In [2]:
```
import PIL.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
import pickle
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
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
from IPython.display import display, Image, SVG, Math, YouTubeVideo

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

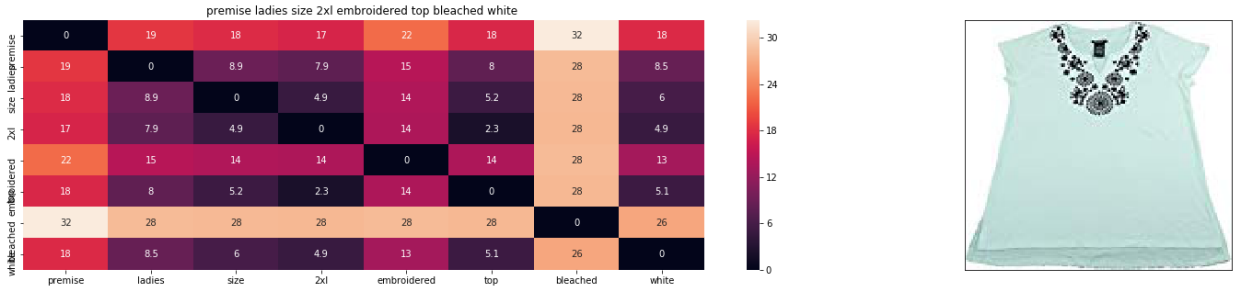
```
Using TensorFlow backend.
```

In [4]:
```
#Word2Vec
with open('word2vec_model', 'rb') as handle:
    model = pickle.load(handle)

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

# Load the original 16K dataset
data = pd.read_pickle('pickels/16k_apperal_data_preprocessed')
df_asins = list(data['asin'])


# 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]


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

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()

# idf_title_features.shape = #data_points * #words_in_corpus
# CountVectorizer().fit_transform(courpus) returns the a sparase matrix of dimensions #data_points * #words_in_corp
us
# idf_title_features[doc_id, index_of_word_in_corpus] = number of times the word occured in that doc

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

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


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

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

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

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


# Utility functions

def get_word_vec(sentence, doc_id, m_name):
    # sentence : title of the apparel
    # doc_id: document id in our corpus
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)
    vec = []
    for i in sentence.split():
        if i in vocab:
            if m_name == 'weighted' and i in  idf_title_vectorizer.vocabulary_:
                vec.append(idf_title_features[doc_id, idf_title_vectorizer.vocabulary_[i]] * model[i])
            elif m_name == 'avg':
                vec.append(model[i])
        else:
            # if the word in our courpus is not there in the google word2vec corpus, 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/avg) 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 i
n give title
    # vec2 = np.array(#number_of_words_title2 * 300), each row is a vector of length 300 corresponds to each word i
n give title

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


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



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

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

    #s1_vec = np.array(#number_of_words_title1 * 300), each row is a vector(weighted/avg) 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 featu
res
                [data['asin'].loc[df_id2],brands[doc_id2], colors[doc_id2], types[doc_id2]]] # recommonded 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])
    # 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([])
```

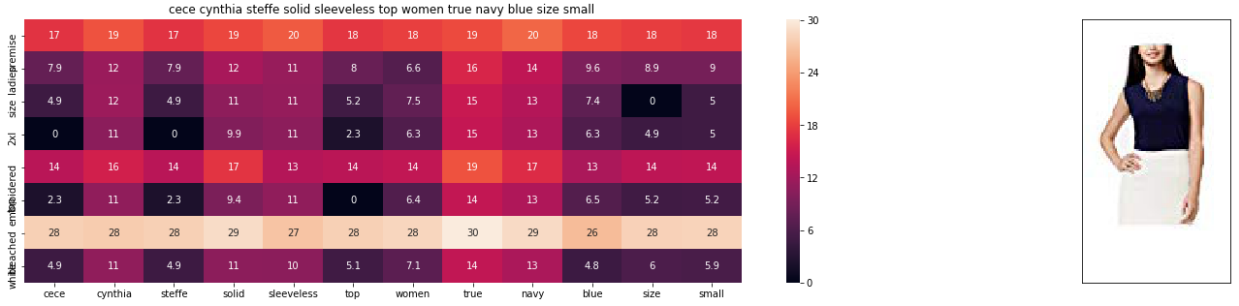
```
    ax2.set_yticks([])

