sklearn.feature_extraction.text import CountVectorizer
title_vectorizer = CountVectorizer()
title_features   = title_vectorizer.fit_transform(data['title'])
title_features.get_shape()
```

Out[17]:

```
(16042, 12609)
```

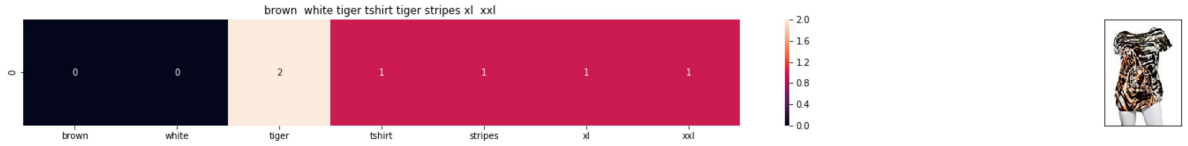
In [0]:

```
def bag_of_words_model(doc_id, num_results):

    pairwise_dist = pairwise_distances(title_features,title_features[doc_id])

    indices = np.argsort(pairwise_dist.flatten())[0:num_results]

    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]
    df_indices = list(data.index[indices])

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


bag_of_words_model(12566, 20)
```



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

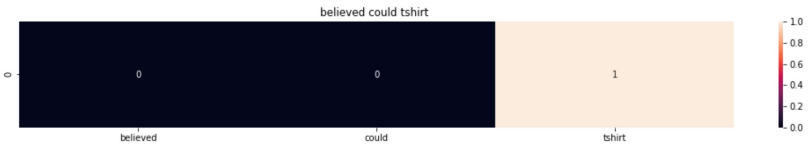
ASIN : B00JXQASS6
Brand: Si Row
Title: pink tiger tshirt zebra stripes xl xxl
Euclidean similarity with the query image : 1.73205080757
==



ASIN : B00JXQCWTO
Brand: Si Row
Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean similarity with the query image : 2.44948974278
==

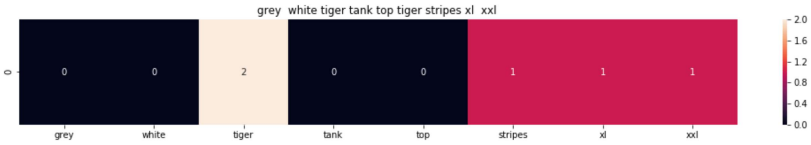
ASIN : B00JXQCUIC

```
Brand: Si Row
Title: yellow tiger tshirt tiger stripes  l
Euclidean similarity with the query image : 2.64575131106
============================================================
```

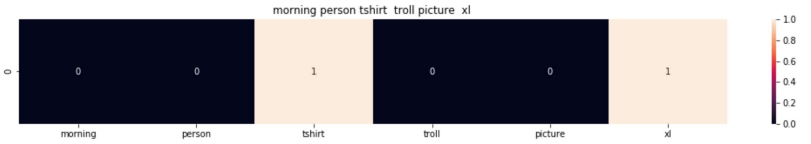


```
ASIN : B07568NZX4
Brand: Rustic Grace
Title: believed could tshirt
Euclidean similarity with the query image : 3.0
============================================================
```

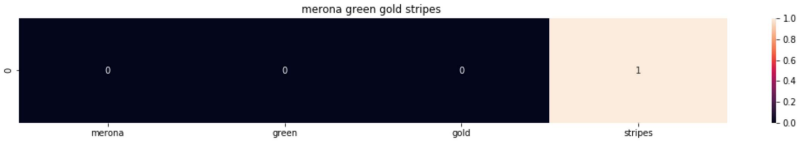
```
ASIN : B01NB0NKRO
Brand: Ideology
Title: ideology graphic tshirt xl white
Euclidean similarity with the query image : 3.0
============================================================
```



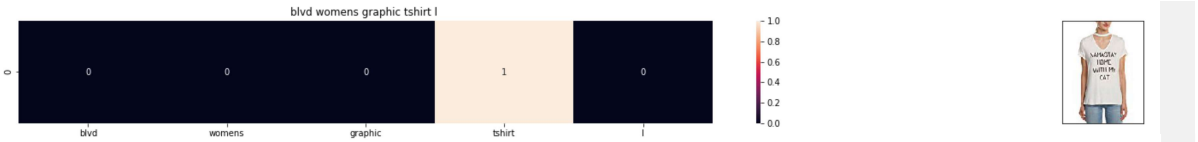
```
ASIN : B00JXQAFZ2
Brand: Si Row
Title: grey  white tiger tank top tiger stripes xl  xxl
Euclidean similarity with the query image : 3.0
============================================================
```



```
ASIN : B01CLS8LMW
Brand: Awake
Title: morning person tshirt  troll picture  xl
Euclidean similarity with the query image : 3.16227766017
============================================================
```

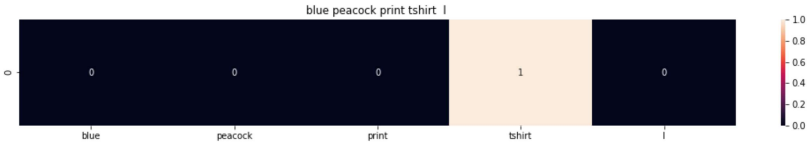


```
ASIN : B01KVZUB6G
Brand: Merona
Title: merona green gold stripes
Euclidean similarity with the query image : 3.16227766017
============================================================
```



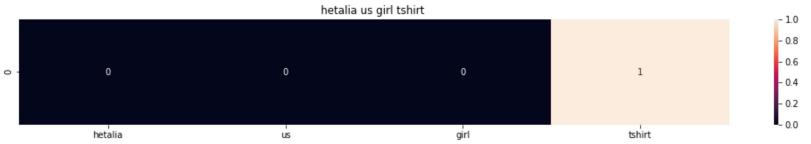
ASIN : B0733R2CJK
Brand: BLVD
Title: blvd womens graphic tshirt l
Euclidean similarity with the query image : 3.16227766017
==========

ASIN : B012VQLT6Y
Brand: KM T-shirt
Title: km tiger printed sleeveless vest tshirt
Euclidean similarity with the query image : 3.16227766017
==========

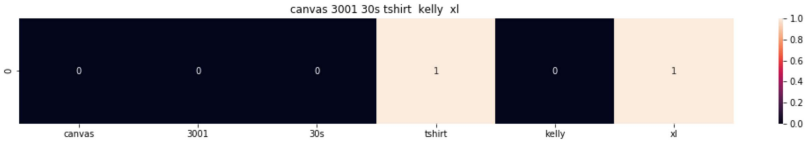


ASIN : B00JXQC8L6
Brand: Si Row
Title: blue peacock print tshirt l
Euclidean similarity with the query image : 3.16227766017
==========

ASIN : B06XC3CZF6
Brand: Fjallraven
Title: fjallraven womens ovik tshirt plum xxl
Euclidean similarity with the query image : 3.16227766017
==========

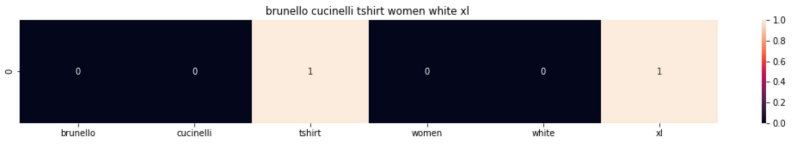


ASIN : B005IT8OBA
Brand: Hetalia
Title: hetalia us girl tshirt
Euclidean similarity with the query image : 3.16227766017
==========



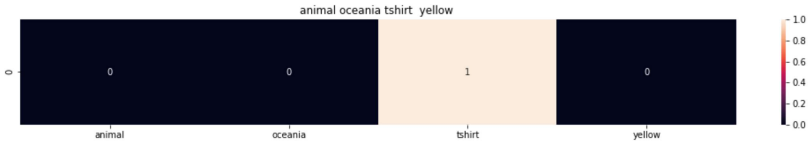
ASIN : B0088PN0LA
Brand: Red House

```
Title: canvas 3001 30s tshirt  kelly  xl
Euclidean similarity with the query image : 3.16227766017
============================================================
```

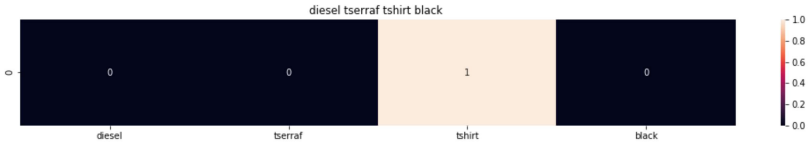


```
ASIN : B06X99V6WC
Brand: Brunello Cucinelli
Title: brunello cucinelli tshirt women white xl
Euclidean similarity with the query image : 3.16227766017
============================================================
```

```
ASIN : B06Y1JPW1Q
Brand: Xhilaration
Title: xhilaration womens lace tshirt salmon xxl
Euclidean similarity with the query image : 3.16227766017
============================================================
```



```
ASIN : B06X6GX6WG
Brand: Animal
Title: animal oceania tshirt  yellow
Euclidean similarity with the query image : 3.16227766017
============================================================
```



```
ASIN : B017X8PW9U
Brand: Diesel
Title: diesel tserraf tshirt black
Euclidean similarity with the query image : 3.16227766017
============================================================
```

```
ASIN : B00IAA4JIQ
Brand: I Love Lucy
Title: juniors love lucywaaaaahhhh tshirt size xl
Euclidean similarity with the query image : 3.16227766017
============================================================
```

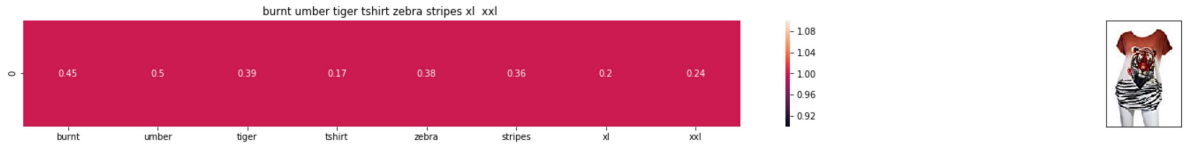
TF-IDF based product similarity

In [0]:

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

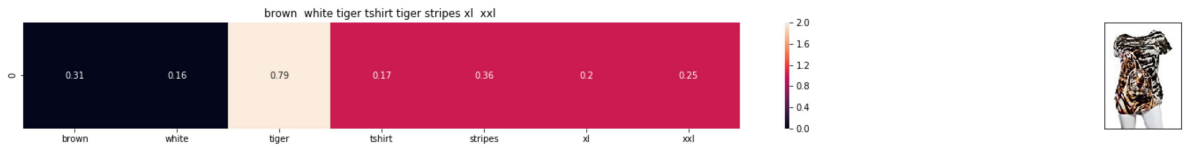
In [0]:

```
def tfidf_model(doc_id, num_results):

    pairwise_dist = pairwise_distances(tfidf_title_features,tfidf_title_features[doc_id

    indices = np.argsort(pairwise_dist.flatten())[0:num_results]

    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]


    df_indices = list(data.index[indices])

    for i in range(0,len(indices)):

