globsyn

Taking People To The Next Level

Globsyn finishing school, Globsyn Crystals, 1st Floor, XI – 11 and 12, Block EP, Sector V, Salt Lake, Kolkata – 700091

ANALYSIS OF FLIPKART DATA

USING MACHINE LEARNING WITH PYTHON

Group Members:

KAUSHIK GHOSH, TECHNO INDIA SALT-LAKE, Registration number 161300110046
MUKUND AGARWAL, The Heritage Academy, Registration number 172131010054
RAJDEEP KONER, SWAMI VIVEKANANDA INSTITUTE OF SCIENCE AND TECHNOLOGY, Reg :- 152410110032
SALINI MUKHERJEE, SWAMI VIVEKANANDA INSTITUTE OF SCIENCE AND TECHNOLOGY,Reg:152410110032

Table of Contents

Acknowledgement 4
Project Objective 5
Project Scope 6
Requirement Specification 7
Database Design 8
Application Work Flow Diagram 9
Future Scope of Improvements 10
Code 11
Project Certificate .....

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to our faculty *Prof. Anubhav Chaturvedi* for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of this assignment.

Kaushik Ghosh
Mukund Agarwal
Rajdeep Koner
Salini Mukherjee

Project Objective

To develop a Machine Learning model which:

- Use Python Code to do the following:
  - Perform analysis of the data and use Naive Bayes and Nearest Neighbour models to predict the following:
  - What is the discount percentage for a given product category and brand
- Whether or not the product is a Flipkart advantage product given product category and brand
- Determine the specifications keys corresponding to each product category from the product_specifications column
- Create lists of specification value pairs for each product. The keys should be determined by the product category of that product
- For any particular product category, is there an association rule between some of the product specification values and the product rating?
- Remember that product rating is not available in many cases
- Is the discount percent related to the brand and product rating?

Project Scope

Our project works on a dataset which consists various features attributes about a Product and it's Brand name, Product category,Product specification,Product rating.

Depending on that our model predicts whether a product is Flipkart advantage product or not. Since our model has 96% accuracy (Native Bayes model).

What is the discount percentage for a given product category and brand

Determine the specifications keys corresponding to each product category from the product_specifications column

For any particular product category, is there an association rule between some of the product specification values and the product rating?
Remember that product rating is not available in many cases
Is the discount percent related to the brand and product rating?

Requirement Specification

4GB RAM
- 32- or 64-bit Architecture
- computer. Anaconda Software
- Minimum 3 GB disk space to download and install.
- Windows, macOS or Linux.

Python 3.5 or 3.6.

Database Design

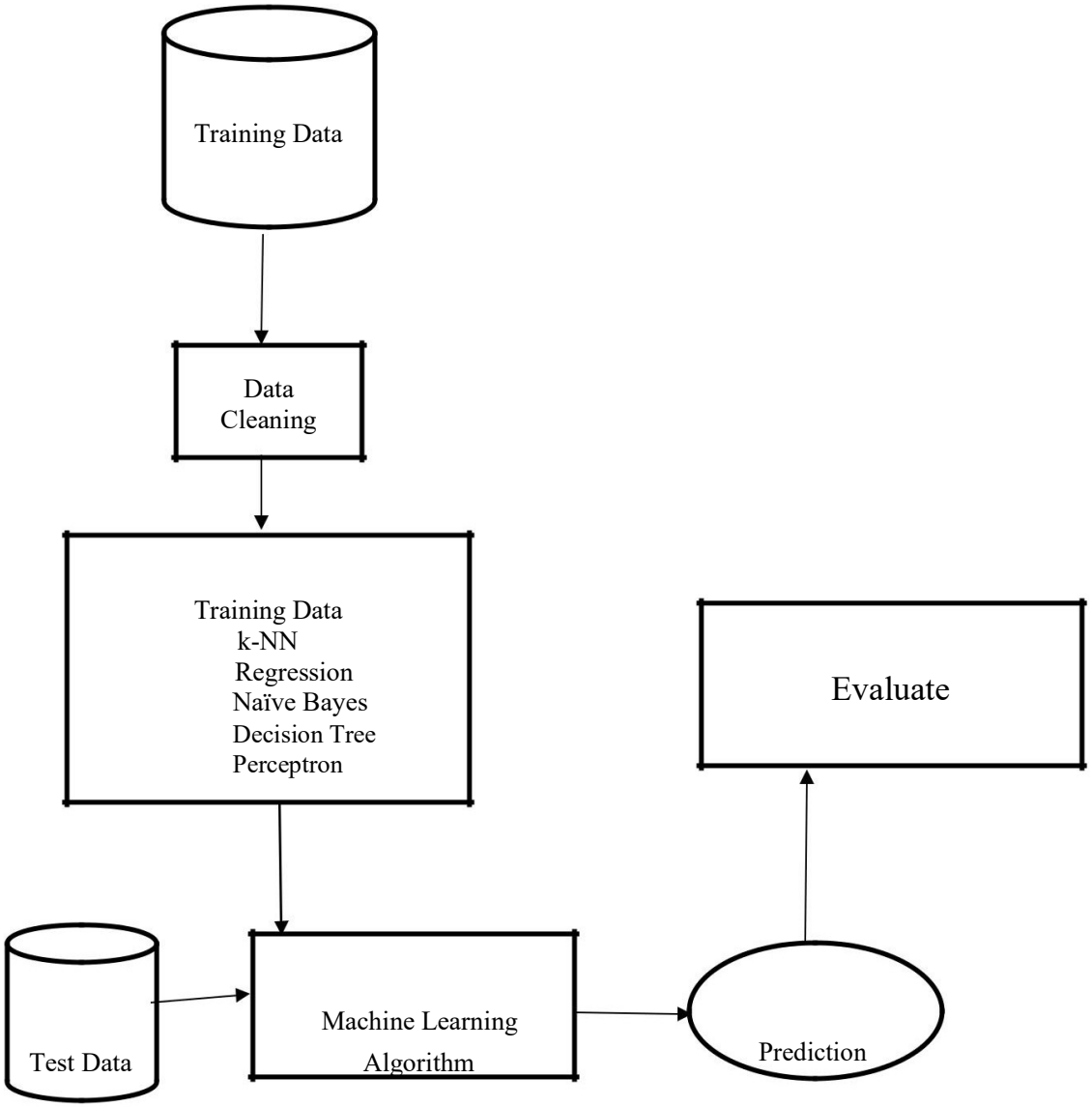
A small loan application dataset taken from Kaggle (containing training and test data).

Contents of dataset:

- product id
- product_name
- product_category_tree
- retail_price
- discounted_price
- is_FK_Advantage_product (whether or not Flipkart advantage product)
- product_rating
- brand
- product_specificationsProperty Area

Application Work Flow

(This section displays the flow of information in the application)



Future Scope of Improvements

1. we can further use a graphical user interface to help easy handling and analysis of data.
2. The same model can be used to determine similar kind of predictions, for e.g. Analysis of Flipkart Data. And thus we can make this a whole system of its own.
3. The data set contained a lot of missing value. Moreover, We saw that most of the feature attributes did not contribute much to the prediction. If we had a proper dataset with better attributes, we could make the predictions more accurate.

Code

(This section contains the source code of the application)

Flipkart E-commerce Data of various products

This project is made by a team of Batch 3

Kaushik Ghosh

Mukund Agarwal

Salini

Rajdeep Koner

Context

This is a pre-crawled dataset, taken as subset of a bigger dataset (more than 5.8 million products) that was created by extracting data from Flipkart.com, a leading Indian eCommerce store.

Content

This dataset has following fields:

product_url product_name product_category_tree pid retail_price discounted_price image is_FK_Advantage_product description product_rating overall_rating brand product_specifications

Acknowledgements

This dataset was created by PromptCloud's in-house web-crawling service.

Inspiration

Analyses of the pricing, product specification and brand can be performed.

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('flipkart.csv')
```

In [3]:
```
df.head()
```

Out[3]:

:ounted_price	image	is_FK_Advantage_product	description	pro
379.0	["http://img5a.flixcart.com/image/short/u/4/a/...	False	Key Features of Alisha Solid Women's Cycling S...	
22646.0	["http://img6a.flixcart.com/image/sofa-bed/j/f...	False	FabHomeDecor Fabric Double Sofa Bed (Finish Co...	
499.0	["http://img5a.flixcart.com/image/shoe/7/z/z/r...	False	Key Features of AW Bellies Sandals Wedges Heel...	
267.0	["http://img5a.flixcart.com/image/short/6/2/h/...	False	Key Features of Alisha Solid Women's Cycling S...	
210.0	["http://img5a.flixcart.com/image/pet-shampoo/...	False	Specifications of Sicons All Purpose Arnica Do...	

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 15 columns):
uniq_id                   20000 non-null object
crawl_timestamp           20000 non-null object
product_url               20000 non-null object
product_name              20000 non-null object
product_category_tree     20000 non-null object
pid                       20000 non-null object
retail_price              19922 non-null float64
discounted_price          19922 non-null float64
image                     19997 non-null object
is_FK_Advantage_product   20000 non-null bool
description               19998 non-null object
product_rating            20000 non-null object
overall_rating            20000 non-null object
brand                     14136 non-null object
product_specifications    19986 non-null object
dtypes: bool(1), float64(2), object(12)
memory usage: 1.2+ MB
```

Around 6000 missing brand names

Checking the relation of existing brand names with pid since patterns showed that the first five letters are common with the same brand names

In [5]:
```
dfnew = df.copy()
```

In [6]:
```
dfnew.drop(['overall_rating'], axis = 1, inplace = True)
```

In [7]:
```
#found the pattern the, first 7 letters correspond to brandname
def firstfive(frame):
    return frame[0:7]
```

In [8]:
```
dfnew['newid'] = dfnew['pid'].apply(firstfive)
```

In [9]:
```
#dfnew.drop(['pid'],axis=1,inplace=True)
dfnew
```

Out[9]:

	uniq_id	crawl_timestamp	product_url	
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	
1	7f7036a6d550aaa89d34c77bd39a5e48	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	F
2	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	
3	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	

In [10]:
```
dfnew['newid'].value_counts()
```

Out[10]:
```
CRTECN2    464
ACCE9Y6     97
TSHE993     87
RNGE7M9     81
RNGEDAK     76
MUGEAGG     72
STIE7KF     65
ACCE9DJ     62
RNGE7GV     61
CRNEE84     58
STIE9F5     55
MUGEACY     54
STIEYZ5     52
RNGEDAP     52
ACCE6SJ     43
NKCDU4R     43
ACCE6GF     42
ACCDVHJ     40
BRAE3TS     40
CRNEEAB     37
BLAEAWA     36
NKCE9PF     36
MUGEBFG     35
PCSEC86     34
SNDEAN4     33
ACCEAZC     33
BRAEBBM     32
ACCE9DK     32
ACCDUV7     31
BRAEGJF     31
            ...
CRNEYW6      1
RUGEJ3N      1
TROEJYT      1
PBXEBWY      1
JCKE2AF      1
SPMDHEZ      1
SHTDVZK      1
KLSEJ3V      1
TSHE62Q      1
SHOEG52      1
TPMEE6Z      1
TSHE7FU      1
NKCDXH8      1
BRAE8NW      1
KRTEE8S      1
ACCEJR2      1
FSEEGBR      1
KLSEGUA      1
BBOEGGV      1
USGEDST      1
SHIEB59      1
JWSEFBQ      1
PTGECWD      1
BRAEA6Z      1
```

```
CAGE9B7      1
LBXEHH5      1
TSHE4CD      1
SHIEB65      1
SWTEA38      1
NKCEGZZ      1
Name: newid, Length: 9253, dtype: int64
```

In [11]:
```
dfnewnodup = dfnew.drop_duplicates()
```

In [12]:
```
dfnewnodup
```

14	83c53f8948f508f51d2249b489ca8e7d	2016-03-25 22:59:23 +0000	http://www.flipkart.com/freelance-vacuum-bottl...	V
15	d95b0456a0350bc42f2393c6e84b0f09	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	
16	849ab05698081a374215d0b7d18047d0	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	F
17	c275ee5ac19f774a3ef7da71b40aabd8	2016-03-25 22:59:23 +0000	http://www.flipkart.com/style-foot-bellies/p/i...	
18	4f3511c33a6869b1d5102cd97818ef00	2016-03-25 22:59:23 +0000	http://www.flipkart.com/carrel-printed-women-s...	

In [13]:
```
df.head()
```

Out[13]:

	uniq_id	crawl_timestamp	product_url	produc
0	c2d766ca982eca8304150849735ffef9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisl W Cyclinç
1	7f7036a6d550aaa89d34c77bd39a5e48	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	FabHom Fabric S
2	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	AW
3	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisl W Cyclinç
4	bc940ea42ee6bef5ac7cea3fb5cfbee7	2016-03-25 22:59:23 +0000	http://www.flipkart.com/sicons-all-purpose-arn...	Si Purpose Dog Sl

There is no such correlation between the pid and brand so we have to discard this.

In [15]:
```
df.tail()
```

Out[15]:

	uniq_id	crawl_timestamp	product_url	produ
19995	7179d2f6c4ad50a17d014ca1d2815156	2015-12-01 10:15:43 +0000	http://www.flipkart.com/walldesign-small-vinyl...	Wa Sn
19996	71ac419198359d37b8fe5e3fffdfee09	2015-12-01 10:15:43 +0000	http://www.flipkart.com/wallmantra-large-vinyl...	Wa La Sticker
19997	93e9d343837400ce0d7980874ece471c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	Elite C Mediur
19998	669e79b8fa5d9ae020841c0c97d5e935	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	Elite C Mediur
19999	cb4fa87a874f715fff567f7b7b3be79c	2015-12-01 10:15:43 +0000	http://www.flipkart.com/elite-collection-mediu...	Elite C Mediur

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 15 columns):
uniq_id                    20000 non-null object
crawl_timestamp            20000 non-null object
product_url                20000 non-null object
product_name               20000 non-null object
product_category_tree      20000 non-null object
pid                        20000 non-null object
retail_price               19922 non-null float64
discounted_price           19922 non-null float64
image                      19997 non-null object
is_FK_Advantage_product    20000 non-null bool
description                19998 non-null object
product_rating             20000 non-null object
overall_rating             20000 non-null object
brand                      14136 non-null object
product_specifications     19986 non-null object
dtypes: bool(1), float64(2), object(12)
memory usage: 1.2+ MB
```

In [17]:
```
dfnew = df.copy()
```

In [18]:
```
dfnew.drop_duplicates()
```

5	c2a17313954882c1dba461863e98adf2	2016-03-25 22:59:23 +0000	http://www.flipkart.com/eternal-gandhi-super-s...	
6	ce5a6818f7707e2cb61fdcdbba61f5ad	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	
7	8542703ca9e6ebdf6d742638dfb1f2ca	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	F
8	29c8d290caa451f97b1c32df64477a2c	2016-03-25 22:59:23 +0000	http://www.flipkart.com/dilli-bazaaar-bellies-...	

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 15 columns):
uniq_id                   20000 non-null object
crawl_timestamp           20000 non-null object
product_url               20000 non-null object
product_name              20000 non-null object
product_category_tree     20000 non-null object
pid                       20000 non-null object
retail_price              19922 non-null float64
discounted_price          19922 non-null float64
image                     19997 non-null object
is_FK_Advantage_product   20000 non-null bool
description               19998 non-null object
product_rating            20000 non-null object
overall_rating            20000 non-null object
brand                     14136 non-null object
product_specifications    19986 non-null object
dtypes: bool(1), float64(2), object(12)
memory usage: 1.2+ MB
```

In [20]:
```
dfnew = dfnew.drop_duplicates()
```

In [21]:
```
dfnew.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 19999
Data columns (total 15 columns):
uniq_id                    20000 non-null object
crawl_timestamp            20000 non-null object
product_url                20000 non-null object
product_name               20000 non-null object
product_category_tree      20000 non-null object
pid                        20000 non-null object
retail_price               19922 non-null float64
discounted_price           19922 non-null float64
image                      19997 non-null object
is_FK_Advantage_product    20000 non-null bool
description                19998 non-null object
product_rating             20000 non-null object
overall_rating             20000 non-null object
brand                      14136 non-null object
product_specifications     19986 non-null object
dtypes: bool(1), float64(2), object(12)
memory usage: 1.4+ MB
```

In [22]:
```
dfnew.drop(['crawl_timestamp','uniq_id','image'],axis=1,inplace=True)
```

In [23]:
```
dfnew.head(2)
dfnew.drop(['pid'],axis=1,inplace=True)
```

'pid','crawl_timestamp','uniq_id','image' dropped since they are irrelevant to our current context

Analysing its relationship with description, product_url, product name

In [24]:
```
dfnew['product_url'][0]
dfnew['description'][20]
```

Out[24]:
```
'Specifications of Sicons Conditioning Conditoner Dog Shampoo (200 ml) General
Pet Type Dog Brand Sicons Quantity 200 ml Model Number SH.DF-02 Type Conditioni
ng Fragrance Conditoner Form Factor Gel In the Box Sales Package Shampoo Sicons
Dog Fashion Conditioner Aloe Rinse'
```

In [25]:
```
dfnew.drop(['description'],axis=1,inplace=True)
dfnew.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 19999
Data columns (total 10 columns):
product_url                20000 non-null object
product_name               20000 non-null object
product_category_tree      20000 non-null object
retail_price               19922 non-null float64
discounted_price           19922 non-null float64
is_FK_Advantage_product    20000 non-null bool
product_rating             20000 non-null object
overall_rating             20000 non-null object
brand                      14136 non-null object
product_specifications     19986 non-null object
dtypes: bool(1), float64(2), object(7)
memory usage: 1.5+ MB
```

In [26]:
```
#dfnew.info()
dfnew['product_url'][18000]
#dfnew['product_name'][0]
```

Out[26]:
```
'http://www.flipkart.com/wildcraft-bonk-25-l-backpack/p/itme9dkby8j6cryh?pid=BK
PE9DKAA8TSKFKW'
```

In [27]:
```
dfnew['product_name'][18000]
```

Out[27]:
```
'Wildcraft Bonk 25 L Backpack'
```

In [28]:
```
dfnew1 = dfnew[dfnew['brand']==pd.np.nan]
```

In [29]:
```
dfnew1
```

Out[29]:

	product_url	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag

In [30]:
```
dfnew.head()
```

Out[30]:

	product_url	product_name	product_category_tree	retail_price	discounted
0	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	
1	http://www.flipkart.com/fabhomedecor-fabric-do...	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	2
2	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	
3	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	
4	http://www.flipkart.com/sicons-all-purpose-arn...	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	

product_url and pid are unique to any product, so dropping these would help in reducing the duplicates.

Approach one, for the brand names whose word length is 2

In [31]:
```
def stripbrand(x):
    l = x.split()
    return l[0] + l[1]
dfnew.drop(['product_url'],axis=1,inplace=True)
```

In [32]:
```
#dfnew['brand_refined'] = dfnew[dfnew['product_name']].apply(stripbrand,axis=1)
#type(dfnew['product_name'])
#dfnew['product_name']
dfnew.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 19999
Data columns (total 9 columns):
product_name               20000 non-null object
product_category_tree      20000 non-null object
retail_price               19922 non-null float64
discounted_price           19922 non-null float64
is_FK_Advantage_product    20000 non-null bool
product_rating             20000 non-null object
overall_rating             20000 non-null object
brand                      14136 non-null object
product_specifications     19986 non-null object
dtypes: bool(1), float64(2), object(6)
memory usage: 1.4+ MB
```

In [33]:
```
dfnew.drop_duplicates(inplace=True)
```

In [34]:
```
dfnew.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19449 entries, 0 to 19999
Data columns (total 9 columns):
product_name               19449 non-null object
product_category_tree      19449 non-null object
retail_price               19371 non-null float64
discounted_price           19371 non-null float64
is_FK_Advantage_product    19449 non-null bool
product_rating             19449 non-null object
overall_rating             19449 non-null object
brand                      13791 non-null object
product_specifications     19435 non-null object
dtypes: bool(1), float64(2), object(6)
memory usage: 930.7+ KB
```

551 duplicate rows dropped

Approach 2. Analising its relationship with product_category_tree

In [35]:
```
x = dfnew['product_category_tree'][100]
```

In [36]:
```
x
```

Out[36]:
```
'["Watches >> Wrist Watches >> Rorlig Wrist Watches"]'
```

In [37]:
```
x = x.strip('[')
```

In [38]:
```
x = x.strip(']')
```

In [39]:
```
x = x.replace('"','')
m = x.split(">>")
```

In [40]:
```
m = x.split(">>")
def subtract(x,y):
    if y in x:
        x = x.replace(y,"")
    return x
subtract('kaushik','ik')
```

Out[40]:
```
'kaush'
```

In [41]:
```
m
```

Out[41]:
```
['Watches ', ' Wrist Watches ', ' Rorlig Wrist Watches']
```

In [42]:
```
for i in range(len(m)):
    m[i] = m[i].replace(' ','')
```

In [43]:
```
m
```

Out[43]:
```
['Watches', 'WristWatches', 'RorligWristWatches']
```

In [44]:
```
for i in range(len(m)-2,0,-1):
    if m[i] in m[-1]:
        m[-1] = m[-1].replace(m[i],'')
```

In [45]:
```
m
```

Out[45]:
```
['Watches', 'WristWatches', 'Rorlig']
```

In [46]:
```
m[-1]
```

Out[46]:
```
'Rorlig'
```

In [47]:
```
def brandprediction(x):
    x = x.strip('[')
    x = x.strip(']')
    x = x.replace('"','')
    m = x.split(">>")
    for i in range(len(m)):
        m[i] = m[i].replace(' ','')
    for i in range(len(m)-2,0,-1):
        if m[i] in m[-1]:
            m[-1] = m[-1].replace(m[i],'')
    return m[-1]
```

In [48]:
```
brandprediction(dfnew['product_category_tree'][876])
```

Out[48]:
```
'LifebyShoppersStop'
```

In [49]:
```
dfnew['brand_predicted'] = dfnew['product_category_tree'].apply(brandprediction)
```

In [50]:
```
dfnew['brand_predicted'].value_counts()
dfnew['brand'].value_counts()
```

Out[50]:
```
Allure Auto                    469
Regular                        311
Voylla                         299
Slim                           279
TheLostPuppy                   229
Karatcraft                     211
Black                          164
DailyObjects                   144
White                          143
Speedwav                       141
Radiant Bay                    132
Red                            102
Enthopia                       101
BlueStone                       99
Pink                            98
HomeeHub                        95
Wallmantra                      80
Purple                          79
AdroitZ                         74
Blue                            71
```

Prediction almost fitting nicely

In [51]:
```
dfnew.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19449 entries, 0 to 19999
Data columns (total 10 columns):
product_name               19449 non-null object
product_category_tree      19449 non-null object
retail_price               19371 non-null float64
discounted_price           19371 non-null float64
is_FK_Advantage_product    19449 non-null bool
product_rating             19449 non-null object
overall_rating             19449 non-null object
brand                      13791 non-null object
product_specifications     19435 non-null object
brand_predicted            19449 non-null object
dtypes: bool(1), float64(2), object(7)
memory usage: 1.5+ MB
```

In [52]:
```
dfnew['brand'].nunique()
```

Out[52]: 3499

In [53]:
```
dfnew['brand_predicted'].nunique()
```

Out[53]: 5985

In [54]:
```
brandprediction(dfnew['product_category_tree'][154])
```

Out[54]: 'Boots'

some cases where the product category tree had some anomalies like not having the brand name at last

In [55]:
```
dfnew['product_category_tree'][154]
```

Out[55]:
```
'["Footwear >> Women\'s Footwear >> Casual Shoes >> Boots"]'
```

In [56]:
```
dfnew['brand'][154]
```

Out[56]:
```
nan
```

In [57]:
```
df['product_url'][154]
```

Out[57]:
```
'http://www.flipkart.com/shuz-touch-boots/p/itmecxpm57rcspy4?pid=SHOECXPMAQXZ3Q
PH'
```

In [58]:
```
df['product_name'][154]
```

Out[58]:
```
'Shuz Touch Boots'
```

In [59]:
```
df['product_name'][155]
```

Out[59]:
```
'Kielz Ladies Boots'
```

In [60]:
```
dfnew['brand'][155]
```

Out[60]:
```
nan
```

In [61]:
```
type(dfnew['brand'][155])
```

Out[61]:
```
float
```

The 'nan' value was a float thus not visible by putting np.nan as the criteria for seperating dataframes

In [62]:
```
type(dfnew['brand'][154])
```

Out[62]:
```
float
```

In [63]:
```
str(dfnew['brand'][155])
```

Out[63]:
```
'nan'
```

In [64]:
```
def convertstring(x):
    return str(x)
```

In [65]:
```
dfnew['brand_refined_into_string'] = df['brand'].apply(convertstring)
```

Approach 3.0 separating dataframes and working on just the one having nan

In [90]:
```
dfbrandnan = dfnew[dfnew['brand_refined_into_string'] == 'nan']
dff = dfnew[dfnew['brand_refined_into_string'] != 'nan']
```

In [91]:
```
dfbrandnan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5658 entries, 25 to 19962
Data columns (total 11 columns):
product_name                 5658 non-null object
product_category_tree        5658 non-null object
retail_price                 5645 non-null float64
discounted_price             5645 non-null float64
is_FK_Advantage_product      5658 non-null bool
product_rating               5658 non-null object
overall_rating               5658 non-null object
brand                        0 non-null object
product_specifications       5658 non-null object
brand_predicted              5658 non-null object
brand_refined_into_string    5658 non-null object
dtypes: bool(1), float64(2), object(8)
memory usage: 314.9+ KB
```

In [92]:
```
dfbrandnan.head()
```

Out[92]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
25	Glus Wedding Lingerie Set	["Clothing >> Women's Clothing >> Lingerie, Sl...	1299.0	699.0	False
26	Veelys Shiny White Quad Roller Skates - Size 4...	["Sports & Fitness >> Other Sports >> Skating ...	3199.0	2499.0	False
27	Bulaky vanity case Jewellery Vanity Case	["Beauty and Personal Care >> Makeup >> Vanity...	499.0	390.0	False
28	FDT Women's Leggings	["Clothing >> Women's Clothing >> Fusion Wear ...	699.0	309.0	False
29	Madcaps C38GR30 Men's Cargos	["Clothing >> Men's Clothing >> Cargos, Shorts...	2199.0	1699.0	False

Approach 3.1 reapplying all the logics one by one to get the best possible brand prediction

In [93]:
```
dfbrandnan['brand'] = dfbrandnan['product_name'].apply(lambda x: x.split()[0])
```

```
C:\Users\h_asp\Anaconda3\lib\site-packages\ipykernel\__main__.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  if __name__ == '__main__':
```

In [94]:
```
def stringcut(x):
    m = x.split()
    y = m[0]+m[1]
    return y
stringcut
```

Out[94]:
```
<function __main__.stringcut>
```

In [95]:
```
dfbrandnan1 = dfbrandnan.copy()
```

In [5]:
```
#dfbrandnan1['brand_name2'] = dfbrandnan1['product_name'].apply(stringcut)
```

In [97]:
```
dfbrandnan1.head()
```

Out[97]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
25	Glus Wedding Lingerie Set	["Clothing >> Women's Clothing >> Lingerie, Sl...	1299.0	699.0	False
26	Veelys Shiny White Quad Roller Skates - Size 4...	["Sports & Fitness >> Other Sports >> Skating ...	3199.0	2499.0	False
27	Bulaky vanity case Jewellery Vanity Case	["Beauty and Personal Care >> Makeup >> Vanity...	499.0	390.0	False
28	FDT Women's Leggings	["Clothing >> Women's Clothing >> Fusion Wear ...	699.0	309.0	False
29	Madcaps C38GR30 Men's Cargos	["Clothing >> Men's Clothing >> Cargos, Shorts...	2199.0	1699.0	False

In [98]:
```
dfbrandnan.head()
```

Out[98]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
25	Glus Wedding Lingerie Set	["Clothing >> Women's Clothing >> Lingerie, Sl...	1299.0	699.0	False
26	Veelys Shiny White Quad Roller Skates - Size 4...	["Sports & Fitness >> Other Sports >> Skating ...	3199.0	2499.0	False
27	Bulaky vanity case Jewellery Vanity Case	["Beauty and Personal Care >> Makeup >> Vanity...	499.0	390.0	False
28	FDT Women's Leggings	["Clothing >> Women's Clothing >> Fusion Wear ...	699.0	309.0	False
29	Madcaps C38GR30 Men's Cargos	["Clothing >> Men's Clothing >> Cargos, Shorts...	2199.0	1699.0	False

Approach 3.2 checking for common nouns from product_category_tree and subtracting it from product_name

In [99]:
```
dfbrandnan['product_name'][25]
```

Out[99]:
```
'Glus Wedding Lingerie Set'
```

In [100]:
```
dfbrandnan['product_category_tree'][25]
```

Out[100]:
```
'["Clothing >> Women\'s Clothing >> Lingerie, Sleep & Swimwear >> Lingerie Sets
>> Glus Lingerie Sets"]'
```

In [101]:
```
def commons(x):
    x = x.strip('[')
    x = x.strip(']')
    x = x.replace('"','')
    #x = x.replace('\'',' ')
    m = x.split(">>")

    return m
```

In [102]:
```
def gettingbrand(x):
    n = commons(x)
    l = n[0:-1]
    xl = []
    for i in l:
        m = i.split()
        for j in m:
            xl.append(j)
    y = dfbrandnan['product_name'][dfbrandnan]
    ylist = y.split()
    brand = ''
    for i in range(len(ylist)):
        if y[i] not in xl:
            brand = brand + y[i]
    return brand
```

In [103]:
```
brand_list = []
for row_index,row in dfbrandnan.iterrows():
    x = row['product_category_tree']
    n = commons(x)
    y = dfbrandnan['product_name'][row_index]
    ylist = y.split()
    for j in range(len(n)):
        if n[j] in ylist:
            l = n[0:-1]
        else:
            l = n
    xl = []
    for i in l:
        m = i.split()
        for j in m:
            xl.append(j)
    #y = dfbrandnan['product_name'][row_index]
    #ylist = y.split()
    #print(ylist)
    brand = ''
    #brand_list = []
    for i in range(len(ylist)):
        #print(ylist[i])
        if ylist[i] not in xl:
            brand = brand + ' ' + ylist[i]
    brand = brand.strip()
    brand_list.append(brand)
```

In [104]:
```
brand_list
```

Out[104]:
```
['Wedding Set',
 'Shiny White Quad Roller - Size 4.5 UK',
 'vanity case Jewellery Case',
 '',
 'C38GR30',
 'CO6394A1 Analog Watch - For Men, Boys',
 'TEN TEN Black Knee Length',
 'G 729 S-BK Analog Watch - For Men, Boys',
 'Carlton',
 'R8851116001 Analog Watch - For Boys',
 '8503B-1RED Cold Light Digital Watch - For Boys, Girls',
 'WM64 Elegance Analog Watch - For Men, Boys',
 'Quechua Arpenaz Novadry',
 'COLAT_MW20 Sheen Analog Watch - For Men, Women, Boys, Girls',
 'Steppings Trendy',
 'RW38 Analog Watch - For Boys',
 'RR-028 Expedition Analog Watch - For Men, Boys',
 'Catwalk',
 'Magnum Lifestyle',
```

In [105]:
```
dfbrandnan2 = dfbrandnan.copy()
```

In [106]:
```
dfbrandnan2['final_brand'] = brand_list
```

In [107]:
```
dfbrandnan2['final_brand'].value_counts()
```

Out[107]:
```
Printed Round Neck T-Shirt                          242
                                                    129
A-line Dress                                        120
Solid Polo Neck T-Shirt                             112
Solid Round Neck T-Shirt                            110
Graphic Print Round Neck T-Shirt                    100
Striped Polo Neck T-Shirt                            73
Casual Printed Kurti                                 70
Casual Sleeveless Solid Top                          69
Shift Dress                                          68
Casual Short Sleeve Solid Top                        56
Solid V-neck T-Shirt                                 51
Combo Set                                            46
Full Sleeve Solid Sweatshirt                         46
Solid Casual Shirt                                   39
Gathered Dress                                       39
Maxi Dress                                           39
Printed V-neck T-Shirt                               35
Necktie Combo                                        34
```

Approach 3.2 resulted in anomalies

In [108]:
```
dfbrandnan2.head(5000)
```

Out[108]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_p
25	Glus Wedding Lingerie Set	["Clothing >> Women's Clothing >> Lingerie, Sl...	1299.0	699.0	
26	Veelys Shiny White Quad Roller Skates - Size 4...	["Sports & Fitness >> Other Sports >> Skating ...	3199.0	2499.0	
27	Bulaky vanity case Jewellery Vanity Case	["Beauty and Personal Care >> Makeup >> Vanity...	499.0	390.0	
28	FDT Women's Leggings	["Clothing >> Women's Clothing >> Fusion Wear ...	699.0	309.0	
29	Madcaps C38GR30 Men's Cargos	["Clothing >> Men's Clothing >> Cargos, Shorts...	2199.0	1699.0	

In [109]:
```
dfbrandnan2['brand'].nunique()
```

Out[109]: 1546

In [110]:
```
dfbrandnan2['brand_predicted'].nunique()
```

Out[110]: 1466

This shows that 'brand' has more unique brands, but 'brand_predicted' has better results

Approach 3.3 searching for common nouns in brand_predicted and replacing it by brand[row_index]

In [116]:
```
dfbrandnan3 = dfbrandnan2.copy()
brand_name_final = []
```

In [118]:
```
dfbrandnan2.reset_index()
brande_final = []
```

In [119]:
```
for row_index,row in dfbrandnan2.iterrows():
    if row['brand_predicted'] == 'Boots' or row['brand_predicted'] == 'Heels' or
        brande_final.append(row['brand'])
    else:
        brande_final.append(row['brand_predicted'])
```

In [120]:
```
brande_final
```

Out[120]:
```
['Glus',
 'Veelys',
 'Bulaky',
 'FDT',
 'Madcaps',
 'CobraParis',
 'TEN',
 'AriesGold',
 'Carlton',
 'MaseratiTime',
 'Vizion',
 'Camerii',
 'Quechua',
 'Colat',
 'Steppings',
 'Rochees',
 'Rorlig',
 'Catwalk',
 'Magnum',
 'ISTAR'
```

These are the analysis that helped framing the brand_prediction function

In [121]:
```
#dfbrandnan3['final_brand_hel']
```

In [123]:
```
#dfbrandnan2['brand_name_final'] = dfbrandnan2[dfbrandnan2['']
```

In [124]:
```
dfbrandnan2['product_category_tree'][19962]
```

Out[124]:
```
'["Footwear >> Women\'s Footwear >> Heels"]'
```

In [125]:
```
dfbrandnan2['final_brand'][19962]
```

Out[125]:
```
'Stylistry Women'
```

In [126]:
```
v = dfbrandnan2['product_name'][19962]
```

In [127]:
```
n = commons('["Footwear >> Women\'s Footwear >> Heels"]')
```

In [128]:
```
l = n
xl = []
for i in l:
    m = i.split()
    for j in m:
        xl.append(j)
```

In [129]:
```
xl
```

Out[129]:
```
['Footwear', "Women's", 'Footwear', 'Heels']
```

In [130]:
```
y = v
ylist = y.split()
print(ylist)
brand = ''
brand_list = []
for i in range(len(ylist)):
    print(ylist[i])
    if ylist[i] not in xl:
        brand = brand + ' ' + ylist[i]
brand = brand.strip()
brand_list.append(brand)
```

```
['Stylistry', 'Women', 'Heels']
Stylistry
Women
Heels
```

In [131]:
```
brand_list
```

Out[131]:
```
['Stylistry Women']
```

In [132]:
```
n = commons(dfbrandnan['product_category_tree'][25])
```

In [134]:
```
#b = gettingbrand(dfbrandnan['product_category_tree'][25])
```

In [ ]:
```

```

In [135]:
```
l = n[0:-1]
```

In [136]:
```
l
```

Out[136]:
```
['Clothing ',
 " Women's Clothing ",
 ' Lingerie, Sleep & Swimwear ',
 ' Lingerie Sets ']
```

In [137]:
```
xl = []
for i in l:
    m = i.split()
    for j in m:
        xl.append(j)
```

In [138]:
```
i.split()
```

Out[138]:
```
['Lingerie', 'Sets']
```

In [139]:
```
xl
```

Out[139]:
```
['Clothing',
 "Women's",
 'Clothing',
 'Lingerie,',
 'Sleep',
 '&',
 'Swimwear',
 'Lingerie',
 'Sets']
```

In [140]:
```
y = dfbrandnan['product_name'][25]
ylist = y.split()
```

In [141]:
```
brand = ''
for i in range(len(ylist)):
    if y[i] not in xl:
        brand = brand + y[i]
```

In [142]:
```
ylist
```

Out[142]:
```
['Glus', 'Wedding', 'Lingerie', 'Set']
```

In [143]:
```
brand
```

Out[143]:
```
'Glus'
```

After analysis , the final brand getting algorithm

In [144]:
```
dfbrandnan2['Brand_names'] = brande_final
```

In [145]:
```
dfcleaned = dfbrandnan2.copy()
```

In [146]:
```
dfcleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5658 entries, 25 to 19962
Data columns (total 13 columns):
product_name                5658 non-null object
product_category_tree       5658 non-null object
retail_price                5645 non-null float64
discounted_price            5645 non-null float64
is_FK_Advantage_product     5658 non-null bool
product_rating              5658 non-null object
overall_rating              5658 non-null object
brand                       5658 non-null object
product_specifications      5658 non-null object
brand_predicted             5658 non-null object
brand_refined_into_string   5658 non-null object
final_brand                 5658 non-null object
Brand_names                 5658 non-null object
dtypes: bool(1), float64(2), object(10)
memory usage: 359.2+ KB
```

In [158]:
```
dfcleaned.drop(['brand','brand_predicted','brand_refined_into_string','final_bran
```

In [159]:
```
dfcleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5658 entries, 25 to 19962
Data columns (total 9 columns):
product_name               5658 non-null object
product_category_tree      5658 non-null object
retail_price               5645 non-null float64
discounted_price           5645 non-null float64
is_FK_Advantage_product    5658 non-null bool
product_rating             5658 non-null object
overall_rating             5658 non-null object
product_specifications     5658 non-null object
Brand_names                5658 non-null object
dtypes: bool(1), float64(2), object(6)
memory usage: 270.7+ KB
```

In [160]:
```
dff.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13791 entries, 0 to 19999
Data columns (total 11 columns):
product_name                13791 non-null object
product_category_tree       13791 non-null object
retail_price                13726 non-null float64
discounted_price            13726 non-null float64
is_FK_Advantage_product     13791 non-null bool
product_rating              13791 non-null object
overall_rating              13791 non-null object
brand                       13791 non-null object
product_specifications      13777 non-null object
brand_predicted             13791 non-null object
brand_refined_into_string   13791 non-null object
dtypes: bool(1), float64(2), object(8)
memory usage: 767.7+ KB
```

In [161]:
```
b_already_cleaned = dff['brand']
dff1 = dff.copy()
```

In [162]:
```
dff1['Brand_names'] = b_already_cleaned
```

In [165]:
```
dff1.info()
#dff1.drop(['brand','brand_predicted','brand_refined_into_string'],axis=1,inplace=
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13791 entries, 0 to 19999
Data columns (total 9 columns):
product_name              13791 non-null object
product_category_tree     13791 non-null object
retail_price              13726 non-null float64
discounted_price          13726 non-null float64
is_FK_Advantage_product   13791 non-null bool
product_rating            13791 non-null object
overall_rating            13791 non-null object
product_specifications    13777 non-null object
Brand_names               13791 non-null object
dtypes: bool(1), float64(2), object(6)
memory usage: 659.9+ KB
```

In [166]:
```
dfcleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5658 entries, 25 to 19962
Data columns (total 9 columns):
product_name               5658 non-null object
product_category_tree      5658 non-null object
retail_price               5645 non-null float64
discounted_price           5645 non-null float64
is_FK_Advantage_product    5658 non-null bool
product_rating             5658 non-null object
overall_rating             5658 non-null object
product_specifications     5658 non-null object
Brand_names                5658 non-null object
dtypes: bool(1), float64(2), object(6)
memory usage: 270.7+ KB
```

In [279]:
```
dff1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13791 entries, 0 to 19999
Data columns (total 9 columns):
product_name               13791 non-null object
product_category_tree      13791 non-null object
retail_price               13726 non-null float64
discounted_price           13726 non-null float64
is_FK_Advantage_product    13791 non-null bool
product_rating             13791 non-null object
overall_rating             13791 non-null object
product_specifications     13777 non-null object
Brand_names                13791 non-null object
dtypes: bool(1), float64(2), object(6)
memory usage: 983.1+ KB
```

In [286]:
```
dl = dff1.merge(dfcleaned)
```

In [167]:
```
frames = [dff1,dfcleaned]
```

In [168]:
```
cleanedflipkart = pd.concat(frames)
```

In [169]:
```
cleanedflipkart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19449 entries, 0 to 19962
Data columns (total 9 columns):
product_name               19449 non-null object
product_category_tree      19449 non-null object
retail_price               19371 non-null float64
discounted_price           19371 non-null float64
is_FK_Advantage_product    19449 non-null bool
product_rating             19449 non-null object
overall_rating             19449 non-null object
product_specifications     19435 non-null object
Brand_names                19449 non-null object
dtypes: bool(1), float64(2), object(6)
memory usage: 930.7+ KB
```

In [171]:
```
cleanedflipkart.to_csv('cleanedflipkart.csv',encoding='utf-8')
```

The csv file now contains brands for which we had 'nan', thus reducing the level of anomaly in the dataset to some extent

Next See the product_specifications cleaning

In [ ]:

Globsyn Summer School 2018, Machine Learning Using Python, a project on analysis of flipkart data, by Kaushik, Mukund, Rajdeep and Salini

In [1]:
```
import pandas as pd
import numpy as np
newdf = pd.read_csv('cleanedflipkart.csv')
```

In [3]:
```
newdf.head(2)
```

Out[3]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	

In [9]:
```
newdf.info()
#newdf.drop(['Unnamed: 0'],axis=0,inplace=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19449 entries, 0 to 19448
Data columns (total 10 columns):
Unnamed: 0                 19449 non-null int64
product_name               19449 non-null object
product_category_tree      19449 non-null object
retail_price               19371 non-null float64
discounted_price           19371 non-null float64
is_FK_Advantage_product    19449 non-null bool
product_rating             19449 non-null object
overall_rating             19449 non-null object
product_specifications     19435 non-null object
Brand_names                19449 non-null object
dtypes: bool(1), float64(2), int64(1), object(6)
memory usage: 930.7+ KB
```

In [14]:
```
newdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19449 entries, 0 to 19448
Data columns (total 10 columns):
Unnamed: 0                 19449 non-null int64
product_name               19449 non-null object
product_category_tree      19449 non-null object
retail_price               19371 non-null float64
discounted_price           19371 non-null float64
is_FK_Advantage_product    19449 non-null bool
product_rating             19449 non-null object
overall_rating             19449 non-null object
product_specifications     19435 non-null object
Brand_names                19449 non-null object
dtypes: bool(1), float64(2), int64(1), object(6)
memory usage: 930.7+ KB
```

14 missing product specifications to be handled

product_category from product_category_tree, taking just the parent category

In [16]:
```
def productcategorypred(x):
    x = x.strip("'")
    x = x.strip("[]")
    x = x.strip(" " " ")
    x = x.split(">>")
    return(x[0][1:])

newdf["product_category"] = newdf["product_category_tree"].apply(lambda x: produc
```

In [17]:
```
newdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19449 entries, 0 to 19448
Data columns (total 11 columns):
Unnamed: 0                 19449 non-null int64
product_name               19449 non-null object
product_category_tree      19449 non-null object
retail_price               19371 non-null float64
discounted_price           19371 non-null float64
is_FK_Advantage_product    19449 non-null bool
product_rating             19449 non-null object
overall_rating             19449 non-null object
product_specifications     19435 non-null object
Brand_names                19449 non-null object
product_category           19449 non-null object
dtypes: bool(1), float64(2), int64(1), object(7)
memory usage: 1006.7+ KB
```

In [20]:
```
newdf['product_category'][100]
```

Out[20]:
```
'Clothing '
```

In [23]:
```
type(newdf['retail_price'][21])
```

Out[23]:
```
numpy.float64
```

retail_price,and discounted_price had some nan value, temporarily filled with 0 to avoid exceptions

In [33]:
```
newdf['retail_price'].fillna(0,inplace=True)
newdf.info()
#newdf['discounted_price'].fillna(0,inplace=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19449 entries, 0 to 19448
Data columns (total 11 columns):
Unnamed: 0                 19449 non-null int64
product_name               19449 non-null object
product_category_tree      19449 non-null object
retail_price               19449 non-null float64
discounted_price           19371 non-null float64
is_FK_Advantage_product    19449 non-null bool
product_rating             19449 non-null object
overall_rating             19449 non-null object
product_specifications     19435 non-null object
Brand_names                19449 non-null object
product_category           19449 non-null object
dtypes: bool(1), float64(2), int64(1), object(7)
memory usage: 1006.7+ KB
```

In [34]:
```
newdf['discounted_price'].fillna(0,inplace=True)
```

The nan value here, too had the same issue, so type casting it all to string, and separating them

In [37]:
```
#newdf.info()
def to_string(x):
    return str(x)
```

In [38]:
```
newdf['product_specs_refined'] = newdf['product_specifications'].apply(to_string)
```

In [40]:
```
dfwithnan = newdf[newdf['product_specs_refined']=='nan']
```

In [46]:
```
dfnotwithnan = newdf[newdf['product_specs_refined']!='nan']
```

In [49]:
```
dfnotwithnan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19435 entries, 0 to 19448
Data columns (total 12 columns):
Unnamed: 0                 19435 non-null int64
product_name               19435 non-null object
product_category_tree      19435 non-null object
retail_price               19435 non-null float64
discounted_price           19435 non-null float64
is_FK_Advantage_product    19435 non-null bool
product_rating             19435 non-null object
overall_rating             19435 non-null object
product_specifications     19435 non-null object
Brand_names                19435 non-null object
product_category           19435 non-null object
product_specs_refined      19435 non-null object
dtypes: bool(1), float64(2), int64(1), object(8)
memory usage: 1.2+ MB
```

In [41]:
```
dfwithnan.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14 entries, 21 to 13643
Data columns (total 12 columns):
Unnamed: 0                 14 non-null int64
product_name               14 non-null object
product_category_tree      14 non-null object
retail_price               14 non-null float64
discounted_price           14 non-null float64
is_FK_Advantage_product    14 non-null bool
product_rating             14 non-null object
overall_rating             14 non-null object
product_specifications     0 non-null object
Brand_names                14 non-null object
product_category           14 non-null object
product_specs_refined      14 non-null object
dtypes: bool(1), float64(2), int64(1), object(8)
memory usage: 910.0+ bytes
```

In [45]:
```
dfwithnan.head(1)
newdf['product_specifications'][0]
```

Out[45]:
```
'{"product_specification"=>[{"key"=>"Number of Contents in Sales Package", "val
ue"=>"Pack of 3"}, {"key"=>"Fabric", "value"=>"Cotton Lycra"}, {"key"=>"Type",
"value"=>"Cycling Shorts"}, {"key"=>"Pattern", "value"=>"Solid"}, {"key"=>"Idea
l For", "value"=>"Women\'s"}, {"value"=>"Gentle Machine Wash in Lukewarm Water,
Do Not Bleach"}, {"key"=>"Style Code", "value"=>"ALTHT_3P_21"}, {"value"=>"3 sh
orts"}]}'
```

In [57]:
```
type(dfnotwithnan['product_name'][0])
```

Out[57]:
```
str
```

In [58]:
```
type(dfnotwithnan['product_specifications'][0])
```

Out[58]:
```
str
```

Logic, matching product names should have near about equal specifications

In [61]:
```
spect_dict = {}
for row_index,row in dfnotwithnan.iterrows():
    #print(row)
    spect_dict[row['product_name']] = row['product_specifications']
```

In [54]:
```
type(spect_dict)
```

Out[54]:
```
dict
```

In [65]:
```
#spect_dict
dfwithnancopy = dfwithnan.copy()
```

In [67]:
```
for row_index,row in dfwithnancopy.iterrows():
    if row['product_name'] in spect_dict.keys():
        dfwithnancopy['product_specifications'][row_index] = spect_dict[row['prod
```

```
C:\Users\h_asp\Anaconda3\lib\site-packages\ipykernel\__main__.py:3: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  app.launch_new_instance()
```

In [74]:
```
for row_index,row in dfwithnancopy.iterrows():
    if row['product_name'] in spect_dict.keys():
        dfwithnancopy['product_specs_refined'][row_index] = spect_dict[row['produ
```

```
C:\Users\h_asp\Anaconda3\lib\site-packages\ipykernel\__main__.py:3: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  app.launch_new_instance()
```

Even after this filtering, 8 of the products' specification values could not be determined, hence dropped

In [75]:
```
dfwithnancopy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14 entries, 21 to 13643
Data columns (total 12 columns):
Unnamed: 0                 14 non-null int64
product_name               14 non-null object
product_category_tree      14 non-null object
retail_price               14 non-null float64
discounted_price           14 non-null float64
is_FK_Advantage_product    14 non-null bool
product_rating             14 non-null object
overall_rating             14 non-null object
product_specifications     6 non-null object
Brand_names                14 non-null object
product_category           14 non-null object
product_specs_refined      14 non-null object
dtypes: bool(1), float64(2), int64(1), object(8)
memory usage: 1.4+ KB
```

In [76]:
```
dfwithnancopy.head(1)
```

Out[76]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
21	21	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	0.0	0.0	

In [77]:
```
dfspecscleaned = dfwithnancopy[dfwithnancopy['product_specs_refined'] != 'nan']
```

In [87]:
```
dfspecscleaned.info()
dff2 = dfspecscleaned.copy()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6 entries, 21 to 12585
Data columns (total 12 columns):
Unnamed: 0                 6 non-null int64
product_name               6 non-null object
product_category_tree      6 non-null object
retail_price               6 non-null float64
discounted_price           6 non-null float64
is_FK_Advantage_product    6 non-null bool
product_rating             6 non-null object
overall_rating             6 non-null object
product_specifications     6 non-null object
Brand_names                6 non-null object
product_category           6 non-null object
product_specs_refined      6 non-null object
dtypes: bool(1), float64(2), int64(1), object(8)
memory usage: 390.0+ bytes
```

In [88]:
```
dff2.drop(['product_specifications'],axis=1,inplace=True)
```

In [83]:
```
dfnotwithnan.head(1)
dff1 = dfnotwithnan.copy()
```

In [84]:
```
dff1.drop(['product_specifications'],axis=1,inplace=True)
```

In [89]:
```
dff1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19435 entries, 0 to 19448
Data columns (total 11 columns):
Unnamed: 0                  19435 non-null int64
product_name                19435 non-null object
product_category_tree       19435 non-null object
retail_price                19435 non-null float64
discounted_price            19435 non-null float64
is_FK_Advantage_product     19435 non-null bool
product_rating              19435 non-null object
overall_rating              19435 non-null object
Brand_names                 19435 non-null object
product_category            19435 non-null object
product_specs_refined       19435 non-null object
dtypes: bool(1), float64(2), int64(1), object(7)
memory usage: 1.1+ MB
```

In [90]:
```
frames = [dff1,dff2]
```

In [91]:
```
final_flipkart = pd.concat(frames)
```

In [92]:
```
final_flipkart.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19441 entries, 0 to 12585
Data columns (total 11 columns):
Unnamed: 0                  19441 non-null int64
product_name                19441 non-null object
product_category_tree       19441 non-null object
retail_price                19441 non-null float64
discounted_price            19441 non-null float64
is_FK_Advantage_product     19441 non-null bool
product_rating              19441 non-null object
overall_rating              19441 non-null object
Brand_names                 19441 non-null object
product_category            19441 non-null object
product_specs_refined       19441 non-null object
dtypes: bool(1), float64(2), int64(1), object(7)
memory usage: 1.1+ MB
```

In [93]:
```
final_flipkart.to_csv('flipkart_spec_cat_brand.csv',encoding ='utf-8')
```

This dataframe now consists of filtered brand names, specification values, and categories, with zeros in nan values of dicounted and retail prices

Next see the Retail price, rating cleaning

In [ ]:

Globsyn Summer School 2018, Machine Learning Using Python, a project on analysis of flipkart data, by Kaushik, Mukund, Rajdeep and Salini

In [1]:
```
import numpy as np
import pandas as pd
```

In [119]:
```
df = pd.read_csv('flipkart_spec_cat_brand.csv')
```

In [120]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 12 columns):
Unnamed: 0                  19441 non-null int64
Unnamed: 0.1                19441 non-null int64
product_name                19441 non-null object
product_category_tree       19441 non-null object
retail_price                19441 non-null float64
discounted_price            19441 non-null float64
is_FK_Advantage_product     19441 non-null bool
product_rating              19441 non-null object
overall_rating              19441 non-null object
Brand_names                 19441 non-null object
product_category            19441 non-null object
product_specs_refined       19441 non-null object
dtypes: bool(1), float64(2), int64(2), object(7)
memory usage: 1.1+ MB
```

In [121]:
```
x = list(df.columns)
x[0],x[1] = 'a','b'
df.columns = x
df.drop(['a','b'],axis=1,inplace=True)
#'product_category_tree','product_name','product_specs_refined','is_FK_Advantage_
```

In [122]:
```
df.head(2)
```

Out[122]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	False
1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	False

In [123]:
```
df[df['retail_price']==0].head(3)
```

Out[123]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
12	Sicons All Purpose Tea Tree Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	0.0	0.0	False
70	Eurospa Cotton Terry Face Towel Set	["Baby Care >> Baby Bath & Skin >> Baby Bath T...	0.0	0.0	False
707	Techware Microwavable Tea Cups WF13115 - Purpl...	["Kitchen & Dining >> Dinnerware & Crockery >>...	0.0	0.0	False

In [124]:
```
df[df['discounted_price']==0].head(3)
```

Out[124]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
12	Sicons All Purpose Tea Tree Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	0.0	0.0	False
70	Eurospa Cotton Terry Face Towel Set	["Baby Care >> Baby Bath & Skin >> Baby Bath T...	0.0	0.0	False
707	Techware Microwavable Tea Cups WF13115 - Purpl...	["Kitchen & Dining >> Dinnerware & Crockery >>...	0.0	0.0	False

The prices which are equal to zero are taken out

In [125]:
```
df[df['retail_price']==0].info()
df[df['discounted_price']==0].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 78 entries, 12 to 19438
Data columns (total 10 columns):
product_name               78 non-null object
product_category_tree      78 non-null object
retail_price               78 non-null float64
discounted_price           78 non-null float64
is_FK_Advantage_product    78 non-null bool
product_rating             78 non-null object
overall_rating             78 non-null object
Brand_names                78 non-null object
product_category           78 non-null object
product_specs_refined      78 non-null object
dtypes: bool(1), float64(2), object(7)
memory usage: 4.0+ KB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 78 entries, 12 to 19438
Data columns (total 10 columns):
product_name               78 non-null object
product_category_tree      78 non-null object
retail_price               78 non-null float64
discounted_price           78 non-null float64
is_FK_Advantage_product    78 non-null bool
product_rating             78 non-null object
overall_rating             78 non-null object
Brand_names                78 non-null object
product_category           78 non-null object
product_specs_refined      78 non-null object
dtypes: bool(1), float64(2), object(7)
memory usage: 4.0+ KB
```

In [126]:
```
df_no_retail_disc = df[df['retail_price']==0]
```

In [127]:
```
df_retail_disc = df[df['retail_price']!=0]
```

Mean had too much potential of tampering with the inferences with mean data almost equal 3300 for retail, hence filling it with median

In [128]:
```
df_retail_disc['retail_price'].median()
```

Out[128]: 1049.0

In [129]:
```
df_retail_disc['discounted_price'].median()
```

Out[129]: 554.0

In [130]:
```
df_no_retail_disc['retail_price'] = df_retail_disc['retail_price'].median()
df_no_retail_disc['discounted_price'] = df_retail_disc['discounted_price'].median
```

```
C:\Users\h_asp\Anaconda3\lib\site-packages\ipykernel\__main__.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  if __name__ == '__main__':
C:\Users\h_asp\Anaconda3\lib\site-packages\ipykernel\__main__.py:2: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  from ipykernel import kernelapp as app
```

In [131]:
```
df_no_retail_disc.head(3)
```

Out[131]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
12	Sicons All Purpose Tea Tree Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	1049.0	554.0	False
70	Eurospa Cotton Terry Face Towel Set	["Baby Care >> Baby Bath & Skin >> Baby Bath T...	1049.0	554.0	False
707	Techware Microwavable Tea Cups WF13115 - Purpl...	["Kitchen & Dining >> Dinnerware & Crockery >>...	1049.0	554.0	False

In [132]:
```
frames = [df_retail_disc,df_no_retail_disc]
```

In [133]:
```
df = pd.concat(frames)
```

In [134]:
```
df.head()
```

Out[134]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	False
1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	False
2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	False
3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	False
4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	False

Checking for 'No ratings available' in overall_rating and product_rating

In [135]:
```
df[df['overall_rating'] == 'No rating available'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17595 entries, 0 to 19438
Data columns (total 10 columns):
product_name               17595 non-null object
product_category_tree      17595 non-null object
retail_price               17595 non-null float64
discounted_price           17595 non-null float64
is_FK_Advantage_product    17595 non-null bool
product_rating             17595 non-null object
overall_rating             17595 non-null object
Brand_names                17595 non-null object
product_category           17595 non-null object
product_specs_refined      17595 non-null object
dtypes: bool(1), float64(2), object(7)
memory usage: 910.7+ KB
```

Seperating and filling it with random integers between 0 to 5, since predicting 17595 on the basis of 1500 data would not be fair

In [136]:
```
df_no_ratings = df[df['overall_rating'] == 'No rating available']
```

In [137]:
```
df_ratings = df[df['overall_rating'] != 'No rating available']
```

In [138]:
```
np.random.seed(123)
df_no_ratings['product_rating'] = np.random.randint(0,6,17595)
df_no_ratings['overall_rating'] = np.random.randint(0,6,17595)
```

```
C:\Users\h_asp\Anaconda3\lib\site-packages\ipykernel\__main__.py:2: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  from ipykernel import kernelapp as app
C:\Users\h_asp\Anaconda3\lib\site-packages\ipykernel\__main__.py:3: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  app.launch_new_instance()
```

In [139]:
```
df_no_ratings.head(6)
```

Out[139]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	False
1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	False
2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	False
3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	False
4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	False
5	Eternal Gandhi Super Series Crystal Paper Weig...	["Eternal Gandhi Super Series Crystal Paper We...	430.0	430.0	False

In [140]:
```
x = np.random.randint(4,5,5)
```

In [141]:
```
x
```

Out[141]:
```
array([4, 4, 4, 4, 4])
```

In [142]:
```
frames = [df_no_ratings,df_ratings]
```

In [143]:
```
df = pd.concat(frames)
df.head(2)
```

Out[143]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product
0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	False
1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	False

In [147]:
```
df.to_csv('flipkart_final_cleared.csv',encoding='utf-8')
```

In [148]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19441 entries, 0 to 19179
Data columns (total 10 columns):
product_name               19441 non-null object
product_category_tree      19441 non-null object
retail_price               19441 non-null float64
discounted_price           19441 non-null float64
is_FK_Advantage_product    19441 non-null bool
product_rating             19441 non-null object
overall_rating             19441 non-null object
Brand_names                19441 non-null object
product_category           19441 non-null object
product_specs_refined      19441 non-null object
dtypes: bool(1), float64(2), object(7)
memory usage: 1006.2+ KB
```

This dataframe is the cleaned data finally and we have now 19441 data with all almost correct data

In [ ]:
```
```

Globsyn Summer School 2018, Machine Learning Using Python, a project on analysis of flipkart data, by Kaushik, Mukund, Rajdeep and Salini

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [2]:
```
df=pd.read_csv('flipkart_final_cleared.csv')
df.head()
```

Out[2]:

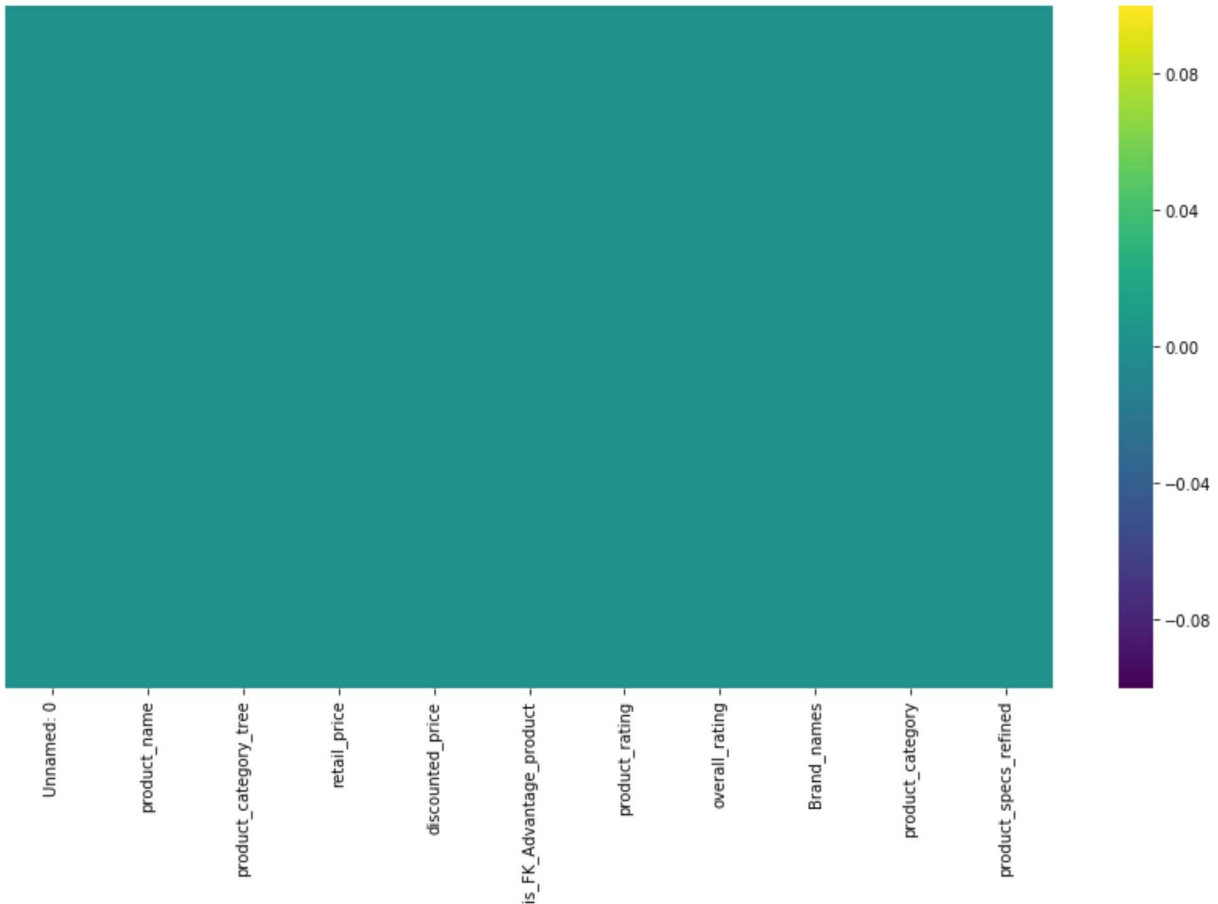
	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 11 columns):
Unnamed: 0                  19441 non-null int64
product_name                19441 non-null object
product_category_tree       19441 non-null object
retail_price                19441 non-null float64
discounted_price            19441 non-null float64
is_FK_Advantage_product     19441 non-null bool
product_rating              19441 non-null float64
overall_rating              19441 non-null float64
Brand_names                 19441 non-null object
product_category            19441 non-null object
product_specs_refined       19441 non-null object
dtypes: bool(1), float64(4), int64(1), object(5)
memory usage: 1.1+ MB
```

In [4]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.isnull(),yticklabels=False,cbar=True,cmap='viridis')
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0xf8f55d0>



In [5]:
```
fk = pd.get_dummies(df["is_FK_Advantage_product"],drop_first = True)
df['is_FK_Advantage_product'] = fk
df.head()
```

Out[5]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

In [6]:
```
df["discount_percent"] = ((df.retail_price - df.discounted_price)*100)/df.retail_
df.head()
```

Out[6]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

In [7]:
```
df[['product_name','Brand_names','retail_price','discounted_price','discount_perc
```

Out[7]:

	product_name	Brand_names	retail_price	discounted_price	discount_percent
0	Alisha Solid Women's Cycling Shorts	Alisha	999.0	379.0	62.062062
1	FabHomeDecor Fabric Double Sofa Bed	FabHomeDecor	32157.0	22646.0	29.576764
2	AW Bellies	AW	999.0	499.0	50.050050
3	Alisha Solid Women's Cycling Shorts	Alisha	699.0	267.0	61.802575
4	Sicons All Purpose Arnica Dog Shampoo	Sicons	220.0	210.0	4.545455

In [8]:
```
df['together'] = df['product_category'] + df['Brand_names']
```

In [9]:
```
df.head(2)
```

Out[9]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	

In [10]:
```
number = LabelEncoder()

df['together'] = number.fit_transform(df['together'])

df['product_category'] = number.fit_transform(df['product_category'])

df['discount_percent'] = number.fit_transform(df['discount_percent'])

df['Brand_names'] = number.fit_transform(df['Brand_names'])
```

In [11]:
```
df.head(2)
```

Out[11]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	

In [12]:
```
X= np.array(df.iloc[:,8:10])
Y = np.array(df.iloc[:,11])
```

In [13]:
```
X
```

Out[13]:
```
array([[ 219,   43],
       [1272,   76],
       [  93,   74],
       ...,
       [3461,   27],
       [4549,   74],
       [2144,   43]], dtype=int32)
```

In [14]:
```
features = X
target = Y
```

In [15]:
```
features_train, features_test, target_train, target_test = train_test_split(featu
```

Naive-Bayes

In [16]:
```
#Create a Gaussian Classifier
model = GaussianNB()

# Train the model using the training sets
model.fit(features_train, target_train)
```

Out[16]:
```
GaussianNB(priors=None)
```

In [17]:
```
predicted= model.predict([[1272,76]])
print(predicted)
```

```
[1985]
```

In [18]:
```
target_pred = model.predict(features_test)
```

In [19]:
```
accuracy_score(target_test, target_pred)
```

Out[19]:
```
0.12889650872817954
```

Linear Regression

In [20]:
```
lm = linear_model.LinearRegression()
model = lm.fit(features_train, target_train)
```

In [21]:
```
lm.score(features_test,target_test)
```

Out[21]:
```
0.02382894936030766
```

In [22]:
```
lm.coef_
```

Out[22]:
```
array([-0.00950612, -5.09897469])
```

In [23]:
```
lm.intercept_
```

Out[23]:
```
3317.7544736250875
```

In [24]:
```
pd.DataFrame(['Category','Brand'], lm.coef_)
```

Out[24]:

	0
-0.009506	Category
-5.098975	Brand

There is a strong negative correlation between Brand and discount_percentage

In [25]:
```
target_pred = lm.predict(features_test)
```

In [26]:
```
import sklearn.metrics
```

In [27]:
```
#etrics.explained_variance_score(target_test,target_pred)
```

In [28]:
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [29]:
```
print("Mean squared error: %.2f"% mean_squared_error(target_test, target_pred))
```

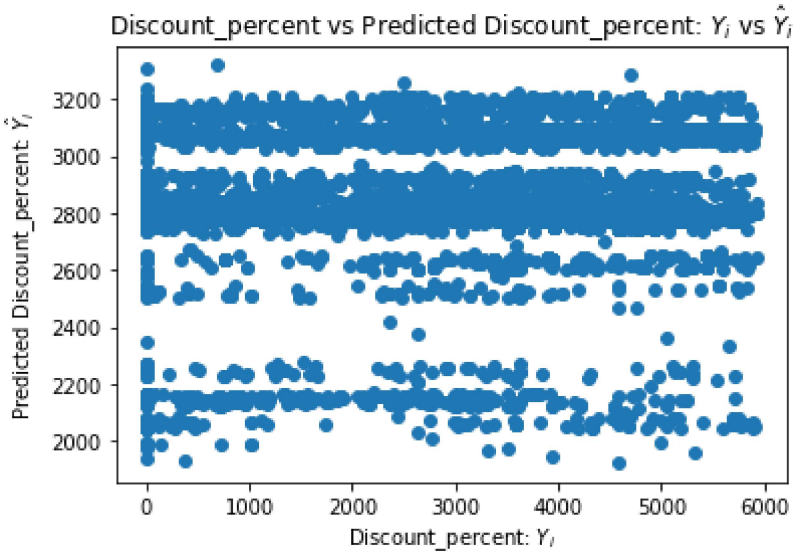
```
Mean squared error: 3222767.00
```

In [30]:
```
print('Variance score: %.2f' % r2_score(target_test, target_pred))
```

```
Variance score: 0.02
```

In [31]:
```
plt.scatter(target_test, target_pred)
plt.xlabel("Discount_percent: $Y_i$")
plt.ylabel("Predicted Discount_percent: $\hat{Y}_i$")
plt.title("Discount_percent vs Predicted Discount_percent: $Y_i$ vs $\hat{Y}_i$")
```

Out[31]:
```
Text(0.5,1,'Discount_percent vs Predicted Discount_percent: $Y_i$ vs $\\hat{Y}_
i$')
```



In [32]:
```
plt.close()
sns.set_style("whitegrid");
sns.pairplot(df,hue="product_rating",size=4);
plt.show()
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

In [ ]:

Flipkart E-commerce Data of various products

This project is made by a team of Batch 3

Kaushik Ghosh

Mukund Agarwal

Salini Mukherjee

Rajdeep Koner

Context

This is a pre-crawled dataset, taken as subset of a bigger dataset (more than 5.8 million products) that was created by extracting data from Flipkart.com, a leading Indian eCommerce store.

Content

This dataset has following fields: product_url product_name product_category_tree pid retail_price discounted_price image is_FK_Advantage_product description product_rating overall_rating brand product_specifications

Acknowledgements

This dataset was created by PromptCloud's in-house web-crawling service.

Inspiration

Analyses of the pricing, product specification and brand can be performed.

In [1]:
```
# Import Statement

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
from sklearn import datasets
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_blobs
from sklearn.model_selection import KFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from mpl_toolkits.mplot3d import Axes3D

import os
os.environ['PATH'] += os.pathsep +'C:\Program Files (x86)/Graphviz2.38/bin/'
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot
get_ipython().run_line_magic('matplotlib', 'inline')
```

In [2]:
```
# Load the Dataset

df=pd.read_csv('flipkart_final_cleared.csv')
df.head()
```

Out[2]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

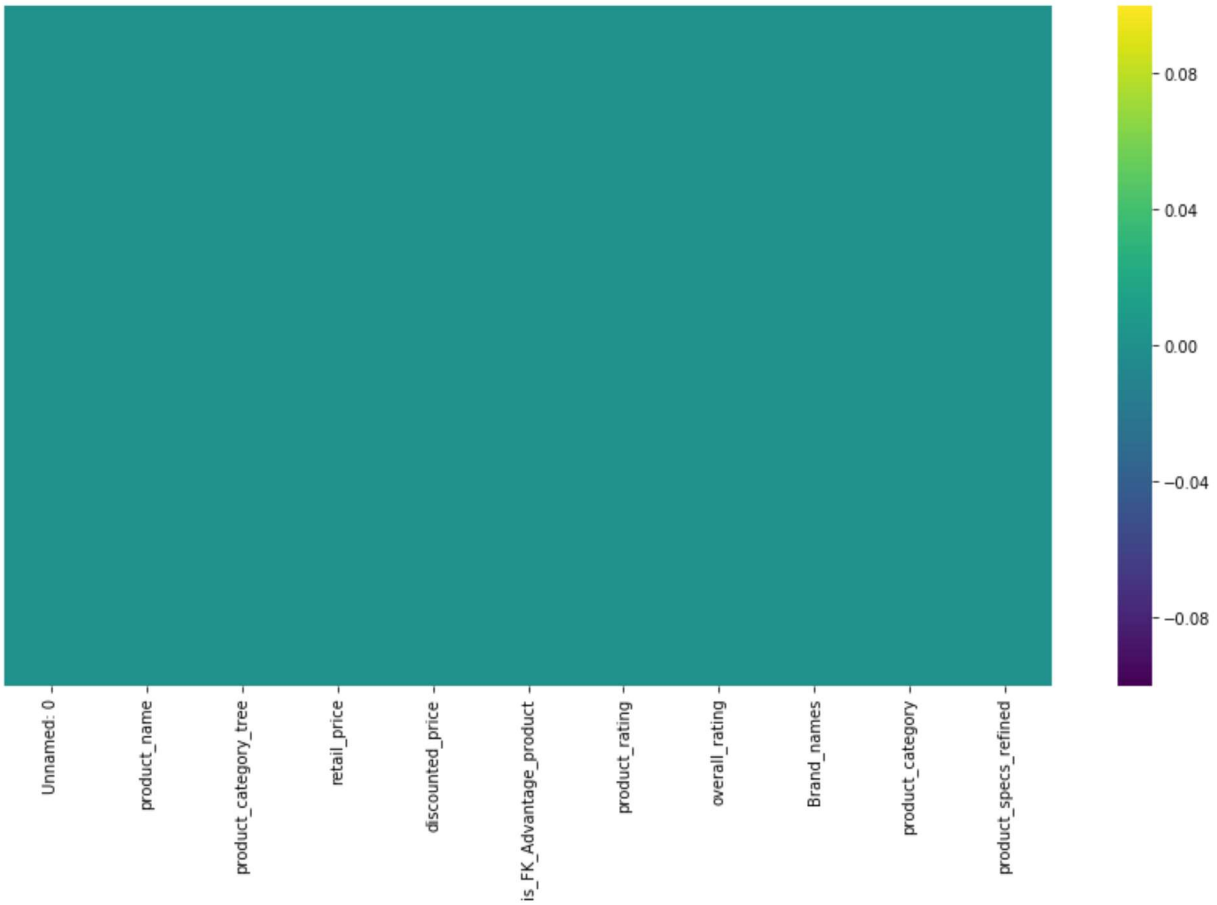
In [3]:
```
# Information of the Datset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 11 columns):
Unnamed: 0                 19441 non-null int64
product_name               19441 non-null object
product_category_tree      19441 non-null object
retail_price               19441 non-null float64
discounted_price           19441 non-null float64
is_FK_Advantage_product    19441 non-null bool
product_rating             19441 non-null float64
overall_rating             19441 non-null float64
Brand_names                19441 non-null object
product_category           19441 non-null object
product_specs_refined      19441 non-null object
dtypes: bool(1), float64(4), int64(1), object(5)
memory usage: 1.5+ MB
```

In [4]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.isnull(),yticklabels=False,cbar=True,cmap='viridis')
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x2461a8b7ef0>



In [5]:
```
fk = pd.get_dummies(df["is_FK_Advantage_product"],drop_first = True)
df['is_FK_Advantage_product'] = fk
df.head()
```

Out[5]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

In [6]:
```
df['product_rating'] = df['product_rating'].replace("No rating available",np.nan)
df['overall_rating'] = df['overall_rating'].replace("No rating available",np.nan)
```

In [7]:
```
pr=round(df.product_rating.isnull().sum()*100/df.shape[0], 2)
pr
```

Out[7]: 0.0

In [8]:
```
orr=round(df.overall_rating.isnull().sum()*100/df.shape[0], 2)
orr
```

Out[8]: 0.0

In [9]:
```
df['product_rating'] = pr
df['overall_rating'] = orr
df.head(10)
```

Out[9]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	
5	5	Eternal Gandhi Super Series Crystal Paper Weig...	["Eternal Gandhi Super Series Crystal Paper We...	430.0	430.0	
6	6	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	1199.0	479.0	
7	7	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
8	8	dilli bazaaar Bellies, Corporate Casuals, Casuals	["Footwear >> Women's Footwear >> Ballerinas >...	699.0	349.0	
9	9	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	1199.0	479.0	

In [10]:
```
# Finding the Discount Percentage
df["discount_percent"] = ((df.retail_price - df.discounted_price)*100)/df.retail_
df.head()
```

Out[10]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

Top Brands

In [11]:
```
top_Brands=df.groupby(['product_category','Brand_names'],as_index=False).agg({'di
top_Brands=np.round(top_Brands,2)
top_Brands=top_Brands.reindex()
top_Brands=top_Brands.sort_values(by='discount_percent',ascending=False)
top_Brands
```

Out[11]:

	product_category	Brand_names	discount_percent
3552	Home Furnishing	Rajcrafts	96.53
4507	Mobiles & Accessories	Bling	94.55
2326	Computers	Instella	91.72
4510	Mobiles & Accessories	Bond Beatz	91.60
258	Baby Care	Poppins	90.90
1364	Clothing	KazamaKraft	90.57
2352	Computers	Mydress Mystyle	90.52
1367	Clothing	Kea	90.05
2982	Gaming	SDZ	90.05
4517	Mobiles & Accessories	CUBA	90.05
354	Bags, Wallets & Belts	Elligator	90.02
3962	Jewellery	Kaizer Jewelry	89.62
3810	Jewellery	Fash Blush	89.26
3811	Jewellery	FashBlush	88.98
1004	Clothing	Dark Green	88.53
4054	Jewellery	Remix	88.43
5003	Sunglasses	barbarik	88.04
4554	Mobiles & Accessories	Head Kik	88.02
48	Automotive	Bracketron	87.94
3813	Jewellery	Fashblush	87.49
5303	Watches	JackKlein	86.89
5388	Watches	palito	86.73
4394	Kitchen & Dining	Rhythem Listers	86.24
424	Bags, Wallets & Belts	Shoprider	85.54
1139	Clothing	FreeSoul	85.03
887	Clothing	Black Bee	85.03
842	Clothing	Autokartindia	85.01
3388	Home Decor & Festive Needs	narayan export	85.01
3858	Jewellery	Gliteri	85.00
5304	Watches	Jackklein	84.98
...	...	...	...
1340	Clothing	KAVACi	0.00

	product_category	Brand_names	discount_percent
4147	Jewellery	Utkalika	0.00
4160	Jewellery	Velvetcase	0.00
4776	Pens & Stationery	Probott	0.00
4734	Pens & Stationery	Filofax	0.00
1493	Clothing	LoweAlpine	0.00
4736	Pens & Stationery	Freelance	0.00
1569	Clothing	ModoVivendi	0.00
1565	Clothing	Mizuno	0.00
1561	Clothing	MissGrace	0.00
4740	Pens & Stationery	Hm International	0.00
1553	Clothing	Mi Dulce An'ya	0.00
1542	Clothing	Megha	0.00
4744	Pens & Stationery	InstaNote	0.00
1539	Clothing	Mee Mee	0.00
1533	Clothing	Max	0.00
1529	Clothing	MasterweaverIndia	0.00
4754	Pens & Stationery	Maped	0.00
4756	Pens & Stationery	Max	0.00
4757	Pens & Stationery	Montana	0.00
4164	Jewellery	Vijisan	0.00
1522	Clothing	Marks & Spencer	0.00
4761	Pens & Stationery	Nourish	0.00
4763	Pens & Stationery	Offspring	0.00
1513	Clothing	Majori	0.00
3176	Home Decor & Festive Needs	Importwala	0.00
4771	Pens & Stationery	Piedpaper	0.00
4773	Pens & Stationery	Platignum	0.00
4774	Pens & Stationery	Platinum	0.00
4678	Nine Maternity Wear Women's Fit and Flare Dress"	Nine Maternity Wear	0.00

5421 rows × 3 columns

In [12]:
```
df[['product_name','Brand_names','retail_price','discounted_price','discount_perc
```

Out[12]:

	product_name	Brand_names	retail_price	discounted_price	discount_percent
0	Alisha Solid Women's Cycling Shorts	Alisha	999.0	379.0	62.062062
1	FabHomeDecor Fabric Double Sofa Bed	FabHomeDecor	32157.0	22646.0	29.576764
2	AW Bellies	AW	999.0	499.0	50.050050
3	Alisha Solid Women's Cycling Shorts	Alisha	699.0	267.0	61.802575
4	Sicons All Purpose Arnica Dog Shampoo	Sicons	220.0	210.0	4.545455

Naive Bayes Algorithm

In [13]:
```
number = LabelEncoder()

df['Brand_names'] = number.fit_transform(df['Brand_names'])

df['product_category'] = number.fit_transform(df['product_category'])

df['is_FK_Advantage_product'] = number.fit_transform(df['is_FK_Advantage_product'
```

In [14]:
```
features = ["Brand_names", "product_category"]
target = "is_FK_Advantage_product"
```

In [15]:
```
features_train, features_test, target_train, target_test = train_test_split(df[fe
```

In [16]:
```
model = GaussianNB()
model.fit(features_train, target_train)
```

Out[16]:
```
GaussianNB(priors=None)
```

In [17]:
```
G_pred = model.predict(features_test)
accuracy = accuracy_score(target_test, G_pred)
print("Accuracy = ",accuracy)
```

```
Accuracy =  0.956359102244389
```

In [18]:
```
print (model.predict([[5,3]]))
```

```
[0]
```

KNN Algorithm

In [19]:
```
knn=KNeighborsClassifier(n_neighbors=2)
```

In [20]:
```
knn.fit(features_train, target_train)
```

Out[20]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=2, p=2,
           weights='uniform')
```

In [21]:
```
knn_pred = knn.predict(features_test)
accuracy_knn = accuracy_score(target_test, knn_pred)
print("Accuracy = ",accuracy_knn)
```

```
Accuracy =  0.9655548628428927
```

In [22]:
```
print(confusion_matrix(target_test, knn_pred))
```

```
[[6112   27]
 [ 194   83]]
```

In [23]:
```
print(classification_report(target_test, knn_pred))
```

```
             precision    recall  f1-score   support

          0       0.97      1.00      0.98      6139
          1       0.75      0.30      0.43       277

avg / total       0.96      0.97      0.96      6416
```

Logistic Regression

In [24]:
```
logmodel = LogisticRegression()
logmodel.fit(features_train, target_train)
```

Out[24]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [25]:
```
log_pred = logmodel.predict(features_test)
accuracy_log = accuracy_score(target_test, log_pred)
print("Accuracy = ",accuracy_log)
```

```
Accuracy =  0.9568266832917706
```

In [26]:
```
print(confusion_matrix(target_test, log_pred))
```

```
[[6139    0]
 [ 277    0]]
```

In [27]:
```
print(classification_report(target_test, log_pred))
```

```
             precision    recall  f1-score   support

          0       0.96      1.00      0.98      6139
          1       0.00      0.00      0.00       277

avg / total       0.92      0.96      0.94      6416
```

```
C:\Users\koner\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:11
35: UndefinedMetricWarning: Precision and F-score are ill-defined and being set
to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

Perceptron

In [28]:
```
sc = StandardScaler()
sc.fit(features_train)
```

Out[28]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [29]:
```
features_train_std = sc.transform(features_train)
features_test_std = sc.transform(features_test)
```

In [30]:
```
ppn = Perceptron(n_iter = 40,eta0 = 0.1, random_state=101)
```

In [31]:
```
ppn.fit(features_train, target_train)
```

```
C:\Users\koner\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:117: DeprecationWarning: n_iter parameter is deprecated in 0.19 and wil
l be removed in 0.21. Use max_iter and tol instead.
  DeprecationWarning)
```

Out[31]:
```
Perceptron(alpha=0.0001, class_weight=None, eta0=0.1, fit_intercept=True,
      max_iter=None, n_iter=40, n_jobs=1, penalty=None, random_state=101,
      shuffle=True, tol=None, verbose=0, warm_start=False)
```

In [32]:
```
P_pred = ppn.predict(features_test_std)
```

In [33]:
```
P_pred
```

Out[33]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [34]:
```
target_test
```

Out[34]:
```
18779    1
12074    0
6986     0
15079    0
16432    0
11603    0
10720    0
495      0
370      0
564      0
215      0
13634    0
14810    0
5223     0
6361     0
8847     0
19057    0
10552    0
16448    0
10115    0
18316    0
269      0
16660    0
16349    0
13617    0
14261    0
7736     0
4158     0
2244     0
17444    0
         ..
10922    0
989      0
4201     0
8718     0
5163     0
8810     0
15427    0
5900     0
18627    0
18348    1
8012     0
119      0
14469    0
9800     0
18849    1
5151     0
16817    0
7691     0
17063    0
14438    0
7265     0
17462    0
11433    0
18229    0
```

```
17935    0
8277     0
1910     0
15622    0
18175    0
2486     0
Name: is_FK_Advantage_product, Length: 6416, dtype: int64
```

Examine Accuracy Metric

In [35]:
```
print('Accuracy: %2f' %accuracy_score(target_test,P_pred))
```

```
Accuracy: 0.956827
```

Decision Tree

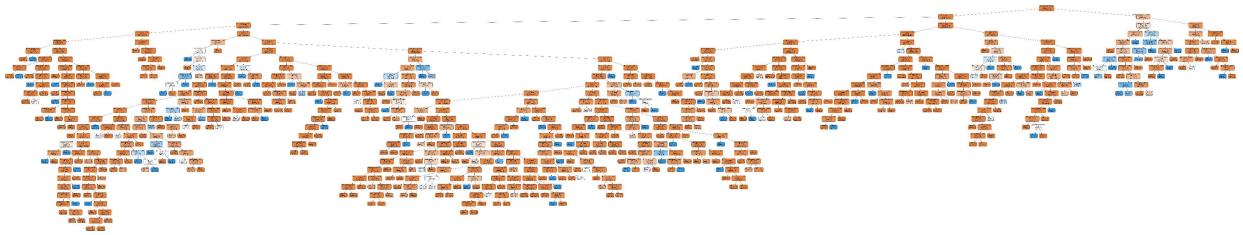
In [36]:
```
dtree=DecisionTreeClassifier()
```

In [37]:
```
dtree.fit(features_train, target_train)
```

Out[37]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [38]:
```
dot_data=StringIO()
export_graphviz(dtree, out_file=dot_data,feature_names=features,filled=True,round
graph=pydot.graph_from_dot_data(dot_data.getvalue())
Image(graph[0].create_png())
```

Out[38]:



In [39]:
```
probability_dtree = dtree.predict_proba(features_test)

print (probability_dtree)
```

```
[[0.26666667 0.73333333]
 [1.         0.        ]
 [1.         0.        ]
 ...
 [1.         0.        ]
 [0.87096774 0.12903226]
 [1.         0.        ]]
```

In [40]:
```
dtree_pred=dtree.predict(features_test)
```

In [41]:
```
print(confusion_matrix(target_test,dtree_pred))
```

```
[[6059   80]
 [ 159  118]]
```

In [42]:
```
print(classification_report(target_test,dtree_pred))
```

```
             precision    recall  f1-score   support

          0       0.97      0.99      0.98      6139
          1       0.60      0.43      0.50       277

avg / total       0.96      0.96      0.96      6416
```

In [43]:
```
accuracy_score(target_test,dtree_pred)
```

Out[43]:
```
0.9627493765586035
```

Random Forest

In [44]:
```
rfc=RandomForestClassifier(n_estimators=100)
rfc.fit(features_train, target_train)
```

Out[44]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [45]:
```
rfc_pred=rfc.predict(features_test)
```

In [46]:
```
print(confusion_matrix(target_test,rfc_pred))
```

```
[[6079   60]
 [ 159  118]]
```

In [47]:
```
accuracy_score(target_test, rfc_pred)
```

Out[47]:
```
0.9658665835411472
```

In [48]:
```
probability_rfc = rfc.predict_proba(features_test)

print (probability_rfc)
```

```
[[0.27193545 0.72806455]
 [1.         0.        ]
 [0.99571429 0.00428571]
 ...
 [1.         0.        ]
 [0.87293956 0.12706044]
 [1.         0.        ]]
```

In [49]:
```
print(classification_report(target_test,rfc_pred))
```

```
             precision    recall  f1-score   support

          0       0.97      0.99      0.98      6139
          1       0.66      0.43      0.52       277

avg / total       0.96      0.97      0.96      6416
```

We will go with the Random Forest model to predict the values since it has the highest accuracy 96.58%

Analysis of Flipkart data

A project by Kaushik,Mukund,Rajdeep,Salini , Batch 3, GLobsyn Machine Learning with Python 2018 Summer School

Determine the specifications keys corresponding to each product category from the product_specifications column

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('flipkart_spec_cat_brand.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	Unnamed: 0.1	product_name	product_category_tree	retail_price	discounted_price	is_F
0	0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

In [4]:
```
df2 = df.columns.get_values()
df2[0] = 'not_req'
df2[1] = 'not_req2'
```

In [5]:
```
df.columns = df2
```

In [6]:
```
df.head(1)
```

Out[6]:

	not_req	not_req2	product_name	product_category_tree	retail_price	discounted_price	is_FK_A
0	0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	

In [7]:
```
df.drop(['not_req','not_req2','product_name','product_category_tree','discounted_
```

In [8]:
```
df.head()
```

Out[8]:

	retail_price	is_FK_Advantage_product	product_rating	overall_rating	Brand_names	product_cat
0	999.0	False	No rating available	No rating available	Alisha	Cl
1	32157.0	False	No rating available	No rating available	FabHomeDecor	Fu
2	999.0	False	No rating available	No rating available	AW	Fo
3	699.0	False	No rating available	No rating available	Alisha	Cl
4	220.0	False	No rating available	No rating available	Sicons	Pet Su

In [9]:
```
df.drop(['retail_price','is_FK_Advantage_product','product_rating','overall_ratin
```

In [10]:
```
df.head()
```

Out[10]:

	product_category	product_specs_refined
0	Clothing	{"product_specification"=>[{"key"=>"Number of ...
1	Furniture	{"product_specification"=>[{"key"=>"Installati...
2	Footwear	{"product_specification"=>[{"key"=>"Ideal For"...
3	Clothing	{"product_specification"=>[{"key"=>"Number of ...
4	Pet Supplies	{"product_specification"=>[{"key"=>"Pet Type",...

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 2 columns):
product_category         19441 non-null object
product_specs_refined    19441 non-null object
dtypes: object(2)
memory usage: 151.9+ KB
```

In [12]:
```
df['product_category'][6000] in df['product_specs_refined'][6000]
```

Out[12]:
```
False
```

In [13]:
```
df['product_category'][4000]
```

Out[13]:
```
'Kitchen & Dining '
```

In [14]:
```
x = df['product_specs_refined'][4000]
```

In [15]:
```
x
```

Out[15]:
```
'{"product_specification"=>[{"key"=>"Pan Type", "value"=>"Chetty"}, {"key"=>"No
n-stick", "value"=>"Yes"}, {"key"=>"Brand", "value"=>"Kumkum"}, {"key"=>"Model
Number", "value"=>"KK110"}, {"key"=>"Induction Bottom", "value"=>"No"}, {"key"=
>"Type", "value"=>"Kadhai"}, {"key"=>"Material", "value"=>"Aluminium"}, {"key"=
>"Model Name", "value"=>"Nonstick Deep"}, {"key"=>"Lid Included", "value"=>"Ye
s"}, {"key"=>"Capacity", "value"=>"3 L"}, {"key"=>"Color", "value"=>"Red, Blac
k"}, {"key"=>"Sales Package", "value"=>"1 Deep Kadhai With Lid"}, {"key"=>"Pack
of", "value"=>"1"}, {"key"=>"Diameter", "value"=>"35 cm"}, {"key"=>"Weight", "v
alue"=>"700 g"}]}'
```

In [16]:
```
x.index('key')
```

Out[16]:
```
29
```

In [17]:
```
x = x[36:]
```

In [18]:
```
x
```

Out[18]:
```
'Pan Type", "value"=>"Chetty"}, {"key"=>"Non-stick", "value"=>"Yes"}, {"key"=
>"Brand", "value"=>"Kumkum"}, {"key"=>"Model Number", "value"=>"KK110"}, {"ke
y"=>"Induction Bottom", "value"=>"No"}, {"key"=>"Type", "value"=>"Kadhai"}, {"k
ey"=>"Material", "value"=>"Aluminium"}, {"key"=>"Model Name", "value"=>"Nonstic
k Deep"}, {"key"=>"Lid Included", "value"=>"Yes"}, {"key"=>"Capacity", "value"=
>"3 L"}, {"key"=>"Color", "value"=>"Red, Black"}, {"key"=>"Sales Package", "val
ue"=>"1 Deep Kadhai With Lid"}, {"key"=>"Pack of", "value"=>"1"}, {"key"=>"Diam
eter", "value"=>"35 cm"}, {"key"=>"Weight", "value"=>"700 g"}]}'
```

In [19]:
```
x.index('"')
```

Out[19]:
```
8
```

In [20]:
```
y = x[:8]
print(y)
```

```
Pan Type
```

In [21]:
```
x = x[8:]
x
```

Out[21]:
```
'", "value"=>"Chetty"}, {"key"=>"Non-stick", "value"=>"Yes"}, {"key"=>"Brand",
"value"=>"Kumkum"}, {"key"=>"Model Number", "value"=>"KK110"}, {"key"=>"Inducti
on Bottom", "value"=>"No"}, {"key"=>"Type", "value"=>"Kadhai"}, {"key"=>"Materi
al", "value"=>"Aluminium"}, {"key"=>"Model Name", "value"=>"Nonstick Deep"},
{"key"=>"Lid Included", "value"=>"Yes"}, {"key"=>"Capacity", "value"=>"3 L"},
{"key"=>"Color", "value"=>"Red, Black"}, {"key"=>"Sales Package", "value"=>"1 D
eep Kadhai With Lid"}, {"key"=>"Pack of", "value"=>"1"}, {"key"=>"Diameter", "v
alue"=>"35 cm"}, {"key"=>"Weight", "value"=>"700 g"}]}'
```

In [22]:
```
key_count = x.count('key')
```

In [23]:
```
key_count
```

Out[23]:
```
14
```

In [24]:
```
x = df['product_specs_refined'][4000]
```

In [25]:
```
keys = []
key_count = x.count('key')
for i in range(key_count):
    m = x.index('key')
    x = x[m+7:]
    m = x.index('"')
    y = x[:m]
    x = x[m:]
    keys.append(y)
```

In [26]:
```
keys
```

Out[26]:
```
['Pan Type',
 'Non-stick',
 'Brand',
 'Model Number',
 'Induction Bottom',
 'Type',
 'Material',
 'Model Name',
 'Lid Included',
 'Capacity',
 'Color',
 'Sales Package',
 'Pack of',
 'Diameter',
 'Weight']
```

In [27]:
```
def getkey(x):
    keys = []
    key_count = x.count('key')
    for i in range(key_count):
        m = x.index('key')
        x = x[m+7:]
        m = x.index('"')
        y = x[:m]
        x = x[m:]
        keys.append(y)
    return keys
```

In [28]:
```
category_keys_dict = {}
```

In [29]:
```
for row_index,row in df.iterrows():
    try:
        category_keys_dict[row['product_category']] = getkey(row['product_specs_r
    except ValueError:
        category_keys_dict[row['product_category']] = 'No keys'
```

In [30]:
```
category_keys_dict
```

Out[30]:
```
{'883 Police Full Sleeve Solid Men\'s Jacket"': ['Sleeve',
  'Hooded',
  'Reversible',
  'Fabric',
  'Pattern',
  'Ideal For',
  'Style Code'],
 'ABEEZ Boys, Men, Girls (Black, Pack of 1)"': ['Material',
  'Adjustable',
  'Size',
  'Number of Contents in Sales Package',
  'Sales Package'],
 'ANAND ARCHIES Girls Flats"': ['Ideal For',
  'Occasion',
  'Sole Material',
  'Type',
  'Heel Height',
  'Outer Material',
  'Color'],
 'ANAND ARCHIES Girls Wedges"': ['Ideal For'
```

In [31]:
```
df.head(1)
```

Out[31]:

	product_category	product_specs_refined
0	Clothing	{"product_specification"=>[{"key"=>"Number of ...

In [32]:
```
categories = category_keys_dict.keys()
type(categories)
categories = list(categories)
```

In [33]:
```
categories
```

Out[33]:
```
['Clothing ',
 'Furniture ',
 'Footwear ',
 'Pet Supplies ',
 'Eternal Gandhi Super Series Crystal Paper Weight..."',
 'Pens & Stationery ',
 'Bengal Blooms Rose Artificial Plant  with Pot (3..."',
 'Bags, Wallets & Belts ',
 'Home Decor & Festive Needs ',
 'Automotive ',
 'Tools & Hardware ',
 'Vishudh Printed Women\'s Straight Kurta"',
 'Vishudh Printed Women\'s Anarkali Kurta"',
 'BuildTrack PIR Wireless Motion Sensor - One Swit..."',
 'Skayvon SUMMERSIBLE SINGLE PHASE PUMP CONTROLLER..."',
 'MASARA Solid Women\'s Straight Kurta"',
 'Skayvon SUBMERSIBBLE THREE PHASE PUMP CONTROLLER..."',
 'Behringer Xenyx 502 Analog Sound Mixer"',
 'Noor Embroidered Women\'s Straight Kurta"',
```

In [34]:
```
keys = category_keys_dict.values()
type(keys)
keys = list(keys)
```

In [35]:
```
keys
```

Out[35]:
```
[['Pattern', 'Pattern', 'Occasion', 'Ideal For', 'Sleeve', 'Fabric'],
 ['Foldable',
  'Brand',
  'Delivery Condition',
  'Type',
  'Style',
  'Seating Capacity',
  'Upholstery Type',
  'Upholstery Included',
  'Suitable For',
  'Model Number',
  'Armrest Included',
  'Finish Type',
  'Care Instructions',
  'Weight',
  'Height',
  'Width',
  'Depth',
  'Covered in Warranty',
```

In [36]:
```
dfkeyscategories = pd.DataFrame(categories)
```

In [37]:
```
dfkeyscategories['Specification_keys'] = keys
```

In [38]:
```
dfkeyscategories.columns = ['Category','Specification_keys']
```

In [39]:
```
dfkeyscategories.head(3)
```

Out[39]:

	Category	Specification_keys
0	Clothing	[Pattern, Pattern, Occasion, Ideal For, Sleeve...
1	Furniture	[Foldable, Brand, Delivery Condition, Type, St...
2	Footwear	[Occasion, Ideal For, Type, Heel Height, Outer...

Create lists of specification value pairs for each product. The keys should be determined by the product category of that product

In [40]:
```
df.head()
```

Out[40]:

	product_category	product_specs_refined
0	Clothing	{"product_specification"=>[{"key"=>"Number of ...
1	Furniture	{"product_specification"=>[{"key"=>"Installati...
2	Footwear	{"product_specification"=>[{"key"=>"Ideal For"...
3	Clothing	{"product_specification"=>[{"key"=>"Number of ...
4	Pet Supplies	{"product_specification"=>[{"key"=>"Pet Type",...

In [41]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 2 columns):
product_category         19441 non-null object
product_specs_refined    19441 non-null object
dtypes: object(2)
memory usage: 151.9+ KB
```

In [42]:
```
x = df['product_specs_refined'][4000]
x
```

Out[42]:
```
'{"product_specification"=>[{"key"=>"Pan Type", "value"=>"Chetty"}, {"key"=>"No
n-stick", "value"=>"Yes"}, {"key"=>"Brand", "value"=>"Kumkum"}, {"key"=>"Model
Number", "value"=>"KK110"}, {"key"=>"Induction Bottom", "value"=>"No"}, {"key"=
>"Type", "value"=>"Kadhai"}, {"key"=>"Material", "value"=>"Aluminium"}, {"key"=
>"Model Name", "value"=>"Nonstick Deep"}, {"key"=>"Lid Included", "value"=>"Ye
s"}, {"key"=>"Capacity", "value"=>"3 L"}, {"key"=>"Color", "value"=>"Red, Blac
k"}, {"key"=>"Sales Package", "value"=>"1 Deep Kadhai With Lid"}, {"key"=>"Pack
of", "value"=>"1"}, {"key"=>"Diameter", "value"=>"35 cm"}, {"key"=>"Weight", "v
alue"=>"700 g"}]}'
```

In [43]:
```
x.index('value')
```

Out[43]:
```
48
```

In [44]:
```
x = x[57:]
x
```

Out[44]:
```
'Chetty"}, {"key"=>"Non-stick", "value"=>"Yes"}, {"key"=>"Brand", "value"=>"Kum
kum"}, {"key"=>"Model Number", "value"=>"KK110"}, {"key"=>"Induction Bottom",
"value"=>"No"}, {"key"=>"Type", "value"=>"Kadhai"}, {"key"=>"Material", "valu
e"=>"Aluminium"}, {"key"=>"Model Name", "value"=>"Nonstick Deep"}, {"key"=>"Lid
Included", "value"=>"Yes"}, {"key"=>"Capacity", "value"=>"3 L"}, {"key"=>"Colo
r", "value"=>"Red, Black"}, {"key"=>"Sales Package", "value"=>"1 Deep Kadhai Wi
th Lid"}, {"key"=>"Pack of", "value"=>"1"}, {"key"=>"Diameter", "value"=>"35 c
m"}, {"key"=>"Weight", "value"=>"700 g"}]}'
```

In [45]:
```
x.index('"')
```

Out[45]:
```
6
```

In [46]:
```
x[:6]
```

Out[46]:
```
'Chetty'
```

In [47]:
```
x[6:]
```

Out[47]:
```
'"}, {"key"=>"Non-stick", "value"=>"Yes"}, {"key"=>"Brand", "value"=>"Kumkum"},
{"key"=>"Model Number", "value"=>"KK110"}, {"key"=>"Induction Bottom", "value"=
>"No"}, {"key"=>"Type", "value"=>"Kadhai"}, {"key"=>"Material", "value"=>"Alumi
nium"}, {"key"=>"Model Name", "value"=>"Nonstick Deep"}, {"key"=>"Lid Include
d", "value"=>"Yes"}, {"key"=>"Capacity", "value"=>"3 L"}, {"key"=>"Color", "val
ue"=>"Red, Black"}, {"key"=>"Sales Package", "value"=>"1 Deep Kadhai With Li
d"}, {"key"=>"Pack of", "value"=>"1"}, {"key"=>"Diameter", "value"=>"35 cm"},
{"key"=>"Weight", "value"=>"700 g"}]}'
```

In [48]:
```
def getvalues(x):
    values = []
    value_count = x.count('value')
    for i in range(value_count):
        m = x.index('value')
        x = x[m+9:]
        m = x.index('"')
        y = x[:m]
        x = x[m:]
        values.append(y)
    return values
```

In [49]:
```
category_values_dict = {}
for row_index,row in df.iterrows():
    try:
        category_values_dict[row['product_category']] = getvalues(row['product_sp
    except ValueError:
        category_values_dict[row['product_category']] = 'No values'
```

In [50]:
```
category_values_dict
```

Out[50]:
```
{'883 Police Full Sleeve Solid Men\'s Jacket"': ['Full Sleeve',
  'No',
  'No',
  'Polyester',
  'Solid',
  "Men's",
  'Jacket',
  'CORBET16A-BLACK',
  'Wash with Similar Colors, Use Detergent for Colors'],
 'ABEEZ Boys, Men, Girls (Black, Pack of 1)"': ['SILICONE RUBBER',
  'Yes',
  'One Size Fit For All',
  'Pack of 1',
  '1wrist band'],
 'ANAND ARCHIES Girls Flats"': ['Girls',
  'Casual',
  'PU',
  'Flats',
  '1 inch',
```

In [51]:
```
df_keys_values = dfkeyscategories.copy()
```

In [52]:
```
df_keys_values.head()
```

Out[52]:

	Category	Specification_keys
0	Clothing	[Pattern, Pattern, Occasion, Ideal For, Sleeve...
1	Furniture	[Foldable, Brand, Delivery Condition, Type, St...
2	Footwear	[Occasion, Ideal For, Type, Heel Height, Outer...
3	Pet Supplies	[Pet Type, Body Material, Brand, Shape, Suitab...
4	Eternal Gandhi Super Series Crystal Paper Weig...	[Model Name, Weight, Paper Weight Material, Pa...

In [53]:
```
values = list(category_values_dict.values())
```

In [54]:
```
values
```

Out[54]:
```
[['Solid',
  'Solid',
  'Casual',
  "Women's",
  'Full Sleeve',
  'Cotton',
  '2 Shrug',
  'Wash with Similar Colors'],
 ['1 Kids Table 1 Kids Chair',
  'The color of some product parts may vary from what is shown in the image',
  "- The color of the product may vary slightly compared to the picture displ
ayed on your screen. This is due to lighting, pixel quality and color setting
s - Please check the product's dimensions to ensure the product will fit in t
he desired location. Also, check if the product will fit through...View More
- The color of the product may vary slightly compared to the picture displaye
d on your screen. This is due to lighting, pixel quality and color settings -
Please check the product's dimensions to ensure the product will fit in the d
esired location. Also, check if the product will fit through the entrance(s)
and door(s) of the premises - Please expect an unevenness of up to 5 mm in th
```

In [55]:
```
category_list = list(category_values_dict.keys())
```

In [56]:
```
df_keys_values['Specification_values'] = values
```

In [57]:
```
df_keys_values.tail(2)
```

Out[57]:

	Category	Specification_keys	Specification_values
264	Sunglasses"	[Style, Style Code, Ideal For, Usage, Size, Co...	[Wayfarer, BLKREDGRADGRY, Men, Women, Eye Prot...
265	eBooks	[Publisher, Publication Date, ISBN, File Size,...	[Elsevier Science, 2012-12-31, 9780123947888, ...

In [58]:
```
for row_index,row in df_keys_values.iterrows():
    df_keys_values['Specification_values'][row_index] = category_values_dict[row[
```

In [59]:
```
df_keys_values.head()
```

Out[59]:

	Category	Specification_keys	Specification_values
0	Clothing	[Pattern, Pattern, Occasion, Ideal For, Sleeve...	[Solid, Solid, Casual, Women's, Full Sleeve, C...
1	Furniture	[Foldable, Brand, Delivery Condition, Type, St...	[1 Kids Table 1 Kids Chair, The color of some ...
2	Footwear	[Occasion, Ideal For, Type, Heel Height, Outer...	[Party, Women, Heels, 3 inch, Synthetic Leathe...
3	Pet Supplies	[Pet Type, Body Material, Brand, Shape, Suitab...	[Bird, Stainless Steel, Pawzone, Round, Adult,...
4	Eternal Gandhi Super Series Crystal Paper Weig...	[Model Name, Weight, Paper Weight Material, Pa...	[Gandhi Paper Weight Mark V, 323 g, Paper Weig...

In [60]:
```
df_keys_values.to_csv('flipkart_keys_values.csv',encoding='utf-8')
```

Applying product specifications values to all the products

In [61]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 2 columns):
product_category         19441 non-null object
product_specs_refined    19441 non-null object
dtypes: object(2)
memory usage: 151.9+ KB
```

In [62]:
```
df['Product_specifications_values'] = np.random.random(19441)
```

In [63]:
```
for row_index,row in df.iterrows():
    try:
        df['Product_specifications_values'][row_index] = getvalues(row['product_s
    except ValueError:
        df['Product_specifications_values'][row_index] = 'No Values'
```

```
C:\Users\h_asp\Anaconda3\lib\site-packages\ipykernel\__main__.py:3: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  app.launch_new_instance()
C:\Users\h_asp\Anaconda3\lib\site-packages\pandas\core\indexing.py:179: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  self._setitem_with_indexer(indexer, value)
```

In [64]:
```
df.head()
```

Out[64]:

	product_category	product_specs_refined	Product_specifications_values
0	Clothing	{"product_specification"=> [{"key"=>"Number of ...	[Pack of 3, Cotton Lycra, Cycling Shorts, Soli...
1	Furniture	{"product_specification"=> [{"key"=>"Installati...	[Installation and demo for this product is don...
2	Footwear	{"product_specification"=>[{"key"=>"Ideal For"...	[Women, Casual, Red, Patent Leather, 1 inch, P...
3	Clothing	{"product_specification"=> [{"key"=>"Number of ...	[Pack of 2, Cotton Lycra, Cycling Shorts, Soli...
4	Pet Supplies	{"product_specification"=>[{"key"=>"Pet Type",...	[Dog, Sicons, 500 ml, SH.DF-14, All Purpose, A...

In [65]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 3 columns):
product_category                19441 non-null object
product_specs_refined           19441 non-null object
Product_specifications_values   19441 non-null object
dtypes: object(3)
memory usage: 227.9+ KB
```

In [66]:
```
#df.drop(['product_specs_refined'],axis=1,inplace=True)
```

In [67]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 3 columns):
product_category                 19441 non-null object
product_specs_refined            19441 non-null object
Product_specifications_values    19441 non-null object
dtypes: object(3)
memory usage: 227.9+ KB
```

In [68]:
```
df.to_csv('flipkart_spec_values_category_2.csv',encoding='utf-8')
```

In [69]:
```
#for row_index,row in df.iterrows():
#    try:
#        df['Product_specifications_values'][row_index] = getvalues(row['product_
#    except ValueError:
#        df['Product_specifications_values'][row_index] = 'No Values'
```

In [70]:
```
#df_values = pd.read_csv('flipkart_final_cleared.csv')
```

In [71]:
```
#df_values['specifications_values'] = np.random.random(19441)
```

In [73]:
```
#for row_index,row in df_values.iterrows():
#    try:
#        df_values['specifications_values'][row_index] = getvalues(row['product_sp
#    except ValueError:
#        df_values['specifications_values'][row_index] = 'No Values'
```

In [ ]:
```

```

Flipkart Analysis Project

For any particular product category, is there an association rule between some of the product specification values and the product rating?

Project made by Kaushik,Mukund,Rajdeep,Salini , Batch 3, Globsyn Machine Learning with Python Summer School 2018

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('flipkart_spec_values_category_2.csv')
```

In [3]:
```
#dataset = dataset['product_specifications_values']
df.head()
```

Out[3]:

	Unnamed: 0	product_category	product_specs_refined	Product_specifications_values
0	0	Clothing	{"product_specification"=> [{"key"=>"Number of ...	['Pack of 3', 'Cotton Lycra', 'Cycling Shorts'...
1	1	Furniture	{"product_specification"=> [{"key"=>"Installati...	['Installation and demo for this product is do...
2	2	Footwear	{"product_specification"=> [{"key"=>"Ideal For"...	['Women', 'Casual', 'Red', 'Patent Leather', '...
3	3	Clothing	{"product_specification"=> [{"key"=>"Number of ...	['Pack of 2', 'Cotton Lycra', 'Cycling Shorts'...
4	4	Pet Supplies	{"product_specification"=> [{"key"=>"Pet Type",...	['Dog', 'Sicons', '500 ml', 'SH.DF-14', 'All P...

In [4]:
```
ind = df.columns
```

In [5]:
```
ind = list(ind)
```

In [6]:
```
ind[0] = 'notreq'
```

In [7]:
```
df.columns = ind
```

In [8]:
```
df.drop(['notreq'],axis=1,inplace=True)
```

In [9]:
```
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 3 columns):
product_category                19441 non-null object
product_specs_refined           19441 non-null object
Product_specifications_values   19441 non-null object
dtypes: object(3)
memory usage: 227.9+ KB
```

In [10]:
```
type(df['product_category'][0])
```

Out[10]:
```
str
```

In [11]:
```
df['product_category'][0]
```

Out[11]:
```
'Clothing '
```

In [12]:
```
def trimstrings(x):
    return x.strip()
```

In [13]:
```
df['product_category'] = df['product_category'].apply(trimstrings)
```

In [14]:
```
df_clothing = df[df['product_category'] == 'Clothing']
```

In [15]:
```
df_clothing.head()
```

Out[15]:

	product_category	product_specs_refined	Product_specifications_values
0	Clothing	{"product_specification"=> [{"key"=>"Number of ...	['Pack of 3', 'Cotton Lycra', 'Cycling Shorts'...
3	Clothing	{"product_specification"=> [{"key"=>"Number of ...	['Pack of 2', 'Cotton Lycra', 'Cycling Shorts'...
6	Clothing	{"product_specification"=> [{"key"=>"Number of ...	['Pack of 4', 'Cotton Lycra', 'Cycling Shorts'...
9	Clothing	{"product_specification"=> [{"key"=>"Number of ...	['Pack of 4', 'Cotton Lycra', 'Cycling Shorts'...
11	Clothing	{"product_specification"=>[{"key"=>"Neck", "va...	['Round Neck', 'SwimLycra', 'Swim-dress', 'Pri...

In [16]:
```
#clothing_double_array = list(df_clothing['Product_specifications_values'])
```

In [17]:
```
#type(df_clothing['Product_specifications_values'][0])
```

In [18]:
```
#def convertlist(x):
    #return list(x)
```

In [19]:
```
#df_clothing['Product_specifications_values'] = df_clothing['Product_specificatio
```

In [20]:
```
#clothing_double_array = list(df_clothing['Product_specifications_values'])
```

In [21]:
```
#clothing_double_array
```

In [22]:
```
'''
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(clothing_double_array).transform(clothing_double_array)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
'''
```

Out[22]:
```
'\nimport pandas as pd\nfrom mlxtend.preprocessing import TransactionEncoder\n
\nte = TransactionEncoder()\nte_ary = te.fit(clothing_double_array).transform(c
lothing_double_array)\ndf = pd.DataFrame(te_ary, columns=te.columns_)\ndf\n'
```

In [23]:
```
'''from mlxtend.frequent_patterns import apriori

apriori(df, min_support=0)
'''
```

Out[23]:
```
'from mlxtend.frequent_patterns import apriori\n\napriori(df, min_support=0)\n'
```

In [24]:
```
value_total_list = []
```

In [25]:
```
def getvalues(x):
    values = []
    value_count = x.count('value')
    for i in range(value_count):
        m = x.index('value')
        x = x[m+9:]
        m = x.index('"')
        y = x[:m]
        x = x[m:]
        values.append(y)
    return values
```

In [26]:
```
for row_index,row in df_clothing.iterrows():
    value_list = []
    try:
        value_list = getvalues(row['product_specs_refined'])
    except ValueError:
        value_list = 'No Values'
    value_total_list.append(value_list)
```

In [27]:
```
value_total_list
```

Out[27]:
```
[['Pack of 3',
  'Cotton Lycra',
  'Cycling Shorts',
  'Solid',
  "Women's",
  'Gentle Machine Wash in Lukewarm Water, Do Not Bleach',
  'ALTHT_3P_21',
  '3 shorts'],
 ['Pack of 2',
  'Cotton Lycra',
  'Cycling Shorts',
  'Solid',
  "Women's",
  'Gentle Machine Wash in Lukewarm Water, Do Not Bleach',
  'ALTGHT_11',
  '2 shorts'],
 ['Pack of 4',
  'Cotton Lycra',
  'Cycling Shorts',
```

In [28]:
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(value_total_list).transform(value_total_list)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

Out[28]:

	# 61-0288-001	#ZWSHRT006	(5 Pockets) 2 Cross Pockets in front, 2 Bone Pockets at back and 1 Watch Pocket in front	- Wash with mild detergent - Do Not Brush - Dry in shadows	0	0 - 0 month	0 - 6 Months	0 carat	0 g	...	t
0	False	False	False	False	False	False	False	False	False	False	...
1	False	False	False	False	False	False	False	False	False	False	...
2	False	False	False	False	False	False	False	False	False	False	...
3	False	False	False	False	False	False	False	False	False	False	...
4	False	False	False	False	False	False	False	False	False	False	...

In [29]:
```
from mlxtend.frequent_patterns import apriori

apriori(df, min_support=0.6)
```

Out[29]:

	support	itemsets
0	0.628082	[1993]
1	0.676027	[5129]
2	0.609247	[6936]

In [30]:
```
apriori(df, min_support=0.6, use_colnames=True)
```

Out[30]:

	support	itemsets
0	0.628082	[Casual]
1	0.676027	[Pack of 1]
2	0.609247	[Women's]

In [31]:
```
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x
frequent_itemsets
```

Out[31]:

	support	itemsets	length
0	0.628082	[Casual]	1
1	0.676027	[Pack of 1]	1
2	0.609247	[Women's]	1

In [4]:
```
#oht_ary = te.fit(value_total_list).transform(value_total_list, sparse=True)
#sparse_df = pd.SparseDataFrame(te_ary, columns=te.columns_, default_fill_value=F
#sparse_df
```

In [3]:
```
#apriori(sparse_df, min_support=0.6, use_colnames=True)
```

In [ ]:
```
frequent_itemsets[ (frequent_itemsets['length'] == 1) &
                   (frequent_itemsets['support'] >= 0.6) ]
```

Apriori Algorithm predicted that Pack of 1 in clothing is the most favourable choice of the companies for selling.

In [34]:
```
frequent_itemsets['support'].max()
```

Out[34]: 0.6760273972602739

In [35]:
```
frequent_itemsets[frequent_itemsets['support'] == frequent_itemsets['support'].ma
```

Out[35]:
```
1    [Pack of 1]
Name: itemsets, dtype: object
```

In [42]:
```
df_cleared = pd.read_csv('flipkart_final_cleared.csv')
df_cleared.head()
```

Out[42]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

In [46]:
```
df_clothing_with_rating = df_cleared[df_cleared['product_category']=='Clothing ']
```

In [66]:
```
value_list_with_rating = []
for row_index,row in df_clothing_with_rating.iterrows():
    value_list = []
    try:
        value_list = getvalues(row['product_specs_refined'])
    except ValueError:
        value_list = ['No Values']
    value_list.append(str(int(row['product_rating'])))
    value_list_with_rating.append(value_list)
```

In [67]:
```
value_list_with_rating[-4]
```

Out[67]:
```
['Solid',
 'Casual',
 "Women's",
 '3/4 Sleeve',
 'Pack of 1',
 'Slim',
 'Georgette',
 'Slim',
 'STARSAT-PICH',
 '5']
```

In [68]:
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(value_list_with_rating).transform(value_list_with_rating)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

Out[68]:

	# 61-0288-001	#ZWSHRT006	(5 Pockets) 2 Cross Pockets in front, 2 Bone Pockets at back and 1 Watch Pocket in front	- Wash with mild detergent - Do Not Brush - Dry in shadows	0	0 - 0 month	0 - 6 Months	0 carat	0 g	0.4 kg	...	t
0	False	False	False	False	False	False	False	False	False	False	...	
1	False	False	False	False	False	False	False	False	False	False	...	
2	False	False	False	False	False	False	False	False	False	False	...	
3	False	False	False	False	False	False	False	False	False	False	...	
4	False	False	False	False	True	False	False	False	False	False	...	

In [71]:
```
from mlxtend.frequent_patterns import apriori

apriori(df, min_support=0.4)
```

Out[71]:

	support	itemsets
0	0.628082	[1993]
1	0.676027	[5129]
2	0.425342	[6146]
3	0.609247	[6935]
4	0.463527	[1993, 5129]
5	0.430479	[5129, 6935]

In [75]:
```
apriori(df, min_support=0.4, use_colnames=True)
```

Out[75]:

	support	itemsets
0	0.628082	[Casual]
1	0.676027	[Pack of 1]
2	0.425342	[Solid]
3	0.609247	[Women's]
4	0.463527	[Casual, Pack of 1]
5	0.430479	[Pack of 1, Women's]

In [77]:
```
apriori(df, min_support=0.4, use_colnames=True)
```

Out[77]:

	support	itemsets
0	0.628082	[Casual]
1	0.676027	[Pack of 1]
2	0.425342	[Solid]
3	0.609247	[Women's]
4	0.463527	[Casual, Pack of 1]
5	0.430479	[Pack of 1, Women's]

In [81]:
```
frequent_itemsets = apriori(df, min_support=0.1, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x
frequent_itemsets
```

Out[81]:

	support	itemsets	length
0	0.148288	[0]	1
1	0.174315	[1]	1
2	0.155822	[2]	1
3	0.177397	[3]	1
4	0.176712	[4]	1
5	0.181336	[5]	1
6	0.628082	[Casual]	1
7	0.392466	[Cotton]	1
8	0.172432	[Full Sleeve]	1
9	0.175514	[Half Sleeve]	1
10	0.279623	[Men's]	1
11	0.166610	[No]	1
12	0.676027	[Pack of 1]	1
13	0.220548	[Printed]	1
14	0.364041	[Regular]	1
15	0.211815	[Round Neck]	1
16	0.128596	[Sleeveless]	1
17	0.425342	[Solid]	1
18	0.137671	[Wirefree]	1
19	0.609247	[Women's]	1
20	0.100171	[0, Pack of 1]	2
21	0.106678	[1, Casual]	2
22	0.114212	[1, Pack of 1]	2
23	0.104966	[1, Women's]	2
24	0.105993	[2, Pack of 1]	2
25	0.113185	[3, Casual]	2
26	0.116438	[3, Pack of 1]	2
27	0.110616	[3, Women's]	2
28	0.108219	[4, Casual]	2
29	0.118836	[4, Pack of 1]	2
...	...	...	...
72	0.112671	[Casual, Cotton, Men's]	3
73	0.203425	[Casual, Cotton, Pack of 1]	3

	support	itemsets	length
74	0.144349	[Casual, Cotton, Regular]	3
75	0.115582	[Casual, Cotton, Solid]	3
76	0.152740	[Casual, Cotton, Women's]	3
77	0.115753	[Casual, Half Sleeve, Men's]	3
78	0.126027	[Casual, Half Sleeve, Pack of 1]	3
79	0.153082	[Casual, Men's, Pack of 1]	3
80	0.130822	[Casual, Pack of 1, Printed]	3
81	0.207363	[Casual, Pack of 1, Regular]	3
82	0.144178	[Casual, Pack of 1, Round Neck]	3
83	0.187158	[Casual, Pack of 1, Solid]	3
84	0.292637	[Casual, Pack of 1, Women's]	3
85	0.121233	[Casual, Regular, Solid]	3
86	0.174315	[Casual, Regular, Women's]	3
87	0.178767	[Casual, Solid, Women's]	3
88	0.115753	[Cotton, Men's, Pack of 1]	3
89	0.133904	[Cotton, Pack of 1, Regular]	3
90	0.146918	[Cotton, Pack of 1, Women's]	3
91	0.102226	[Cotton, Regular, Women's]	3
92	0.110103	[Half Sleeve, Men's, Pack of 1]	3
93	0.117123	[Pack of 1, Regular, Solid]	3
94	0.175171	[Pack of 1, Regular, Women's]	3
95	0.189897	[Pack of 1, Solid, Women's]	3
96	0.113356	[Regular, Solid, Women's]	3
97	0.114041	[Regular, Wirefree, Women's]	3
98	0.102055	[Casual, Cotton, Pack of 1, Regular]	4
99	0.104452	[Casual, Cotton, Pack of 1, Women's]	4
100	0.126712	[Casual, Pack of 1, Regular, Women's]	4
101	0.125171	[Casual, Pack of 1, Solid, Women's]	4

102 rows × 3 columns

In [86]:
```
frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                   (frequent_itemsets['support'] >= 0.4) ]
```

Out[86]:

	support	itemsets	length
39	0.463527	[Casual, Pack of 1]	2
63	0.430479	[Pack of 1, Women's]	2

This shows that the casual pack of 1 selling is the most frequent choice of flipkart

In [112]:
```
itemsets = frequent_itemsets[ (frequent_itemsets['length'] == 2) &
                   (frequent_itemsets['support'] >= 0.1) ]
```

In [117]:
```
type(itemsets['itemsets'])
```

Out[117]:
```
pandas.core.series.Series
```

In [118]:
```
def to_string(x):
    return str(x)
```

In [119]:
```
itemsets['itemsets'].apply(to_string)
```

Out[119]:
```
20             ['0', 'Pack of 1']
21                ['1', 'Casual']
22             ['1', 'Pack of 1']
23               ['1', "Women's"]
24             ['2', 'Pack of 1']
25                ['3', 'Casual']
26             ['3', 'Pack of 1']
27               ['3', "Women's"]
28                ['4', 'Casual']
29             ['4', 'Pack of 1']
30               ['4', "Women's"]
31                ['5', 'Casual']
32             ['5', 'Pack of 1']
33               ['5', "Women's"]
34           ['Casual', 'Cotton']
35      ['Casual', 'Full Sleeve']
36      ['Casual', 'Half Sleeve']
37            ['Casual', "Men's"]
38               ['Casual', 'No']
39        ['Casual', 'Pack of 1']
40          ['Casual', 'Printed']
41          ['Casual', 'Regular']
42       ['Casual', 'Round Neck']
43            ['Casual', 'Solid']
44          ['Casual', "Women's"]
45            ['Cotton', "Men's"]
46        ['Cotton', 'Pack of 1']
47          ['Cotton', 'Printed']
48          ['Cotton', 'Regular']
49            ['Cotton', 'Solid']
50          ['Cotton', "Women's"]
51   ['Full Sleeve', 'Pack of 1']
52       ['Half Sleeve', "Men's"]
53   ['Half Sleeve', 'Pack of 1']
55         ["Men's", 'Pack of 1']
57             ["Men's", 'Solid']
59       ['Pack of 1', 'Printed']
61    ['Pack of 1', 'Round Neck']
63       ['Pack of 1', "Women's"]
65           ['Regular', 'Solid']
67         ['Regular', "Women's"]
69      ['Sleeveless', "Women's"]
71        ['Wirefree', "Women's"]
Name: itemsets, dtype: object
```

In [120]:
```
itemsets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 43 entries, 20 to 71
Data columns (total 3 columns):
support     43 non-null float64
itemsets    43 non-null object
length      43 non-null int64
dtypes: float64(1), int64(1), object(1)
memory usage: 2.2+ KB
```

In [151]:
```
rule_list_total = []
for row_index,row in itemsets.iterrows():
    rule_list = []
    if row['itemsets'][0] in ['0','1','2','3','4','5']:
        rule_list.append(str(row['itemsets']))
        rule_list.append(row['support'])
        rule_list_total.append(rule_list)
```

In [152]:
```
rule_list_total
```

Out[152]:
```
[["['0', 'Pack of 1']", 0.10017123287671233],
 ["['1', 'Casual']", 0.10667808219178082],
 ["['1', 'Pack of 1']", 0.11421232876712328],
 ['[\'1\', "Women\'s"]', 0.10496575342465754],
 ["['2', 'Pack of 1']", 0.10599315068493151],
 ["['3', 'Casual']", 0.11318493150684932],
 ["['3', 'Pack of 1']", 0.11643835616438356],
 ['[\'3\', "Women\'s"]', 0.11061643835616439],
 ["['4', 'Casual']", 0.10821917808219178],
 ["['4', 'Pack of 1']", 0.11883561643835616],
 ['[\'4\', "Women\'s"]', 0.10308219178082192],
 ["['5', 'Casual']", 0.12037671232876712],
 ["['5', 'Pack of 1']", 0.12465753424657534],
 ['[\'5\', "Women\'s"]', 0.11044520547945205]]
```

In [157]:
```
rule_df_clothing = pd.DataFrame(data=rule_list_total,columns=['rating_specificati
```

In [158]:
```
rule_df_clothing.head()
```

Out[158]:

	rating_specificationvalue	support
0	['0', 'Pack of 1']	0.100171
1	['1', 'Casual']	0.106678
2	['1', 'Pack of 1']	0.114212
3	['1', "Women's"]	0.104966
4	['2', 'Pack of 1']	0.105993

In [160]:
```
rule_df_clothing.max()
```

Out[160]:
```
rating_specificationvalue    ['5', 'Pack of 1']
support                                0.124658
dtype: object
```

Pack of 1 with a rating of 5 gets a support value of 0.124658

In [ ]:
```

```

Globsyn Summer School 2018, Machine Learning Using Python, a project on analysis of flipkart data, by Kaushik, Mukund, Rajdeep and Salini

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
%matplotlib inline
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [2]:
```
# Load the Dataset

df=pd.read_csv('flipkart_final_cleared.csv')
df.head()
```

Out[2]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

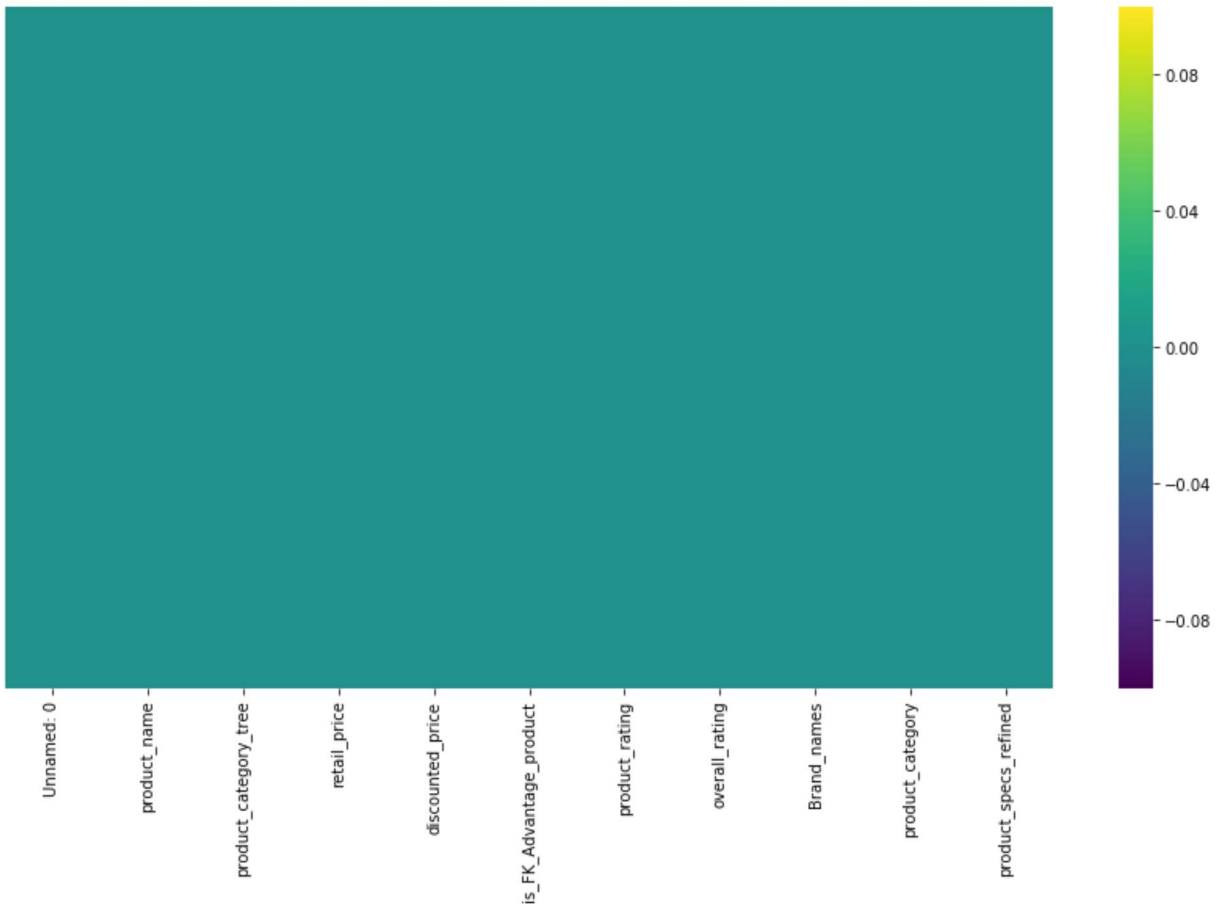
In [3]:
```
# Information of the Datset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19441 entries, 0 to 19440
Data columns (total 11 columns):
Unnamed: 0                  19441 non-null int64
product_name                19441 non-null object
product_category_tree       19441 non-null object
retail_price                19441 non-null float64
discounted_price            19441 non-null float64
is_FK_Advantage_product     19441 non-null bool
product_rating              19441 non-null float64
overall_rating              19441 non-null float64
Brand_names                 19441 non-null object
product_category            19441 non-null object
product_specs_refined       19441 non-null object
dtypes: bool(1), float64(4), int64(1), object(5)
memory usage: 1.1+ MB
```

In [4]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.isnull(),yticklabels=False,cbar=True,cmap='viridis')
```

Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0xfc67bb0>



In [5]:
```
df["discount_percent"] = ((df.retail_price - df.discounted_price)*100)/df.retail_
df.head()
```

Out[5]:

	Unnamed: 0	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantag
0	0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	
1	1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	
2	2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	
3	3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	
4	4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	

In [6]:
```
df[['product_name','Brand_names','product_rating','retail_price','discounted_pric
```

Out[6]:

	product_name	Brand_names	product_rating	retail_price	discounted_price	discount_percent
0	Alisha Solid Women's Cycling Shorts	Alisha	5.0	999.0	379.0	62.062062
1	FabHomeDecor Fabric Double Sofa Bed	FabHomeDecor	2.0	32157.0	22646.0	29.576764
2	AW Bellies	AW	4.0	999.0	499.0	50.050050
3	Alisha Solid Women's Cycling Shorts	Alisha	2.0	699.0	267.0	61.802575
4	Sicons All Purpose Arnica Dog Shampoo	Sicons	1.0	220.0	210.0	4.545455

In [7]:
```
number = LabelEncoder()

df['Brand_names'] = number.fit_transform(df['Brand_names'])

df['product_rating'] = number.fit_transform(df['product_rating'])

df['discount_percent'] = number.fit_transform(df['discount_percent'])
```

In [8]:
```
features = df[["Brand_names", "product_rating"]]
target = df["discount_percent"]
```

In [9]:
```
features_train, features_test, target_train, target_test = train_test_split(featu
```

In [10]:
```
model = GaussianNB()
model.fit(features_train, target_train)
```

Out[10]:
```
GaussianNB(priors=None)
```

In [11]:
```
G_pred = model.predict(features_test)
accuracy = accuracy_score(target_test, G_pred)
print("Accuracy = ",accuracy)
```

```
Accuracy =  0.1059850374064838
```

In [12]:
```
knn=KNeighborsClassifier(n_neighbors=2)
```

In [13]:
```
knn.fit(features_train, target_train)
```

Out[13]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=2, p=2,
           weights='uniform')
```

In [14]:
```
knn_pred = knn.predict(features_test)
```

In [15]:
```
print(confusion_matrix(target_test, knn_pred))
```

```
[[382   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 ...
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]]
```

In [16]:
```
print(classification_report(target_test, knn_pred))
```

```
             precision    recall  f1-score   support

          0       0.33      0.53      0.40       721
          1       0.00      0.00      0.00         1
          2       0.00      0.00      0.00         1
          7       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         0
          9       0.00      0.00      0.00         1
         10       0.00      0.00      0.00         1
         12       0.00      0.00      0.00         0
         15       0.00      0.00      0.00         1
         16       0.00      0.00      0.00         1
         18       0.00      0.00      0.00         1
         19       0.00      0.00      0.00         0
         20       0.00      0.00      0.00         2
         23       0.00      0.00      0.00         1
         26       0.00      0.00      0.00         1
         27       0.00      0.00      0.00         0
         29       0.00      0.00      0.00         1
         30       0.00      0.00      0.00         1
```

In [17]:
```
regr = linear_model.LinearRegression()
```

In [18]:
```
regr.fit(features_train, target_train)
```

Out[18]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [19]:
```
L_pred = regr.predict(features_test)
```

In [20]:
```
coeff_df=pd.DataFrame(regr.coef_,features.columns,columns=['Coefficient'])
coeff_df#the increase of a parameter
```

Out[20]:

	Coefficient
Brand_names	-0.023526
product_rating	-0.509280

In [21]:
```
print(regr.intercept_)
```

```
2944.2594306069673
```

In [22]:
```
from sklearn.metrics import mean_squared_error, r2_score
```

In [23]:
```
print("Mean squared error: %.2f"% mean_squared_error(target_test, L_pred))
```

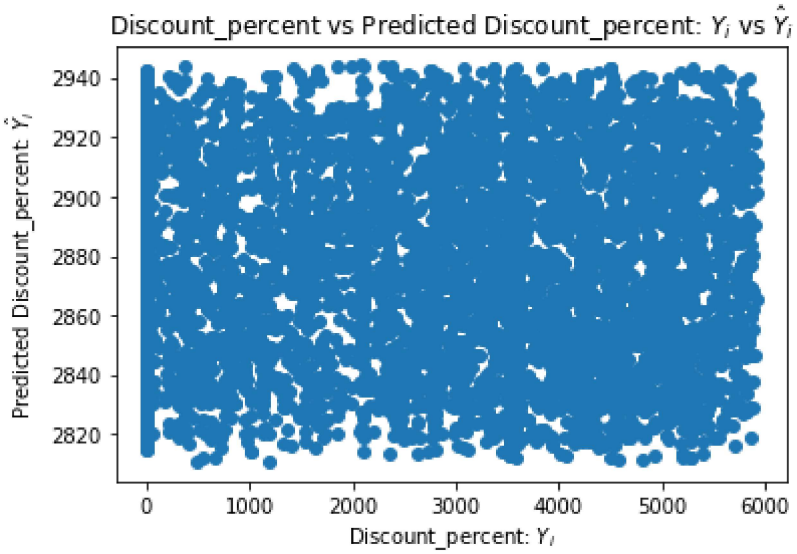
```
Mean squared error: 3304009.12
```

In [24]:
```
print('Variance score: %.2f' % r2_score(target_test, L_pred))
```

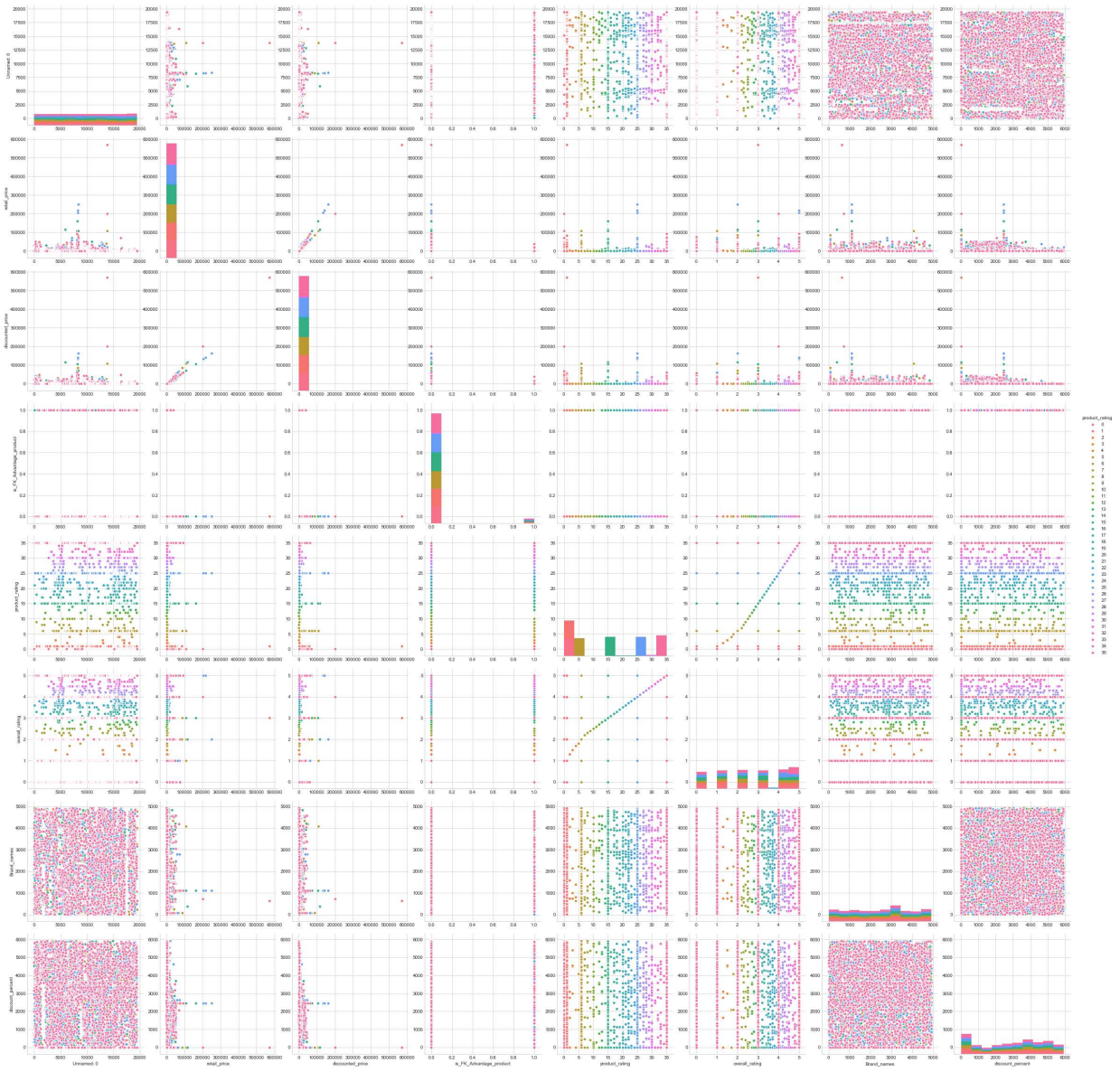
```
Variance score: -0.00
```

In [25]:
```
plt.scatter(target_test, L_pred)
plt.xlabel("Discount_percent: $Y_i$")
plt.ylabel("Predicted Discount_percent: $\hat{Y}_i$")
plt.title("Discount_percent vs Predicted Discount_percent: $Y_i$ vs $\hat{Y}_i$")
```

Out[25]:
```
Text(0.5,1,'Discount_percent vs Predicted Discount_percent: $Y_i$ vs $\\hat{Y}_
i$')
```

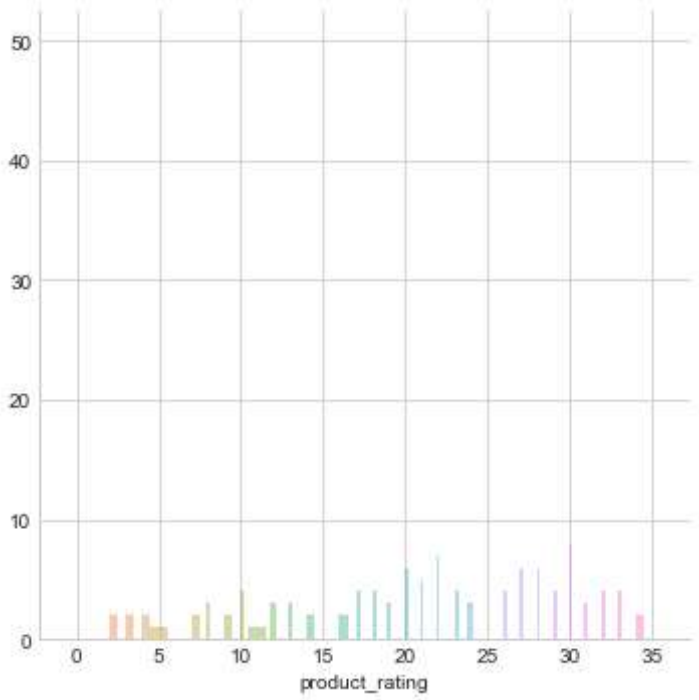


In [26]:
```
plt.close()
sns.set_style("whitegrid");
sns.pairplot(df,hue="product_rating",size=4);
plt.show()
```

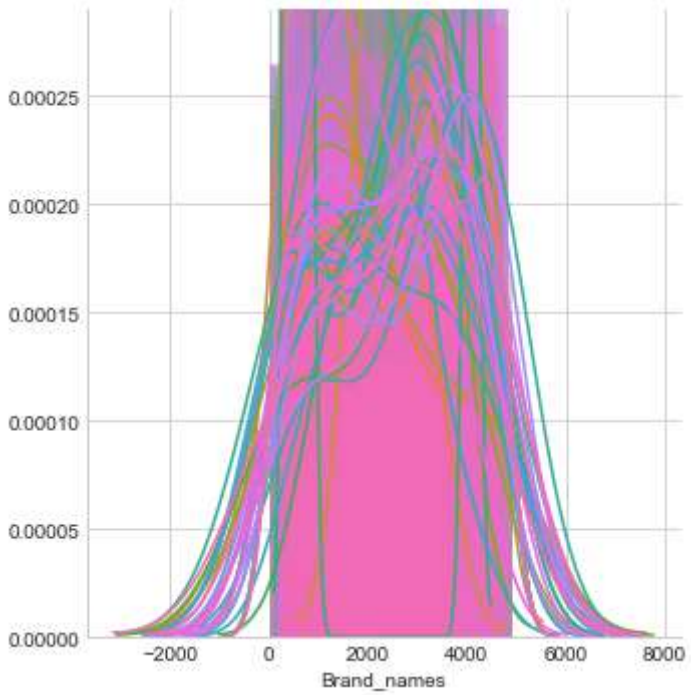


In [30]:
```
sns.FacetGrid(df,hue="product_rating",size=5) \
.map(sns.distplot,"product_rating")
plt.show();
```

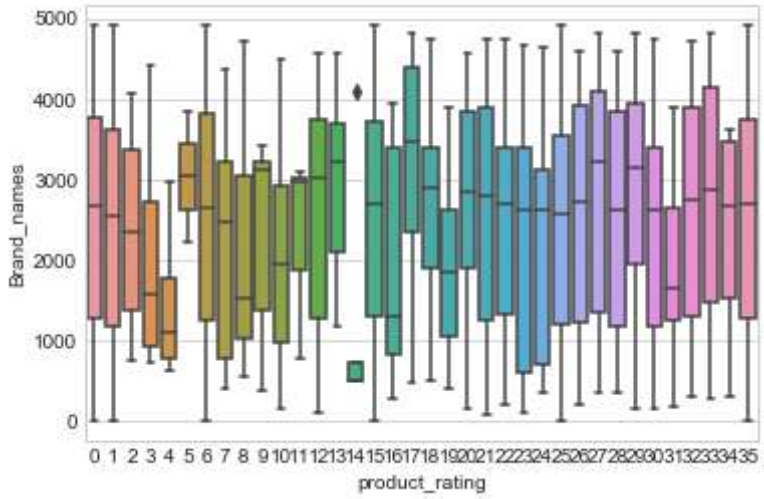
```
C:\Users\h_asp\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:49
4: RuntimeWarning: invalid value encountered in true_divide
  binned = fast_linbin(X,a,b,gridsize)/(delta*nobs)
C:\Users\h_asp\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
y:34: RuntimeWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
C:\Users\h_asp\Anaconda3\lib\site-packages\numpy\core\_methods.py:26: RuntimeWa
rning: invalid value encountered in reduce
  return umr_maximum(a, axis, None, out, keepdims)
```



In [29]:
```
sns.FacetGrid(df,hue="product_rating",size=5) \
.map(sns.distplot,"Brand_names")
plt.show();
```

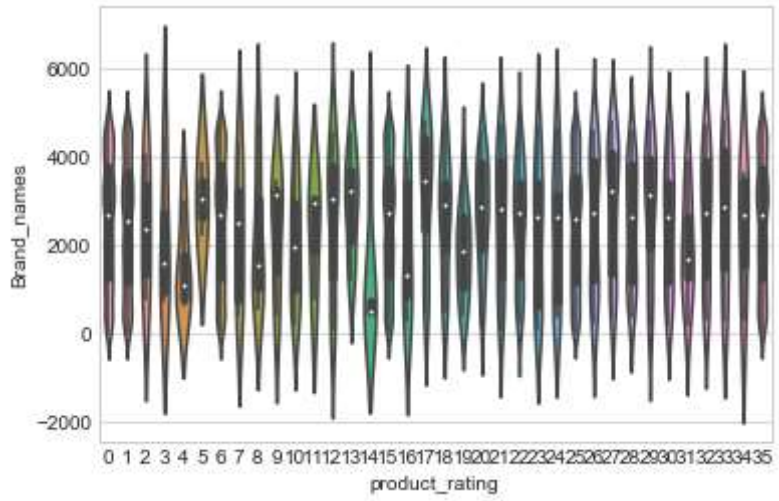


In [34]:
```
sns.boxplot(x='product_rating',y='Brand_names',data=df)
plt.show()
```



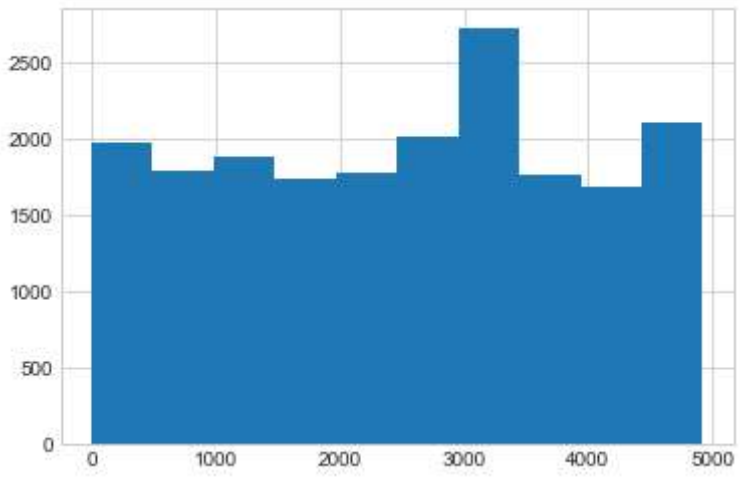
In [32]:
```
sns.set_style("whitegrid")
sns.violinplot(x='product_rating',y='Brand_names',data=df)
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2379bbd0>
```



In [33]:
```
plt.hist(x='Brand_names',data=df)
```

Out[33]:
```
(array([1976., 1795., 1879., 1737., 1772., 2010., 2718., 1765., 1689.,
        2100.]),
 array([   0. ,  492.2,  984.4, 1476.6, 1968.8, 2461. , 2953.2, 3445.4,
        3937.6, 4429.8, 4922. ]),
 <a list of 10 Patch objects>)
```



So, we see that using Knn, we get an accuracy of 0.16