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

    plt.show()




# 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):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vector, its values = 300
    # m_name: model information it will take two values
        # if  m_name == 'avg', we will append the model[i], w2v representation of word i
        # if m_name == 'weighted', we will multiply each w2v[word] with the idf(word)

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

    for word in sentence.split():
        nwords += 1
        if word in vocab:
            if m_name == 'weighted' and word in  idf_title_vectorizer.vocabulary_:
                featureVec = np.add(featureVec, idf_title_features[doc_id, idf_title_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


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)


def idf_w2v_comp(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) = <X, Y> / (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/metrics.html#cosine-similarity
    doc_id = asins.index(df_asins[doc_id])
    idf_w2v_dist  = pairwise_distances(w2v_title_weight, w2v_title_weight[doc_id].reshape(1,-1))
    ex_feat_dist = pairwise_distances(extra_features, extra_features[doc_id])
    img_feat_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,-1))

    pairwise_dist   = (w1 * idf_w2v_dist +  w2 * ex_feat_dist + w3 * img_feat_dist)/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 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_ur
l'].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('Amazon Url: www.amzon.com/dp/'+ asins[indices[i]])
        print('='*125)

idf_w2v_comp(12566, 5, 5, 10, 20)
# in the give heat map, each cell contains the euclidean distance between words i, j
```

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01M0IDUCV	Premise	Bleached-White	SHIRT



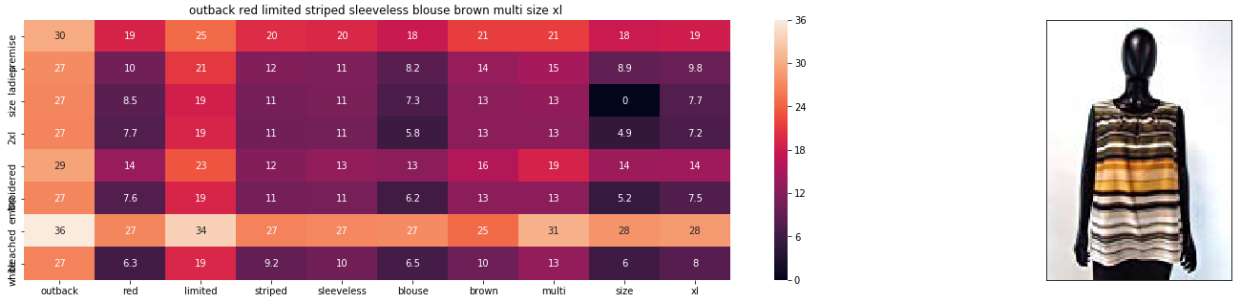
ASIN : B01M0IDUCV
Brand : Premise
euclidean distance from input : 0.03125
Amazon Url: www.amzon.com/dp/B00JXQB5FQ
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01N4NQ7LX	CeCe-by-Cynthia-Steffe	True-Navy	SHIRT



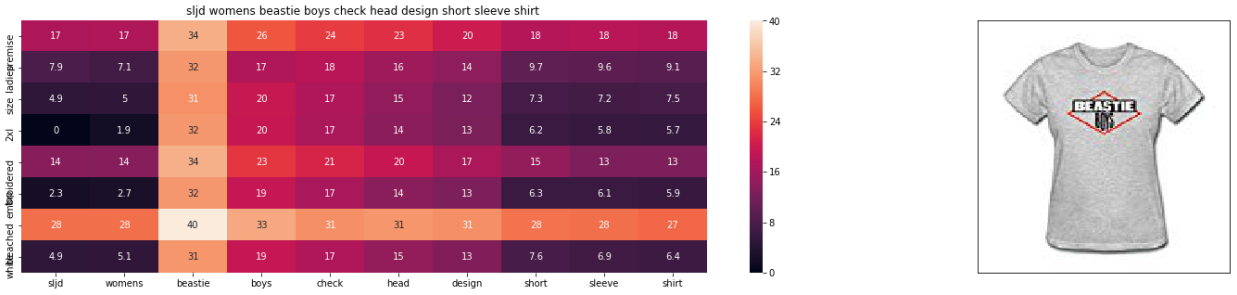
ASIN : B01N4NQ7LX
Brand : CeCe by Cynthia Steffe
euclidean distance from input : 16.875419998168944
Amazon Url: www.amzon.com/dp/B00JXQASS6