        get_result(indices[i], data['title'].loc[df_indices[0]], data['title'].loc[df_i
        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)
```

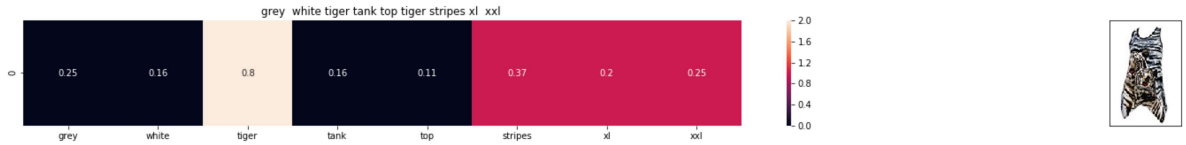


ASIN : B00JXQB5FQ
BRAND : Si Row
Eucliden distance from the given image : 0.0
==========
==========

ASIN : B00JXQASS6
BRAND : Si Row
Eucliden distance from the given image : 0.753633191245
==========
==========



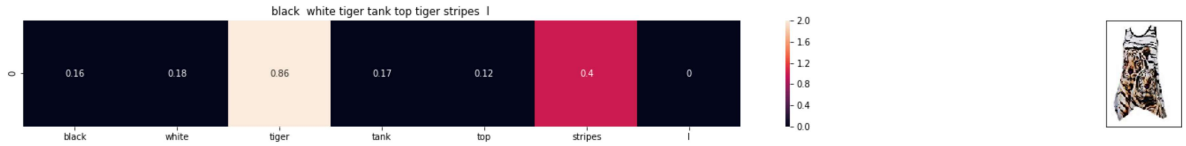
ASIN : B00JXQCWTO
BRAND : Si Row
Eucliden distance from the given image : 0.935764394377
==========
==========



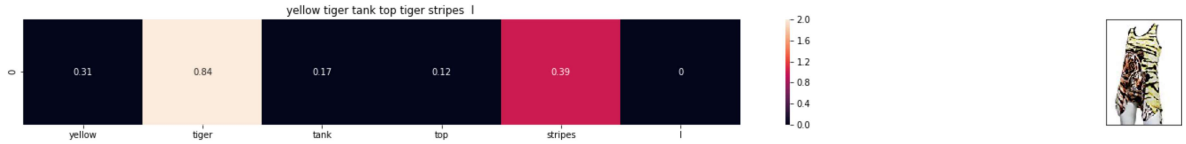
ASIN : B00JXQAFZ2

BRAND : Si Row
Eucliden distance from the given image : 0.95861535242
==========
==========

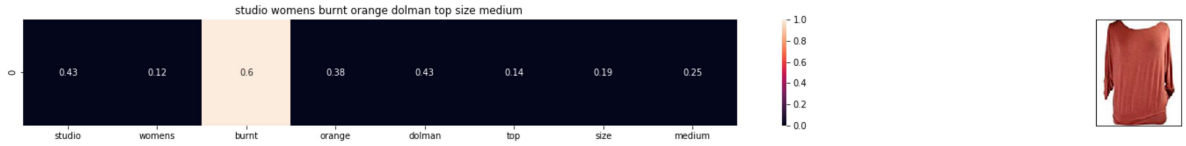
ASIN : B00JXQCUIC
BRAND : Si Row
Eucliden distance from the given image : 1.00007496145
==========
==========



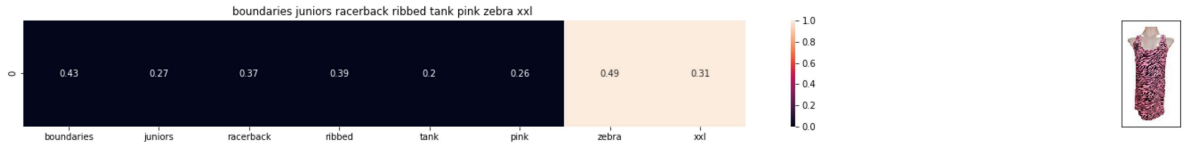
ASIN : B00JXQAO94
BRAND : Si Row
Eucliden distance from the given image : 1.02321555246
==========
==========



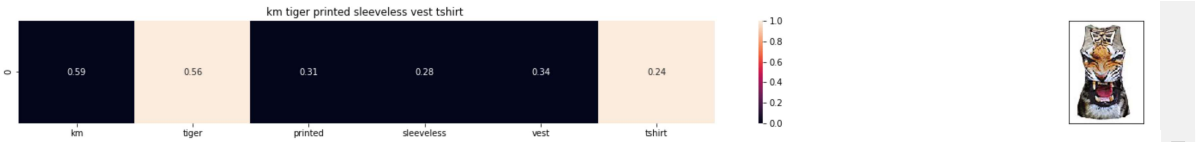
ASIN : B00JXQAUWA
BRAND : Si Row
Eucliden distance from the given image : 1.0319918463
==========
==========



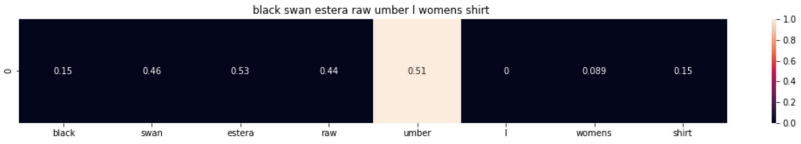
ASIN : B06XSCVFT5
BRAND : Studio M
Eucliden distance from the given image : 1.21068436704
==========
==========



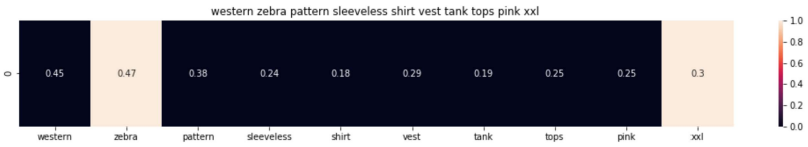
ASIN : B06Y2GTYPM
BRAND : No Boundaries
Eucliden distance from the given image : 1.21216838107
==========
==========



ASIN : B012VQLT6Y
BRAND : KM T-shirt
Eucliden distance from the given image : 1.21979064028
==========
==========

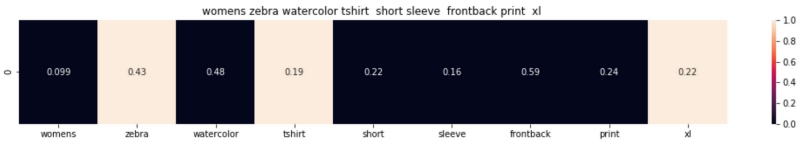


ASIN : B06Y1VN8WQ
BRAND : Black Swan
Eucliden distance from the given image : 1.220684966
==========
==========

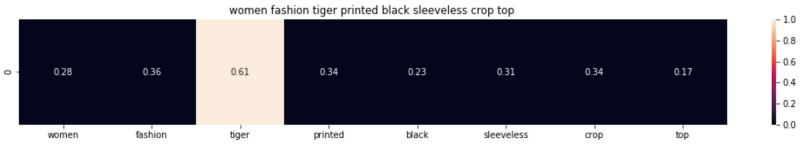


ASIN : B00Z6HEXWI
BRAND : Black Temptation
Eucliden distance from the given image : 1.22128139212
==========
==========

ASIN : B074TR12BH
BRAND : Ultra Flirt
Eucliden distance from the given image : 1.23133640946
==========
==========

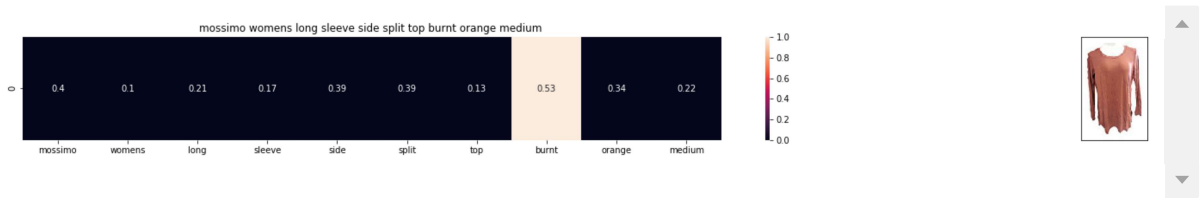


ASIN : B072R2JXKW
BRAND : WHAT ON EARTH
Eucliden distance from the given image : 1.23184519726
==========
==========

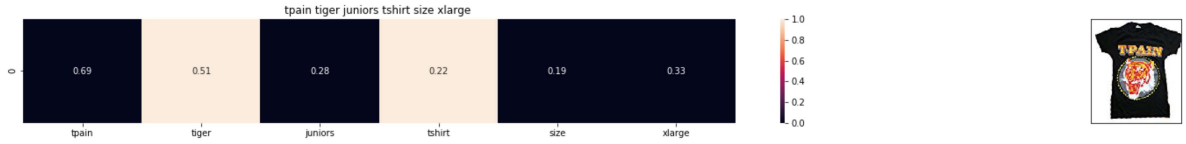


ASIN : B074T8ZYGX
BRAND : MKP Crop Top

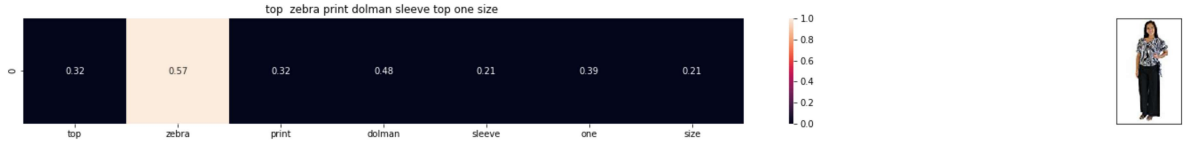
```
Eucliden distance from the given image : 1.23406074574
============================================================================
```




```
ASIN : B071ZDF6T2
BRAND : Mossimo
Eucliden distance from the given image : 1.23527855777
============================================================================
============================================================================
```

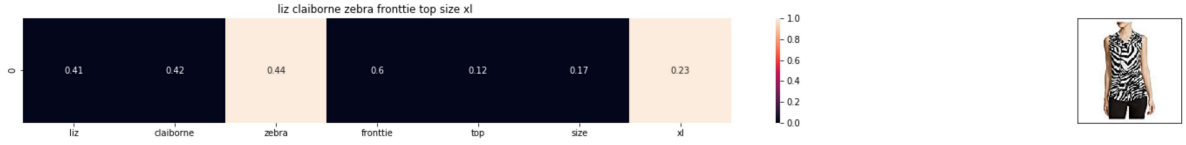



```
ASIN : B01K0H02OG
BRAND : Tultex
Eucliden distance from the given image : 1.23645729881
============================================================================
============================================================================
```




```
ASIN : B00H8A6ZLI
BRAND : Vivian's Fashions
Eucliden distance from the given image : 1.24996155053
============================================================================
============================================================================
```


```
ASIN : B010NN9RXO
BRAND : YICHUN
Eucliden distance from the given image : 1.25354614209
============================================================================
============================================================================
```




```
ASIN : B06XBY5QXL
BRAND : Liz Claiborne
Eucliden distance from the given image : 1.25388329384
============================================================================
============================================================================
```

IDF based product similarity

In [0]:

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

In [0]:

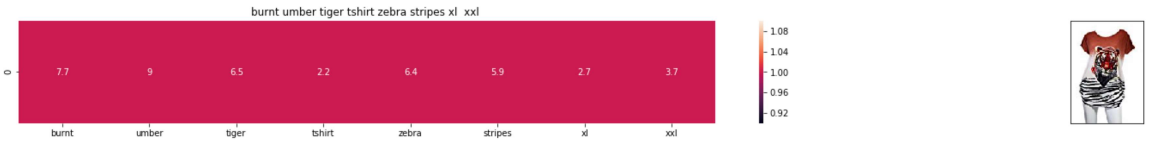
```
def n_containing(word):

    return sum(1 for blob in data['title'] if word in blob.split())

def idf(word):

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

In [0]:

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

for i in idf_title_vectorizer.vocabulary_.keys():

    idf_val = idf(i)


    for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]:

        idf_title_features[j,idf_title_vectorizer.vocabulary_[i]] = idf_val

```

In [0]:

```
def idf_model(doc_id, num_results):
    pairwise_dist = pairwise_distances(idf_title_features,idf_title_features[doc_id])

    indices = np.argsort(pairwise_dist.flatten())[0:num_results]

    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]


    df_indices = list(data.index[indices])

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




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


```
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from the given image : 12.2050713112
=============================================================================
```

Text Semantics based product similarity

In [0]:

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

with open('word2vec_model', 'rb') as handle:
    model = pickle.load(handle)

```

In [0]:

```
def get_word_vec(sentence, doc_id, m_name):

    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in  idf_title_vectorizer.vocabulary_:
                vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[
            elif m_name == 'avg':
                vec.append(model[i])
        else:

            vec.append(np.zeros(shape=(300,)))
    return  np.array(vec)

def get_distance(vec1, vec2):

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


def heat_map_w2v(sentence1, sentence2, url, doc_id1, doc_id2, model):

    s1_vec = get_word_vec(sentence1, doc_id1, model)
    s2_vec = get_word_vec(sentence2, doc_id2, model)

    s1_s2_dist = get_distance(s1_vec, s2_vec)




    gs = gridspec.GridSpec(2, 2, width_ratios=[4,1],height_ratios=[2,1])
    fig = plt.figure(figsize=(15,15))

    ax = plt.subplot(gs[0])

    ax = sns.heatmap(np.round(s1_s2_dist,4), annot=True)

    ax.set_xticklabels(sentence2.split())

    ax.set_yticklabels(sentence1.split())

    ax.set_title(sentence2)

    ax = plt.subplot(gs[1])

    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])
    display_img(url, ax, fig)

    plt.show()
```

In [0]:

```
vocab = model.keys()

def build_avg_vec(sentence, num_features, doc_id, m_name):

    featureVec = np.zeros((num_features,), dtype="float32")

    nwords = 0

    for word in sentence.split():
        nwords += 1
        if word in vocab:
            if m_name == 'weighted' and word in  idf_title_vectorizer.vocabulary_:
                featureVec = np.add(featureVec, idf_title_features[doc_id, idf_title_ve
            elif m_name == 'avg':
                featureVec = np.add(featureVec, model[word])
    if(nwords>0):
        featureVec = np.divide(featureVec, nwords)

    return featureVec
```

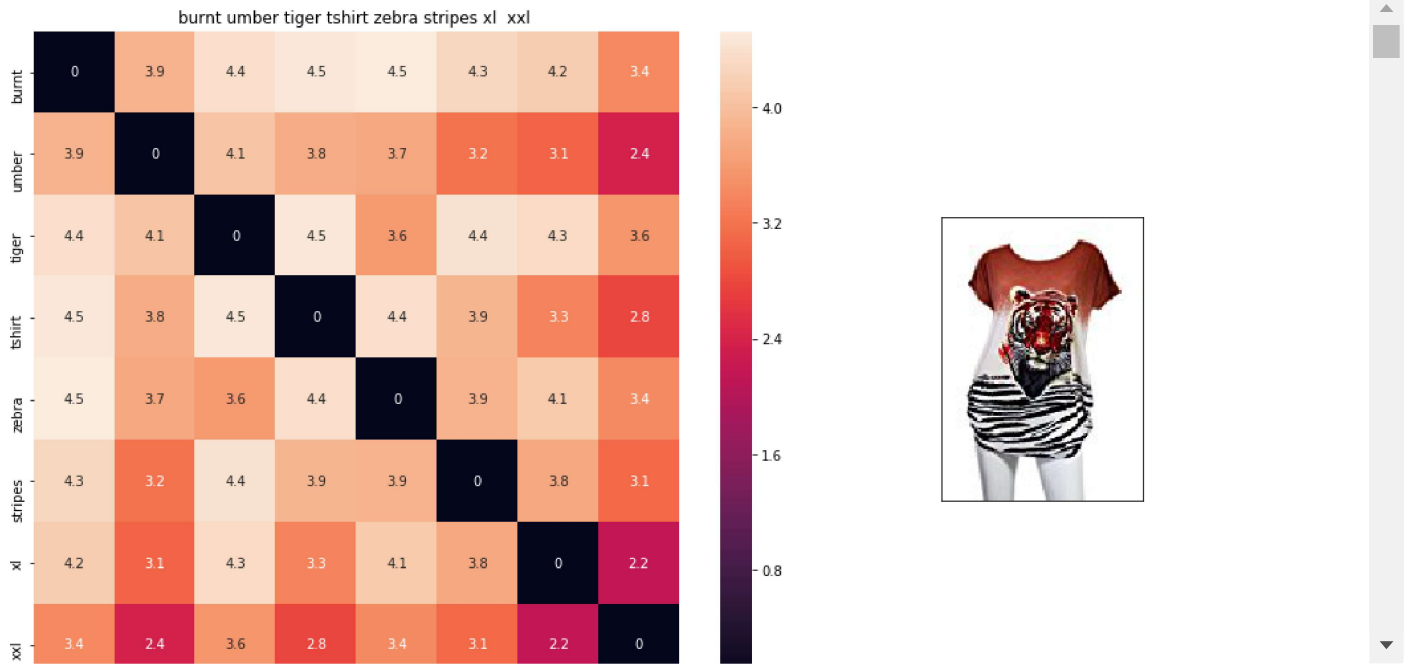
Average Word2Vec product similarity.

In [0]:

