Amazon Apparel Recommandation system

February 27, 2019

1 Amazon Apperal Recommendation system

In [1]:
```
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")
```

1.1 Loading pre-trained data

In [2]:
```
img_data = np.load('16k_data_cnn_features.npy')
asins = np.load('16k_data_cnn_feature_asins.npy')
data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
```

```
        df_asins = list(data['asin'])
        asins = list(asins)

In [3]: import pickle
        with open('word2vec_model', 'rb') as handle:
            model = pickle.load(handle)

In [4]: data['brand'].fillna(value="Not given", inplace=True )

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

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

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

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

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

In [5]: idf_title_vectorizer = CountVectorizer()
        idf_title_features = idf_title_vectorizer.fit_transform(data['title'])

In [6]: vocab = model.keys()
        # this function will add the vectors of each word and returns the avg vector of given
        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
                # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)

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

            for word in sentence.split():
                nwords += 1
                if word in vocab:
                    if m_name == 'weighted' and word in  idf_title_vectorizer.vocabulary_:
                        featureVec = np.add(featureVec, idf_title_features[doc_id, idf_title_ve
                    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 [7]: 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 do
        w2v_title_weight = np.array(w2v_title_weight)
        #print ("hi")

In [8]: 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 [9]: 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
                # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
            vec = []
            for i in sentence.split():
                if i in vocab:
                    if m_name == 'weighted' and i in  idf_title_vectorizer.vocabulary_:
                        vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_