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01IU645VU	Outback-Red	Brown-Stripe	SHIRT



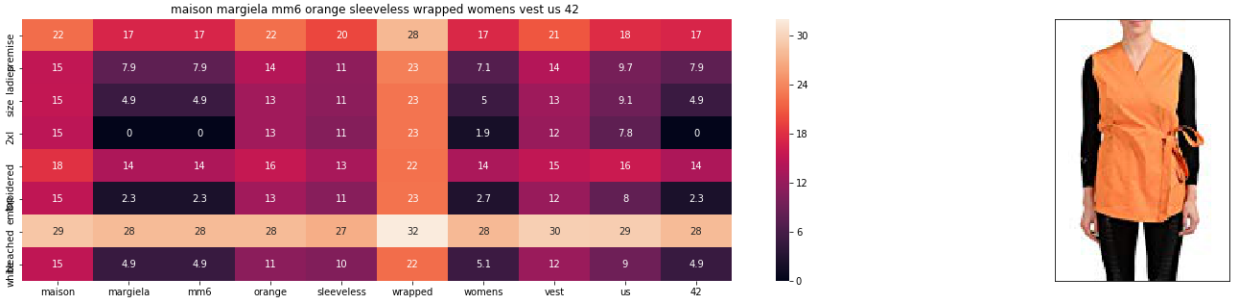
ASIN : B01IU645VU
Brand : Outback Red
euclidean distance from input : 22.643414813697543
Amazon Url: www.amzon.com/dp/B00JXQCUIC
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01FQLKKMK	SLJD	Grey	BOOKS_1973_AND_L



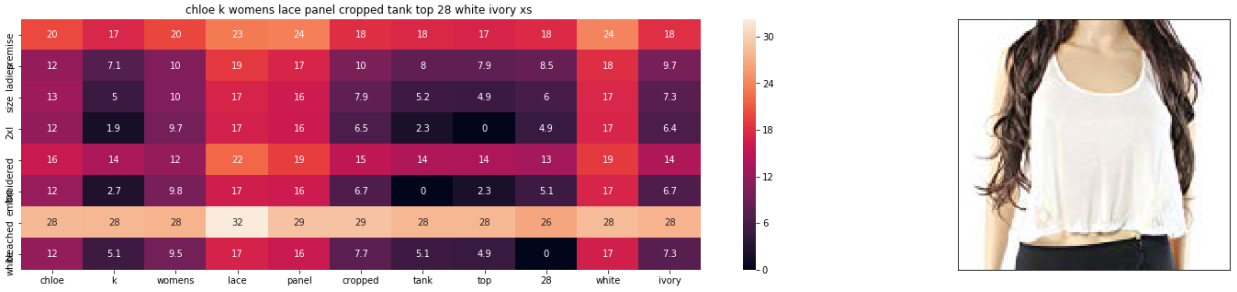
ASIN : B01FQLKKMK
Brand : SLJD
euclidean distance from input : 24.22756350678226
Amazon Url: www.amzon.com/dp/B00JXQCWTO
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01MXI5L4G	Maison-Margiela-MM6	Orange	SHIRT



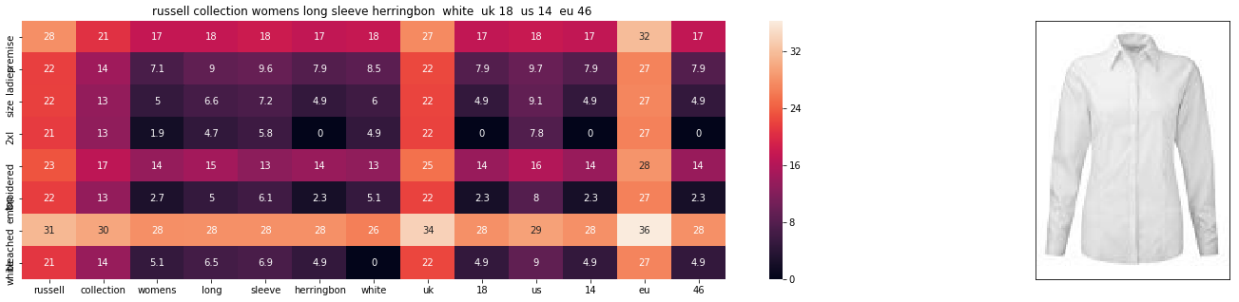
ASIN : B01MXI5L4G
Brand : Maison Margiela MM6
euclidean distance from input : 25.563327247321812
Amazon Url: www.amzon.com/dp/B071FCWD97
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B071VZCT5W	Chloe-K.	White-Ivory	SHIRT



ASIN : B071VZCT5W
Brand : Chloe K.
euclidean distance from input : 25.859312534332275
Amazon Url: www.amzon.com/dp/B01GXAZTRY
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B00K77AN5S	Russell-Collection	White	SHIRT

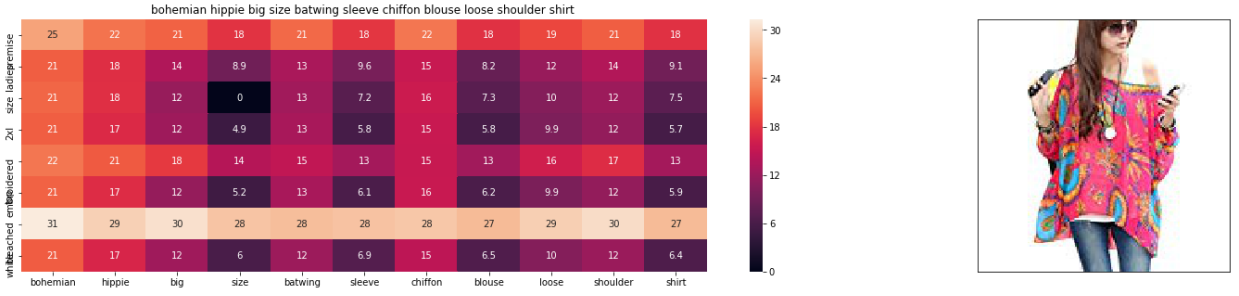


