Amazon Apparel Recommendations

[4.3] Overview of the data

In [1]:

```
#import all the necessary packages.

from PIL import Image
# 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
from sklearn.feature_extraction.text import Cou
ntVectorizer
from sklearn.feature_extraction.text import Tfi
```

```
dfVectorizer
from sklearn.metrics.pairwise import cosine_sim
ilarity
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=Tru
e)
warnings.filterwarnings("ignore")
```

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://acco
unts.google.com/o/oauth2/auth?client_id=9
47318989803-6bn6qk8qdgf4n4g3pfee6491hc0br
c4i.apps.googleusercontent.com&redirect_u
ri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&re
sponse_type=code&scope=email%20https%3a%2
f%2fwww.googleapis.com%2fauth%2fdocs.tes
t%20https%3a%2f%2fwww.googleapis.com%2fau
th%2fdrive%20https%3a%2f%2fwww.googleapi
s.com%2fauth%2fdrive.photos.readonly%20ht
tps%3a%2f%2fwww.googleapis.com%2fauth%2fp
eopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

In [0]:

```
# we have give a json file which consists of al
```

```
l information about
# the products
# loading the data using pandas' read_json fil
e.
data = pd.read_json('/content/drive/My Drive/Ap
plied_AI_Workshop_Code_Data/tops_fashion.json')
```

In [4]:
```
print ('Number of data points : ', data.shape[0
], \
       'Number of features/variables:', data.sh
ape[1])
```

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

Terminology:

What is a dataset?

Rows and columns

Data-point

Feature/variable

In [5]:
```
# each product/item has 19 features in the raw
 dataset.
data.columns # prints column-names or feature-n
ames.
```

Out[5]:
```
Index(['sku', 'asin', 'product_type_nam
e', 'formatted_price', 'author',
       'color', 'brand', 'publisher', 'av
ailability', 'reviews',
       'large_image_url', 'availability_t
ype', 'small_image_url',
```

```
       'editorial_review', 'title', 'mode
l', 'medium_image_url',
       'manufacturer', 'editorial_reive
w'],
      dtype='object')
```

In [0]:
```
data = data[['asin', 'brand', 'color', 'medium_
image_url', 'product_type_name', 'title', 'form
atted_price']]
```

In [7]:
```
print ('Number of data points : ', data.shape[0
], \
       'Number of features:', data.shape[1])
data.head() # prints the top rows in the table.
```

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

Out[7]:

	asin	brand	color	me
0	B016I2TS4W	FNC7C	None	https: image amaz
1	B01N49AI08	FIG Clothing	None	https: image amaz
2	B01JDPCOHO	FIG Clothing	None	https: image amaz

	asin	brand	color	me
3	B01N19U5H5	Focal18	None	https: image amaz
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https: image amaz

[5.1] Missing data for various features.

Basic stats for the feature: product_type_name

In [8]:
```
# We have total 72 unique type of product_type_
names
print(data['product_type_name'].describe())

# 91.62% (167794/183138) of the products are sh
irts,
```

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

In [9]:
```
# names of different product types
```

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

```
['SHIRT' 'SWEATER' 'APPAREL' 'OUTDOOR_REC
REATION_PRODUCT'
 'BOOKS_1973_AND_LATER' 'PANTS' 'HAT' 'SP
ORTING_GOODS' 'DRESS' 'UNDERWEAR'
 'SKIRT' 'OUTERWEAR' 'BRA' 'ACCESSORY' 'A
RT_SUPPLIES' 'SLEEPWEAR'
 'ORCA_SHIRT' 'HANDBAG' 'PET_SUPPLIES' 'S
HOES' 'KITCHEN' 'ADULT_COSTUME'
 'HOME_BED_AND_BATH' 'MISC_OTHER' 'BLAZE
R' 'HEALTH_PERSONAL_CARE'
 'TOYS_AND_GAMES' 'SWIMWEAR' 'CONSUMER_EL
ECTRONICS' 'SHORTS' 'HOME'
 'AUTO_PART' 'OFFICE_PRODUCTS' 'ETHNIC_WE
AR' 'BEAUTY'
 'INSTRUMENT_PARTS_AND_ACCESSORIES' 'POWE
RSPORTS_PROTECTIVE_GEAR' 'SHIRTS'
 'ABIS_APPAREL' 'AUTO_ACCESSORY' 'NONAPPA
RELMISC' 'TOOLS' 'BABY_PRODUCT'
 'SOCKSHOSIERY' 'POWERSPORTS_RIDING_SHIR
T' 'EYEWEAR' 'SUIT'
 'OUTDOOR_LIVING' 'POWERSPORTS_RIDING_JAC
KET' 'HARDWARE' 'SAFETY_SUPPLY'
 'ABIS_DVD' 'VIDEO_DVD' 'GOLF_CLUB' 'MUSI
C_POPULAR_VINYL'
 'HOME_FURNITURE_AND_DECOR' 'TABLET_COMPU
TER' 'GUILD_ACCESSORIES'
 'ABIS_SPORTS' 'ART_AND_CRAFT_SUPPLY' 'BA
G' 'MECHANICAL_COMPONENTS'
 'SOUND_AND_RECORDING_EQUIPMENT' 'COMPUTE
R_COMPONENT' 'JEWELRY'
 'BUILDING_MATERIAL' 'LUGGAGE' 'BABY_COST
UME' 'POWERSPORTS_VEHICLE_PART'

 'PROFESSIONAL_HEALTHCARE' 'SEEDS_AND_PLA
NTS' 'WIRELESS_ACCESSORY']
```

In [10]:
```
# find the 10 most frequent product_type_names.
product_type_count = Counter(list(data['product
_type_name']))
product_type_count.most_common(10)
```

Out[10]:
```
[('SHIRT', 167794),
 ('APPAREL', 3549),
 ('BOOKS_1973_AND_LATER', 3336),
 ('DRESS', 1584),
 ('SPORTING_GOODS', 1281),
 ('SWEATER', 837),
 ('OUTERWEAR', 796),
 ('OUTDOOR_RECREATION_PRODUCT', 729),
 ('ACCESSORY', 636),
 ('UNDERWEAR', 425)]
```

Basic stats for the feature: brand

In [11]:
```
# there are 10577 unique brands
print(data['brand'].describe())

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

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

In [12]:
```
brand_count = Counter(list(data['brand']))
brand_count.most_common(10)
```

Out[12]:
```
[('Zago', 223),
```

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

Basic stats for the feature: color

In [13]:
```
print(data['color'].describe())


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

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

In [14]:
```
color_count = Counter(list(data['color']))
color_count.most_common(10)
```

Out[14]:
```
[(None, 118182),
 ('Black', 13207),
 ('White', 8616),
 ('Blue', 3570),

 ('Red', 2289),
```

```
 ('Pink', 1842),
 ('Grey', 1499),
 ('*', 1388),
 ('Green', 1258),
 ('Multi', 1203)]
```

Basic stats for the feature: formatted_price

In [15]:
```
print(data['formatted_price'].describe())

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

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

In [16]:
```
price_count = Counter(list(data['formatted_pric
e']))
price_count.most_common(10)
```

Out[16]:
```
[(None, 154743),
 ('$19.99', 945),
 ('$9.99', 749),
 ('$9.50', 601),
 ('$14.99', 472),
 ('$7.50', 463),
 ('$24.99', 414),
 ('$29.99', 370),
 ('$8.99', 343),
 ('$9.01', 336)]
```

Basic stats for the feature: title

In [17]:
```
print(data['title'].describe())
```

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

6. Text pre-processing

In [0]:
```
data = pd.read_pickle('/content/drive/My Drive/
Applied_AI_Workshop_Code_Data/pickels/16k_apper
al_data')
```

In [19]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords
to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopword
s.zip.
```

Out[19]:
```
True
```

In [20]:
```
# we use the list of stop words that are downlo
aded 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 revie
w like '"#$@!%^&*()_+-~?>< etc.
            word = ("".join(e for e in words if
e.isalnum()))
            # Conver all letters to lower-case
            word = word.lower()
            # stop-word removal
            if not word in stop_words:
                string += word + " "
        data[column][index] = string
```

```
list of stop words: {'has', 'yourselves',
'll', 'does', 'any', 'very', 'our', "wo
n't", 'mightn', 'my', 'against', "have
n't", 'only', 'here', 'am', "mightn't",
"you've", "couldn't", 'he', 'some', 'if',
'theirs', "mustn't", 'where', 'or', 'abou
t', 'ourselves', "weren't", 'him', 're',
'should', "you'll", 'wouldn', 'own', 'bel
ow', 'had', 'they', "she's", "needn't",
"aren't", 'no', 'y', 'when', "isn't", 's
o', 'over', 'off', "doesn't", 'weren', 'd
own', 'who', 'through', 'hasn', 'not', 'i
t', 'have', 'having', 'all', "you're", 'o
urs', 'were', 'your', "didn't", 'doing',
"shan't", "don't", 'again', "that'll", "s
hould've", 'than', 'isn', 'himself', 'the
re', 'same', 'them', 'yours', "wasn't",
'being', "you'd", 'an', 'because', 'of',
```

```
's', 'as', 'in', 'shouldn', 'won', 'is',
'didn', 'just', 'their', 'haven', 'thos
e', 'between', 'nor', 'i', 'his', 'ain',
'before', 'ma', 'what', 'its', 'while',
'until', 'that', 'you', 'me', 'both', 'ea
ch', 'we', 'such', 'other', 'hadn', 'whic
h', 'for', 'out', 'under', 'mustn', "i
t's", 'needn', 'the', "hasn't", 'why', 'a
bove', 'shan', 'be', 'been', 'yourself',
'but', 'doesn', 'this', 'myself', 'a', 'a
t', 'can', 'further', 'once', "should
n't", 'from', 'couldn', 'to', 'most', 'wa
s', 'how', 'don', 'wasn', 'm', "hadn't",
'on', 'after', 'during', 'by', 'o', 'sh
e', 'then', 'too', 'into', 'with', 't',
'whom', 'few', 'did', 'now', 'herself',
'hers', 'will', 'more', "wouldn't", 'd',
'aren', 'and', 'her', 'these', 'do', 'v
e', 'itself', 'are', 'up', 'themselves'}
```

In [21]:
```
start_time = time.clock()
# we take each title and we text-preprocess it.
for index, row in data.iterrows():
    nlp_preprocessing(row['title'], index, 'tit
le')
# we print the time it took to preprocess whole
titles
print(time.clock() - start_time, "seconds")
```

```
7.848568 seconds
```

In [22]:
```
data.shape
```

Out[22]:
```
(16042, 7)
```

In [23]:
```

```

```
data.head()
```

Out[23]:

	asin	brand	color	me
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https: image amaz
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https: image amaz
15	B003BSRPB0	FeatherLite	White	https: image amaz
27	B014ICEJ1Q	FNC7C	Purple	https: image amaz
46	B01NACPBG2	Fifth Degree	Black	https: image amaz

Stemming

In [24]:
```
from nltk.stem.porter import *
```

```
stemmer = PorterStemmer()
print(stemmer.stem('arguing'))
print(stemmer.stem('fishing'))


# We tried using stemming on our titles and it
 didnot work very well.
```

```
argu
fish
```

[8] Text based product similarity

In [25]:
```
data = pd.read_pickle('/content/drive/My Drive/
Applied_AI_Workshop_Code_Data/pickels/16k_apper
al_data_preprocessed')
data.head()
```

Out[25]:

	asin	brand	color	me
4	B004GSI2OS	FeatherLite	Onyx Black/ Stone	https: image amaz
6	B012YX2ZPI	HX-Kingdom Fashion T-shirts	White	https: image amaz

	asin	brand	color	me
15	B003BSRPB0	FeatherLite	White	https:/ image amaz
27	B014ICEJ1Q	FNC7C	Purple	https:/ image amaz
46	B01NACPBG2	Fifth Degree	Black	https:/ image amaz

In [0]:

```
# Utility Functions which we will use through t
he rest of the workshop.


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

#plotting code to understand the algorithm's de
cision.
def plot_heatmap(keys, values, labels, url, tex
t):
        # keys: list of words of recommended ti
tle
```

```
        # values: len(values) ==  len(keys), va
lues(i) represents the occurence of the word ke
ys(i)
        # labels: len(labels) == len(keys), the
values of labels depends on the model we are us
ing
                # if model == 'bag of words': l
abels(i) = values(i)
                # if model == 'tfidf weighted b
ag of words':labels(i) = tfidf(keys(i))
                # if model == 'idf weighted bag
of words':labels(i) = idf(keys(i))
        # url : apparel's url

        # we will devide the whole figure into
 two parts
        gs = gridspec.GridSpec(2, 2, width_rati
os=[4,1], height_ratios=[4,1])
        fig = plt.figure(figsize=(25,3))

        # 1st, ploting heat map that represents
the count of commonly ocurred words in title2
        ax = plt.subplot(gs[0])
        # it displays a cell in white color if
 the word is intersection(lis of words of title
1 and list of words of title2), in black if not
        ax = sns.heatmap(np.array([values]), an
not=np.array([labels]))
        ax.set_xticklabels(keys) # set that axi
s labels as the words of title
        ax.set_title(text) # apparel title

        # 2nd, plotting image of the the appare
l
        ax = plt.subplot(gs[1])
        # we don't want any grid lines for imag
```

```
e and no labels on x-axis and y-axis
        ax.grid(False)
        ax.set_xticks([])
        ax.set_yticks([])

        # we call dispaly_img based with parame
te url
        display_img(url, ax, fig)

        # displays combine figure ( heat map an
d image together)
        plt.show()

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

    # doc_id : index of the title1
    # vec1 : input apparels's vector, it is of
 a dict type {word:count}
    # vec2 : recommended apparels's vector, it
 is of a dict type {word:count}
    # url : apparels image url
    # text: title of recomonded apparel (used t
o keep title of image)
    # model, it can be any of the models,
        # 1. bag_of_words
        # 2. tfidf
        # 3. idf

    # we find the common words in both titles,
 because these only words contribute to the dis
tance between two title vec's
    intersection = set(vec1.keys()) & set(vec2.
keys())

    # we set the values of non intersecting wor
```

```
ds to zero, this is just to show the difference
in heatmap
    for i in vec2:
        if i not in intersection:
            vec2[i]=0

    # for labeling heatmap, keys contains list
 of all words in title2
    keys = list(vec2.keys())
    #  if ith word in intersection(lis of words
of title1 and list of words of title2): values
(i)=count of that word in title2 else values(i)
=0
    values = [vec2[x] for x in vec2.keys()]

    # labels: len(labels) == len(keys), the val
ues of labels depends on the model we are using
        # if model == 'bag of words': labels(i)
= values(i)
        # if model == 'tfidf weighted bag of wo
rds':labels(i) = tfidf(keys(i))
        # if model == 'idf weighted bag of word
s':labels(i) = idf(keys(i))

    if model == 'bag_of_words':
        labels = values
    elif model == 'tfidf':
        labels = []
        for x in vec2.keys():
            # tfidf_title_vectorizer.vocabulary
_ it contains all the words in the corpus
            # tfidf_title_features[doc_id, inde
x_of_word_in_corpus] will give the tfidf value
 of word in given document (doc_id)
            if x in  tfidf_title_vectorizer.voc
abulary_:
```

```
                labels.append(tfidf_title_featu
res[doc_id, tfidf_title_vectorizer.vocabulary_[
x]])
            else:
                labels.append(0)
    elif model == 'idf':
        labels = []
        for x in vec2.keys():
            # idf_title_vectorizer.vocabulary_
 it contains all the words in the corpus
            # idf_title_features[doc_id, index_
of_word_in_corpus] will give the idf value of w
ord in given document (doc_id)
            if x in  idf_title_vectorizer.vocab
ulary_:
                labels.append(idf_title_feature
s[doc_id, idf_title_vectorizer.vocabulary_[x]])
            else:
                labels.append(0)

    plot_heatmap(keys, values, labels, url, tex
t)


# this function gets a list of wrods along with
the frequency of each
# word given "text"
def text_to_vector(text):
    word = re.compile(r'\w+')
    words = word.findall(text)
    # words stores list of all words in given s
tring, you can try 'words = text.split()' this
 will also gives same result
    return Counter(words) # Counter counts the
 occurence of each word in list, it returns dic
t type object {word1:count}
```

```
def get_result(doc_id, content_a, content_b, ur
l, model):
    text1 = content_a
    text2 = content_b

    # vector1 = dict{word11:#count, word12:#cou
nt, etc.}
    vector1 = text_to_vector(text1)

    # vector1 = dict{word21:#count, word22:#cou
nt, etc.}
    vector2 = text_to_vector(text2)

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

[8.2] Bag of Words (BoW) on product titles

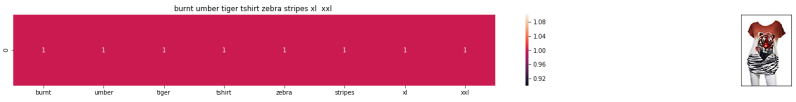
In [27]:
```
from sklearn.feature_extraction.text import Cou
ntVectorizer
title_vectorizer = CountVectorizer()
title_features   = title_vectorizer.fit_transfo
rm(data['title'])
title_features.get_shape() # get number of rows
and columns in feature matrix.
```

Out[27]:
```
(16042, 12609)
```

In [28]:
```
def bag_of_words_model(doc_id, num_results):
```

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

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    pairwise_dist = pairwise_distances(title_fe
atures,title_features[doc_id])

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

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

    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3.
title2, url, model
        get_result(indices[i],data['title'].loc
[df_indices[0]], data['title'].loc[df_indices[i
]], data['medium_image_url'].loc[df_indices[i
]], 'bag_of_words')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print ('Brand:', data['brand'].loc[df_i
ndices[i]])
        print ('Title:', data['title'].loc[df_i
```

```
ndices[i]])
        print ('Euclidean similarity with the q
uery image :', pdists[i])
        print('='*60)

#call the bag-of-words model for a product to g
et similar products.
bag_of_words_model(12566, 20) # change the inde
x if you want to.
# In the output heat map each value represents
 the count value
# of the label word, the color represents the i
ntersection
# with inputs title.

#try 12566
#try 931
```



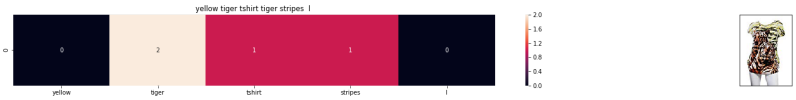
ASIN : B00JXQB5FQ
Brand: Si Row
Title: burnt umber tiger tshirt zebra str
ipes xl xxl
Euclidean similarity with the query image
: 0.0
==
==========

ASIN : B00JXQASS6
Brand: Si Row
Title: pink tiger tshirt zebra stripes xl

xxl

Euclidean similarity with the query image
: 1.7320508075688772
==========
==========

ASIN : B00JXQCWTO
Brand: Si Row
Title: brown white tiger tshirt tiger st
ripes xl xxl
Euclidean similarity with the query image
: 2.449489742783178
==========
==========



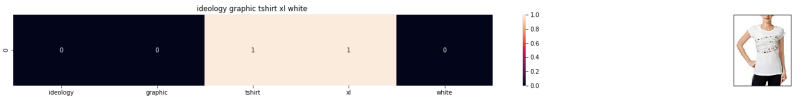
ASIN : B00JXQCUIC
Brand: Si Row
Title: yellow tiger tshirt tiger stripes
l
Euclidean similarity with the query image
: 2.6457513110645907
==========
==========

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

: 3.0

==========

==========



ASIN : B01NB0NKRO

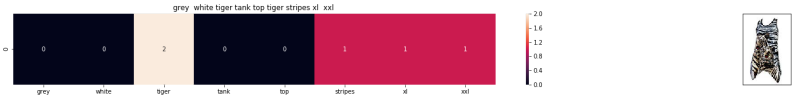
Brand: Ideology

Title: ideology graphic tshirt xl white

Euclidean similarity with the query image : 3.0

==========

==========



ASIN : B00JXQAFZ2

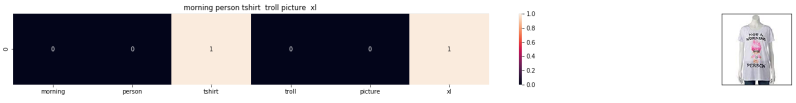
Brand: Si Row

Title: grey white tiger tank top tiger stripes xl xxl

Euclidean similarity with the query image : 3.0

==========

==========



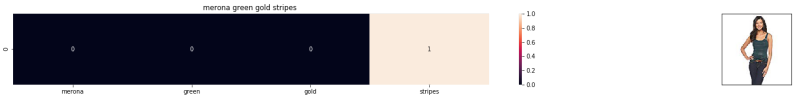
ASIN : B01CLS8LMW

Brand: Awake

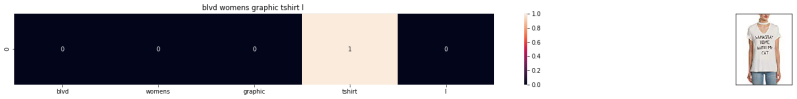
Title: morning person tshirt troll picture xl

Euclidean similarity with the query image : 3.1622776601683795

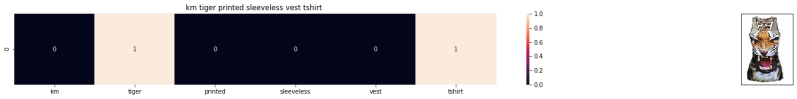
==========



ASIN : B01KVZUB6G
Brand: Merona
Title: merona green gold stripes
Euclidean similarity with the query image : 3.1622776601683795
==========

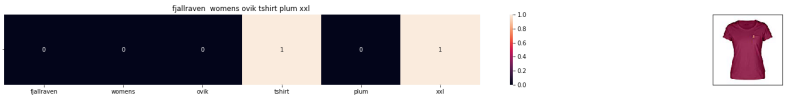


ASIN : B0733R2CJK
Brand: BLVD
Title: blvd womens graphic tshirt l
Euclidean similarity with the query image : 3.1622776601683795
==========



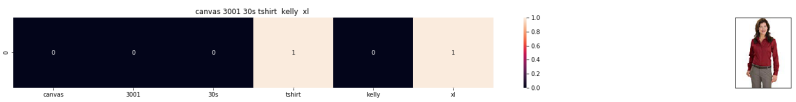
ASIN : B012VQLT6Y
Brand: KM T-shirt
Title: km tiger printed sleeveless vest t shirt
Euclidean similarity with the query image : 3.1622776601683795
==========

ASIN : B00JXQC8L6
Brand: Si Row
Title: blue peacock print tshirt l
Euclidean similarity with the query image : 3.1622776601683795
==========



ASIN : B06XC3CZF6
Brand: Fjallraven
Title: fjallraven womens ovik tshirt p lum xxl
Euclidean similarity with the query ima ge : 3.1622776601683795
==========

ASIN : B005IT8OBA
Brand: Hetalia
Title: hetalia us girl tshirt
Euclidean similarity with the query image : 3.1622776601683795
==========

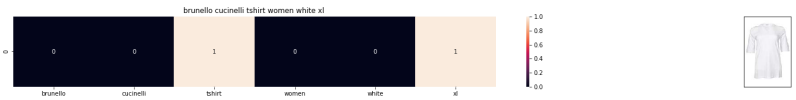


ASIN : B0088PN0LA
Brand: Red House

Title: canvas 3001 30s tshirt kelly xl

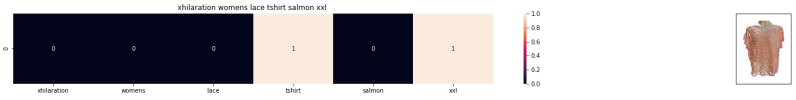
Euclidean similarity with the query image
: 3.1622776601683795

==========



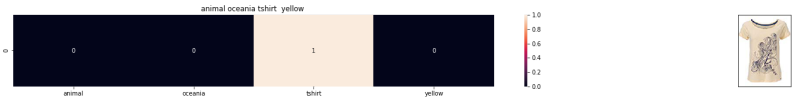
ASIN : B06X99V6WC
Brand: Brunello Cucinelli
Title: brunello cucinelli tshirt women wh
ite xl
Euclidean similarity with the query image
: 3.1622776601683795

==========



ASIN : B06Y1JPW1Q
Brand: Xhilaration
Title: xhilaration womens lace tshirt sal
mon xxl
Euclidean similarity with the query image
: 3.1622776601683795

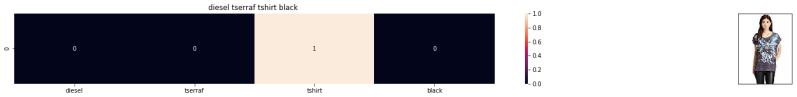
==========



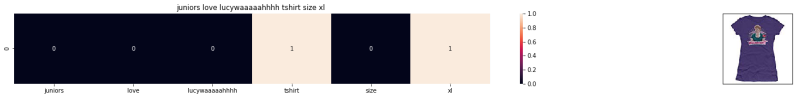
ASIN : B06X6GX6WG
Brand: Animal
Title: animal oceania tshirt yellow
Euclidean similarity with the query image
: 3.1622776601683795

==========

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




```
ASIN : B017X8PW9U
Brand: Diesel
Title: diesel tserraf tshirt black
Euclidean similarity with the query image
: 3.1622776601683795
=========================================
====================
```




```
ASIN : B00IAA4JIQ
Brand: I Love Lucy
Title: juniors love lucywaaaaahhhh tshirt
size xl
Euclidean similarity with the query image
: 3.1622776601683795
=========================================
====================
```