                    elif m_name == 'avg':
                        vec.append(model[i])
                else:
                    # if the word in our courpus is not there in the google word2vec corpus, w
                    vec.append(np.zeros(shape=(300,)))
            # we will return a numpy array of shape (#number of words in title * 300 ) 300 = l
            # each row represents the word2vec representation of each word (weighted/avg) in g
            return  np.array(vec)

        def get_distance(vec1, vec2):
            # vec1 = np.array(#number_of_words_title1 * 300), each row is a vector of length 3
            # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length 3

            final_dist = []
            # for each vector in vec1 we caluclate the distance(euclidean) to all vectors in v
```

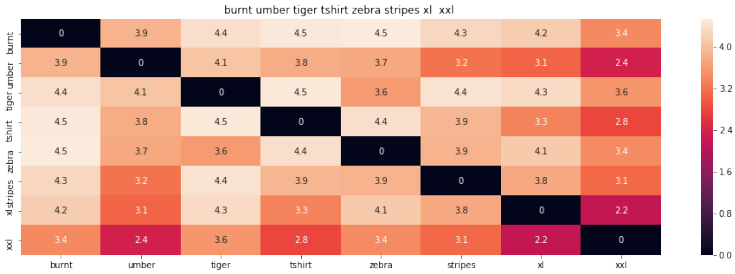
```
    for i in vec1:
        dist = []
        for j in vec2:
            # np.linalg.norm(i-j) will result the euclidean distance between vectors i
            dist.append(np.linalg.norm(i-j))
        final_dist.append(np.array(dist))
    # final_dist = np.array(#number of words in title1 * #number of words in title2)
    # final_dist[i,j] = euclidean distance between vectors i, j
    return np.array(final_dist)
def heat_map_w2v_brand(sentance1, sentance2, url, doc_id1, doc_id2, df_id1, df_id2, mo

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

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/a
    s1_vec = get_word_vec(sentance1, doc_id1, model)
    #s2_vec = np.array(#number_of_words_title2 * 300), each row is a vector(weighted/a
    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_i
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], types[doc_i

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

    # 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()
```

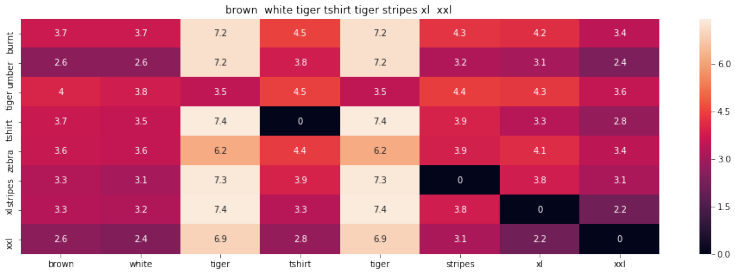
2 Weighted similarity using Title, brand and color

```
In [10]: def idf_w2v_brand(doc_id, w1, w2, 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
             # the metric we used here is cosine, the coside distance is mesured as K(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].resh
             ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
             pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist)/float(w1 + w2)

             # 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_indi
                 print('ASIN :',data['asin'].loc[df_indices[i]])
                 print('Brand :',data['brand'].loc[df_indices[i]])
                 print('euclidean distance from input :', pdists[i])
```

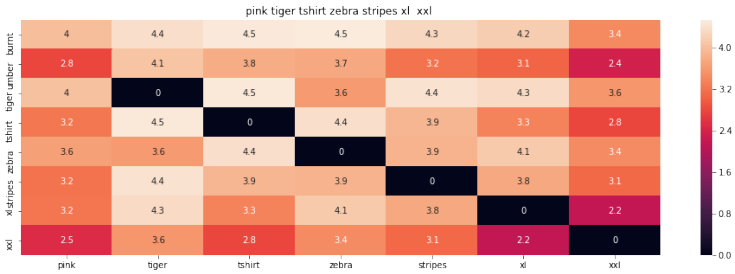
```
            print('='*125)

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



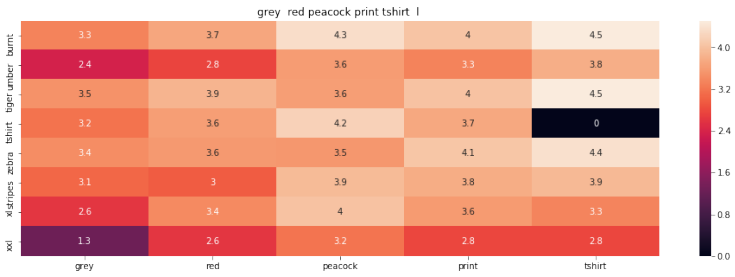
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0
==========



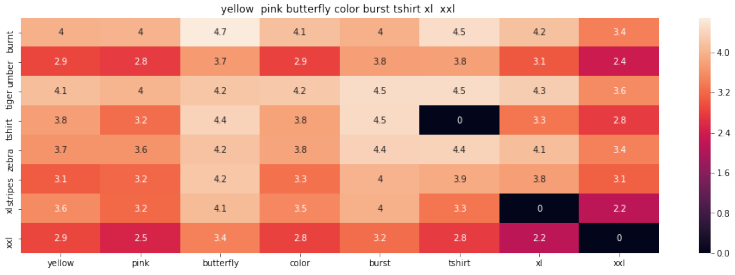
ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.6528800010681153
==========



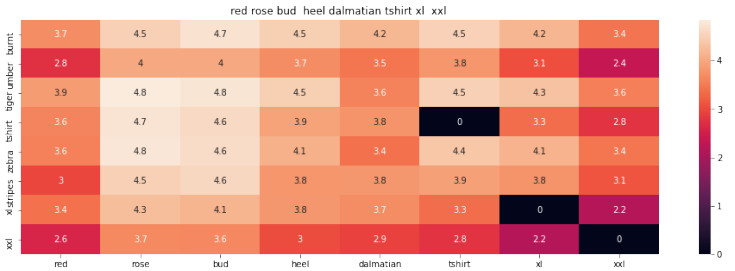
ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 1.001703190984216
==========



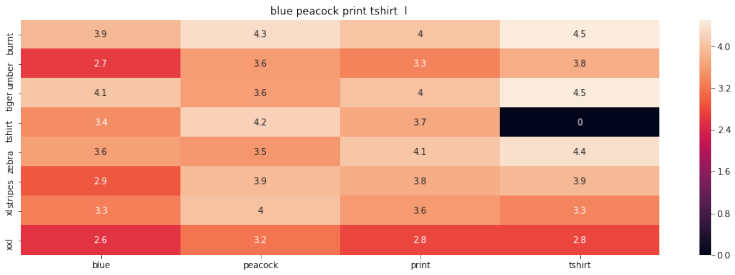
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.2372834207434322
==========



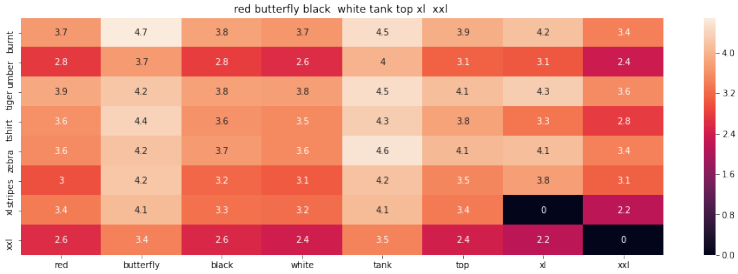
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.268622684659448
==========



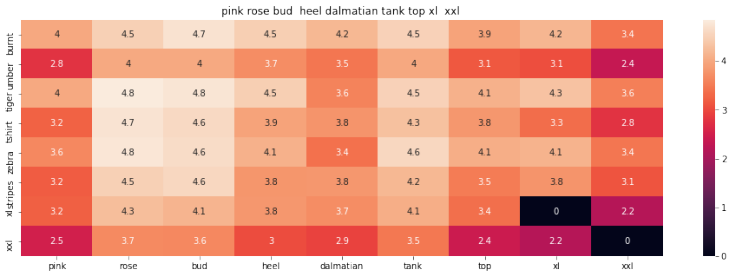
ASIN : B00JXQABB0
Brand : Si Row
euclidean distance from input : 1.2733484746832517
==========



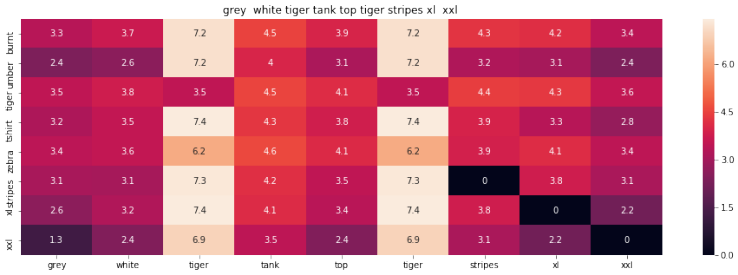
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 1.2818686487097408
==========



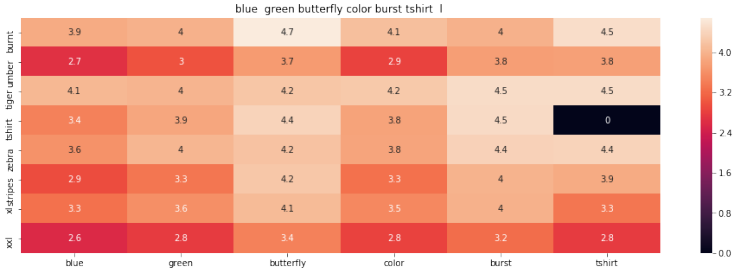
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 1.294748926343408
==========



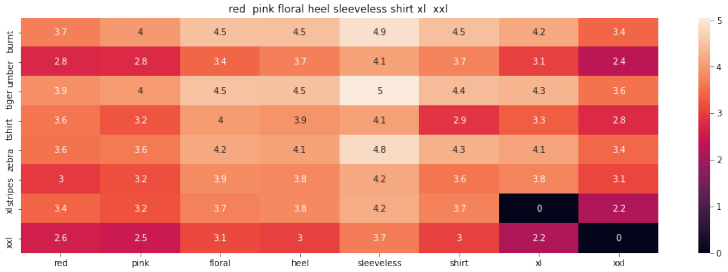
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 1.3236358644385007
==========



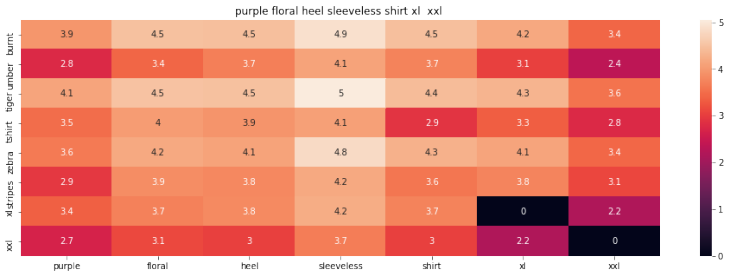
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 1.3293567659277585
==========



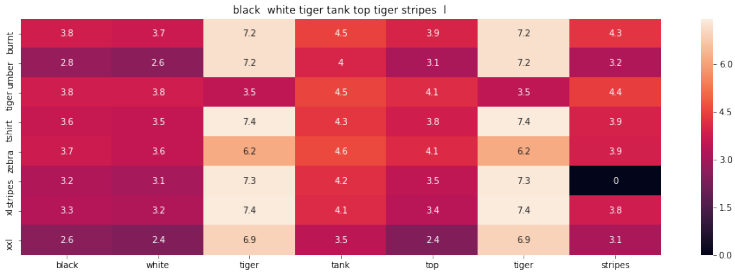
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 1.3499385358709959
==========



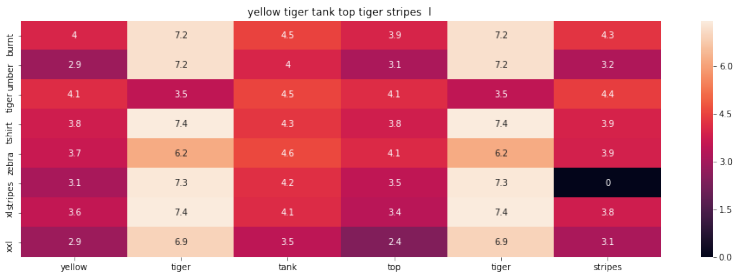
ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 1.3649899007696775
==========



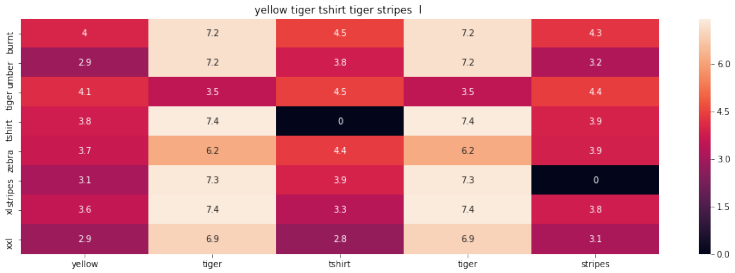
ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 1.408390236081567
==========



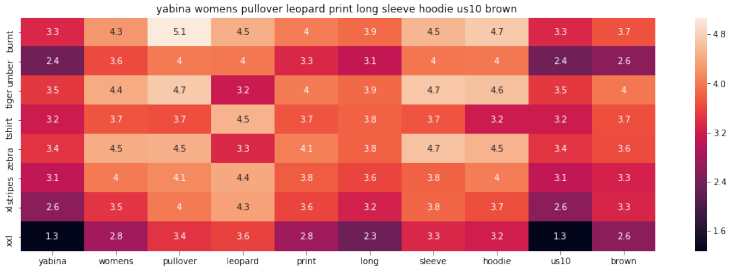
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 1.501950264157739
==========



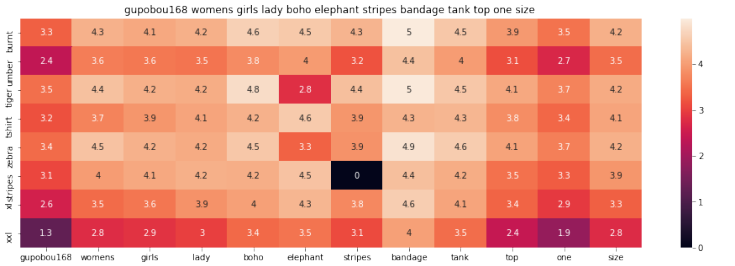
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 1.5693790437597896
==========



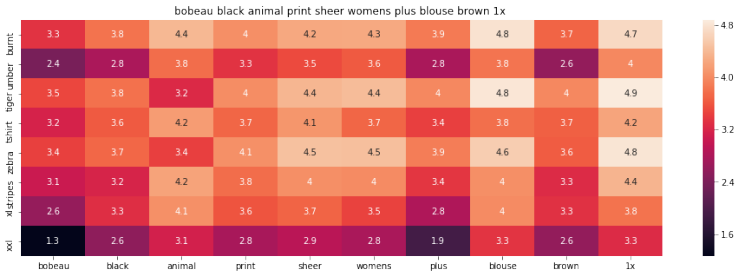
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 1.6333651544470456
==========



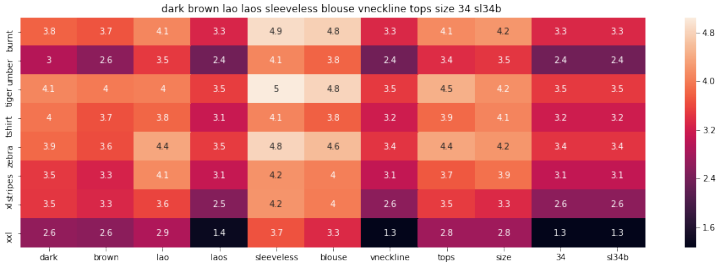
ASIN : B01KJUM6JI
Brand : YABINA
euclidean distance from input : 1.7323769166826342
==========



ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 1.7352904394029711
==========



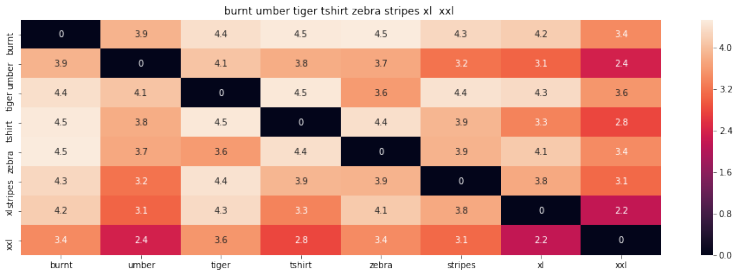
ASIN : B074MH886R
Brand : Bobeau
euclidean distance from input : 1.7428852632402514
==========



ASIN : B074J48RGW
Brand : Nanon
euclidean distance from input : 1.7484709337114428
==========

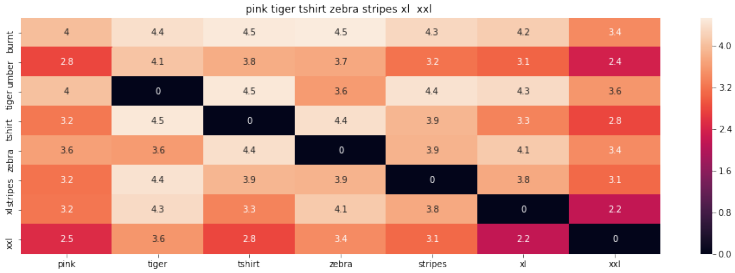
2.0.1 More Weight to Color

In [11]: idf_w2v_brand(12566, 5, 10, 20)



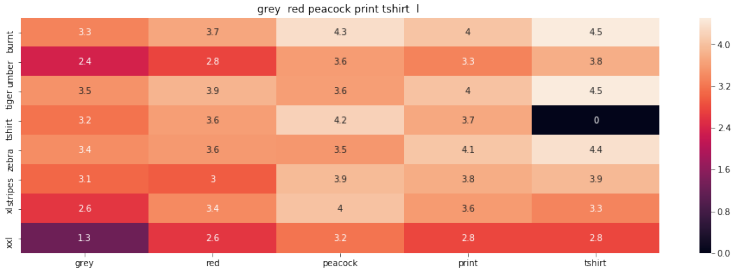
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 0.0
==========

ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from input : 0.43525333404541017
==========

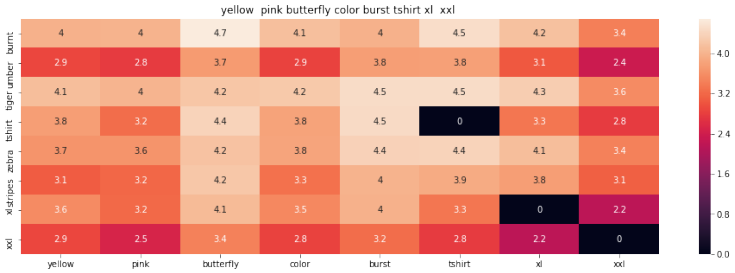


ASIN : B00JXQASS6
Brand : Si Row

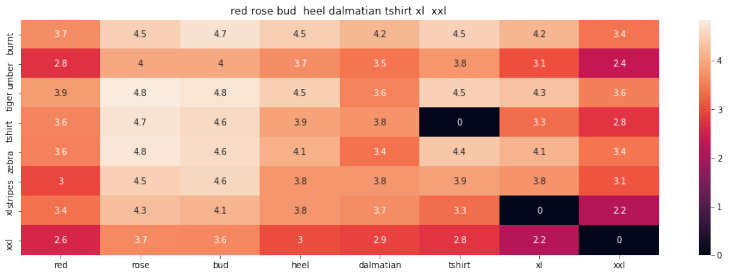
euclidean distance from input : 1.1392066481138423



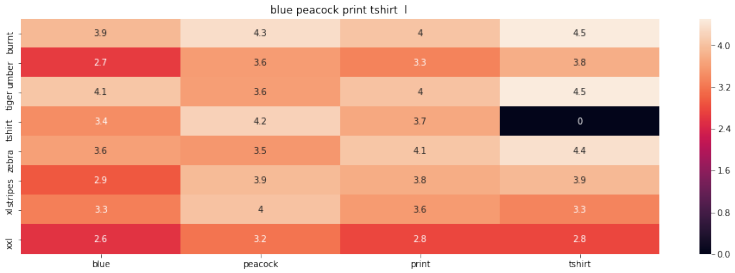
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.2962601346199867



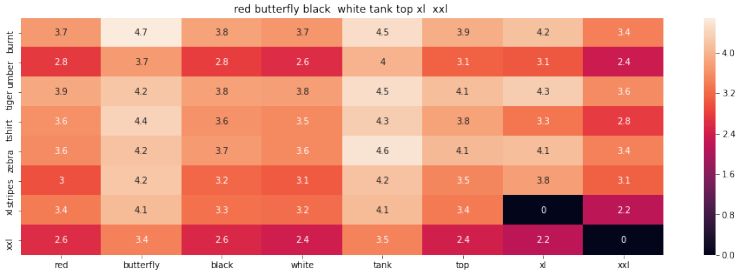
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.3171529772306636



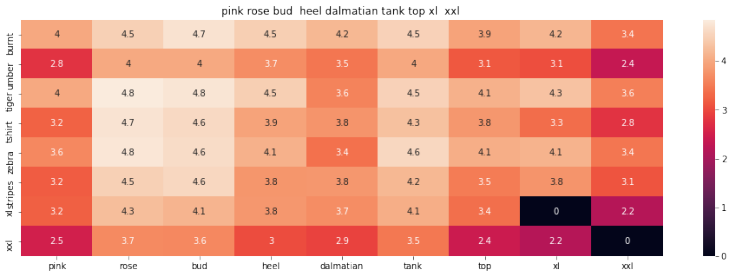
ASIN : B00JXQABB0
Brand : Si Row
euclidean distance from input : 1.3203035039131994
==========



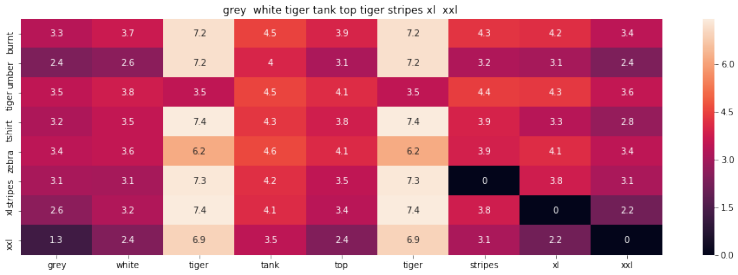
ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 1.325983619930859
==========



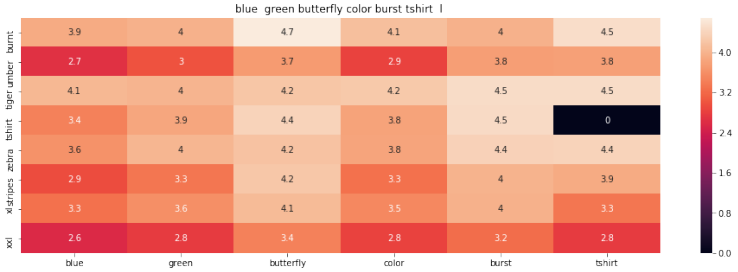
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 1.334570471686637
==========



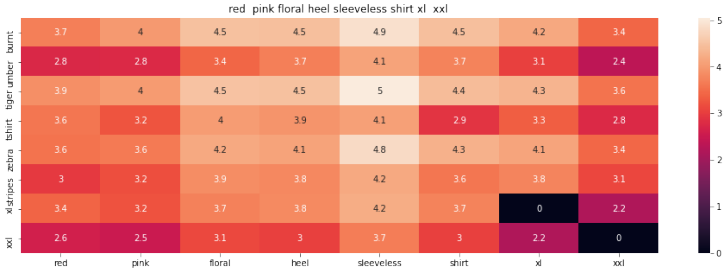
ASIN : B00JXQAX2C
Brand : Si Row
euclidean distance from input : 1.3538284304166988
==========



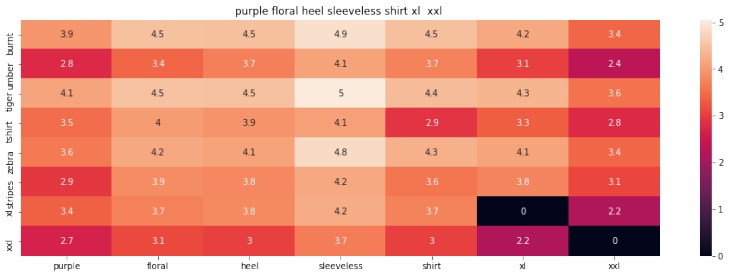
ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from input : 1.3576423647428706
==========



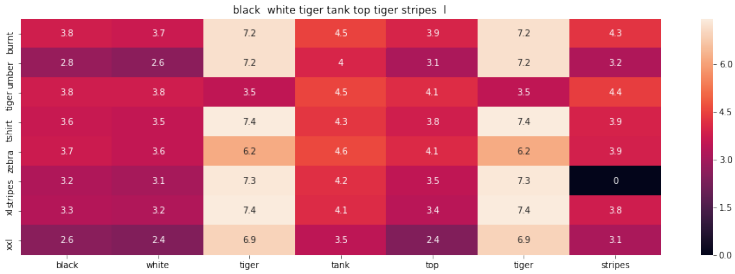
ASIN : B00JXQC0C8
Brand : Si Row
euclidean distance from input : 1.371363544705029
==========



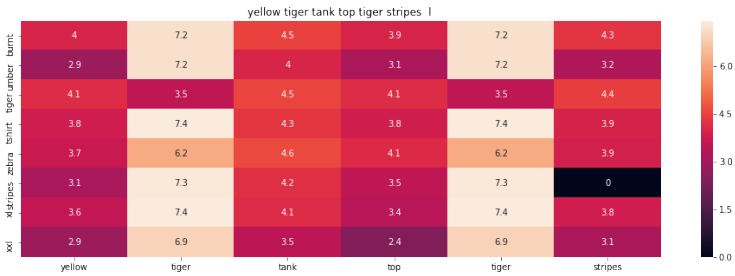
ASIN : B00JV63QQE
Brand : Si Row
euclidean distance from input : 1.3813977879708166
==========



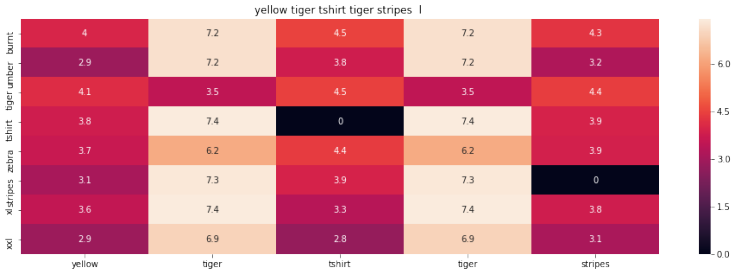
ASIN : B00JV63VC8
Brand : Si Row
euclidean distance from input : 1.4103313448454098
==========



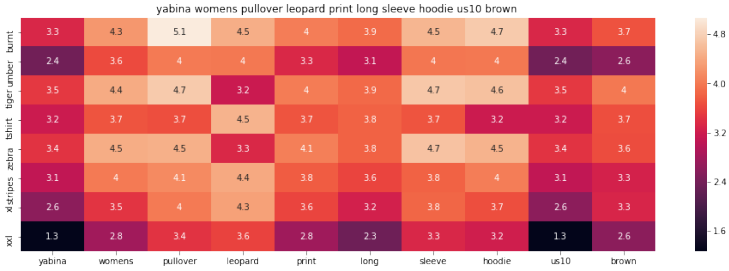
ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 1.472704696896191
==========



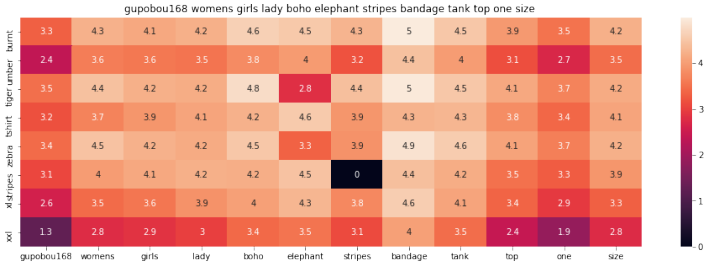
ASIN : B00JXQAUWA
Brand : Si Row
euclidean distance from input : 1.5176572166308915
==========



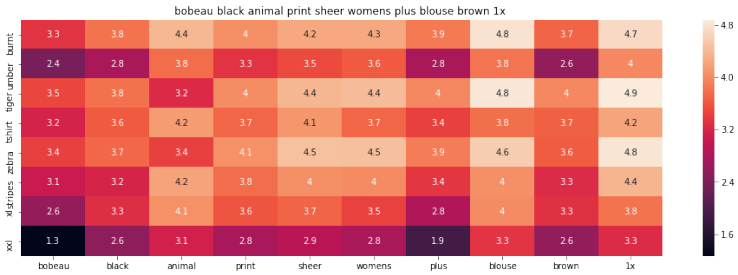
ASIN : B00JXQCUIC
Brand : Si Row
euclidean distance from input : 1.5603146237557288
==========



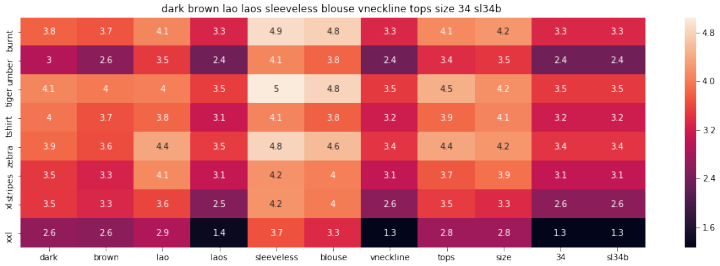
ASIN : B01KJUM6JI
Brand : YABINA
euclidean distance from input : 1.9002739369550194
==========



ASIN : B01ER184O6
Brand : GuPoBoU168
euclidean distance from input : 1.902216285435244
==========



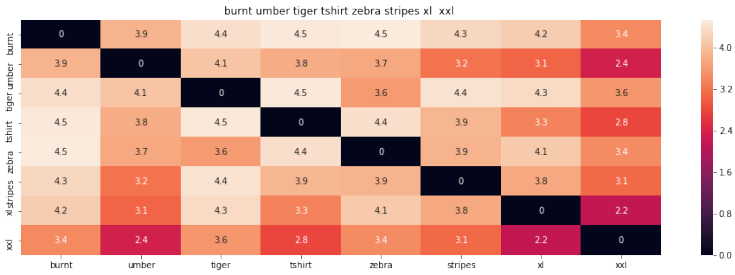
ASIN : B074MH886R
Brand : Bobeau
euclidean distance from input : 1.9072795013267643
==========



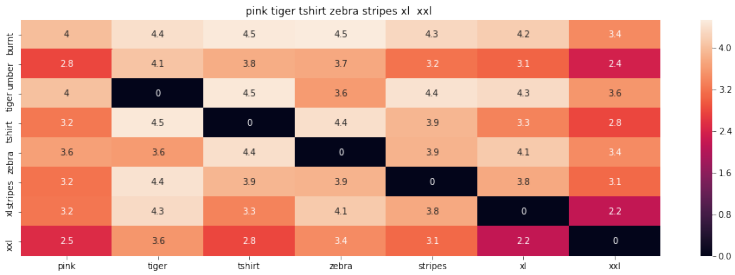
ASIN : B074J48RGW
Brand : Nanon
euclidean distance from input : 1.9110032816408917
==========

2.0.2 More Weight to Title

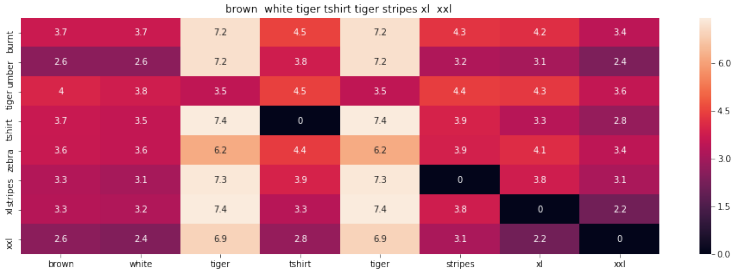
In [12]: idf_w2v_brand(12566, 20, 5, 5)



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

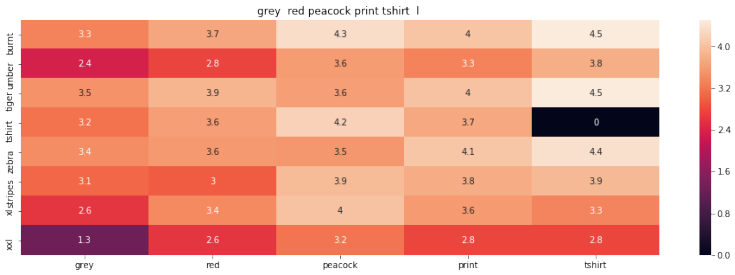


ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 0.7541969681508885
==========

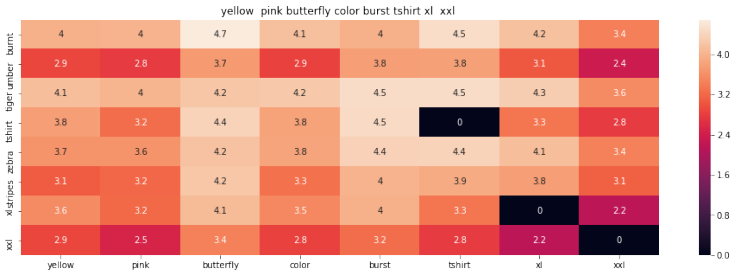


ASIN : B00JXQCWTO
Brand : Si Row

```
euclidean distance from input : 1.0446080017089843
==========================================================================================================
```



```
ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 1.1311253357656346
==========================================================================================================
```

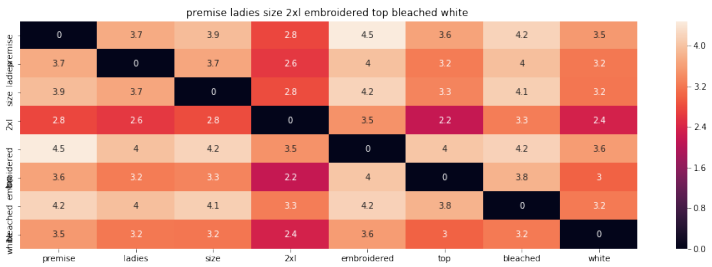


```
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 1.1812681580312596
==========================================================================================================
```

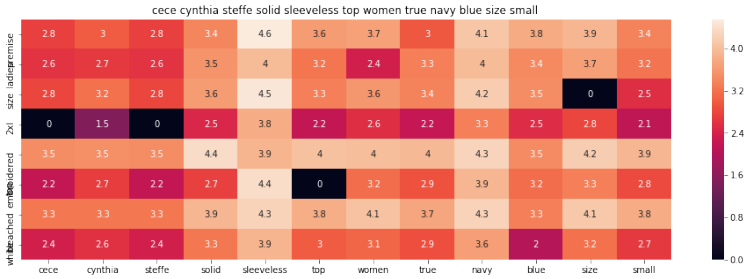
3 Weighted similarity using Title,brand ,color and Image

```
In [13]: 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
             # w3: weight for image

             # pairwise_dist will store the distance from given input apparel to all remaining
```

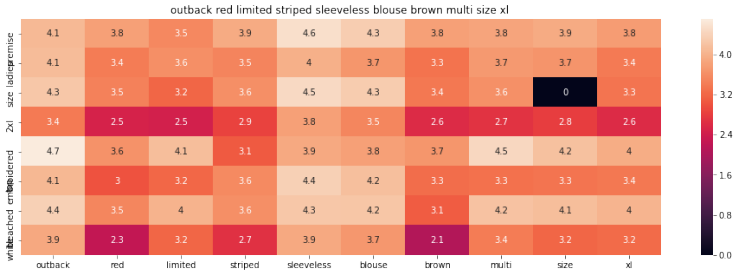
```
        # the metric we used here is cosine, the coside distance is mesured as K(X, Y) =
        # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
        idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].resh
        ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
        doc_id = asins.index(df_asins[doc_id])
        img_dist = pairwise_distances(img_data, img_data[doc_id].reshape(1,-1))
        pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist + w3 * img_dist)/float(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_indic
            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, 7, 5, 10, 10)
    # in the give heat map, each cell contains the euclidean distance between words i, j
```



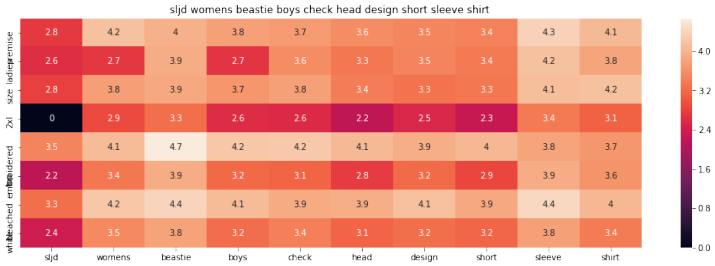
```
ASIN : B01M0IDUCV
Brand : Premise
euclidean distance from input : 1.079261541530839
=============================================================================================
```



ASIN : B01N4NQ7LX
Brand : CeCe by Cynthia Steffe
euclidean distance from input : 14.846791604686153
==========

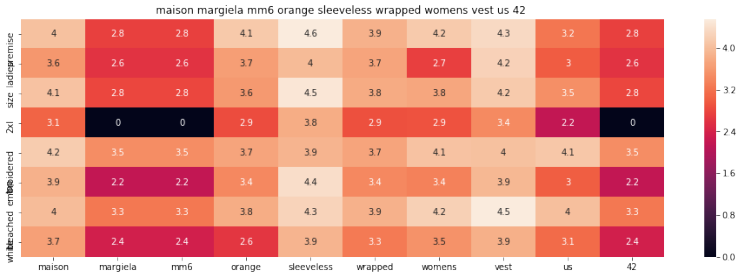


ASIN : B01IU645VU
Brand : Outback Red
euclidean distance from input : 19.769986700307427
==========

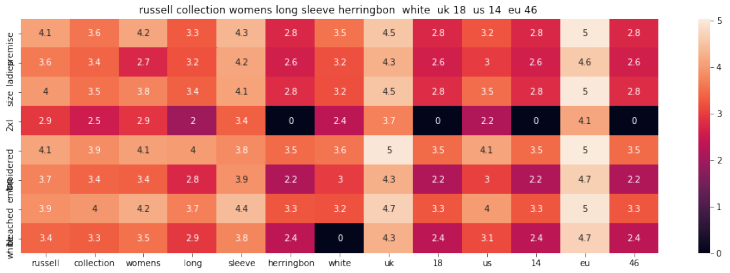


ASIN : B01FQLKKMK
Brand : SLJD

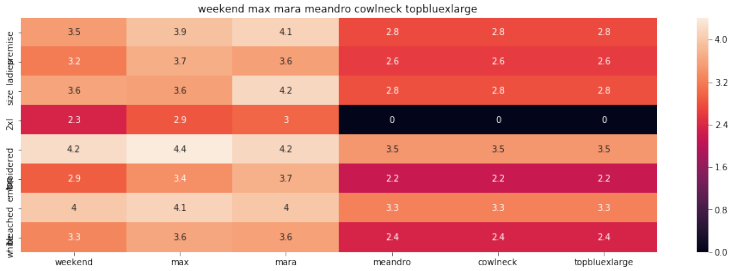
euclidean distance from input : 21.03167796714758



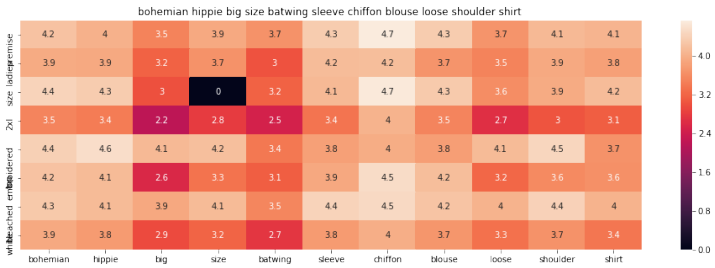
ASIN : B01MXI5L4G
Brand : Maison Margiela MM6
euclidean distance from input : 22.643057736483488



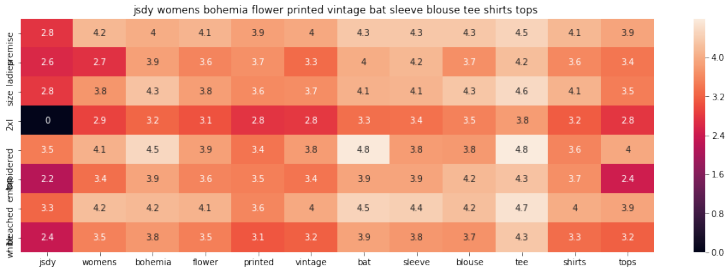
ASIN : B00K77AN5S
Brand : Russell Collection
euclidean distance from input : 22.736819961018533



ASIN : B01MG83UB4
Brand : MaxMara
euclidean distance from input : 22.840920042287408
==========



ASIN : B00YC92VRU
Brand : Display Promotion
euclidean distance from input : 22.92748453980161
==========

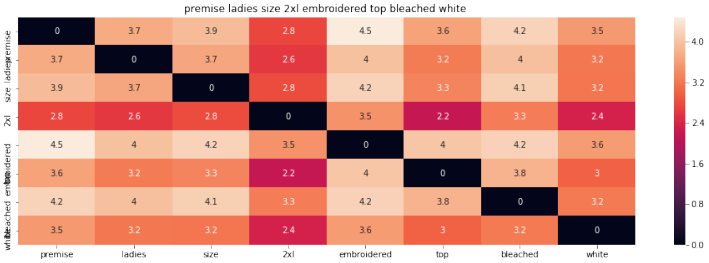


ASIN : B00L8RE3PC
Brand : JSDY-Cloth
euclidean distance from input : 22.97990924678236
==========

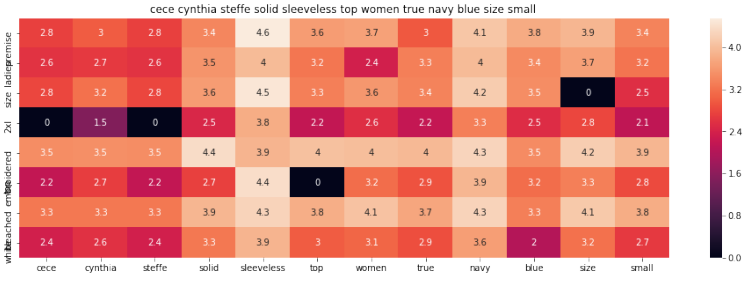
ASIN : B071LMW4YG
Brand : Standard James Perse
euclidean distance from input : 22.981727079911664
==========

3.0.1 More Weight to Image

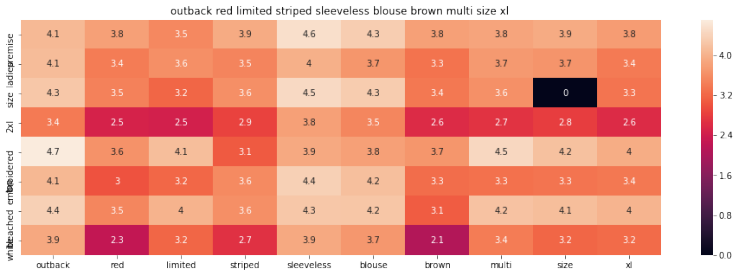
In [15]: idf_w2v_brand(12566, 7, 10, 10, 30)



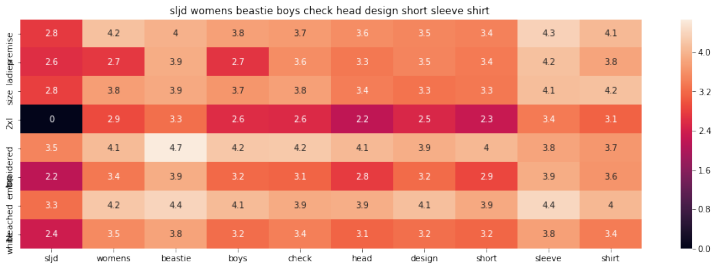
ASIN : B01M0IDUCV
Brand : Premise
euclidean distance from input : 1.4031810939781264
==========



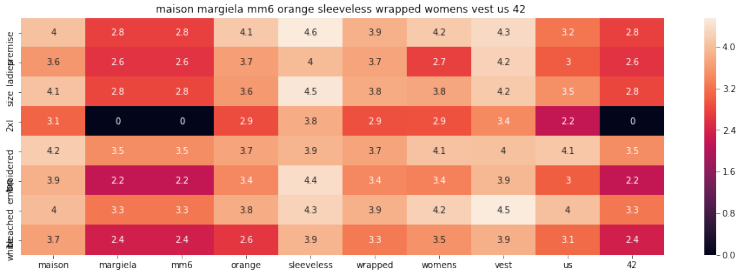
ASIN : B01N4NQ7LX
Brand : CeCe by Cynthia Steffe
euclidean distance from input : 12.711575527958235
==========



ASIN : B01IU645VU
Brand : Outback Red
euclidean distance from input : 16.598831998595788
==========

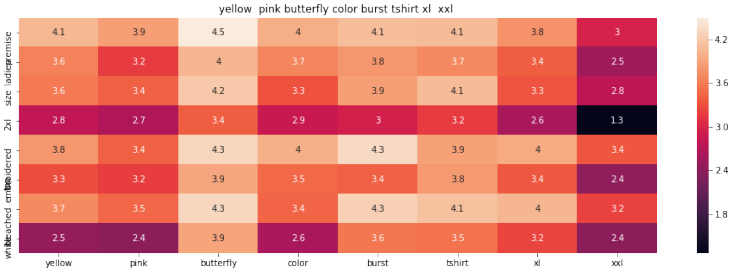


ASIN : B01FQLKKMK
Brand : SLJD
euclidean distance from input : 17.626876734539618
==========

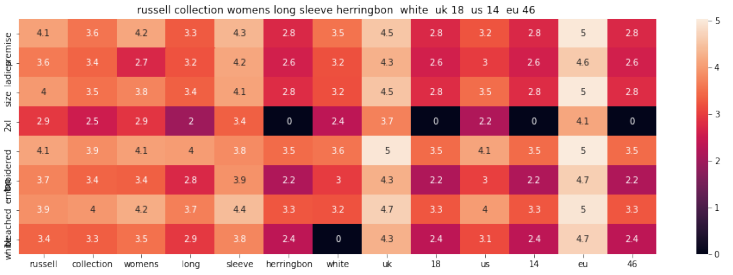


ASIN : B01MXI5L4G
Brand : Maison Margiela MM6

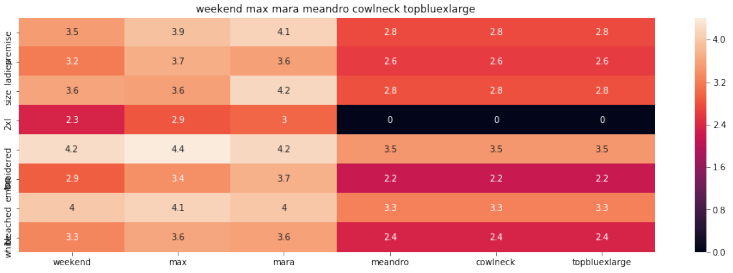
euclidean distance from input : 19.005454451949507



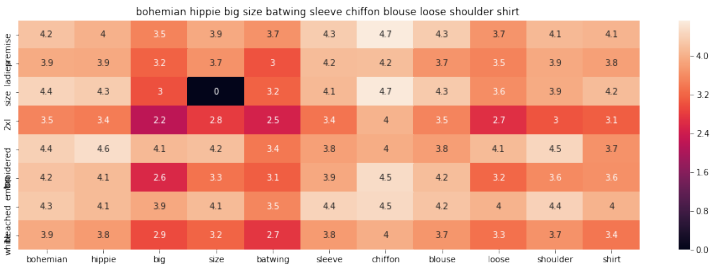
ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 19.00678853648819



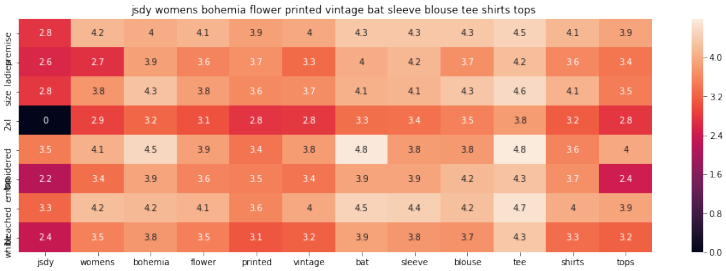
ASIN : B00K77AN5S
Brand : Russell Collection
euclidean distance from input : 19.050080546894023



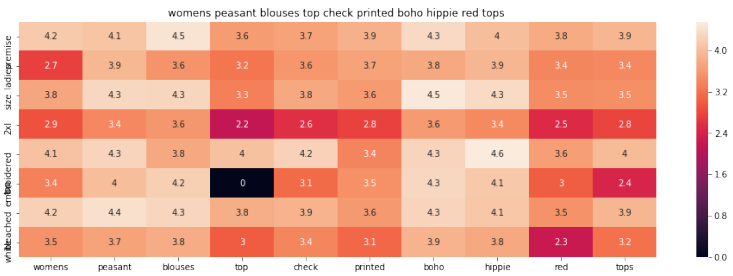
ASIN : B01MG83UB4
Brand : MaxMara
euclidean distance from input : 19.10107398094985
==========



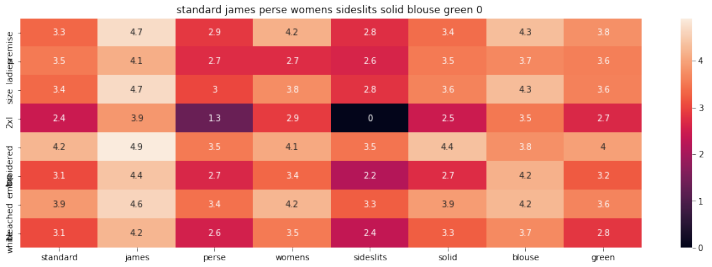
ASIN : B00YC92VRU
Brand : Display Promotion
euclidean distance from input : 19.17160801596142
==========



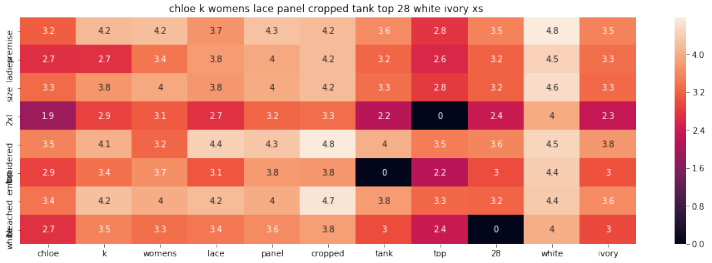
ASIN : B00L8RE3PC
Brand : JSDY-Cloth
euclidean distance from input : 19.248153298257144
==========



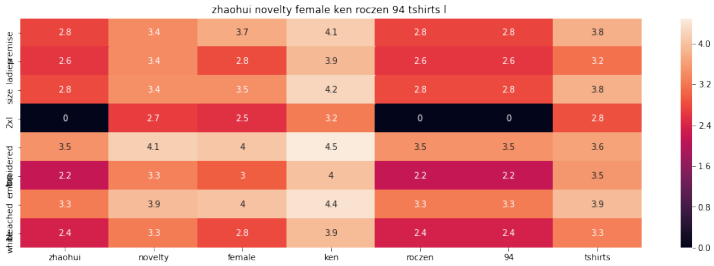
ASIN : B01MXMG6KB
Brand : Mogul Interior
euclidean distance from input : 19.25121356849595
==========



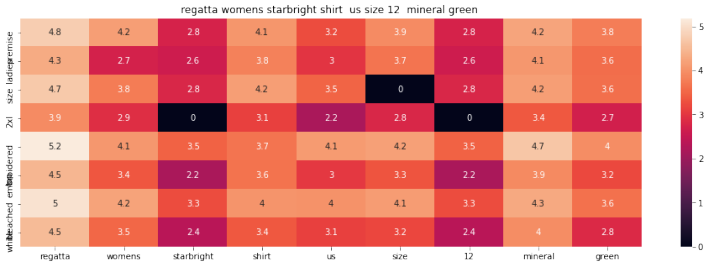
ASIN : B071LMW4YG
Brand : Standard James Perse
euclidean distance from input : 19.281407250298393
==========



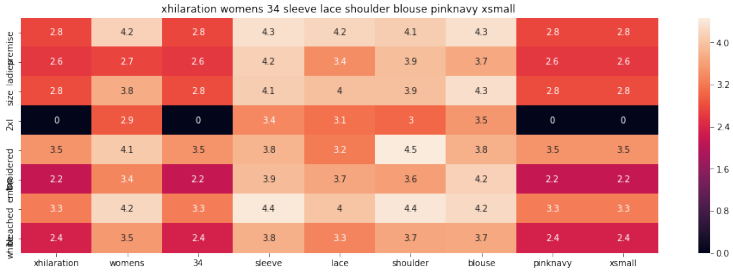
ASIN : B071VZCT5W
Brand : Chloe K.
euclidean distance from input : 19.289090192144133
==========



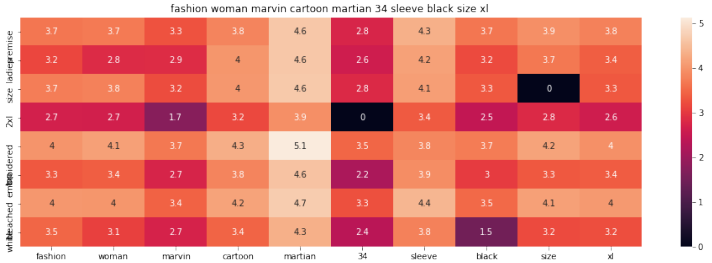
ASIN : B017EEVCV0
Brand : ZhaoHui
euclidean distance from input : 19.29022627468493
==========



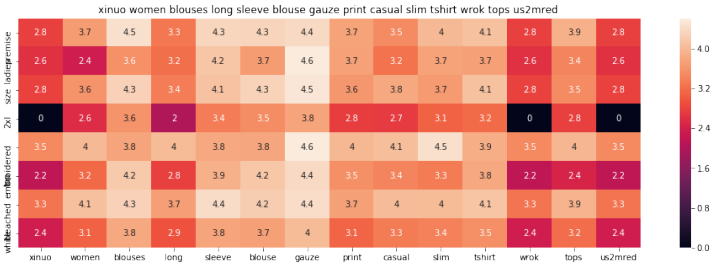
ASIN : B017EC8BOI
Brand : Regatta
euclidean distance from input : 19.29047068869904
==========



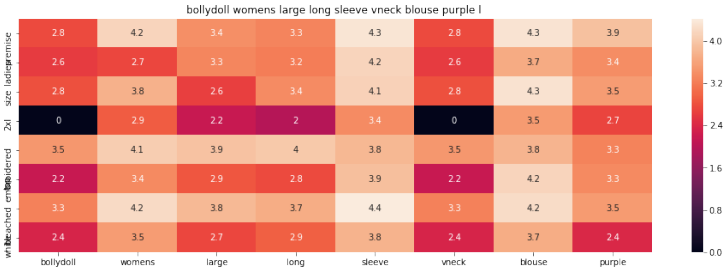
ASIN : B06ZYLKPRT
Brand : Xhilaration
euclidean distance from input : 19.297822952538194
==========



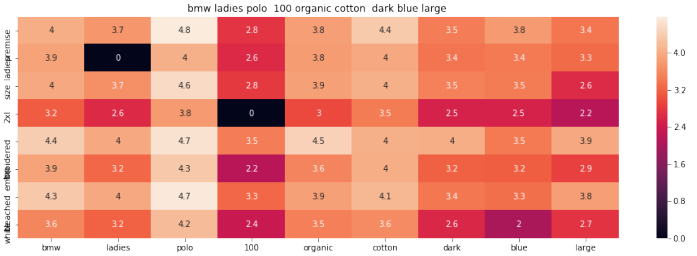
ASIN : B01FRPIX2Y
Brand : LOVELIF Sleeve Raglan
euclidean distance from input : 19.329345137984664
==========



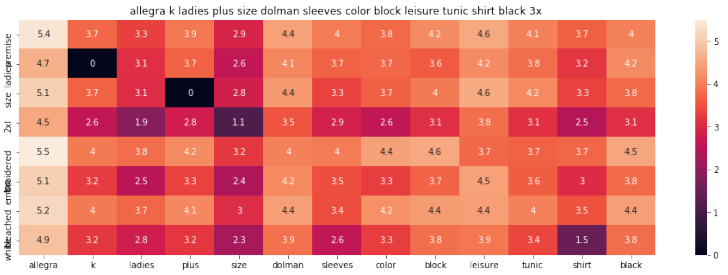
ASIN : B01MU874KK
Brand : XINUO
euclidean distance from input : 19.346327261306925
==========



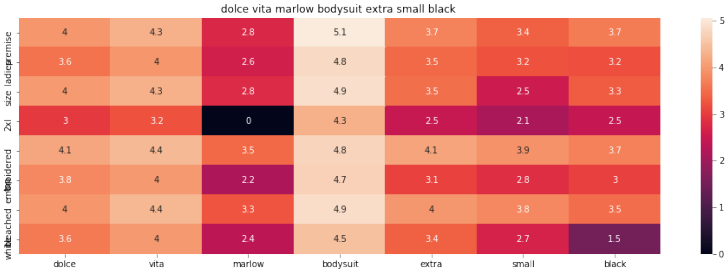
ASIN : B074MJPLCB
Brand : BollyDoll
euclidean distance from input : 19.35135690433038
==========



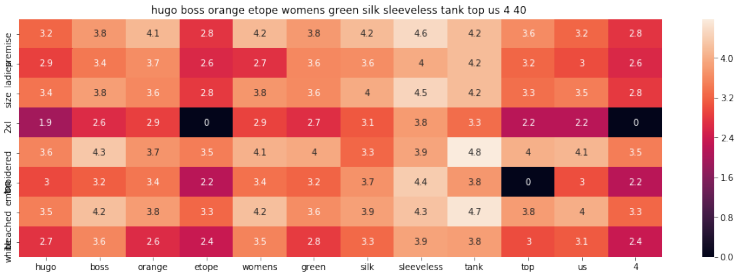
ASIN : B0060MKVX8
Brand : Not given
euclidean distance from input : 19.37092597423054
==========



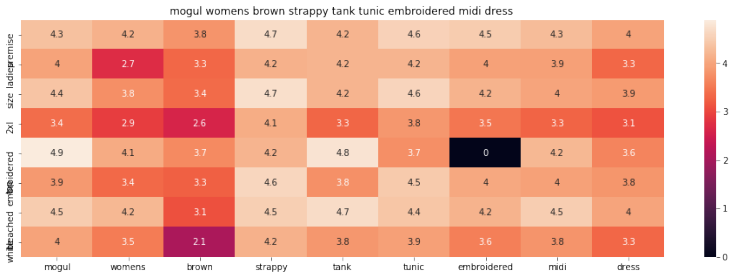
ASIN : B00D2J5HPO
Brand : Kaia
euclidean distance from input : 19.371627569817036
==========



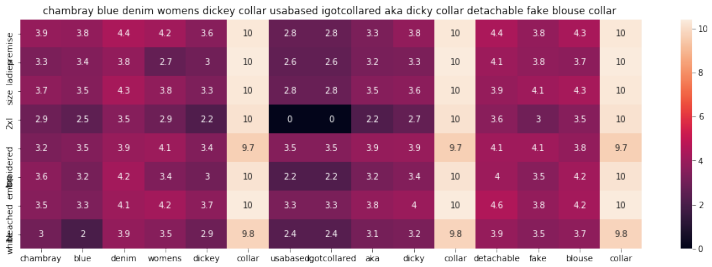
ASIN : B01N5K8RVX
Brand : Dolce Vita
euclidean distance from input : 19.425070727580692
==========



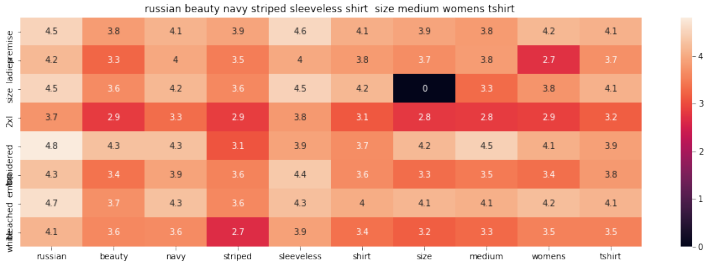
ASIN : B00WTE3XC2
Brand : HUGO BOSS
euclidean distance from input : 19.467913486607113
==========



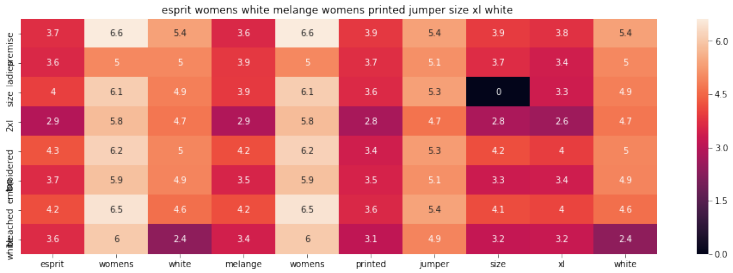
ASIN : B07191F766
Brand : Mogul Interior
euclidean distance from input : 19.499226837344917
==========



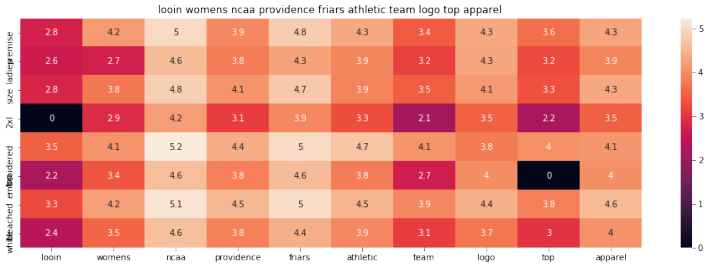
ASIN : B01MQWKWME
Brand : IGotCollared
euclidean distance from input : 19.49952877892388
==========



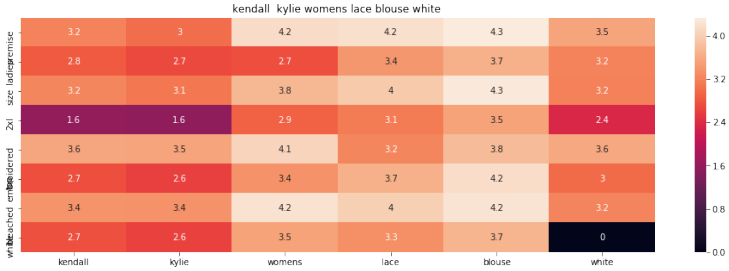
ASIN : B01LMFZKXS
Brand : Books.And.More
euclidean distance from input : 19.506712984155726
==========



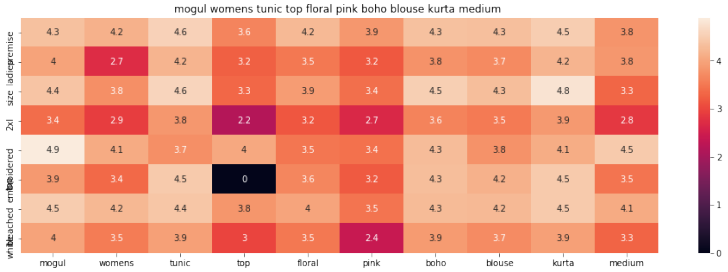
ASIN : B01MSZ1EO7
Brand : Esprit
euclidean distance from input : 19.512084431118435
==========



ASIN : B016XOMKA8
Brand : LOOIN Women Top
euclidean distance from input : 19.53490635200783
==========



ASIN : B071KG15YM
Brand : KENDALL + KYLIE
euclidean distance from input : 19.53784306870844
==========



ASIN : B06XZ38GTV
Brand : Mogul Interior
euclidean distance from input : 19.53916701590851
==========

3.0.2 Conclusion

1.We can see when we gave more importance to Brand ,the more recommendations are come from same Brand.Same like Image and Color

2.When we have tried with different weight of image we are not slightly different not much different product with change in weight of the image