ASIN : B00K77AN5S
Brand : Russell Collection
euclidean distance from input : 25.888132190704347
Amazon Url: www.amzon.com/dp/B01JUNHBRM
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B00L8RE3PC	JSDY-Cloth	White	SHIRT

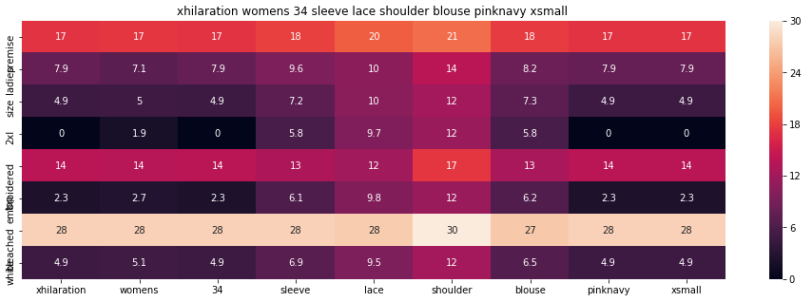
ASIN : B00L8RE3PC
Brand : JSDY-Cloth
euclidean distance from input : 25.893374538421632
Amazon Url: www.amzon.com/dp/B00JV63QQE
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B00YC92VRU	Display-Promotion	R	SHIRT



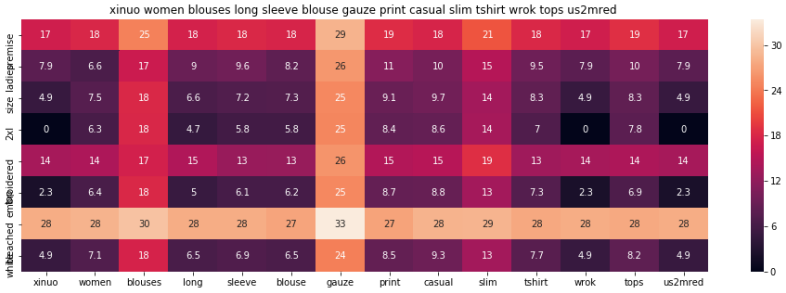
ASIN : B00YC92VRU
Brand : Display Promotion
euclidean distance from input : 25.992698577779485
Amazon Url: www.amzon.com/dp/B01CUPYBM0
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B06ZYLKPRT	Xhilaration	,-Pink/Navy,	SHIRT



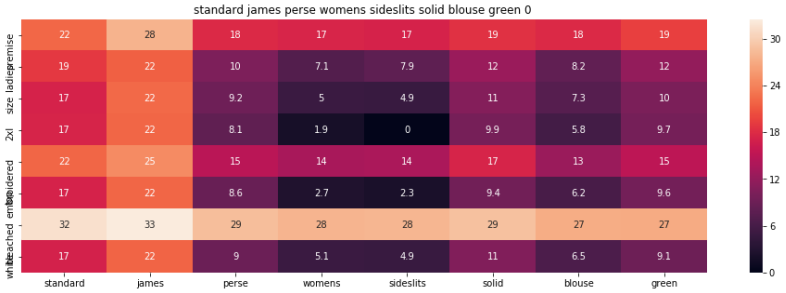
ASIN : B06ZYLKPRT
Brand : Xhilaration
euclidean distance from input : 26.010217036373284
Amazon Url: www.amzon.com/dp/B01CR57YY0
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01MU874KK	XINUO	Red	SHIRT



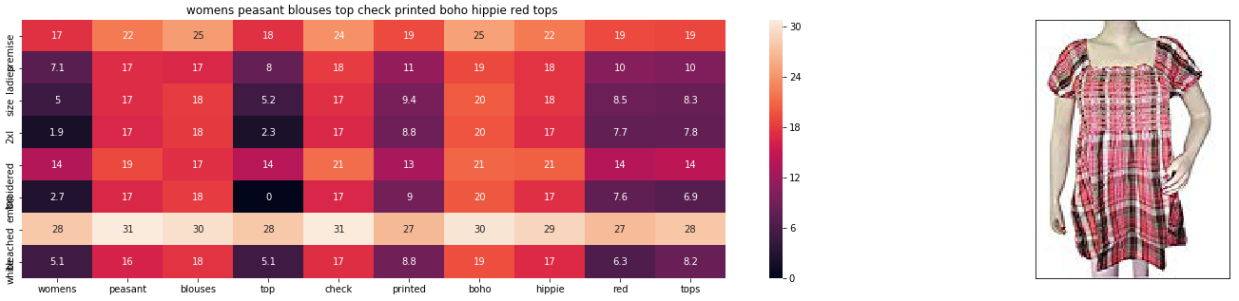
ASIN : B01MU874KK
Brand : XINUO
euclidean distance from input : 26.035236076253607