[8.5] TF-IDF based product similarit

In [0]:

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

In [30]:

```
def tfidf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    pairwise_dist = pairwise_distances(tfidf_ti
tle_features,tfidf_title_features[doc_id])

    # np.argsort will return indices of 9 small
est 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 dista
ce's
    df_indices = list(data.index[indices])

    for i in range(0,len(indices)):
        # we will pass 1. doc_id, 2. title1, 3.
title2, url, model
        get_result(indices[i], data['title'].lo
c[df_indices[0]], data['title'].loc[df_indices[
i]], data['medium_image_url'].loc[df_indices[i
]], 'tfidf')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('BRAND :',data['brand'].loc[df_in
dices[i]])
```

```
        print ('Eucliden distance from the give
n image :', pdists[i])
        print('='*125)
tfidf_model(12566, 20)
# in the output heat map each value represents
 the tfidf values of the label word, the color
 represents the intersection with inputs title
```

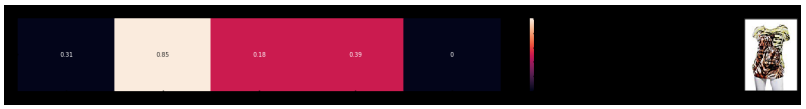
```
ASIN : B00JXQB5FQ
BRAND : Si Row
Eucliden distance from the given image
: 0.0
=========================================
=========================================
=========================================
========
```

```
ASIN : B00JXQASS6
BRAND : Si Row
Eucliden distance from the given image :
0.7536331912451363
=========================================
=========================================
=========================================
==
```

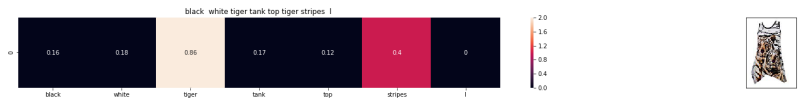
```
ASIN : B00JXQCWTO
BRAND : Si Row
```

Eucliden distance from the given image :
0.9357643943769647

ASIN : B00JXQAFZ2
BRAND : Si Row
Eucliden distance from the given image
: 0.9586153524200749

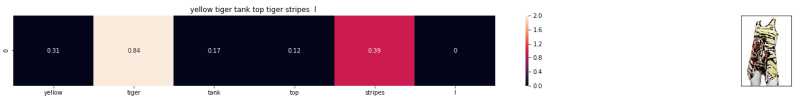


ASIN : B00JXQCUIC
BRAND : Si Row
Eucliden distance from the given image :
1.000074961446881

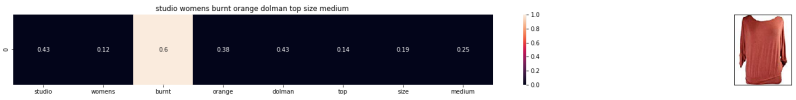


ASIN : B00JXQAO94
BRAND : Si Row
Eucliden distance from the given image :
1.023215552457452

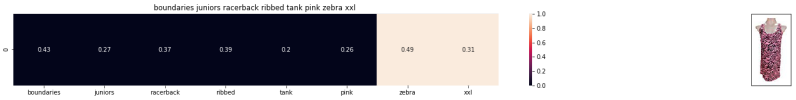
==========
==========
==========
==



ASIN : B00JXQAUWA
BRAND : Si Row
Eucliden distance from the given image :
1.031991846303421
==========
==========
==========
==



ASIN : B06XSCVFT5
BRAND : Studio M
Eucliden distance from the given image :
1.2106843670424716
==========
==========
==========
==

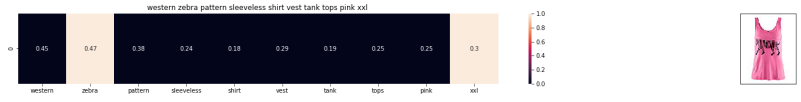


ASIN : B06Y2GTYPM
BRAND : No Boundaries
Eucliden distance from the given image :
1.2121683810720831
==========
==========

==========
==

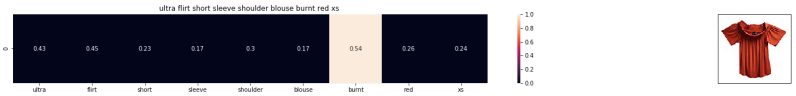
ASIN : B012VQLT6Y
BRAND : KM T-shirt
Eucliden distance from the given image :
1.219790640280982
==========
==========
==========
==

ASIN : B06Y1VN8WQ
BRAND : Black Swan
Eucliden distance from the given image :
1.2206849659998316
==========
==========
==========
==

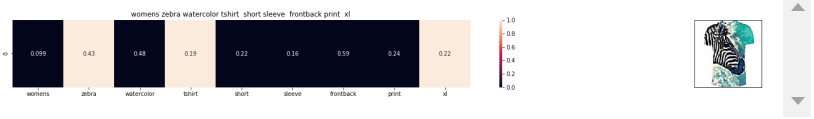


ASIN : B00Z6HEXWI
BRAND : Black Temptation
Eucliden distance from the given image :
1.221281392120943
==========
==========

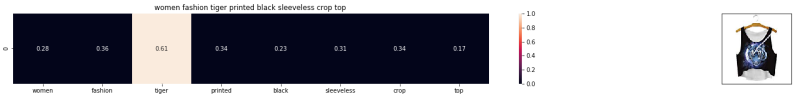
==========
==



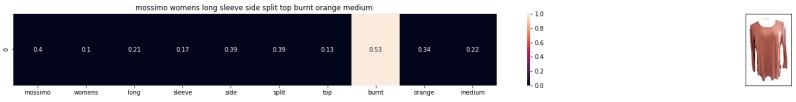
ASIN : B074TR12BH
BRAND : Ultra Flirt
Eucliden distance from the given image :
1.2313364094597743
==========
==========
==========
==



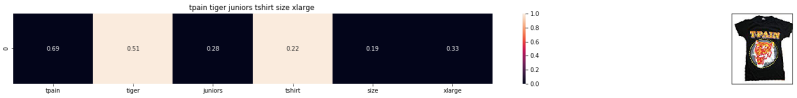
ASIN : B072R2JXKW
BRAND : WHAT ON EARTH
Eucliden distance from the given image :
1.2318451972624516
==========
==========
==========
==



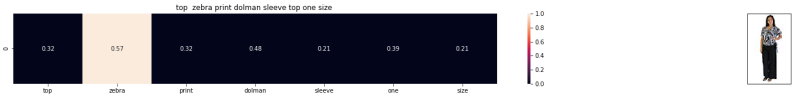
ASIN : B074T8ZYGX
BRAND : MKP Crop Top
Eucliden distance from the given image :
1.2340607457359425
==========
==========
==========
==



ASIN : B071ZDF6T2
BRAND : Mossimo
Eucliden distance from the given image :
1.2352785577664824
==========
==========
==========
==

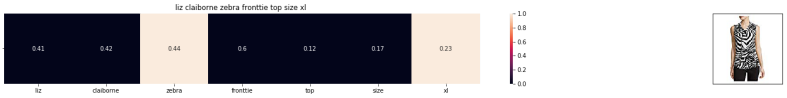


ASIN : B01K0H02OG
BRAND : Tultex
Eucliden distance from the given image
: 1.236457298812782
==========
==========
==========
========



ASIN : B00H8A6ZLI
BRAND : Vivian's Fashions
Eucliden distance from the given image :
1.24996155052848
==========
==========
==========
==


```
ASIN : B010NN9RXO
BRAND : YICHUN
Eucliden distance from the given image :
1.2535461420856102
============================================
============================================
============================================
==
```




```
ASIN : B06XBY5QXL
BRAND : Liz Claiborne
Eucliden distance from the given image :
1.2538832938357722
============================================
============================================
============================================
==
```

[8.5] IDF based product similarity

In [0]:
```
idf_title_vectorizer = CountVectorizer()
idf_title_features = idf_title_vectorizer.fit_t
ransform(data['title'])
```

In [0]:
```
def n_containing(word):
    # return the number of documents which had
 the given word
```

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

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

In [0]:

```
# 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 fi
nd its idf value
    idf_val = idf(i)

    # to calculate idf_title_features we need t
o replace the count values with the idf values
 of the word
    # idf_title_features[:, idf_title_vectorize
r.vocabulary_[i]].nonzero()[0] will return all
 documents in which the word i present
    for j in idf_title_features[:, idf_title_ve
ctorizer.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_w
ord_in_courpus] = idf value of word
        idf_title_features[j,idf_title_vectoriz
er.vocabulary_[i]] = idf_val
```

In [34]:

```
def idf_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    pairwise_dist = pairwise_distances(idf_titl
e_features,idf_title_features[doc_id])

    # np.argsort will return indices of 9 small
est 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 dista
ce's
    df_indices = list(data.index[indices])

    for i in range(0,len(indices)):
        get_result(indices[i],data['title'].loc
[df_indices[0]], data['title'].loc[df_indices[i
]], data['medium_image_url'].loc[df_indices[i
]], 'idf')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('Brand :',data['brand'].loc[df_in
dices[i]])
        print ('euclidean distance from the giv
en image :', pdists[i])
```

```
        print('='*125)



idf_model(12566,20)
# in the output heat map each value represents
 the idf values of the label word, the color re
presents the intersection with inputs title
```

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

ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from the given image :
12.20507131122177
==========
==========
==========
==

ASIN : B00JXQCWTO
Brand : Si Row
euclidean distance from the given image :
14.468362685603465
==========
==========
==========
==

ASIN : B00JXQAFZ2
Brand : Si Row
euclidean distance from the given image
: 14.486832924778964
==========
==========
==========
========

ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from the given image :
14.833392966672909
==========
==========
==========
==

ASIN : B00JXQCUIC
Brand : Si Row

euclidean distance from the given image : 14.898744516719225

==========

ASIN : B00JXQAUWA

Brand : Si Row

euclidean distance from the given image : 15.224458287343769

==========

ASIN : B074T8ZYGX

Brand : MKP Crop Top

euclidean distance from the given image : 17.080812955631995

==========

ASIN : B00KF2N5PU

Brand : Vietsbay

euclidean distance from the given image :

17.090168125645416

==========

==========

==========

==

ASIN : B00JPOZ9GM

Brand : Sofra

euclidean distance from the given image : 17.153215337562703

==========

==========

==========

==

ASIN : B074T9KG9Q

Brand : Rain

euclidean distance from the given image : 17.33671523874989

==========

==========

==========

==

ASIN : B00H8A6ZLI

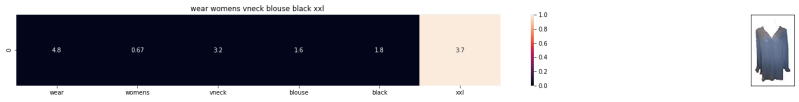
Brand : Vivian's Fashions

euclidean distance from the given image : 17.410075941001253

==========
==========
==========
==

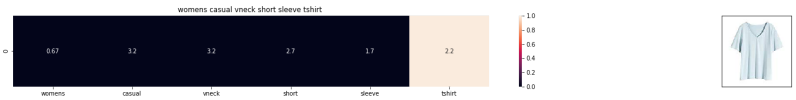
ASIN : B074G5G5RK
Brand : ERMANNO SCERVINO
euclidean distance from the given image : 17.539921335459557
==========
==========
==========
==

ASIN : B06XSCVFT5
Brand : Studio M
euclidean distance from the given image : 17.61275854366134
==========
==========
==========
==



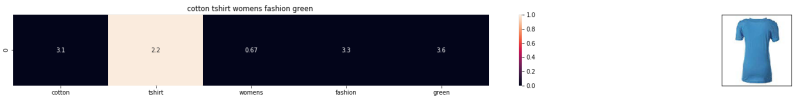
ASIN : B06Y6FH453
Brand : Who What Wear
euclidean distance from the given image : 17.623745282500135
==========
==========

==========
==



ASIN : B074V45DCX
Brand : Rain
euclidean distance from the given image :
17.634342496835046
==========
==========
==========
==

ASIN : B07583CQFT
Brand : Very J
euclidean distance from the given image :
17.63753712743611
==========
==========
==========
==



ASIN : B073GJGVBN
Brand : Ivan Levi
euclidean distance from the given image :
17.7230738913371
==========
==========

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

```
ASIN : B012VQLT6Y
Brand : KM T-shirt
euclidean distance from the given image :
17.762588561202364
=========================================
=========================================
=========================================
==
```


```
ASIN : B00ZZMYBRG
Brand : HP-LEISURE
euclidean distance from the given image
: 17.779536864674238
=========================================
=========================================
=========================================
========
```

[9] Text Semantics based product similarity

In [35]:

```
# credits: https://www.kaggle.com/c/word2vec-nl
p-tutorial#part-2-word-vectors
# Custom Word2Vec using your own text data.
# Do NOT RUN this code.
```

```
# It is meant as a reference to build your own
 Word2Vec when you have
# lots of data.

'''
# Set values for various parameters
num_features = 300    # Word vector dimensional
ity
min_word_count = 1    # Minimum word count
num_workers = 4       # Number of threads to ru
n in parallel
context = 10          # Context window size
downsampling = 1e-3   # Downsample setting for
 frequent words

# Initialize and train the model (this will tak
e some time)
from gensim.models import word2vec
print ("Training model...")
model = word2vec.Word2Vec(sen_corpus, workers=n
um_workers, \
            size=num_features, min_count = min_
word_count, \
            window = context)

'''
```

Out[35]:

```
'\n# Set values for various parameters\nn
um_features = 300    # Word vector dimens
ionality                      \nmin_word_
count = 1    # Minimum word count
\nnum_workers = 4       # Number of threa
ds to run in parallel\ncontext = 10
# Context window size
\ndownsampling = 1e-3   # Downsample sett
ing for frequent words\n\n# Initialize an
```

```
d train the model (this will take some ti
me)\nfrom gensim.models import word2vec\n
print ("Training model...")\nmodel = word
2vec.Word2Vec(sen_corpus, workers=num_wor
kers,            size=num_features, min_
count = min_word_count,            windo
w = context)\n    \n'
```

In [0]:

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

# in this project we are using a pretrained mod
el by google
# its 3.3G file, once you load this into your m
emory
# 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 "GoogleNew
s-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwp
I5KDYNlNUTTlSS21pQmM/edit
# it's 1.9GB in size.

'''
model = KeyedVectors.load_word2vec_format('Goog
leNews-vectors-negative300.bin', binary=True)
'''

#if you do NOT have RAM >= 12GB, use the code b
elow.
```

```
with open('/content/drive/My Drive/Applied_AI_Workshop_Code_Data/word2vec_model', 'rb') as handle:
    model = pickle.load(handle)
```

In [0]:

```
# 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 represen
```

```
tation of each word (weighted/avg) in given sen
tance
    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 corres
ponds to each word in give title
    # vec2 = np.array(#number_of_words_title2 *
300), each row is a vector of length 300 corres
ponds to each word in 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 t
he 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 betw
een 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 appar
```

```
el
    # 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 ti
tle
    s1_vec = get_word_vec(sentence1, doc_id1, m
odel)
    #s2_vec = np.array(#number_of_words_title1
 * 300), each row is a vector(weighted/avg) of
 length 300 corresponds to each word in give ti
tle
    s2_vec = get_word_vec(sentence2, doc_id2, m
odel)

    # s1_s2_dist = np.array(#number of words in
title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance betw
een words i, j
    s1_s2_dist = get_distance(s1_vec, s2_vec)



    # devide whole figure into 2 parts 1st part
displays heatmap 2nd part displays image of app
arel
    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 pairwis
e distances
    ax = sns.heatmap(np.round(s1_s2_dist,4), an
```

```
not=True)
    # set the x axis labels as recommended appa
rels title
    ax.set_xticklabels(sentence2.split())
    # set the y axis labels as input apparels t
itle
    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 i
mage
    ax.grid(False)
    ax.set_xticks([])
    ax.set_yticks([])
    display_img(url, ax, fig)

    plt.show()
```

In [0]:

```
# vocab = stores all the words that are there i
n google w2v model
# vocab = model.wv.vocab.keys() # if you are us
ing Google word2Vec

vocab = model.keys()
# this function will add the vectors of each wo
rd and returns the avg vector of given sentance
def build_avg_vec(sentence, num_features, doc_i
d, m_name):
    # sentace: its title of the apparel
    # num_features: the lenght of word2vec vect
or, its values = 300
    # m_name: model information it will take tw
o values
```

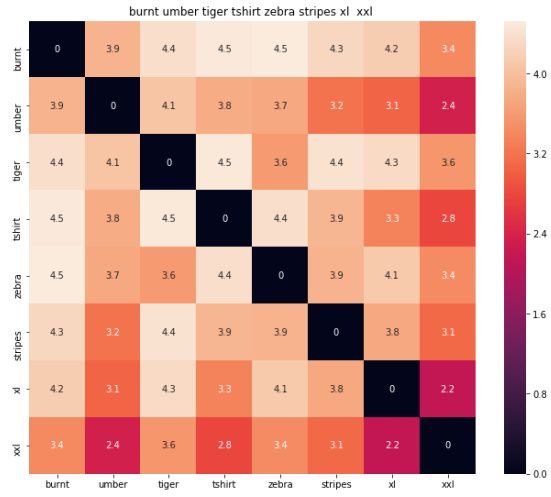
```
        # if  m_name == 'avg', we will append t
he model[i], w2v representation of word i
        # if m_name == 'weighted', we will mult
iply each w2v[word] with the idf(word)

    featureVec = np.zeros((num_features,), dtyp
e="float32")
    # we will intialize a vector of size 300 wi
th all zeros
    # we add each word2vec(wordi) to this fetur
eVec
    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, nwor
ds)
    # returns the avg vector of given sentance,
its of shape (1, 300)
    return featureVec
```

[9.2] Average Word2Vec product similarity

In [0]:
```
doc_id = 0
w2v_title = []
# for every title we build a avg vector represe
ntation
for i in data['title']:
    w2v_title.append(build_avg_vec(i, 300, doc_
id,'avg'))
    doc_id += 1

# w2v_title = np.array(# number of doc in courp
us * 300), each row corresponds to a doc
w2v_title = np.array(w2v_title)
```

In [40]:
```
%%timeit
def avg_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # dist(x, y) = sqrt(dot(x, x) - 2 * dot(x,
 y) + dot(y, y))
    pairwise_dist = pairwise_distances(w2v_titl
e, w2v_title[doc_id].reshape(1,-1))

    # np.argsort will return indices of 9 small
est 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 dista
ce's
    df_indices = list(data.index[indices])

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

```
        heat_map_w2v(data['title'].loc[df_indic
es[0]],data['title'].loc[df_indices[i]], data
['medium_image_url'].loc[df_indices[i]], indice
s[0], indices[i], 'avg')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('BRAND :',data['brand'].loc[df_in
dices[i]])
        print ('euclidean distance from given i
nput image :', pdists[i])
        print('='*125)


avg_w2v_model(12566, 20)
# in the give heat map, each cell contains the
 euclidean distance between words i, j
```

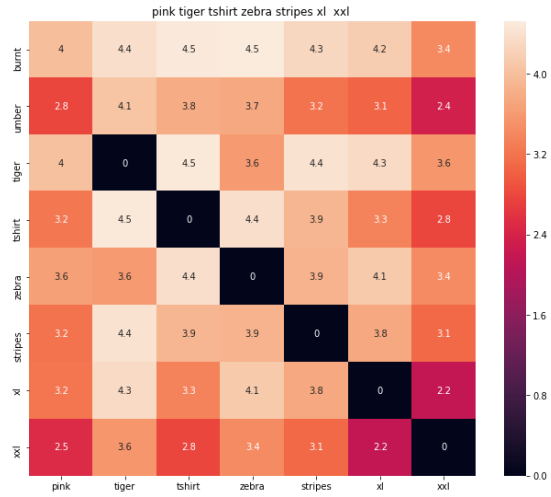


ASIN : B00JXQB5FQ

BRAND : Si Row

euclidean distance from given input image : 0.0

==========

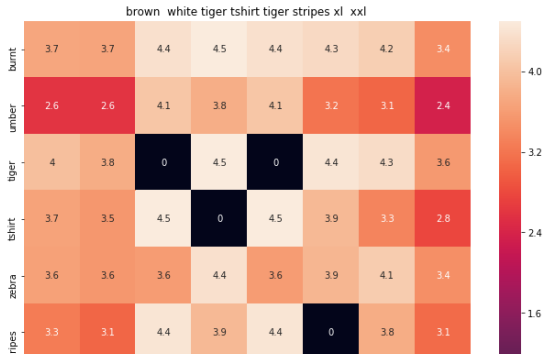


ASIN : B00JXQASS6

BRAND : Si Row

euclidean distance from given input image : 0.5891926

==========



ASIN : B00JXQCWTO

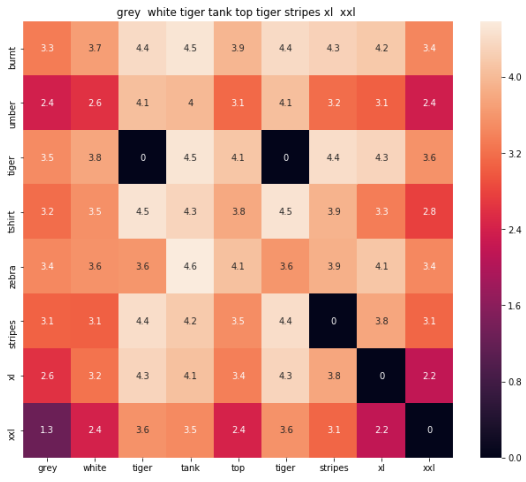
BRAND : Si Row

euclidean distance from given input image : 0.7003438

==========

==========

==



ASIN : B00JXQAFZ2

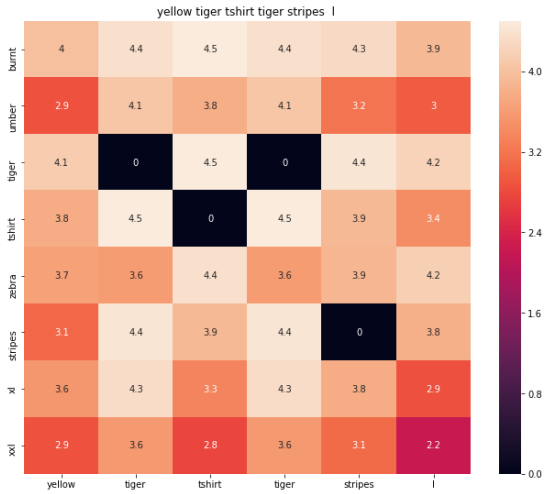
BRAND : Si Row

euclidean distance from given input image : 0.89283955

==========

==========

==

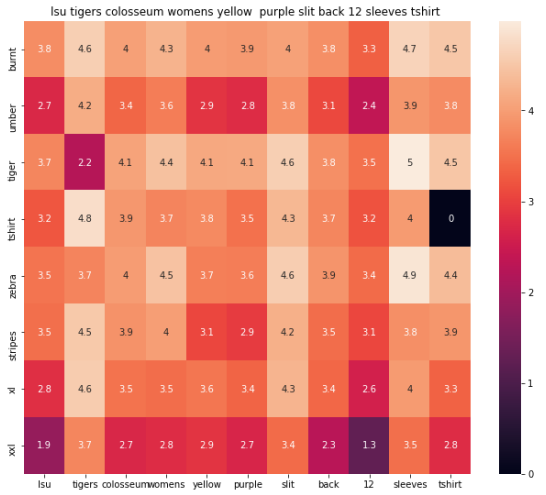


ASIN : B00JXQCUIC

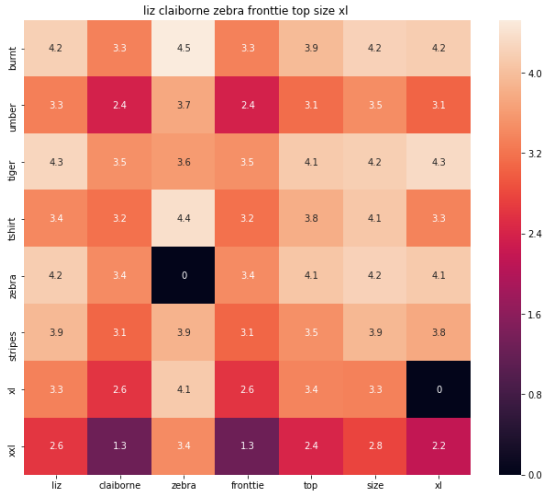
BRAND : Si Row

euclidean distance from given input image : 0.95601255

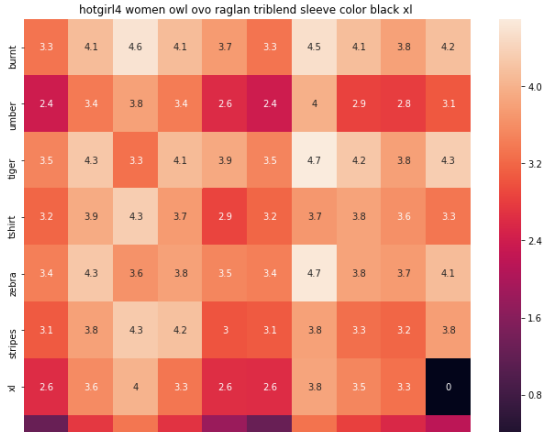
==========



ASIN : B073R5Q8HD
BRAND : Colosseum
euclidean distance from given input image : 1.022969
==========
==========
==========
==



ASIN : B06XBY5QXL
BRAND : Liz Claiborne
euclidean distance from given input image : 1.0669324
==========
==========
==========
==



ASIN : B01L8L73M2

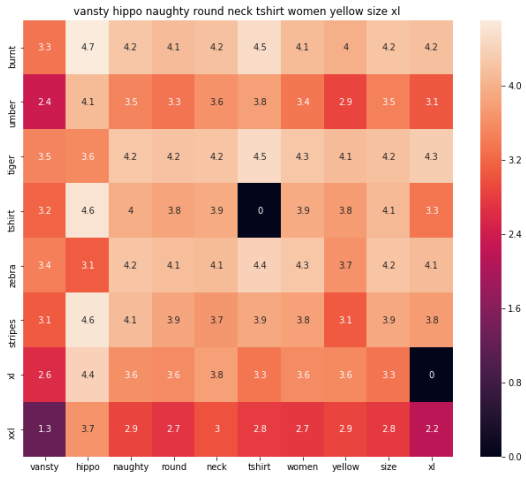
BRAND : Hotgirl4 Raglan Design

euclidean distance from given input image : 1.0731405

==========

==========

==========



ASIN : B01EJS5H06

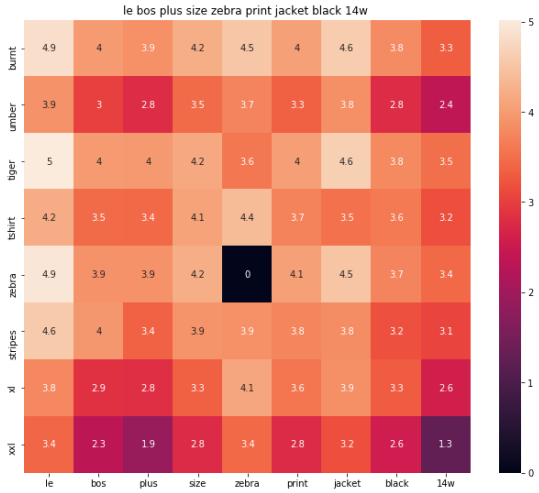
BRAND : Vansty

euclidean distance from given input image : 1.075719

==========

==========

==========



ASIN : B01BO1XRK8

BRAND : Le Bos

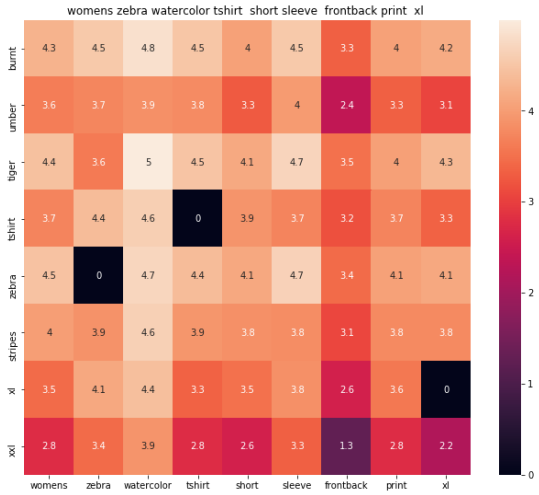
euclidean distance from given input image : 1.0839964

==========

==========

==========

========

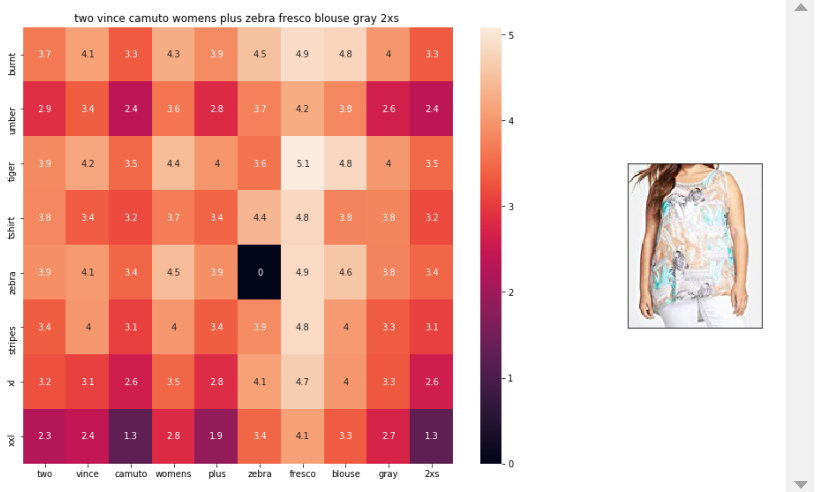


ASIN : B072R2JXKW

BRAND : WHAT ON EARTH

euclidean distance from given input image : 1.0842218

==========

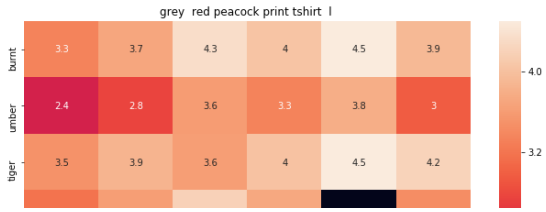


ASIN : B074MJRGW6

BRAND : Two by Vince Camuto

euclidean distance from given input image : 1.0895038

==========



ASIN : B00JXQCFRS

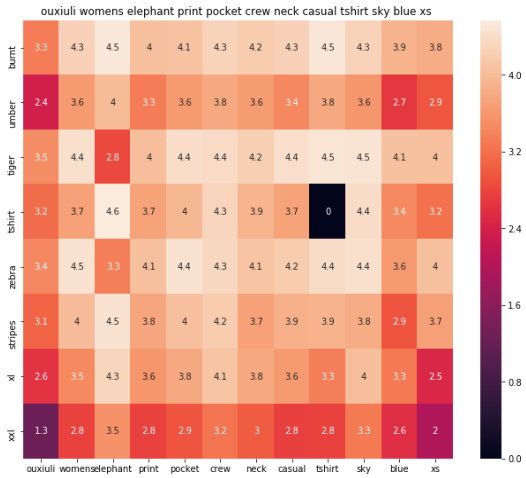
BRAND : Si Row

euclidean distance from given input image : 1.0900588

==========

==========

==========



ASIN : B01I53HU6K

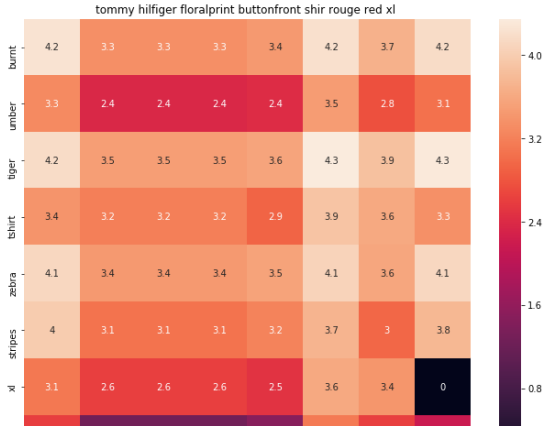
BRAND : ouxiuli

euclidean distance from given input image : 1.0920111

==========

==========

==========



ASIN : B0711NGTQM

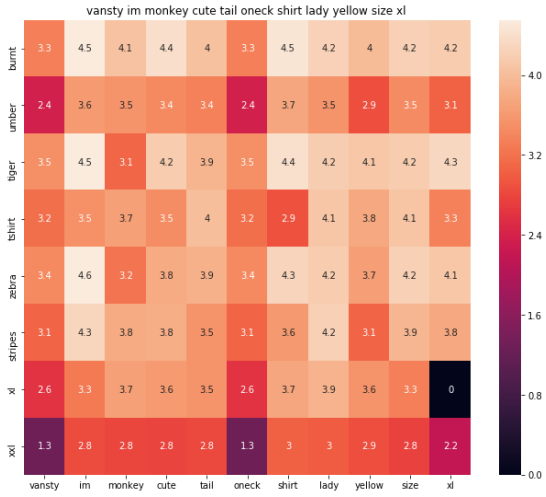
BRAND : THILFIGER RTW

euclidean distance from given input image : 1.0923415

==========

==========

==========



ASIN : B01EFSLO8Y

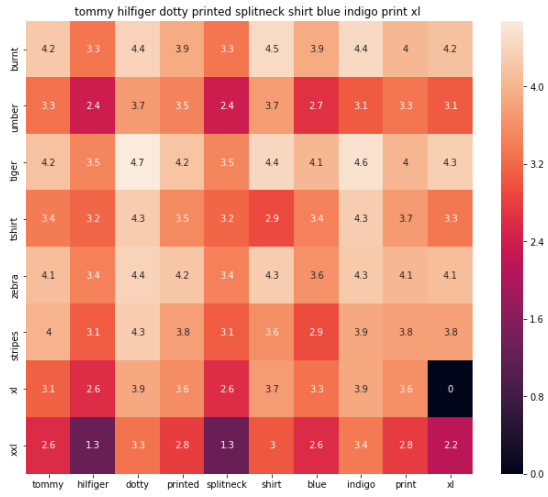
BRAND : Vansty

euclidean distance from given input image : 1.0934004

==========

==========

==========



ASIN : B0716TVWQ4

BRAND : THILFIGER RTW

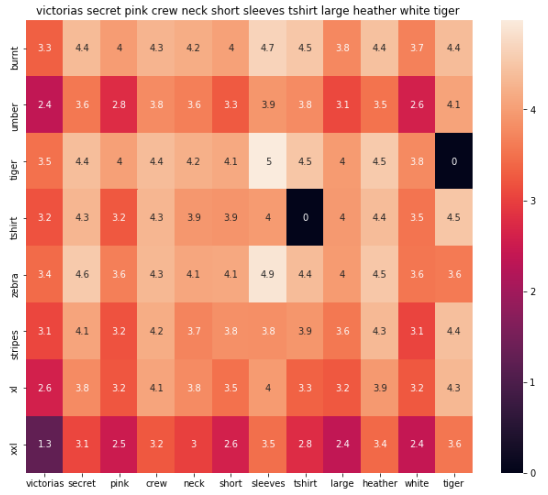
euclidean distance from given input image : 1.0942024

==========

==========

==========

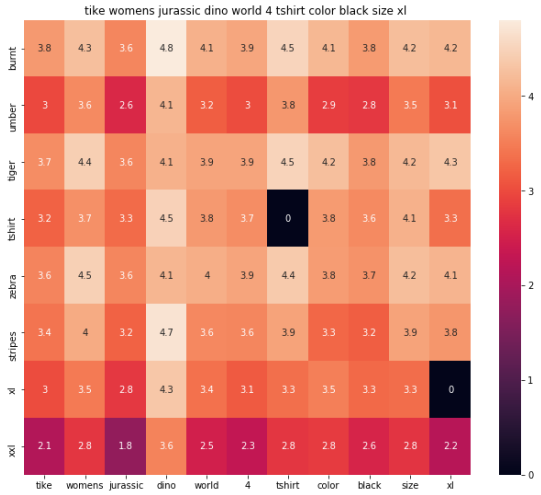
========



ASIN : B0716MVPGV

BRAND : V.Secret

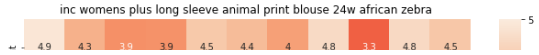
euclidean distance from given input image : 1.0948304



ASIN : B016OPN4OI

BRAND : TIKE Fashions

euclidean distance from given input image : 1.0951275

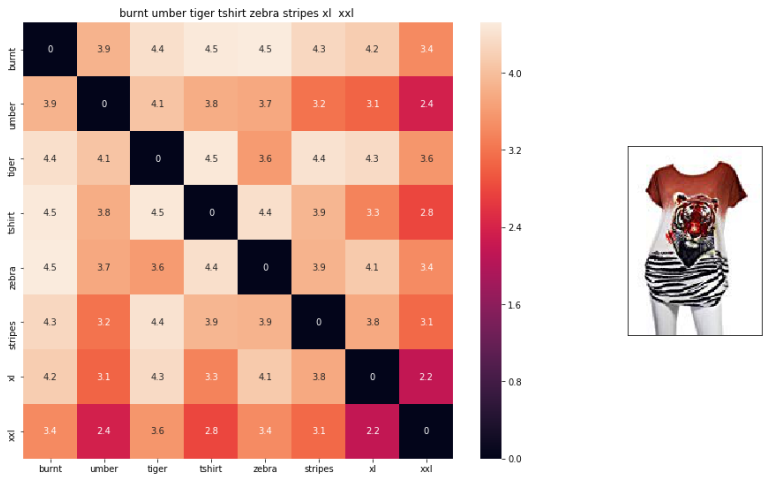


ASIN : B018WDJCUA

BRAND : INC - International Concepts Woma
n

euclidean distance from given input image
: 1.0966892

==========
==========
==========
==

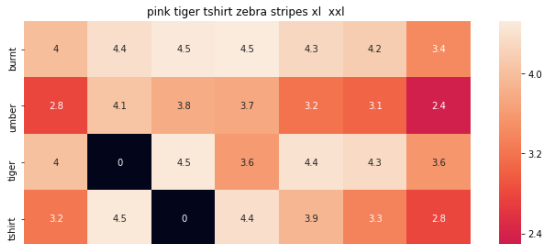


ASIN : B00JXQB5FQ

BRAND : Si Row

euclidean distance from given input image
: 0.0

==========
==========
==========
==

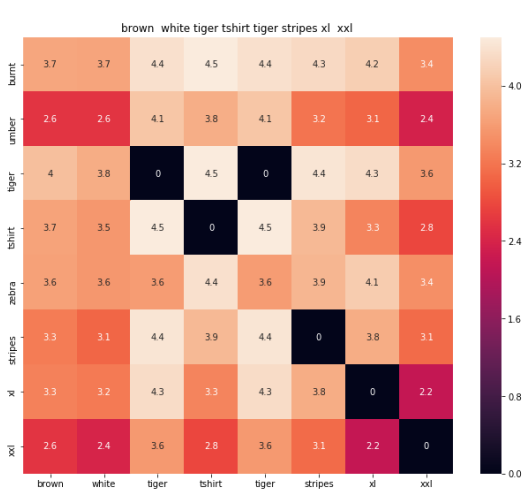


ASIN : B00JXQASS6

BRAND : Si Row

euclidean distance from given input image : 0.5891926

==========

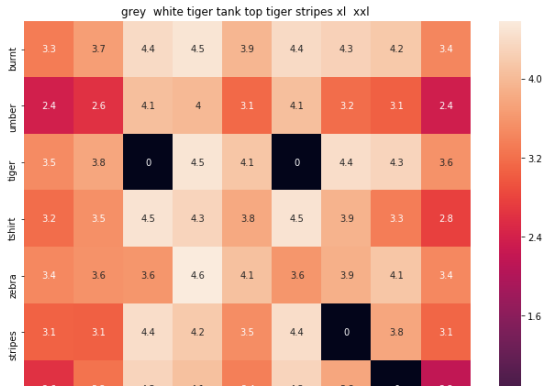


ASIN : B00JXQCWTO

BRAND : Si Row

euclidean distance from given input image : 0.7003438

==========



ASIN : B00JXQAFZ2

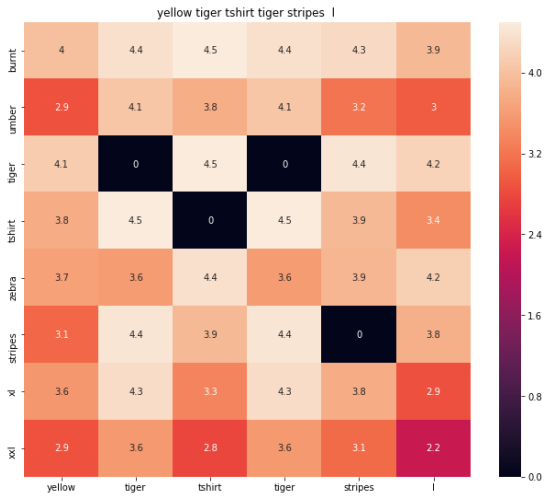
BRAND : Si Row

euclidean distance from given input image : 0.89283955

==========

==========

==========



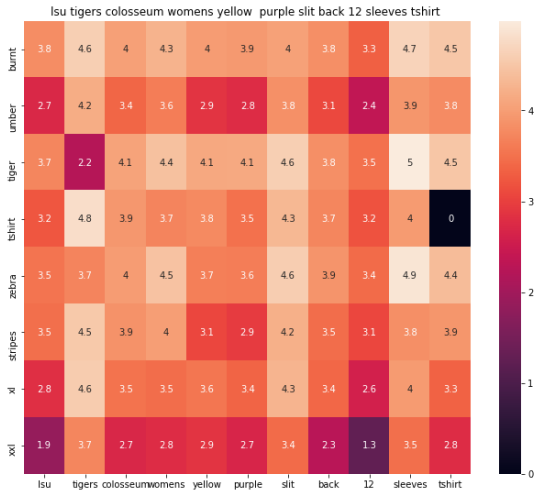
ASIN : B00JXQCUIC

BRAND : Si Row

euclidean distance from given input image : 0.95601255

==========

==========

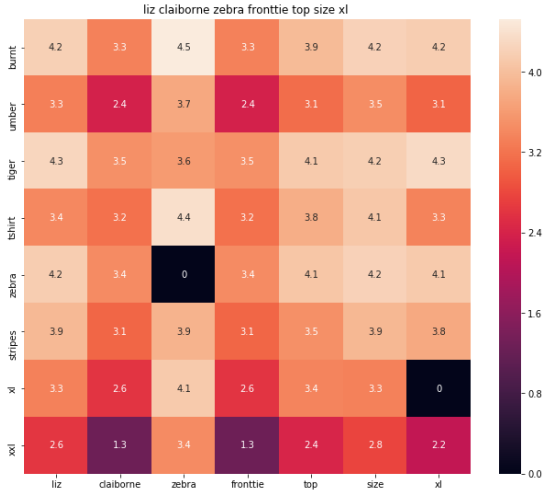


ASIN : B073R5Q8HD

BRAND : Colosseum

euclidean distance from given input image : 1.022969

==========

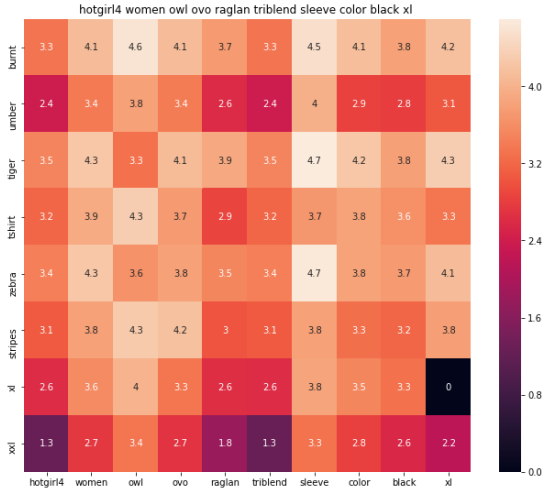


ASIN : B06XBY5QXL

BRAND : Liz Claiborne

euclidean distance from given input image : 1.0669324

==========
==========
==========
==

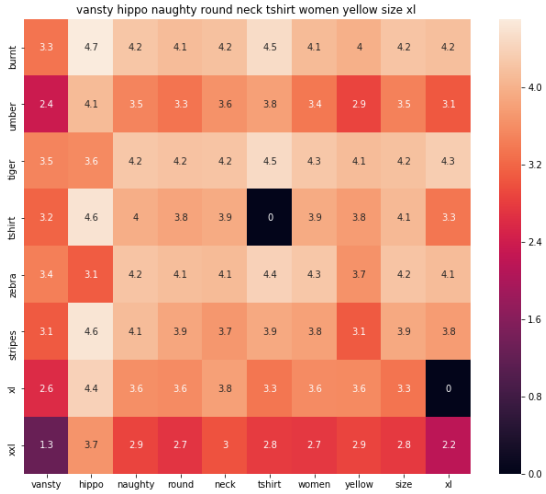


ASIN : B01L8L73M2

BRAND : Hotgirl4 Raglan Design

euclidean distance from given input ima
ge : 1.0731405

==========
==========
==========
========

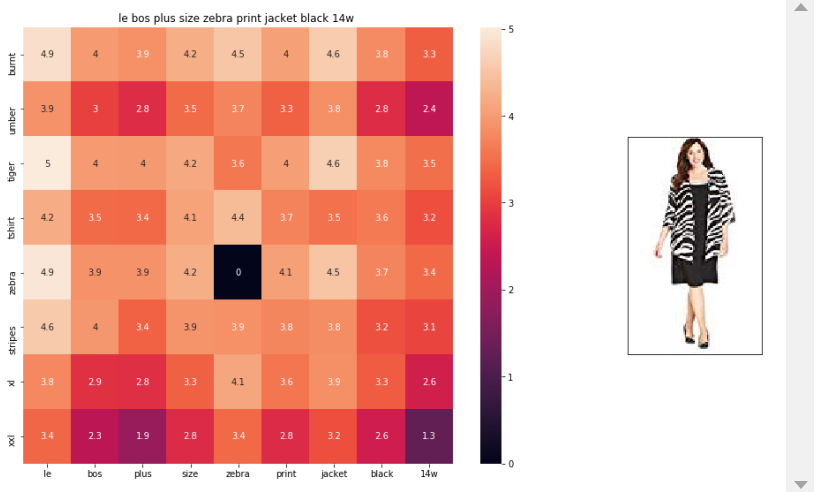


ASIN : B01EJS5H06

BRAND : Vansty

euclidean distance from given input image : 1.075719

==========



ASIN : B01BO1XRK8

BRAND : Le Bos

euclidean distance from given input image : 1.0839964

==========

ASIN : B072R2JXKW

BRAND : WHAT ON EARTH

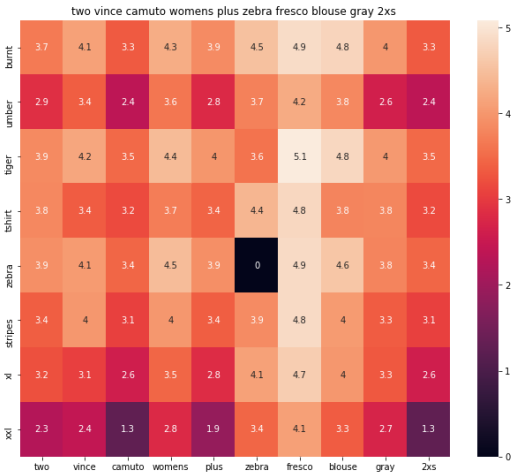
euclidean distance from given input image : 1.0842218

==========

==========

==========

==



ASIN : B074MJRGW6

BRAND : Two by Vince Camuto

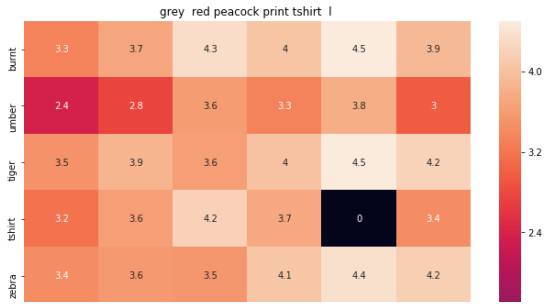
euclidean distance from given input image : 1.0895038

==========

==========

==========

==



ASIN : B00JXQCFRS

BRAND : Si Row

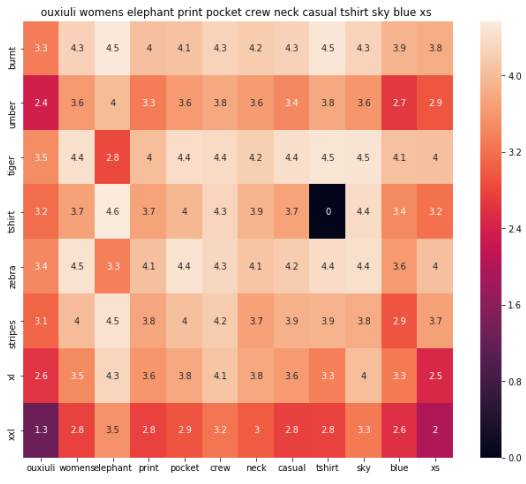
euclidean distance from given input image : 1.0900588

==========

==========

==========

==



ASIN : B01I53HU6K

BRAND : ouxiuli

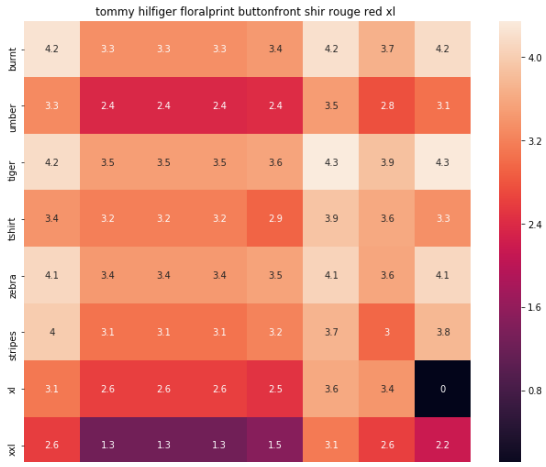
euclidean distance from given input image : 1.0920111

==========

==========

==========

==



ASIN : B0711NGTQM

BRAND : THILFIGER RTW

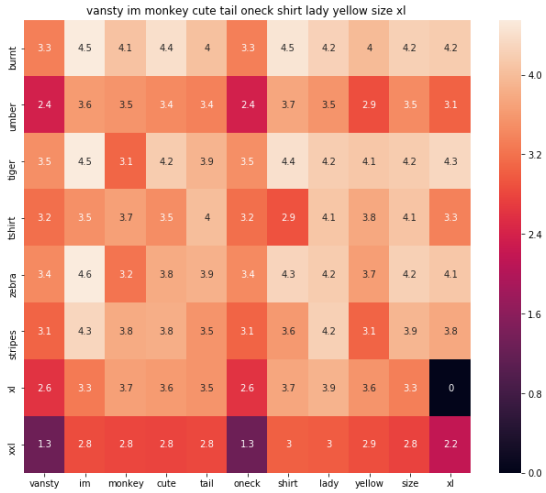
euclidean distance from given input image : 1.0923415

==========

==========

==========

========



ASIN : B01EFSLO8Y

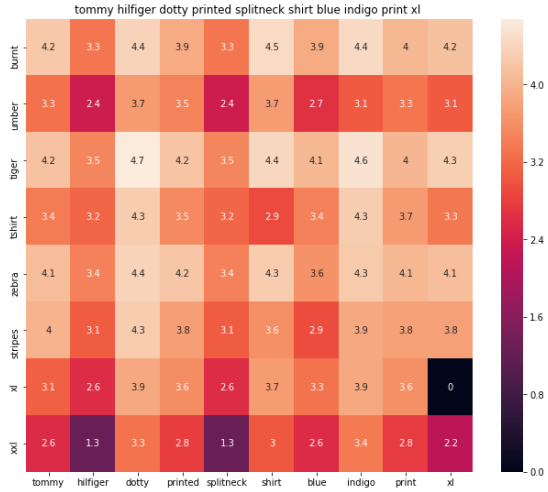
BRAND : Vansty

euclidean distance from given input image : 1.0934004

==========

==========

==========



ASIN : B0716TVWQ4

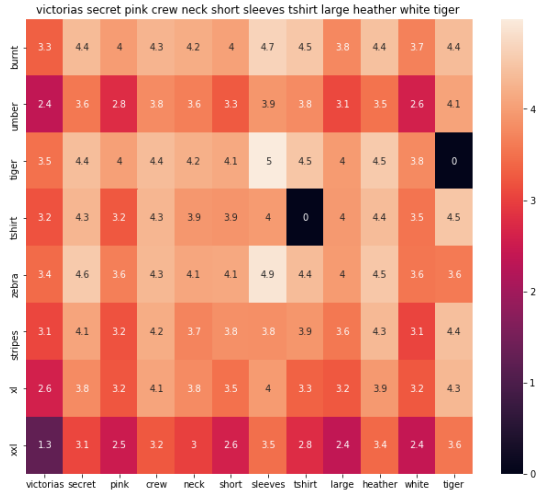
BRAND : THILFIGER RTW

euclidean distance from given input image : 1.0942024

==========

==========

==========

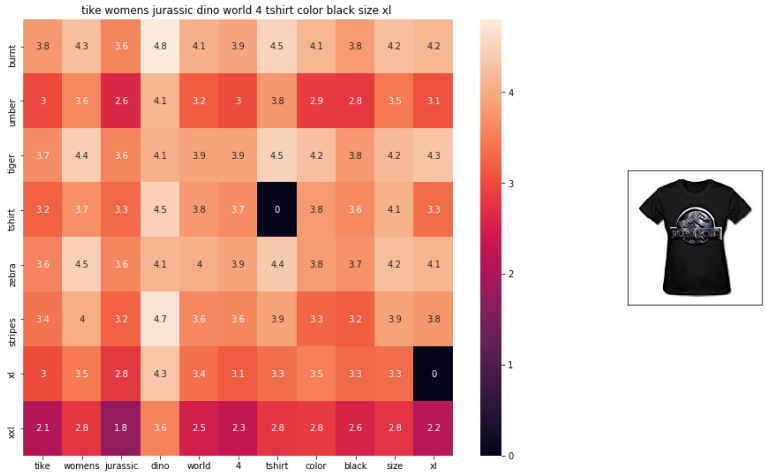


ASIN : B0716MVPGV

BRAND : V.Secret

euclidean distance from given input image : 1.0948304

==========

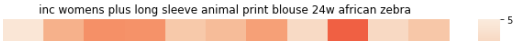


ASIN : B016OPN4OI

BRAND : TIKE Fashions

euclidean distance from given input image : 1.0951275

==========

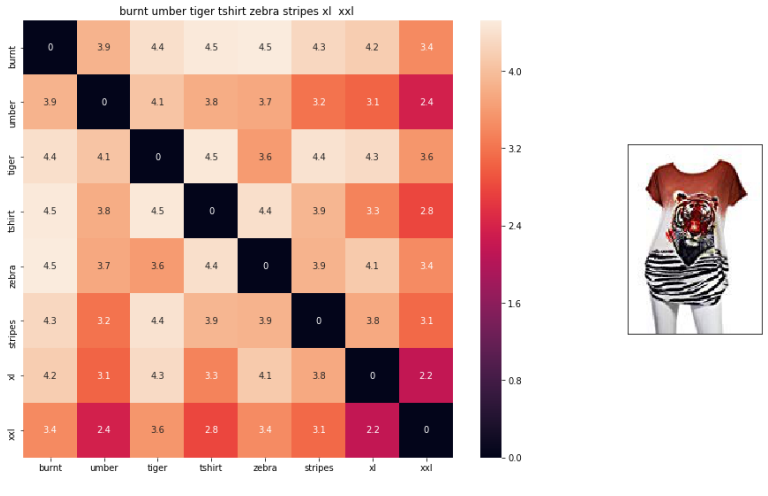


ASIN : B018WDJCUA

BRAND : INC - International Concepts Woma
n

euclidean distance from given input image
: 1.0966892

==========
==========
==========
==



ASIN : B00JXQB5FQ

BRAND : Si Row

euclidean distance from given input image
: 0.0

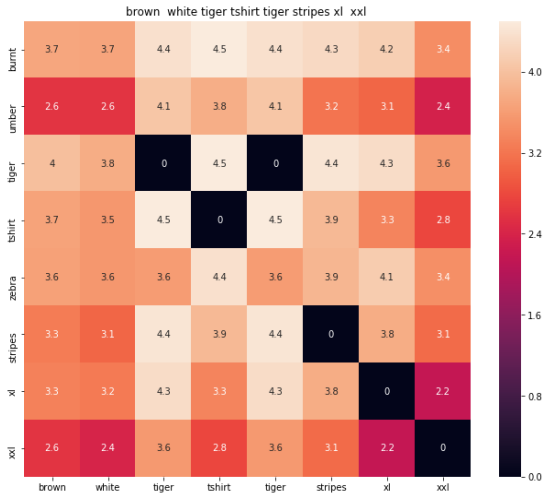
==========
==========
==========
==

ASIN : B00JXQASS6

BRAND : Si Row

euclidean distance from given input image : 0.5891926

==========

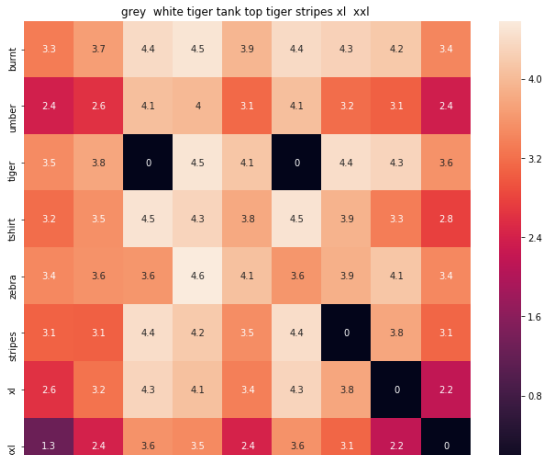


ASIN : B00JXQCWTO

BRAND : Si Row

euclidean distance from given input image : 0.7003438

==========

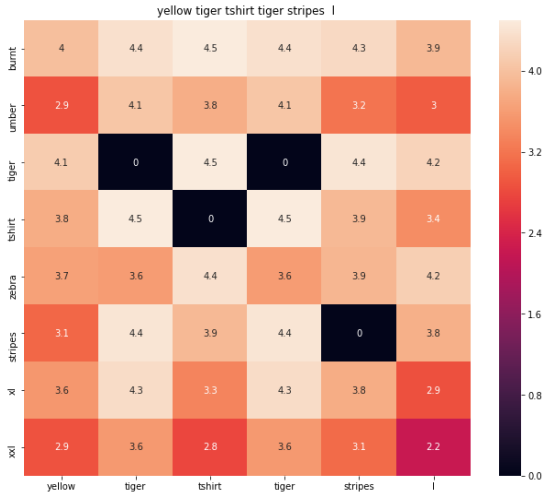


ASIN : B00JXQAFZ2

BRAND : Si Row

euclidean distance from given input image : 0.89283955

==========



ASIN : B00JXQCUIC

BRAND : Si Row

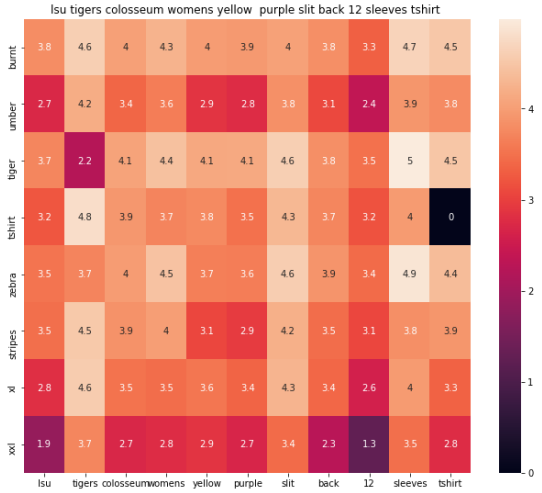
euclidean distance from given input image : 0.95601255

==========

==========

==========

==



ASIN : B073R5Q8HD

BRAND : Colosseum

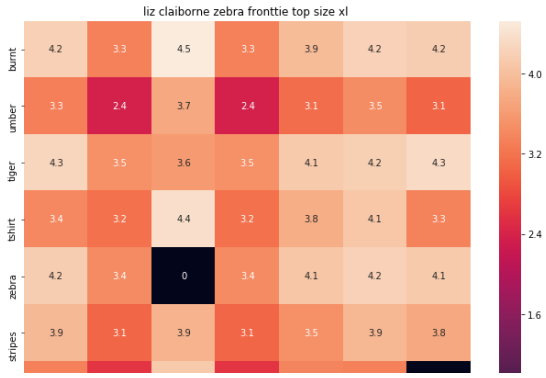
euclidean distance from given input image : 1.022969

==========

==========

==========

==



ASIN : B06XBY5QXL

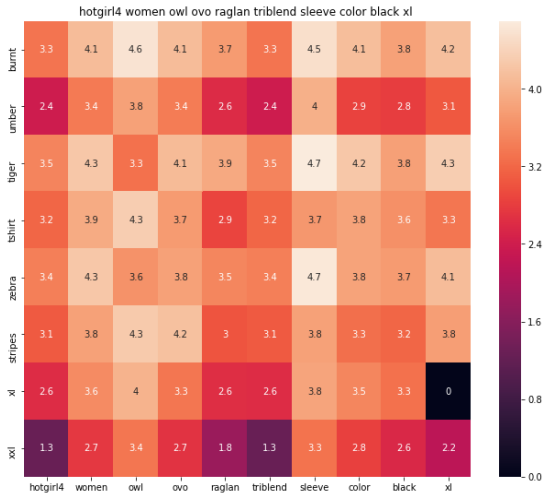
BRAND : Liz Claiborne

euclidean distance from given input image : 1.0669324

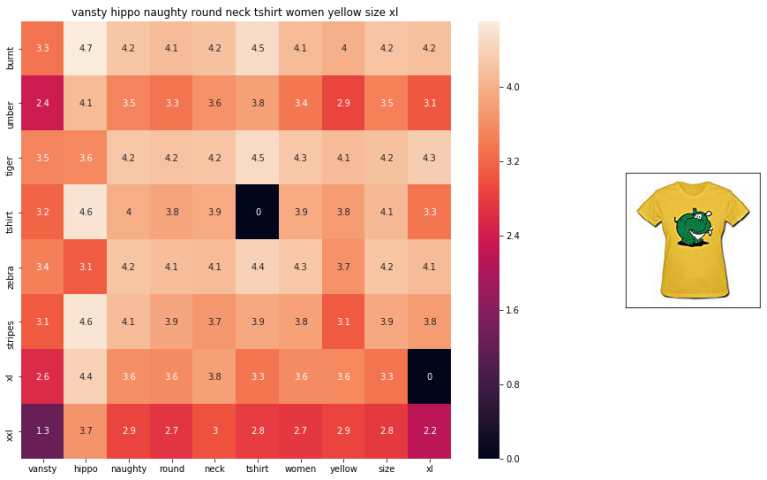
==========

==========

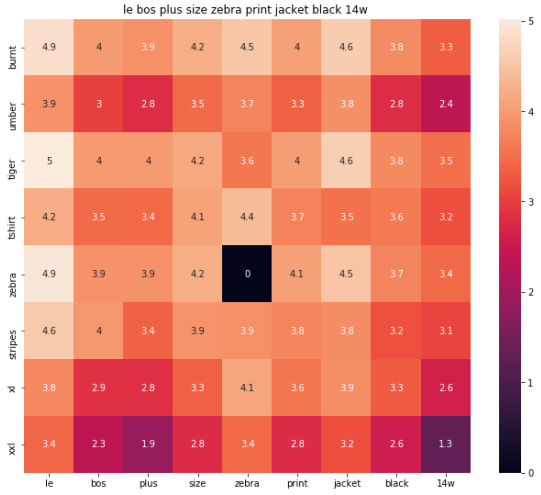
==========



ASIN : B01L8L73M2
BRAND : Hotgirl4 Raglan Design
euclidean distance from given input image : 1.0731405
==========
==========
==========
==



ASIN : B01EJS5H06
BRAND : Vansty
euclidean distance from given input image : 1.075719
==========
==========
==========
==



ASIN : B01BO1XRK8

BRAND : Le Bos

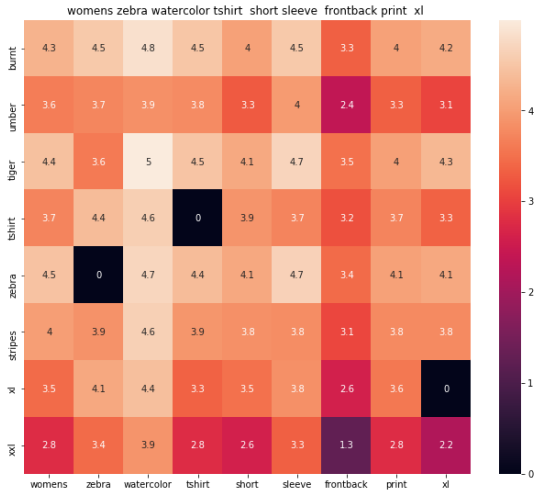
euclidean distance from given input image : 1.0839964

==========

==========

==========

========



ASIN : B072R2JXKW

BRAND : WHAT ON EARTH

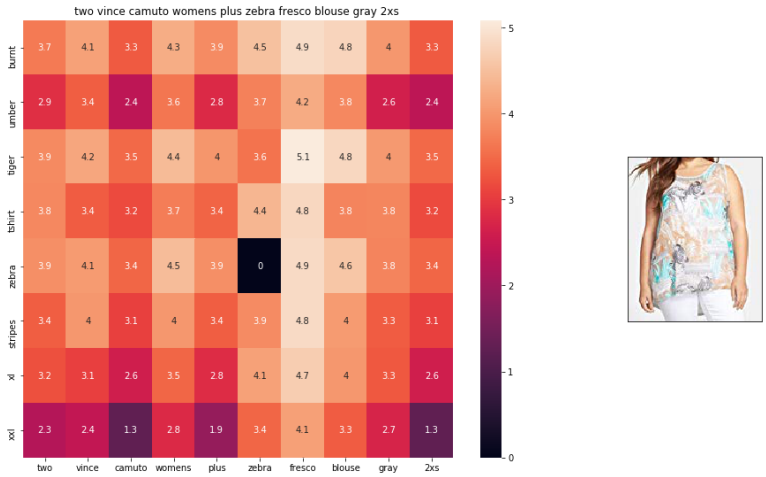
euclidean distance from given input image

: 1.0842218

==========

==========

==



ASIN : B074MJRGW6

BRAND : Two by Vince Camuto

euclidean distance from given input image : 1.0895038

==========

==========

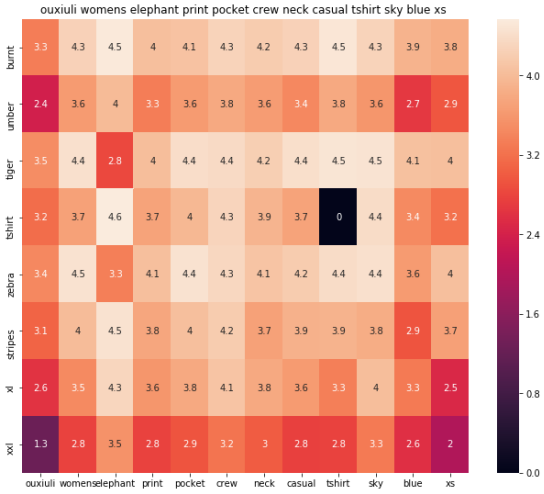
==

grey red peacock print tshirt l

ASIN : B00JXQCFRS

BRAND : Si Row

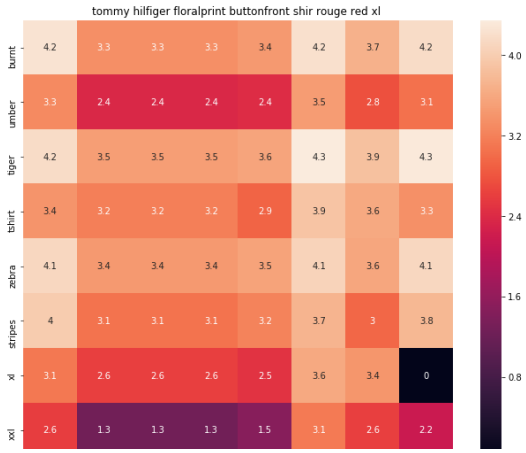
euclidean distance from given input image : 1.0900588



ASIN : B01I53HU6K

BRAND : ouxiuli

euclidean distance from given input image : 1.0920111



ASIN : B0711NGTQM

BRAND : THILFIGER RTW

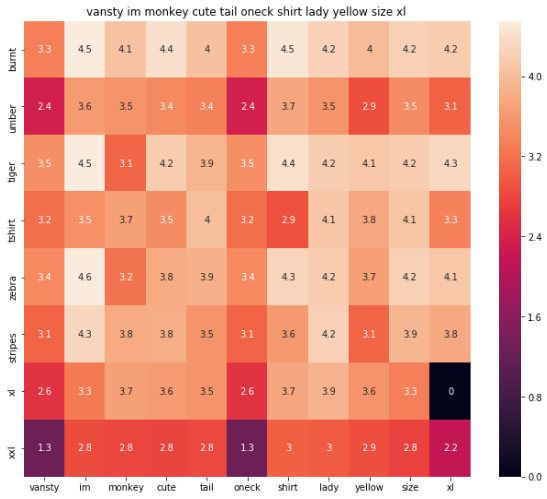
euclidean distance from given input image : 1.0923415

==========

==========

==========

==

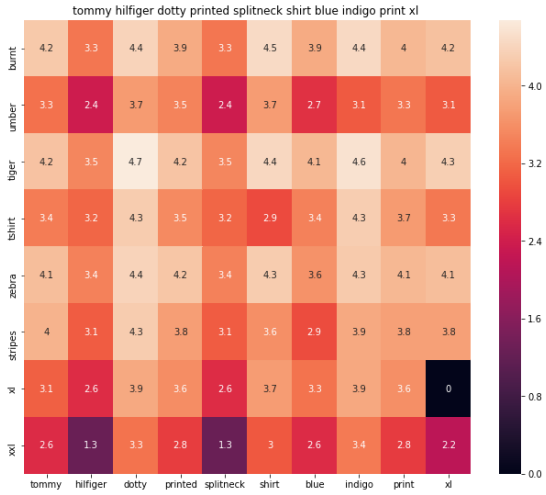


ASIN : B01EFSLO8Y

BRAND : Vansty

euclidean distance from given input ima
ge : 1.0934004

==========
==========
==========
========

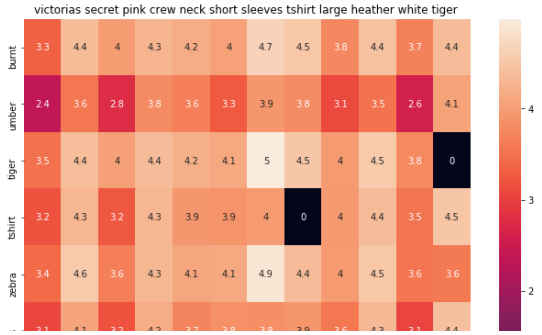


ASIN : B0716TVWQ4

BRAND : THILFIGER RTW

euclidean distance from given input image
: 1.0942024

==========
==========
==========
==



ASIN : B0716MVPGV

BRAND : V.Secret

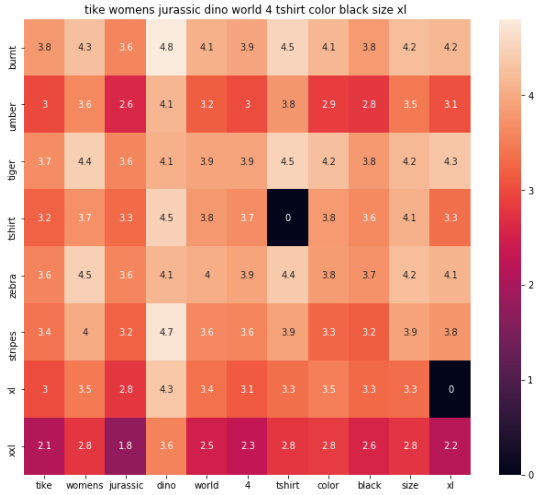
euclidean distance from given input image : 1.0948304

==========

==========

==========

==



ASIN : B016OPN4OI

BRAND : TIKE Fashions

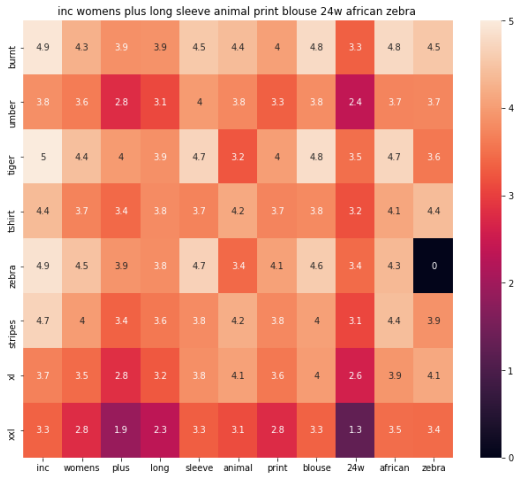
euclidean distance from given input image : 1.0951275

==========

==========

==========

==

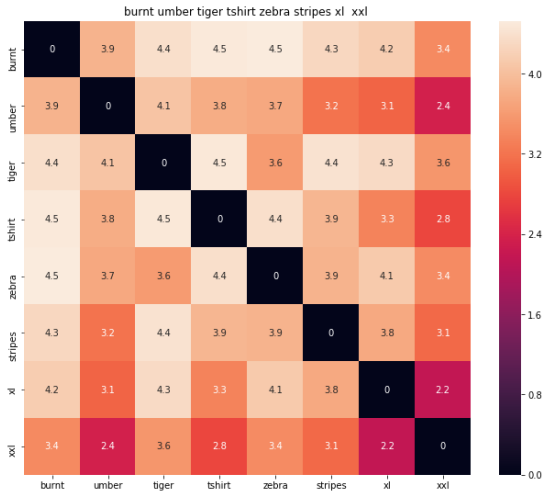


ASIN : B018WDJCUA

BRAND : INC - International Concepts Woma
n

euclidean distance from given input image
: 1.0966892

==========
==========
==========
==



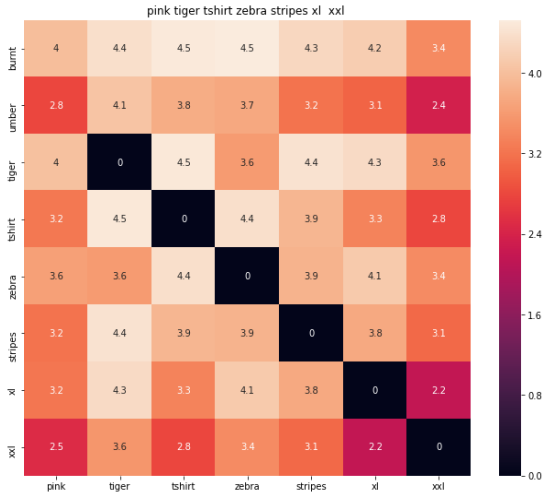
ASIN : B00JXQB5FQ

BRAND : Si Row

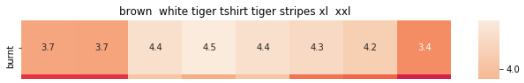
euclidean distance from given input ima
ge : 0.0

==========

==========
==========
========



ASIN : B00JXQASS6
BRAND : Si Row
euclidean distance from given input image : 0.5891926
==========
==========
==========
==



ASIN : B00JXQCWTO

BRAND : Si Row

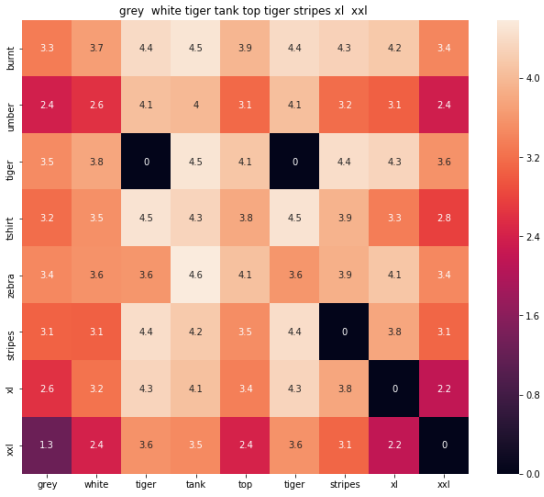
euclidean distance from given input image : 0.7003438

==========

==========

==========

==



ASIN : B00JXQAFZ2

BRAND : Si Row

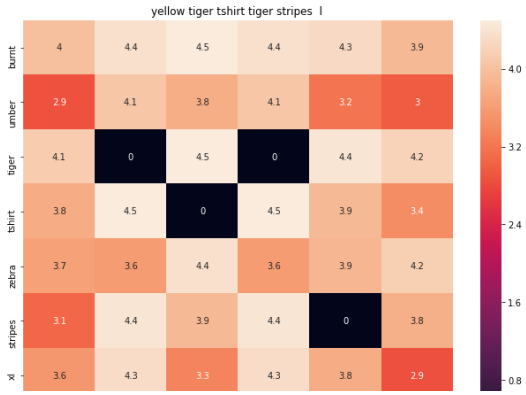
euclidean distance from given input image : 0.89283955

==========

==========

==========

==

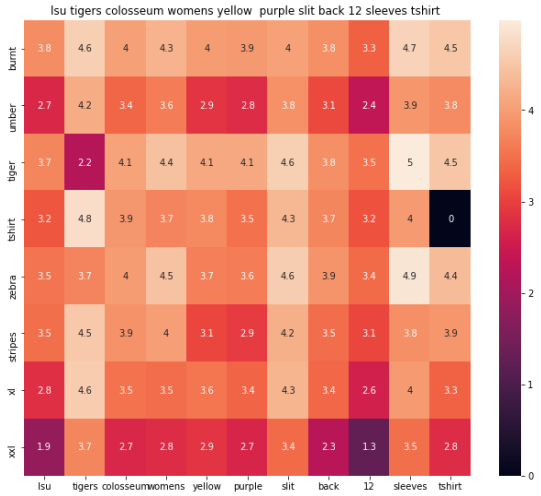


ASIN : B00JXQCUIC

BRAND : Si Row

euclidean distance from given input image : 0.95601255

==========



ASIN : B073R5Q8HD

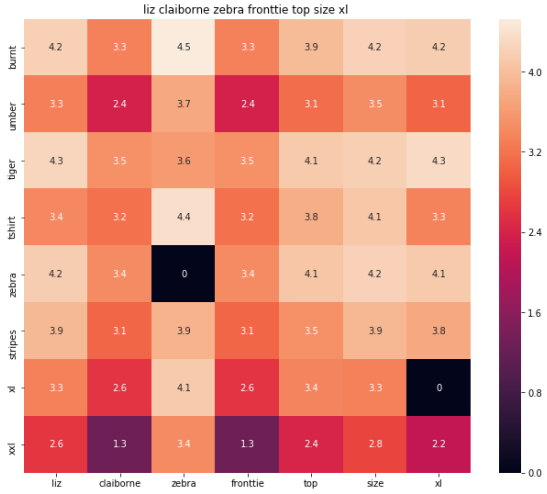
BRAND : Colosseum

euclidean distance from given input image : 1.022969

==========

==========

==========



ASIN : B06XBY5QXL

BRAND : Liz Claiborne

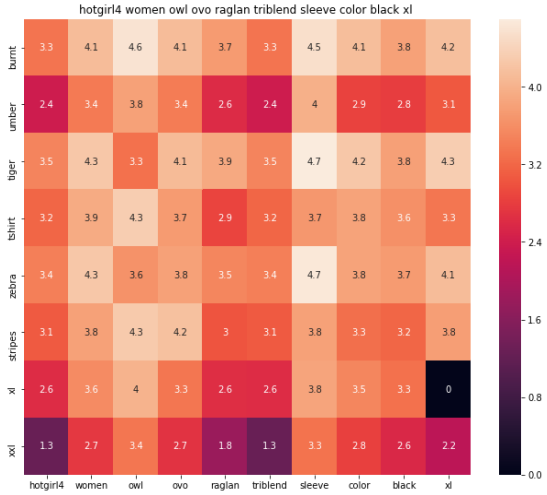
euclidean distance from given input image : 1.0669324

==========

==========

==========

==



ASIN : B01L8L73M2

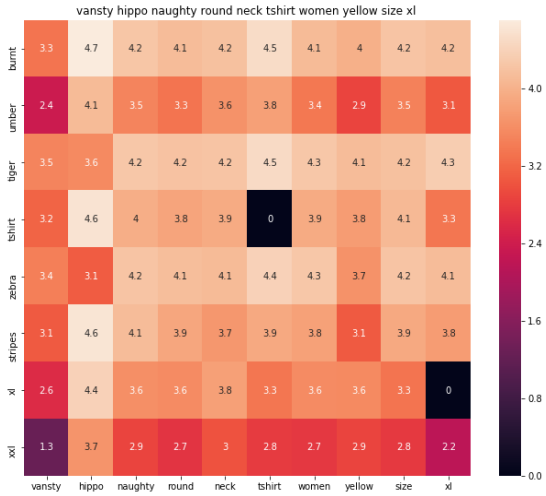
BRAND : Hotgirl4 Raglan Design

euclidean distance from given input image : 1.0731405

==========

==========

==



ASIN : B01EJS5H06

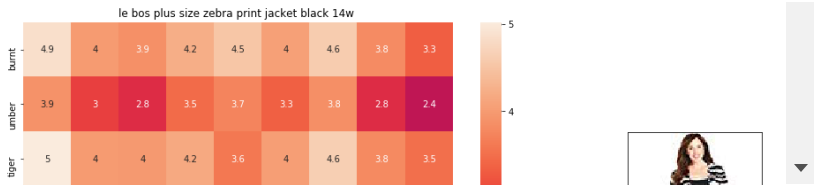
BRAND : Vansty

euclidean distance from given input image : 1.075719

==========

==========

==



ASIN : B01BO1XRK8

BRAND : Le Bos

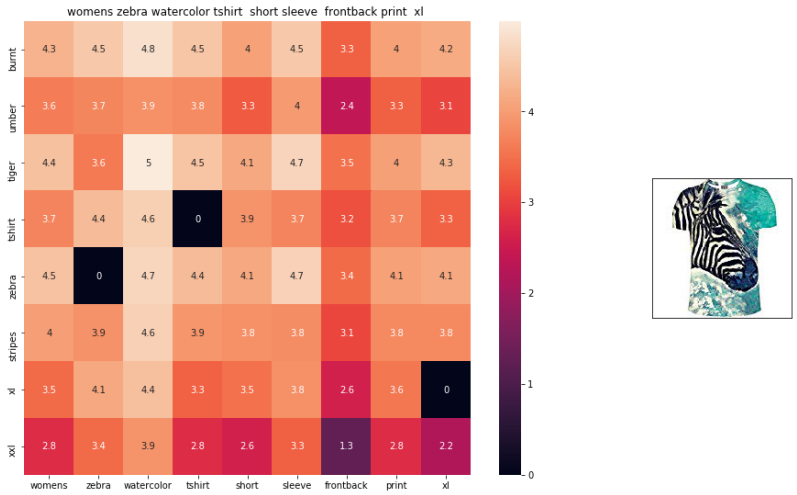
euclidean distance from given input image : 1.0839964

==========

==========

==========

==



ASIN : B072R2JXKW

BRAND : WHAT ON EARTH

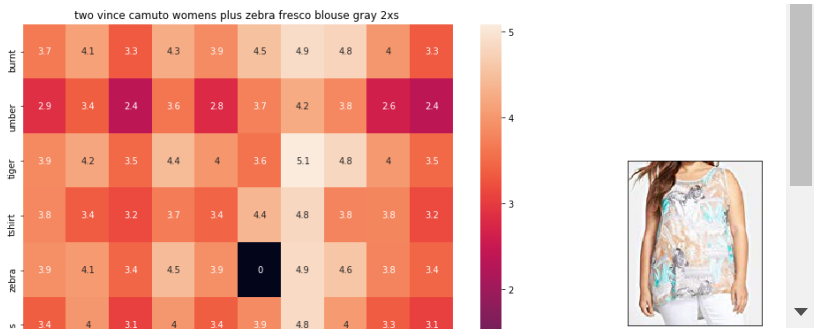
euclidean distance from given input image : 1.0842218

==========

==========

==========

==



ASIN : B074MJRGW6

BRAND : Two by Vince Camuto

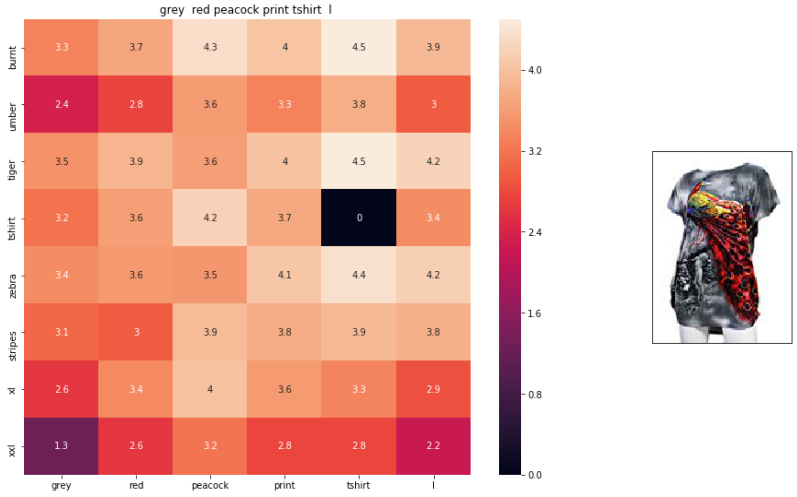
euclidean distance from given input image : 1.0895038

==========

==========

==========

==



ASIN : B00JXQCFRS

BRAND : Si Row

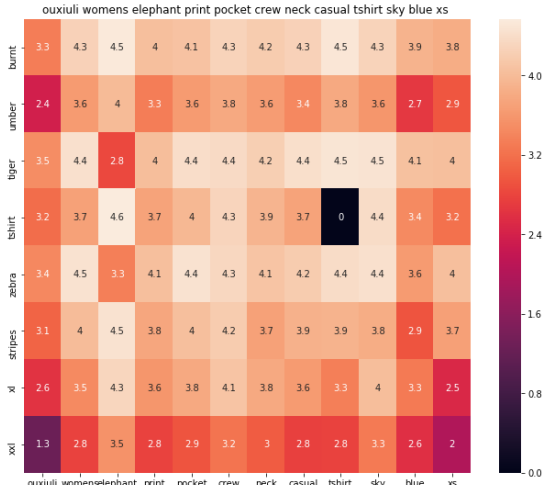
euclidean distance from given input image : 1.0900588

==========

==========

==========

========



ASIN : B01I53HU6K

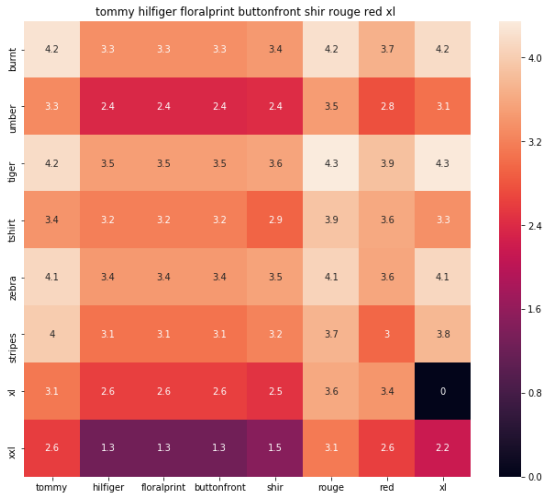
BRAND : ouxiuli

euclidean distance from given input image : 1.0920111

==========

==========

==========



ASIN : B0711NGTQM

BRAND : THILFIGER RTW

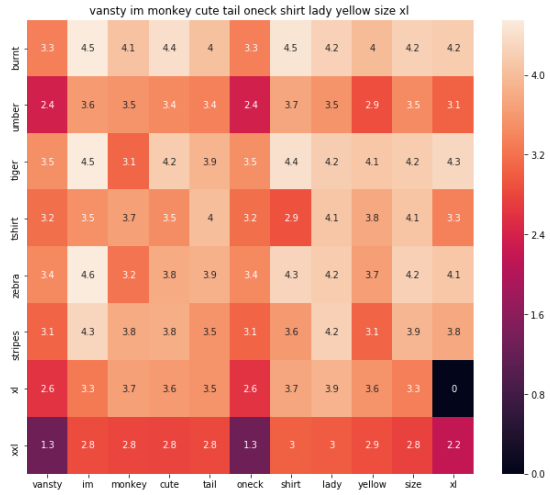
euclidean distance from given input image : 1.0923415

==========

==========

==========

========



ASIN : B01EFSLO8Y

BRAND : Vansty

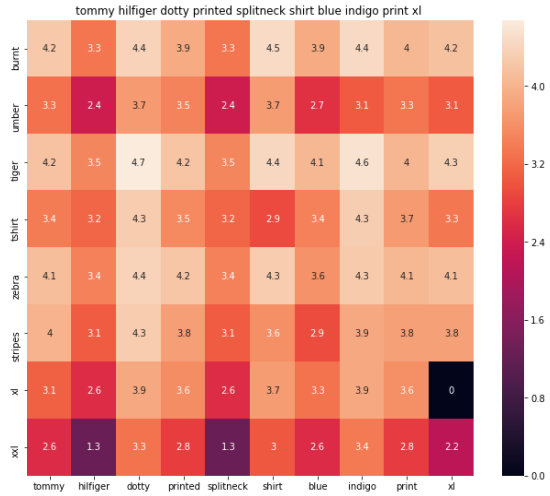
euclidean distance from given input image : 1.0934004

==========

==========

==========

==



ASIN : B0716TVWQ4

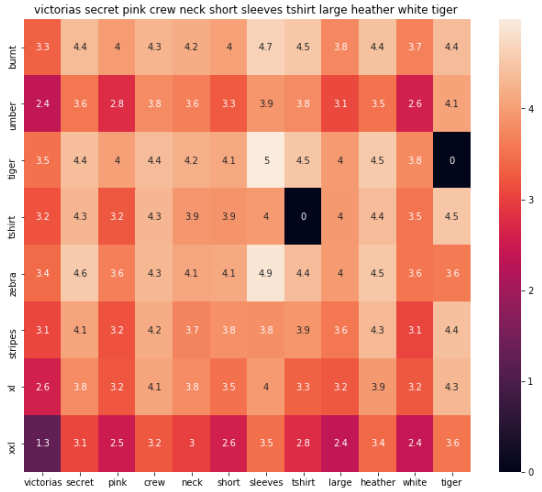
BRAND : THILFIGER RTW

euclidean distance from given input image : 1.0942024

==========

==========

==========



ASIN : B0716MVPGV

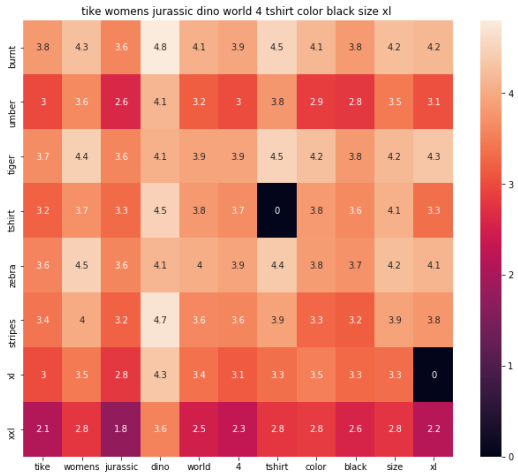
BRAND : V.Secret

euclidean distance from given input image : 1.0948304

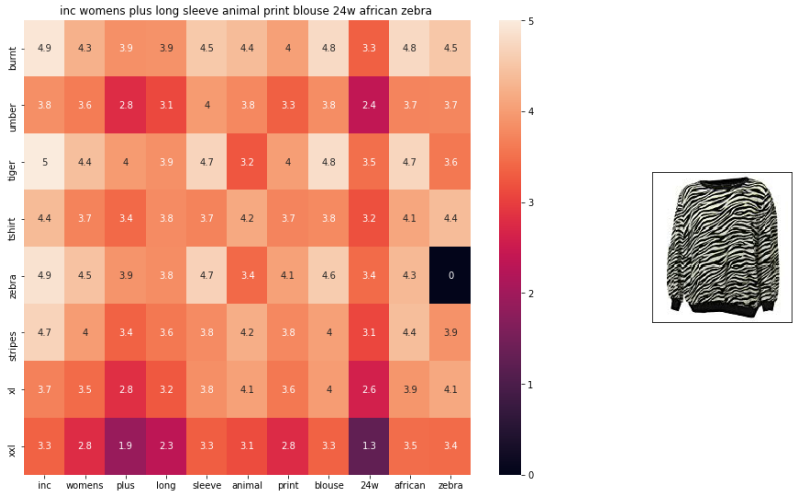
==========

==========

==========



ASIN : B016OPN4OI
BRAND : TIKE Fashions
euclidean distance from given input image : 1.0951275
==========
==========
==========
==



ASIN : B018WDJCUA
BRAND : INC - International Concepts Woman
euclidean distance from given input image : 1.0966892
==========
==========
==========
==
1 loop, best of 3: 17.4 s per loop

[9.4] IDF weighted Word2Vec for product similarity

In [0]:

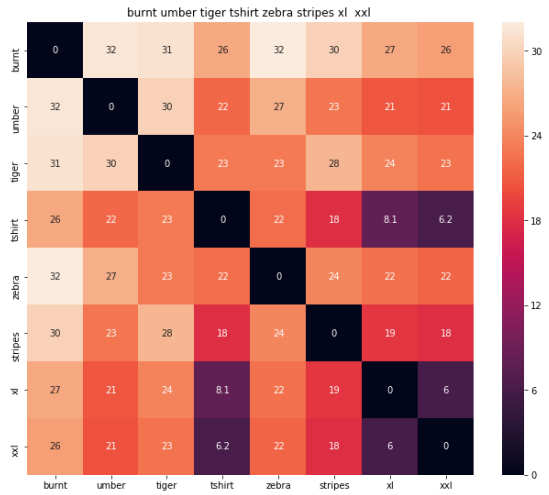
```
doc_id = 0
w2v_title_weight = []
# for every title we build a weighted vector re
presentation
for i in data['title']:
    w2v_title_weight.append(build_avg_vec(i, 30
0, doc_id,'weighted'))
    doc_id += 1
# w2v_title = np.array(# number of doc in courp
```

```
us * 300), each row corresponds to a doc
w2v_title_weight = np.array(w2v_title_weight)
```

In [51]:

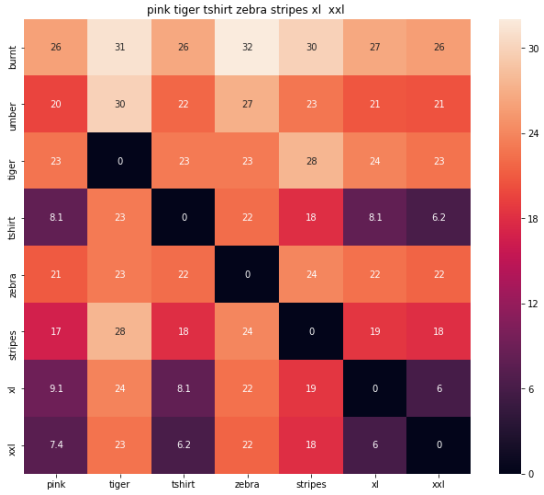
```
from PIL import Image
def weighted_w2v_model(doc_id, num_results):
    # doc_id: apparel's id in given corpus

    # pairwise_dist will store the distance fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    pairwise_dist = pairwise_distances(w2v_titl
e_weight, w2v_title_weight[doc_id].reshape(1,-1
))

    # np.argsort will return indices of 9 small
est 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 dista
ce's
    df_indices = list(data.index[indices])

    for i in range(0, len(indices)):
        heat_map_w2v(data['title'].loc[df_indic
es[0]],data['title'].loc[df_indices[i]], data[
'medium_image_url'].loc[df_indices[i]], indices
[0], indices[i], 'weighted')
```

```
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('Brand :',data['brand'].loc[df_in
dices[i]])
        print('euclidean distance from input :'
, pdists[i])
        print('='*125)

weighted_w2v_model(12566, 20)
#931
#12566
# in the give heat map, each cell contains the
 euclidean distance between words i, j
```



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

==========
==

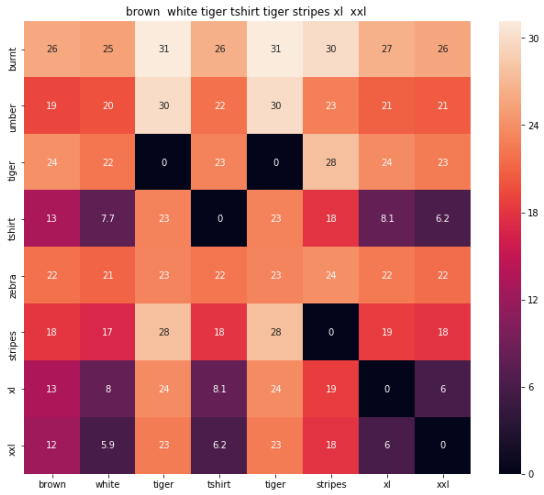


ASIN : B00JXQASS6

Brand : Si Row

euclidean distance from input : 4.0638866

==========
==========
==========
========



ASIN : B00JXQCWTO

Brand : Si Row

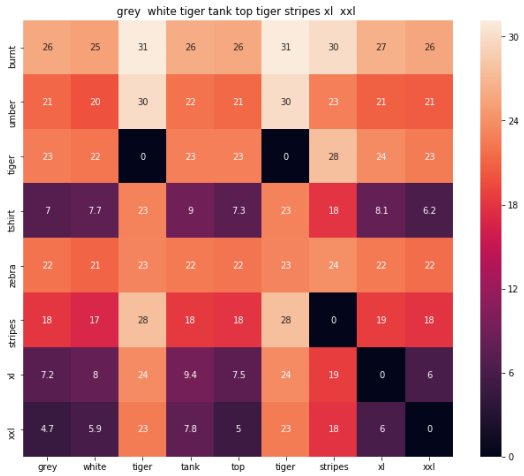
euclidean distance from input : 4.7709413

==========

==========

==========

==



ASIN : B00JXQAFZ2

Brand : Si Row

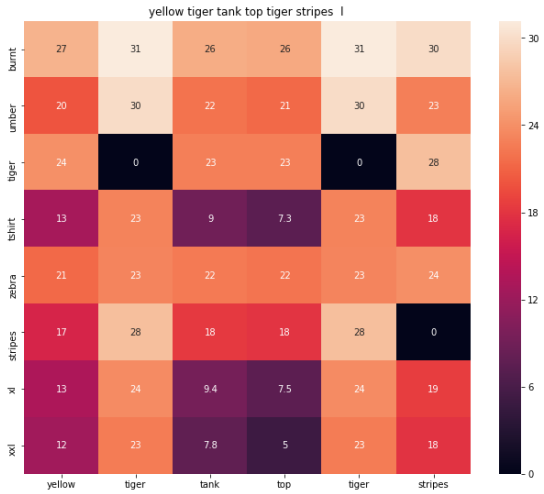
euclidean distance from input : 5.3601604

==========

==========

==========

==



ASIN : B00JXQAUWA

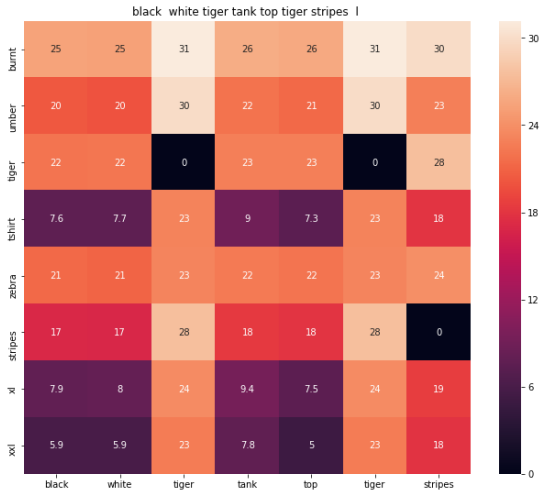
Brand : Si Row

euclidean distance from input : 5.6895227

==========

==========

==========



ASIN : B00JXQAO94

Brand : Si Row

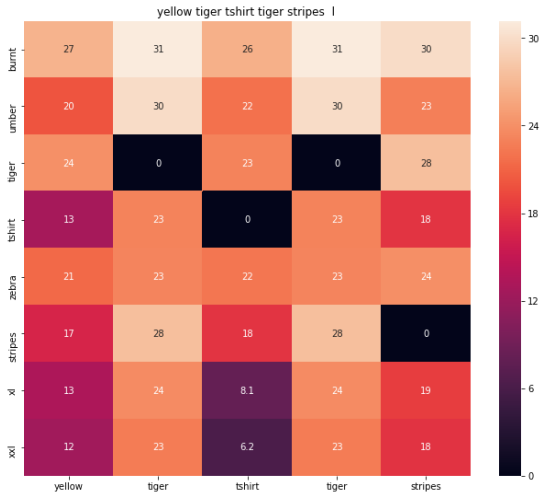
euclidean distance from input : 5.693021

==========

==========

==========

==



ASIN : B00JXQCUIC

Brand : Si Row

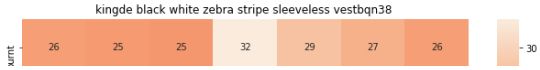
euclidean distance from input : 5.893442

==========

==========

==========

==



ASIN : B015H41F6G

Brand : KINGDE

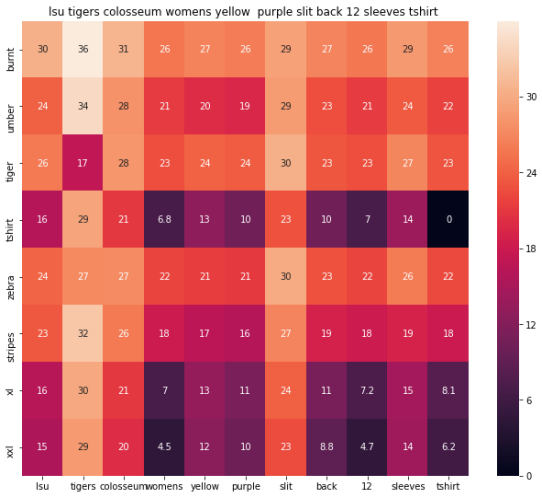
euclidean distance from input : 6.1329894

==========

==========

==========

==



ASIN : B073R5Q8HD

Brand : Colosseum

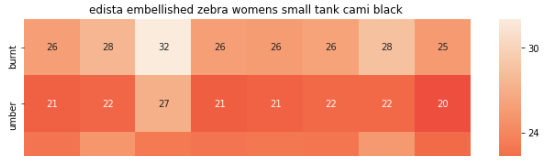
euclidean distance from input : 6.2567053

==========

==========

==========

==



ASIN : B074P8MD22

Brand : Edista

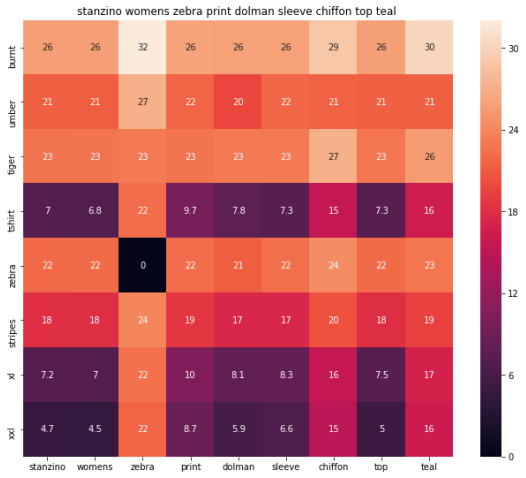
euclidean distance from input : 6.3922033

==========

==========

==========

==



ASIN : B00C0I3U3E

Brand : Stanzino

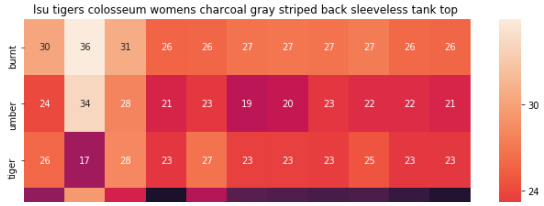
euclidean distance from input : 6.4149003

==========

==========

==========

==



ASIN : B073R4ZM7Y

Brand : Colosseum

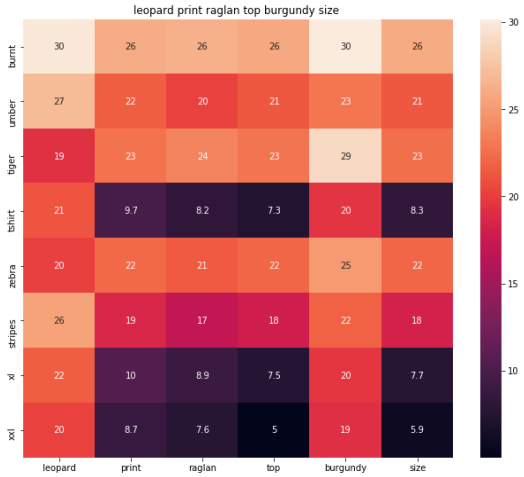
euclidean distance from input : 6.450959

==========

==========

==========

==



ASIN : B01C6ORLDQ

Brand : 1 Mad Fit

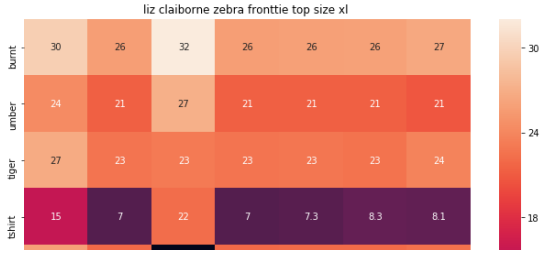
euclidean distance from input : 6.463409

==========

==========

==========

==



ASIN : B06XBY5QXL

Brand : Liz Claiborne

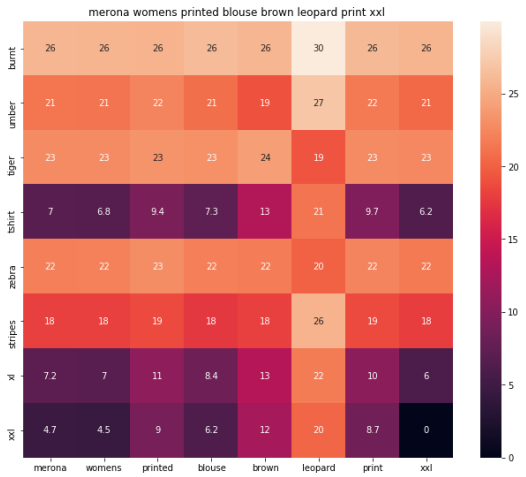
euclidean distance from input : 6.5392227

==========

==========

==========

==



ASIN : B071YF3WDD

Brand : Merona

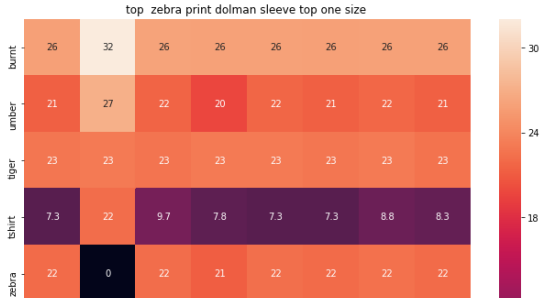
euclidean distance from input : 6.5755024

==========

==========

==========

==



ASIN : B00H8A6ZLI

Brand : Vivian's Fashions

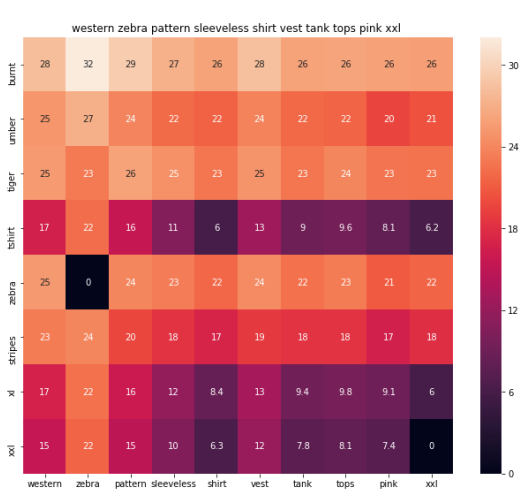
euclidean distance from input : 6.6382146

==========

==========

==========

==



ASIN : B00Z6HEXWI

Brand : Black Temptation

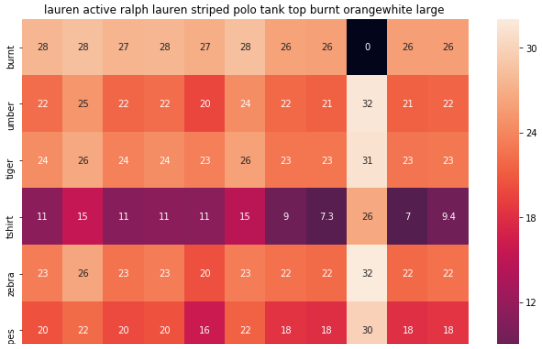
euclidean distance from input : 6.6607366

==========

==========

==========

==



ASIN : B00ILGH5OY

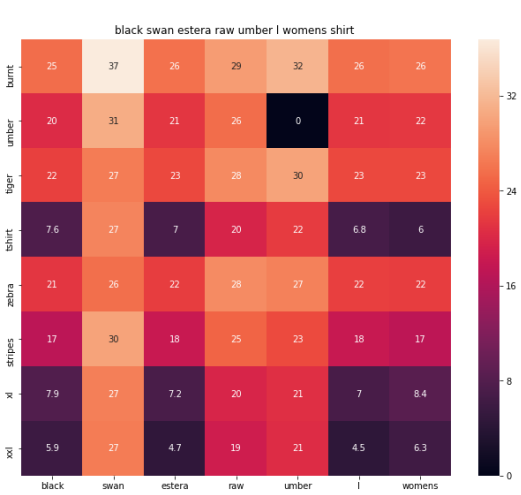
Brand : Ralph Lauren Active

euclidean distance from input : 6.6839046

==========

==========

==



ASIN : B06Y1VN8WQ

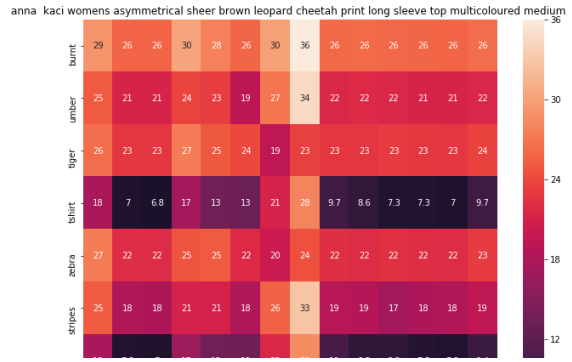
Brand : Black Swan

euclidean distance from input : 6.705763

==========

==========

==



ASIN : B00KSNTY7Y
Brand : Anna-Kaci
euclidean distance from input : 6.7061243
==========
==========
==========
==

[9.6] Weighted similarity using brand and color.

In [0]:

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

# replace spaces with hypen
brands = [x.replace(" ", "-") for x in data['br
and'].values]
types = [x.replace(" ", "-") for x in data['pro
duct_type_name'].values]
colors = [x.replace(" ", "-") for x in data['co
lor'].values]

brand_vectorizer = CountVectorizer()
```

```
brand_features = brand_vectorizer.fit_transform
(brands)

type_vectorizer = CountVectorizer()
type_features = type_vectorizer.fit_transform(t
ypes)

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

extra_features = hstack((brand_features, type_f
eatures, color_features)).tocsr()
```

In [0]:

```
from PIL import Image
def heat_map_w2v_brand(sentance1, sentance2, ur
l, 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 appar
el
    # df_id1: index of document1 in the data fr
ame
    # df_id2: index of document2 in the data fr
ame
    # 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 ti
tle
```

```
    s1_vec = get_word_vec(sentance1, doc_id1, m
odel)
    #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 ti
tle
    s2_vec = get_word_vec(sentance2, doc_id2, m
odel)

    # s1_s2_dist = np.array(#number of words in
title1 * #number of words in title2)
    # s1_s2_dist[i,j] = euclidean distance betw
een words i, j
    s1_s2_dist = get_distance(s1_vec, s2_vec)

    data_matrix = [['Asin','Brand', 'Color', 'P
roduct type'],
                   [data['asin'].loc[df_id1],brands
[doc_id1], colors[doc_id1], types[doc_id1]], #
 input apparel's features
                   [data['asin'].loc[df_id2],brands
[doc_id2], colors[doc_id2], types[doc_id2]]] #
 recommonded apparel's features

    colorscale = [[0, '#1d004d'],[.5, '#f2e5ff'
],[1, '#f2e5d1']] # to color the headings of ea
ch 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='simpl
e_table')

    # devide whole figure space into 25 * 1:10
```

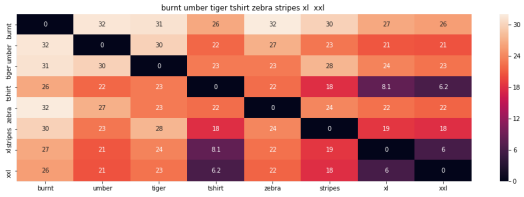
```
 grids
    gs = gridspec.GridSpec(25, 15)
    fig = plt.figure(figsize=(25,5))

    # in first 25*10 grids we plot heatmap
    ax1 = plt.subplot(gs[:, :-5])
    # ploting the heap map based on the pairwis
e distances
    ax1 = sns.heatmap(np.round(s1_s2_dist,6), a
nnot=True)
    # set the x axis labels as recommended appa
rels title
    ax1.set_xticklabels(sentance2.split())
    # set the y axis labels as input apparels t
itle
    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 [74]:

```
import numpy as np
from keras.preprocessing.image import ImageData
Generator
```

```
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dens
e
from keras import applications
from sklearn.metrics import pairwise_distances
import matplotlib.pyplot as plt
import requests
from PIL import Image
import pandas as pd
import pickle




from PIL import Image

def idf_w2v_brand(doc_id,num_results, w1, w2 ,
w3 = 0):
    # 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 fro
m given input apparel to all remaining apparels
    # the metric we used here is cosine, the co
side distance is mesured as K(X, Y) = <X, Y> /
 (||X||*||Y||)
    # http://scikit-learn.org/stable/modules/me
trics.html#cosine-similarity
    idf_w2v_dist  = pairwise_distances(w2v_titl
e_weight, w2v_title_weight[doc_id].reshape(1,-1
))
    ex_feat_dist = pairwise_distances(extra_fea
tures, extra_features[doc_id])
    image_features_dist = pairwise_distances(bo
ttleneck_features_train, bottleneck_features_tr
```

```
ain[doc_id].reshape(1,-1))

    pairwise_dist   = (w1 * idf_w2v_dist +  w2
* ex_feat_dist + w3 * image_features_dist)/floa
t(w1 + w2+ w3)

    # np.argsort will return indices of 9 small
est 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 dista
ce'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]], in
dices[0], indices[i],df_indices[0], df_indices[
i], 'weighted')
        print('ASIN :',data['asin'].loc[df_indi
ces[i]])
        print('Brand :',data['brand'].loc[df_in
dices[i]])
        print('euclidean distance from input :'
, pdists[i])
        print('='*125)

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



ASIN : B00JXQB5FQ

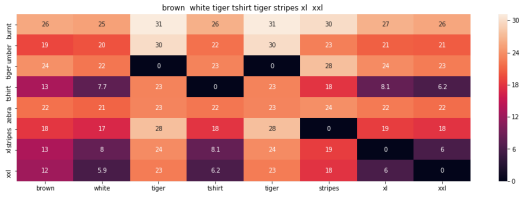
Brand : Si Row

euclidean distance from input : 0.0

==========

==========

==

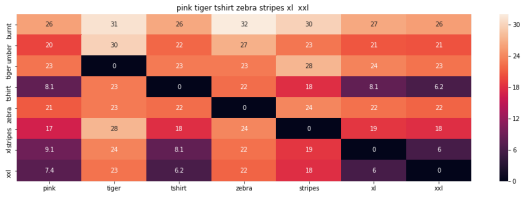


ASIN : B00JXQCWTO

Brand : Si Row

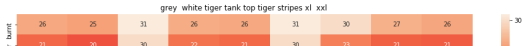
euclidean distance from input : 2.3854705
810546877

==========
==========
==========
==



ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 2.7390501
02414575

==========
==========
==========
==



ASIN : B00JXQAFZ2

Brand : Si Row

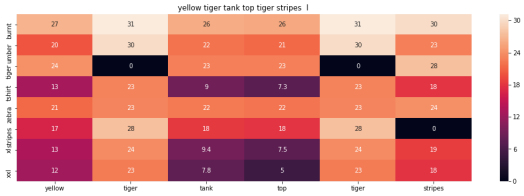
euclidean distance from input : 3.3871870 04270044

==========

==========

==========

==



ASIN : B00JXQAUWA

Brand : Si Row

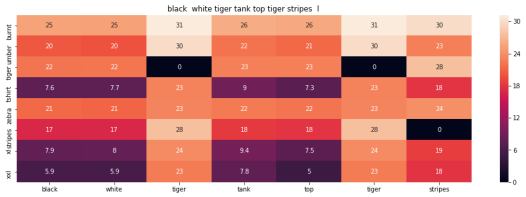
euclidean distance from input : 3.5518680 574316646

==========

==========

==========

==

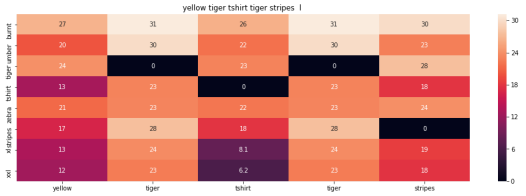


ASIN : B00JXQAO94

Brand : Si Row

euclidean distance from input : 3.5536170961279536

==========

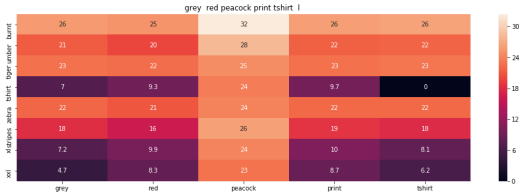


ASIN : B00JXQCUIC

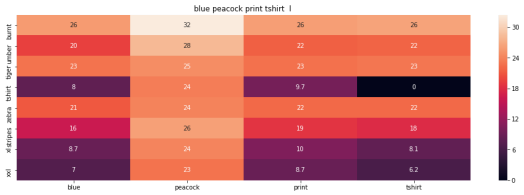
Brand : Si Row

euclidean distance from input : 3.6538278581518795

==========

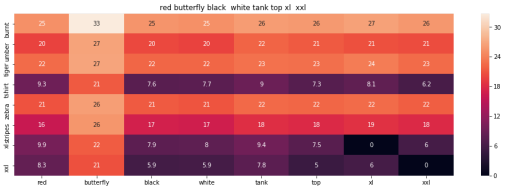


ASIN : B00JXQCFRS
Brand : Si Row
euclidean distance from input : 4.12881
1264218774
==========
==========
==========
========

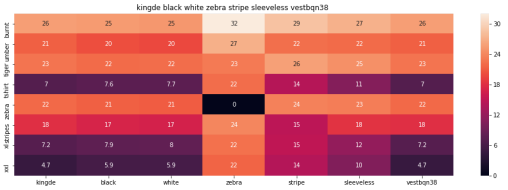


ASIN : B00JXQC8L6
Brand : Si Row
euclidean distance from input : 4.2039001
46665063
==========
==========

==========
==



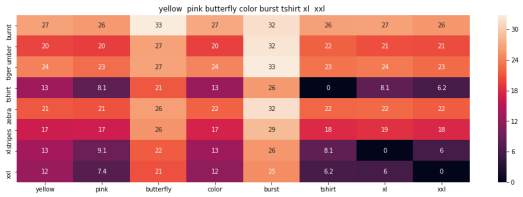
ASIN : B00JV63CW2
Brand : Si Row
euclidean distance from input : 4.2865863
80185571
==========
==========
==========
==



ASIN : B015H41F6G
Brand : KINGDE

euclidean distance from input : 4.3893704
06508858

==========
==========
==========
==



ASIN : B00JXQBBMI
Brand : Si Row
euclidean distance from input : 4.3979099
27548852

==========
==========
==========
==

ASIN : B073R5Q8HD

Brand : Colosseum

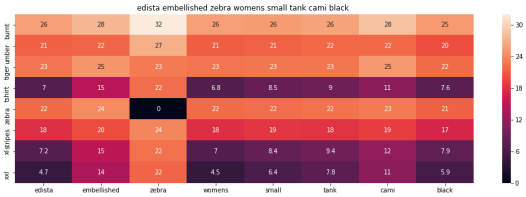
euclidean distance from input : 4.45122 8392959053

==========

==========

==========

========



ASIN : B074P8MD22

Brand : Edista

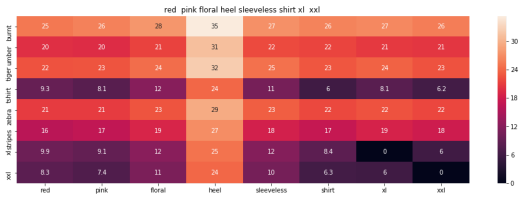
euclidean distance from input : 4.5189774 16396553

==========

==========

==========

==



ASIN : B00JV63QQE

Brand : Si Row

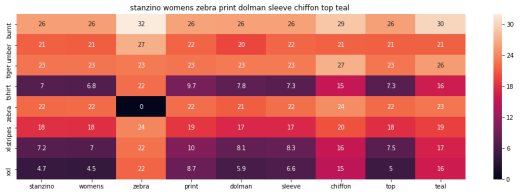
euclidean distance from input : 4.52937 4695004907

==========

==========

==========

========



ASIN : B00C0I3U3E

Brand : Stanzino

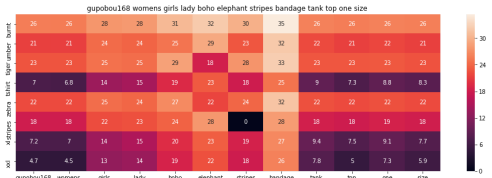
euclidean distance from input : 4.5303257 59292061

==========

==========

==========

==



ASIN : B01ER184O6

Brand : GuPoBoU168

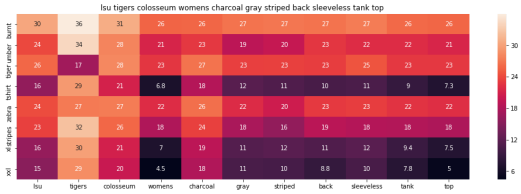
euclidean distance from input : 4.5468166 42558488

==========

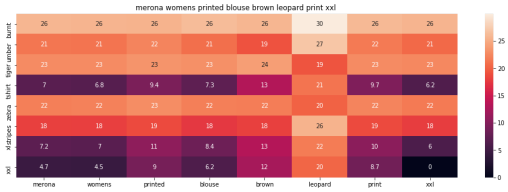
==========

==========

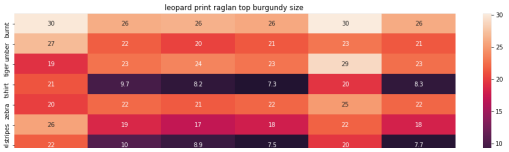
==



ASIN : B073R4ZM7Y
Brand : Colosseum
euclidean distance from input : 4.5483551
62978584
==========
==========
==========
==



ASIN : B071YF3WDD
Brand : Merona
euclidean distance from input : 4.6106266
62612374
==========
==========
==========
==



```
ASIN : B01C6ORLDQ
Brand : 1 Mad Fit
euclidean distance from input : 4.64591
7892817431
========================================
========================================
========================================
========
```

[10.2] Keras and Tensorflow to extract features

In [0]:

```
import numpy as np
from keras.preprocessing.image import ImageData
Generator
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dens
e
from keras import applications
from sklearn.metrics import pairwise_distances
import matplotlib.pyplot as plt
import requests
from PIL import Image
import pandas as pd
import pickle
```

In [76]:

```
# https://gist.github.com/fchollet/f35fbc80e066
a49d65f1688a7e99f069
# Code reference: https://blog.keras.io/buildin
```

```
g-powerful-image-classification-models-using-very-little-data.html



# This code takes 40 minutes to run on a modern GPU (graphics card)
# like Nvidia  1050.
# GPU (NVidia 1050): 0.175 seconds per image

# This codse takes 160 minutes to run on a high end i7 CPU
# CPU (i7): 0.615 seconds per image.

#Do NOT run this code unless you want to wait a few hours for it to generate output

# each image is converted into 25088 length dense-vector


'''
# dimensions of our images.
img_width, img_height = 224, 224

top_model_weights_path = 'bottleneck_fc_model.h5'
train_data_dir = 'images2/'
nb_train_samples = 16042
epochs = 50
batch_size = 1


def save_bottlebeck_features():

    #Function to compute VGG-16 CNN for image f
```

```
eature extraction.

    asins = []
    datagen = ImageDataGenerator(rescale=1. / 2
55)

    # build the VGG16 network
    model = applications.VGG16(include_top=Fals
e, weights='imagenet')
    generator = datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode=None,
        shuffle=False)

    for i in generator.filenames:
        asins.append(i[2:-5])

    bottleneck_features_train = model.predict_g
enerator(generator, nb_train_samples // batch_s
ize)
    bottleneck_features_train = bottleneck_feat
ures_train.reshape((16042,25088))

    np.save(open('16k_data_cnn_features.npy',
 'wb'), bottleneck_features_train)
    np.save(open('16k_data_cnn_feature_asins.np
y', 'wb'), np.array(asins))


save_bottlebeck_features()

'''
```

Out[76]: "\n# dimensions of our images.\nimg_widt

```
h, img_height = 224, 224\n\ntop_model_wei
ghts_path = 'bottleneck_fc_model.h5'\ntra
in_data_dir = 'images2/'\nnb_train_sample
s = 16042\nepochs = 50\nbatch_size = 1\n
\n\ndef save_bottlebeck_features():\n
\n    #Function to compute VGG-16 CNN for
image feature extraction.\n    \n    asin
s = []\n    datagen = ImageDataGenerator
(rescale=1. / 255)\n    \n    # build the
VGG16 network\n    model = applications.V
GG16(include_top=False, weights='imagene
t')\n    generator = datagen.flow_from_di
rectory(\n        train_data_dir,\n
target_size=(img_width, img_height),\n
batch_size=batch_size,\n        class_mod
e=None,\n        shuffle=False)\n\n    fo
r i in generator.filenames:\n        asin
s.append(i[2:-5])\n\n    bottleneck_featu

res_train = model.predict_generator(gener
ator, nb_train_samples // batch_size)\n
bottleneck_features_train = bottleneck_fe
atures_train.reshape((16042,25088))\n
\n    np.save(open('16k_data_cnn_feature
s.npy', 'wb'), bottleneck_features_train)
\n    np.save(open('16k_data_cnn_feature_
asins.npy', 'wb'), np.array(asins))\n
\n\nsave_bottlebeck_features()\n\n"
```

[10.3] Visual features based product similarity.

In [79]:
```
#load the features and corresponding ASINS inf
```

```
o.
bottleneck_features_train = np.load('/content/drive/My Drive/Applied_AI_Workshop_Code_Data/16k_data_cnn_features.npy')
asins = np.load('/content/drive/My Drive/Applied_AI_Workshop_Code_Data/16k_data_cnn_feature_asins.npy')
asins = list(asins)

# load the original 16K dataset
data = pd.read_pickle('/content/drive/My Drive/Applied_AI_Workshop_Code_Data/pickels/16k_apperal_data_preprocessed')
df_asins = list(data['asin'])


from IPython.display import display, Image, SVG, Math, YouTubeVideo


#get similar products using CNN features (VGG-16)
def get_similar_products_cnn(doc_id, num_results):
    doc_id = asins.index(df_asins[doc_id])
    pairwise_dist = pairwise_distances(bottleneck_features_train, bottleneck_features_train[doc_id].reshape(1,-1))

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

    for i in range(len(indices)):
        rows = data[['medium_image_url','title'
```

```
]].loc[data['asin']==asins[indices[i]]]
        for indx, row in rows.iterrows():
            display(Image(url=row['medium_image
_url'], embed=True))
            print('Product Title: ', row['titl
e'])
            print('Euclidean Distance from inpu
t image:', pdists[i])
            print('Amazon Url: www.amzon.com/d
p/'+ asins[indices[i]])

get_similar_products_cnn(12566, 20)
```

```
Product Title:  burnt umber tiger tshirt
zebra stripes xl  xxl
Euclidean Distance from input image: 6.32
596e-06
Amazon Url: www.amzon.com/dp/B00JXQB5FQ
```

```
Product Title:  pink tiger tshirt zebra s
tripes xl  xxl
```

Euclidean Distance from input image: 30.0
5017

Product Title: yellow tiger tshirt tiger stripes l
Euclidean Distance from input image: 41.2
61116
Amazon Url: www.amzon.com/dp/B00JXQCUIC

Product Title: brown white tiger tshirt tiger stripes xl xxl
Euclidean Distance from input image: 44.0
00156
Amazon Url: www.amzon.com/dp/B00JXQCWTO

Product Title: kawaii pastel tops tees p
ink flower design
Euclidean Distance from input image: 47.3
8248
Amazon Url: www.amzon.com/dp/B071FCWD97

Product Title: womens thin style tops
tees pastel watermelon print
Euclidean Distance from input image: 4
7.71842
Amazon Url: www.amzon.com/dp/B01JUNHBRM

Product Title: kawaii pastel tops tees b
aby blue flower design
Euclidean Distance from input image: 47.9

0206

Product Title: edv cheetah run purple mu
lti xl
Euclidean Distance from input image: 48.0
46482
Amazon Url: www.amzon.com/dp/B01CUPYBM0

Product Title: danskin womens vneck loos
e performance tee xsmall pink ombre
Euclidean Distance from input image: 48.1
01837
Amazon Url: www.amzon.com/dp/B01F7PHXY8

Product Title: summer alpaca 3d pastel c
asual loose tops tee design
Euclidean Distance from input image: 48.1
18866
Amazon Url: www.amzon.com/dp/B01I80A93C

Product Title: miss chievous juniors str
iped peplum tank top medium shadowpeach
Euclidean Distance from input image: 48.1
3122
Amazon Url: www.amzon.com/dp/B0177DM70S

Product Title: red pink floral heel sle
eveless shirt xl xxl
Euclidean Distance from input image: 48.1
6945
Amazon Url: www.amzon.com/dp/B00JV63QQE

Product Title: moana logo adults hot v neck shirt black xxl
Euclidean Distance from input image: 48.256786
Amazon Url: www.amzon.com/dp/B01LX6H43D

Product Title: abaday multicolor cartoon cat print short sleeve longline shirt large
Euclidean Distance from input image: 48.265686
Amazon Url: www.amzon.com/dp/B01CR57YY0

Product Title: kawaii cotton pastel tops tees peach pink cactus design

Euclidean Distance from input image: 48.362602
Amazon Url: www.amzon.com/dp/B071WYLBZS

Product Title: chicago chicago 18 shirt women pink
Euclidean Distance from input image: 48.383606
Amazon Url: www.amzon.com/dp/B01GXAZTRY

Product Title: yichun womens tiger printed summer tshirts tops
Euclidean Distance from input image: 48.449356
Amazon Url: www.amzon.com/dp/B010NN9RXO

Product Title: nancy lopez whimsy short sleeve whiteblacklemon drop xs
Euclidean Distance from input image: 48.47889
Amazon Url: www.amzon.com/dp/B01MPX6IDX

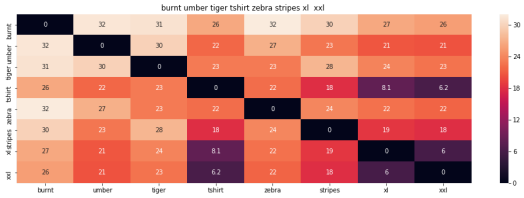
Product Title: womens tops tees pastel p each ice cream cone print
Euclidean Distance from input image: 48.557957
Amazon Url: www.amzon.com/dp/B0734GRKZL

Product Title: uswomens mary j blige wit hout tshirts shirt
Euclidean Distance from input image: 48.6

14372

Eqaul Weightage

In [84]:
```
idf_w2v_brand(12566,20, 5,5,5)
```



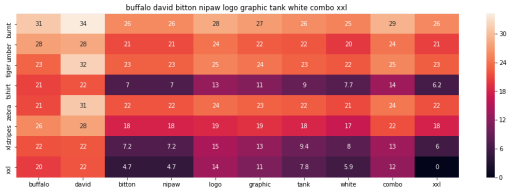
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 2.5030795465378714e-06
==========
==========
==========
==

ASIN : B06XBHNM7J

Brand : Xhilaration

euclidean distance from input : 15.982580820960319

==========

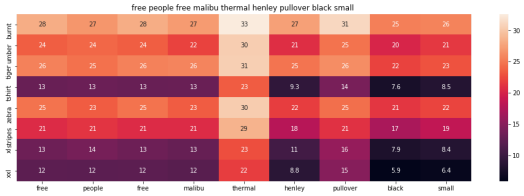


ASIN : B018H5AZXQ

Brand : Buffalo

euclidean distance from input : 16.603902435543652

==========

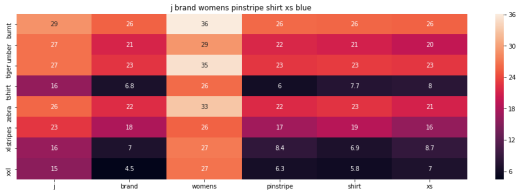


ASIN : B074MXY984

Brand : We The Free

euclidean distance from input : 16.931583150227866

==========



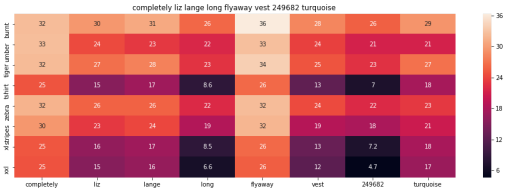
ASIN : B06XYP1X1F

Brand : J Brand Jeans

euclidean distance from input : 16.955521793298313

==========

==========



ASIN : B074LTBWSW

Brand : Liz Lange

euclidean distance from input : 16.957035064938182

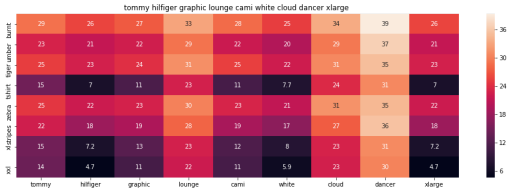
==========

==========

==

ASIN : B01L7ROZNC

Brand : Bila
euclidean distance from input : 17.0947
87383636113
==========
==========
==========
========



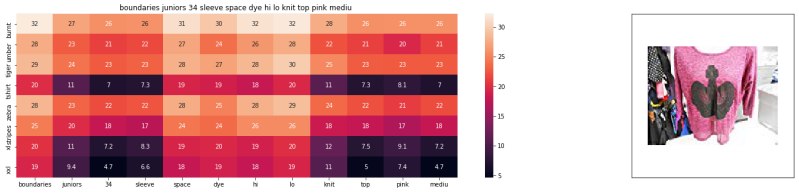
ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 17.281140
028568494
==========
==========
==========
==

ASIN : B01FJVZST2

Brand : KONGYII

euclidean distance from input : 17.551688170989628

==========

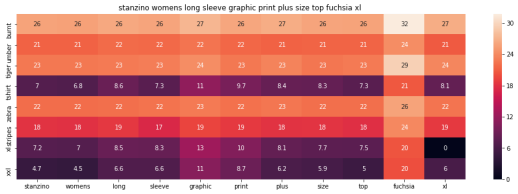


ASIN : B01EXXFS4M

Brand : No Boundaries

euclidean distance from input : 17.564554850501334

==========

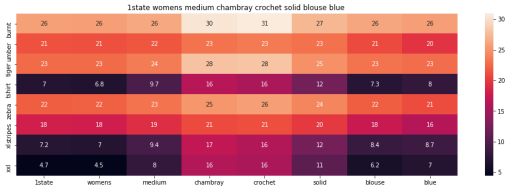


ASIN : B00DP4VHWI

Brand : Stanzino

euclidean distance from input : 17.585503682057336

==========

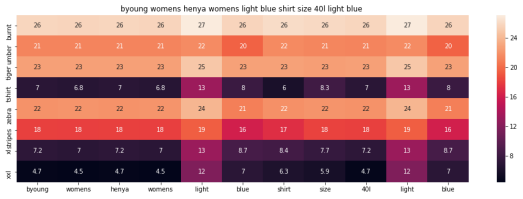


ASIN : B074MK6LV2

Brand : 1.State

euclidean distance from input : 17.596073190927778

==========

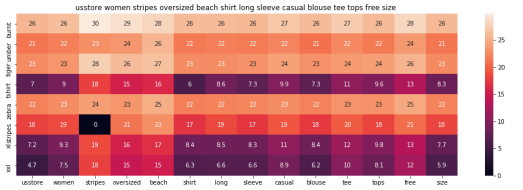


ASIN : B06Y41MRCH

Brand : Byoung

euclidean distance from input : 17.624775 1874154

==========



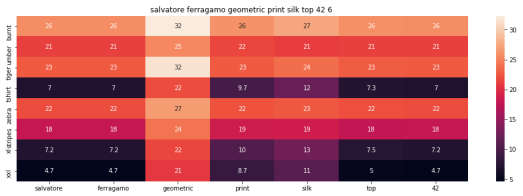
ASIN : B01DNNI1RO

Brand : Usstore

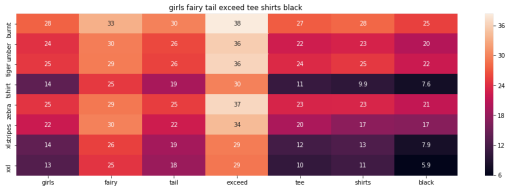
euclidean distance from input : 17.655587260169302

==========

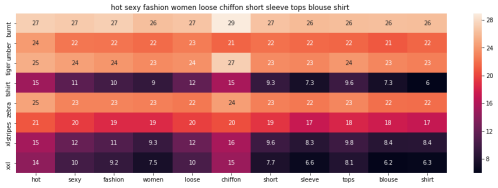
==========
==========
========



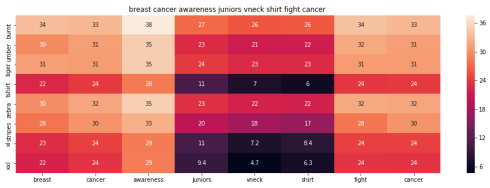
ASIN : B0756JTS1F
Brand : Salvatore Ferragamo
euclidean distance from input : 17.681511688232423
==========
==========
==========
==



ASIN : B01L9F153U
Brand : ATYPEMX
euclidean distance from input : 17.6936
3178467778
==========
==========
==========
========



ASIN : B00JMAASRO
Brand : Wotefusi
euclidean distance from input : 17.705771
677255903
==========
==========
==========
==

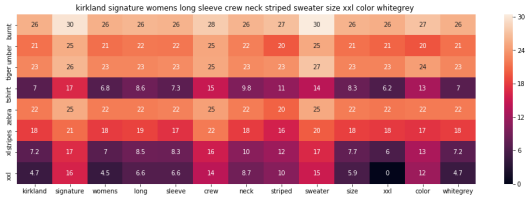


ASIN : B016CU40IY

Brand : Juiceclouds

euclidean distance from input : 17.709564 50359054

==========

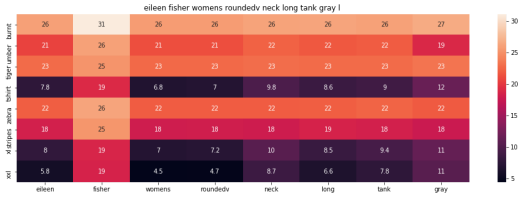


ASIN : B06XTPC3FP

Brand : Kirkland Signature

euclidean distance from input : 17.753966 776770866

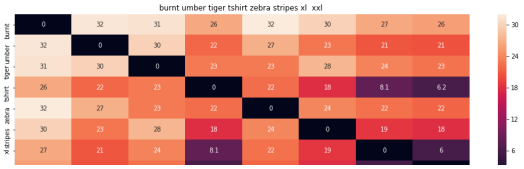
==========



ASIN : B01N2JSRDM
Brand : Eileen Fisher
euclidean distance from input : 17.801963
80639326
==========
==========
==========
==

When Image feature has maximum weightage than 'Title' and 'Brand & Color'

In [85]:
```
idf_w2v_brand(12566,20, 5,5,50)
```

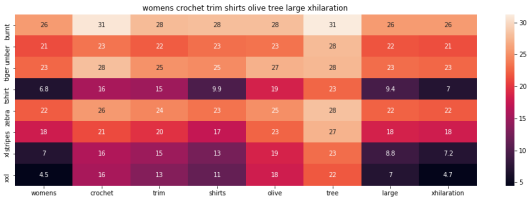


ASIN : B00JXQB5FQ

Brand : Si Row

euclidean distance from input : 6.257698987610638e-06

==========

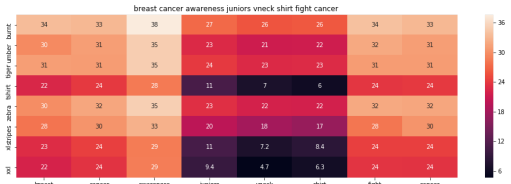


ASIN : B06XBHNM7J

Brand : Xhilaration

euclidean distance from input : 31.886473401447763

==========



ASIN : B016CU40IY

Brand : Juiceclouds

euclidean distance from input : 33.162233

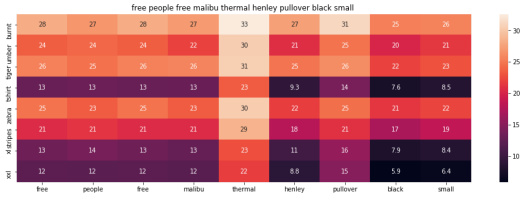
04484295

==========

==========

==========

==



ASIN : B074MXY984

Brand : We The Free

euclidean distance from input : 33.7560

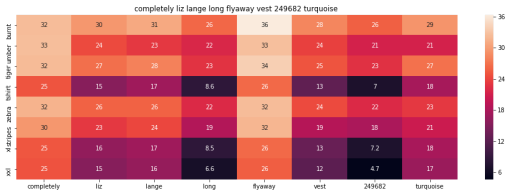
43370564775

==========

==========

==========

========



ASIN : B074LTBWSW

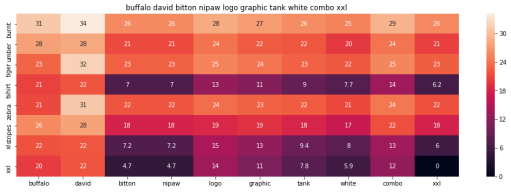
Brand : Liz Lange

euclidean distance from input : 33.762406 34924236

==========

==========

==========

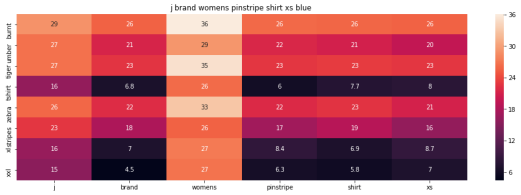


ASIN : B018H5AZXQ

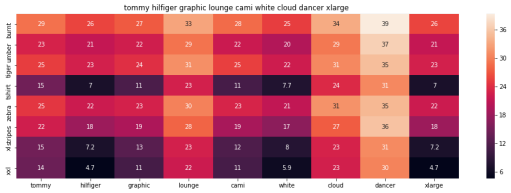
Brand : Buffalo

euclidean distance from input : 33.8225 35769204926

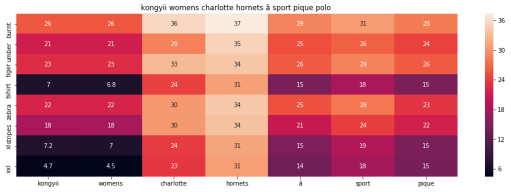
==========
==========
==========
========



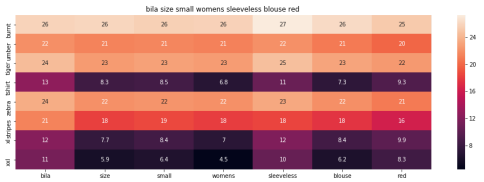
ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 33.931834448161815
==========
==========
==========
==



ASIN : B01BMSFYW2
Brand : igertommy hilf
euclidean distance from input : 34.120533
21677754
==========
==========
==========
==



ASIN : B01FJVZST2
Brand : KONGYII
euclidean distance from input : 34.933592
615664075
==========
==========
==========
==



ASIN : B01L7ROZNC

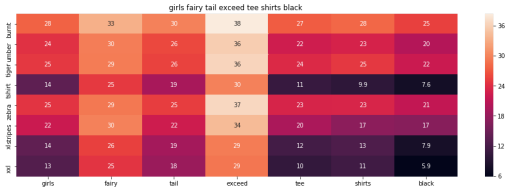
Brand : Bila

euclidean distance from input : 35.051030 93200929

==========

==========

==



ASIN : B01L9F153U

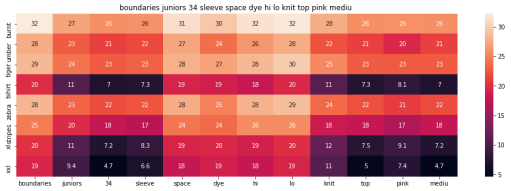
Brand : ATYPEMX

euclidean distance from input : 35.613362 83101645

==========

==========

==



ASIN : B01EXXFS4M

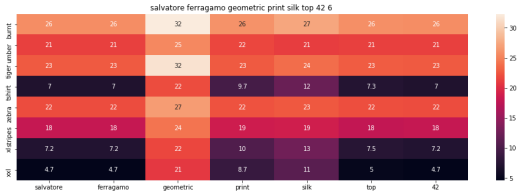
Brand : No Boundaries

euclidean distance from input : 35.658759562234714

==========

==========

==========



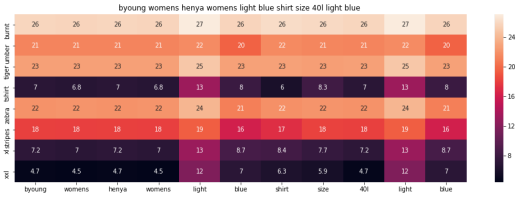
ASIN : B0756JTS1F

Brand : Salvatore Ferragamo

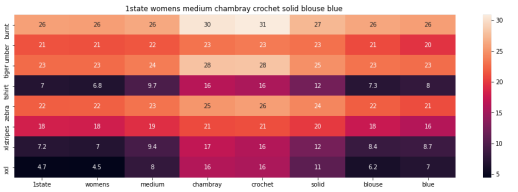
euclidean distance from input : 35.82349681854248

==========

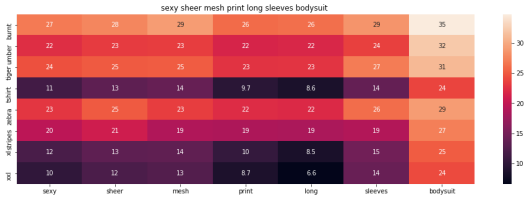
==========
==========
========



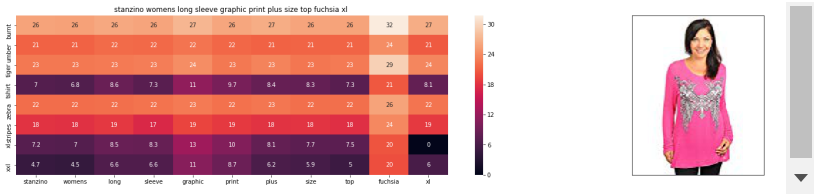
ASIN : B06Y41MRCH
Brand : Byoung
euclidean distance from input : 35.866533
40663575
==========
==========
==========
==



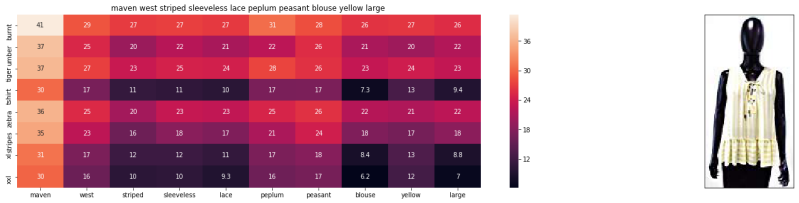
ASIN : B074MK6LV2
Brand : 1.State
euclidean distance from input : 35.9361299615503
==========
==========
==========
========



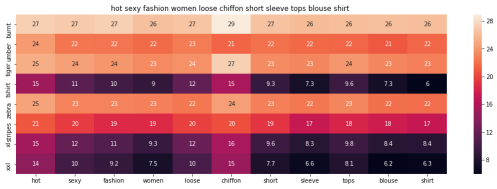
ASIN : B074Z5C98D
Brand : Ariella's closet
euclidean distance from input : 36.04980732251578
==========
==========
==========
==



ASIN : B00DP4VHWI
Brand : Stanzino
euclidean distance from input : 36.178742
60956056
==========
==========
==========
==



ASIN : B01M8GB3AL
Brand : Maven West
euclidean distance from input : 36.211964
98876873
==========
==========
==========
==

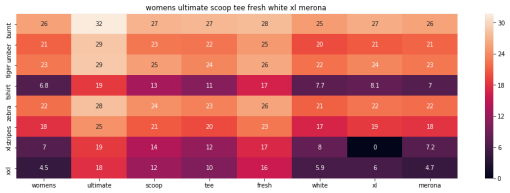


ASIN : B00JMAASRO

Brand : Wotefusi

euclidean distance from input : 36.230698341270354

==========



ASIN : B01G7XE50E

Brand : Merona

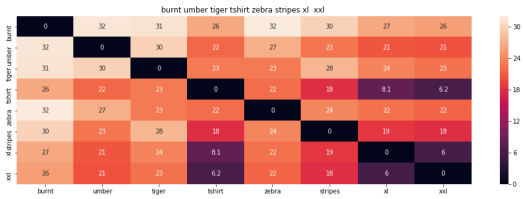
euclidean distance from input : 36.258359855552584

==========

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

When 'Brand & Color' has maximum weightage than 'Image feature'

In [86]:
```
idf_w2v_brand(12566,20, 1,10,1)
```



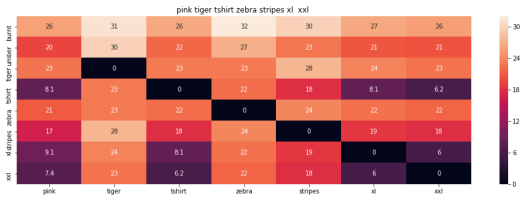
```
ASIN : B00JXQB5FQ
Brand : Si Row
euclidean distance from input : 6.2576990
93718353e-07
==========================================
==========================================
==========================================
==
```

ASIN : B00JXQCWTO

Brand : Si Row

euclidean distance from input : 5.0265657504399615

==========

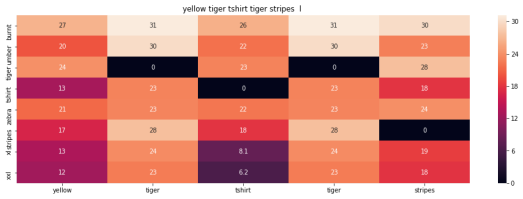


ASIN : B00JXQASS6

Brand : Si Row

euclidean distance from input : 5.559652805629516

==========



ASIN : B00JXQCUIC

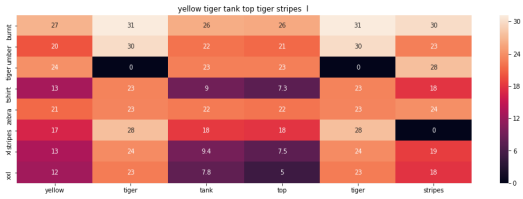
Brand : Si Row

euclidean distance from input : 6.0177200 63828572

==========

==========

==



ASIN : B00JXQAUWA

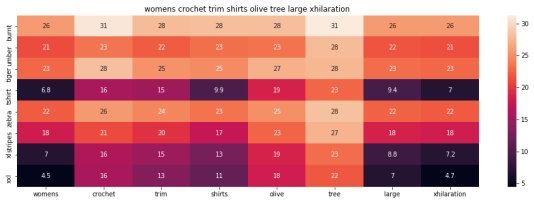
Brand : Si Row

euclidean distance from input : 6.1092126 372487785

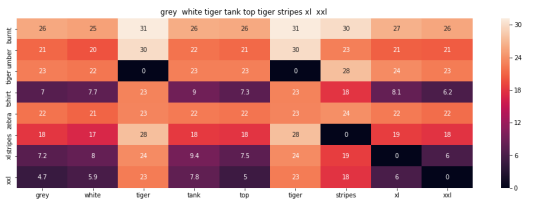
==========

==========

==========
==

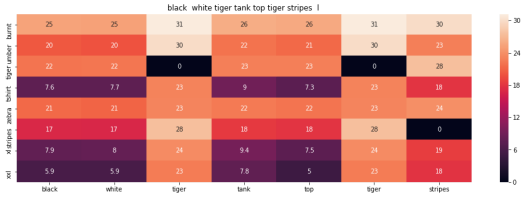


ASIN : B06XBHNM7J
Brand : Xhilaration
euclidean distance from input : 6.1169654931687205
==========
==========
==

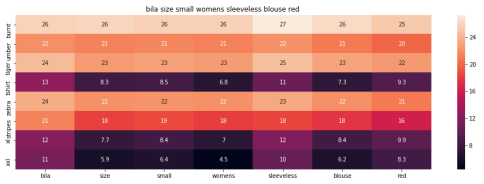


ASIN : B00JXQAFZ2

Brand : Si Row
euclidean distance from input : 6.11907
5735711202
==========
==========
==========
========



ASIN : B00JXQAO94
Brand : Si Row
euclidean distance from input : 6.2553724
45089762
==========
==========
==========
==

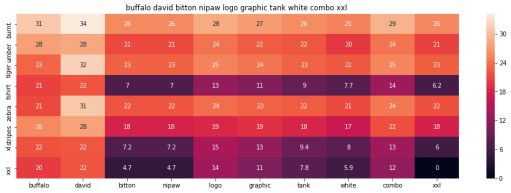


ASIN : B01L7ROZNC

Brand : Bila

euclidean distance from input : 6.2580104 8815319

==========

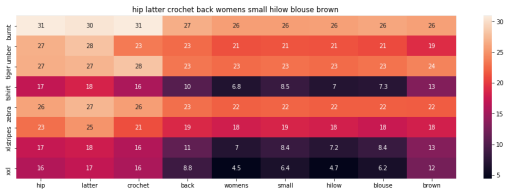


ASIN : B018H5AZXQ

Brand : Buffalo

euclidean distance from input : 6.2722960 716548455

==========



ASIN : B074MJN1K9

Brand : Hip

euclidean distance from input : 6.350508463839546

==========

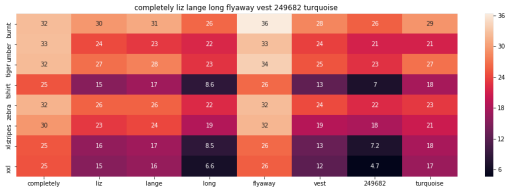
ASIN : B06XK2ZRFH

Brand : Acquaa

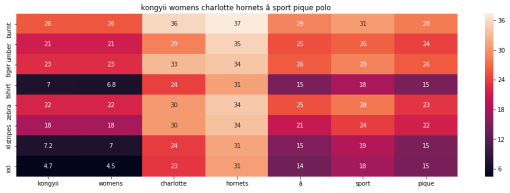
euclidean distance from input : 6.354369446856668

==========

==========
========

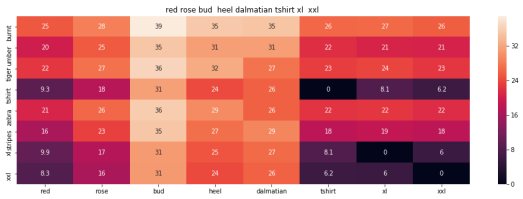


ASIN : B074LTBWSW
Brand : Liz Lange
euclidean distance from input : 6.3605790938996165
==========
==========
==

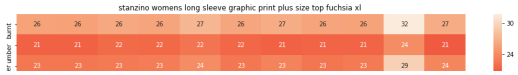


ASIN : B01FJVZST2
Brand : KONGYII

euclidean distance from input : 6.37223
5398889274
==========
==========
==========
========



ASIN : B00JXQABB0
Brand : Si Row
euclidean distance from input : 6.3803722
86144042
==========
==========
==========
==



ASIN : B00DP4VHWI

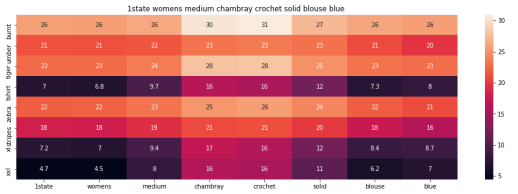
Brand : Stanzino

euclidean distance from input : 6.3806893 64076347

==========

==========

==



ASIN : B074MK6LV2

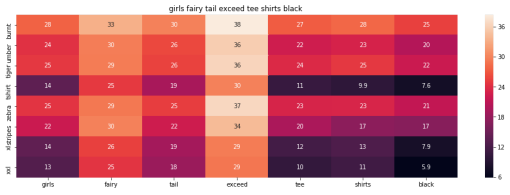
Brand : 1.State

euclidean distance from input : 6.3833317 9692496

==========

==========

==

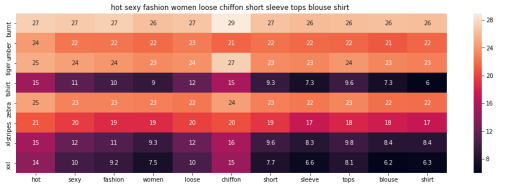


ASIN : B01L9F153U

Brand : ATYPEMX

euclidean distance from input : 6.407721381784172

==========



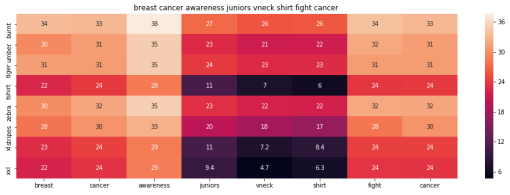
ASIN : B00JMAASRO

Brand : Wotefusi

euclidean distance from input : 6.410756450296135

==========

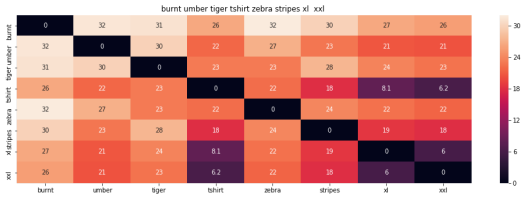
==========



ASIN : B016CU40IY
Brand : Juiceclouds
euclidean distance from input : 6.411704561512362
==========
==========
==========
==

When Image feature has maximum wieghtage, Title has 2nd highest weightage and 'Brand Color' has minimum weightage

In [87]:
```
idf_w2v_brand(12566,20, 5,1,10)
```

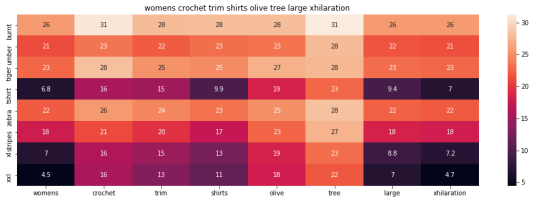


ASIN : B00JXQB5FQ

Brand : Si Row

euclidean distance from input : 4.6932741497585084e-06

==========

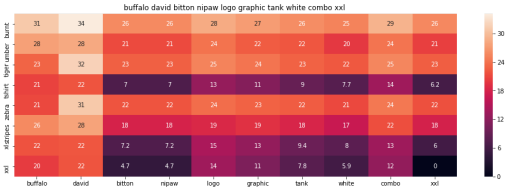


ASIN : B06XBHNM7J

Brand : Xhilaration

euclidean distance from input : 25.897741365477913

==========



ASIN : B018H5AZXQ

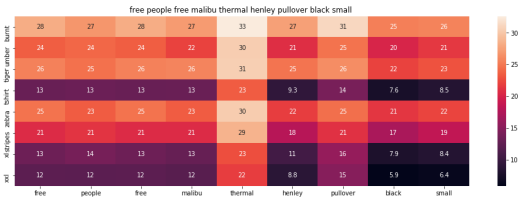
Brand : Buffalo

euclidean distance from input : 27.222201395079964

==========

==========

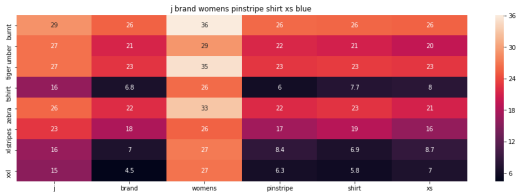
==



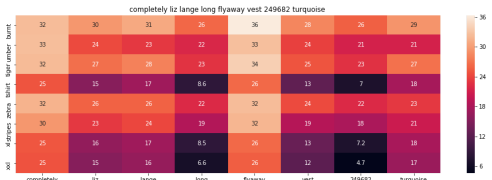
ASIN : B074MXY984

Brand : We The Free

euclidean distance from input : 27.4246
70696258545
==========
==========
==========
========



ASIN : B06XYP1X1F
Brand : J Brand Jeans
euclidean distance from input : 27.477296
856629177
==========
==========
==========
==

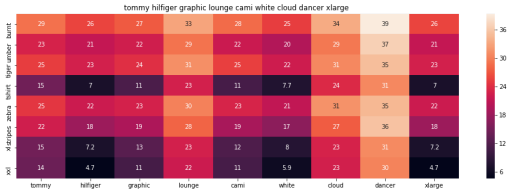


ASIN : B074LTBWSW

Brand : Liz Lange

euclidean distance from input : 27.491425085112922

==========

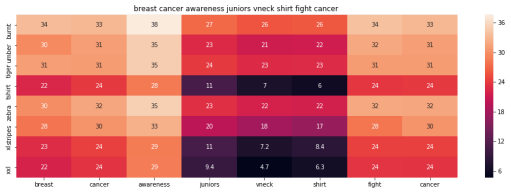


ASIN : B01BMSFYW2

Brand : igertommy hilf

euclidean distance from input : 27.827270296799465

==========



ASIN : B016CU40IY

Brand : Juiceclouds

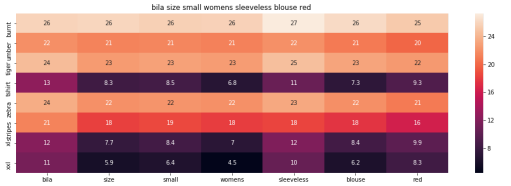
euclidean distance from input : 27.914129 69391053

==========

==========

==========

==



ASIN : B01L7ROZNC

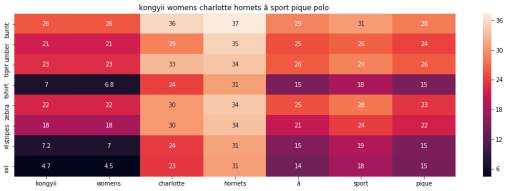
Brand : Bila

euclidean distance from input : 28.188814 123079233

==========

==========

==========



ASIN : B01FJVZST2

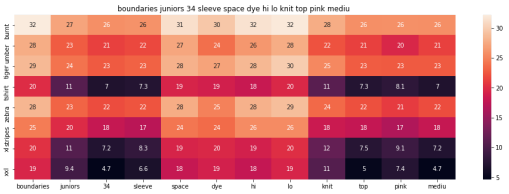
Brand : KONGYII

euclidean distance from input : 28.520633 41896241

==========

==========

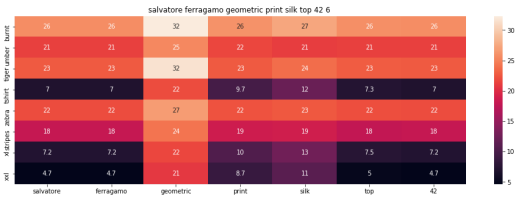
==========



ASIN : B01EXXFS4M

Brand : No Boundaries

euclidean distance from input : 28.7878
3874516236
==========
==========
==========
========



ASIN : B0756JTS1F
Brand : Salvatore Ferragamo
euclidean distance from input : 28.911050
08125305
==========
==========
==========
==

ASIN : B01L9F153U

Brand : ATYPEMX

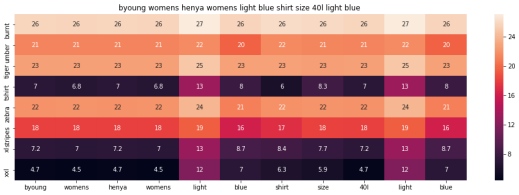
euclidean distance from input : 28.9221 56293794565

==========

==========

==========

========



ASIN : B06Y41MRCH

Brand : Byoung

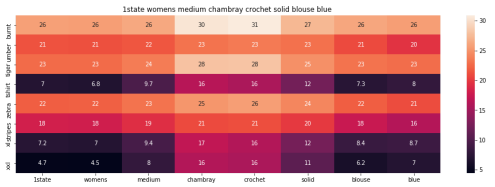
euclidean distance from input : 28.924593 734786384

==========

==========

==========

==

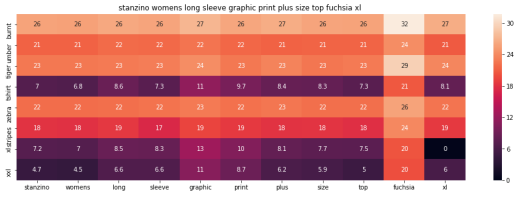


ASIN : B074MK6LV2

Brand : 1.State

euclidean distance from input : 28.975344617769174

==========



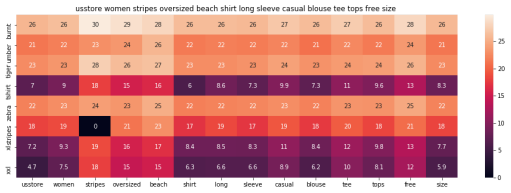
ASIN : B00DP4VHWI

Brand : Stanzino

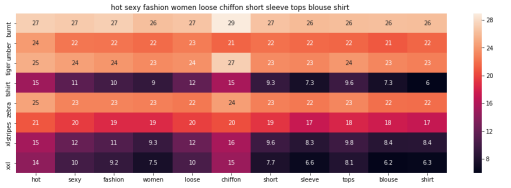
euclidean distance from input : 29.06762572090333

==========

==========
==========
==

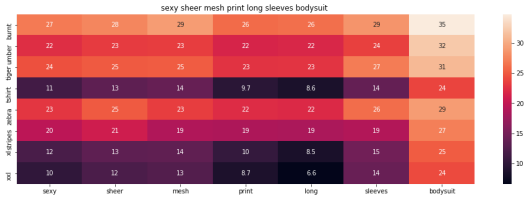


ASIN : B01DNNI1RO
Brand : Usstore
euclidean distance from input : 29.153651
52363526
==========
==========
==



ASIN : B00JMAASRO

Brand : Wotefusi
euclidean distance from input : 29.1894
96000215463
==========
==========
==========
========

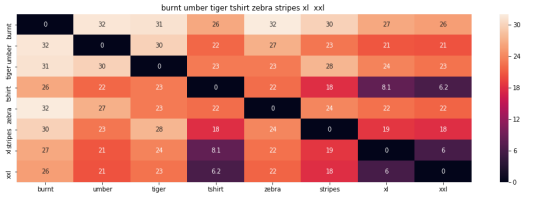


ASIN : B074Z5C98D
Brand : Ariella's closet
euclidean distance from input : 29.238543
579350214
==========
==========
==========
==

```
ASIN : B01M8GB3AL
Brand : Maven West
euclidean distance from input : 29.2707
45325133674
=======================================
=======================================
=======================================
========
```

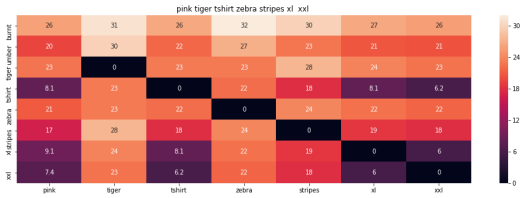
When Title has max weightage, 'Brand&color' has second highest weightage

In [88]:
```
idf_w2v_brand(12566,20, 10,5,1)
```

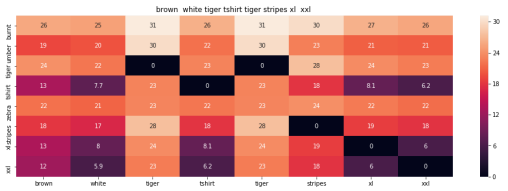


```
ASIN : B00JXQB5FQ
Brand : Si Row
```

euclidean distance from input : 4.69327
4320288765e-07
==========
==========
==========
========



ASIN : B00JXQASS6
Brand : Si Row
euclidean distance from input : 6.0137341
02362076
==========
==========
==========
==



ASIN : B00JXQCWTO

Brand : Si Row

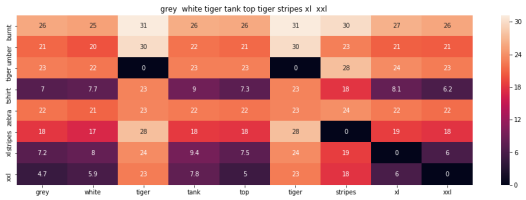
euclidean distance from input : 6.4535787 1055603

==========

==========

==========

==



ASIN : B00JXQAFZ2

Brand : Si Row

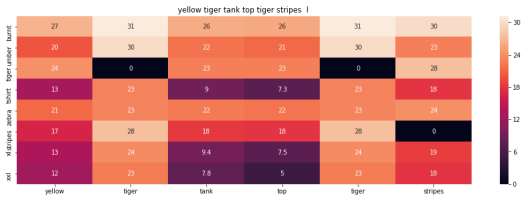
euclidean distance from input : 7.1624553 204712065

==========

==========

==========

==



ASIN : B00JXQAUWA

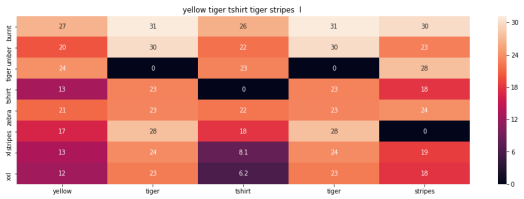
Brand : Si Row

euclidean distance from input : 7.3403241 6354982

==========

==========

==========



ASIN : B00JXQCUIC

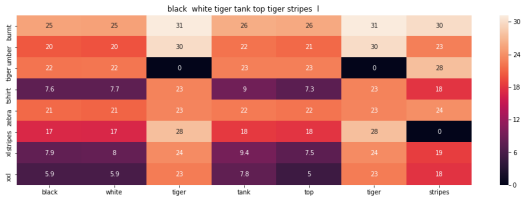
Brand : Si Row

euclidean distance from input : 7.3864095 21215835

==========

==========

==========



ASIN : B00JXQAO94

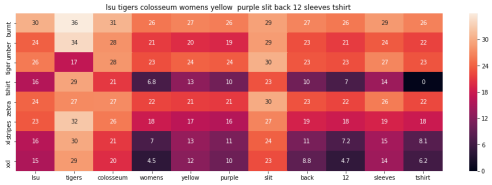
Brand : Si Row

euclidean distance from input : 7.4519116 87963882

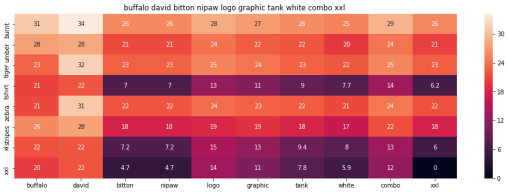
==========

==========

==========



ASIN : B073R5Q8HD
Brand : Colosseum
euclidean distance from input : 7.83241
3949594159
==========
==========
==========
========



ASIN : B018H5AZXQ
Brand : Buffalo
euclidean distance from input : 7.9947643
2822879
==========
==========
==========
==

ASIN : B072BVB47Z

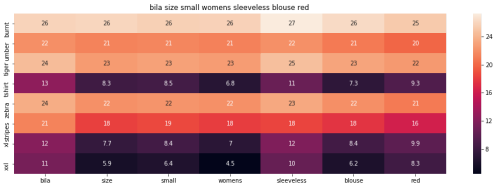
Brand : H By Bordeaux

euclidean distance from input : 8.0794647 20645134

==========

==========

==



ASIN : B01L7ROZNC

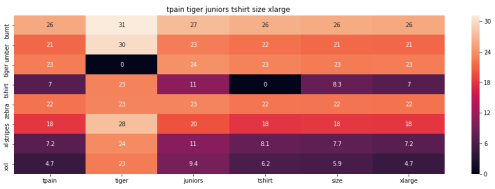
Brand : Bila

euclidean distance from input : 8.1429293 53341717

==========

==========

==

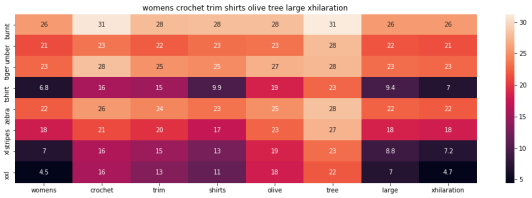


ASIN : B01K0H02OG

Brand : Tultex

euclidean distance from input : 8.149009027108807

==========

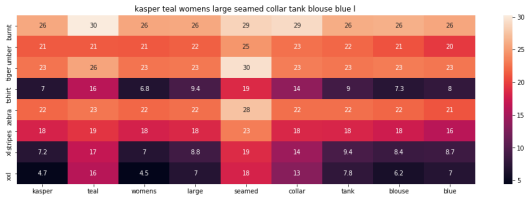


ASIN : B06XBHNM7J

Brand : Xhilaration

euclidean distance from input : 8.165333986508209

==========

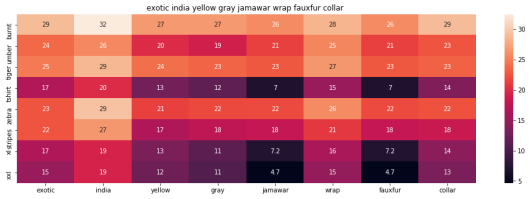


ASIN : B0722DJVQP

Brand : Kasper

euclidean distance from input : 8.210881032571454

==========



ASIN : B073ZHRBV8

Brand : Exotic India

euclidean distance from input : 8.21492

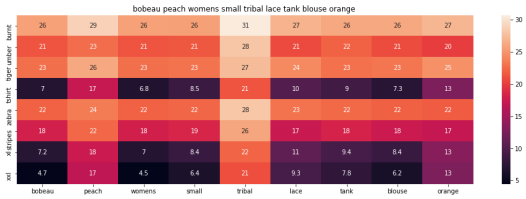
5289379913

==========

==========

==========

========



ASIN : B072JTHCX6

Brand : Bobeau

euclidean distance from input : 8.2541485

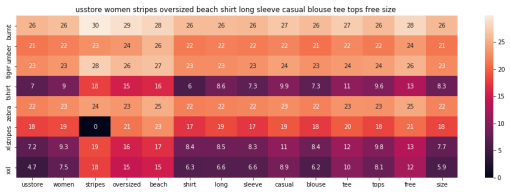
21051068

==========

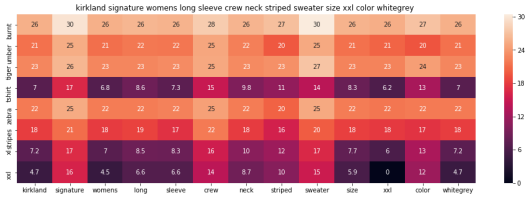
==========

==========

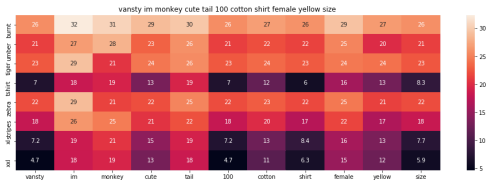
==



ASIN : B01DNNI1RO
Brand : Usstore
euclidean distance from input : 8.26478123687442
==========



ASIN : B06XTPC3FP
Brand : Kirkland Signature
euclidean distance from input : 8.269010305630523
==========

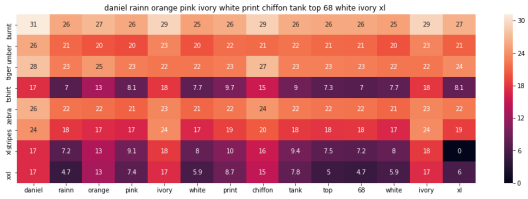


ASIN : B01EFSLHW2

Brand : Vansty

euclidean distance from input : 8.2741954 70437665

==========



ASIN : B01IPV1SFQ

Brand : Daniel Rainn

euclidean distance from input : 8.2856290 34042358

==========