```
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)

```

In [0]:

```
def avg_w2v_model(doc_id, num_results):

    pairwise_dist = pairwise_distances(w2v_title, w2v_title[doc_id].reshape(1,-1))

    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(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 given input image :', pdists[i])
        print('='*125)


avg_w2v_model(12566, 20)

```



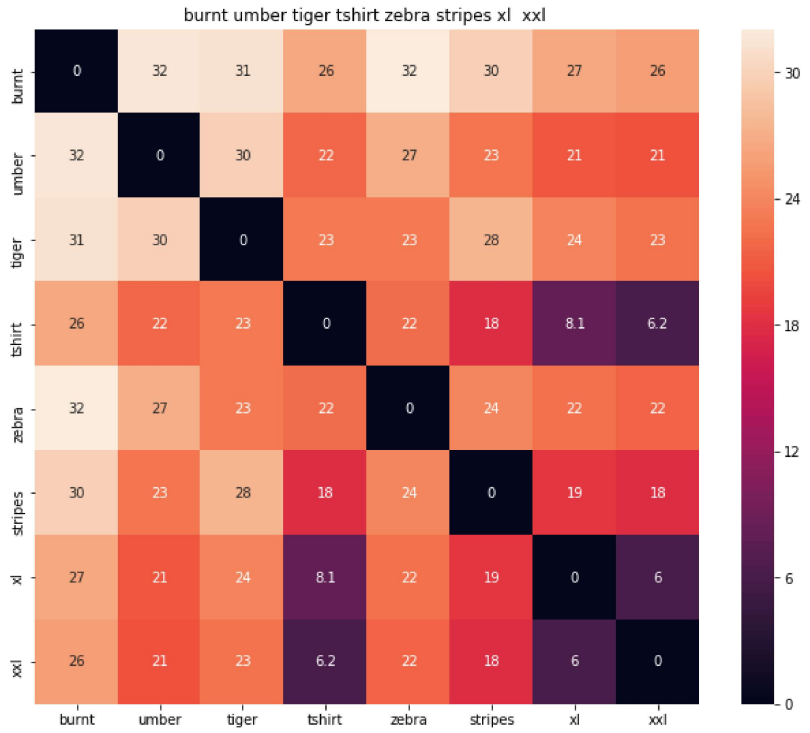
IDF weighted Word2Vec for product similarity

In [0]:

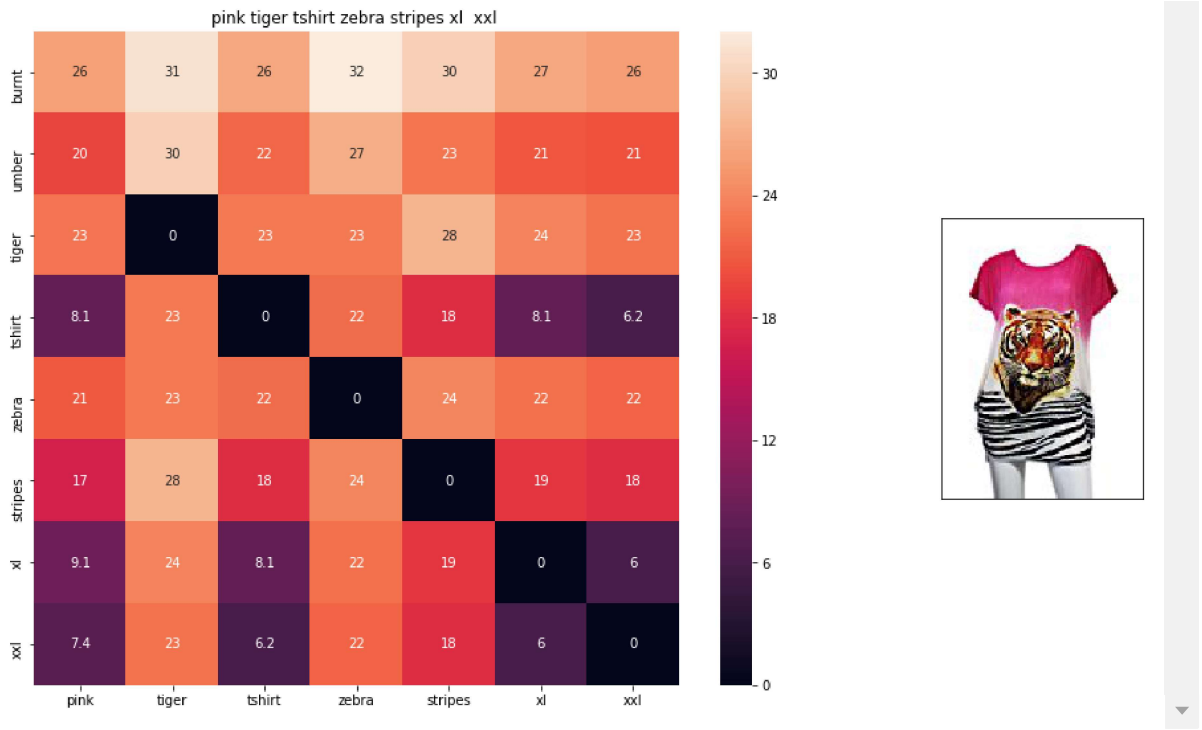
```
doc_id = 0
w2v_title_weight = []
# for every title we build a weighted vector representation
for i in data['title']:
    w2v_title_weight.append(build_avg_vec(i, 300, doc_id,'weighted'))
    doc_id += 1
# w2v_title = np.array(# number of doc in courpus * 300), each row corresponds to a doc
w2v_title_weight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

In [0]:

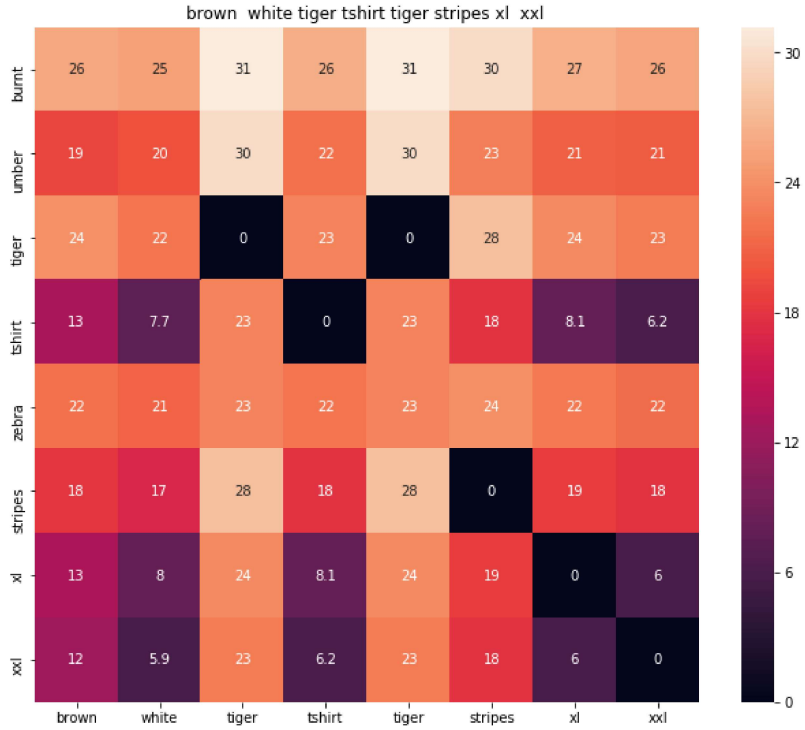
```
def weighted_w2v_model(doc_id, num_results):

    pairwise_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resha

    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(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)

weighted_w2v_model(12566, 20)
```



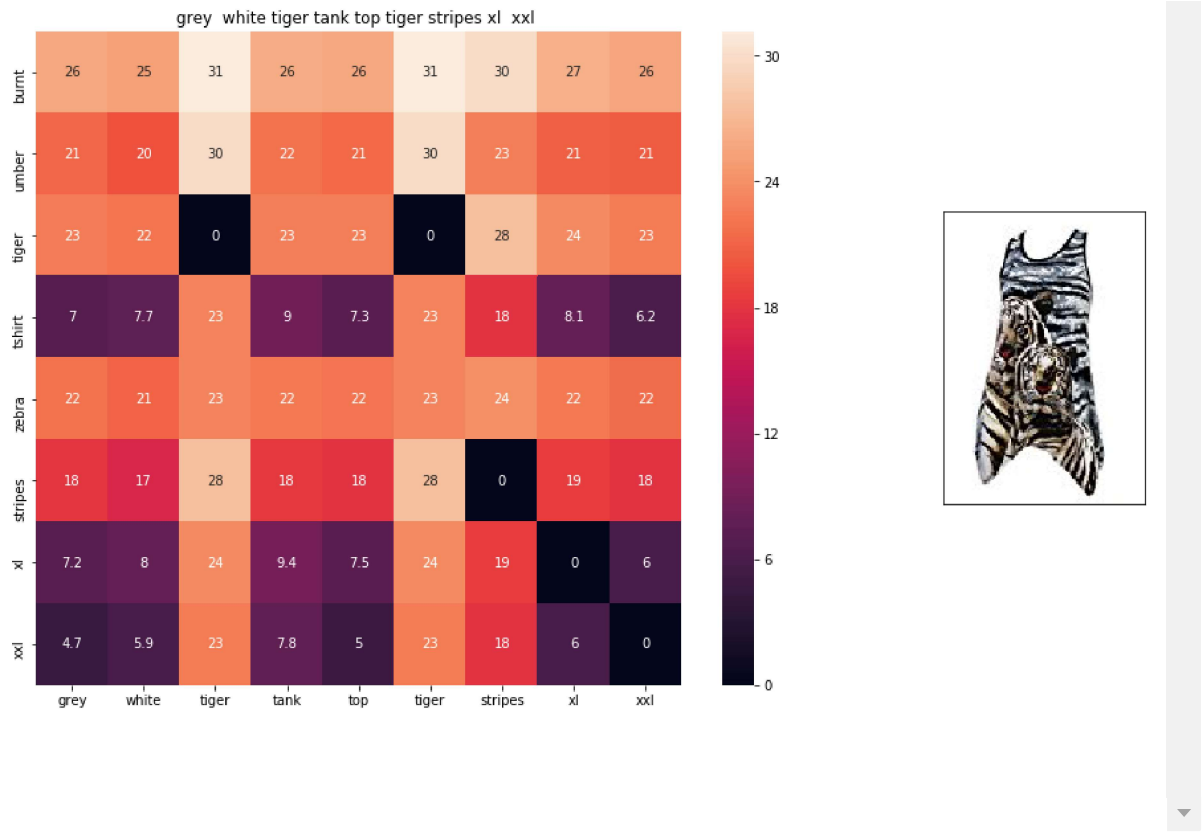
```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.00390625
=============================================================================
==============================================
```



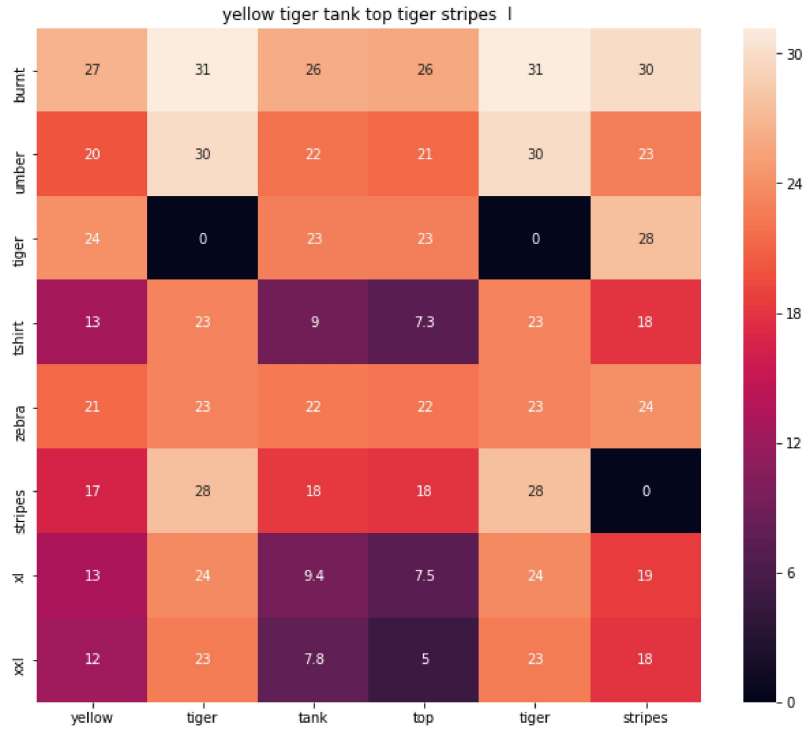
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 4.06389
==========
==========



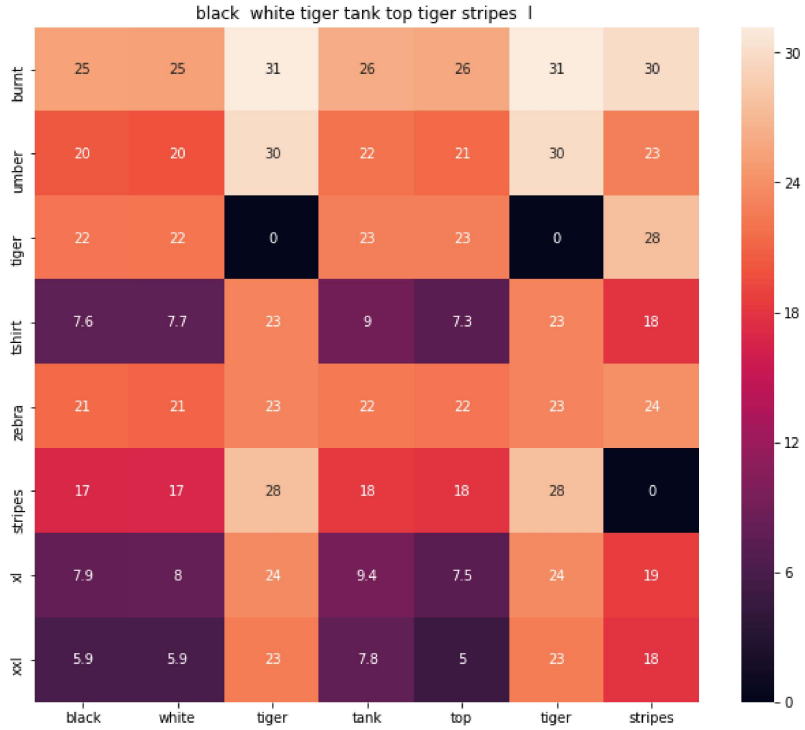
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 4.77094
==========
==========



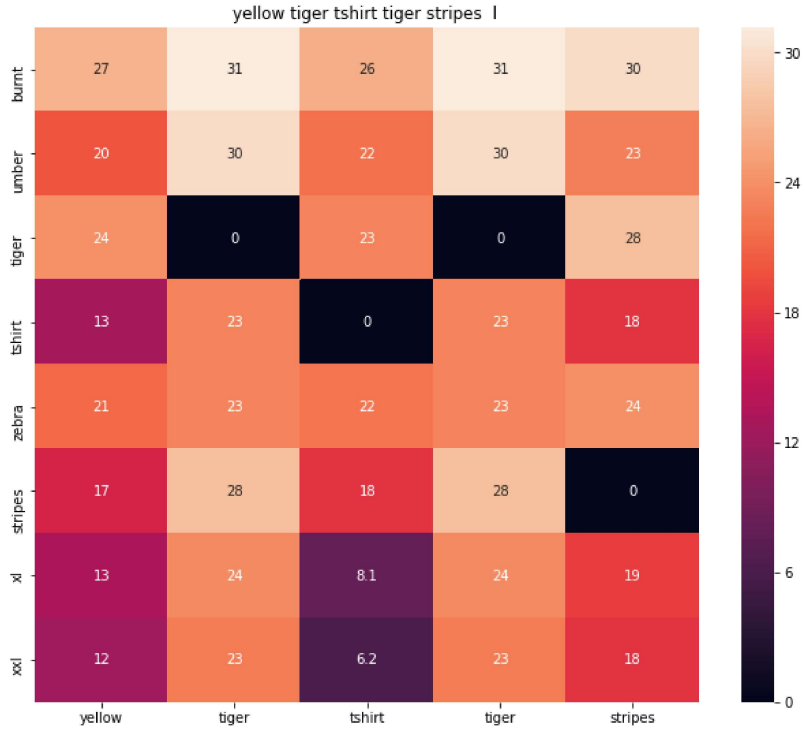
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 5.36016
==========
==========



ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 5.68952
==========
==========

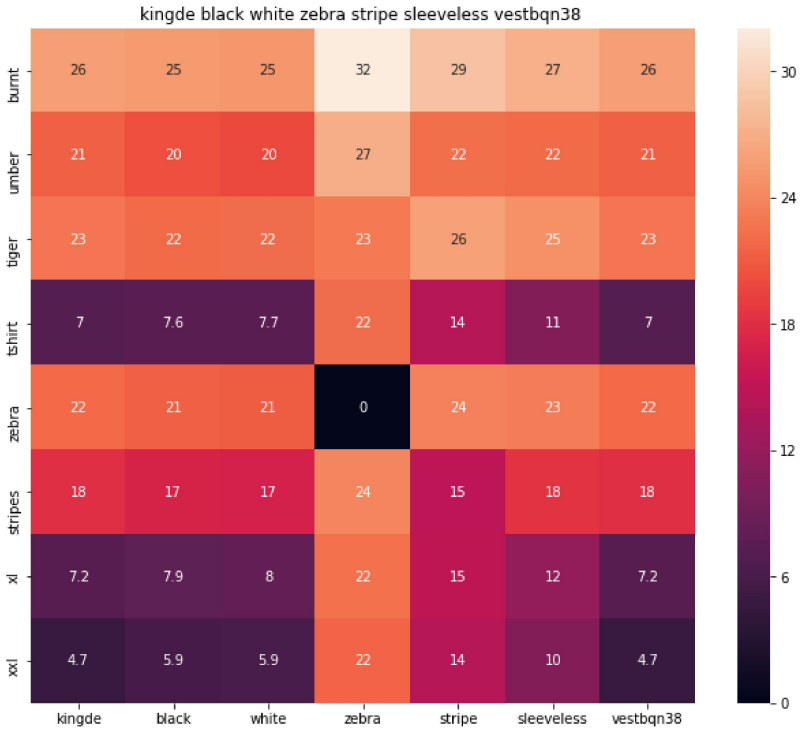


ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 5.69302
==========
==========

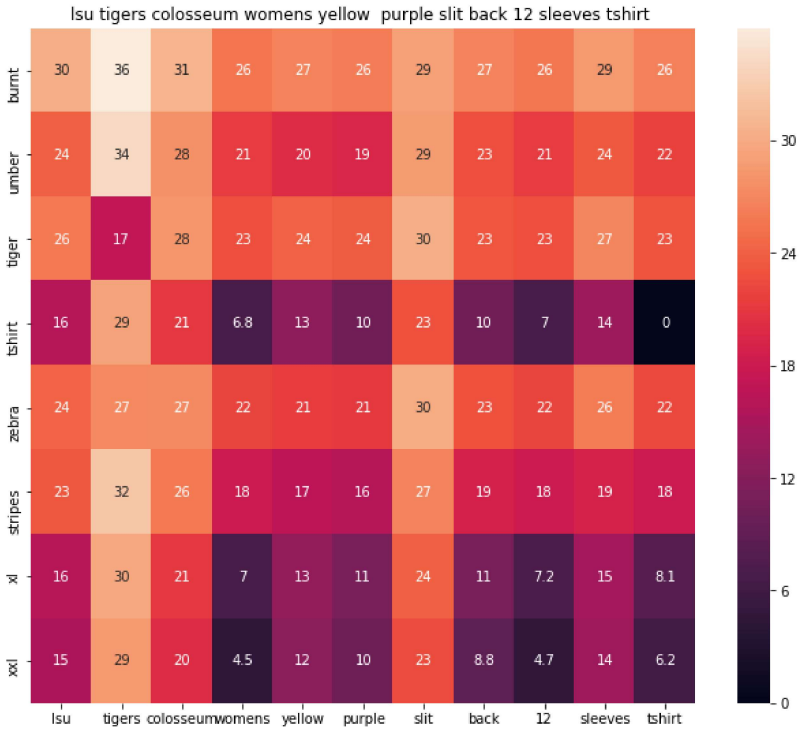


ASIN : B00JXQCUIC
Brand : Si Row

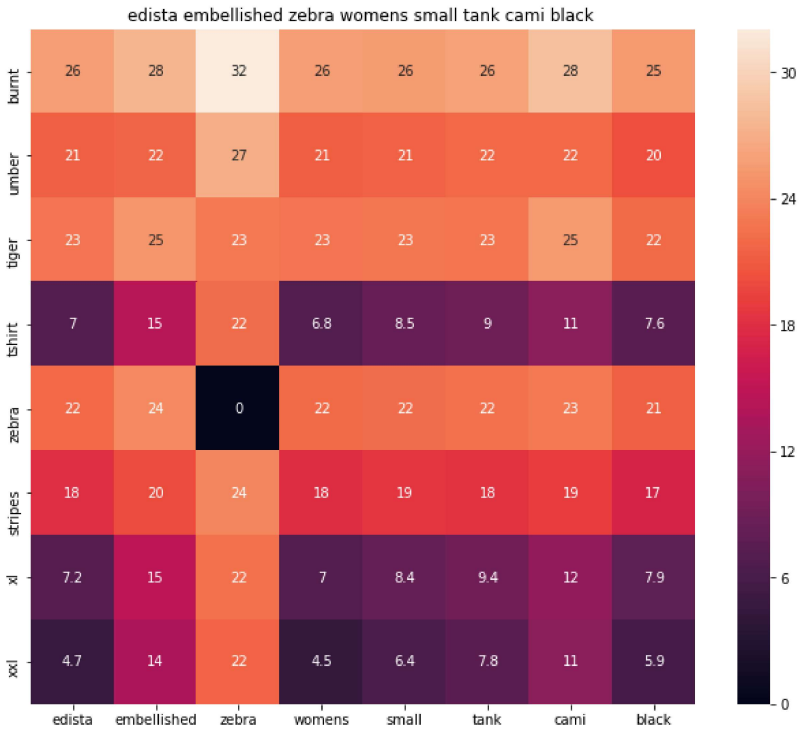
```
euclidean distance from input : 5.89344
================================================================================
================================================
```



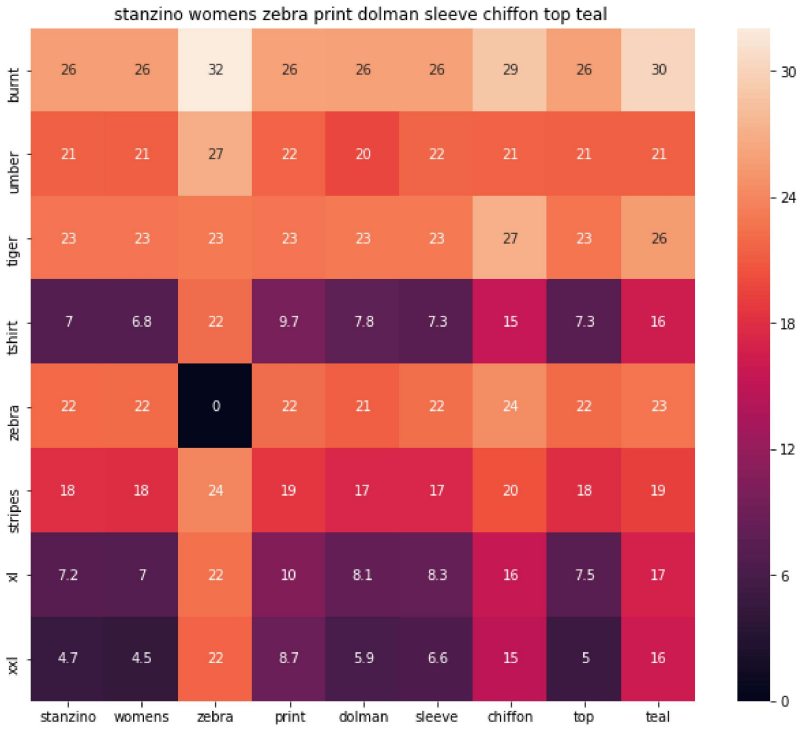
```
ASIN : B015H41F6G
Brand : KINGDE
euclidean distance from input : 6.13299
================================================================================
================================================
```



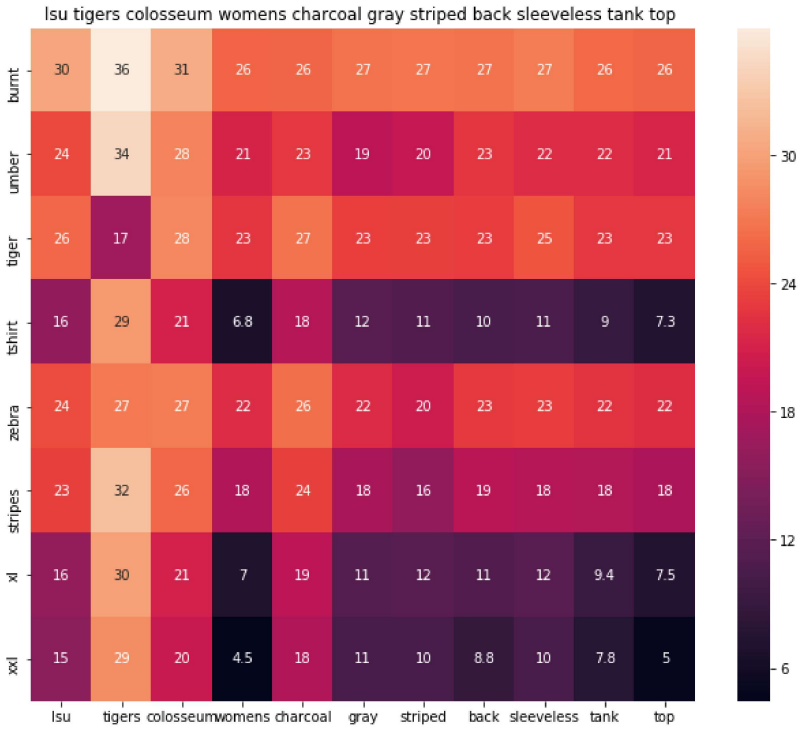
```
ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 6.25671
================================================================================
================================================
```