Amazon Url: www.amzon.com/dp/B01JQ096HW
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B071LMW4YG	Standard-James-Perse	Green	SHIRT



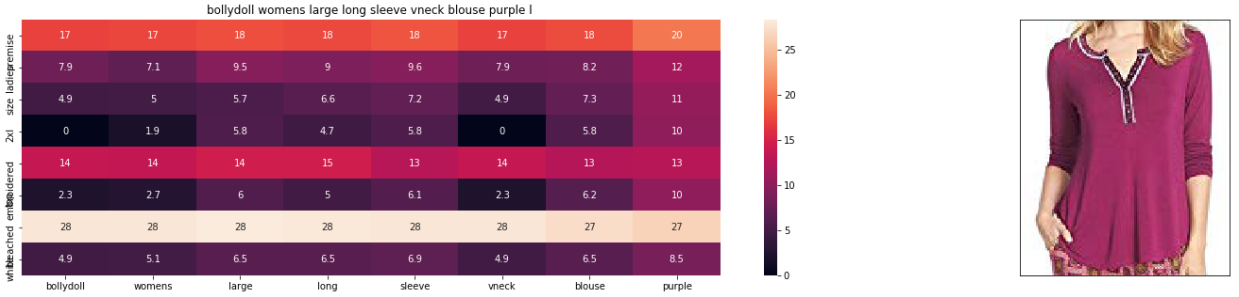
ASIN : B071LMW4YG
Brand : Standard James Perse
euclidean distance from input : 26.055414802253456
Amazon Url: www.amzon.com/dp/B01F7PHXY8
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01MXMG6KB	Mogul-Interior	Red	SHIRT



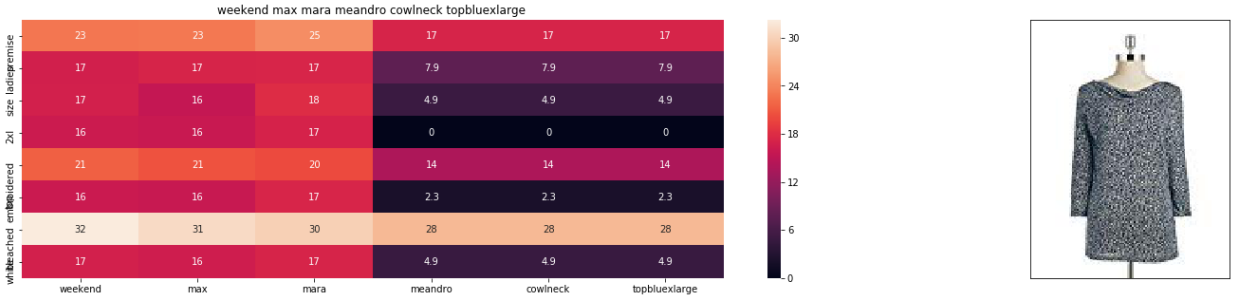
ASIN : B01MXMG6KB
Brand : Mogul Interior
euclidean distance from input : 26.060955180294183
Amazon Url: www.amzon.com/dp/B0177DM70S
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B074MJPLCB	BollyDoll	Purple	SHIRT



ASIN : B074MJPLCB
Brand : BollyDoll
euclidean distance from input : 26.07773113622065
Amazon Url: www.amzon.com/dp/B015H3W9BM
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01MG83UB4	MaxMara	Blue	SHIRT

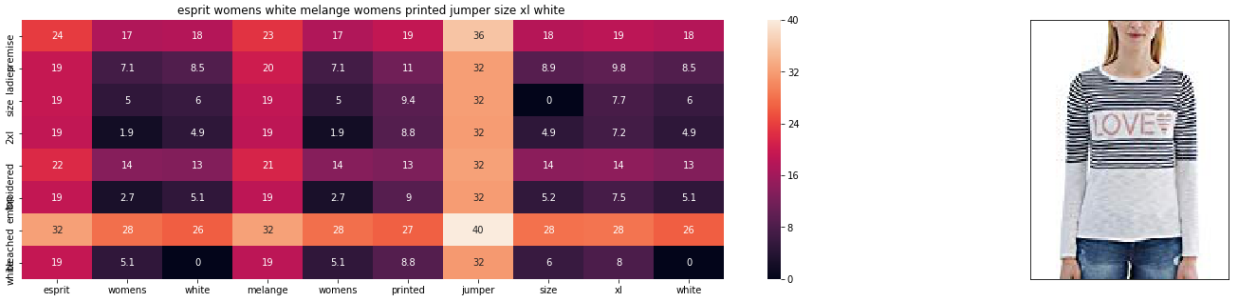


ASIN : B01MG83UB4
Brand : MaxMara
euclidean distance from input : 26.13668079747553
Amazon Url: www.amzon.com/dp/B071SBCY9W
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01MQWKWME	IGotCollared	Chambray-Blue-Denim	SHIRT

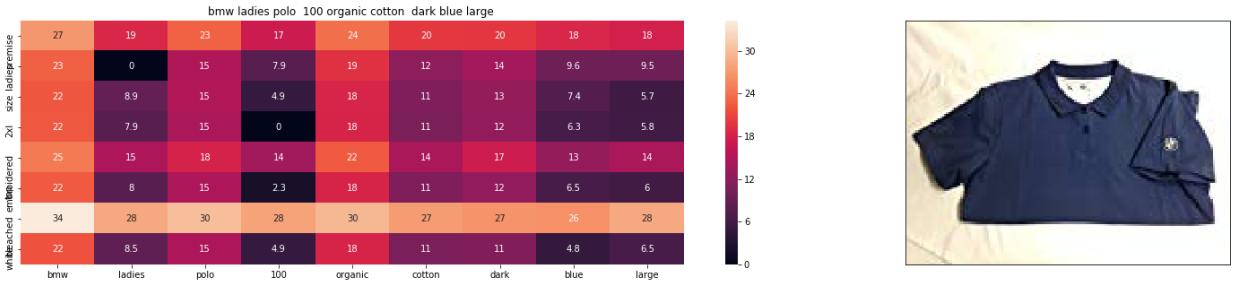
