

# Amazon Apparel Recommendations

## [4.2] Data and Code:

<https://drive.google.com/open?id=0BwNkduBnePt2VWhCYXhMV3p4dTg>

## [4.3] Overview of the data

In [0]:

```
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

In [0]:

```
data.columns
```

Out[0]:

```
Index(['asin', 'author', 'availability', 'availability_type', 'brand', 'color',
       'editorial_review', 'editorial_review', 'formatted_price',
       'large_image_url', 'manufacturer', 'medium_image_url', 'model',
       'product_type_name', 'publisher', 'reviews', 'sku', 'small_image_url',
       'title'],
      dtype='object')
```

In [0]:

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

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[0]:

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

In [0]:

```
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]:

```
product_type_count = Counter(list(data['product_type_name']))
product_type_count.most_common(10)
```

Out[0]:

```
[('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)]
```

In [0]:

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

```
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[0]:

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

In [0]:

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

4%.

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

In [0]:

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

Out[0]:

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

In [0]:

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

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

In [0]:

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

Out[0]:

```
[ (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 [0]:

```
data.to_pickle('pickels/180k_apparel_data')
```

In [0]:

```
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]:

```
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

In [0]:

```
data.to_pickle('pickels/28k_apparel_data')
```

In [0]:

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

In [0]:

```
data.head()
```

Out[0]:

asin	brand	color	medium_image_url	product_type_name	title	formatted_price
------	-------	-------	------------------	-------------------	-------	-----------------

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

In [0]:

```
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

In [0]:

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

Out[0]:

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

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

    # store the list of words of ith string in a, ex: a = ['tokidoki', 'The', 'Queen', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']
    a = data['title'].loc[indices[i]].split()

    # search for the similar products sequentially
    j = i+1
    while j < num_data_points:
```

```

while j < num_data_points:

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

    # store the maximum length of two strings
    length = max(len(a), len(b))

    # count is used to store the number of words that are matched in both strings
    count = 0

    # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will
    # appended None in case of unequal strings
    # example: a =['a', 'b', 'c', 'd']
    # b = ['a', 'b', 'd']
    # itertools.zip_longest(a,b): will give [('a','a'), ('b','b'), ('c',None), ('d', None)]
    for k in itertools.zip_longest(a,b):
        if (k[0] == k[1]):
            count += 1

    # if the number of words in which both strings differ are > 2 , we are considering it as those two apperals are different
    # if the number of words in which both strings differ are < 2 , we are considering it as those two apperals are same, hence we are ignoring them
    if (length - count) > 2: # number of words in which both sensences differ
        # if both strings are differ by more than 2 words we include the 1st string index
        stage1_dedupe_asins.append(data_sorted['asin'].loc[indices[i]])

    # start searching for similar apperals corresponds 2nd string
    i = j
    break
else:
    j += 1
if previous_i == i:
    break

```

In [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')
```

In [0]:

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

In [0]:

```

# 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', 'Diamonds', 'Women's', 'Shirt', 'X-Large']
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', 'of', 'Diamonds', 'Women's', 'Shirt', 'X-Large']

    length = max(len(a), len(b))

    # count is used to store the number of words that are matched in both strings
    count = 0

    # itertools.zip_longest(a,b): will map the corresponding words in both strings, it will
    # appened None in case of unequal strings
    # example: a = ['a', 'b', 'c', 'd']
    # b = ['a', 'b', 'd']
    # itertools.zip_longest(a,b): will give [('a', 'a'), ('b', 'b'), ('c', 'd'), ('d', None)]
    for k in itertools.zip_longest(a,b):
        if (k[0]==k[1]):
            count += 1

    # if the number of words in which both strings differ are < 3 , we are considering it as t
    # hose two apperals are same, hence we are ignoring them
    if (length - count) < 3:
        indices.remove(j)

```

In [0]:

```

# 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)]

```

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

In [0]:

```
data.to_pickle('pickels/16k_apperal_data')
```

In [0]:

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

In [0]:

```

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

```

list of stop words: {'such', 'and', 'hers', 'up', 'she', 'd', 'further', 'all', 'than', 'under', 'is', 'off', 'both', 'most', 'few', 'should', 're', 'very', 'just', 'then', 'didn', 'myself', 'in', 'too', 's', 'shouldn', 'herself', 'because', 'how', 'itself', 'what', 'shan', 'weren', 'doing', 'hem', 'couldn', 'their', 'so', 'ain', 'haven', 'yourself', 'now', 'll', 'isn', 'about', 'over', 'nto', 'before', 'during', 'on', 'as', 'aren', 'against', 'above', 'down', 'they', 'below', 'me', 'ain', 'for', 'but', 'than', 'you', 'yourself', 'man', 'that', 'can', 'am', 'was'}

again', 'for', 'why', 'been', 'yourselves', 'more', 'ther', 'that', 'can', 'an', 'was', 'themselves', 'mightn', 'does', 'those', 'only', 'hasn', 'any', 'ma', 'are', 'nor', 'out', 'you', 'ourselves', 'the', 'an', 'has', 'where', 'i', 'while', 'ours', 'its', 'your', 'had', 'were', 'bein', 'no', 'or', 'needn', 've', 'y', 'a', 'each', 'have', 'through', 'when', 'mustn', 'by', 'won', 'from', 'own', 'will', 'there', 't', 'him', 'these', 'doesn', 'theirs', 'my', 'did', 'of', 'who', 'until', 'wouldn', 'we', 'do', 'having', 'yours', 'other', 'wasn', 'it', 'with', 'once', 'here', 'don', 'o', 'whom', 'this', 'if', 'but', 'hadn', 'our', 'some', 'm', 'not', 'between', 'himself', 'same', 'at', 'be', 'he', 'after', 'which', 'to', '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]:

```
data.head()
```

Out[0]:

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

In [0]:

```
data.to_pickle('pickels/16k_apperial_data_preprocessed')
```

In [0]:

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

Out[0]:

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

asin	brand	color	image url	product_type_name	title	formatted_price
B01NACPBG2	Fifth Degree	Black	https://images-na.ssl-images-amazon.com/images/I/61	SHIRT	gold foil graphic tees Jun...	\$6.95

In [0]:

```
# 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)
```

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_[i]) * model[i])
            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 = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg) of length 300 corresponds to each word in give title
    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) of length 300 corresponds to each word in give title
    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 of appare
```

```

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

ax = plt.subplot(gs[0])
# plotting 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 = 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()

```

In [0]:

```

# 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 sentance
def build_avg_vec(sentence, num_features, doc_id, m_name):
    # sentence: 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 festureVec
nwords = 0

for word in sentence.split():
    nwords += 1
    if word in vocab:
        if m_name == 'weighted' and word in idf_title_vectorizer.vocabulary_:
            featureVec = np.add(featureVec, idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[word]] * model[word])
        elif m_name == 'avg':
            featureVec = np.add(featureVec, model[word])
    if(nwords>0):
        featureVec = np.divide(featureVec, nwords)
# returns the avg vector of given sentance, its of shape (1, 300)
return featureVec

```

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):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance from given input apparel to all remaining apparels
    # the metric we used here is cosine, the coside distance is mesured as K(X, Y) = <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(1,-1))

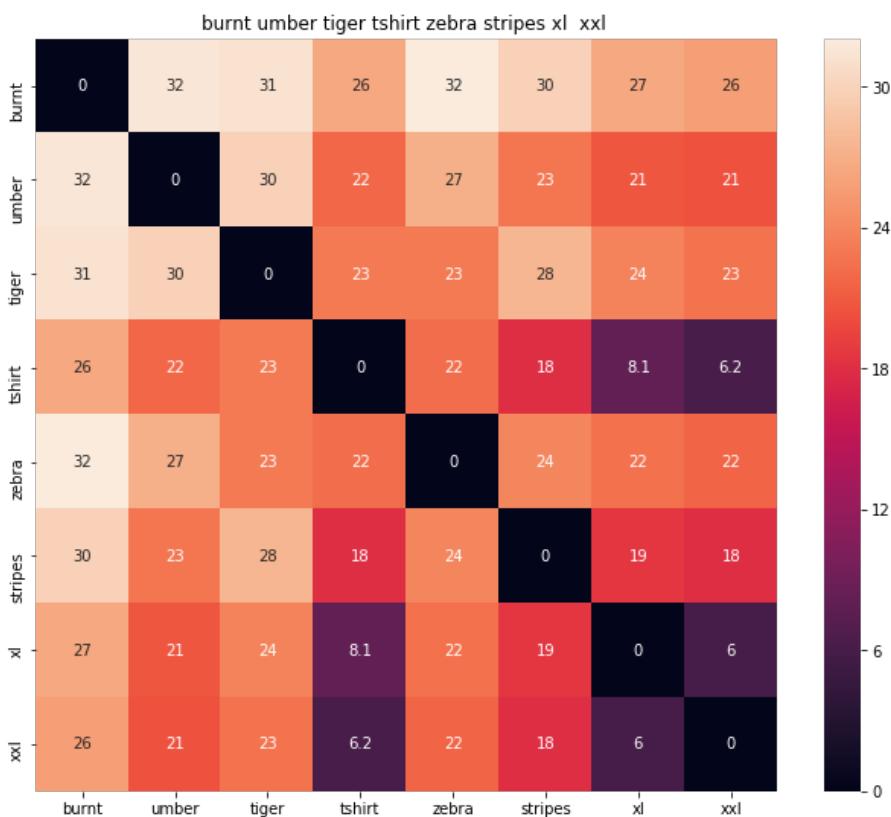
# np.argsort will return indices of 9 smallest distances
indices = np.argsort(pairwise_dist.flatten())[0:num_results]
#pdists will store the 9 smallest distances
pdists = np.sort(pairwise_dist.flatten())[0:num_results]

#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]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], 'weighted')
    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

```



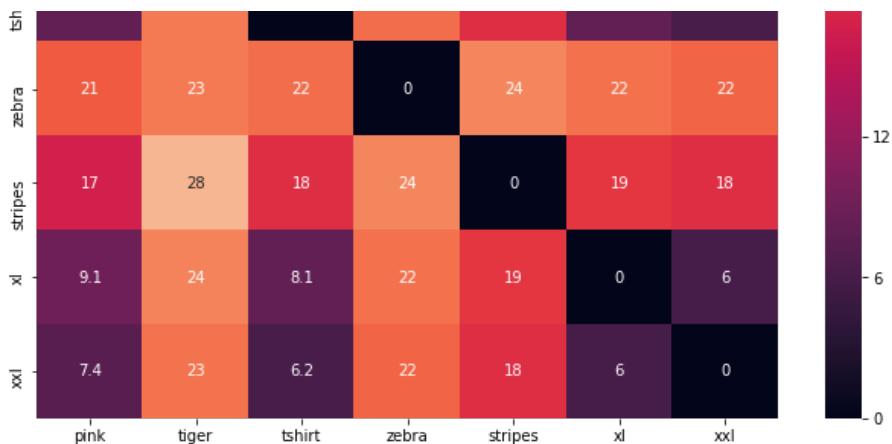
ASIN : B00JXQB5FQ  
 Brand : Si Row  
 euclidean distance from input : 0.00390625

---



---





ASIN : B00JXQASS6

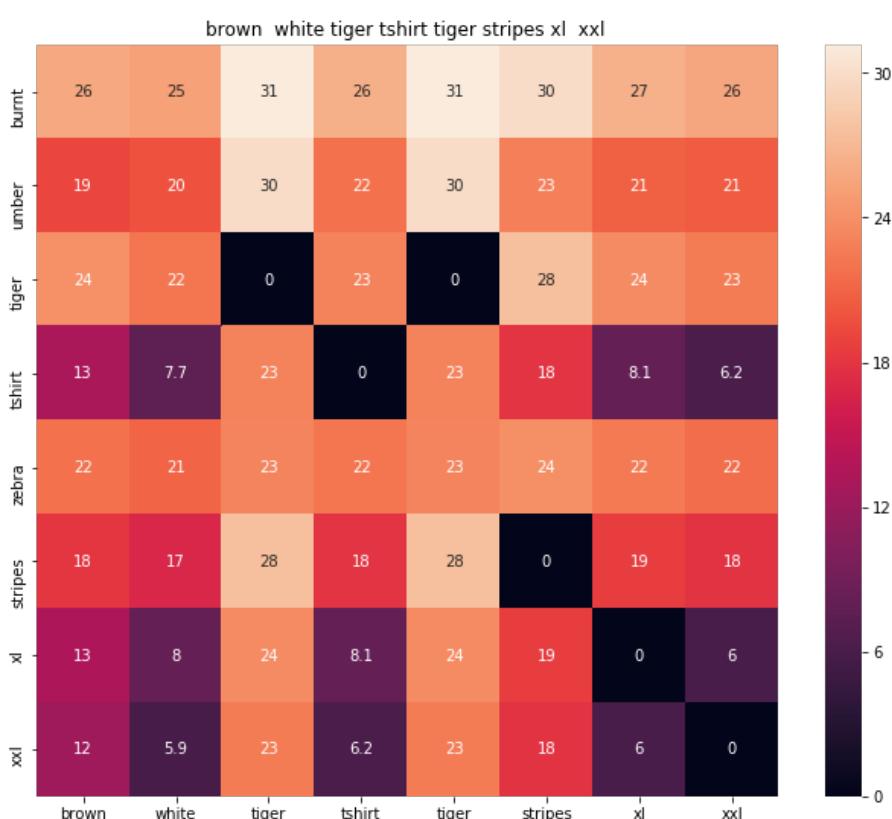
Brand : Si Row

euclidean distance from input : 4.06389

---



---



ASIN : B00JXQCWT0

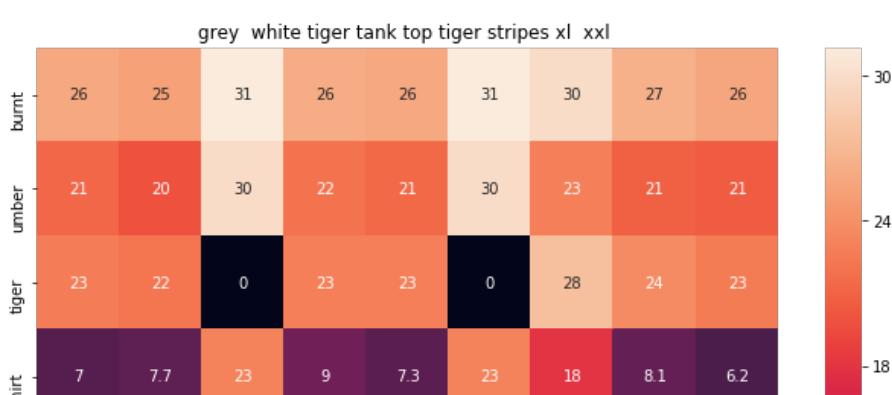
Brand : Si Row

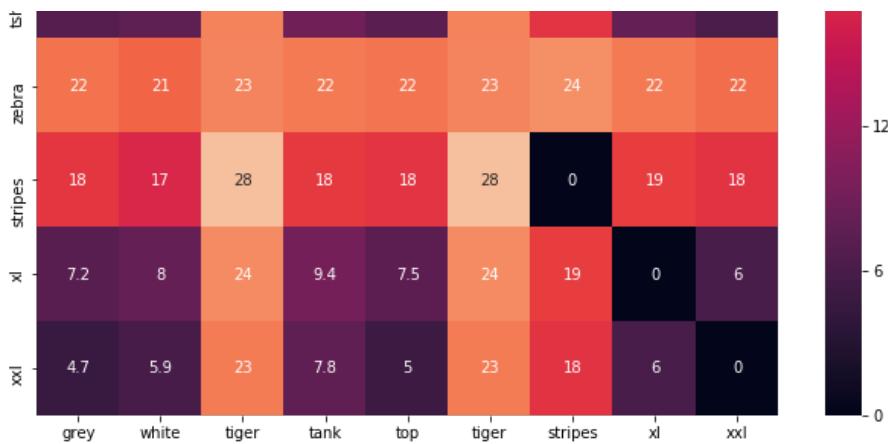
euclidean distance from input : 4.77094

---



---





ASIN : B00JXQAFZ2

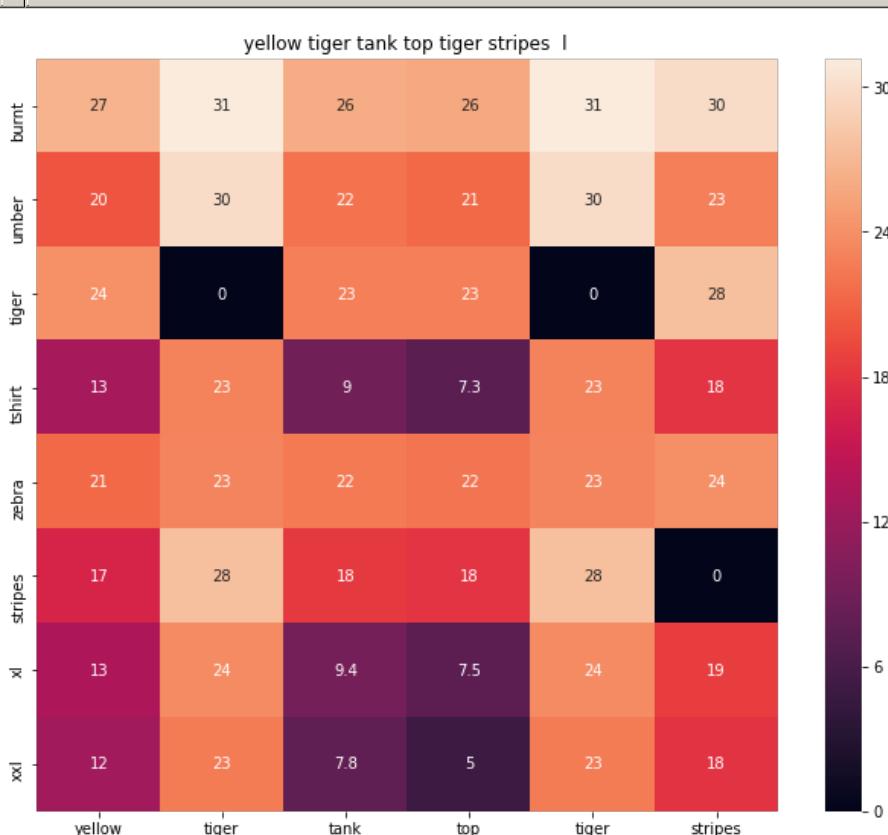
Brand : Si Row

euclidean distance from input : 5.36016

---



---



ASIN : B00JXQAUWA

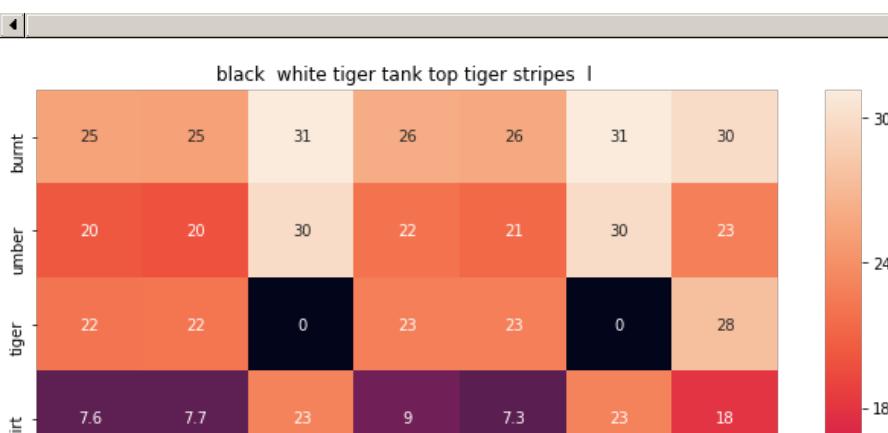
Brand : Si Row

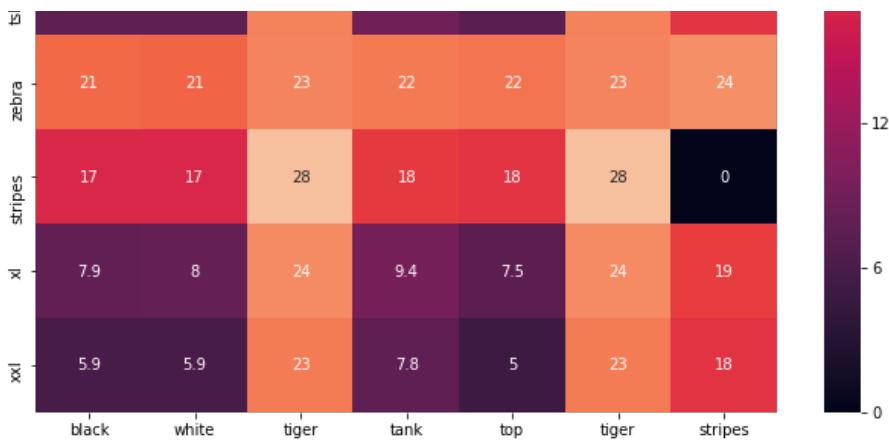
euclidean distance from input : 5.68952

---



---





ASIN : B00JXQAO94

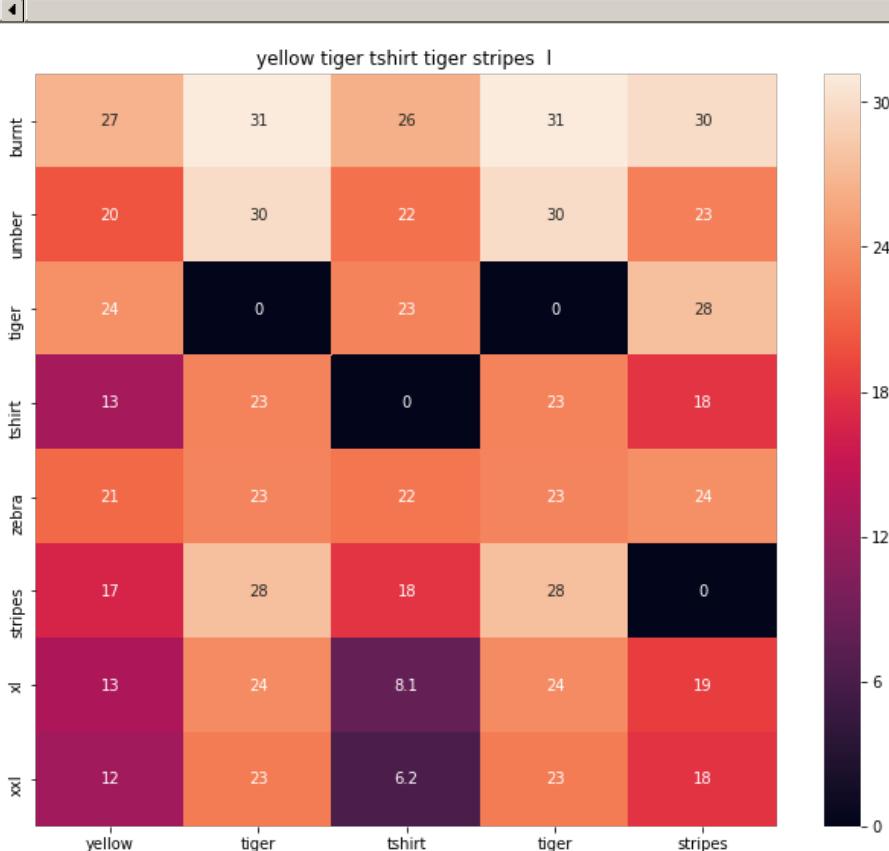
Brand : Si Row

euclidean distance from input : 5.69302

---



---



ASIN : B00JXQCUIC

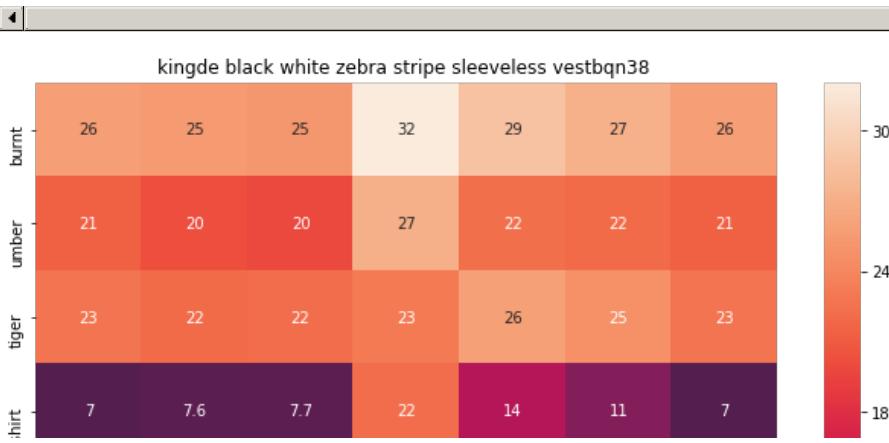
Brand : Si Row

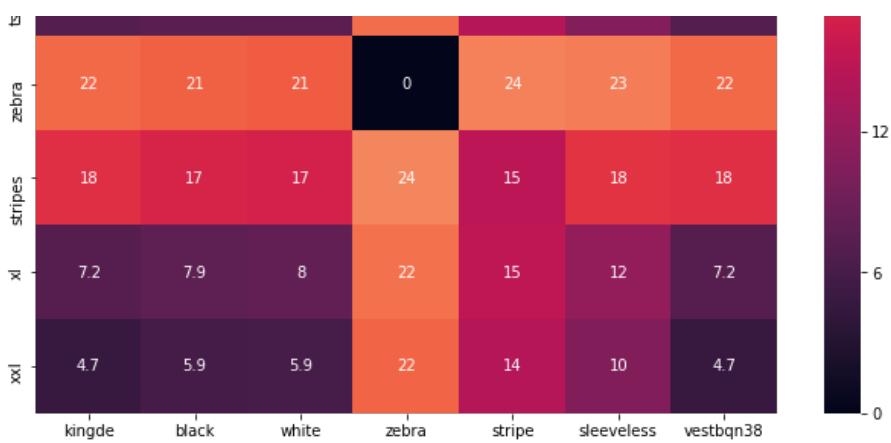
euclidean distance from input : 5.89344

---



---





ASIN : B015H41F6G

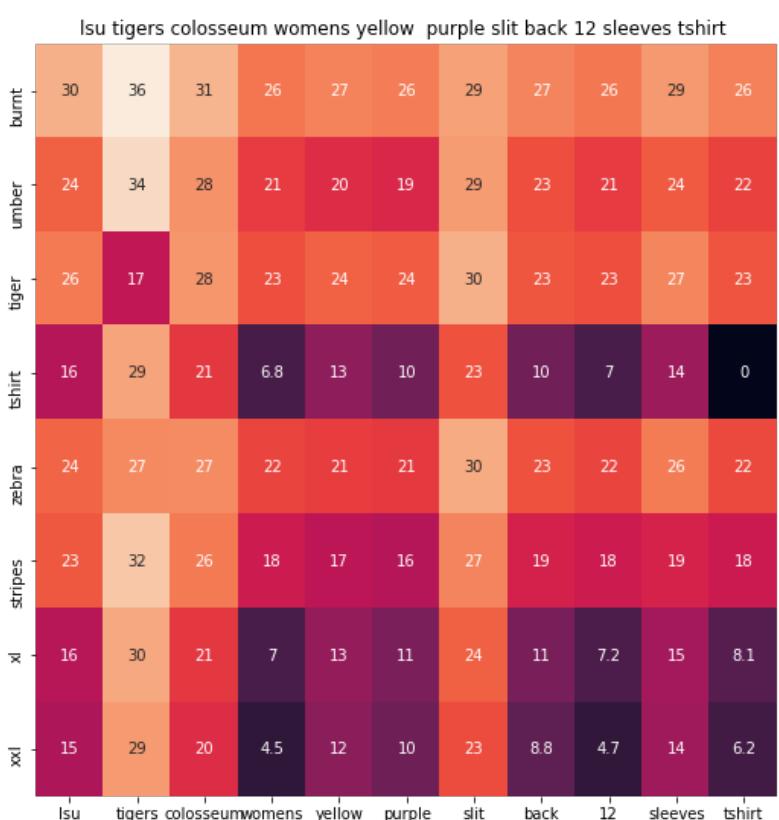
Brand : KINGDE

euclidean distance from input : 6.13299

---



---



ASIN : B073R5Q8HD

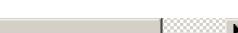
Brand : Colosseum

euclidean distance from input : 6.25671

---



---





ASIN : B074P8MD22

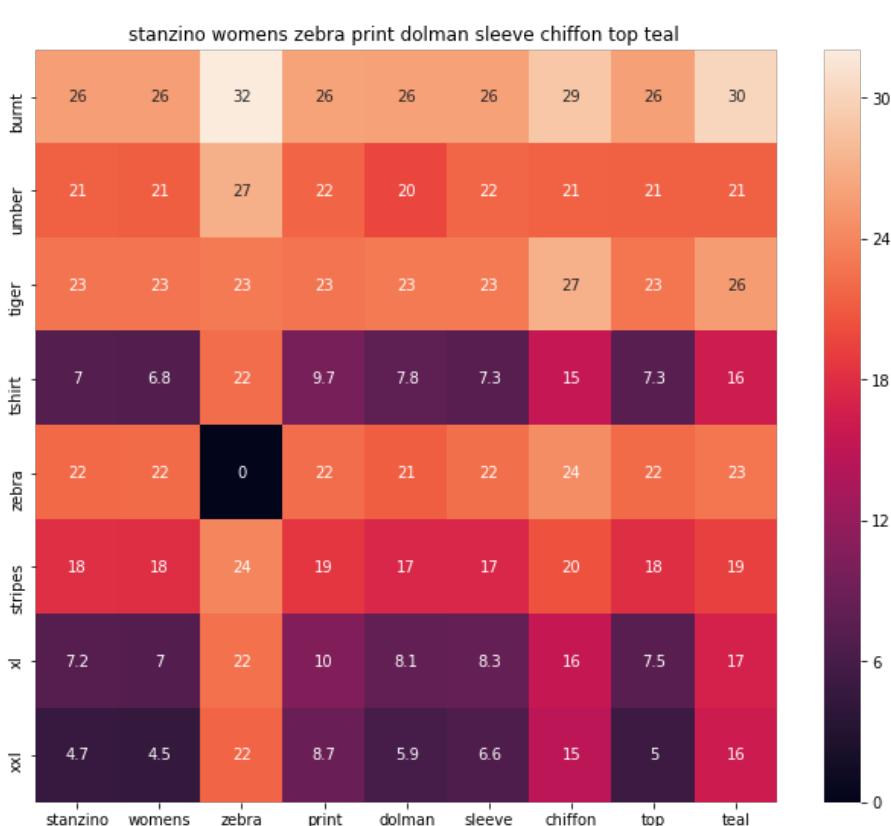
Brand : Edista

euclidean distance from input : 6.3922

---



---



ASIN : B00C0I3U3E

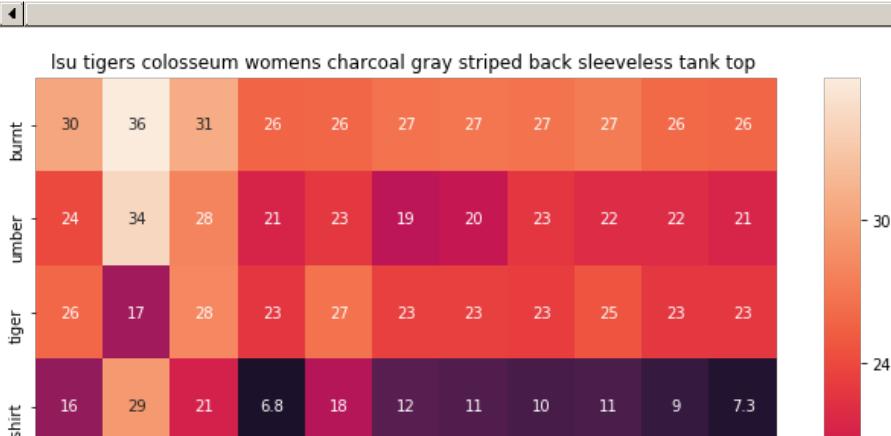
Brand : Stanzino

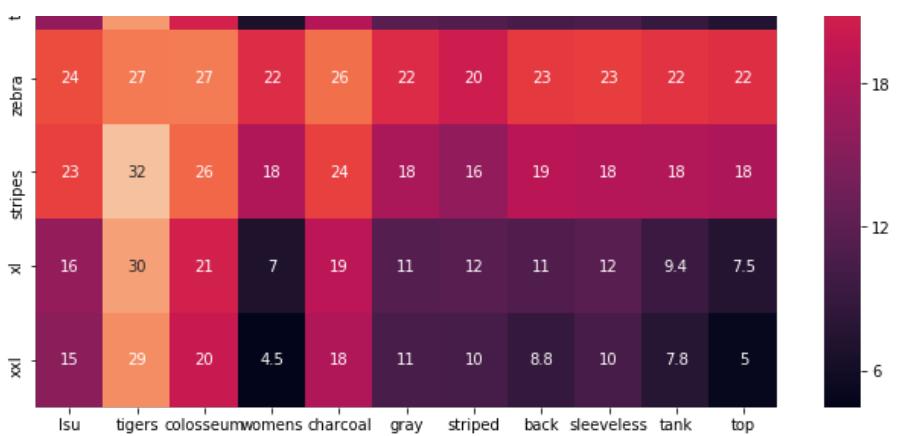
euclidean distance from input : 6.4149

---



---





ASIN : B073R4ZM7Y

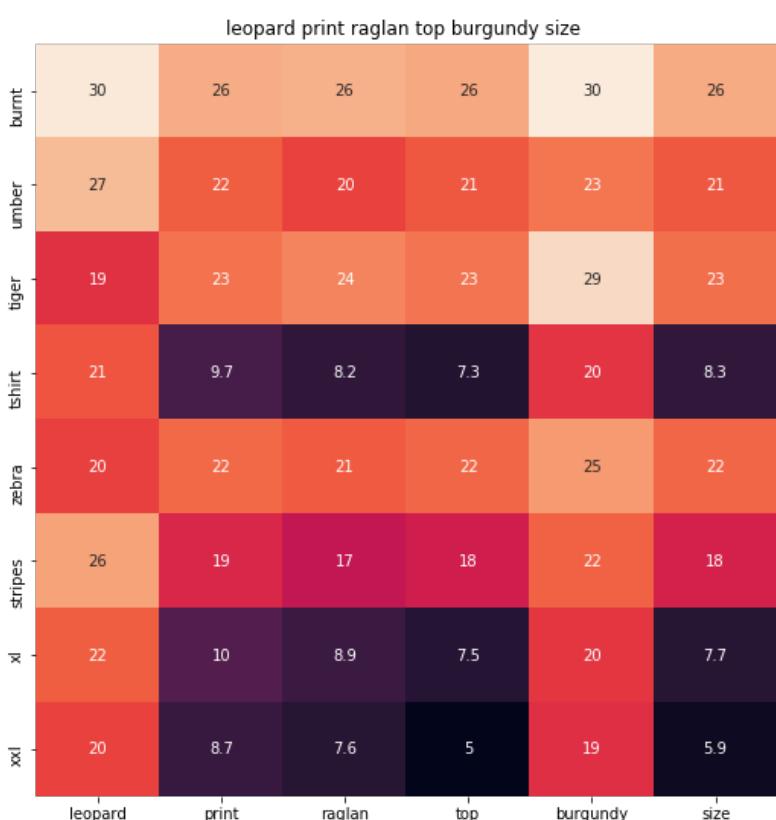
Brand : Colosseum

euclidean distance from input : 6.45096

---



---



ASIN : B01C60RLDQ

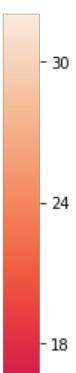
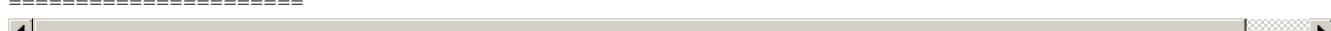
Brand : 1 Mad Fit

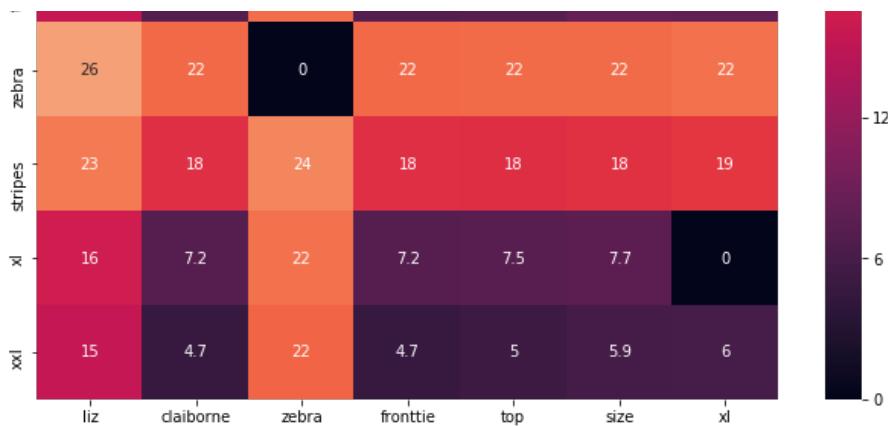
euclidean distance from input : 6.46341

---



---





ASIN : B06XBY5QXL

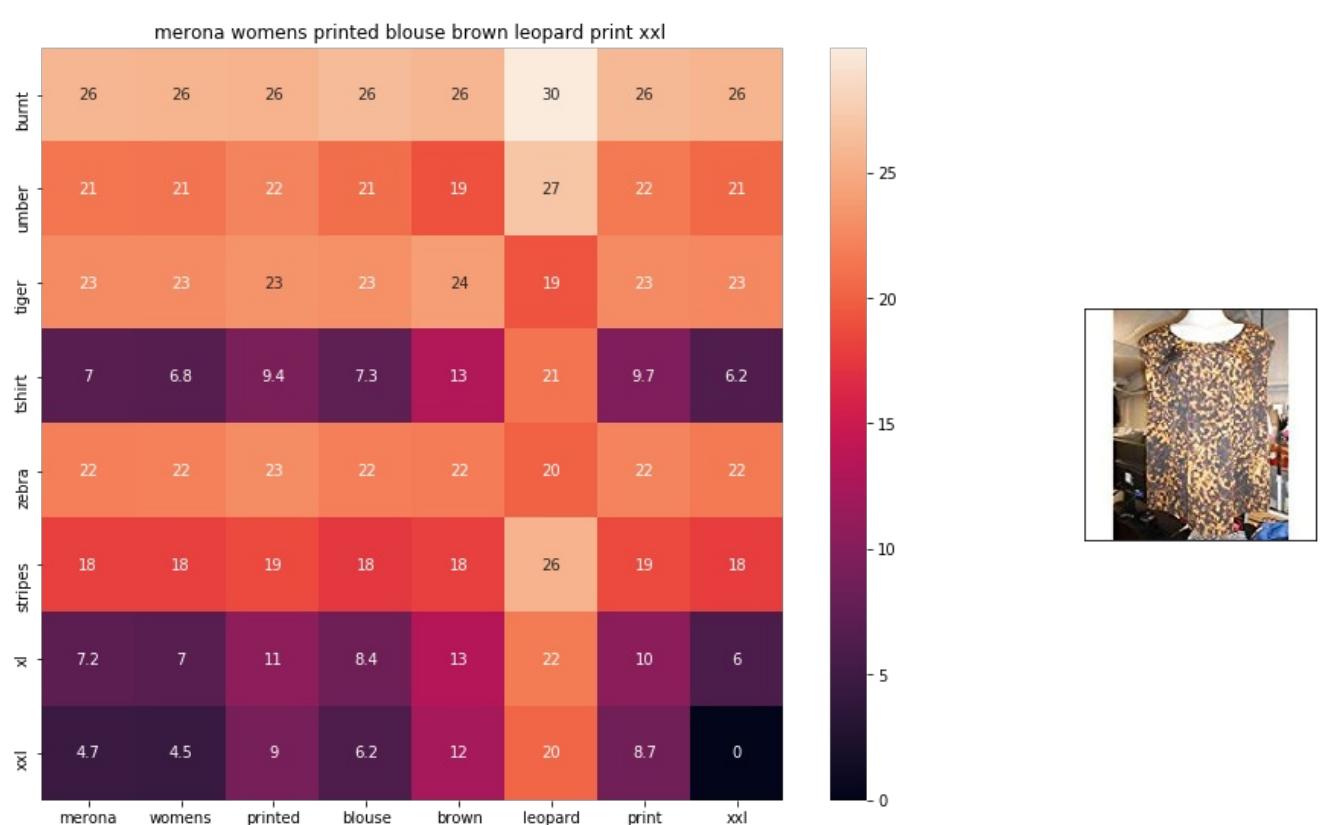
Brand : Liz Claiborne

euclidean distance from input : 6.53922

---



---



ASIN : B071YF3WDD

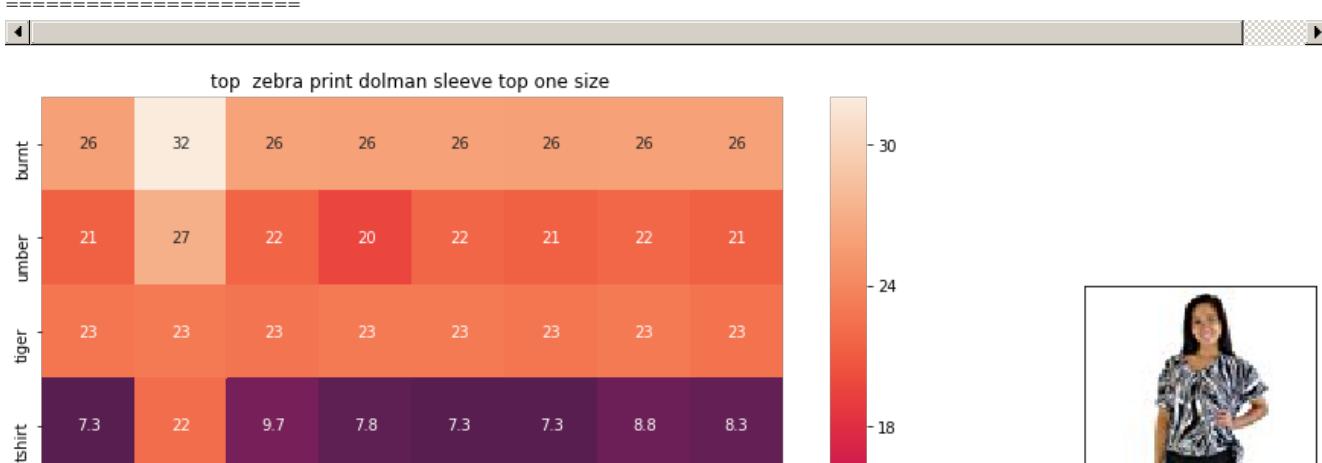
Brand : Merona

euclidean distance from input : 6.5755

---



---





ASIN : B00H8A6ZLI

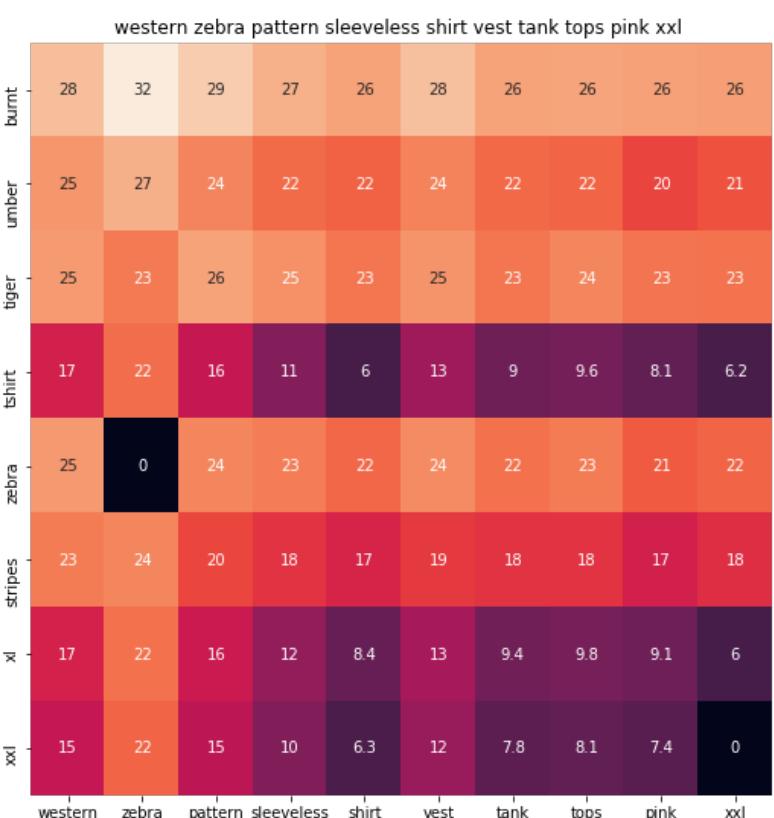
Brand : Vivian's Fashions

euclidean distance from input : 6.63821

---



---



ASIN : B00Z6HEXWI

Brand : Black Temptation

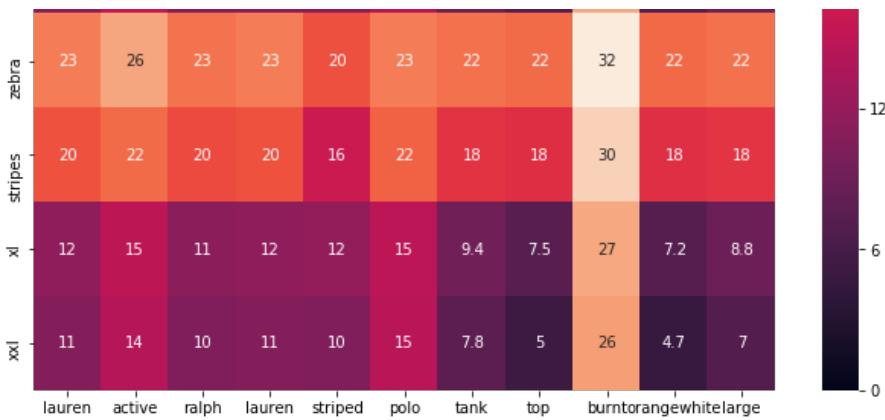
euclidean distance from input : 6.66074

---



---





ASIN : B00ILGH5OY

Brand : Ralph Lauren Active

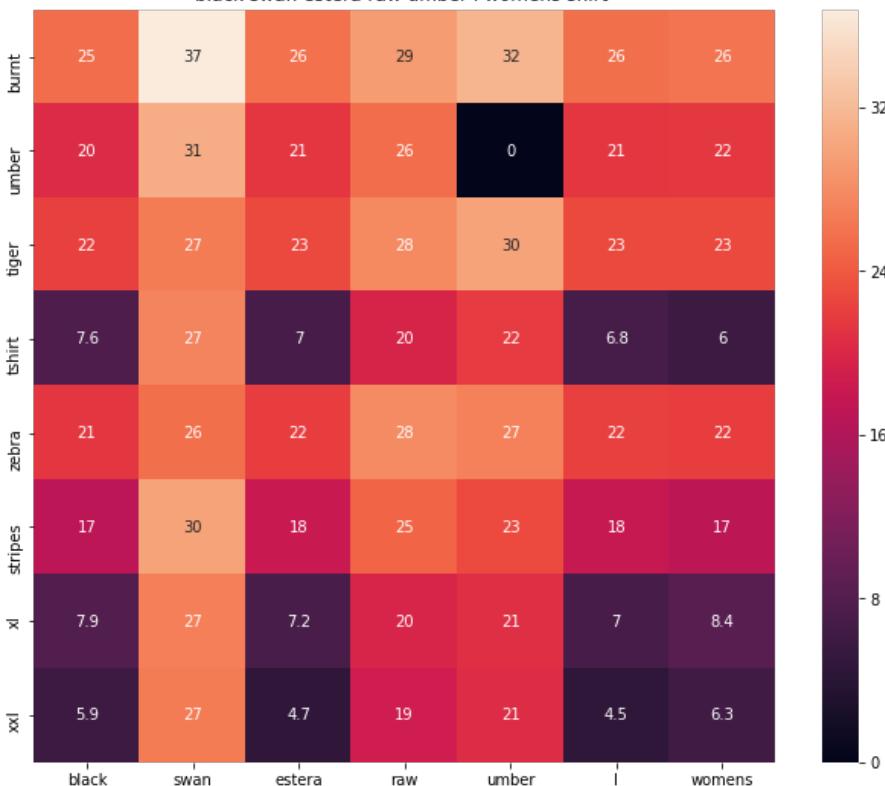
euclidean distance from input : 6.68391

---



---

**black swan estera raw umber l womens shirt**



ASIN : B06Y1VN8WQ

Brand : Black Swan

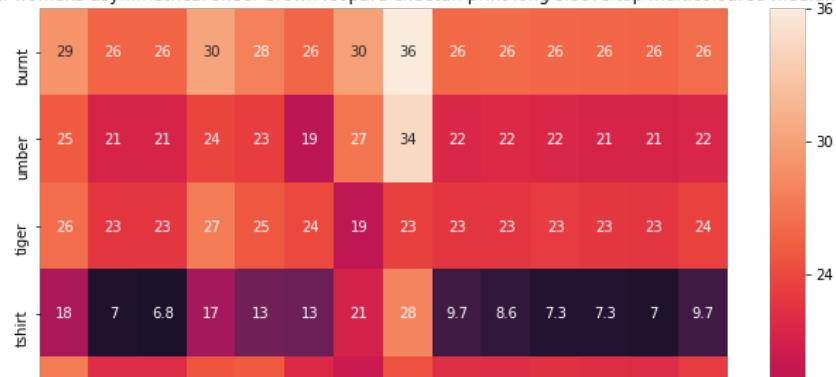
euclidean distance from input : 6.70576

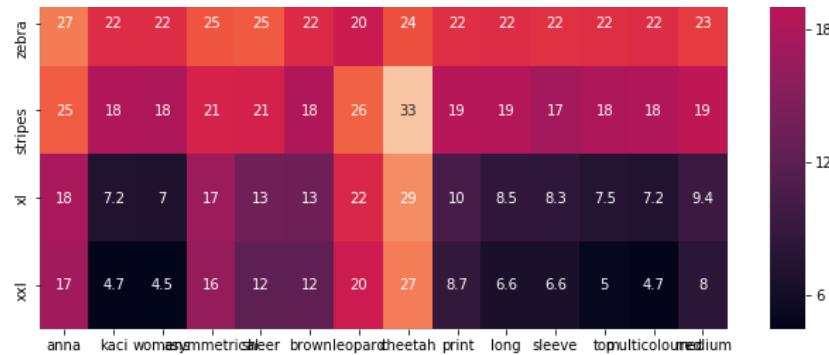
---



---

**anna kaci womens asymmetrical sheer brown leopard cheetah print long sleeve top multicoloured medium**





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

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

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

# 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 hyphen
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, model):
    # sentance1 : title1, input apparel
    # sentance2 : title2, output apparel
```

```

# sentance1 : title1, 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) of length 300 corresponds to each word in give title
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) of length 300 corresponds to each word in give title
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]], # input apparel's features
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], types[doc_id2]]]] # recommended apparel's features

colorscale = [[0, '#1d004d'], [.5, '#f2e5ff'], [1, '#f2e5d1']] # to color the headings of each column

# 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])
# plotting 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 lens 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,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 apparels
    # the metric we used here is cosine, the coside distance is mesured as  $K(X, Y) = \langle X, Y \rangle / (\|X\| * \|Y\|)$ 
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    idf_w2v_dist = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
    dist_img = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,-1))
    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
    pairwise_dist = (w1 * idf_w2v_dist + w2 * ex_feat_dist+ w3* dist_img)/float(w1 + w2+ w3)

```

```

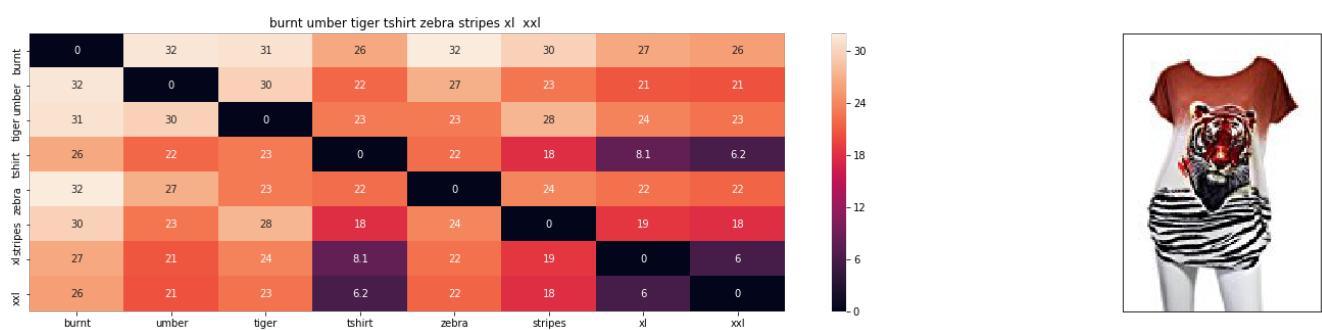
# np.argsort will return indices of 9 smallest distances
indices = np.argsort(pairwise_dist.flatten())[0:num_results]
#pdists will store the 9 smallest distances
pdists = np.sort(pairwise_dist.flatten())[0:num_results]

#data frame indices of the 9 smallest distance'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]], data['medium_image_url'].loc[df_indices[i]], indices[0], indices[i], df_indices[0], df_indices[i], 'weighted')
    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)

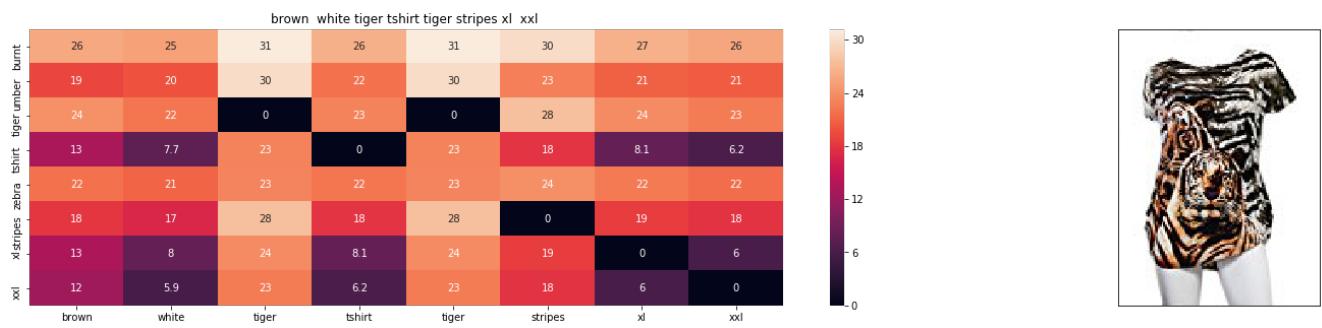
idf_w2v_brand(12566, 5, 5, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j

```



```

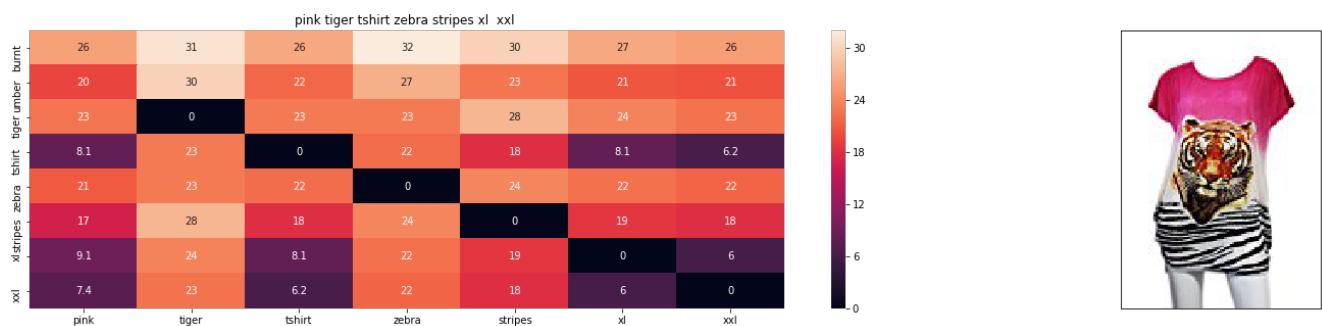
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.001953125
=====
=====
```



```

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

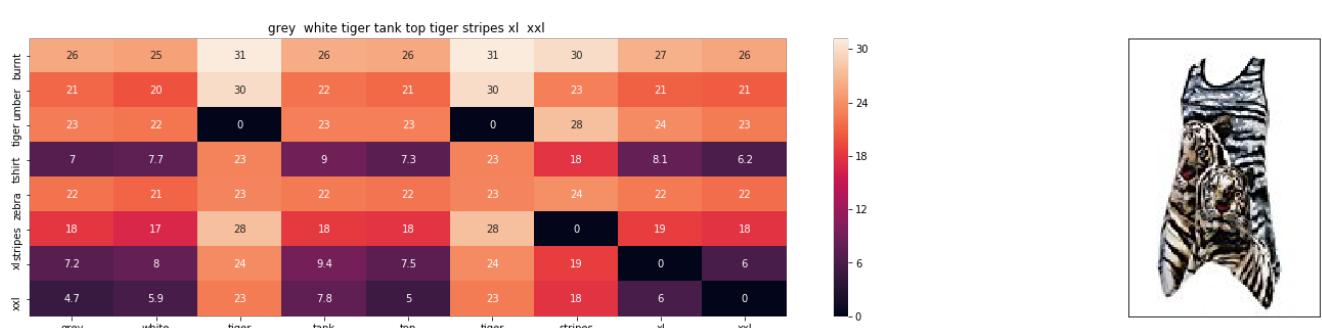
```



ASIN : B00JXQASS6

Brand : Si Row

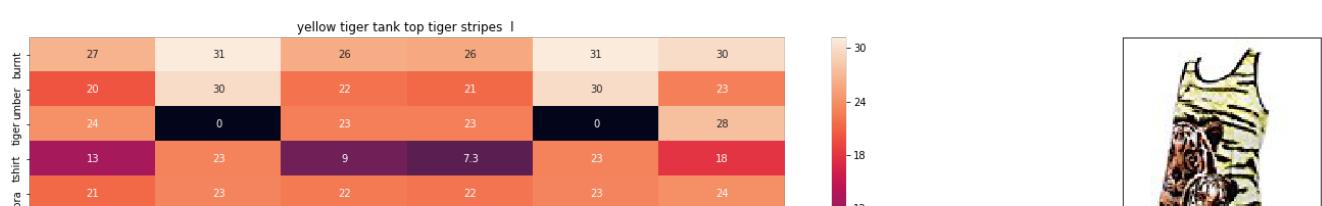
euclidean distance from input : 2.73905105609

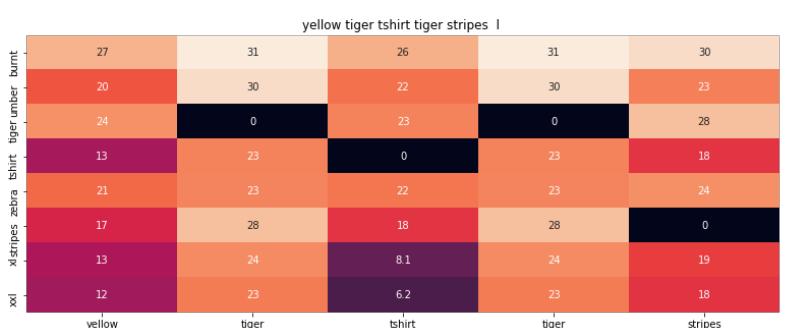
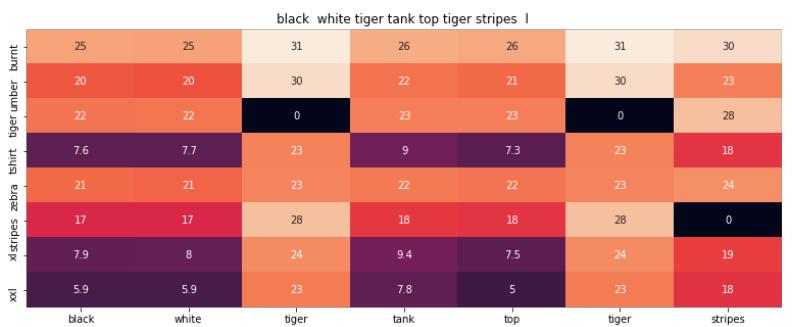


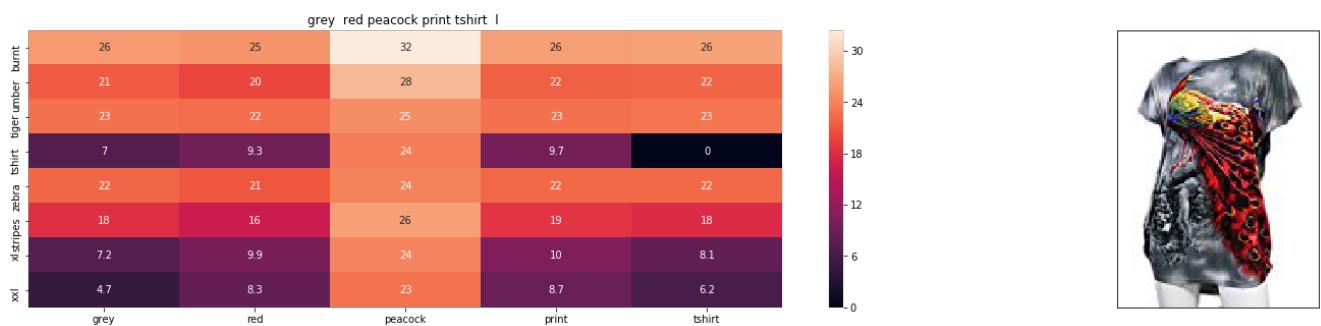
ASIN : B00JXQAFZ2

Brand : Si Row

euclidean distance from input : 3.387187195







ASIN : B00JXQCFRS

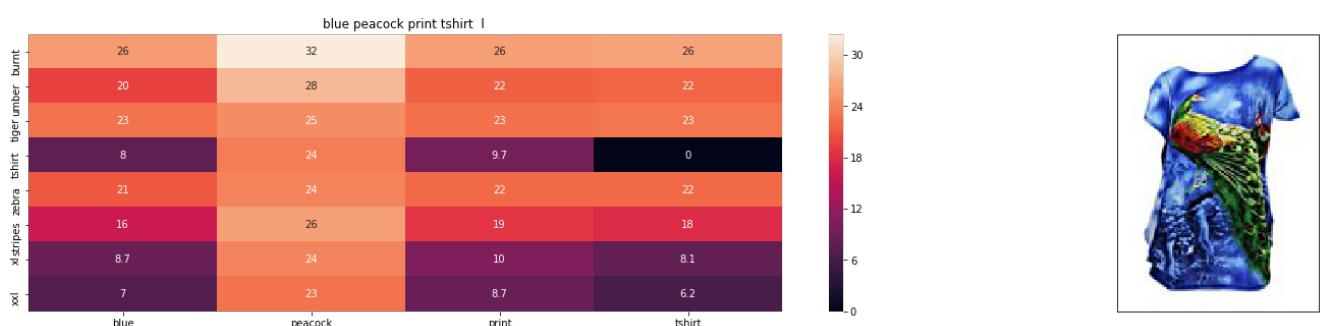
Brand : Si Row

euclidean distance from input : 4.12881164569

---



---



ASIN : B00JXQC8L6

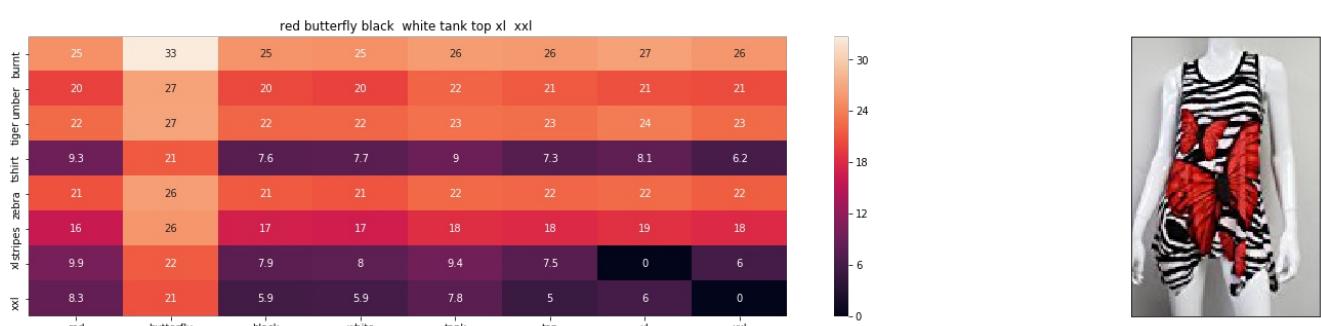
Brand : Si Row

euclidean distance from input : 4.20390052813

---



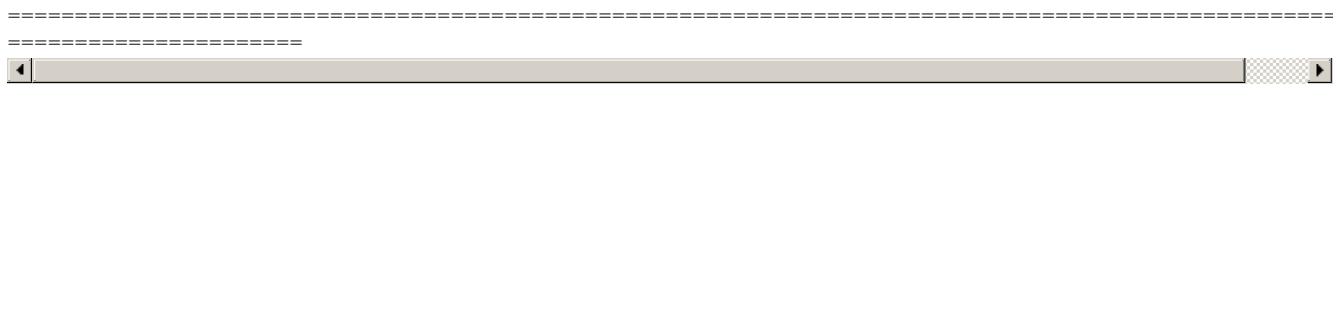
---



ASIN : B00JV63CW2

Brand : Si Row

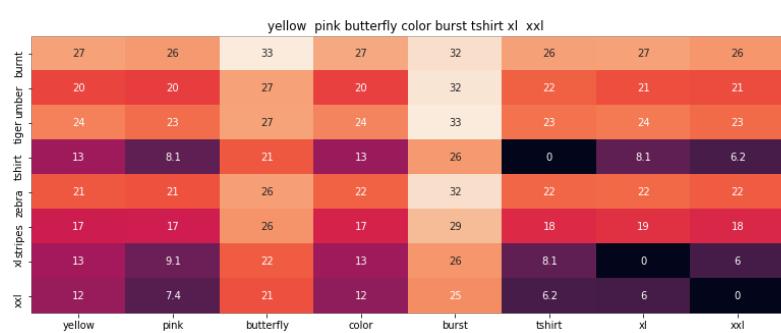
euclidean distance from input : 4.28658676166



ASIN : B015H41F6G

Brand : KINGDE

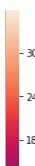
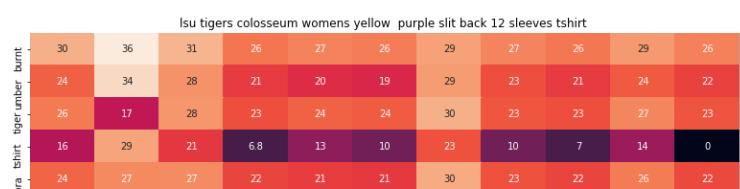
euclidean distance from input : 4.38937078798



ASIN : B00JXQBBM1

Brand : Si Row

euclidean distance from input : 4.39790992755





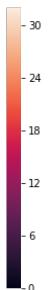
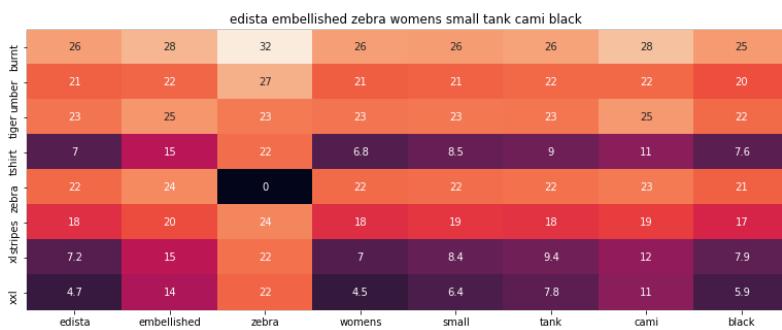
ASIN : B073R5Q8HD

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 : B01ER18406

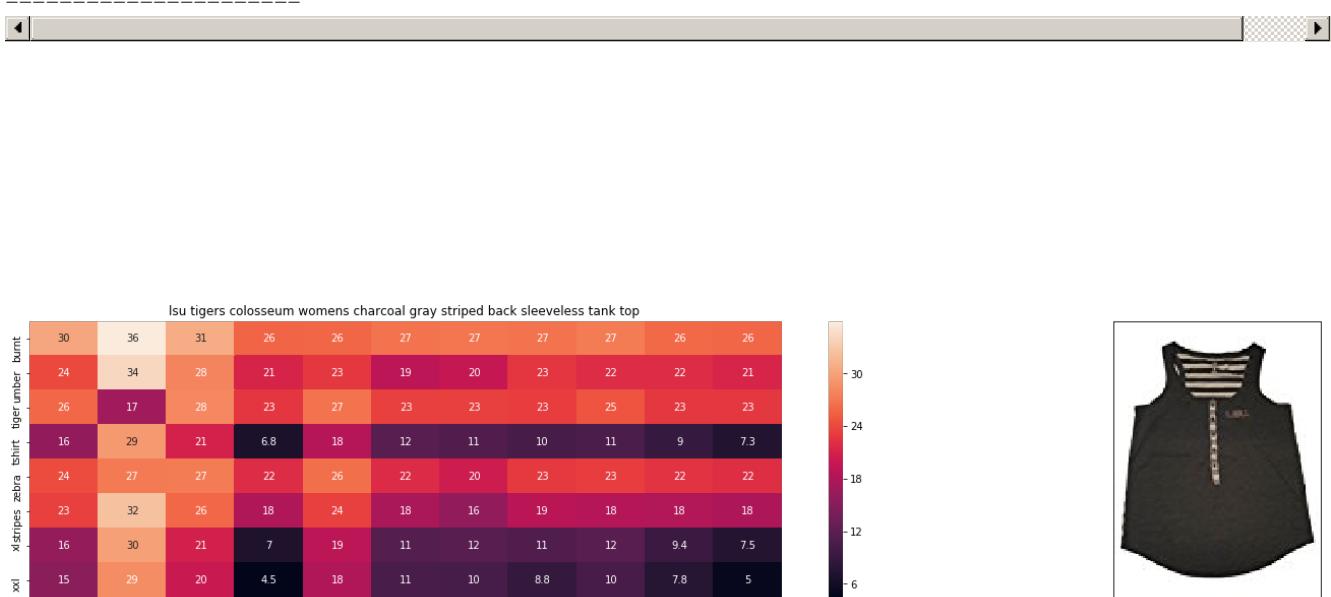
Brand : GuPoBoU168

euclidean distance from input : 4.54681702403

---



---



ASIN : B073R4ZM7Y

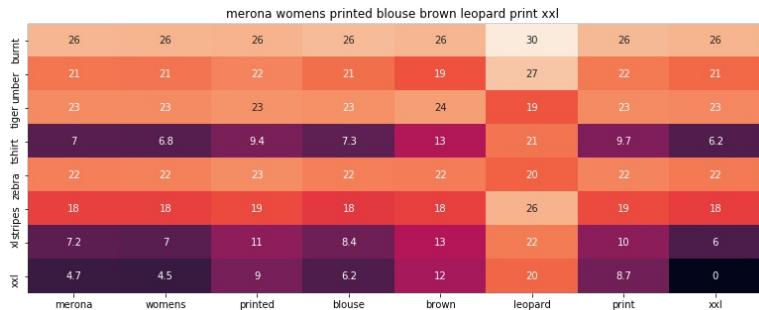
Brand : Colosseum

euclidean distance from input : 4.54835554445

---



---



ASIN : B071YF3WDD

Brand : Merona

euclidean distance from input : 4.61062742555

=====



ASIN : B01C60RLDQ

Brand : 1 Mad Fit

euclidean distance from input : 4.64591789282

=====



In [0]:

```
# brand and color weight =50
# title vector weight = 5

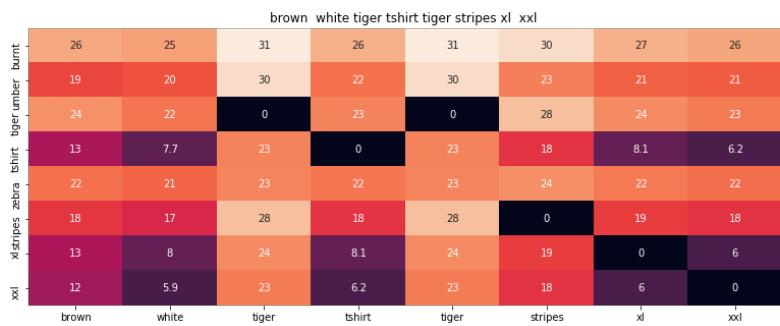
idf_w2v_brand(12566, 5, 50, 20)
```



ASIN : B00JXQB5FQ

Brand : Si Row

euclidean distance from input : 0.000355113636364



ASIN : B00JXQCWT0

Brand : Si Row

euclidean distance from input : 0.433722027865

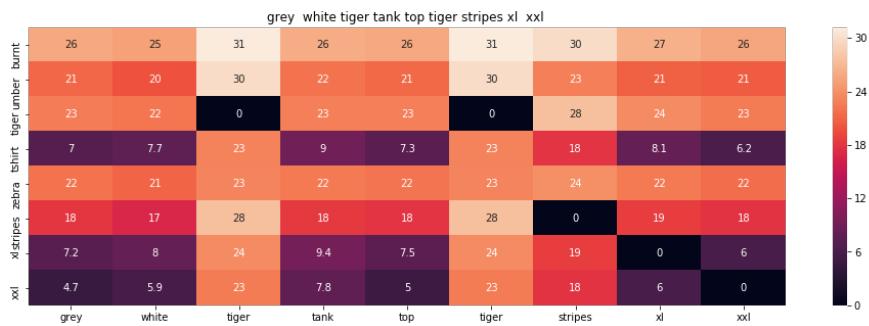


ASIN : B00JXQASS6

Brand : Si Row

euclidean distance from input : 1.65509310669

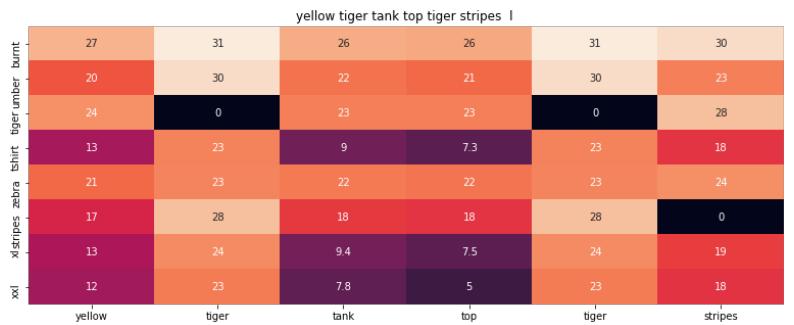




ASIN : B00JXQAFZ2

Brand : Si Row

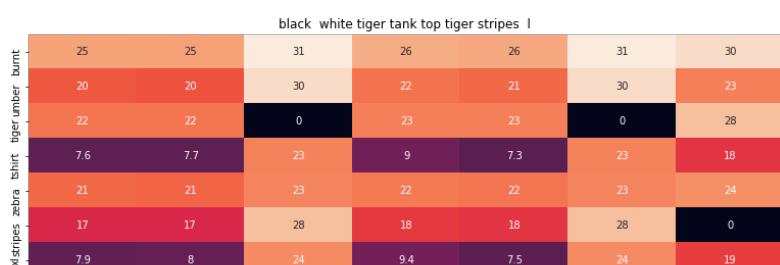
euclidean distance from input : 1.77293604103



ASIN : B00JXQAUWA

Brand : Si Row

euclidean distance from input : 1.80287808538





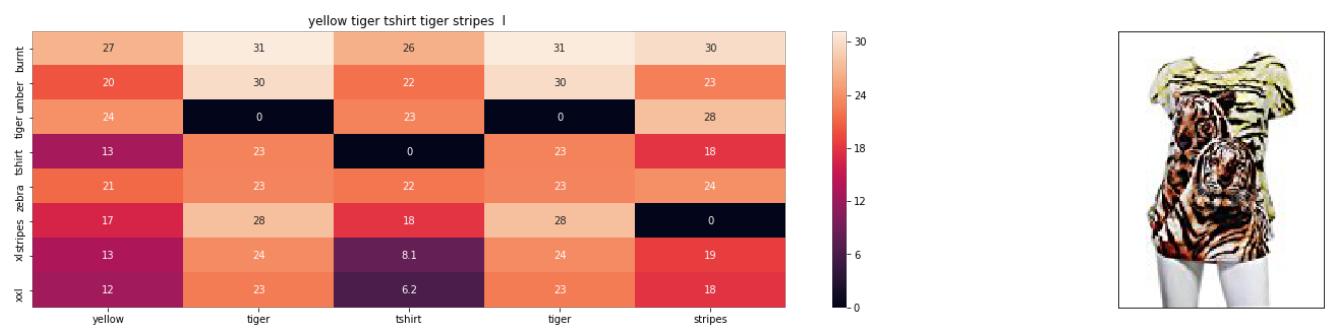
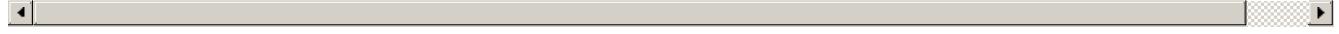
ASIN : B00JXQAO94

Brand : Si Row

euclidean distance from input : 1.80319609241

=====

=====



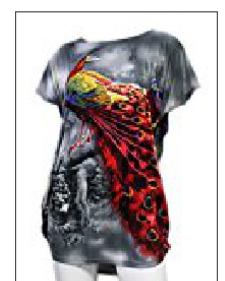
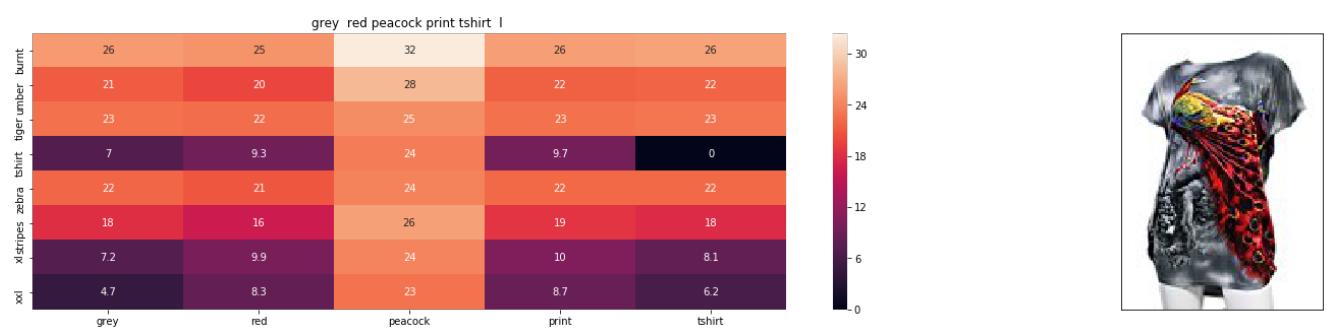
ASIN : B00JXQCUIC

Brand : Si Row

euclidean distance from input : 1.82141619628

=====

=====



ASIN : B00JXQCFRS

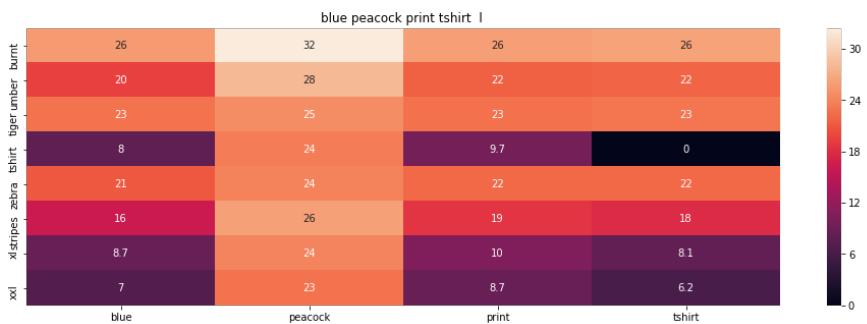
Brand : Si Row

euclidean distance from input : 1.90777685025

=====

=====

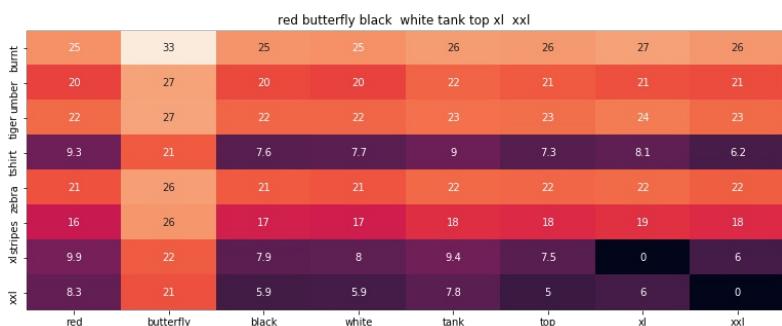




ASIN : B00JXQC8L6

Brand : Si Row

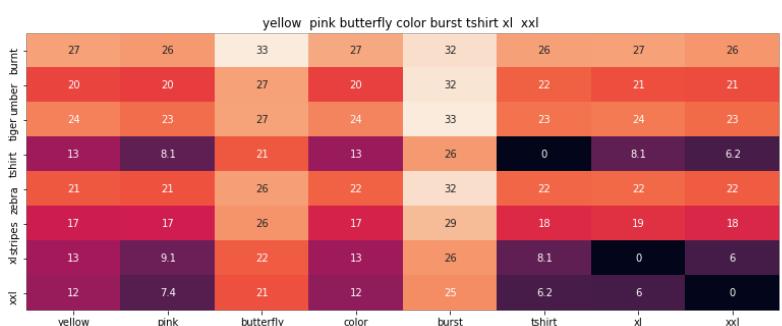
euclidean distance from input : 1.92142937433



ASIN : B00JV63CW2

Brand : Si Row

euclidean distance from input : 1.93646323497

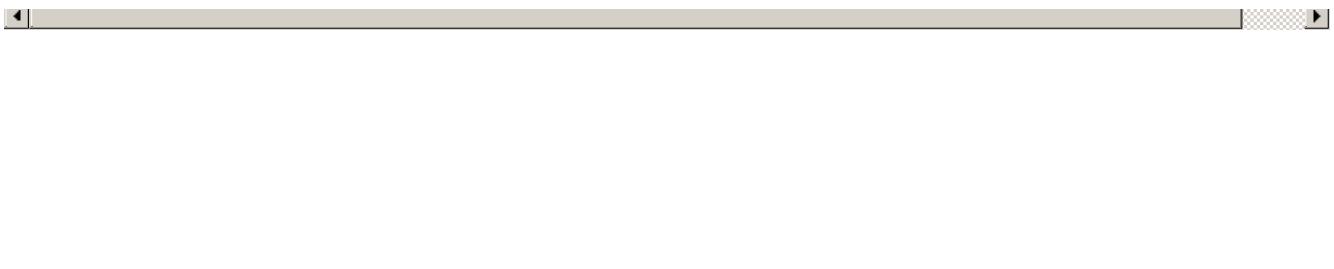


ASIN : B00JXQBBM1

Brand : Si Row

euclidean distance from input : 1.95670381059





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

---



---

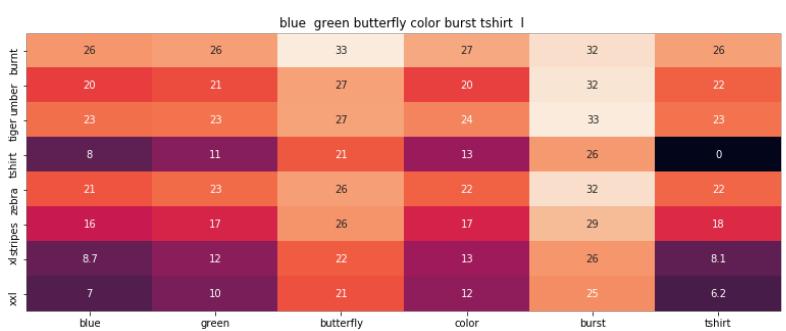




ASIN : B00JXQAX2C

Brand : Si Row

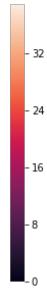
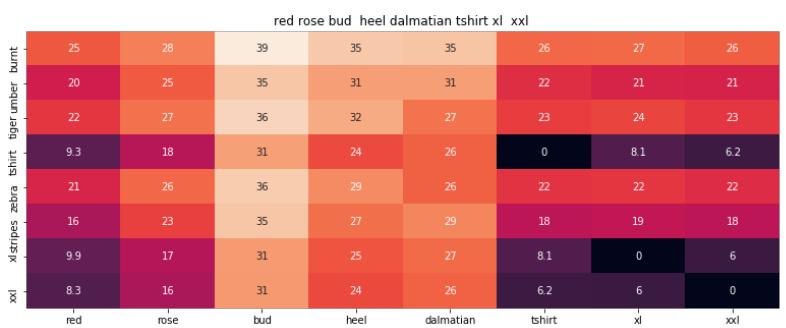
euclidean distance from input : 2.01335178755



ASIN : B00JXQC0C8

Brand : Si Row

euclidean distance from input : 2.01388334827

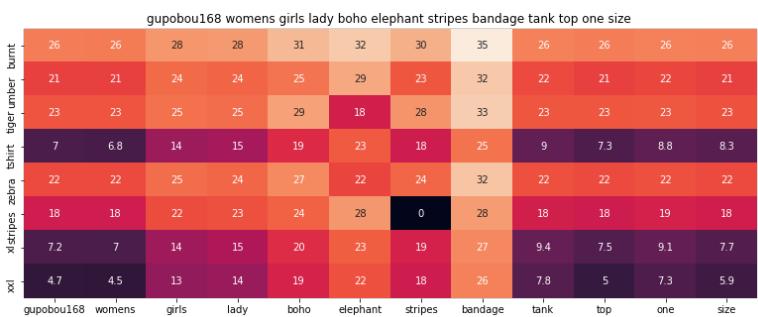


ASIN : B00JXQABB0

Brand : Si Row

euclidean distance from input : 2.0367257555





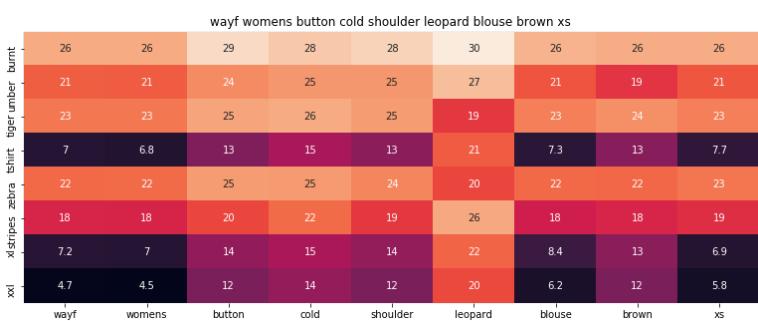
ASIN : B01ER18406

Brand : GuPoBoU168

euclidean distance from input : 2.65620416778

=====

=====



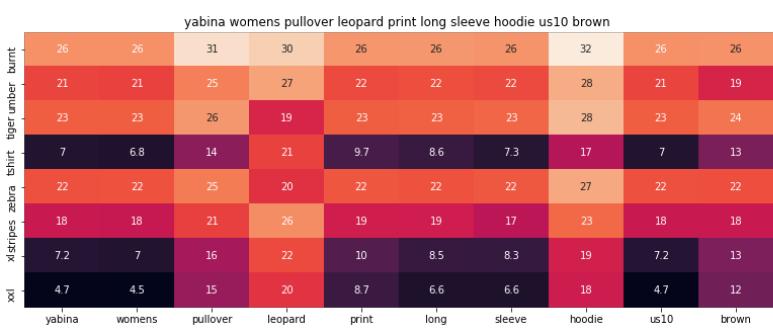
ASIN : B01LZ7BQ4H

Brand : WAYF

euclidean distance from input : 2.6849067823

=====

=====



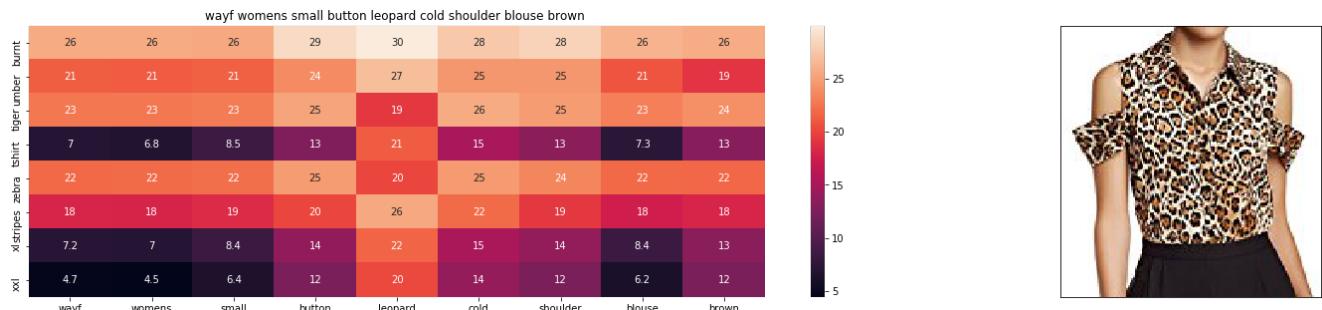
ASIN : B01KJUM6JI

Brand : YABINA

euclidean distance from input : 2.68583819266

=====

=====



ASIN : B01M06V4X1

Brand : WAYF

euclidean distance from input : 2.69476194865

---



---



---

In [0] :