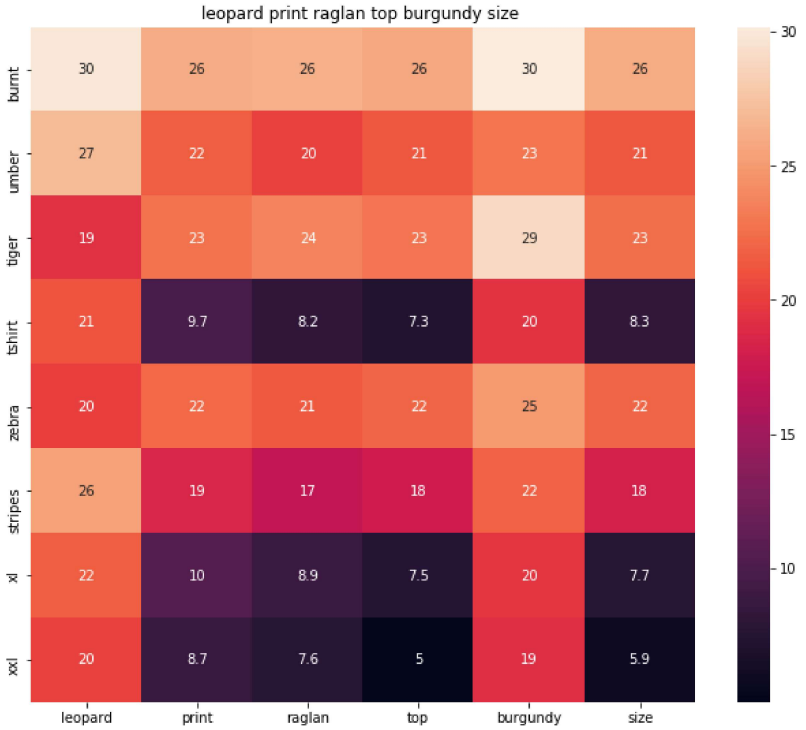
```
ASIN : B074P8MD22
Brand : Edista
euclidean distance from input : 6.3922
================================================================================
================================================
```



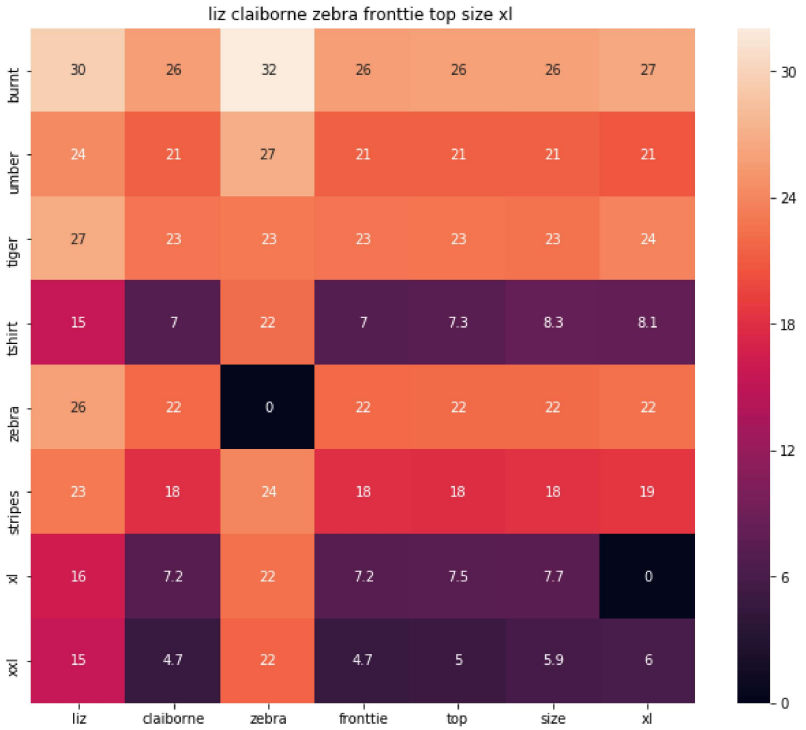
```
ASIN : B00C0I3U3E
Brand : Stanzino
euclidean distance from input : 6.4149
================================================================================
================================================
```



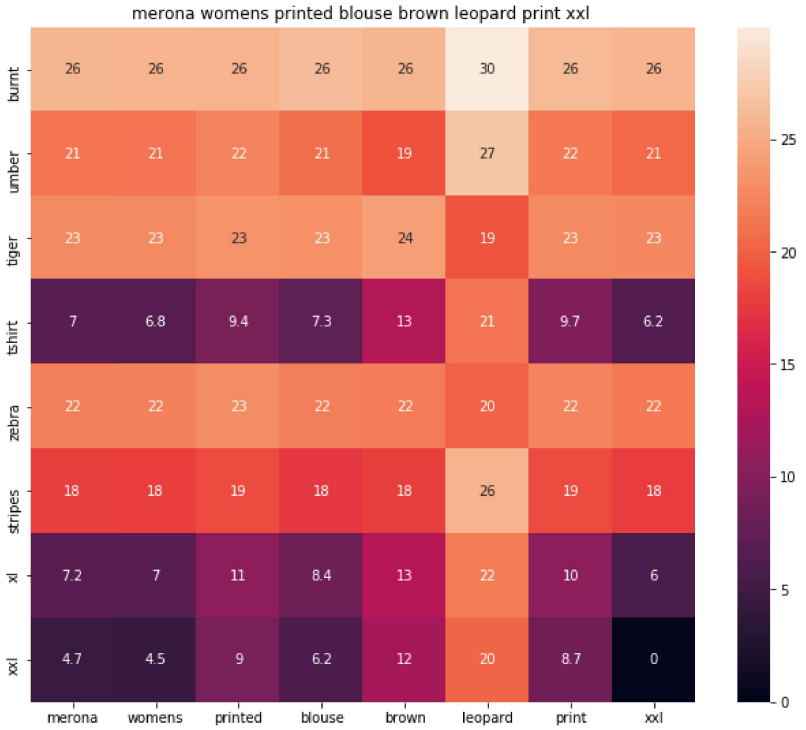
```
ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 6.45096
================================================================================
================================================
```



ASIN : B01C6ORLDQ
Brand : 1 Mad Fit
euclidean distance from input : 6.46341
==========
==========

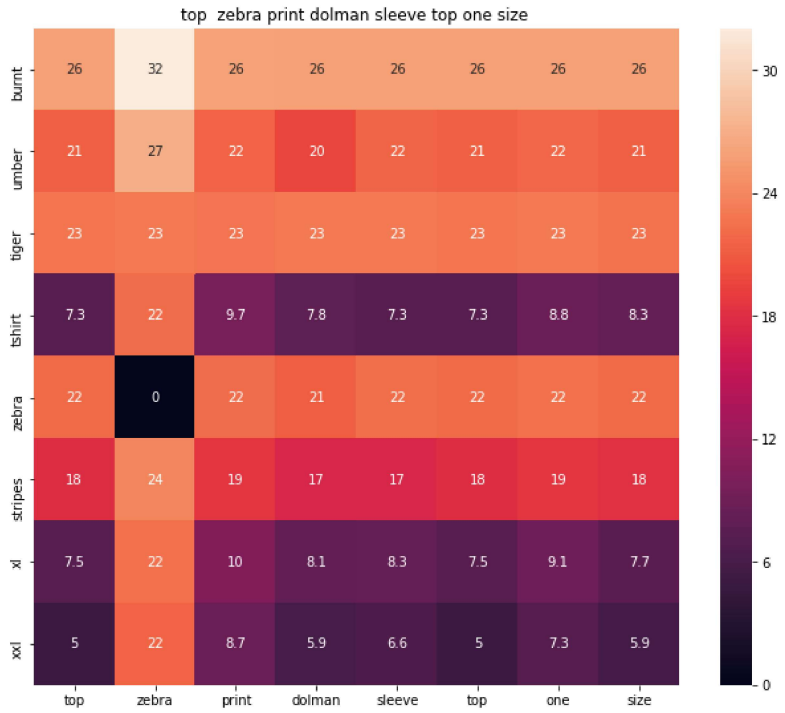


ASIN : B06XBY5QXL
Brand : Liz Claiborne
euclidean distance from input : 6.53922
==========
==========

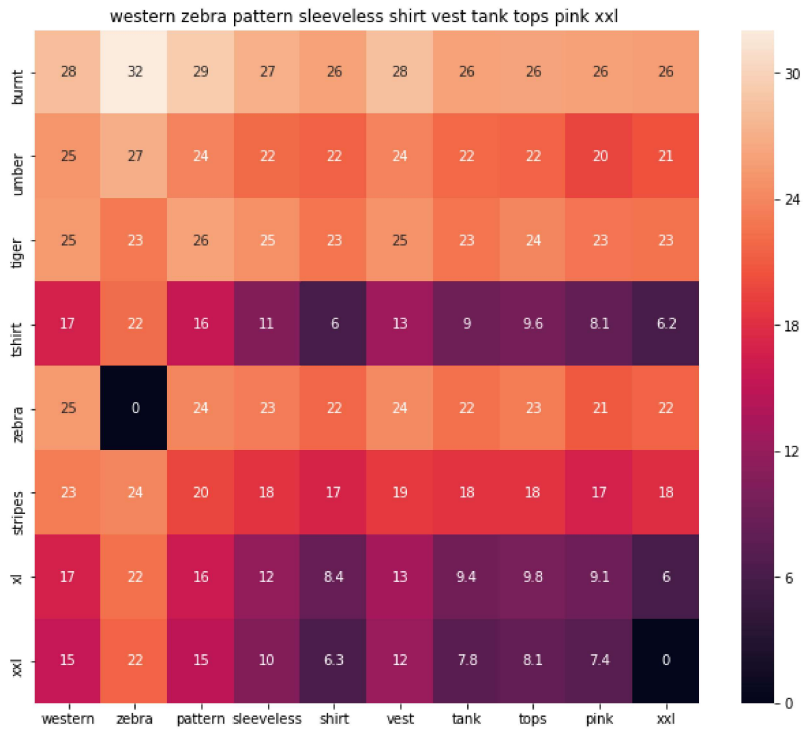


ASIN : B071YF3WDD

```
Brand : Merona
euclidean distance from input : 6.5755
================================================================================
================================================
```

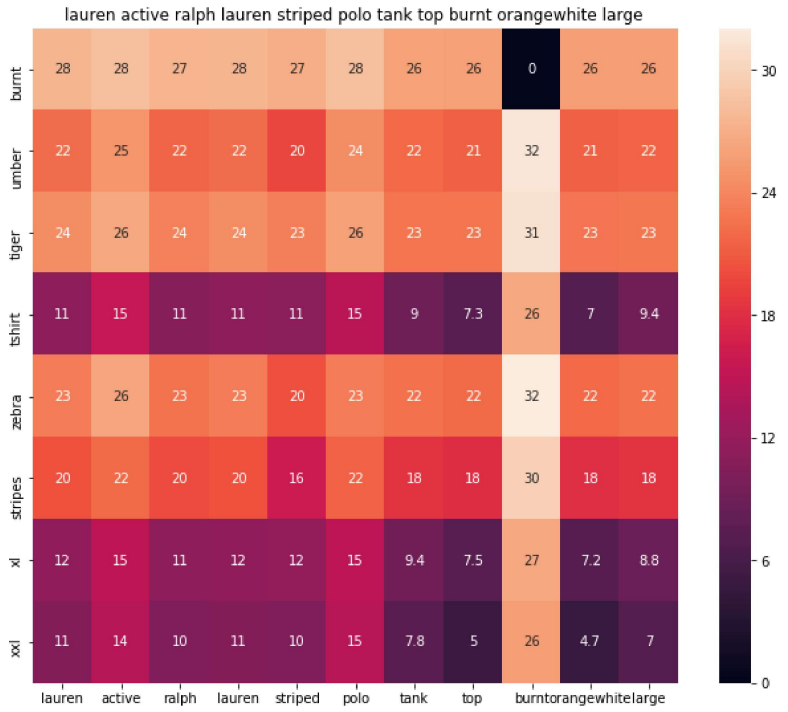


```
ASIN : B00H8A6ZLI
Brand : Vivian's Fashions
euclidean distance from input : 6.63821
================================================================================
================================================
```

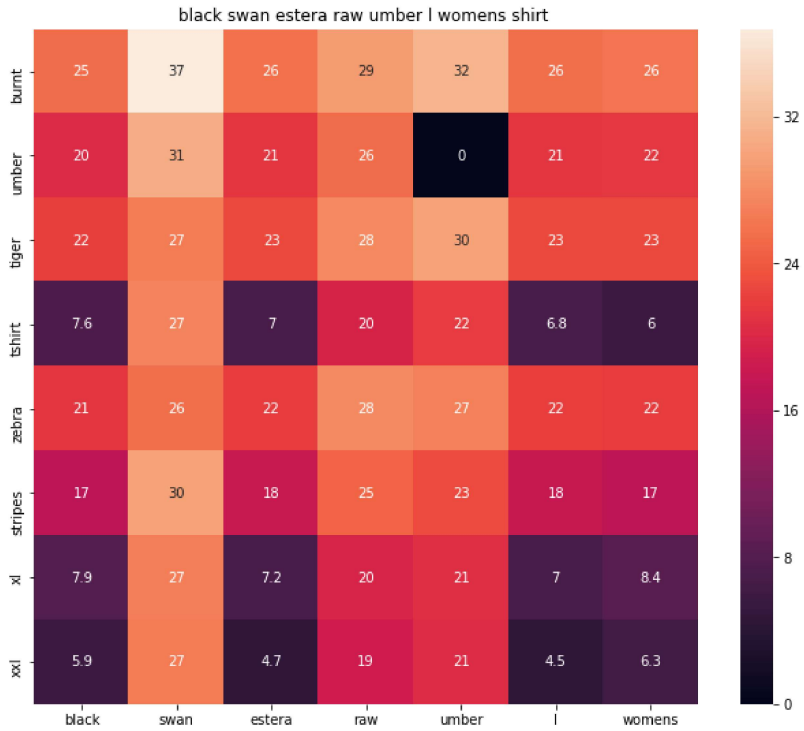


```
ASIN : B00Z6HEXWI
Brand : Black Temptation
```

```
euclidean distance from input : 6.66074
================================================================================
================================================
```

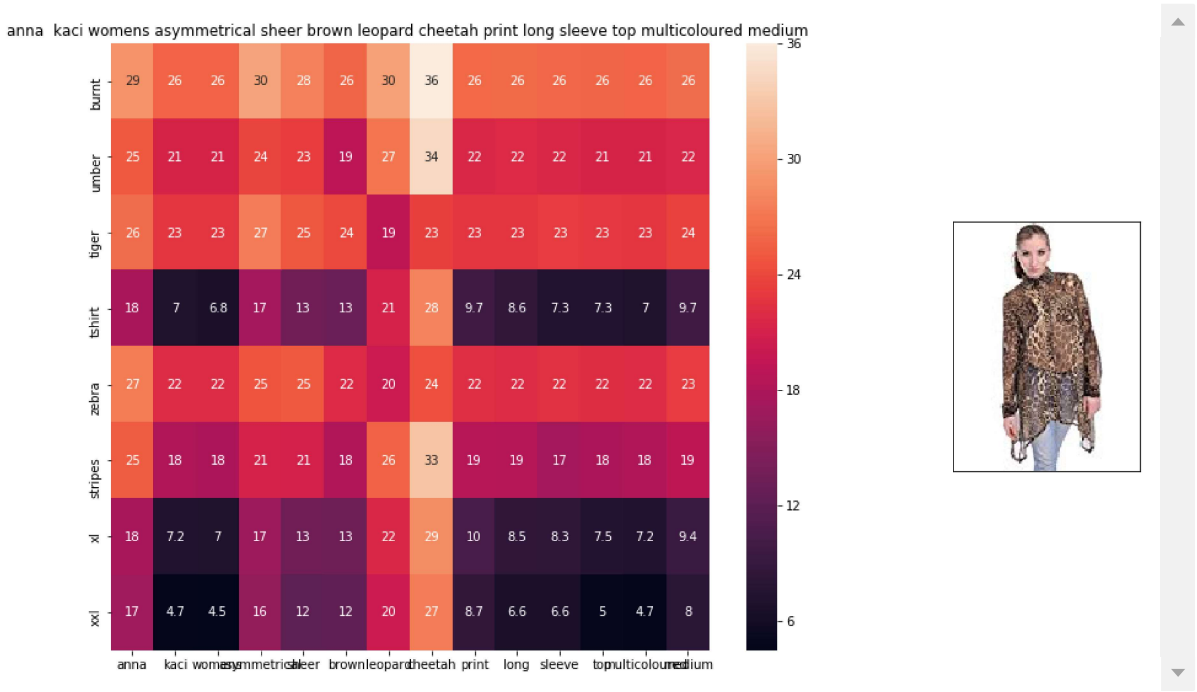


```
ASIN : B00ILGH5OY
Brand : Ralph Lauren Active
euclidean distance from input : 6.68391
================================================================================
================================================
```



```
ASIN : B06Y1VN8WQ
Brand : Black Swan
euclidean distance from input : 6.70576
```

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




```
ASIN : B00KSNTY7Y
Brand : Anna-Kaci
euclidean distance from input : 6.70612
==========================================================================
================================================
```

Weighted similarity using brand and color.

In [0]:

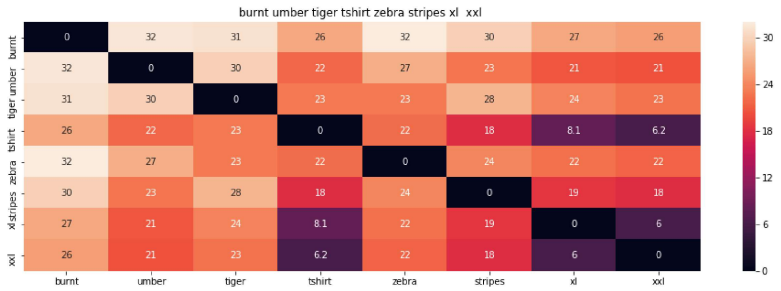
```
# some of the brand values are empty.
# Need to replace Null with string "NULL"
data['brand'].fillna(value="Not given", inplace=True )

# 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()
```

In [0]:

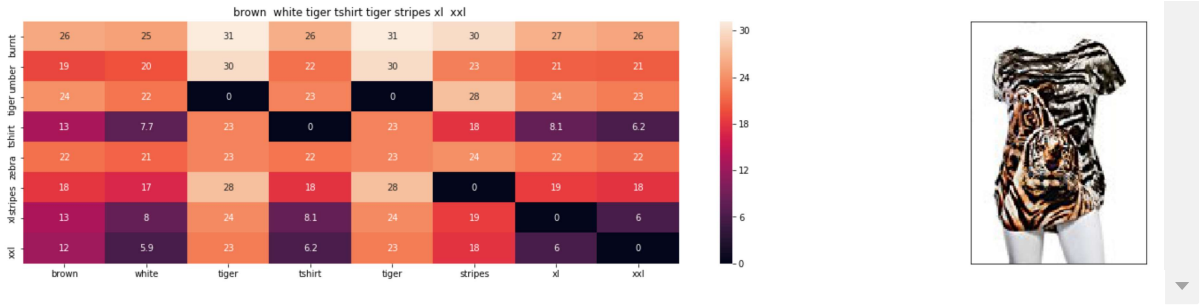
```
def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, mod

    s1_vec = get_word_vec(sentance1, doc_id1, model)
    s2_vec = get_word_vec(sentance2, doc_id2, model)

    s1_s2_dist = get_distance(s1_vec, s2_vec)

    data_matrix = [['Asin','Brand', 'Color', 'Product type'],
                [data['asin'].loc[df_id1],brands[doc_id1], colors[doc_id1], types[doc_id
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], types[doc_id

    colorscale = [[0, '#1d004d'],[.5, '#f2e5ff'],[1, '#f2e5d1']] # to color the heading

    table = ff.create_table(data_matrix, index=True, colorscale=colorscale)
    plotly.offline.iplot(table, filename='simple_table')

    gs = gridspec.GridSpec(25, 15)
    fig = plt.figure(figsize=(25,5))

    ax1 = plt.subplot(gs[:, :-5])
    # 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()
```

In [0]:

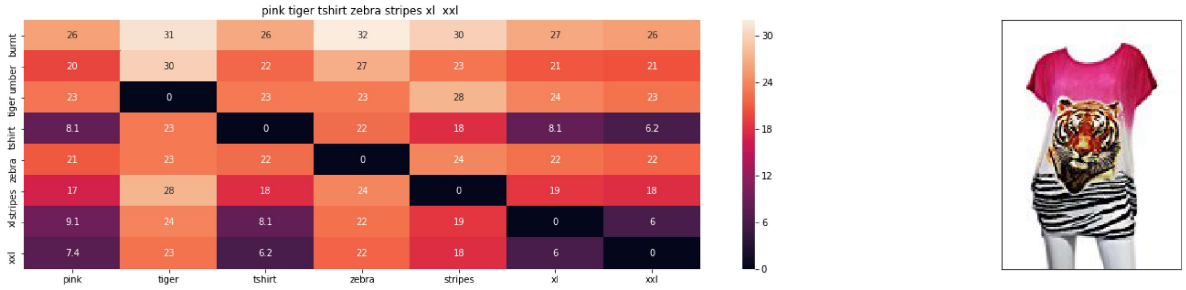
```
def idf_w2v_brand(doc_id, w1, w2, num_results):
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resha
    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
    pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist)/float(w1 + w2)

    indices = np.argsort(pairwise_dist.flatten())[0:num_results]
    pdists  = np.sort(pairwise_dist.flatten())[0:num_results]

    df_indices = list(data.index[indices])


    for i in range(0, len(indices)):
        heat_map_w2v_brand(data['title'].loc[df_indices[0]],data['title'].loc[df_indice
        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, 5, 20)
```



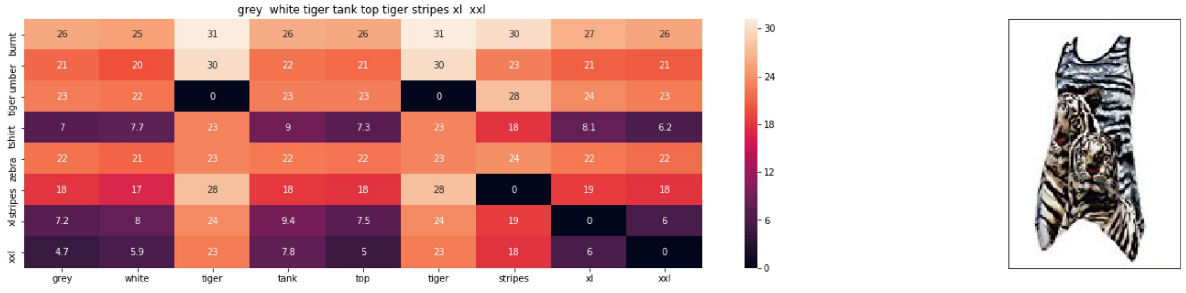
```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.001953125
=============================================================================
=============================================
```



ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 2.38547115326
==========
==========

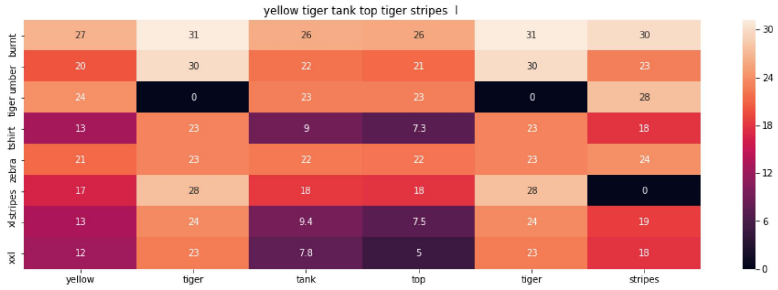


ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 2.73905105609
==========
==========



ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 3.387187195

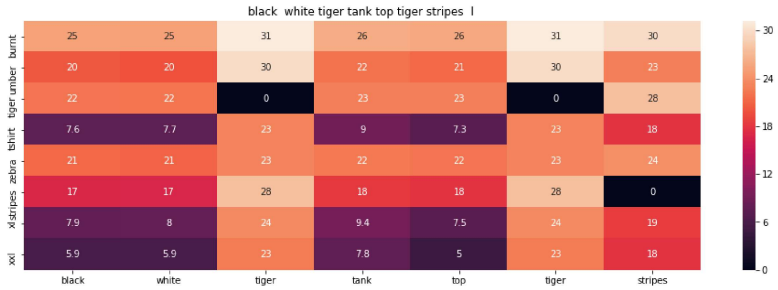
==========



ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 3.5518684389

==========

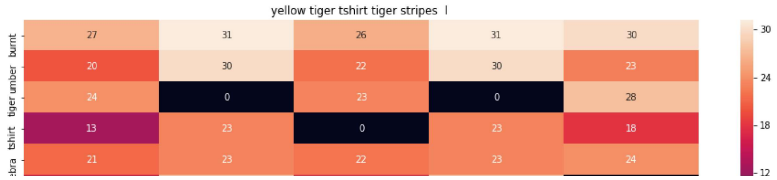
==========



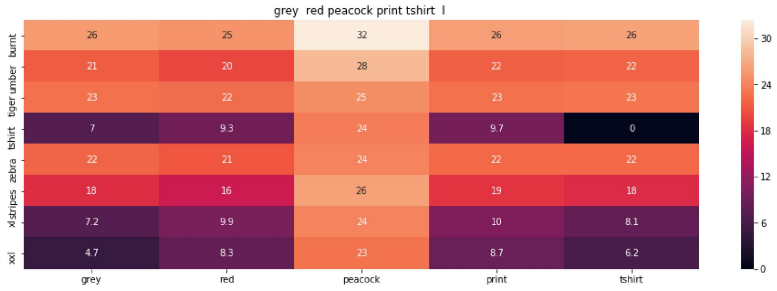
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 3.5536174776

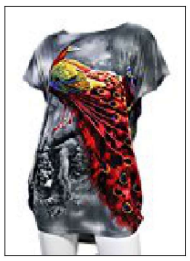
==========

==========

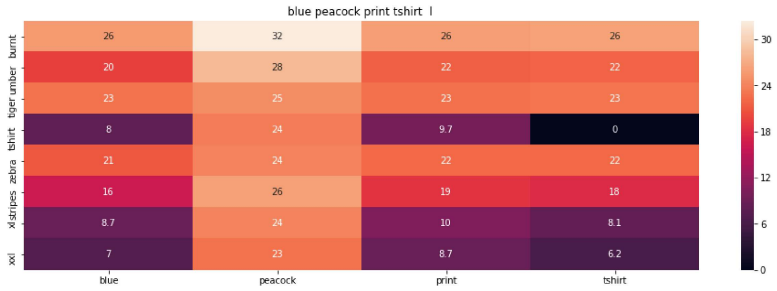


ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 3.65382804889
==========
==========

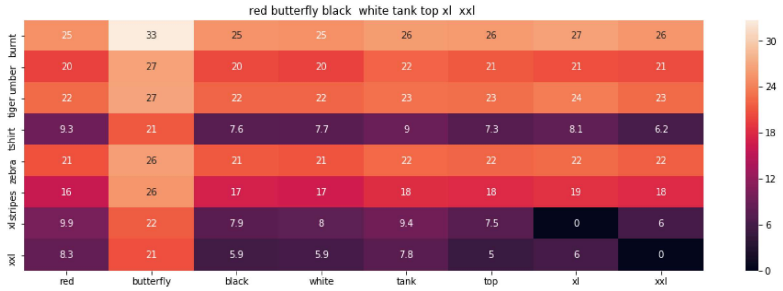




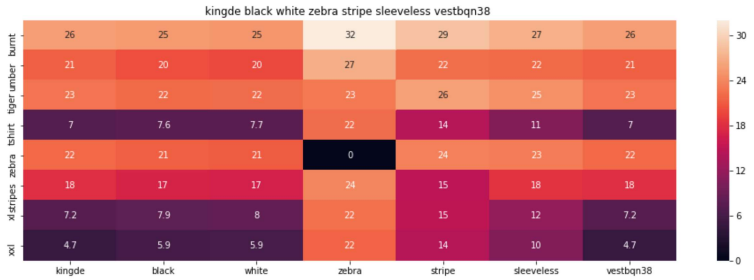
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 4.12881164569
==========
==========



ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 4.20390052813
==========
==========

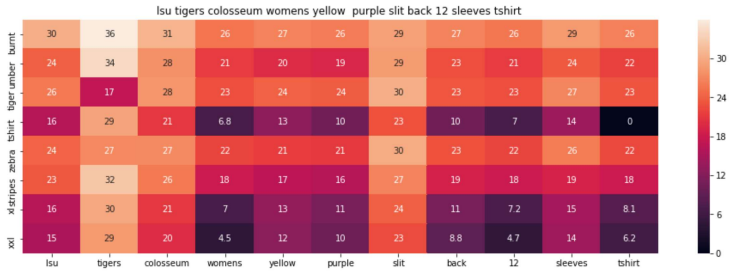


ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 4.28658676166
==========
==========

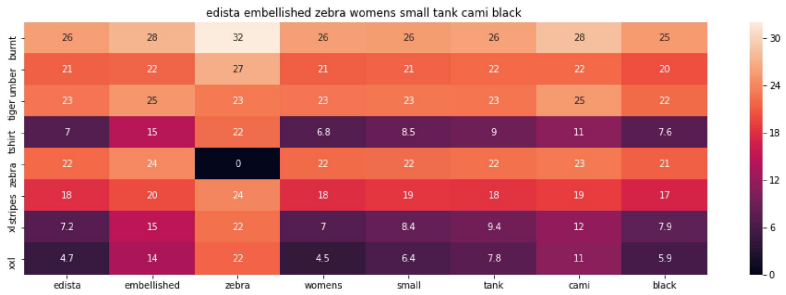


ASIN : B015H41F6G
Brand : KINGDE
euclidean distance from input : 4.38937078798
==========
==========

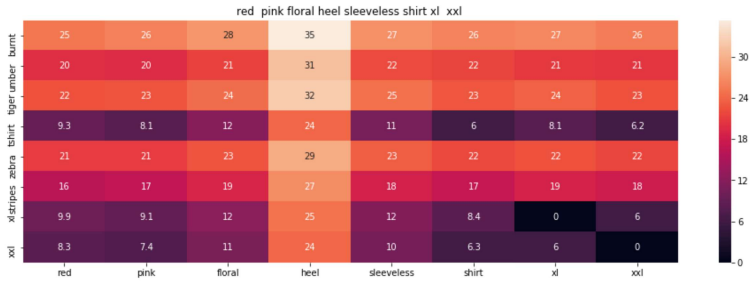
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 4.39790992755
==========
==========



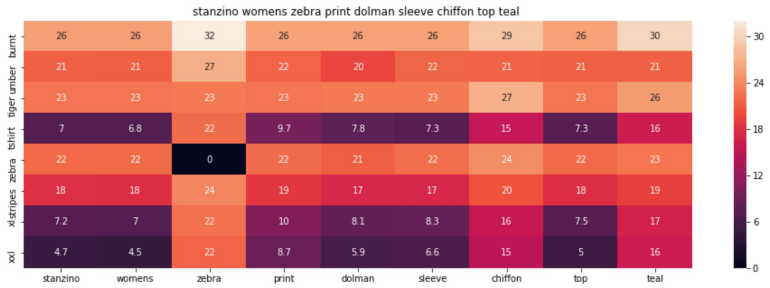
ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 4.45122858369
==========
==========



ASIN : B074P8MD22
Brand : Edista
euclidean distance from input : 4.51897779787
==========
==========

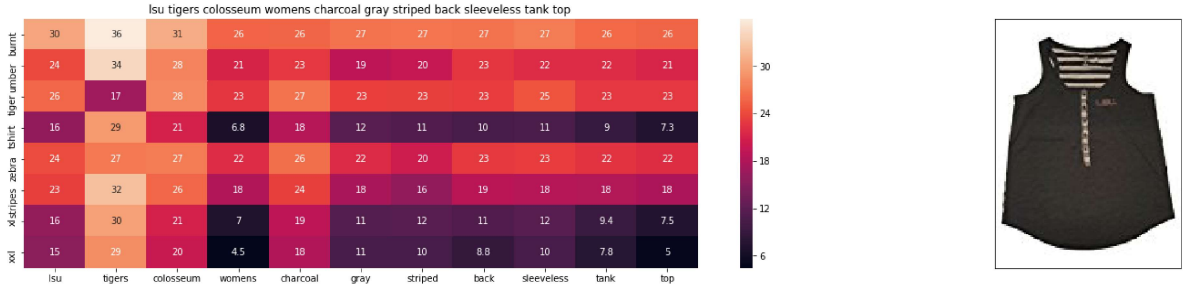


ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 4.52937545794
==========
==========

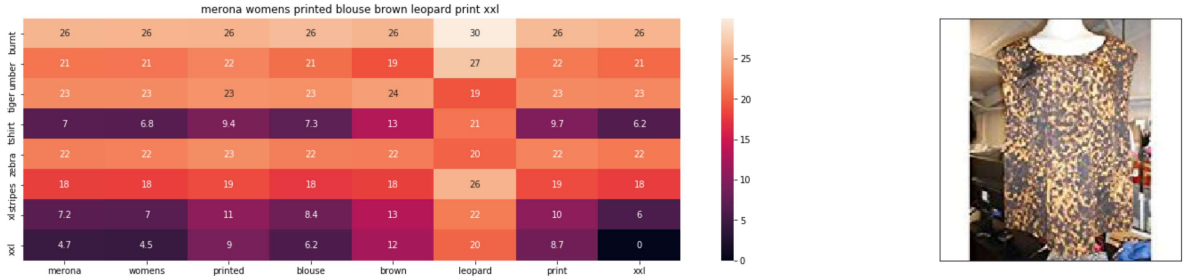


ASIN : B00C0I3U3E
Brand : Stanzino
euclidean distance from input : 4.53032614076
==========
==========

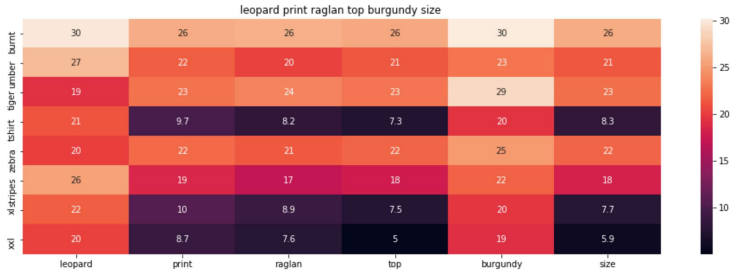
ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 4.54681702403
==========
==========



ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 4.54835554445
==========
==========



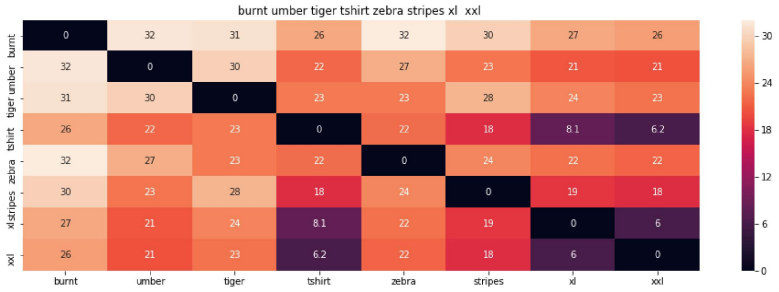
ASIN : B071YF3WDD
Brand : Merona
euclidean distance from input : 4.61062742555
==========
==========



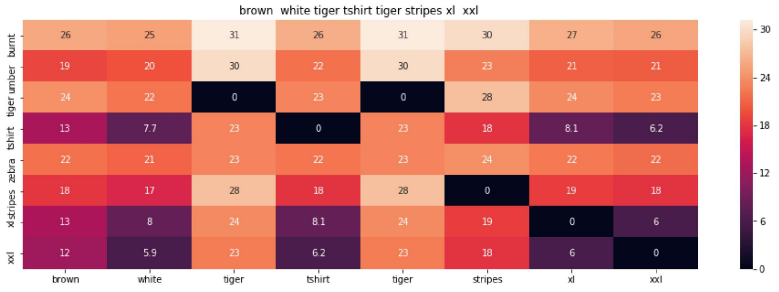
ASIN : B01C6ORLDQ
Brand : 1 Mad Fit
euclidean distance from input : 4.64591789282
==========
==========

In [0]:

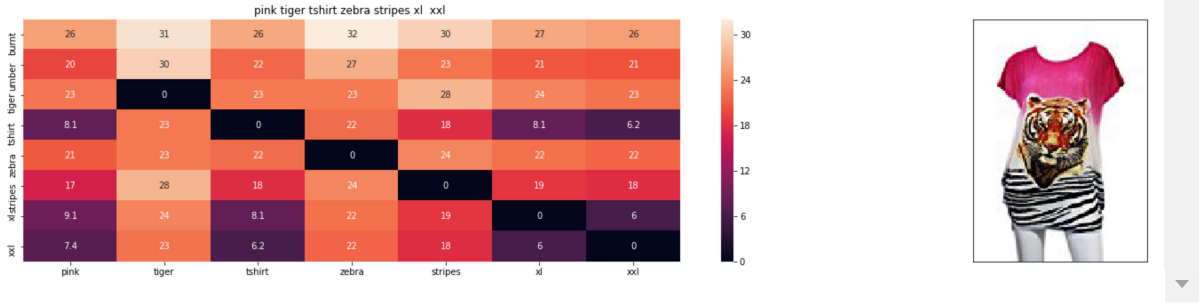
```
1 idf_w2v_brand(12566, 5, 50, 20)
```



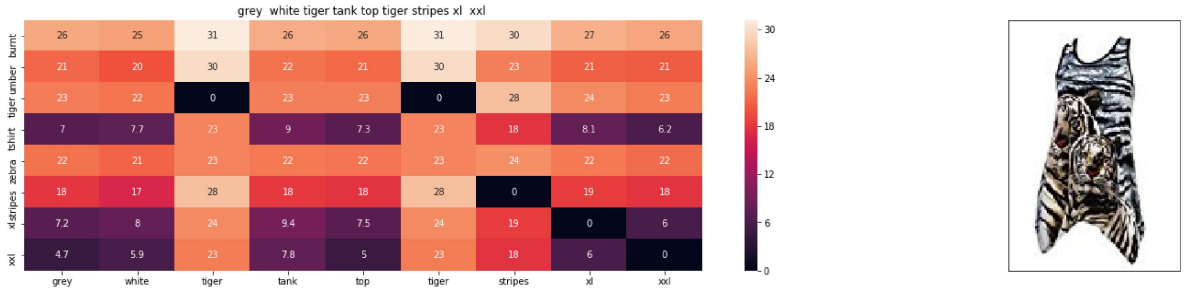
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.000355113636364
==========
==========



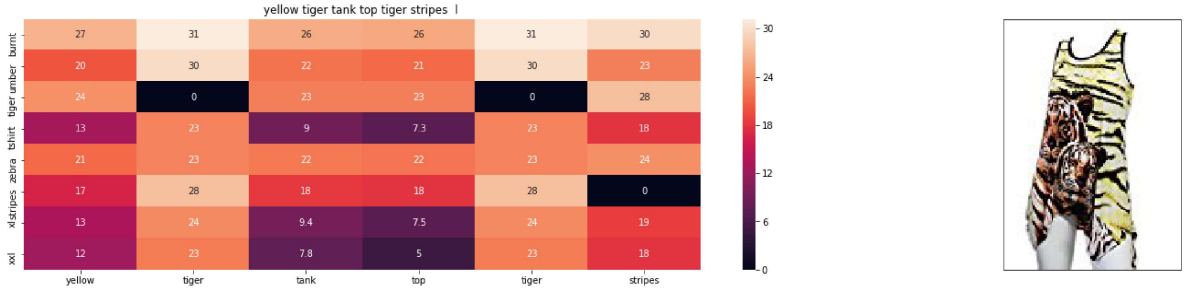
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.433722027865
==========
==========



ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 1.65509310669
==========
==========

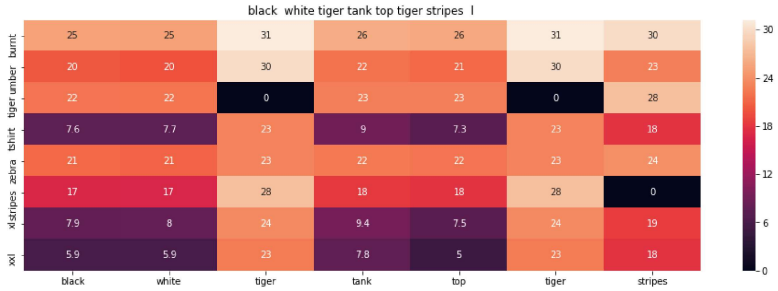


ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 1.77293604103
==========
==========



ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 1.80287808538

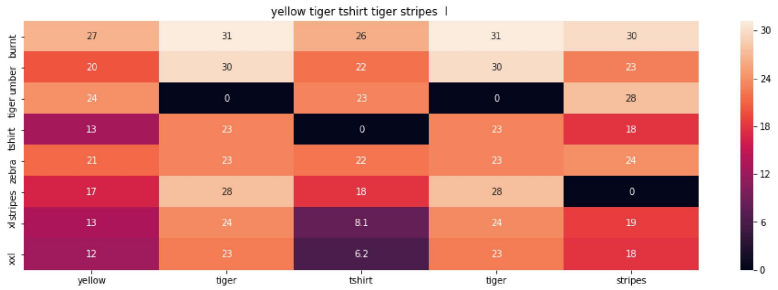
==========



ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 1.80319609241

==========

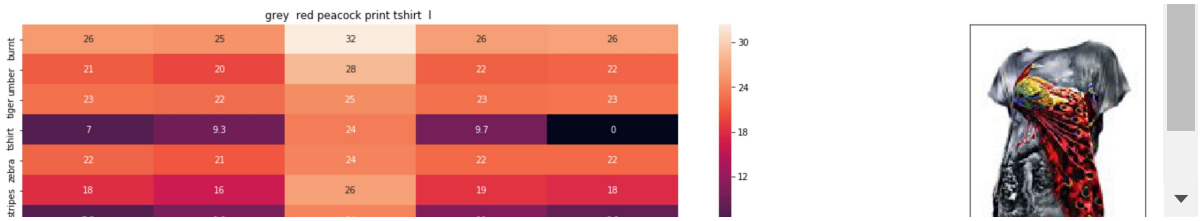
==========



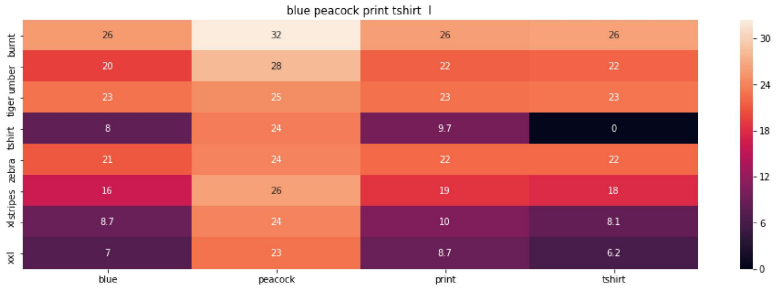
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 1.82141619628

==========

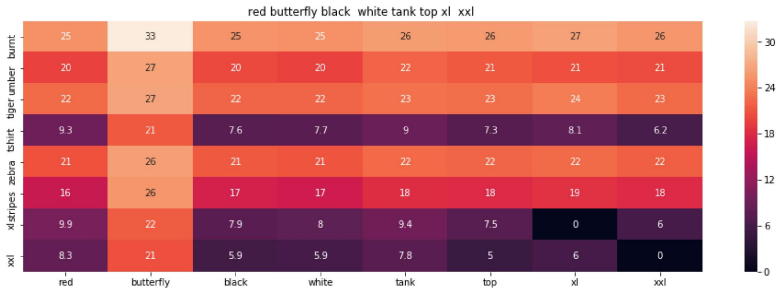
==========



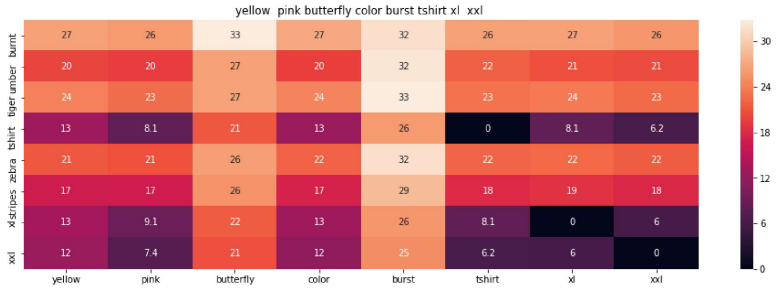
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.90777685025
==========
==========



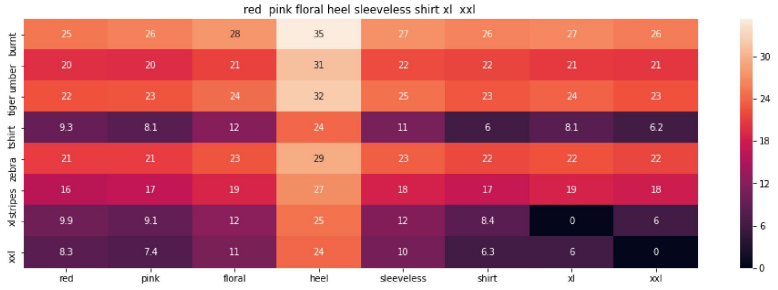
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 1.92142937433
==========
==========



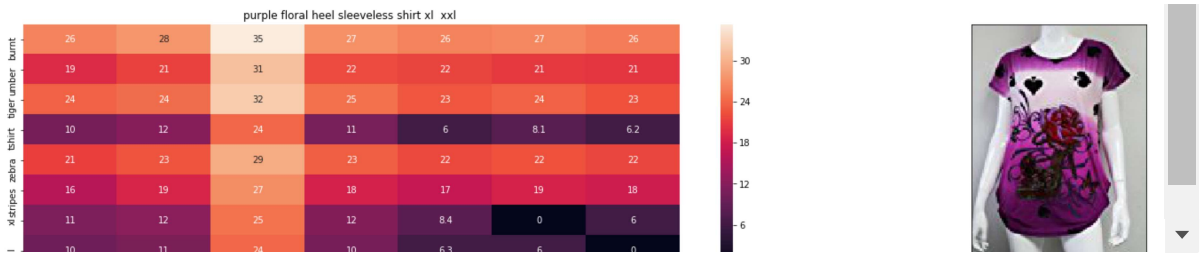
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 1.93646323497
==========
==========



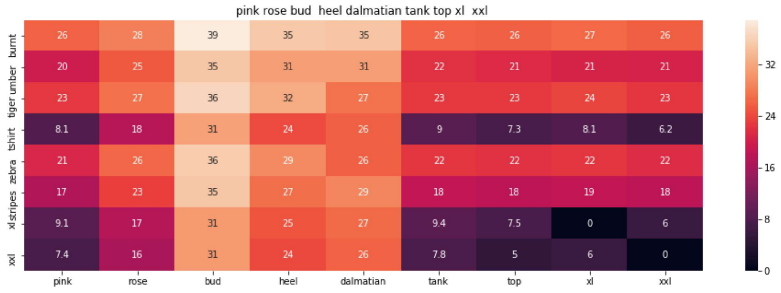
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.95670381059
==========
==========



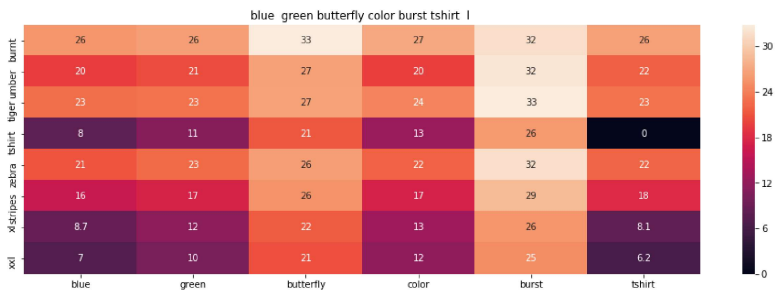
ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 1.9806066343
==========
==========



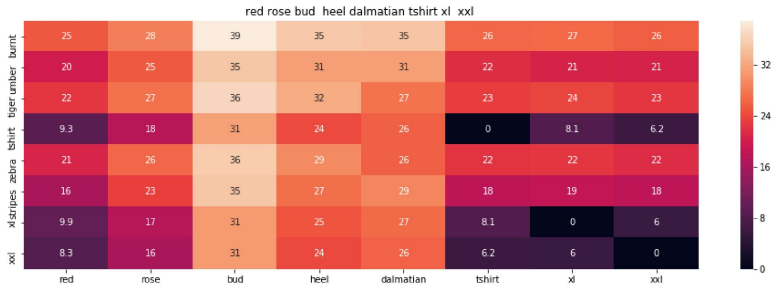
ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 2.01218559992
==========
==========



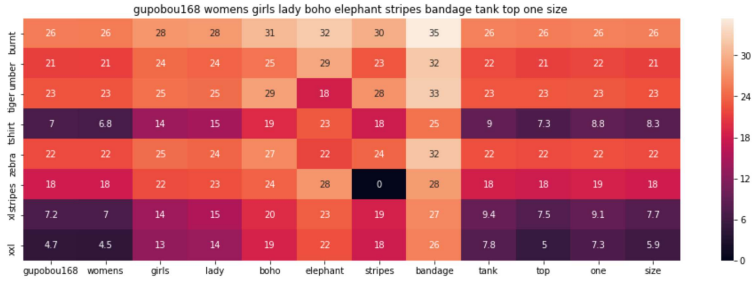
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 2.01335178755
==========
==========



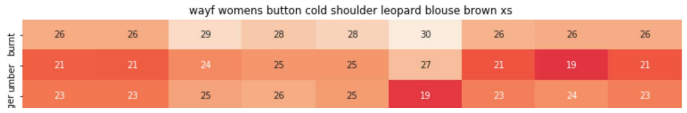
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 2.01388334827
==========
==========



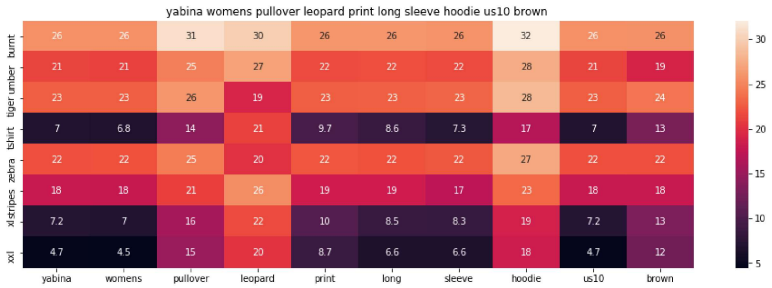
ASIN : B00JXQABB0
Brand : Si Row
euclidean distance from input : 2.0367257555
==========
==========



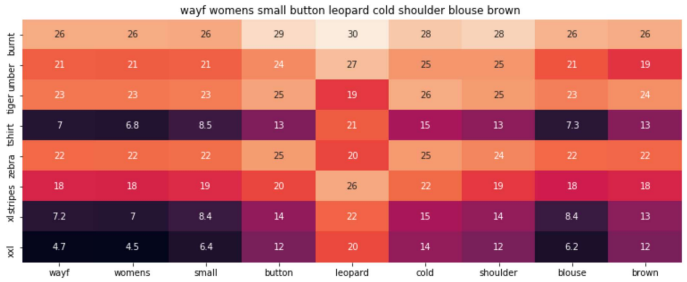
ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 2.65620416778
==========
==========



ASIN : B01LZ7BQ4H
Brand : WAYF
euclidean distance from input : 2.6849067823
==========
==========



ASIN : B01KJUM6JI
Brand : YABINA
euclidean distance from input : 2.68583819266
==========
==========



ASIN : B01M06V4X1
Brand : WAYF
euclidean distance from input : 2.69476194865
==========
==========

Keras and Tensorflow to extract features

In [0]:

```
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
```

```
Using TensorFlow backend.
```

Visual features based product similarity.

In [0]:

```
#load the features and corresponding ASINS info.
bottleneck_features_train = np.load('16k_data_cnn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.npy')
asins = list(asins)

# load the original 16K dataset
data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
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_t

    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: 0.0
Amazon Url: www.amzon.com/dp/B00JXQB5FQ
```

```
Product Title:  pink tiger tshirt zebra stripes xl  xxl
Euclidean Distance from input image: 30.0501
Amazon Url: www.amzon.com/dp/B00JXQASS6
```

Product Title: yellow tiger tshirt tiger stripes l
Euclidean Distance from input image: 41.2611
Amazon Url: www.amzon.com/dp/B00JXQCUIC

Product Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean Distance from input image: 44.0002
Amazon Url: www.amzon.com/dp/B00JXQCWTO

Product Title: kawaii pastel tops tees pink flower design
Euclidean Distance from input image: 47.3825
Amazon Url: www.amzon.com/dp/B071FCWD97

Product Title: womens thin style tops tees pastel watermelon print
Euclidean Distance from input image: 47.7184
Amazon Url: www.amzon.com/dp/B01JUNHBRM

Product Title: kawaii pastel tops tees baby blue flower design
Euclidean Distance from input image: 47.9021
Amazon Url: www.amzon.com/dp/B071SBCY9W

Product Title: edv cheetah run purple multi xl
Euclidean Distance from input image: 48.0465
Amazon Url: www.amzon.com/dp/B01CUPYBM0

Product Title: danskin womens vneck loose performance tee xsmall pink ombre
Euclidean Distance from input image: 48.1019
Amazon Url: www.amzon.com/dp/B01F7PHXY8

Product Title: summer alpaca 3d pastel casual loose tops tee design
Euclidean Distance from input image: 48.1189
Amazon Url: www.amzon.com/dp/B01I80A93G

Product Title: miss chievous juniors striped peplum tank top medium shado wpeach
Euclidean Distance from input image: 48.1313
Amazon Url: www.amzon.com/dp/B0177DM70S

Product Title: red pink floral heel sleeveless shirt xl xxl
Euclidean Distance from input image: 48.1695
Amazon Url: www.amzon.com/dp/B00JV63QQE

Product Title: moana logo adults hot v neck shirt black xxl
Euclidean Distance from input image: 48.2568
Amazon Url: www.amzon.com/dp/B01LX6H43D

Product Title: abaday multicolor cartoon cat print short sleeve longline sh irt large
Euclidean Distance from input image: 48.2657
Amazon Url: www.amzon.com/dp/B01CR57YY0

Product Title: kawaii cotton pastel tops tees peach pink cactus design
Euclidean Distance from input image: 48.3626
Amazon Url: www.amzon.com/dp/B071WYLBZS

Product Title: chicago chicago 18 shirt women pink
Euclidean Distance from input image: 48.3836
Amazon Url: www.amzon.com/dp/B01GXAZTRY

Product Title: yichun womens tiger printed summer tshirts tops
Euclidean Distance from input image: 48.4493
Amazon Url: www.amzon.com/dp/B010NN9RXO

Product Title: nancy lopez whimsy short sleeve whiteblacklemon drop xs
Euclidean Distance from input image: 48.4788
Amazon Url: www.amzon.com/dp/B01MPX6IDX

Product Title: womens tops tees pastel peach ice cream cone print
Euclidean Distance from input image: 48.558
Amazon Url: www.amzon.com/dp/B0734GRKZL

```
Product Title:  uswomens mary j blige without tshirts shirt
Euclidean Distance from input image: 48.6144
Amazon Url: www.amzon.com/dp/B01M0XXFKK
```

In [0]:

```
1
```