ASIN : B01MQWKWME
Brand : IGotCollared
euclidean distance from input : 26.206554729163855
Amazon Url: www.amzon.com/dp/B01I80A93G
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B01MSZ1EO7	Esprit	110-Off-White	SWEATER



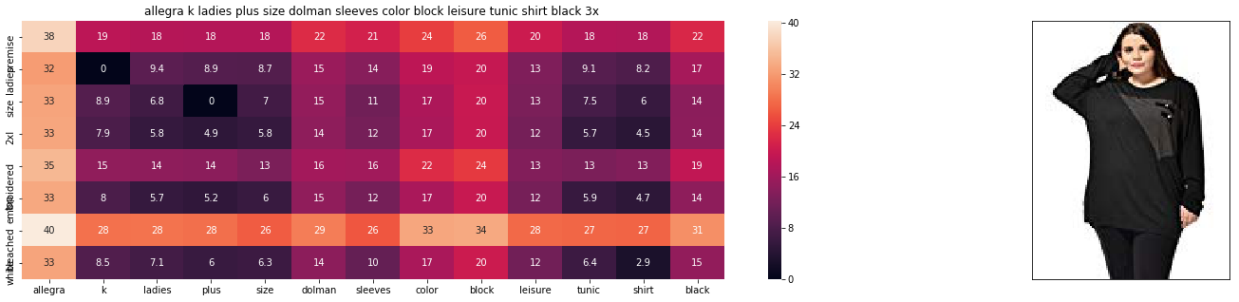
ASIN : B01MSZ1EO7
Brand : Esprit
euclidean distance from input : 26.271313220679968
Amazon Url: www.amzon.com/dp/B010NN9RXO
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B0060MKVX8	Not-given	Blue	AUTO_ACCESSORY



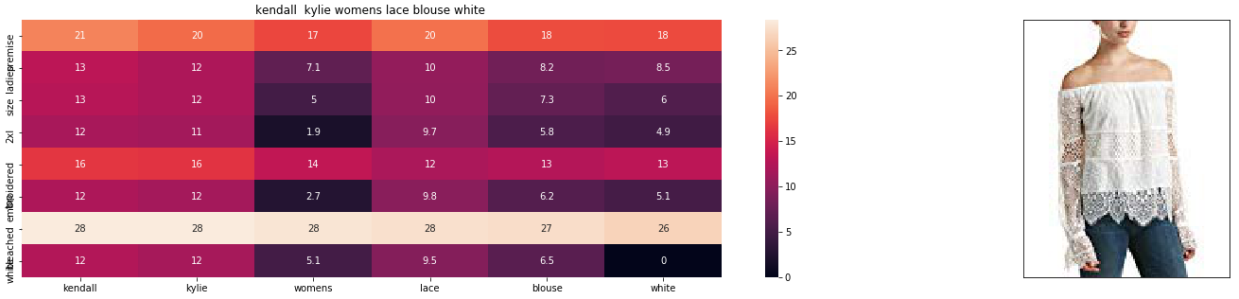
ASIN : B0060MKVX8
Brand : Not given
euclidean distance from input : 26.28657417315432
Amazon Url: www.amzon.com/dp/B0734GRKZL
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B00D2J5HPO	Kaia	Black	SHIRT



ASIN : B00D2J5HPO
Brand : Kaia
euclidean distance from input : 26.305787757772162
Amazon Url: www.amzon.com/dp/B016EXUZC4
==========

Asin	Brand	Color	Product type
B01M0IDUCV	Premise	Bleached-White	SHIRT
B071KG15YM	KENDALL-+-KYLIE	Bwt	SHIRT



ASIN : B071KG15YM
Brand : KENDALL + KYLIE
euclidean distance from input : 26.331705321437983
Amazon Url: www.amzon.com/dp/B071P4YKH5
==========

In [ ]: