In [ ]:
```
#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 [ ]:
```
# we have give a json file which consists of all information about
# the products
# loading the data using pandas' read_json file.
data = pd.read_json('tops_fashion.json')
```

In [4]:
```
# each product/item has 19 features in the raw dataset.
data.columns # prints column-names or feature-names.
```

```
Index(['asin', 'author', 'availability', 'availability_type', 'brand', 'colo
r',
       'editorial_reivew', '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 [ ]:
```
# 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():
            # 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
```

In [ ]:
```
# we take each title and we text-preprocess it.
for index, row in data.iterrows():
    nlp_preprocessing(row['title'], index, 'title')
```

In [ ]:
```
#saving the data
data.to_pickle('pickels/16k_apperal_data_preprocessed')
```

In [ ]:
```
data = pd.read_pickle('pickels/16k_apperal_data_preprocessed') # data after re
moving stopwords
data.head()
```

Out[ ]:

	asin	brand	color	medium_image_url	product_type_name	title	forn
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 [ ]:
```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_transform(data['title'])

# idf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dim
ensions #data_points * #words_in_corpus
# idf_title_features[doc_id, index_of_word_in_corpus] = number of times the wo
rd occured in that doc
```

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

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

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

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

    # to calculate idf_title_features we need to replace the count values with
the idf values of the word
    # idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzero()[0]
 will return all documents in which the word i present
    for j in idf_title_features[:, idf_title_vectorizer.vocabulary_[i]].nonzer
o()[0]:

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

In [ ]:
```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# it occupies ~9Gb, so please do this step only if you have >12G of ram
# we will provide a pickle file wich contains a dict ,
# and it contains all our courpus words as keys and  model[word] as values
# To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
# it's 1.9GB in size.

'''
model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bi
n', binary=True)
'''

#if you do NOT have RAM >= 12GB, use the code below.
with open('word2vec_model', 'rb') as handle:
    model = pickle.load(handle)
```

In [ ]:
```
# Utility functions
#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)

def get_word_vec(sentence, doc_id, m_name):
    # sentence : title of the apparel
    # doc_id: document id in our corpus
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation
of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the id
f(word)
    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in  idf_title_vectorizer.vocabulary_
:
                vec.append(idf_title_features[doc_id, idf_title_vectorizer.voc
abulary_[i]] * model[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our courpus is not there in the google word2vec c
orpus, we are just ignoring it
            vec.append(np.zeros(shape=(300,)))
    # we will return a numpy array of shape (#number of words in title * 300 )
300 = len(w2v_model[word])
    # each row represents the word2vec representation of each word (weighted/a
vg) in given sentance
    return  np.array(vec)

def get_distance(vec1, vec2):
    # vec1 = np.array(#number_of_words_title1 * 300), each row is a vector of
 length 300 corresponds to each word in give title
    # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of
 length 300 corresponds to each word in give title

    final_dist = []
    # for each vector in vec1 we caluclate the distance(euclidean) to all vect
ors in vec2
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result the euclidean distance between v
ectors i, j
            dist.append(np.linalg.norm(i-j))
        final_dist.append(np.array(dist))
    # final_dist = np.array(#number of words in title1 * #number of words in t
itle2)
    # final_dist[i,j] = euclidean distance between vectors i, j
    return np.array(final_dist)
```

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

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(we
ighted/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(we
ighted/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 t
itle2)
    # 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 disp
lays image of apparel
    gs = gridspec.GridSpec(2, 2, width_ratios=[4,1],height_ratios=[2,1])
    fig = plt.figure(figsize=(15,15))

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

    ax = 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 [ ]:
```
# 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 o
f given sentance
def build_avg_vec(sentence, num_features, doc_id, m_name):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vector, its values = 300
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation
of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the id
f(word)

    featureVec = np.zeros((num_features,), dtype="float32")
    # we will intialize a vector of size 300 with all zeros
    # we add each word2vec(wordi) to this fetureVec
    nwords = 0

    for word in sentence.split():
        nwords += 1
        if word in vocab:
            if m_name == 'weighted' and word in  idf_title_vectorizer.vocabula
ry_:
                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 [ ]:
```
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 [ ]:
```
# 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)
```

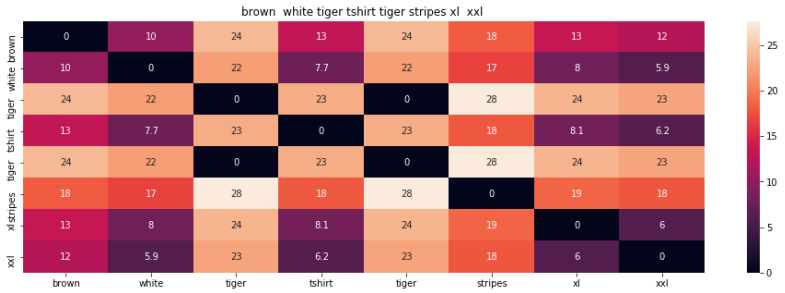
In [ ]:
```
#load the features and corresponding ASINS info.
bottleneck_features_train = np.load('16k_data_cnn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.npy')
asins = list(asins)
df_asins = list(data['asin'])
```

In [ ]:
```
def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df
_id2, model):

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

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(we
ighted/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(we
ighted/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 t
itle2)
    # 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], typ
es[doc_id1]], # input apparel's features
                   [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], typ
es[doc_id2]]] # recommonded apparel's features

    colorscale = [[0, '#1d004d'],[.5, '#f2e5ff'],[1, '#f2e5d1']] # to color th
e 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])
    # 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()
```

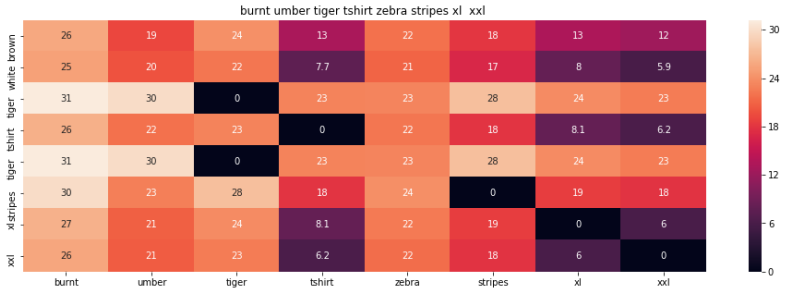
In [ ]:
```
def idf_w2v_brand_image(doc_id, text_w, brand_w,color_w,image_w, num_results):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    # pairwise_dist will store the distance from given input apparel to all re
maining 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
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_
id].reshape(1,-1))
    brand_feat_dist = pairwise_distances(brand_features, brand_features[doc_id
])
    color_feat_dist = pairwise_distances(color_features, color_features[doc_id
])
    image_dist= pairwise_distances(bottleneck_features_train, bottleneck_featu
res_train[doc_id].reshape(1,-1))
    pairwise_dist   = (text_w * idf_w2v_dist +  brand_w * brand_feat_dist +col
or_w * color_feat_dist +image_w*image_dist)/float(text_w + brand_w+color_w+ima
ge_w)

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

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


    for i in range(0, len(indices)):
        heat_map_w2v_brand(data['title'].loc[df_indices[0]],data['title'].loc[
df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indic
es[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_image(12566, 20,15,15,30, 20)
# in the give heat map, each cell contains the euclidean distance between word
s i, j
```

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQCWTO	Si-Row	Brown



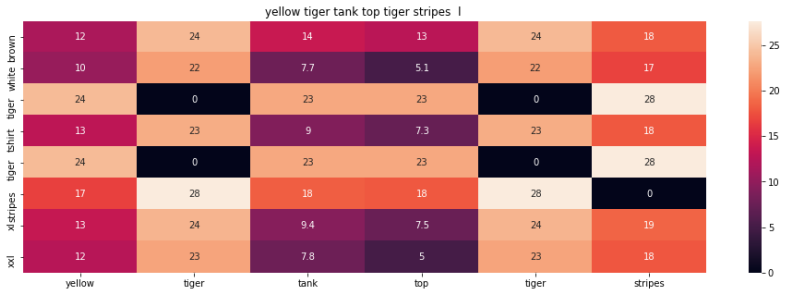
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.0
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQB5FQ	Si-Row	Brown



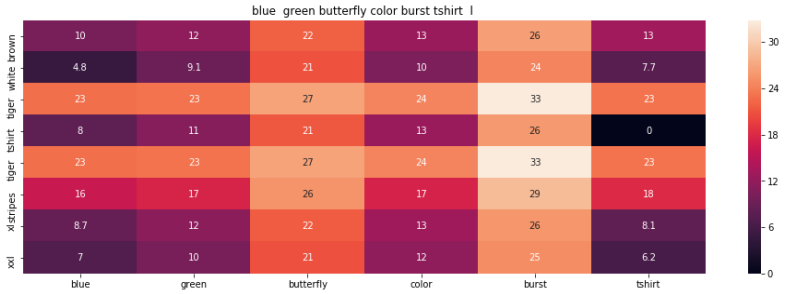
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 8.231806347680503e-11
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQAUWA	Si-Row	Yellow



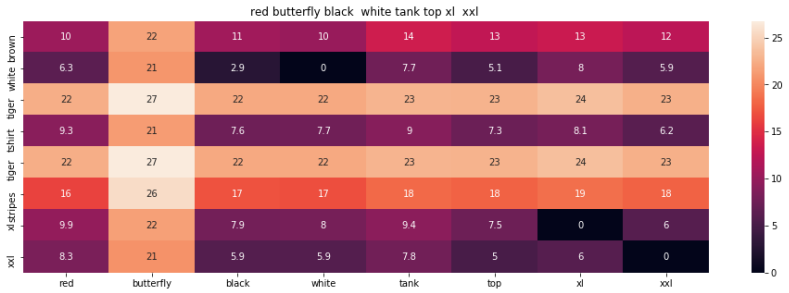
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 0.26516504294495535
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQC0C8	Si-Row	Blue



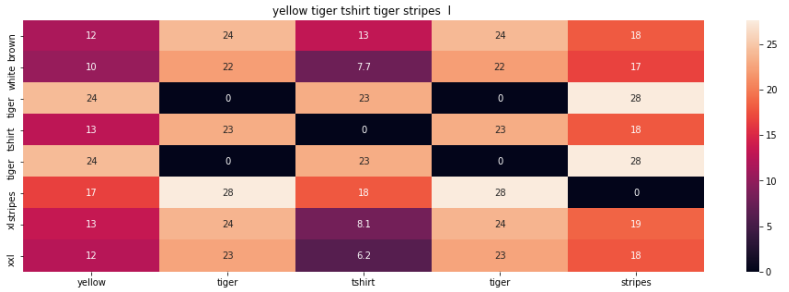
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 0.26516504294495535
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JV63CW2	Si-Row	Red



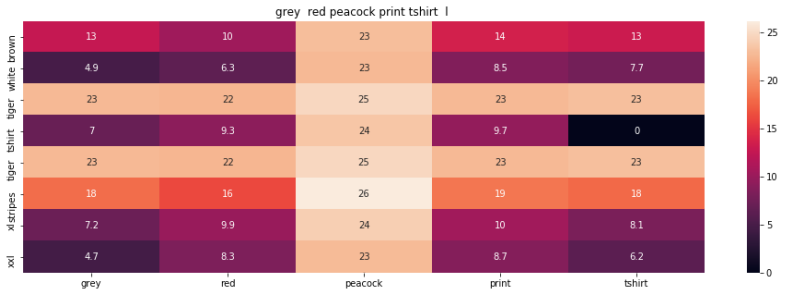
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 0.26516504294495535
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQCUIC	Si-Row	Yellow



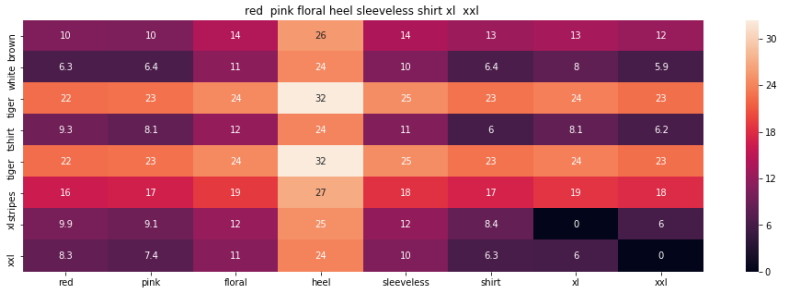
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 0.26516504294495535
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQCFRS	Si-Row	Grey



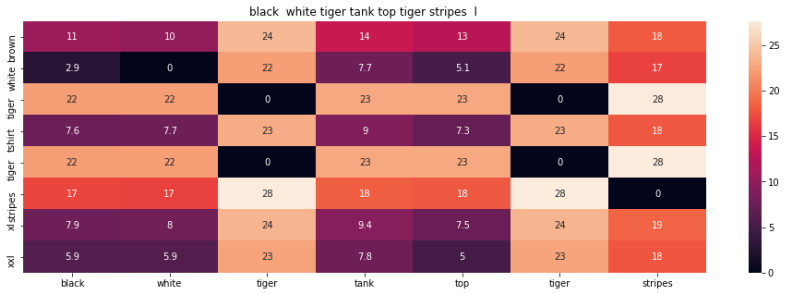
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 0.26516504294495535
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JV63QQE	Si-Row	Red



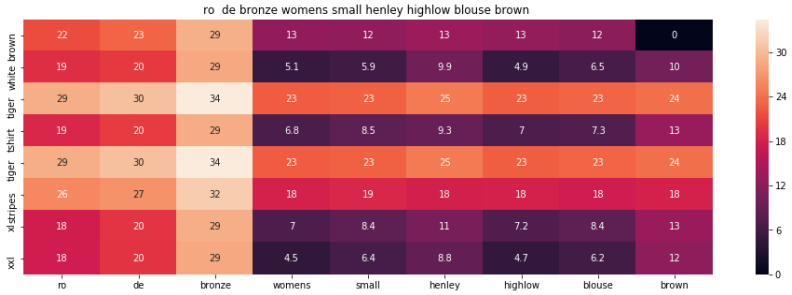
ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 0.26516504294495535
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQAO94	Si-Row	White



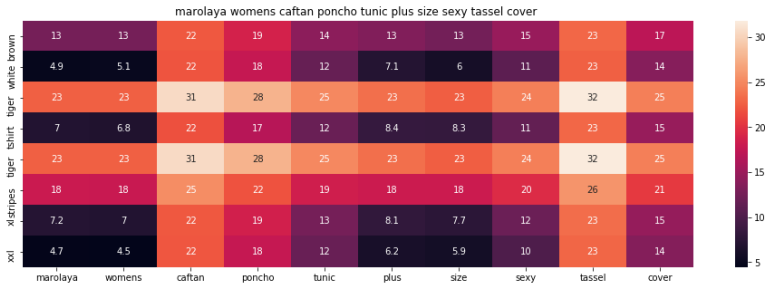
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 0.26516504294495535
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01L0T5PMW	Rode	Brown



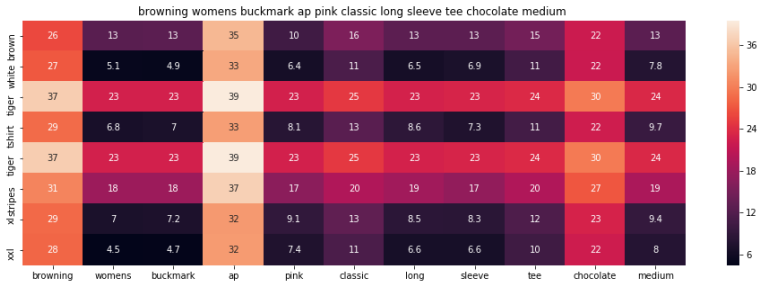
ASIN : B01L0T5PMW
Brand : Rode
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01CE40W16	Marolaya	Brown



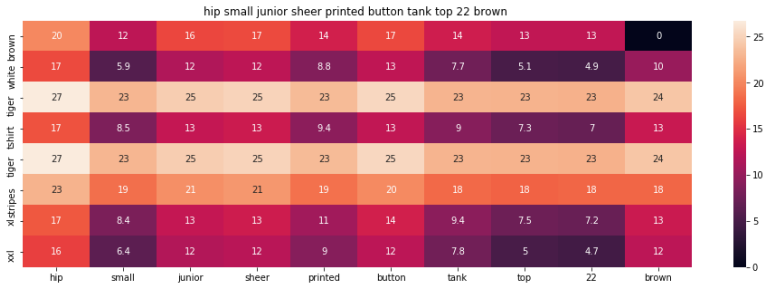
ASIN : B01CE40W16
Brand : Marolaya
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00NQFH7MA	Browning	Brown



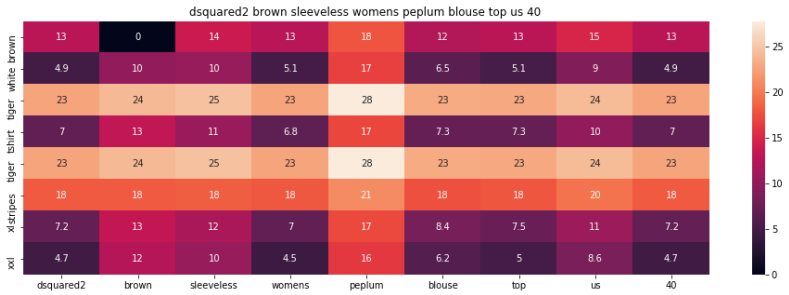
ASIN : B00NQFH7MA
Brand : Browning
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B071LDTQ1F	Hip	Brown



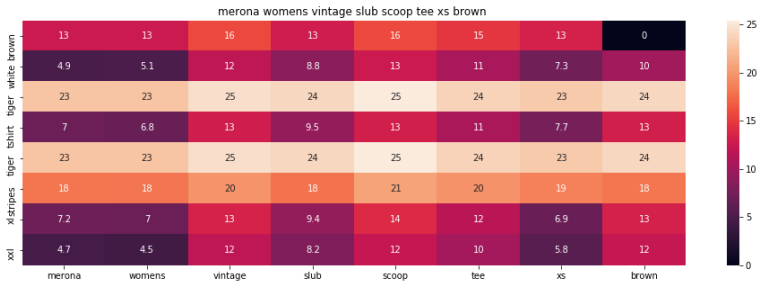
ASIN : B071LDTQ1F
Brand : Hip
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01EG15FVC	DSQUARED2	Brown



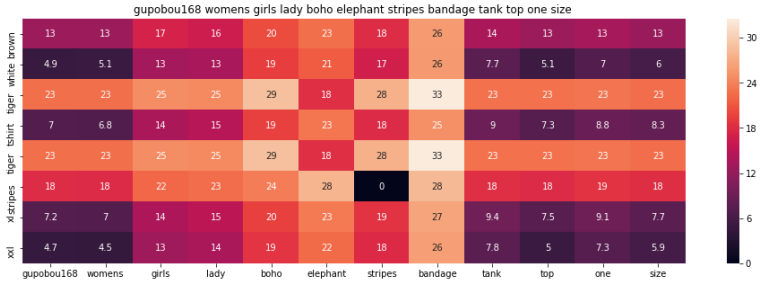
ASIN : B01EG15FVC
Brand : DSQUARED2
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01M5KO072	Merona	Brown



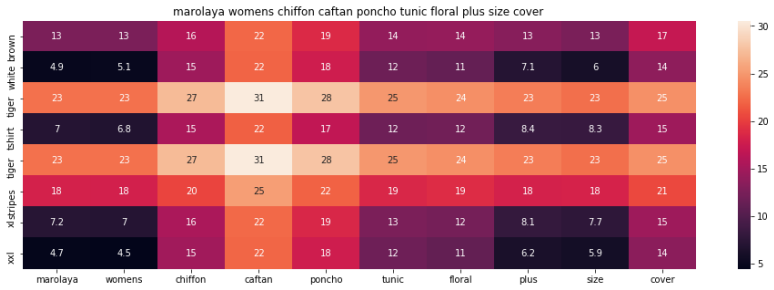
ASIN : B01M5KO072
Brand : Merona
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01ER184O6	GuPoBoU168	Brown



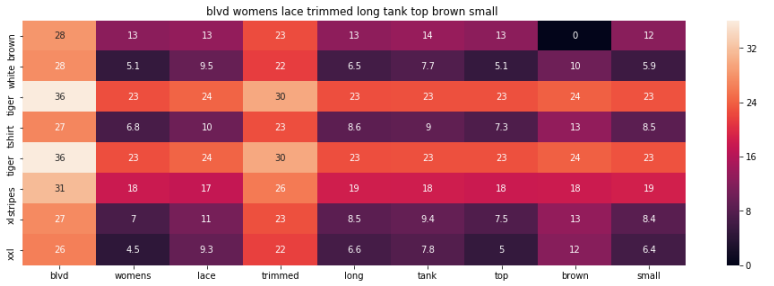
ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01CE40VX0	Marolaya	Brown



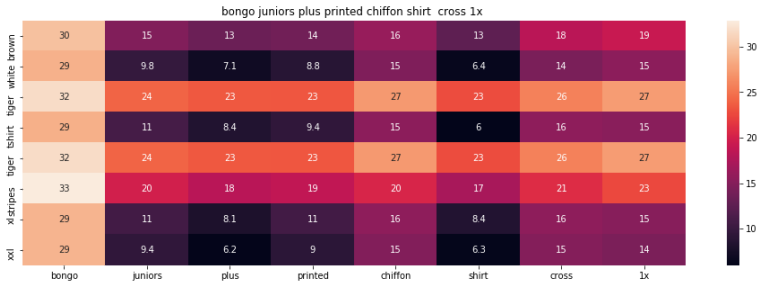
ASIN : B01CE40VX0
Brand : Marolaya
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00RWB03MK	BLVD	Brown



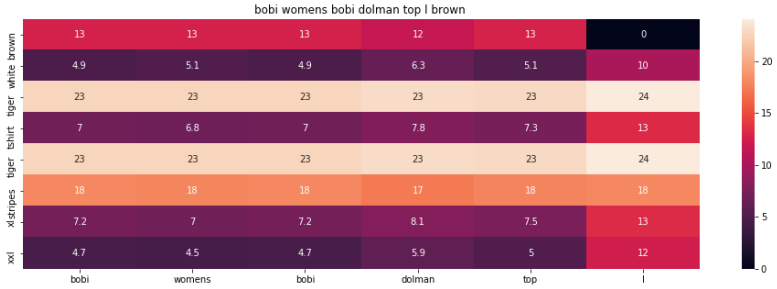
ASIN : B00RWB03MK
Brand : BLVD
euclidean distance from input : 0.32475952641916445
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00K02DWJO	Bongo	brown



ASIN : B00K02DWJO
Brand : Bongo
euclidean distance from input : 0.32475952641916445
==========
==========

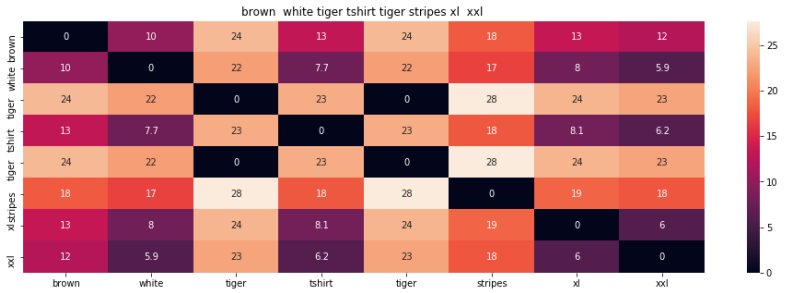
Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B06ZZX46Z6	Bobi	Brown



ASIN : B06ZZX46Z6
Brand : Bobi
euclidean distance from input : 0.32475952641916445
==========
==========

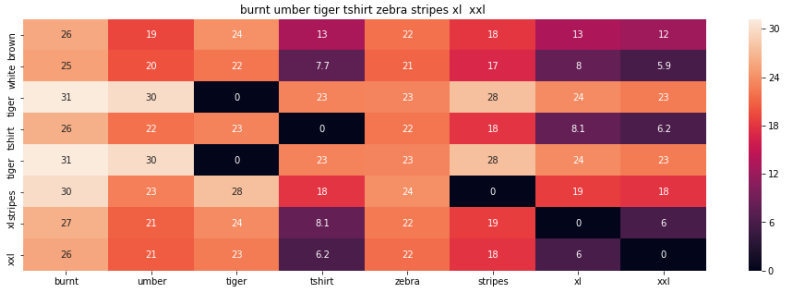
In [ ]:
```
def idf_w2v_brand_image(doc_id, text_w, brand_w,types_w,color_w,image_w, num_r
esults):
    # doc_id: apparel's id in given corpus
    # w1: weight for  w2v features
    # w2: weight for brand and color features

    # pairwise_dist will store the distance from given input apparel to all re
maining 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
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_
id].reshape(1,-1))
    brand_feat_dist = pairwise_distances(brand_features, brand_features[doc_id
])
    color_feat_dist = pairwise_distances(color_features, color_features[doc_id
])
    type_feat_dist = pairwise_distances(type_features, type_features[doc_id])
    image_dist= pairwise_distances(bottleneck_features_train, bottleneck_featu
res_train[doc_id].reshape(1,-1))
    pairwise_dist   = (text_w * idf_w2v_dist +types_w * type_feat_dist+  brand
_w * brand_feat_dist +color_w * color_feat_dist +image_w*image_dist)/float(tex
t_w + brand_w+color_w+types_w+image_w)

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

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


    for i in range(0, len(indices)):
        heat_map_w2v_brand(data['title'].loc[df_indices[0]],data['title'].loc[
df_indices[i]], data['medium_image_url'].loc[df_indices[i]], indices[0], indic
es[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_image(12566, 20,15,17,15,30, 30)
# in the give heat map, each cell contains the euclidean distance between word
s i, j
```

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQCWTO	Si-Row	Brown



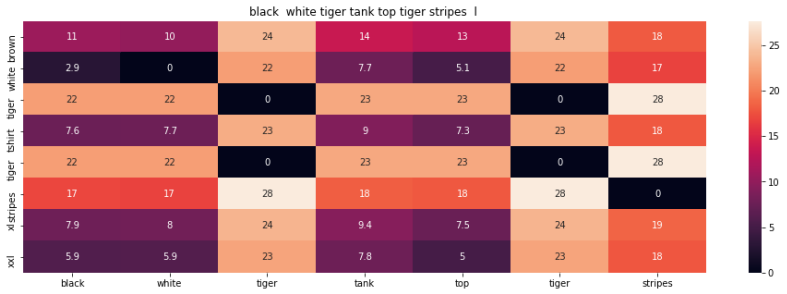
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.0
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQB5FQ	Si-Row	Brown



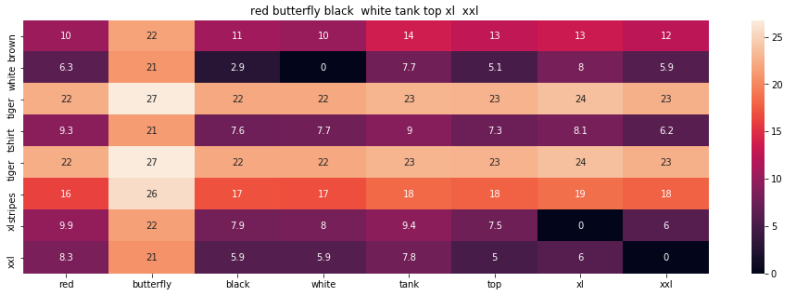
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 6.789118637262271e-11
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQAO94	Si-Row	White



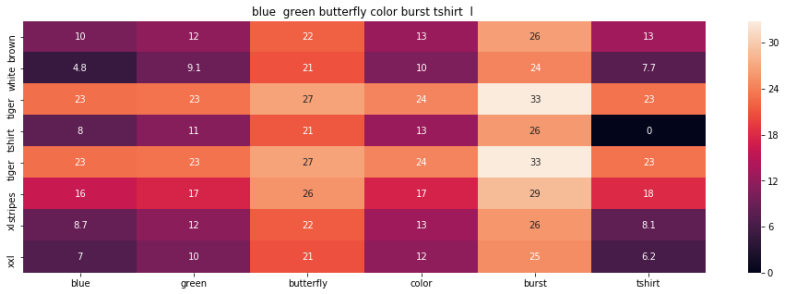
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 0.2186928189236745
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JV63CW2	Si-Row	Red



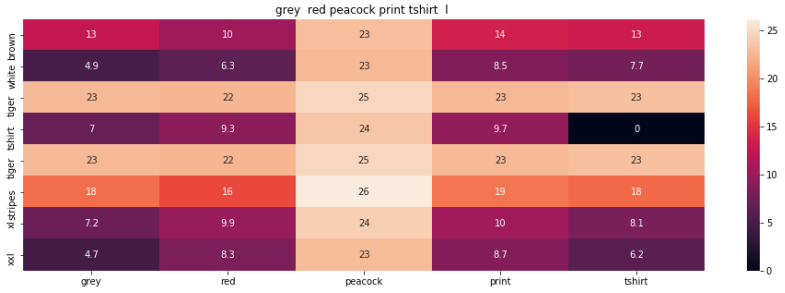
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 0.2186928189236745
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQC0C8	Si-Row	Blue



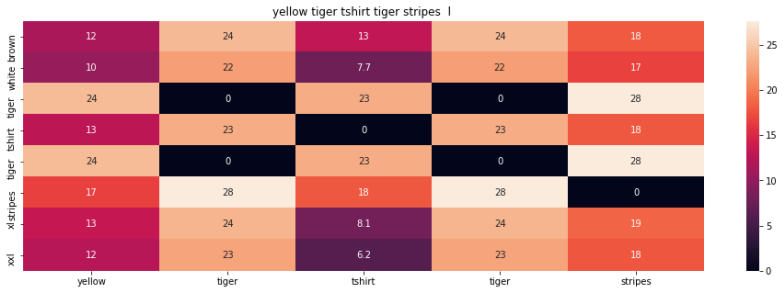
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 0.2186928189236745
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQCFRS	Si-Row	Grey



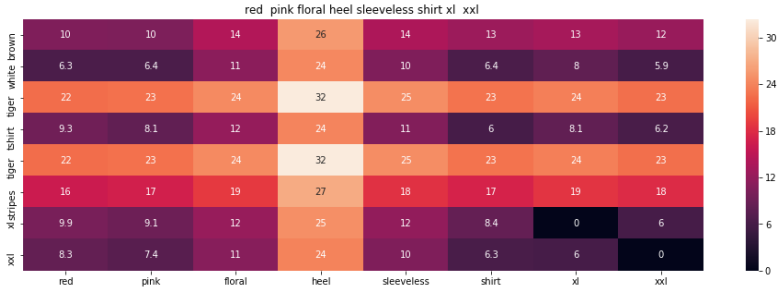
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 0.2186928189236745
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQCUIC	Si-Row	Yellow



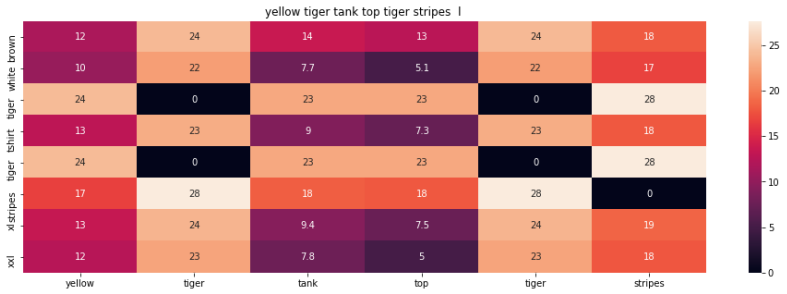
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 0.2186928189236745
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JV63QQE	Si-Row	Red



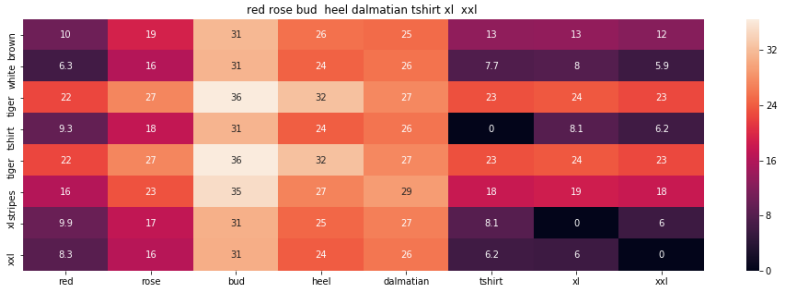
ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 0.2186928189236745
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQAUWA	Si-Row	Yellow



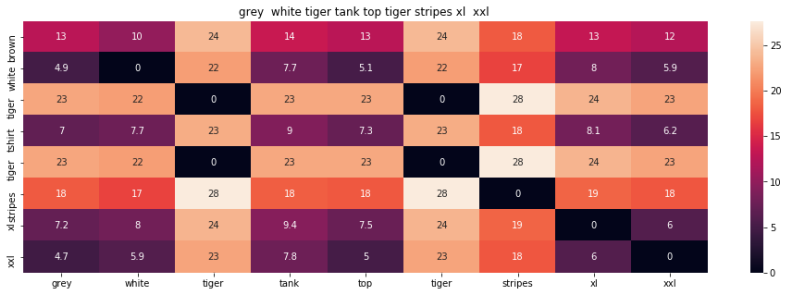
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 0.2186928189236745
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQABB0	Si-Row	Red



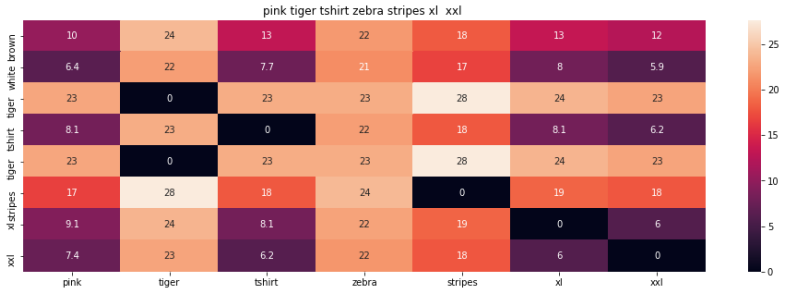
ASIN : B00JXQABB0
Brand : Si Row
euclidean distance from input : 0.4542097111897022
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQAFZ2	Si-Row	Grey



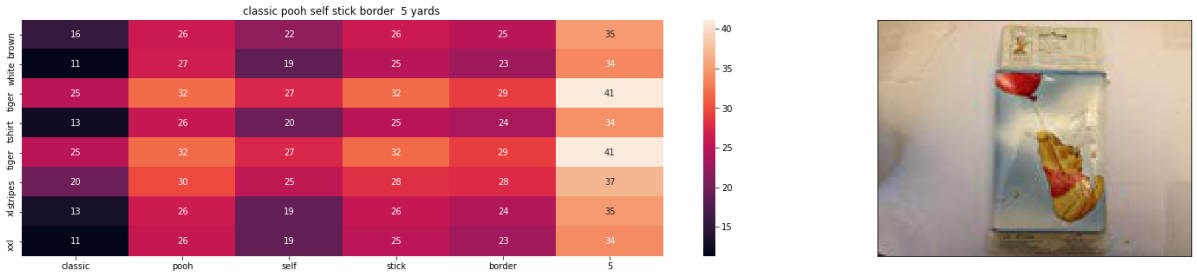
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 0.4654653097495242
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQASS6	Si-Row	Pink



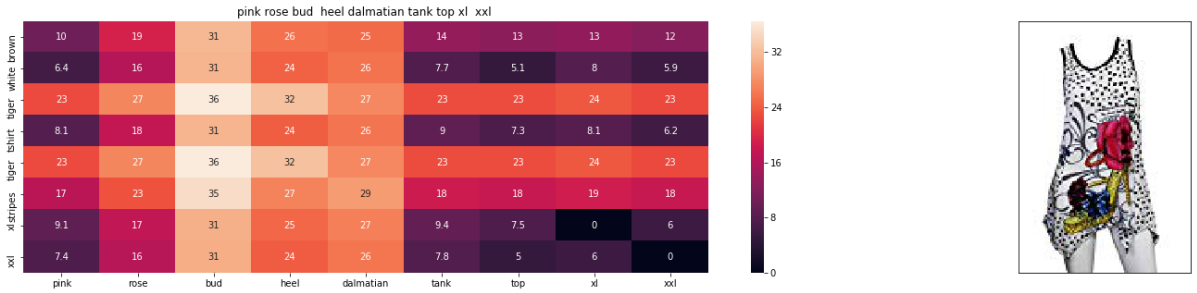
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 0.46859131651498814
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B001PO9BNW	Merona	Pink



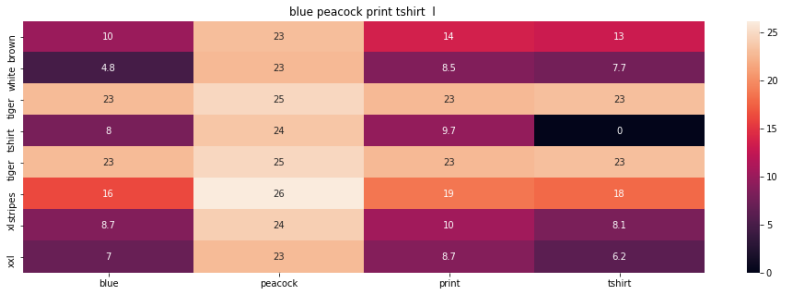
ASIN : B001PO9BNW
Brand : Merona
euclidean distance from input : 0.4865357273106143
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQAX2C	Si-Row	Pink



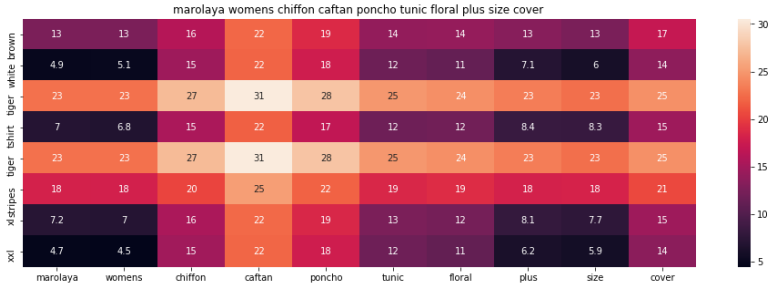
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 0.4882549404427829
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00JXQC8L6	Si-Row	Blue



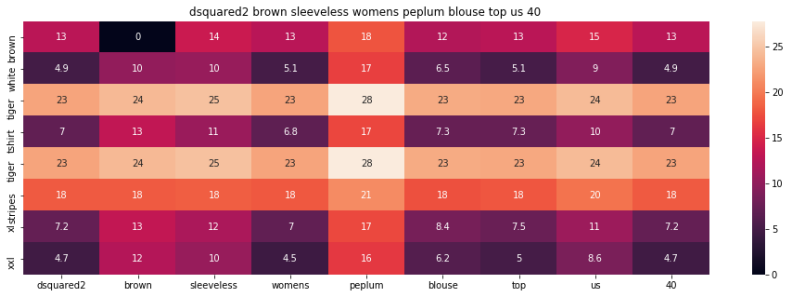
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 0.5102832440807299
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01CE40VX0	Marolaya	Brown



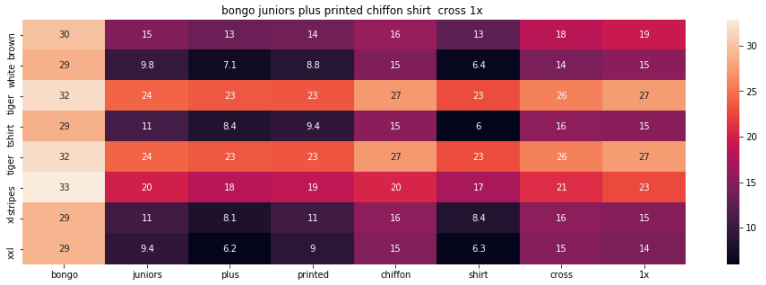
ASIN : B01CE40VX0
Brand : Marolaya
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01EG15FVC	DSQUARED2	Brown



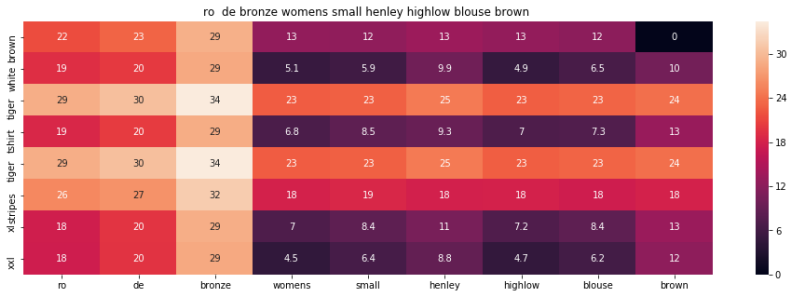
ASIN : B01EG15FVC
Brand : DSQUARED2
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00K02DWJO	Bongo	brown



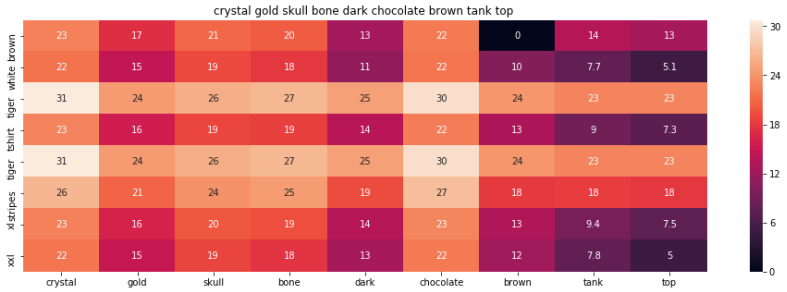
ASIN : B00K02DWJO
Brand : Bongo
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01L0T5PMW	Rode	Brown



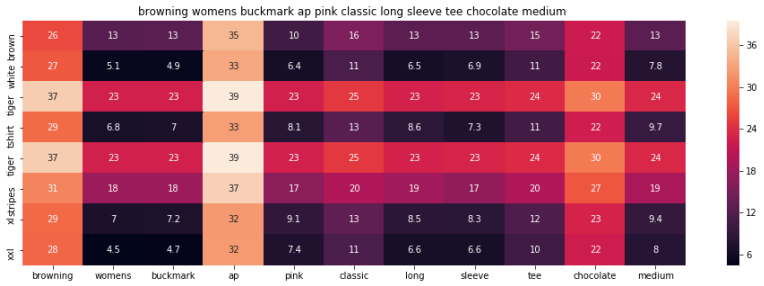
ASIN : B01L0T5PMW
Brand : Rode
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B004YR55O6	stonepowerss	Brown



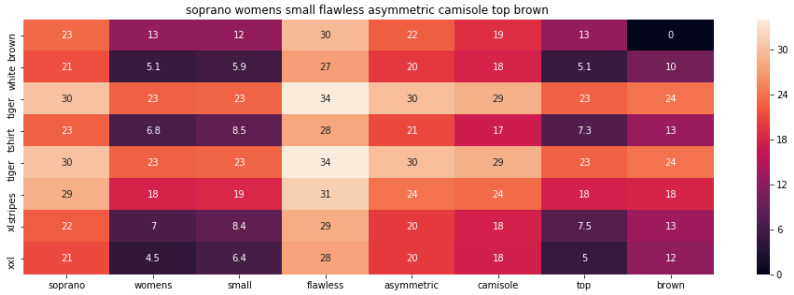
ASIN : B004YR55O6
Brand : stonepowerss
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00NQFH7MA	Browning	Brown



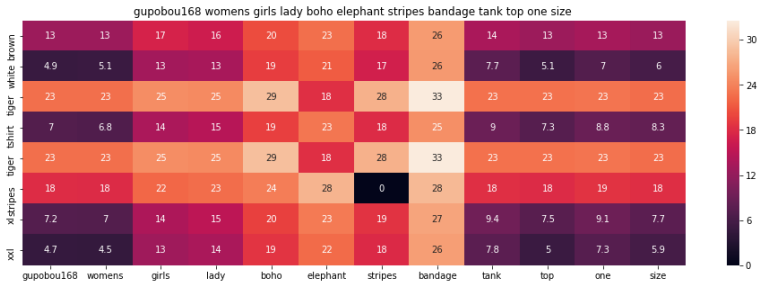
ASIN : B00NQFH7MA
Brand : Browning
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B0758356K3	Soprano	Brown



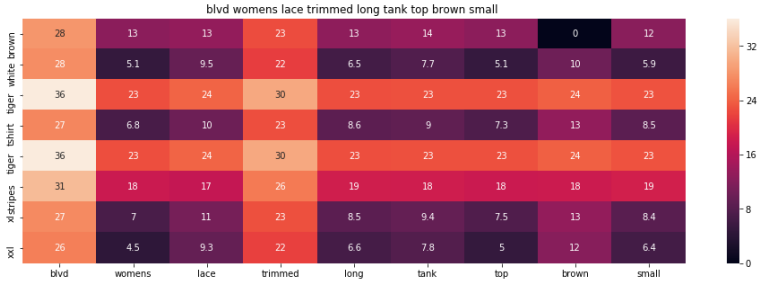
ASIN : B0758356K3
Brand : Soprano
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01ER184O6	GuPoBoU168	Brown



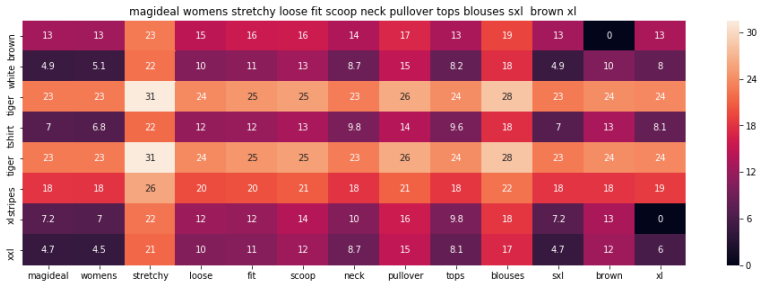
ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B00RWB03MK	BLVD	Brown



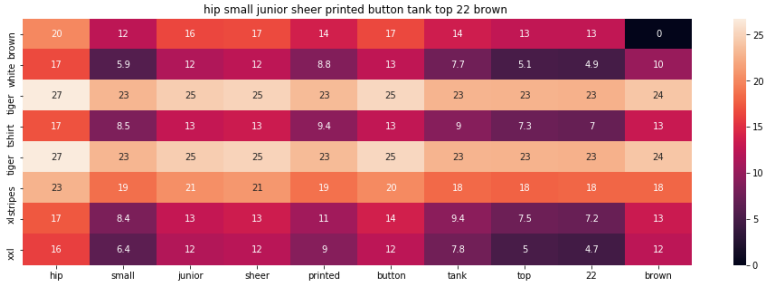
ASIN : B00RWB03MK
Brand : BLVD
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B07515JFBF	MagiDeal	Brown



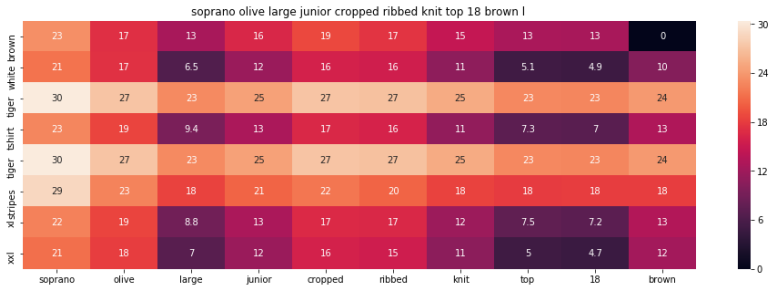
ASIN : B07515JFBF
Brand : MagiDeal
euclidean distance from input : 0.5156947698337708
==========
==========

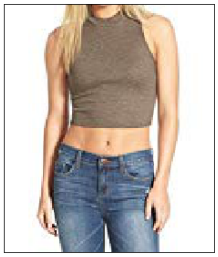
Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B071LDTQ1F	Hip	Brown



ASIN : B071LDTQ1F
Brand : Hip
euclidean distance from input : 0.5156947698337708
==========
==========

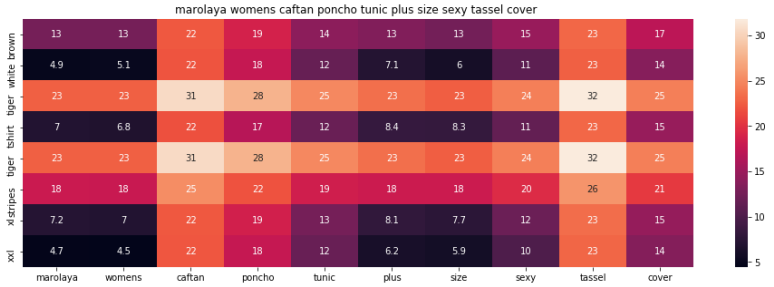
Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B07288KFHF	Soprano	Brown





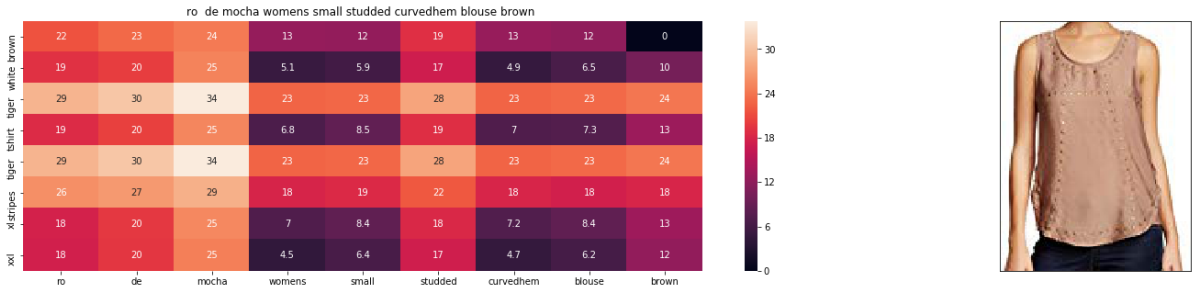
ASIN : B07288KFHF
Brand : Soprano
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01CE40W16	Marolaya	Brown



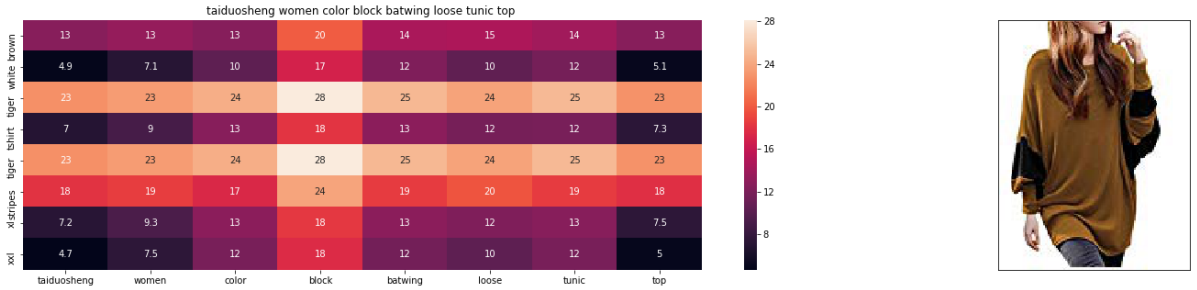
ASIN : B01CE40W16
Brand : Marolaya
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01M35MN7L	Rode	Brown



ASIN : B01M35MN7L
Brand : Rode
euclidean distance from input : 0.5156947698337708
==========
==========

Asin	Brand	Color
B00JXQCWTO	Si-Row	Brown
B01L8TH6B2	Taiduosheng	Brown



ASIN : B01L8TH6B2
Brand : Taiduosheng
euclidean distance from input : 0.5156947698337708
==========
==========

In [ ]: