In [74]:
```
import pandas as pd
import numpy as np
import mysql.connector
import seaborn as sns
import matplotlib.pyplot as plt
```

In [75]:
```
mydb = mysql.connector.connect(
 host="localhost",
 user="root",
 password="Sag251408",
 database = "project1050"
)
print(mydb)
if mydb.is_connected():
 print("CONNECTION SUCCESSFUL")
```

```
<mysql.connector.connection.MySQLConnection object at 0x7fe599f2a520>
CONNECTION SUCCESSFUL
```

In [76]:
```
mycursor = mydb.cursor(buffered=True,dictionary = True)
```

Created 2 tables, product_categories and product with appropriate primary key and Foreign key constraints.

In [114]:
```
mycursor.execute("CREATE TABLE product_categories (\
product_type varchar(255) NOT NULL,\
category varchar(255) NOT NULL,\
PRIMARY KEY (product_type))")
```

In [115]:
```
mycursor.execute("CREATE TABLE products (\
product_name varchar(255) NOT NULL, \
product_URL varchar(255) NOT NULL,\
product_type varchar(255) NOT NULL,\
PRIMARY KEY (product_name),\
FOREIGN KEY (product_type) REFERENCES product_categories(product_type))")

mycursor.execute("SHOW TABLES")
for x in mycursor:
 print(x)
```

```
{'Tables_in_project1050': 'product_categories'}
{'Tables_in_project1050': 'products'}
```

In [77]:
```
log = pd.read_csv("log2.csv")
print(log.shape)
log.head()
```

```
(10000, 6)
```

Out[77]:

	Sentiment	Publication_URL	Product URL	clickORnot	gender	age_group
0	positive	https://www.foxnews.com/	https://lees.com/jeans	0	female	juvenile
1	neutral	https://www.mirror.co.uk/news/	https://coach.com/purses	0	male	young
2	negative	https://www.nbcnews.com/	https://covergirl.co/lipsticks	0	male	middle-age
3	positive	https://www.examiner.com/	https://covergirl.co/makeup	0	male	juvenile
4	negative	https://www.nj.com	https://dell.com/computers	1	female	young

In [78]:
```
log.columns = log.columns.str.replace(' ', '')
log.columns
```

Out[78]:
```
Index(['Sentiment', 'Publication_URL', 'ProductURL', 'clickORnot', 'gender',
       'age_group'],
      dtype='object')
```

In [79]:
```
print(np.sum(pd.isnull(log)))
```

```
Sentiment          0
Publication_URL    0
ProductURL         0
clickORnot         0
gender             0
age_group          0
dtype: int64
```

In [80]:
```
log.dtypes
```

Out[80]:
```
Sentiment          object
Publication_URL    object
ProductURL         object
clickORnot          int64
gender             object
age_group          object
dtype: object
```

In [81]:
```
prod = pd.read_csv("products.csv")
print(prod.shape)
prod.head()
```

```
(51, 3)
```

Out[81]:

	product	product_URL	product_type
0	Vitamix blender	https://vitamix.com/blenders	blender
1	Lenova laptop	https://lenova.com/laptops	computer
2	InstantPot pressure cooker	https://InstantPot.com/cookers	pressure cooker
3	NemoK blender	http://nemoK.co/blenders	blender
4	Hamilton Beach blender	https://HamiltonBeach/blenders	blender

In [82]:
```
print(np.sum(pd.isnull(prod)))
```

```
product         0
product_URL     0
product_type    0
dtype: int64
```

In [83]:
```
prod.dtypes
```

Out[83]:
```
product         object
product_URL     object
product_type    object
dtype: object
```

In [84]:
```
print(sum(prod["product"].value_counts()>1))
```

```
1
```

In [85]:
```
prod.drop_duplicates(inplace = True)
print(prod.shape)
```

```
(50, 3)
```

In [86]:
```
prod_cat = pd.read_csv("product_categories.csv")
print(prod_cat.shape)
prod_cat.head()
```

```
(25, 2)
```

Out[86]:

	product_type	category
0	blender	small kitchen appliances
1	pressure cooker	small kitchen appliances
2	computer	consumer electronics
3	coffee	packaged food
4	vitamin	health

In [87]:
```
print(np.sum(pd.isnull(prod_cat)))
```

```
product_type    0
category        0
dtype: int64
```

In [88]:
```
prod_cat.dtypes
```

Out[88]:
```
product_type    object
category        object
dtype: object
```

In [89]:
```
print(sum(prod_cat["product_type"].value_counts()>1))
```

```
0
```

In [90]:
```
prod_cat.drop_duplicates(inplace = True)
print(prod_cat.shape)
```

```
(25, 2)
```

Insert values into product and product_categories tables.

In [1011]:
```
for i,row in prod_cat.iterrows():
    sql = "INSERT INTO product_categories VALUES (%s,%s)"
    row_final = tuple([row['product_type'].strip(), row['category'].strip()])
    mycursor.execute(sql, row_final)
    mydb.commit()
```

In [127]:
```
for i,row in prod.iterrows():
    sql = "INSERT INTO products VALUES (%s,%s,%s)"
    row_final = tuple([row['product'].strip(),row['product_URL'].strip(),row['product_type'].strip()])
    mycursor.execute(sql, row_final)
    mydb.commit()
```

1. Some of the Product_URLs in the log file might have been corrupted. Write a Python (or PySpark) procedure to determine which Product_URLs are corrupted. Let us assume that if a Product_url in the log file doesn't occur in the products table, it is regarded as corrupted. Using this procedure identify and list the corrupted URLs. (10)

In [91]:
```
mycursor.execute("SELECT product_URL FROM project1050.products;")
```

In [92]:
```
result_set = mycursor.fetchall()
urls = []
for row in result_set:
    urls.append(row['product_URL'])
```

In [93]:
```
corrupted_urls = []
for row in log.iloc[:,2]:
    if row not in urls:
        corrupted_urls.append(row)
```

In [94]:
```
print(len(corrupted_urls))
```

```
216
```

Number of corrupted urls are 216. Below are the corrupted Product_URLs list.

In [95]:
```
print(corrupted_urls)
```

```
['https://haier.com/refrigermtors', 'https://sony.comftelevisions', 'https://lg.com/gashers', 'https://leks.com/jeans', 'https://InstantPot.con/coo
kers', 'https://lenova.comslaptops', 'https://broyhill.cvm/recliners', 'https://apple.cfm/ipads', 'https://soundwavemai/speakers', 'https://haieq.c
om/refrigerators', 'https://reminxton.com/shavers', 'https://kaxi.com/handbags', 'https://delk.com/computers', 'https://apple.com/tomputers', 'http
s://levis.comojeans', 'https://Stagbucks.com/coffee', 'https://besla.com', 'http://nemoK.wo/blenders', 'https://samsung.chm/washers', 'https://cent
rum.com/vitamigs', 'https://uord.com/sedans', 'https://InstanqPot.com/cookers', 'https://HamiltonBeach/blenkers', 'https://HamilbonBeach/blenders',
'https://docker.comlpants', 'https://Starbucks.com/poffee', 'https://Ikea.com/sofcs', 'https://samsung.com/dgyers', 'https://kemington.com/shaver
s', 'https://covergiol.co/makeup', 'https://InstantPot.com/coohers', 'https://apkle.com/computers', 'https://maytag.com/wkshers', 'https://covergir
l.co/lipstocks', 'https://levis.com/leans', 'https://basilbasel.io/ierfumes', 'http://maybellije.com/lipstick', 'https://gony.com/televisions', 'ht
tps://bosexcom/speakers', 'https://samsung.com/wdshers', 'https://Starbuckn.com/coffee', 'https://HamiltonBeach/bgenders', 'https://levos.com/jean
s', 'https://ford.com/seduns', 'https://lees.com/keans', 'https://kocker.com/pants', 'https://cotergirl.co/lipsticks', 'https://qtarbucks.com/coffe
e', 'https://bmoyhill.com/recliners', 'https://souudwave.ai/speakers', 'https://NordicTracklelliptical', 'https://ktarbucks.com/coffee', 'https://c
overgirl.cr/lipsticks', 'https://covch.com/purses', 'https://lenova.com/laptjps', 'https://bole.com/speakers', 'https://HamiltonBeach/bxenders', 'h
ttps://symsung.com/washers', 'https://Ikea.comvsofas', 'http://nemoK.no/blenders', 'https://remingtonccom/shavers', 'https://ytarbucks.com/coffee',
'https://coacy.com/purses', 'https://clinique.com/moisturdzers', 'https://maytag.com/walhers', 'https://lenova.com/lapfops', 'https://spple.com/com
puters', 'https://dell.ccm/laptops', 'https://camsung.com/dryers', 'https://maydag.com/washers', 'https://gillette.com/sdavers', 'https://dell.com/
laptoys', 'https://wivenchy.com/perfumes', 'https://InstantPot.eom/cookers', 'https://samsung.com/telfvisions', 'https://samsung.com/televisizns',
'https://rembngton.com/shavers', 'https://geess.com/perfumes', 'https://broyhfll.com/recliners', 'https://clinique.com/mxisturizers', 'https://haie
r.com/refrigeratoes', 'https://sbundwave.ai/speakers', 'https://oenova.com/laptops', 'https://sounduave.ai/speakers', 'https://coverrirl.co/makeu
p', 'https://samsung.cok/washers', 'https://gbess.com/perfumes', 'http://maybelline.com/xipstick', 'https://HamiloonBeach/blenders', 'https://docke
r.comgpants', 'https://soundwave.ai/speakjrs', 'https://gillette.cou/shavers', 'https://NordicTrack.com/tveadmills', 'https://bose.com/spkakers',
'https://cougar.co/jtans', 'https://cougar.co/jpans', 'https://coach.com/pufses', 'https://lenova.com/lnptops', 'https://leei.com/jeans', 'https://
coujar.co/jeans', 'https://jell.com/laptops', 'https://soundwave.ai/speaeers', 'https://dell.comfcomputers', 'https://Starbubks.com/coffee', 'http
s://apple.fom/ipads', 'https://IestantPot.com/cookers', 'http://maybelline.com/lipstuck', 'https://pord.com/sedans', 'https://maltag.com/refrigerat
ors', 'https://basilbasel.io/perfumgs', 'https://covergirl.co/lspsticks', 'https://afple.com/ipads', 'https://yaai.com/handbags', 'https://maytag.c
wm/washers', 'https://haier.com/refdigerators', 'https://guess.cam/perfumes', 'https://docker.uom/pants', 'https://sony.com/televisirns', 'https://
centrui.com/vitamins', 'https://basilbasel.io/perfumis', 'https://NordicTrack.com/tbeadmills', 'https://deul.com/computers', 'https://broyhvll.com/
recliners', 'https://tesla.rom', 'https://ipple.com/ipads', 'https://guess.com/perfuies', 'https://applebcom/ipads', 'https://gilletde.com/shaver
s', 'https://ooach.com/purses', 'https://cougar.co/jepns', 'https://levis.com/qeans', 'https://NordicTrack.com/treadmihls', 'https://lg.com/drrer
s', 'https://apple.cdm/computers', 'https://aplle.com/ipads', 'https://NordicTrack.com/zreadmills', 'https://docker.com/panrs', 'https://centeum.co
m/vitamins', 'https://lg.com/washprs', 'https://kbai.com/handbags', 'https://gifenchy.com/perfumes', 'https://samsung.com/wasuers', 'https://givenc
hy.com/perfuhes', 'https://covergirn.co/makeup', 'https://mayttg.com/washers', 'https://givenchy.com/oerfumes', 'https://Starbucks.con/coffee', 'ht
tps://Starbuzks.com/coffee', 'https://centrum.com/votamins', 'https://dofker.com/pants', 'https://apple.com/ipajs', 'https://covergirl.co/makeyp',
'https://samsung.cof/dryers', 'https://givcnchy.com/perfumes', 'https://gillette.com/ahavers', 'https://giorgiojcom/perfumes', 'https://NordicTrac
k.coh/rowers', 'https://covergirl.cs/makeup', 'https://covergirw.co/makeup', 'https://lenova.com/laptlps', 'https://lg.comxtvs', 'https://covergir
l.covlipsticks', 'https://Starpucks.com/coffee', 'https://broyhill.cov/recliners', 'https://docker.com/paets', 'https://HamdltonBeach/blenders', 'h
ttps://jaguar.co/perfuues', 'https://jaguar.co/perfymes', 'https://lg.fom/washers', 'https://covergirl.co/makezp', 'http://nemoK.co/blendess', 'htt
ps://kapi.com/handbags', 'https://gillette.cor/shavers', 'https://dell.com/lapwops', 'https://sonx.com/televisions', 'https://lg.com/washeas', 'htt
ps://HamiltonBeach/bsenders', 'https://delj.com/computers', 'https://samsung.col/dryers', 'https://haier.com/refrigebators', 'https://giorgyo.com/p
erfumes', 'https://guess.fom/perfumes', 'https://NordicTralk/elliptical', 'https://NordicTrack.cvm/treadmills', 'https://yamsung.com/televisions',
'https://dockerrcom/pants', 'https://bose.cam/speakers', 'https://docker.pom/pants', 'https://givenchy.com/psrfumes', 'https://gillette.comsshaver
s', 'https://covergirl.co/lipstdcks', 'https://samsung.comxwashers', 'https://covergirl.co/makezp', 'https://vitamixrcom/blenders', 'https://couga
r.co/jhans', 'https://apnle.com/laptops', 'https://cotergirl.co/lipsticks', 'https://vitamix.com/blhnders', 'https://coxgar.co/jeans', 'https://fel
l.com/computers', 'https://haiee.com/refrigerators', 'https://levps.com/jeans', 'https://samsung.com/washeks', 'https://lg.com/wascers', 'https://k
aaa.com/handbags', 'https://basilbasel.io/peyfumes', 'https://basilbasel.io/perfunes', 'https://cougar.co/jeaas', 'https://levia.com/jeans', 'http
s://Ikea.lom/sofas', 'https://apple.com/iqads', 'http://nejoK.co/blenders', 'https://maytag.cpm/washers', 'https://guessmcom/perfumes', 'https://sa
msuag.com/televisions', 'https://InstantPotycom/cookers']
```

2.For each corrupted URL what will you do with it? Don't assume that for each corrupted URL the correct approach is to delete that log entry. What if the URL contained '.cam' instead of '.com' but otherwise corresponded with a URL in the 'products' table? In that case the proper approach would be to correct the URL. In other cases, the URL might be so corrupted that the best approach would be to delete that log entry (the entire row). Describe your approach to dealing with corrupted URLs. That is, describe your approach to determining that a URL is too corrupted to be rescued. It must describe a) a procedure for determining the degree to which the URL is corrupted, b) a threshold for determining in terms of this degree of corruption whether it can be corrected, and c) for those which can be corrected, identifying its corrected form. For extra credit implement this in a Python (or PySpark) program.

Answer: Corrupted URLs are the URLs which are not present in the products table.
The URLS which are corrupted with minor spelling mistakes can be corrected.
I have 2 approaches of dealing with the corrupted URLs.

Approach 1:

(1) We can notice that the URL consists of 4 parts:
https, company name, com, product name.
I have split the corrupted URLS into 4 parts.

(2)Then I check that the 3rd part of the URL has com in it.
If it does not have com due to spelling mistake, it's changed to com.

(3)Then the 2nd part of the URL is checked to see if it matches with
any of the existing company names. All the company names are stored in prod_names list
If a close match is found then it is replaced.

(4)Then the 4th part of the URL is checked to see if it matches with
any of the existing product type names. All the product type names are stored in prod_types
If a close match is found it is replaced.
If it doesn't match with any of the existing company names or product type names
it is considered corrupted beyound correction and the record is deleted.

Approach 1:

(1)Corrupted URLs are checked to see if there is any close match to the existing Product URLs.
If there is a close match then it is replaced with the match that has highest rank among
the list of close matches.
(2)If it was not not able to find any close match then
it is considered as beyound correction and it is deleted.

By these approaches I was able to correct 168 URLs

In [96]:
```
non_pural_replaces = ["coffee", "elliptical", "lipstick"]
```

In [97]:
```
prod_types = []
p_types = prod['product'].str.split(" ")
for p in p_types:
    p = p[-1].lower()
    p = p.rstrip("s")
    if p not in non_pural_replaces:
        p = p+"s"
    prod_types.append(p)
prod_types = set(prod_types)
prod_types
```

Out[97]:
```
{'blenders',
 'coffee',
 'computers',
 'cookers',
 'dryers',
 'elliptical',
 'handbags',
 'ipads',
 'jeans',
 'laptops',
 'lipstick',
 'makeups',
 'moisturizers',
 'multivitamins',
 'pants',
 'perfumes',
 'purses',
 'recliners',
 'refrigerators',
 'rowers',
 'sedans',
 'shavers',
 'sofas',
 'speakers',
 'teslas',
 'treadmills',
 'tvs',
 'washers'}
```

In [98]:
```
prod_names = []
p_names = prod['product'].str.split(" ")
for p in p_names:
    p = p[0].lower()
    prod_names.append(p)
prod_names = set(prod_names)
prod_names
```

Out[98]:
```
{'apple',
 'basilbasel',
 'bose',
 'broyhill',
 'centrum',
 'clinique',
 'coach',
 'cougar',
 'covergirl',
 'dell',
 'docker',
 'ford',
 'gillette',
 'giorgio',
 'givenchy',
 'guess',
 'haier',
 'hamilton',
 'ikea',
 'instantpot',
 'jaguar',
 'kaai',
 'lavazza',
 'lee',
 'lenova',
 'levis',
 'lg',
 'maybelline',
 'maytag',
 'nemok',
 'nordictrack',
 'remington',
 'samsung',
 'sony',
 'soundwave',
 'starbucks',
 'tesla',
 'vitamix'}
```

Approach 1:

In [99]:
```
import difflib
import re

count = 0
for c_url in corrupted_urls:
    ls = re.split(r"/|\.", c_url)
    ls = list(filter(None, ls))
    changed = 0
    if len(ls) == 4:
        if ls[2] != "com":
            ls[2] = "com"
            changed = 1

        if ls[1] not in prod_names:
            pn = difflib.get_close_matches(ls[1], prod_names,cutoff = 0.8)
            if len(pn) == 1:
                ls[1] = pn[0]
                changed = 2

        if ls[3] not in prod_types:
            pt = difflib.get_close_matches(ls[3], prod_types, cutoff = 0.8)
            if len(pt) == 1:
                ls[3] = pt[0]
                changed = 3

        if (ls[3] not in prod_types) or (ls[1] not in prod_names):
            continue

        if changed:
            count += 1
            print("Before :", c_url)
            correct_url = ls[0]+"//"+ls[1]+"."+ls[2]+"/"+ls[3]
            print("Final corrected url :", correct_url)
            log['ProductURL'] = log['ProductURL'].replace(c_url, correct_url)
```

```
Before : https://haier.com/refrigerators (https://haier.com/refrigerators)
Final corrected url : https://haier.com/refrigerators (https://haier.com/refrigerators)
Before : https://levps.com/jeans (https://levps.com/jeans)
Final corrected url : https://levis.com/jeans (https://levis.com/jeans)
Before : https://samsung.com/washeks (https://samsung.com/washeks)
Final corrected url : https://samsung.com/washers (https://samsung.com/washers)
Before : https://lg.com/wascers (https://lg.com/wascers)
Final corrected url : https://lg.com/washers (https://lg.com/washers)
Before : https://basilbasel.io/peyfumes (https://basilbasel.io/peyfumes)
Final corrected url : https://basilbasel.com/perfumes (https://basilbasel.com/perfumes)
Before : https://basilbasel.io/perfunes (https://basilbasel.io/perfunes)
Final corrected url : https://basilbasel.com/perfumes (https://basilbasel.com/perfumes)
Before : https://cougar.co/jeaas (https://cougar.co/jeaas)
Final corrected url : https://cougar.com/jeans (https://cougar.com/jeans)
Before : https://levia.com/jeans (https://levia.com/jeans)
Final corrected url : https://levis.com/jeans (https://levis.com/jeans)
Before : https://apple.com/iqads (https://apple.com/iqads)
Final corrected url : https://apple.com/ipads (https://apple.com/ipads)
Before : https://maytag.cpm/washers (https://maytag.cpm/washers)
Final corrected url : https://maytag.com/washers (https://maytag.com/washers)
```

Approach 2 :

In [28]:
```
count = 0
for c_url in corrupted_urls:
    print("Before(corrupted url):", c_url)
    corrected_url = difflib.get_close_matches(c_url, urls, cutoff = 0.9)[0]
    print("After, correct url:", corrected_url)
    log['ProductURL'] = log['ProductURL'].replace(c_url, corrected_url)
```

```
Before(corrupted url): https://haier.com/refrigermtors (https://haier.com/refrigermtors)
After, correct url: https://haier.com/refrigerators (https://haier.com/refrigerators)
Before(corrupted url): https://sony.comftelevisions (https://sony.comftelevisions)
After, correct url: https://sony.com/televisions (https://sony.com/televisions)
Before(corrupted url): https://lg.com/gashers (https://lg.com/gashers)
After, correct url: https://lg.com/washers (https://lg.com/washers)
Before(corrupted url): https://leks.com/jeans (https://leks.com/jeans)
After, correct url: https://lees.com/jeans (https://lees.com/jeans)
Before(corrupted url): https://InstantPot.con/cookers (https://InstantPot.con/cookers)
After, correct url: https://InstantPot.com/cookers (https://InstantPot.com/cookers)
Before(corrupted url): https://lenova.comslaptops (https://lenova.comslaptops)
After, correct url: https://lenova.com/laptops (https://lenova.com/laptops)
Before(corrupted url): https://broyhill.cvm/recliners (https://broyhill.cvm/recliners)
After, correct url: https://broyhill.com/recliners (https://broyhill.com/recliners)
Before(corrupted url): https://apple.cfm/ipads (https://apple.cfm/ipads)
After, correct url: https://apple.com/ipads (https://apple.com/ipads)
Before(corrupted url): https://soundwavemai/speakers (https://soundwavemai/speakers)
After, correct url: https://soundwave.ai/speakers (https://soundwave.ai/speakers)
Before(corrupted url): https://haieq.com/refrigerators (https://haieq.com/refrigerators)
```

In [29]:
```
corrupted_urls = []
for row in log.iloc[:,2]:
    if row not in urls:
        corrupted_urls.append(row)
print("Number of corrupted urls now:", len(corrupted_urls))
```

```
Number of corrupted urls now: 48
```

The 48 URLs which are beyound correction are deleted.

In [30]:
```
for c_url in corrupted_urls:
    log.drop(log.loc[log['ProductURL'] == c_url].index, inplace=True)
```

3. For each product, compute all the Publication_URLs containing an ad for that product. (Don't just give the results. Show all the work by which you got those results. This applies to all the questions below.)

In [31]:
```
mycursor.execute("SELECT * FROM project1050.products;")
prod_set = mycursor.fetchall()
prod_dict = {}
for row in prod_set:
    #name = row['product_name']
    prod_dict[row['product_name']] = row['product_URL']
```

prod_dict is a dictionary with product names as keys and corresponding product_URL as values.

In [32]:
```
prod_dict
```

Out[32]:
```
{'Apple computer': 'https://apple.com/computers',
 'Apple iPad': 'https://apple.com/ipads',
 'Apple laptop': 'https://apple.com/laptops',
 'BasilBasel perfume': 'https://basilbasel.io/perfumes',
 'bose speakers': 'https://bose.com/speakers',
 'Broyhill recliner': 'https://broyhill.com/recliners',
 'Centrum MultiVitamins': 'https://centrum.com/vitamins',
 'Clinique moisturizer': 'https://clinique.com/moisturizers',
 'Coach purse': 'https://coach.com/purses',
 'Cougar jeans': 'https://cougar.co/jeans',
 'covergirl lipstick': 'https://covergirl.co/lipsticks',
 'Covergirl makeup': 'https://covergirl.co/makeup',
 'Dell computer': 'https://dell.com/computers',
 'Dell laptop': 'https://dell.com/laptops',
 'Docker pants': 'https://docker.com/pants',
 'Ford  sedan': 'https://ford.com/sedans',
 'Gillette shaver': 'https://gillette.com/shavers',
 'Giorgio perfume': 'https://giorgio.com/perfumes',
 'Givenchy perfume': 'https://givenchy.com/perfumes',
 'Guess perfume': 'https://guess.com/perfumes',
 'Haier refrigerator': 'https://haier.com/refrigerators',
 'Hamilton Beach blender': 'https://HamiltonBeach/blenders',
 'Ikea sofa': 'https://Ikea.com/sofas',
 'InstantPot pressure cooker': 'https://InstantPot.com/cookers',
 'Jaguar perfume': 'https://jaguar.co/perfumes',
 'Kaai handbags': 'https://kaai.com/handbags',
 'Lavazza Coffee': 'https://Lavazza.com/coffee',
 'Lee jeans': 'https://lees.com/jeans',
 'Lenova laptop': 'https://lenova.com/laptops',
 'Levis Jeans': 'https://levis.com/jeans',
 'LG dryer': 'https://lg.com/dryers',
 'LG TV': 'https://lg.com/tvs',
 'LG washer': 'https://lg.com/washers',
 'Maybelline lipstick': 'http://maybelline.com/lipstick',
 'Maytag dryer': 'https://maytag.com/dryers',
 'Maytag refrigerator': 'https://maytag.com/refrigerators',
 'Maytag washer': 'https://maytag.com/washers',
 'NemoK blender': 'http://nemoK.co/blenders',
 'NordicTrack elliptical': 'https://NordicTrack/elliptical',
 'NordicTrack rower': 'https://NordicTrack.com/rowers',
 'NordicTrack treadmill': 'https://NordicTrack.com/treadmills',
 'Remington shaver': 'https://remington.com/shavers',
 'Samsung dryer': 'https://samsung.com/dryers',
 'Samsung TV': 'https://samsung.com/televisions',
 'Samsung washer': 'https://samsung.com/washers',
 'Sony TV': 'https://sony.com/televisions',
 'Soundwave speakers': 'https://soundwave.ai/speakers',
 'Starbucks Coffee': 'https://Starbucks.com/coffee',
 'Tesla': 'https://tesla.com',
 'Vitamix blender': 'https://vitamix.com/blenders'}
```

In [33]:
```
prod_pub_url = {}
for prd, url in prod_dict.items():
    prod_pub_url[prd] = log[log.ProductURL == url].Publication_URL
```

Each product and the Publication_URLs containing an ad for that product is shown below:

In [34]:
```
total_pub = 0
for k in prod_pub_url.keys():
    total_pub += len(prod_pub_url[k])
    print("\n",k ,":\n")
    print(prod_pub_url[k])
```

```
 Apple computer :

77        https://www.nydailynews.com/ (https://www.nydailynews.com/)
89      https://www.mirror.co.uk/news/ (https://www.mirror.co.uk/news/)
141           https://www.cbsnews.com/ (https://www.cbsnews.com/)
181          https://www.engadget.com/ (https://www.engadget.com/)
310          https://www.engadget.com/ (https://www.engadget.com/)
                     ...
9815         https://www.usatoday.com/ (https://www.usatoday.com/)
9828         https://www.upworthy.com/ (https://www.upworthy.com/)
9839               https://www.nj.com (https://www.nj.com)
9886    https://www.mirror.co.uk/news/ (https://www.mirror.co.uk/news/)
9982      https://www.bostonglobe.com/ (https://www.bostonglobe.com/)
Name: Publication_URL, Length: 203, dtype: object

 Apple iPad :

22               https://mashable.com/ (https://mashable.com/)
```

In [35]:
```
print(total_pub)
```

```
9952
```

4. For each product type, compute all the Publication_URLs containing an ad for that product type. Your solution must be scalable. That is, it should work well even if there are hundreds of products in each product_type and there are hundreds of product_types. (Hint: To make it scalable you should consider using a Python or PySpark script instead of a SQL query.)

In [36]:
```
mycursor.execute("SELECT * FROM project1050.products;")
prodtype_set = mycursor.fetchall()
prodtype_dict = {}
for row in prodtype_set:
    #name = row['product_name']
    if row['product_type'] not in prodtype_dict:
        prodtype_dict[row['product_type']] = [row['product_URL']]
    else:
        prodtype_dict[row['product_type']].append(row['product_URL'])
```

prodtype_dict is a dictionary that contains product types as keys
and corresponding list of product urls as values.

In [37]:
```
prodtype_dict
```

Out[37]:
```
{'computer': ['https://apple.com/computers',
  'https://apple.com/laptops',
  'https://dell.com/computers',
  'https://dell.com/laptops',
  'https://lenova.com/laptops'],
 'tablet': ['https://apple.com/ipads'],
 'perfume': ['https://basilbasel.io/perfumes',
  'https://giorgio.com/perfumes',
  'https://givenchy.com/perfumes',
  'https://guess.com/perfumes',
  'https://jaguar.co/perfumes'],
 'speakers': ['https://bose.com/speakers', 'https://soundwave.ai/speakers'],
 'furniture': ['https://broyhill.com/recliners', 'https://Ikea.com/sofas'],
 'vitamin': ['https://centrum.com/vitamins'],
 'face cream': ['https://clinique.com/moisturizers'],
 "women's purse": ['https://coach.com/purses', 'https://kaai.com/handbags'],
 'jeans': ['https://cougar.co/jeans',
  'https://lees.com/jeans',
  'https://levis.com/jeans'],
 'lipstick': ['https://covergirl.co/lipsticks',
  'http://maybelline.com/lipstick'],
 'makeup': ['https://covergirl.co/makeup'],
 'pants': ['https://docker.com/pants'],
 'car': ['https://ford.com/sedans', 'https://tesla.com'],
 'shaver': ['https://gillette.com/shavers', 'https://remington.com/shavers'],
 'refrigerator': ['https://haier.com/refrigerators',
  'https://maytag.com/refrigerators'],
 'blender': ['https://HamiltonBeach/blenders',
  'http://nemoK.co/blenders',
  'https://vitamix.com/blenders'],
 'pressure cooker': ['https://InstantPot.com/cookers'],
 'coffee': ['https://Lavazza.com/coffee', 'https://Starbucks.com/coffee'],
 'dryer': ['https://lg.com/dryers',
  'https://maytag.com/dryers',
  'https://samsung.com/dryers'],
 'television': ['https://lg.com/tvs',
  'https://samsung.com/televisions',
  'https://sony.com/televisions'],
 'washer': ['https://lg.com/washers',
  'https://maytag.com/washers',
  'https://samsung.com/washers'],
 'elliptical trainer': ['https://NordicTrack/elliptical'],
 'rowing machine': ['https://NordicTrack.com/rowers'],
 'treadmill': ['https://NordicTrack.com/treadmills']}
```

In [38]:
```
import warnings
warnings.filterwarnings("ignore")

ptype_pub_url = {}
for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        if ptype not in ptype_pub_url:
            ptype_pub_url[ptype] = log[log.ProductURL == url].Publication_URL
        else:
            ptype_pub_url[ptype].append(log[log.ProductURL == url].Publication_URL)
```

In [39]:
```
ind = 0
ptype_publication = {"ptype":[], "pub_urls":[] }

for k in ptype_pub_url.keys():
    for url in ptype_pub_url[k]:
        ptype_publication["ptype"].append(k)
        ptype_publication["pub_urls"].append(url)
ptype_publication = pd.DataFrame(ptype_publication)
```

Below is a DataFrame that consists of each product type and the Publication_URLs containing an ad for that product type.

In [40]:
```
ptype_publication
```

Out[40]:

	ptype	pub_urls
0	computer	https://www.nydailynews.com/
1	computer	https://www.mirror.co.uk/news/
2	computer	https://www.cbsnews.com/
3	computer	https://www.engadget.com/
4	computer	https://www.engadget.com/
...	...	...
4614	treadmill	https://www.usnews.com/
4615	treadmill	https://www.upworthy.com/
4616	treadmill	https://www.boston.com
4617	treadmill	https://www.businessinsider.com/
4618	treadmill	https://www.nytimes.com/

4619 rows × 2 columns

5. Save this information in the database. Should you save it in the products table or the product_categories table or should you create a new table, product_type_pubURLs, and save this information in this table? If you create a new table, make sure to set up all the appropriate foreign key constraints. On the other hand, if you use one of the existing tables, explain how you will avoid redundancy in your data. In either case, justify your decision.

Answer: A new table called product_type_pubURLs is created appropriate foreign key constraints and the information is saved in that table. I have created a new table as we can not add this information to product table or product_categories table because other columns present in

the respective tables gets repeated and become redundant.

In [41]:
```
ptype_publication.drop_duplicates(inplace=True)
```

In [42]:
```
sum(ptype_publication[["ptype","pub_urls"]].value_counts()>1)
```

Out[42]:
```
0
```

In [206]:
```
mycursor.execute("CREATE TABLE product_type_pubURLs (\
product_type varchar(255) NOT NULL,\
Publication_URL varchar(255) NOT NULL,\
PRIMARY KEY (product_type, Publication_URL),\
FOREIGN KEY (product_type) REFERENCES product_categories(product_type))")
```

In [43]:
```
mycursor.execute("SHOW TABLES;")
for x in mycursor:
 print(x)
```

```
{'Tables_in_project1050': 'product_categories'}
{'Tables_in_project1050': 'product_type_pubURLs'}
{'Tables_in_project1050': 'product_type_sentiment_clickrate'}
{'Tables_in_project1050': 'products'}
```

In [1263]:
```
for i,row in ptype_publication.iterrows():
    sql = "INSERT INTO product_type_pubURLs VALUES (%s,%s)"
    row_final = tuple([row['ptype'].strip(), row['pub_urls'].strip()])
    mycursor.execute(sql, row_final)
    mydb.commit()
```

6. For each product, compute the click rate for it. (Click rate is the number of times a display of an ad was clicked on (by any user) divided by the number of times it was displayed (to any user). That is, the click rate is not specific to each user.)

In [44]:
```
prod_click_rate = {}
total_click = 0
pos_click = 0
for prd, url in prod_dict.items():
    clicks = log[log.ProductURL == url].clickORnot
    total_click = len(clicks)
    pos_click = len(clicks[clicks == 1])
    prod_click_rate[prd] = pos_click/total_click
```

Click rate for each product.

In [45]:
```
p_click_rate = pd.DataFrame(prod_click_rate.items(), columns=['Product', 'Click_rate'])
p_click_rate
```

Out[45]:

	Product	Click_rate
0	Apple computer	0.793103
1	Apple iPad	0.503788
2	Apple laptop	0.564516
3	BasilBasel perfume	0.651007
4	bose speakers	0.525510
5	Broyhill recliner	0.539216
6	Centrum MultiVitamins	0.626556
7	Clinique moisturizer	0.805556
8	Coach purse	0.388646
9	Cougar jeans	0.258427
10	covergirl lipstick	0.827068
11	Covergirl makeup	0.258883
12	Dell computer	0.651584
13	Dell laptop	0.315186
14	Docker pants	0.685897
15	Ford sedan	0.136564
16	Gillette shaver	0.713376
17	Giorgio perfume	0.799065
18	Givenchy perfume	0.459716
19	Guess perfume	0.406977
20	Haier refrigerator	0.207547
21	Hamilton Beach blender	0.407767
22	Ikea sofa	0.573034
23	InstantPot pressure cooker	0.502646
24	Jaguar perfume	0.472868
25	Kaai handbags	0.660000
26	Lavazza Coffee	0.564103
27	Lee jeans	0.466667
28	Lenova laptop	0.637255
29	Levis Jeans	0.668142
30	LG dryer	0.630435
31	LG TV	0.480769
32	LG washer	0.495327
33	Maybelline lipstick	0.560976
34	Maytag dryer	0.344828
35	Maytag refrigerator	0.396552

	Product	Click_rate
36	Maytag washer	0.506944
37	NemoK blender	0.572614
38	NordicTrack elliptical	0.528409
39	NordicTrack rower	0.224599
40	NordicTrack treadmill	0.487395
41	Remington shaver	0.349650
42	Samsung dryer	0.435897
43	Samsung TV	0.731250
44	Samsung washer	0.550943
45	Sony TV	0.392857
46	Soundwave speakers	0.549505
47	Starbucks Coffee	0.277228
48	Tesla	0.591667
49	Vitamix blender	0.507317

7. For each product, compute the click rate for each sentiment type.

In [46]:
```
prod_pos_click_rate = {}
total_click = 0
pos_click = 0
for prd, url in prod_dict.items():
    clicks = log[(log.ProductURL == url) & (log.Sentiment == "positive")].clickORnot
    total_click = len(clicks)
    pos_click = len(clicks[clicks == 1])
    prod_pos_click_rate[prd] = pos_click/total_click
```

In [47]:
```
prod_neutral_click_rate = {}
total_click = 0
pos_click = 0
for prd, url in prod_dict.items():
    clicks = log[(log.ProductURL == url) & (log.Sentiment == "neutral")].clickORnot
    total_click = len(clicks)
    pos_click = len(clicks[clicks == 1])
    prod_neutral_click_rate[prd] = pos_click/total_click
```

In [48]:
```
prod_negative_click_rate = {}
total_click = 0
pos_click = 0
for prd, url in prod_dict.items():
    clicks = log[(log.ProductURL == url) & (log.Sentiment == "negative")].clickORnot
    total_click = len(clicks)
    pos_click = len(clicks[clicks == 1])
    prod_negative_click_rate[prd] = pos_click/total_click
```

In [49]:

```
pos_click_rate = pd.DataFrame\
(prod_pos_click_rate.items(), columns=['Product', 'positive_Click_rate'])
neu_click_rate = pd.DataFrame\
(prod_neutral_click_rate.items(), columns=['Product', 'neutral_Click_rate'])
neg_click_rate = pd.DataFrame\
(prod_negative_click_rate.items(), columns=['Product', 'negative_Click_rate'])
```

Click rate for each product and sentiment type.

In [50]:
```
merg = pd.merge(pos_click_rate, neg_click_rate)
prod_senti_click_rate = pd.merge(merg,neu_click_rate)
prod_senti_click_rate
```

Out[50]:

	Product	positive_Click_rate	negative_Click_rate	neutral_Click_rate
0	Apple computer	0.764706	0.700000	0.923077
1	Apple iPad	0.500000	0.391304	0.627907
2	Apple laptop	0.772727	0.153846	0.731707
3	BasilBasel perfume	0.368421	0.771429	0.859649
4	bose speakers	0.523077	0.596774	0.463768
5	Broyhill recliner	0.242857	0.814286	0.562500
6	Centrum MultiVitamins	0.186667	0.835294	0.814815
7	Clinique moisturizer	0.583333	0.902778	0.930556
8	Coach purse	0.400000	0.316456	0.453333
9	Cougar jeans	0.385417	0.085106	0.311688
10	covergirl lipstick	1.000000	0.425000	1.000000
11	Covergirl makeup	0.250000	0.117647	0.389610
12	Dell computer	0.719512	0.830986	0.382353
13	Dell laptop	0.452830	0.145299	0.357143
14	Docker pants	0.333333	0.835821	0.818182
15	Ford sedan	0.265823	0.013158	0.125000
16	Gillette shaver	0.139535	0.904762	0.960784
17	Giorgio perfume	0.830769	0.638554	0.969697
18	Givenchy perfume	0.347826	0.757576	0.302632
19	Guess perfume	0.788462	0.172414	0.306452
20	Haier refrigerator	0.283019	0.173077	0.166667
21	Hamilton Beach blender	0.216216	0.671642	0.353846
22	Ikea sofa	0.322034	0.840000	0.594203
23	InstantPot pressure cooker	0.725806	0.000000	0.877193
24	Jaguar perfume	0.302326	0.434783	0.700000
25	Kaai handbags	0.512821	0.490566	0.956522
26	Lavazza Coffee	0.342857	0.840909	0.447368
27	Lee jeans	0.328947	0.338028	0.777778
28	Lenova laptop	0.419355	0.750000	0.714286
29	Levis Jeans	0.844156	0.404762	0.800000
30	LG dryer	0.395349	0.680851	0.791667
31	LG TV	0.644444	0.298507	0.590909
32	LG washer	0.828947	0.153846	0.516667
33	Maybelline lipstick	0.698413	0.885246	0.209877
34	Maytag dryer	0.256757	0.117647	0.704918
35	Maytag refrigerator	0.428571	0.052632	0.722222

	Product	positive_Click_rate	negative_Click_rate	neutral_Click_rate
36	Maytag washer	0.278846	0.922222	0.361702
37	NemoK blender	0.322581	0.978022	0.329545
38	NordicTrack elliptical	0.344828	0.491228	0.737705
39	NordicTrack rower	0.301370	0.200000	0.142857
40	NordicTrack treadmill	0.567568	0.302632	0.579545
41	Remington shaver	0.744186	0.145161	0.236842
42	Samsung dryer	0.311111	0.275862	0.716981
43	Samsung TV	0.703125	0.666667	0.823529
44	Samsung washer	0.897959	0.000000	0.698795
45	Sony TV	0.431034	0.055556	0.678571
46	Soundwave speakers	0.557143	0.123077	0.955224
47	Starbucks Coffee	0.201923	0.303030	0.330000
48	Tesla	0.057471	0.791667	0.987654
49	Vitamix blender	0.639344	0.536585	0.338710

8. For each product type, compute the click rate for it.

In [51]:
```
ptype_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[log.ProductURL == url].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_click_rate[ptype] = pos_click/total_click
```

Click rate for each product type.

In [52]:
```
pd.DataFrame(ptype_click_rate.items(), columns = ["Product_Type", "Click_rate"])
```

Out[52]:

	Product_Type	Click_rate
0	computer	0.558583
1	tablet	0.547985
2	perfume	0.555357
3	speakers	0.552691
4	furniture	0.552980
5	vitamin	0.558418
6	face cream	0.573770
7	women's purse	0.567332
8	jeans	0.549794
9	lipstick	0.557712
10	makeup	0.546267
11	pants	0.550377
12	car	0.535807
13	shaver	0.536015
14	refrigerator	0.525229
15	blender	0.522877
16	pressure cooker	0.522344
17	coffee	0.513218
18	dryer	0.509506
19	television	0.510834
20	washer	0.511496
21	elliptical trainer	0.511809
22	rowing machine	0.506280
23	treadmill	0.505828

9. For each product type compute the click rate for each sentiment type.

In [53]:
```
ptype_pos_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "positive")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_pos_click_rate[ptype] = pos_click/total_click
```

In [54]:
```
ptype_neutral_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "neutral")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_neutral_click_rate[ptype] = pos_click/total_click
```

In [55]:
```
ptype_neg_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "negative")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_neg_click_rate[ptype] = pos_click/total_click
```

In [56]:
```
ptype_pos_click_rate = pd.DataFrame\
(ptype_pos_click_rate.items(), columns=['Product_Type', 'positive_Click_rate'])
ptype_neu_click_rate = pd.DataFrame\
(ptype_neutral_click_rate.items(), columns=['Product_Type', 'neutral_Click_rate'])
ptype_neg_click_rate = pd.DataFrame\
(ptype_neg_click_rate.items(), columns=['Product_Type', 'negative_Click_rate'])
```

Click rate for each product type and sentiment type.

In [57]:
```
merg = pd.merge(ptype_pos_click_rate, ptype_neg_click_rate)
ptype_senti_click_rate = pd.merge(merg,ptype_neu_click_rate)
ptype_senti_click_rate
```

Out[57]:

	Product_Type	positive_Click_rate	negative_Click_rate	neutral_Click_rate
0	computer	0.604972	0.501355	0.570270
1	tablet	0.584821	0.479393	0.581140
2	perfume	0.565395	0.508678	0.591810
3	speakers	0.561565	0.486301	0.609183
4	furniture	0.525050	0.527108	0.605263
5	vitamin	0.501398	0.551341	0.620596
6	face cream	0.506550	0.573287	0.639525
7	women's purse	0.500770	0.554086	0.645503
8	jeans	0.502262	0.507171	0.640707
9	lipstick	0.522975	0.519266	0.630277
10	makeup	0.514654	0.503230	0.619597
11	pants	0.509994	0.515819	0.624508
12	car	0.479395	0.506257	0.621118
13	shaver	0.477783	0.507587	0.622464
14	refrigerator	0.471878	0.491327	0.612506
15	blender	0.464052	0.516646	0.587274
16	pressure cooker	0.470938	0.501842	0.594209
17	coffee	0.457933	0.500000	0.581515
18	dryer	0.448834	0.488945	0.590757
19	television	0.457345	0.479316	0.596683
20	washer	0.474380	0.471199	0.590687
21	elliptical trainer	0.472003	0.471552	0.593550
22	rowing machine	0.468151	0.466202	0.586608
23	treadmill	0.470375	0.462519	0.586418

10. Save this information you computed in 9 above in a database table. Should you save it in the products table or the product_categories table or the product_type_pubURLs table, or should you create a new table product_type_sentiment_clickrate, and save this information in this table? If you create a new table, make sure to set up all the appropriate foreign key constraints. On the other hand, if you use one of the existing tables, explain how you will avoid redundancy in your data. In either case, justify your decision

I have created a new table called product_type_pubURLs with appropriate foreign key constraints.
I have created a new table as we can not add this information to
product table or product_categories table or product_type_pubURLs table
because other columns present in the respective tables
gets repeated and become redundant.

In [222]:
```
mycursor.execute("CREATE TABLE product_type_sentiment_clickrate (\
product_type varchar(255) NOT NULL, \
positive_Click_rate DOUBLE(40,2) NULL,\
negative_Click_rate DOUBLE(40,2) NOT NULL,\
neutral_Click_rate DOUBLE(40,2) NOT NULL,\
PRIMARY KEY (product_type),\
FOREIGN KEY (product_type) REFERENCES product_categories(product_type))")
```

In [58]:
```
mycursor.execute("SHOW TABLES;")
for x in mycursor:
    print(x)
```

```
{'Tables_in_project1050': 'product_categories'}
{'Tables_in_project1050': 'product_type_pubURLs'}
{'Tables_in_project1050': 'product_type_sentiment_clickrate'}
{'Tables_in_project1050': 'products'}
```

In [1298]:
```
for i,row in ptype_senti_click_rate.iterrows():
    sql = "INSERT INTO product_type_sentiment_clickrate VALUES (%s,%s,%s,%s)"
    row_final = tuple([row['Product_Type'].strip(), row['positive_Click_rate'],\
                       row['negative_Click_rate'], row['neutral_Click_rate']])
    mycursor.execute(sql, row_final)
    mydb.commit()
```

11. Determine if the gender of the person viewing ads make a difference with regard to the click rate of ads shown in different sentiment context. That is, determine if there are any ‘significant’ differences in the correlation between the sentiment type of the ad context and clicking on the product type conditioned on gender. You can decide if any difference counts as ‘significant’. (This is not a yes or no question. Compute the different correlations.)

In [59]:
```
ptype_male_pos_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "positive") & (log.gender == "male")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_male_pos_click_rate[ptype] = pos_click/total_click
```

In [60]:
```
ptype_male_neg_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "negative") & (log.gender == "male")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_male_neg_click_rate[ptype] = pos_click/total_click
```

In [61]:
```
ptype_male_neu_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "neutral") & (log.gender == "male")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_male_neu_click_rate[ptype] = pos_click/total_click
```

In [62]:
```
ptype_female_pos_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "positive") & (log.gender == "female")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_female_pos_click_rate[ptype] = pos_click/total_click
```

In [63]:
```
ptype_female_neg_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "negative") & (log.gender == "female")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_female_neg_click_rate[ptype] = pos_click/total_click
```

In [64]:
```
ptype_female_neu_click_rate = {}
total_click = 0
pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        clicks = log[(log.ProductURL == url) & (log.Sentiment == "neutral") & (log.gender == "female")].clickORnot
        total_click += len(clicks)
        pos_click += len(clicks[clicks == 1])
    ptype_female_neu_click_rate[ptype] = pos_click/total_click
```

In [65]:
```
ptype_male_pos_click_rate = pd.DataFrame\
(ptype_male_pos_click_rate.items(), columns=['Product_Type', 'Male_positive_Click_rate'])

ptype_female_pos_click_rate = pd.DataFrame\
(ptype_female_pos_click_rate.items(), columns=['Product_Type', 'Female_positive_Click_rate'])

ptype_male_neg_click_rate = pd.DataFrame\
(ptype_male_neg_click_rate.items(), columns=['Product_Type', 'Male_negative_Click_rate'])

ptype_female_neg_click_rate = pd.DataFrame\
(ptype_female_neg_click_rate.items(), columns=['Product_Type', 'Female_negative_Click_rate'])

ptype_male_neu_click_rate = pd.DataFrame\
(ptype_male_neu_click_rate.items(), columns=['Product_Type', 'Male_neutral_Click_rate'])

ptype_female_neu_click_rate = pd.DataFrame\
(ptype_female_neu_click_rate.items(), columns=['Product_Type', 'Female_neutral_Click_rate'])
```

In [66]:
```
merg1 = pd.merge(ptype_male_pos_click_rate, ptype_female_pos_click_rate)
merg2 = pd.merge(ptype_male_neg_click_rate, ptype_female_neg_click_rate)
merg3 = pd.merge(ptype_male_neu_click_rate, ptype_female_neu_click_rate)
merg4 = pd.merge(merg1, merg2)
ptype_senti_click_rate = pd.merge(merg4, merg3)
ptype_senti_click_rate
```

Out[66]:

	Product_Type	Male_positive_Click_rate	Female_positive_Click_rate	Male_negative_Click_rate	Female_negative_Click_rate	Male_neutral_Click_rate	Female_neutral_Click_rate
0	computer	0.614035	0.596859	0.500000	0.502618	0.584270	0.557292
1	tablet	0.577465	0.591489	0.475336	0.483193	0.606635	0.559184
2	perfume	0.565826	0.564987	0.506775	0.510526	0.590659	0.592875
3	speakers	0.552632	0.569845	0.490909	0.481651	0.606977	0.611231
4	furniture	0.517526	0.532164	0.525253	0.528942	0.595142	0.614662
5	vitamin	0.488506	0.513612	0.548148	0.554529	0.604167	0.635579
6	face cream	0.496390	0.516074	0.566901	0.579487	0.625887	0.652033
7	women's purse	0.491228	0.509687	0.549606	0.558462	0.640310	0.650442
8	jeans	0.499325	0.504963	0.509383	0.505076	0.634179	0.646983
9	lipstick	0.526250	0.519906	0.523869	0.514899	0.625154	0.635183
10	makeup	0.513806	0.515464	0.509709	0.497156	0.618545	0.620612
11	pants	0.507026	0.512821	0.522648	0.509351	0.623717	0.625277
12	car	0.477152	0.481557	0.512169	0.500514	0.624738	0.617587
13	shaver	0.479397	0.476190	0.508911	0.506292	0.624121	0.620858
14	refrigerator	0.471387	0.472352	0.491412	0.491244	0.609990	0.614953
15	blender	0.464349	0.463768	0.522592	0.510833	0.583113	0.591253
16	pressure cooker	0.469974	0.471854	0.507871	0.495955	0.588336	0.599836
17	coffee	0.453953	0.461781	0.503502	0.496541	0.574675	0.588053
18	dryer	0.448591	0.449071	0.491983	0.485941	0.583969	0.597232
19	television	0.456115	0.458537	0.478767	0.479863	0.587726	0.605245
20	washer	0.473026	0.475679	0.470477	0.471924	0.582891	0.598209
21	elliptical trainer	0.471990	0.472015	0.471605	0.471499	0.586207	0.600627
22	rowing machine	0.466877	0.469375	0.465215	0.467193	0.580128	0.592844
23	treadmill	0.471059	0.469715	0.460036	0.465006	0.580424	0.592192

In [67]:
```
Means = {"index" : 0,
    "Male_positive_mean" : np.round(np.mean(ptype_senti_click_rate.Male_positive_Click_rate),3),
"Female_positive_mean" : np.round(np.mean(ptype_senti_click_rate.Female_positive_Click_rate),3),
"Male_negative_mean" : np.round(np.mean(ptype_senti_click_rate.Male_negative_Click_rate),3),
"Female_negative_mean" : np.round(np.mean(ptype_senti_click_rate.Female_negative_Click_rate),3),
"Male_neutral_mean" : np.round(np.mean(ptype_senti_click_rate.Male_neutral_Click_rate),3),
"Female_neutral_mean" : np.round(np.mean(ptype_senti_click_rate.Female_neutral_Click_rate),3)}
```

In [68]:
```
ptype_gender = pd.DataFrame(Means, index=[0])
ptype_gender = ptype_gender.melt(id_vars="index", var_name="Gender")
del ptype_gender['index']
ptype_gender.columns = ['Gender','Average_Click_rate']
ptype_gender
```

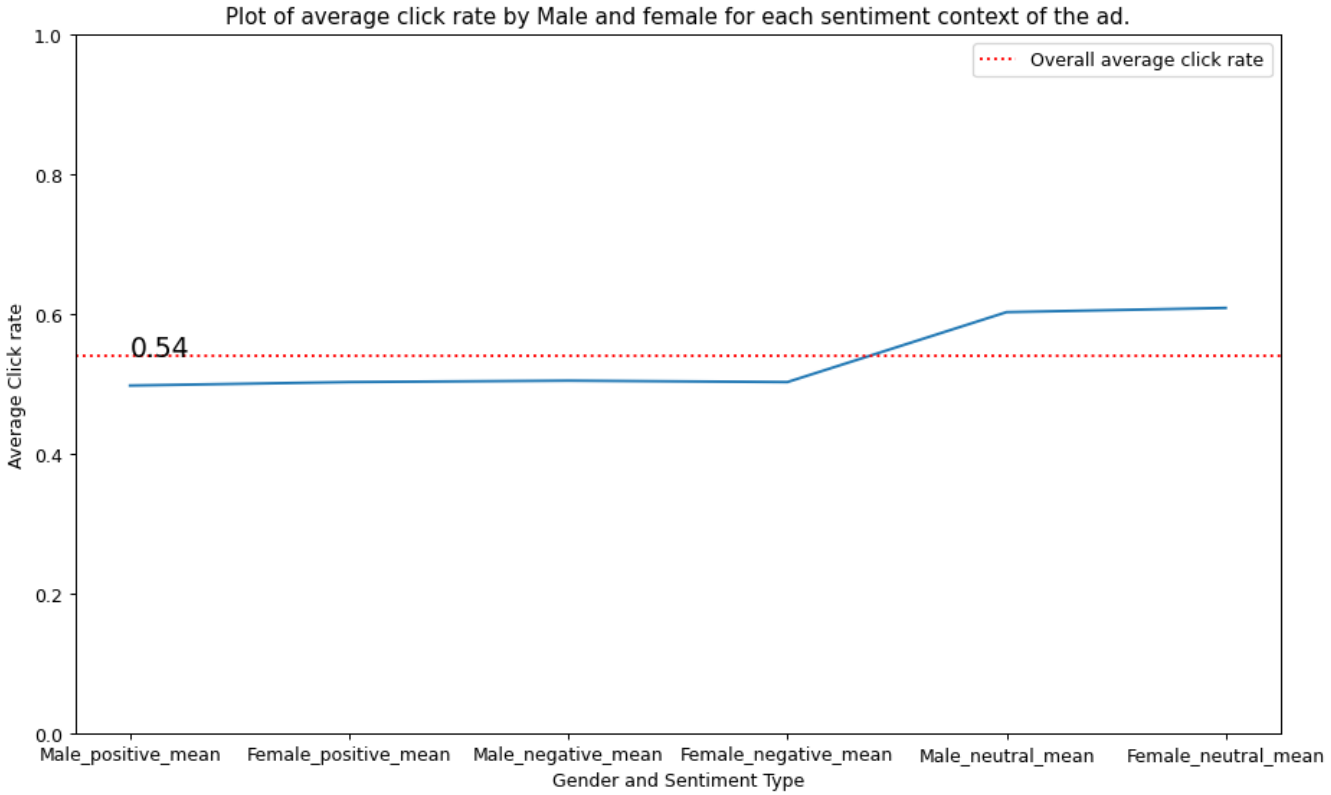
Out[68]:

	Gender	Average_Click_rate
0	Male_positive_mean	0.498
1	Female_positive_mean	0.503
2	Male_negative_mean	0.505
3	Female_negative_mean	0.503
4	Male_neutral_mean	0.603
5	Female_neutral_mean	0.609

In [69]:
```
overall_average_click_rate = np.round(np.mean(ptype_gender.Average_Click_rate), 2)
overall_average_click_rate
```

Out[69]: 0.54

In [70]:
```
from matplotlib.pyplot import figure
figure(figsize=(12, 7), dpi=90)
plt.plot(ptype_gender.Gender, ptype_gender.Average_Click_rate)
plt.ylim((0,1))
plt.axhline(y = overall_average_click_rate, color = 'r', linestyle = ':', label = "Overall average click rate")
plt.text(0, 0.54, overall_average_click_rate, fontsize=15)
plt.title("Plot of average click rate by Male and female for each sentiment context of the ad.")
plt.ylabel('Average Click rate')
plt.legend()
plt.xlabel('Gender and Sentiment Type')
plt.show()
```



From the above plot we can see that average click rate by Male and female
for Neutral sentiment type of the ad context is higer than the average click rate across
the genders and all the sentiment types(overall average click rate).

Average click rates by Male and female for positive and negative sentiment types of the ad context
are not significantly different. They are however below the overall average click rate.

From the result of the ptype_senti_click_rate Dataframe we get to know the following :
(1)when the sentiment type is positive tablet, perfumes, speakers has higher click rate by women.
(2)when the sentiment type is positive computer, lipstick has higher click rate by men.
(3)When sentiment type is negative Face cream, vitamin, women'purse has higher click rate by women.

(4)When sentiment type is negative Blender, pants, car, lipstick has has higher click rate by men.
(5)when sentiment type is neutral Furniture, vitamin, face cream, lipstick, women'purse, jeans
has higher click rate by women.
(6)when sentiment type is neutral computer, tablet has higher click rate by men.

Based on above analysis, I can make following recommendation:
(1)Face cream, vitamins, women's purse, jeans, lipstick, television has higher click rate by women when the sentiment type is neutral.
(2)tablet, car,shaver has higher click rate by men when the when the sentiment type is neutral.
(3)computer has higher click rate by men when the sentiment is positive.

12. The same question as 9 above but replace gender with age-group

In [71]:
```
log.age_group.unique()
```

Out[71]:
```
array(['juvenile', 'young', 'middle-age', 'senior'], dtype=object)
```

In [1431]:
```
ptype_juvenile_pos_click_rate = {}
ptype_juvenile_neg_click_rate = {}
ptype_juvenile_neu_click_rate = {}
pos_juv_total_click = 0
pos_juv_pos_click = 0
neg_juv_total_click = 0
neg_juv_pos_click = 0
neu_juv_total_click = 0
neu_juv_pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        pos_clicks = log[(log.ProductURL == url) & (log.Sentiment == "positive") & (log.age_group == "juvenile")].clickORnot
        pos_juv_total_click += len(pos_clicks)
        pos_juv_pos_click += len(pos_clicks[pos_clicks == 1])

        neg_clicks = log[(log.ProductURL == url) & (log.Sentiment == "negative") & (log.age_group == "juvenile")].clickORnot
        neg_juv_total_click += len(neg_clicks)
        neg_juv_pos_click += len(neg_clicks[neg_clicks == 1])

        neu_clicks = log[(log.ProductURL == url) & (log.Sentiment == "neutral") & (log.age_group == "juvenile")].clickORnot
        neu_juv_total_click += len(neu_clicks)
        neu_juv_pos_click += len(neu_clicks[neu_clicks == 1])

    ptype_juvenile_pos_click_rate[ptype] = pos_juv_pos_click/pos_juv_total_click
    ptype_juvenile_neg_click_rate[ptype] = neg_juv_pos_click/neg_juv_total_click
    ptype_juvenile_neu_click_rate[ptype] = neu_juv_pos_click/neu_juv_total_click
```

In [1432]:
```
ptype_juv_pos_click_rate = pd.DataFrame\
(ptype_juvenile_pos_click_rate.items(), columns = ["Product_Type", "juvenile_positive_click_rate"])

ptype_juv_neg_click_rate = pd.DataFrame\
(ptype_juvenile_neg_click_rate.items(), columns = ["Product_Type", "juvenile_negative_click_rate"])

ptype_juv_neu_click_rate = pd.DataFrame\
(ptype_juvenile_neu_click_rate.items(), columns = ["Product_Type", "juvenile_neutral_click_rate"])

merg1 = pd.merge(ptype_juv_pos_click_rate, ptype_juv_neg_click_rate)
juv_senti_click_rate = pd.merge(merg1, ptype_juv_neu_click_rate)
juv_senti_click_rate
```

Out[1432]:

	Product_Type	juvenile_positive_click_rate	juvenile_negative_click_rate	juvenile_neutral_click_rate
0	computer	0.517647	0.504673	0.531915
1	tablet	0.509615	0.480000	0.530435
2	perfume	0.493902	0.497537	0.539604
3	speakers	0.475000	0.472803	0.564315
4	furniture	0.424893	0.500000	0.552239
5	vitamin	0.396000	0.528571	0.574394
6	face cream	0.402985	0.555184	0.594771
7	women's purse	0.411565	0.556575	0.607038
8	jeans	0.414286	0.523438	0.602978
9	lipstick	0.433511	0.533333	0.586907
10	makeup	0.424870	0.512941	0.577681
11	pants	0.420918	0.518265	0.579060
12	car	0.386574	0.508475	0.578529
13	shaver	0.383073	0.512671	0.583650
14	refrigerator	0.383648	0.497207	0.573260
15	blender	0.378327	0.522613	0.551089
16	pressure cooker	0.386617	0.511475	0.562399
17	coffee	0.363322	0.507764	0.549317
18	dryer	0.362460	0.495601	0.563380
19	television	0.368502	0.490960	0.567385
20	washer	0.395062	0.485969	0.559850
21	elliptical trainer	0.395442	0.486181	0.566382
22	rowing machine	0.386423	0.478580	0.557554
23	treadmill	0.386973	0.476591	0.556461

In [1433]:
```
ptype_young_pos_click_rate = {}
ptype_young_neg_click_rate = {}
ptype_young_neu_click_rate = {}
pos_y_total_click = 0
pos_y_pos_click = 0
neg_y_total_click = 0
neg_y_pos_click = 0
neu_y_total_click = 0
neu_y_pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        pos_clicks = log[(log.ProductURL == url) & (log.Sentiment == "positive") & (log.age_group == "young")].clickORnot
        pos_y_total_click += len(pos_clicks)
        pos_y_pos_click += len(pos_clicks[pos_clicks == 1])

        neg_clicks = log[(log.ProductURL == url) & (log.Sentiment == "negative") & (log.age_group == "young")].clickORnot
        neg_y_total_click += len(neg_clicks)
        neg_y_pos_click += len(neg_clicks[neg_clicks == 1])

        neu_clicks = log[(log.ProductURL == url) & (log.Sentiment == "neutral") & (log.age_group == "young")].clickORnot
        neu_y_total_click += len(neu_clicks)
        neu_y_pos_click += len(neu_clicks[neu_clicks == 1])


    ptype_young_pos_click_rate[ptype] = pos_y_pos_click/pos_y_total_click
    ptype_young_neg_click_rate[ptype] = neg_y_pos_click/neg_y_total_click
    ptype_young_neu_click_rate[ptype] = neu_y_pos_click/neu_y_total_click
```

In [1434]:
```
ptype_y_pos_click_rate = pd.DataFrame\
(ptype_young_pos_click_rate.items(), columns = ["Product_Type", "young_positive_click_rate"])

ptype_y_neg_click_rate = pd.DataFrame\
(ptype_young_neg_click_rate.items(), columns = ["Product_Type", "young_negative_click_rate"])

ptype_y_neu_click_rate = pd.DataFrame\
(ptype_young_neu_click_rate.items(), columns = ["Product_Type", "young_neutral_click_rate"])

merg1 = pd.merge(ptype_y_pos_click_rate, ptype_y_neg_click_rate)
young_senti_click_rate = pd.merge(merg1, ptype_y_neu_click_rate)
young_senti_click_rate
```

Out[1434]:

	Product_Type	young_positive_click_rate	young_negative_click_rate	young_neutral_click_rate
0	computer	0.648936	0.468085	0.584158
1	tablet	0.618644	0.460870	0.598425
2	perfume	0.581522	0.480000	0.631313
3	speakers	0.597285	0.443396	0.654321
4	furniture	0.569767	0.485477	0.655797
5	vitamin	0.551601	0.515267	0.668896
6	face cream	0.552980	0.542553	0.689655
7	women's purse	0.533898	0.517241	0.695157
8	jeans	0.538095	0.476316	0.690000
9	lipstick	0.560000	0.490291	0.680653
10	makeup	0.556034	0.473193	0.670379
11	pants	0.550420	0.488789	0.672489
12	car	0.517176	0.479253	0.660000
13	shaver	0.515483	0.486275	0.662813
14	refrigerator	0.512238	0.465421	0.653285
15	blender	0.508013	0.499160	0.621087
16	pressure cooker	0.514961	0.488487	0.625402
17	coffee	0.504505	0.486656	0.610350
18	dryer	0.495063	0.484581	0.617391
19	television	0.505305	0.471154	0.619835
20	washer	0.529340	0.467516	0.615776
21	elliptical trainer	0.527141	0.468244	0.617794
22	rowing machine	0.530249	0.466258	0.613665
23	treadmill	0.530707	0.463855	0.612394

In [1435]:
```
ptype_middle_pos_click_rate = {}
ptype_middle_neg_click_rate = {}
ptype_middle_neu_click_rate = {}
pos_m_total_click = 0
pos_m_pos_click = 0
neg_m_total_click = 0
neg_m_pos_click = 0
neu_m_total_click = 0
neu_m_pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        pos_clicks = log[(log.ProductURL == url) & (log.Sentiment == "positive") & (log.age_group == "middle-age")].clickORnot
        pos_m_total_click += len(pos_clicks)
        pos_m_pos_click += len(pos_clicks[pos_clicks == 1])

        neg_clicks = log[(log.ProductURL == url) & (log.Sentiment == "negative") & (log.age_group == "middle-age")].clickORnot
        neg_m_total_click += len(neg_clicks)
        neg_m_pos_click += len(neg_clicks[neg_clicks == 1])

        neu_clicks = log[(log.ProductURL == url) & (log.Sentiment == "neutral") & (log.age_group == "middle-age")].clickORnot
        neu_m_total_click += len(neu_clicks)
        neu_m_pos_click += len(neu_clicks[neu_clicks == 1])


    ptype_middle_pos_click_rate[ptype] = pos_m_pos_click/pos_m_total_click
    ptype_middle_neg_click_rate[ptype] = neg_m_pos_click/neg_m_total_click
    ptype_middle_neu_click_rate[ptype] = neu_m_pos_click/neu_m_total_click
```

In [1436]:
```
ptype_m_pos_click_rate = pd.DataFrame\
(ptype_middle_pos_click_rate.items(), columns = ["Product_Type", "Middle_age_positive_click_rate"])

ptype_m_neg_click_rate = pd.DataFrame\
(ptype_middle_neg_click_rate.items(), columns = ["Product_Type", "Middle_age__negative_click_rate"])

ptype_m_neu_click_rate = pd.DataFrame\
(ptype_middle_neu_click_rate.items(), columns = ["Product_Type", "Middle_age_neutral_click_rate"])

merg1 = pd.merge(ptype_m_pos_click_rate, ptype_m_neg_click_rate)
middle_senti_click_rate = pd.merge(merg1, ptype_m_neu_click_rate)
middle_senti_click_rate
```

Out[1436]:

	Product_Type	Middle_age_positive_click_rate	Middle_age__negative_click_rate	Middle_age_neutral_click_rate
0	computer	0.557895	0.559524	0.565217
1	tablet	0.536364	0.514019	0.598214
2	perfume	0.526596	0.536458	0.578035
3	speakers	0.507042	0.527027	0.590000
4	furniture	0.485714	0.569170	0.578059
5	vitamin	0.463602	0.582418	0.592157
6	face cream	0.476534	0.602076	0.612319
7	women's purse	0.462025	0.583851	0.599349
8	jeans	0.458115	0.518135	0.596591
9	lipstick	0.479115	0.523227	0.581818
10	makeup	0.466825	0.513002	0.562654
11	pants	0.462069	0.531532	0.576190
12	car	0.427386	0.520576	0.569536
13	shaver	0.425150	0.522330	0.567568
14	refrigerator	0.420455	0.509398	0.554672
15	blender	0.410435	0.534338	0.531646
16	pressure cooker	0.417085	0.516181	0.539964
17	coffee	0.412975	0.519260	0.528523
18	dryer	0.399404	0.502882	0.539185
19	television	0.404762	0.492476	0.551775
20	washer	0.417834	0.474811	0.542234
21	elliptical trainer	0.414695	0.474074	0.548732
22	rowing machine	0.414545	0.463855	0.540470
23	treadmill	0.421115	0.458382	0.543257

In [1437]:
```
ptype_senior_pos_click_rate = {}
ptype_senior_neg_click_rate = {}
ptype_senior_neu_click_rate = {}
pos_s_total_click = 0
pos_s_pos_click = 0
neg_s_total_click = 0
neg_s_pos_click = 0
neu_s_total_click = 0
neu_s_pos_click = 0

for ptype in prodtype_dict.keys():
    for url in prodtype_dict[ptype]:
        pos_clicks = log[(log.ProductURL == url) & (log.Sentiment == "positive") & (log.age_group == "senior")].clickORnot
        #print(pos_clicks)
        pos_s_total_click += len(pos_clicks)
        pos_s_pos_click += len(pos_clicks[pos_clicks == 1])

        neg_clicks = log[(log.ProductURL == url) & (log.Sentiment == "negative") & (log.age_group == "senior")].clickORnot
        #print(neg_clicks)
        neg_s_total_click += len(neg_clicks)
        neg_s_pos_click += len(neg_clicks[neg_clicks == 1])

        neu_clicks = log[(log.ProductURL == url) & (log.Sentiment == "neutral") & (log.age_group == "senior")].clickORnot
        #print(neu_clicks)
        neu_s_total_click += len(neu_clicks)
        neu_s_pos_click += len(neu_clicks[neu_clicks == 1])


    ptype_senior_pos_click_rate[ptype] = pos_s_pos_click/pos_s_total_click
    ptype_senior_neg_click_rate[ptype] = neg_s_pos_click/neg_s_total_click
    ptype_senior_neu_click_rate[ptype] = neu_s_pos_click/neu_s_total_click
```

In [1438]:
```
ptype_s_pos_click_rate = pd.DataFrame\
(ptype_senior_pos_click_rate.items(), columns = ["Product_Type", "Senior_positive_click_rate"])

ptype_s_neg_click_rate = pd.DataFrame\
(ptype_senior_neg_click_rate.items(), columns = ["Product_Type", "Senior__negative_click_rate"])

ptype_s_neu_click_rate = pd.DataFrame\
(ptype_senior_neu_click_rate.items(), columns = ["Product_Type", "Senior_neutral_click_rate"])

merg1 = pd.merge(ptype_s_pos_click_rate, ptype_s_neg_click_rate)
senior_senti_click_rate = pd.merge(merg1, ptype_s_neu_click_rate)
senior_senti_click_rate
```

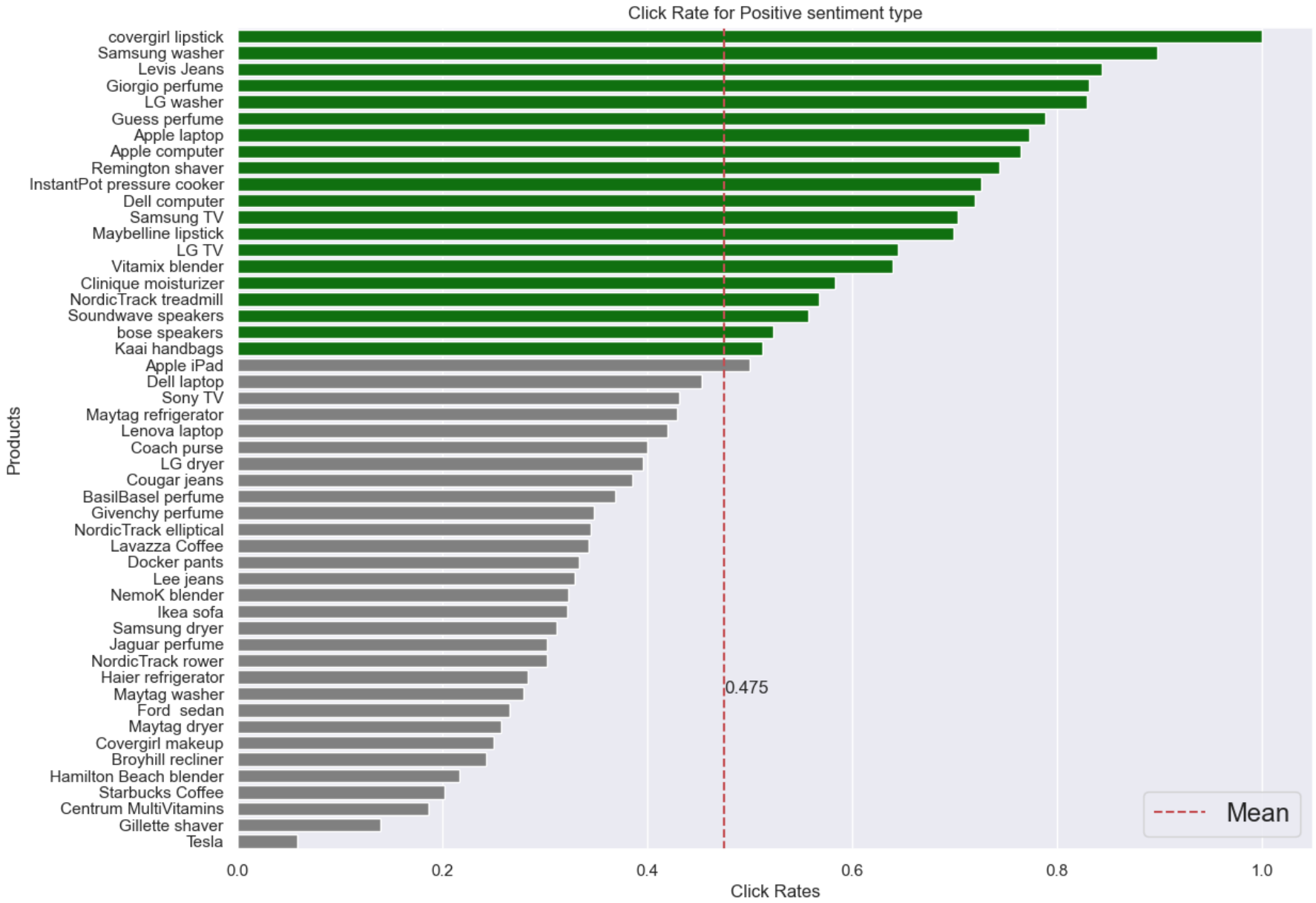
Out[1438]:

	Product_Type	Senior_positive_click_rate	Senior__negative_click_rate	Senior_neutral_click_rate
0	computer	0.693182	0.476190	0.602410
1	tablet	0.663793	0.464912	0.598039
2	perfume	0.646465	0.519553	0.619565
3	speakers	0.651064	0.502463	0.626794
4	furniture	0.606870	0.553719	0.632653
5	vitamin	0.580071	0.578947	0.643939
6	face cream	0.580537	0.593640	0.658273
7	women's purse	0.580838	0.558360	0.675926
8	jeans	0.584810	0.510417	0.670241
9	lipstick	0.605701	0.530562	0.670823
10	makeup	0.596774	0.514085	0.665877
11	pants	0.591518	0.524887	0.669746
12	car	0.574113	0.516736	0.674370
13	shaver	0.573413	0.508911	0.674747
14	refrigerator	0.560461	0.493384	0.667315
15	blender	0.549123	0.510274	0.643234
16	pressure cooker	0.555366	0.490939	0.647160
17	coffee	0.541935	0.486280	0.637110
18	dryer	0.530303	0.472934	0.643411
19	television	0.541963	0.463186	0.647826
20	washer	0.548638	0.457002	0.644860
21	elliptical trainer	0.545338	0.458182	0.641361
22	rowing machine	0.536250	0.456392	0.635309
23	treadmill	0.537241	0.451688	0.634207

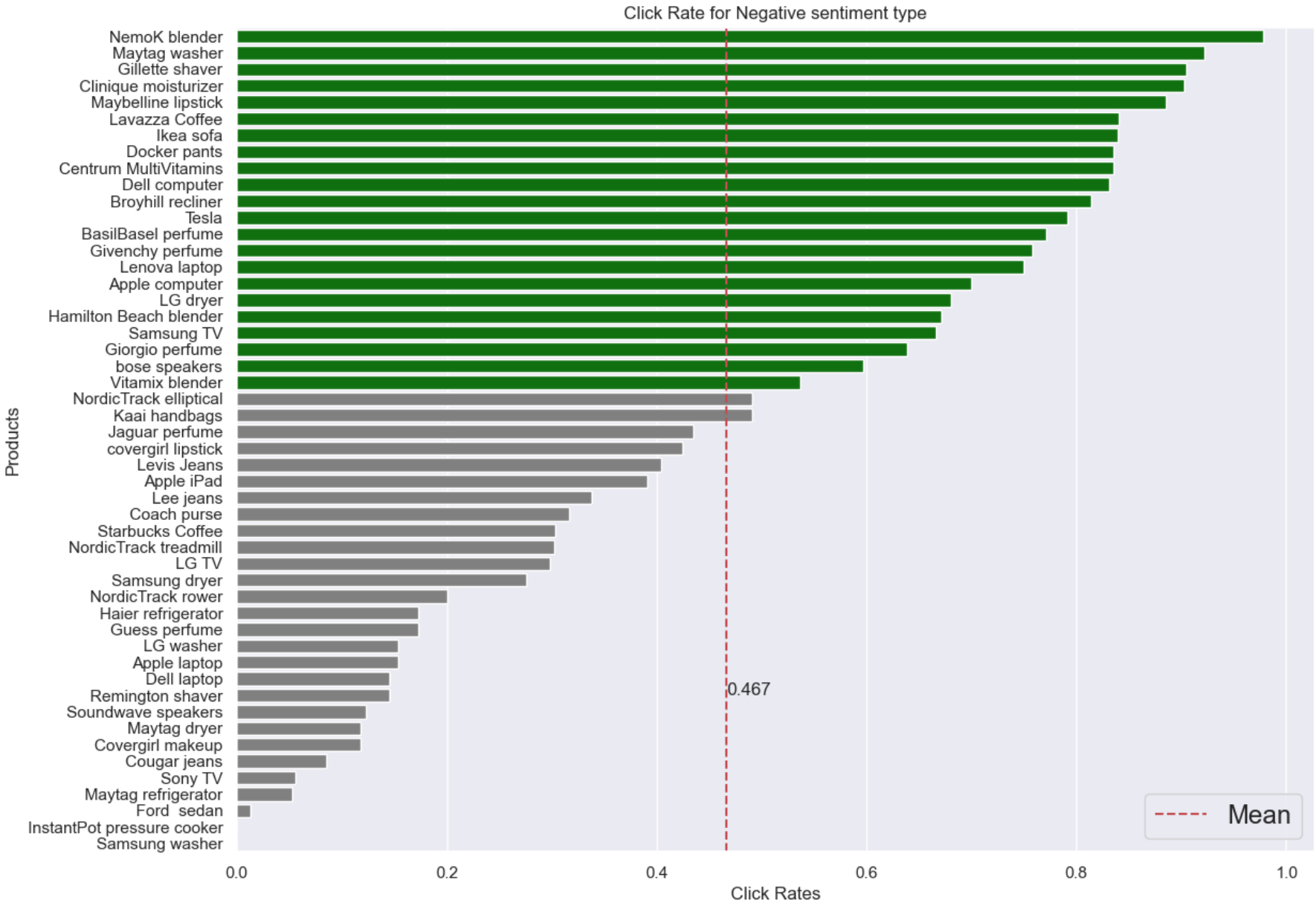
13. Based on your results make your recommendations. These should be in the form:
    a. Based on our analysis (give details of your analysis), ads for such and such product are most likely to produce clicks in such and sentiment context (or state that we see no correlation between click rate of an ad for a product and the sentiment context of the ad)

Answer: The products in the plots are arranged in decending order of their click rate
In the below plots the vertical red dotted line represents the average clickrate of all products for a particular sentiment.
The bars(horizontal height) of the products are green when the the click rate is more than 0.5 and it is grey when it is less than 0.5

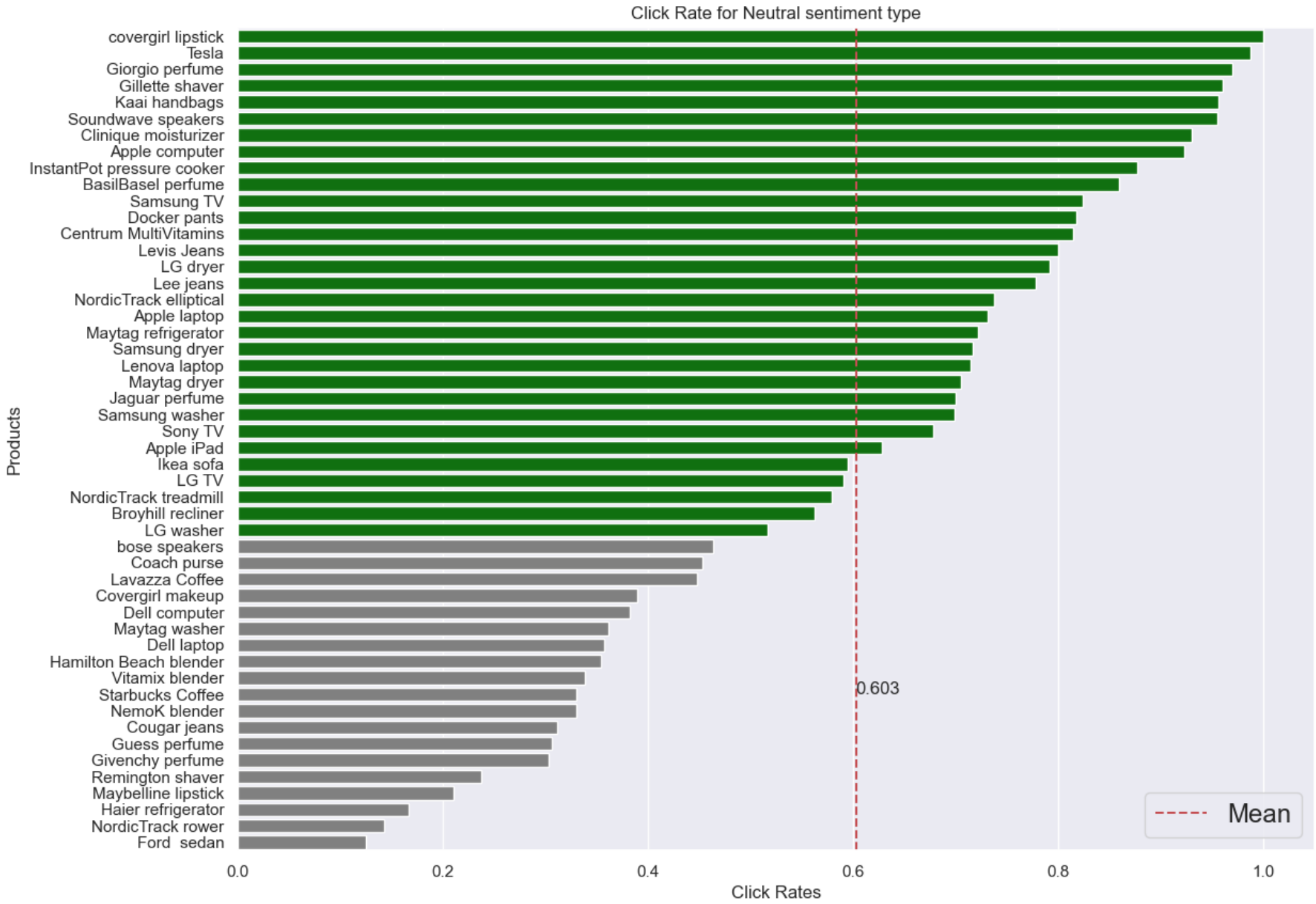
In [1439]:
```
p_senti = prod_senti_click_rate.sort_values('positive_Click_rate',ascending = False).positive_Click_rate
clrs = ['green' if (x > 0.5) else 'grey' for x in p_senti]#green when click rate is more than 0.5

ax = sns.barplot(data = prod_senti_click_rate, x = 'positive_Click_rate', y = 'Product',\
            order = prod_senti_click_rate.sort_values('positive_Click_rate',ascending = False).Product,\
                 palette=clrs
           )
ax.axvline(np.mean(prod_senti_click_rate['positive_Click_rate']),color='r', linestyle='--', label="Mean")
plt.text(np.mean(prod_senti_click_rate['positive_Click_rate'])\
         ,40,np.round(np.mean(prod_senti_click_rate['positive_Click_rate']),3))

ax.set_title('Click Rate for Positive sentiment type')
ax.set_ylabel('Products')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'lower right',fontsize='x-large', title_fontsize='40')
plt.show()
```


Click Rate for Positive sentiment type
covergirl lipstick
Samsung washer
Levis Jeans
Giorgio perfume
LG washer
Guess perfume
Apple laptop
Apple computer
Remington shaver
InstantPot pressure cooker
Dell computer
Samsung TV
Maybelline lipstick
LG TV
Vitamix blender
Clinique moisturizer
NordicTrack treadmill
Soundwave speakers
bose speakers
Kaai handbags
Apple iPad
Dell laptop
Sony TV
Maytag refrigerator
Lenova laptop
Coach purse
LG dryer
Cougar jeans
BasilBasel perfume
Givenchy perfume
NordicTrack elliptical
Lavazza Coffee
Docker pants
Lee jeans
NemoK blender
Ikea sofa
Samsung dryer
Jaguar perfume
NordicTrack rower
Haier refrigerator
Maytag washer
Ford sedan
Maytag dryer
Covergirl makeup
Broyhill recliner
Hamilton Beach blender
Starbucks Coffee
Centrum MultiVitamins
Gillette shaver
Tesla
Products
0.475
Mean
0.0
0.2
0.4
0.6
0.8
1.0
Click Rates

In [1403]:
```
n_senti = prod_senti_click_rate.sort_values('negative_Click_rate',ascending = False).negative_Click_rate
clrs = ['green' if (x > 0.5) else 'grey' for x in n_senti]

ax = sns.barplot(data = prod_senti_click_rate, x = 'negative_Click_rate', y = 'Product',\
            order = prod_senti_click_rate.sort_values('negative_Click_rate',ascending = False).Product,\
                 palette=clrs
           )
ax.axvline(np.mean(prod_senti_click_rate['negative_Click_rate']),color='r', linestyle='--', label="Mean")
plt.text(np.mean(prod_senti_click_rate['negative_Click_rate'])\
         ,40,np.round(np.mean(prod_senti_click_rate['negative_Click_rate']),3))
ax.set_title('Click Rate for Negative sentiment type')
ax.set_ylabel('Products')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'lower right',fontsize='x-large', title_fontsize='40')
plt.show()
```


Click Rate for Negative sentiment type
NemoK blender
Maytag washer
Gillette shaver
Clinique moisturizer
Maybelline lipstick
Lavazza Coffee
Ikea sofa
Docker pants
Centrum MultiVitamins
Dell computer
Broyhill recliner
Tesla
BasilBasel perfume
Givenchy perfume
Lenova laptop
Apple computer
LG dryer
Hamilton Beach blender
Samsung TV
Giorgio perfume
bose speakers
Vitamix blender
NordicTrack elliptical
Kaai handbags
Jaguar perfume
covergirl lipstick
Levis Jeans
Apple iPad
Lee jeans
Coach purse
Starbucks Coffee
NordicTrack treadmill
LG TV
Samsung dryer
NordicTrack rower
Haier refrigerator
Guess perfume
LG washer
Apple laptop
Dell laptop
Remington shaver
Soundwave speakers
Maytag dryer
Covergirl makeup
Cougar jeans
Sony TV
Maytag refrigerator
Ford sedan
InstantPot pressure cooker
Samsung washer
Products
0.467
Mean
0.0
0.2
0.4
0.6
0.8
1.0
Click Rates

In [1404]:
```
neu_senti = prod_senti_click_rate.sort_values('neutral_Click_rate',ascending = False).neutral_Click_rate
clrs = ['green' if (x > 0.5) else 'grey' for x in neu_senti]

ax = sns.barplot(data = prod_senti_click_rate, x = 'neutral_Click_rate', y = 'Product',\
            order = prod_senti_click_rate.sort_values('neutral_Click_rate',ascending = False).Product,\
                 palette=clrs
           )
ax.axvline(np.mean(prod_senti_click_rate['neutral_Click_rate']),color='r', linestyle='--', label="Mean")
plt.text(np.mean(prod_senti_click_rate['neutral_Click_rate'])\
         ,40,np.round(np.mean(prod_senti_click_rate['neutral_Click_rate']),3))
ax.set_title('Click Rate for Neutral sentiment type')
ax.set_ylabel('Products')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'lower right',fontsize='x-large', title_fontsize='40')
plt.show()
```



Based on the analysis done above, I can say that the mean click rate for all the products is higher when the sentiment type is neutral.
I can make following recommendations:
(1)Covergirl lipstick, samsung washer,Levis Jeans, LG washer,Guess Perfume get more clicks when the sentiment type is positive.
(2)Tesla, Giorgio perfume,Kaai handbags,soundwave speakers, Apple computer,InstantPot Pressure cooker
gets more clicks when the sentiment type is neutral.
(3)NemoK Blender,Maytag Washer,Ikea sofa gets more clicks when the sentiment type is negative.

b. Based on our analysis (with details), ads for such and such product are most likely to produce clicks in such and sentiment context by viewers of such and such gender (or state that we see no correlation between click rate of an ad for a product and the sentiment context of the ad and the gender of the viewer).

In [1440]:
```
ptype_pos_senti_click_rate = ptype_senti_click_rate\
[['Product_Type','Male_positive_Click_rate','Female_positive_Click_rate']]
```

In [1441]:
```
ptype_pos_senti_click_rate = ptype_pos_senti_click_rate.melt(id_vars="Product_Type", var_name="Gender")
```

In [1442]:
```
ptype_pos_senti_click_rate.columns = ['Product_Type','Gender','Click_rate']
```

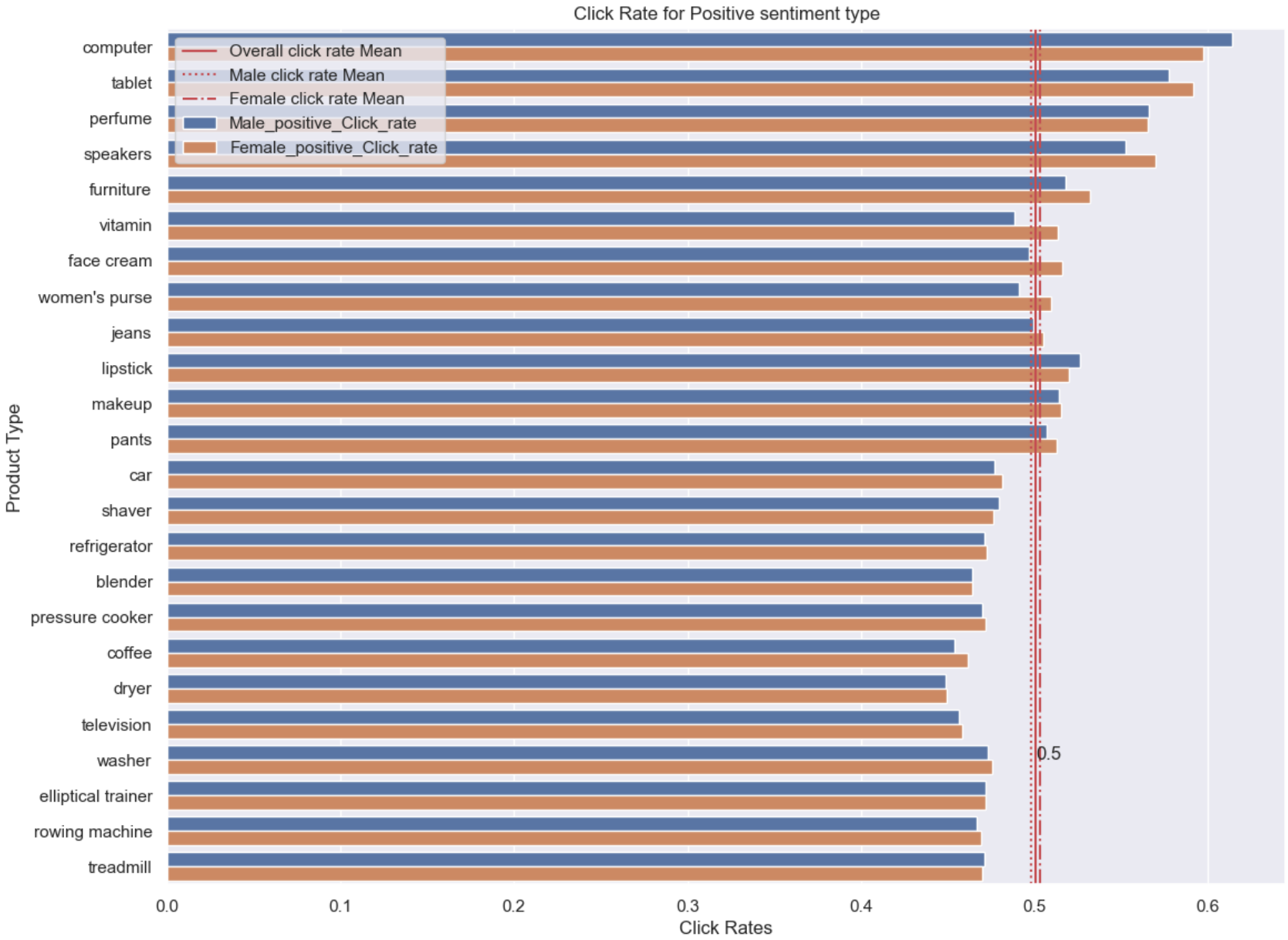
In [1443]:
```
ptype_pos_senti_click_rate.head()
```

Out[1443]:

	Product_Type	Gender	Click_rate
0	computer	Male_positive_Click_rate	0.614035
1	tablet	Male_positive_Click_rate	0.577465
2	perfume	Male_positive_Click_rate	0.565826
3	speakers	Male_positive_Click_rate	0.552632
4	furniture	Male_positive_Click_rate	0.517526

In [1444]:

```
ax = sns.barplot(data = ptype_pos_senti_click_rate, x = 'Click_rate', y = 'Product_Type',\
                 hue= 'Gender'
            )
ax.axvline(np.mean(ptype_pos_senti_click_rate['Click_rate'])\
           ,color='r', linestyle='-', label="Overall click rate Mean")

ax.axvline(np.mean(ptype_senti_click_rate['Male_positive_Click_rate'])\
           ,color='r', linestyle='dotted', label="Male click rate Mean")

ax.axvline(np.mean(ptype_senti_click_rate['Female_positive_Click_rate'])\
           ,color='r', linestyle='dashdot', label="Female click rate Mean")

plt.text(np.mean(ptype_pos_senti_click_rate['Click_rate'])\
         ,20,np.round(np.mean(ptype_pos_senti_click_rate['Click_rate']),3))

ax.set_title('Click Rate for Positive sentiment type')
ax.set_ylabel('Product Type')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'upper left')
plt.show()
```



In [1411]:
```
ptype_neg_senti_click_rate = ptype_senti_click_rate\
[['Product_Type','Male_negative_Click_rate','Female_negative_Click_rate']]
```

In [1412]:
```
ptype_neg_senti_click_rate = ptype_neg_senti_click_rate.melt(id_vars="Product_Type", var_name="Gender")
```

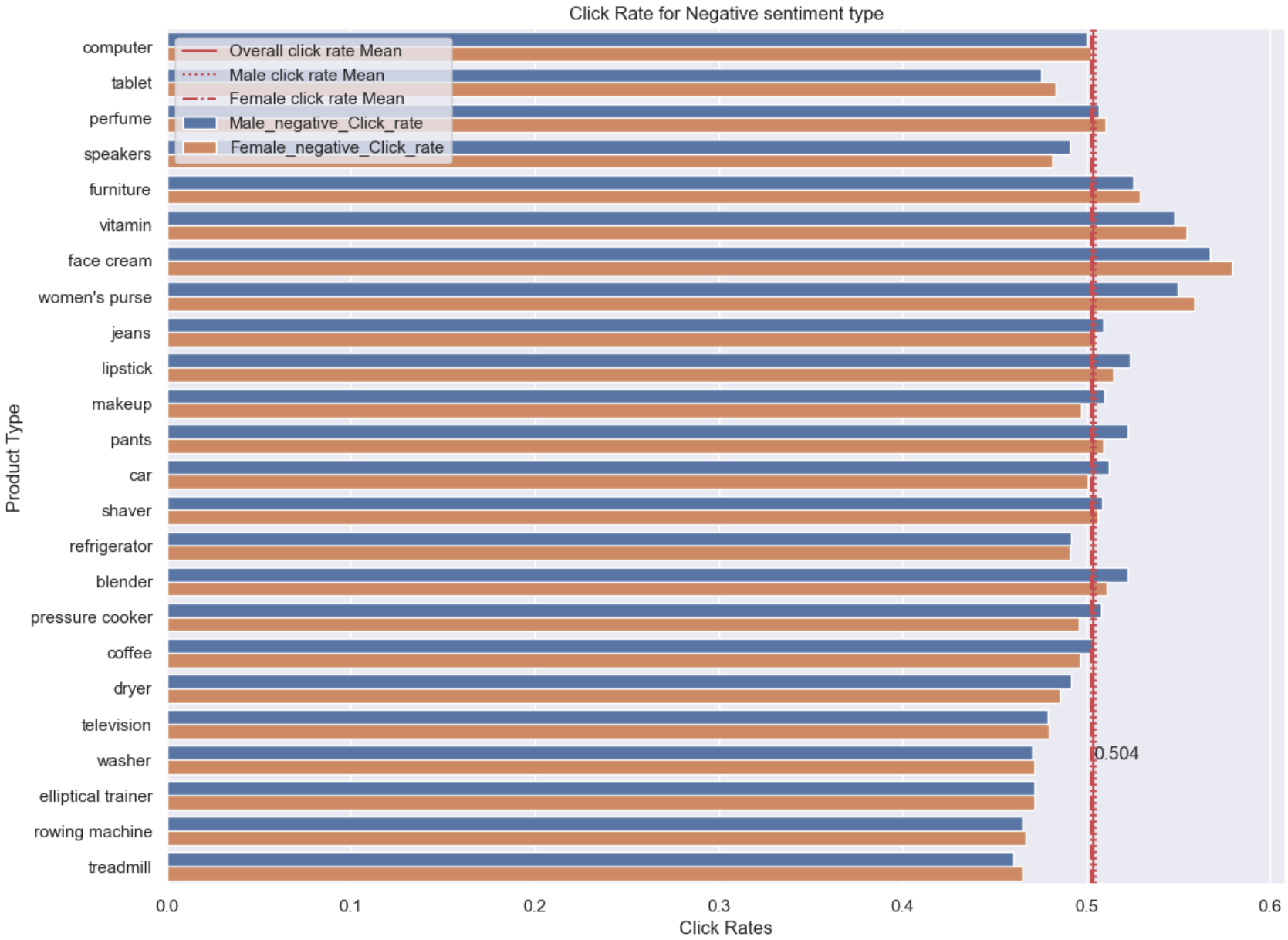
In [1413]:
```
ptype_neg_senti_click_rate.columns = ['Product_Type','Gender','Click_rate']
```

In [1414]:
```
ptype_neg_senti_click_rate.head()
```

Out[1414]:

	Product_Type	Gender	Click_rate
0	computer	Male_negative_Click_rate	0.500000
1	tablet	Male_negative_Click_rate	0.475336
2	perfume	Male_negative_Click_rate	0.506775
3	speakers	Male_negative_Click_rate	0.490909
4	furniture	Male_negative_Click_rate	0.525253

In [1415]:
```
ax = sns.barplot(data = ptype_neg_senti_click_rate, x = 'Click_rate', y = 'Product_Type',\
                 hue= 'Gender')
ax.axvline(np.mean(ptype_neg_senti_click_rate['Click_rate']),color='r'\
           , linestyle='-', label="Overall click rate Mean")

ax.axvline(np.mean(ptype_senti_click_rate['Male_negative_Click_rate'])\
           ,color='r', linestyle='dotted', label="Male click rate Mean")

ax.axvline(np.mean(ptype_senti_click_rate['Female_negative_Click_rate'])\
           ,color='r', linestyle='dashdot', label="Female click rate Mean")

plt.text(np.mean(ptype_neg_senti_click_rate['Click_rate'])\
         ,20,np.round(np.mean(ptype_neg_senti_click_rate['Click_rate']),3))

ax.set_title('Click Rate for Negative sentiment type')
ax.set_ylabel('Product Type')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'upper left')
plt.show()
```

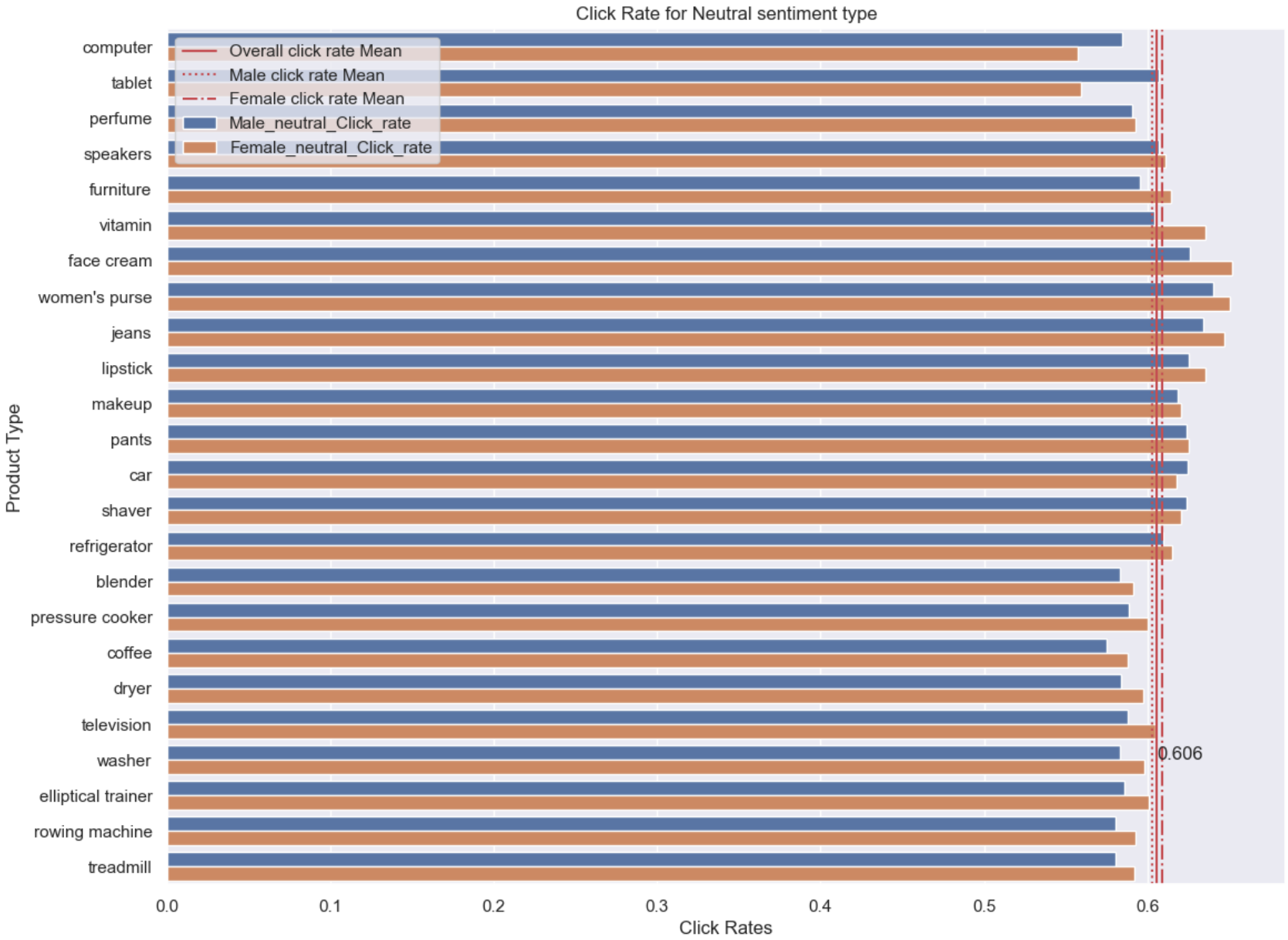

Click Rate for Negative sentiment type
Overall click rate Mean
Male click rate Mean
Female click rate Mean
Male_negative_Click_rate
Female_negative_Click_rate
computer
tablet
perfume
speakers
furniture
vitamin
face cream
women's purse
jeans
lipstick
makeup
pants
car
shaver
refrigerator
blender
pressure cooker
coffee
dryer
television
washer
elliptical trainer
rowing machine
treadmill
Product Type
0.504
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Click Rates

In [1416]:
```
ptype_neu_senti_click_rate = ptype_senti_click_rate\
[['Product_Type','Male_neutral_Click_rate','Female_neutral_Click_rate']]

ptype_neu_senti_click_rate = ptype_neu_senti_click_rate.melt(id_vars="Product_Type", var_name="Gender")
ptype_neu_senti_click_rate.columns = ['Product_Type','Gender','Click_rate']
ptype_neu_senti_click_rate.head()
```

Out[1416]:

	Product_Type	Gender	Click_rate
0	computer	Male_neutral_Click_rate	0.584270
1	tablet	Male_neutral_Click_rate	0.606635
2	perfume	Male_neutral_Click_rate	0.590659
3	speakers	Male_neutral_Click_rate	0.606977
4	furniture	Male_neutral_Click_rate	0.595142

In [1446]:
```
ax = sns.barplot(data = ptype_neu_senti_click_rate, x = 'Click_rate', y = 'Product_Type',\
                 hue= 'Gender')
ax.axvline(np.mean(ptype_neu_senti_click_rate['Click_rate'])\
           ,color='r', linestyle='-', label="Overall click rate Mean")

ax.axvline(np.mean(ptype_senti_click_rate['Male_neutral_Click_rate'])\
           ,color='r', linestyle='dotted', label="Male click rate Mean")

ax.axvline(np.mean(ptype_senti_click_rate['Female_neutral_Click_rate'])\
           ,color='r', linestyle='dashdot', label="Female click rate Mean")

plt.text(np.mean(ptype_neu_senti_click_rate['Click_rate'])\
         ,20,np.round(np.mean(ptype_neu_senti_click_rate['Click_rate']),3))

ax.set_title('Click Rate for Neutral sentiment type')
ax.set_ylabel('Product Type')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'upper left')
plt.show()
```



Based on the analysis done above, I can say that the mean click rate for all the product types
is higher when the sentiment type is neutral and gender is female.
From the above analysis we can say that on average click rate by women is slightly more than men on all the sentiment types.

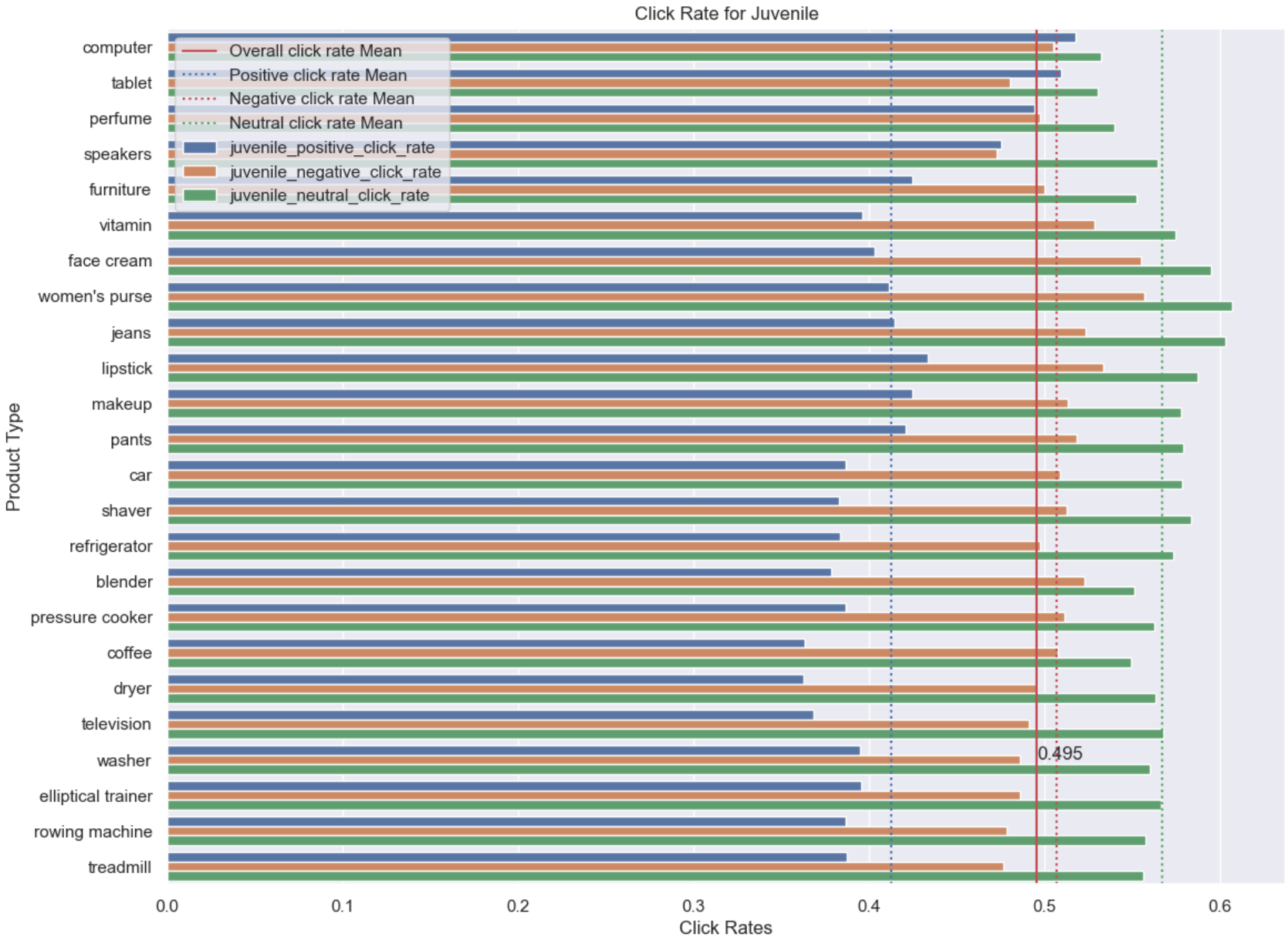
Based on above analysis, I can make following recommendation:
(1)Face cream, vitamins, women's purse, jeans, lipstick, television has higher click rate by women when the sentiment type is neutral.
(2)tablet, car,shaver has higher click rate by men when the when the sentiment type is neutral.
(3)computer has higher click rate by men when the sentiment is positive.

c. Based on our analysis (with details), ads for such and such product are most likely to produce clicks in such and sentiment context by viewers of such and such age-group (or state that we see no correlation between click rate of an ad for a product and the sentiment context of the ad and the age-group of the viewer).

In [1453]:
```
ptype_juv_senti_click_rate = juv_senti_click_rate.melt(id_vars="Product_Type", var_name="Gender")
```

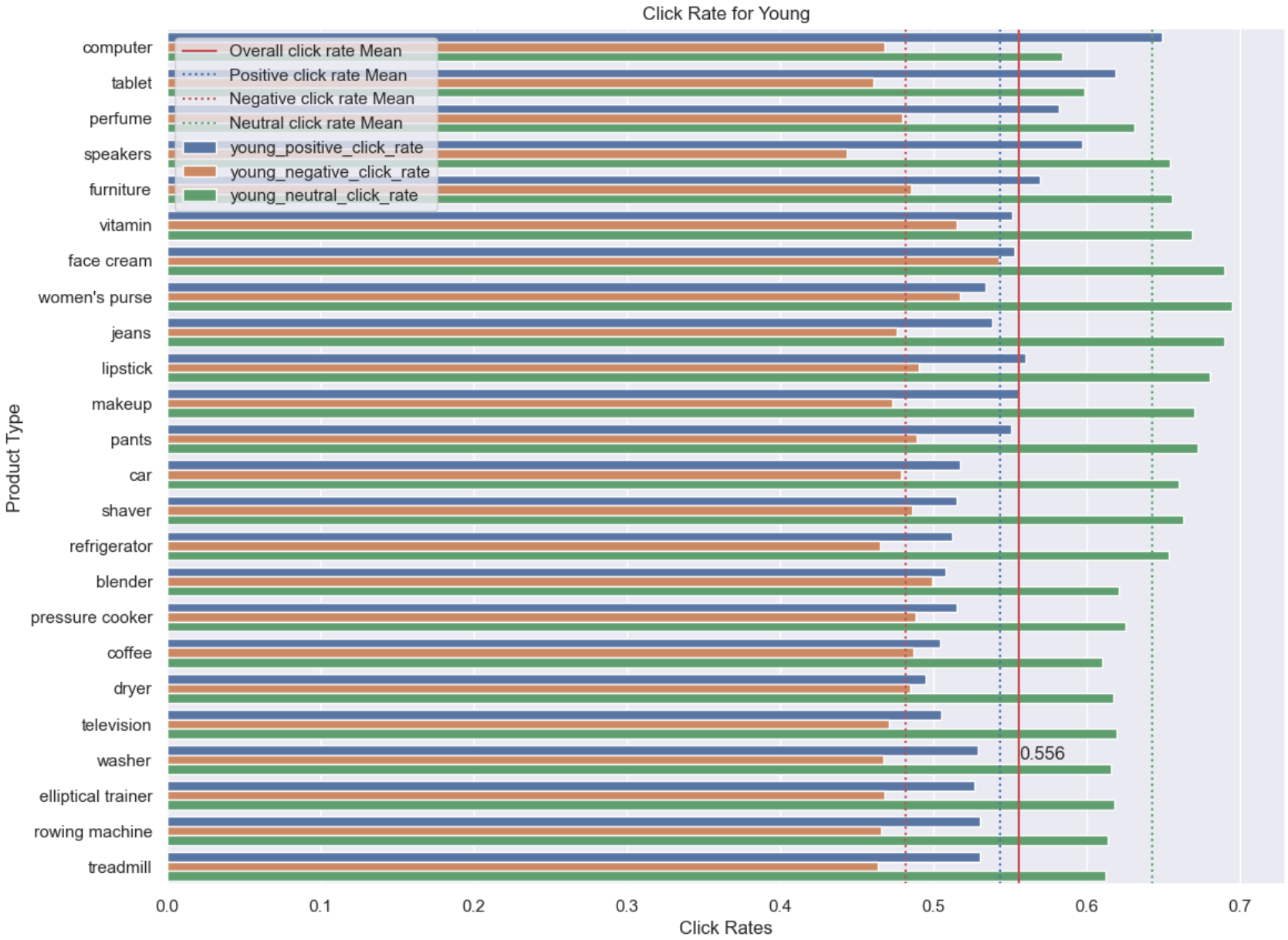
In [1454]:
```
ptype_juv_senti_click_rate.columns = ['Product_Type','Sentiment_Type','Click_rate']
```

In [1455]:
```
ax = sns.barplot(data = ptype_juv_senti_click_rate, x = 'Click_rate', y = 'Product_Type',\
                 hue= 'Sentiment_Type')
ax.axvline(np.mean(ptype_juv_senti_click_rate['Click_rate']),color='r'\
           , linestyle='-', label="Overall click rate Mean")

ax.axvline(np.mean(juv_senti_click_rate['juvenile_positive_click_rate'])\
           ,color='b', linestyle='dotted', label="Positive click rate Mean")

ax.axvline(np.mean(juv_senti_click_rate['juvenile_negative_click_rate'])
           ,color='r', linestyle='dotted', label="Negative click rate Mean")

ax.axvline(np.mean(juv_senti_click_rate['juvenile_neutral_click_rate'])\
           ,color='g', linestyle='dotted', label="Neutral click rate Mean")

plt.text(np.mean(ptype_juv_senti_click_rate['Click_rate'])\
         ,20,np.round(np.mean(ptype_juv_senti_click_rate['Click_rate']),3))

ax.set_title('Click Rate for Juvenile')
ax.set_ylabel('Product Type')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'upper left')
plt.show()
```



In [1456]:
```
ptype_young_senti_click_rate = young_senti_click_rate.\
melt(id_vars="Product_Type", var_name="Gender")
ptype_young_senti_click_rate.columns = ['Product_Type','Sentiment_Type','Click_rate']
```

In [1457]:
```
ax = sns.barplot(data = ptype_young_senti_click_rate, x = 'Click_rate', y = 'Product_Type',\
                 hue= 'Sentiment_Type')
ax.axvline(np.mean(ptype_young_senti_click_rate['Click_rate']),color='r'\
           , linestyle='-', label="Overall click rate Mean")
ax.axvline(np.mean(young_senti_click_rate['young_positive_click_rate'])\
           ,color='b', linestyle='dotted', label="Positive click rate Mean")

ax.axvline(np.mean(young_senti_click_rate['young_negative_click_rate'])\
           ,color='r', linestyle='dotted', label="Negative click rate Mean")

ax.axvline(np.mean(young_senti_click_rate['young_neutral_click_rate'])\
           ,color='g', linestyle='dotted', label="Neutral click rate Mean")

plt.text(np.mean(ptype_young_senti_click_rate['Click_rate'])\
         ,20,np.round(np.mean(ptype_young_senti_click_rate['Click_rate']),3))

ax.set_title('Click Rate for Young')
ax.set_ylabel('Product Type')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'upper left')
plt.show()
```

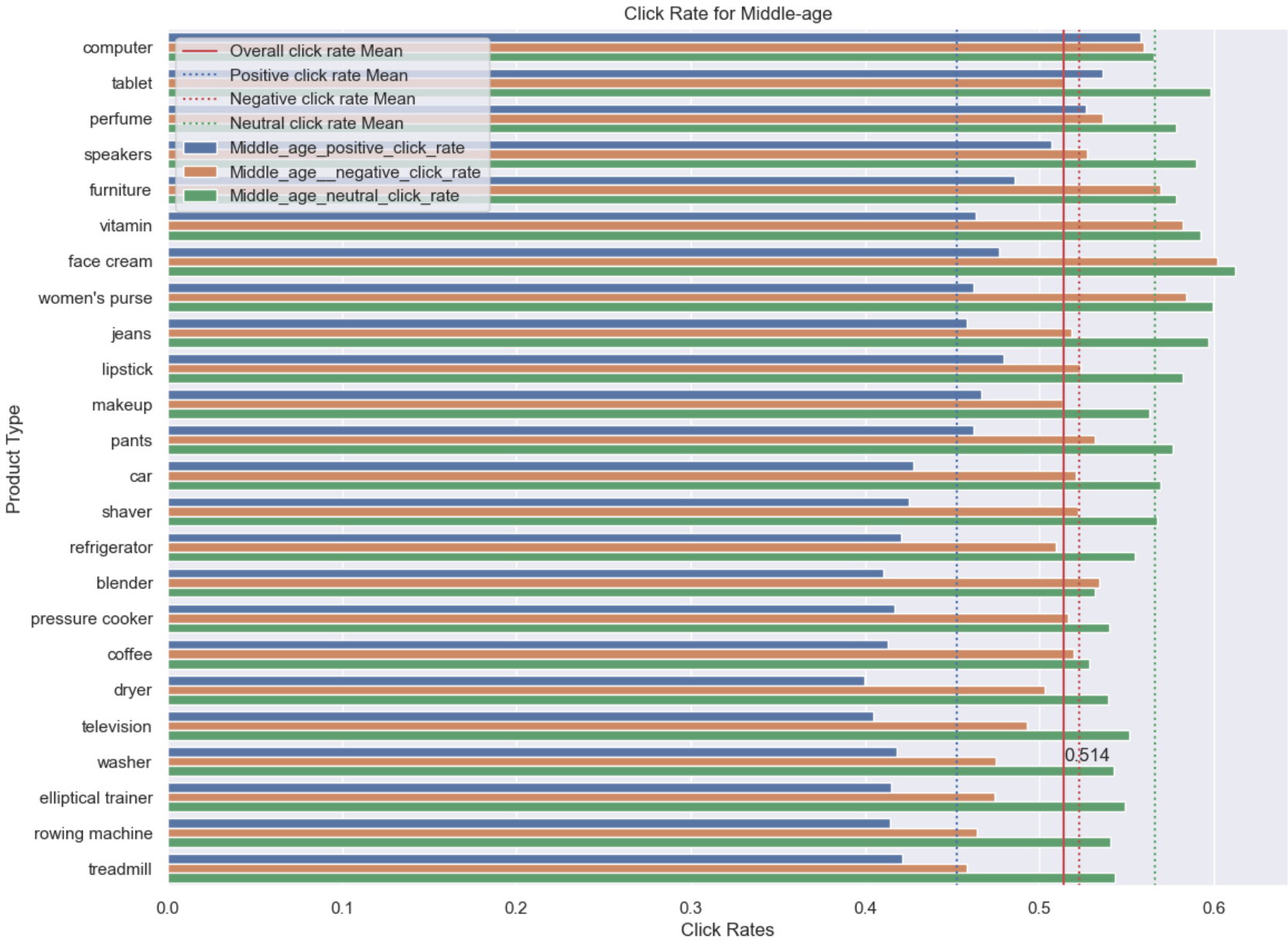

Click Rate for Young
Overall click rate Mean
Positive click rate Mean
Negative click rate Mean
Neutral click rate Mean
young_positive_click_rate
young_negative_click_rate
young_neutral_click_rate
computer
tablet
perfume
speakers
furniture
vitamin
face cream
women's purse
jeans
lipstick
makeup
pants
car
shaver
refrigerator
blender
pressure cooker
coffee
dryer
television
washer
elliptical trainer
rowing machine
treadmill
0.556
Product Type
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Click Rates

In [1458]:
```
ptype_middle_senti_click_rate = middle_senti_click_rate.melt(id_vars="Product_Type", var_name="Gender")
ptype_middle_senti_click_rate.columns = ['Product_Type','Sentiment_Type','Click_rate']
ptype_middle_senti_click_rate.head()
```

Out[1458]:

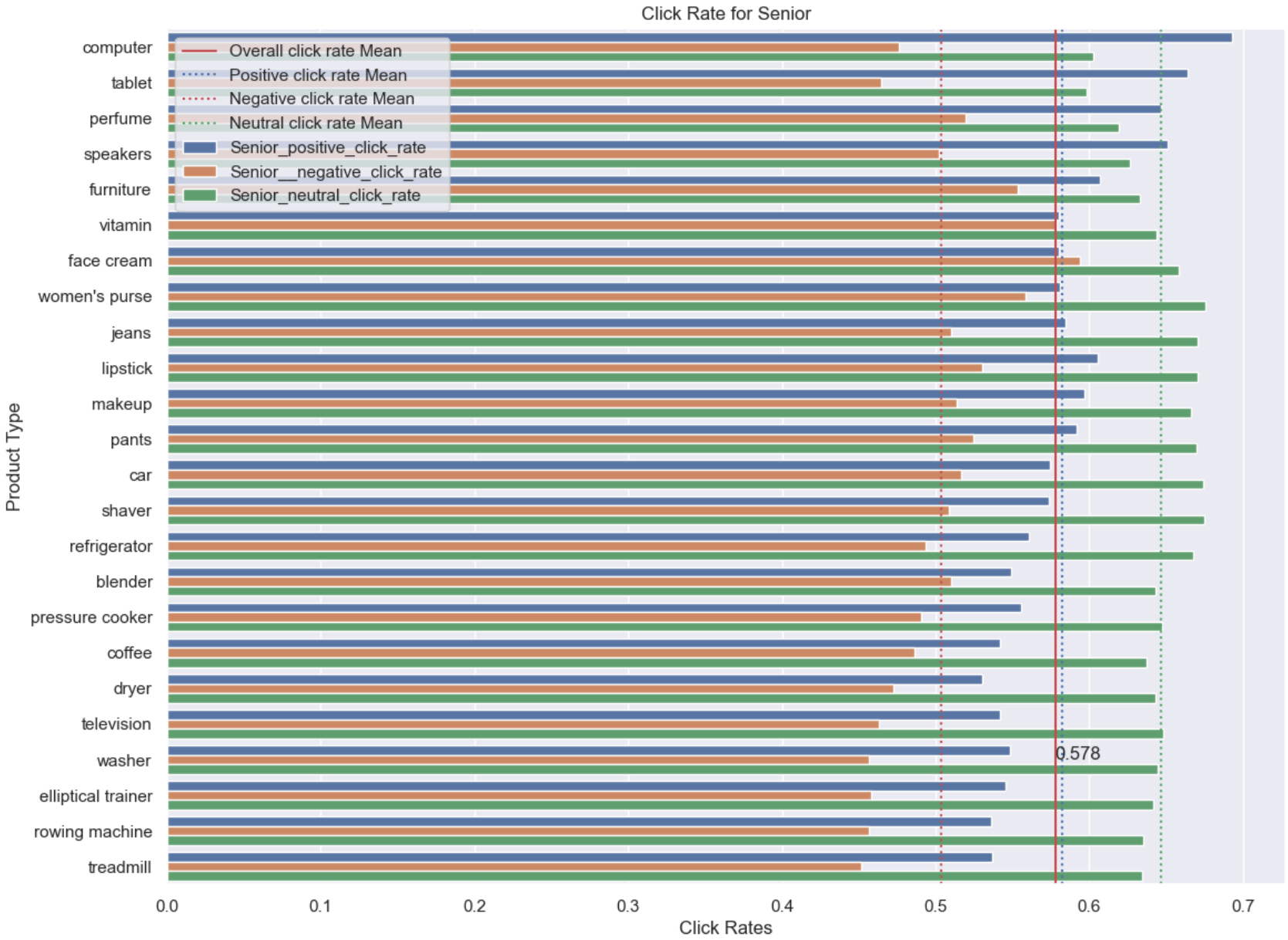
	Product_Type	Sentiment_Type	Click_rate
0	computer	Middle_age_positive_click_rate	0.557895
1	tablet	Middle_age_positive_click_rate	0.536364
2	perfume	Middle_age_positive_click_rate	0.526596
3	speakers	Middle_age_positive_click_rate	0.507042
4	furniture	Middle_age_positive_click_rate	0.485714

In [1459]:
```
ax = sns.barplot(data = ptype_middle_senti_click_rate, x = 'Click_rate', y = 'Product_Type',\
                 hue= 'Sentiment_Type')
ax.axvline(np.mean(ptype_middle_senti_click_rate['Click_rate']),color='r'\
           , linestyle='-', label="Overall click rate Mean")
ax.axvline(np.mean(middle_senti_click_rate['Middle_age_positive_click_rate'])\
           ,color='b', linestyle='dotted', label="Positive click rate Mean")

ax.axvline(np.mean(middle_senti_click_rate['Middle_age__negative_click_rate'])\
           ,color='r', linestyle='dotted', label="Negative click rate Mean")

ax.axvline(np.mean(middle_senti_click_rate['Middle_age_neutral_click_rate'])\
           ,color='g', linestyle='dotted', label="Neutral click rate Mean")

plt.text(np.mean(ptype_middle_senti_click_rate['Click_rate'])\
         ,20,np.round(np.mean(ptype_middle_senti_click_rate['Click_rate']),3))

ax.set_title('Click Rate for Middle-age')
ax.set_ylabel('Product Type')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'upper left')
plt.show()
```



In [1460]:
```
ptype_senior_senti_click_rate = senior_senti_click_rate\
.melt(id_vars="Product_Type", var_name="Gender")
ptype_senior_senti_click_rate.columns = ['Product_Type','Sentiment_Type','Click_rate']
```

In [1462]:
```
ax = sns.barplot(data = ptype_senior_senti_click_rate, x = 'Click_rate', y = 'Product_Type',\
                 hue= 'Sentiment_Type')
ax.axvline(np.mean(ptype_senior_senti_click_rate['Click_rate']),color='r'\
           , linestyle='-', label="Overall click rate Mean")
ax.axvline(np.mean(senior_senti_click_rate['Senior_positive_click_rate'])\
           ,color='b', linestyle='dotted', label="Positive click rate Mean")

ax.axvline(np.mean(senior_senti_click_rate['Senior__negative_click_rate'])\
           ,color='r', linestyle='dotted', label="Negative click rate Mean")

ax.axvline(np.mean(senior_senti_click_rate['Senior_neutral_click_rate'])\
           ,color='g', linestyle='dotted', label="Neutral click rate Mean")

plt.text(np.mean(ptype_senior_senti_click_rate['Click_rate'])\
         ,20,np.round(np.mean(ptype_senior_senti_click_rate['Click_rate']),3))

ax.set_title('Click Rate for Senior')
ax.set_ylabel('Product Type')
ax.set_xlabel('Click Rates')
ax.legend(loc = 'upper left')
plt.show()
```



From the above analysis we can say that the mean click rate of all the products is higher when the age-group is senior and sentiment type is neutral. And it is lowest when the age-group is juvenile.

We can make the following recommendations:
(1)treadmil, rowing machine, elliptical trainer, washer, television has higher clicks when the sentiment is neutral and age-group is senior.
(2)face cream, women's purse, lipstick, jeans, furniture, vitamin has higher clicks when the sentiment is neutral and age-group is young.
(3)computer, tablet, perfume, spekers has higher clicks when the sentiment is positive and age-group is senior.

In [ ]: