Wish.com Product Rating Prediction

ABOUT THE ASSIGNMENT:-

- The dataset is the wish.com product dataset.
- They collected the data combined with some available data.
- Some noises are added to the dataset.
- The goal is to predict the product ratings given the other features known for a product on Wish.com
- Ratings are in categories from 1 to 5. For one product, the higher the rating is, the more the customers like the product.

PROBLEM FORMULATION

- Define the problem.

The problem is predicting the products rating to gain prior knowledge of customers' preferences for products, which further improves the website's sales process.

What is the input?

Our inputs are a lot of features I will show them next slides

- What is the output?

Rating of the products depend on some features.

- What data mining function is required?

The model is classification model.

PROBLEM FORMULATION

- What could be the challenges?

The challenges that we have a lot of noise in the dataset, and it took a lot of time to make preprocessing on it.

- What is the impact?

The impact is making misleading to the model to predict and classify the right rating for each model.

- What is an ideal solution?

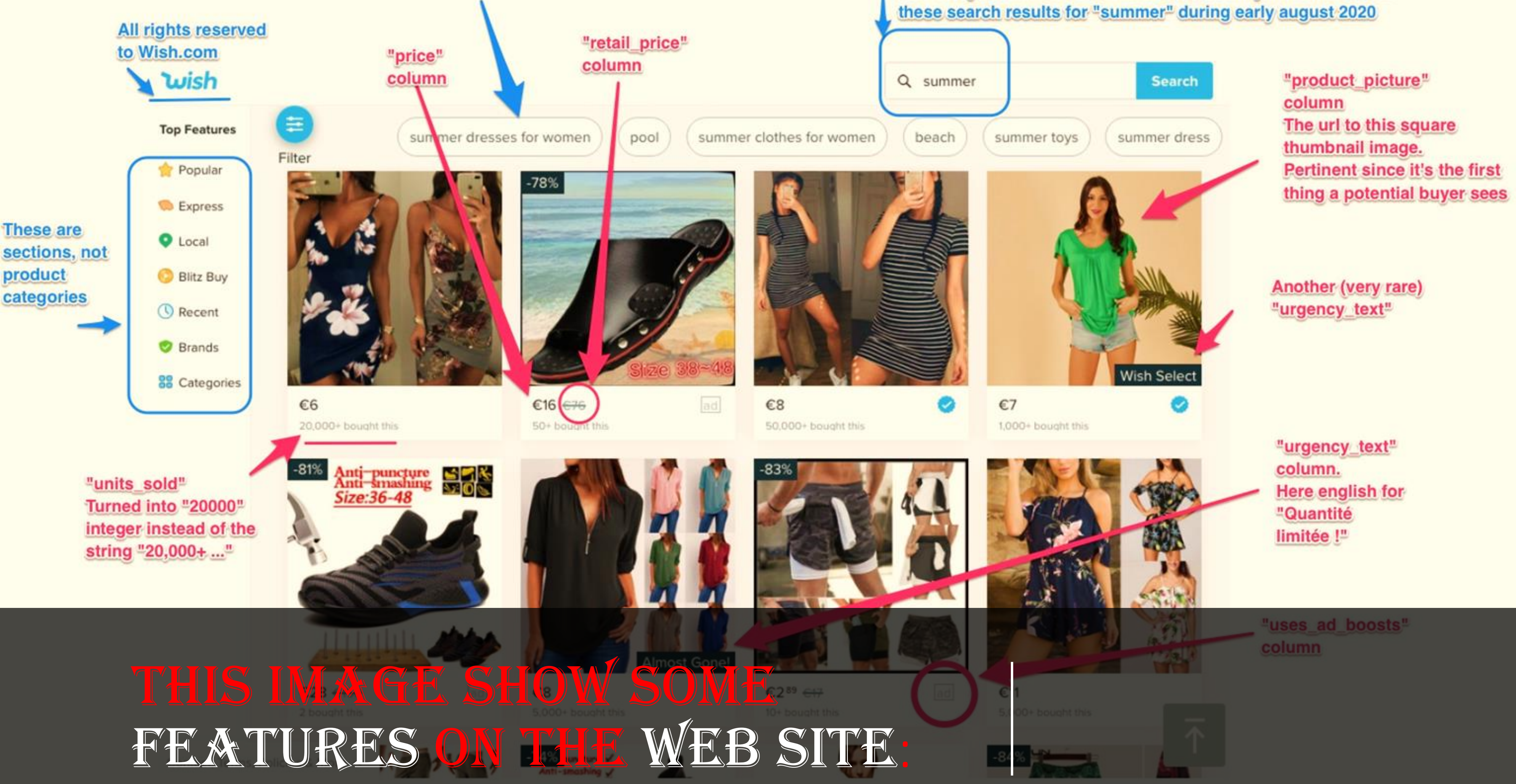
The ideal solution is to make good preprocessing on each column and take the columns that will help the model to predict the right rating

ABOUT THE DATASET:-

- Dataset of this project contain a lot of features:-

```
'price', 'retail_price', 'units_sold',
'uses_ad_boosts', 'rating', 'rating_count',
'badges_count', 'badge_local_product',
'badge_product_quality', 'badge_fast_shipping',
'tags',
'product_color', 'product_variation_size_id',
'product_variation_inventory',
'shipping_option_name', 'shipping_option_price',
'shipping_is_express', 'countries_shipped_to',
'inventory_total', 'origin_country',
'merchant_title', 'merchant_name',
'merchant_info_subtitle',
'merchant_rating_count', 'merchant_rating',
'merchant_has_profile_picture',
'merchant_profile_picture', 'discounted_price'
```

THIS IMAGE SHOW SOME FEATURES ON THE WEB SITE:



SOME FEATURES DESCRIPTION:

retail_price: reference price for similar articles on the market, or in other stores/places. Used by the seller to indicate a regular value or the price before discount.

currency_buyer: currency of the prices.

uses_ad_boosts: Whether the seller paid to boost his product within the platform (highlighting, better placement or whatever).

rating_count: Total number of ratings of the product.

badges_count: Number of badges the product or the seller have.

badge_product_quality: is quality product.

countries_shipped_to: Number of countries this product is shipped to. Sellers may choose to limit where they ship a product to

tags: tags set by the seller

CONT.

product_variation_size_id: One of the available size variation for this product

product_variation_inventory: Inventory the seller has. Max allowed quantity is 50

shipping_option_price: shipping price

shipping_option_price: shipping price

merchant_name: Merchant's canonical name. A name not shown publicly. Used by the website under the hood as a canonical name. Easier to process since all lowercase without white space

merchant_profile_picture: Custom profile picture of the seller (if the seller has one). Empty otherwise.

rating_count: Total number of ratings of the product

has_urgency_banner: whether there was an urgency banner with an urgency

urgency_text: A text banner that appear over some products in the search results.

FIRST LOAD DATASET:

- Load train_new dataset.
- Load test_new dataset.
- Then combine them in one csv file called df

```
train_data=pd.read_csv('train_new.csv')  #Load traind_new Dataset
test_data = pd.read_csv('test_new.csv')  #Load test_new Dataset
test_data['rating'] = 0 #make column called rating in test_new dataset
df = pd.concat([train_data, test_data], ignore_index=True, sort=False) #concatenate the train_new dataset and test_new dataset
                                                                        #and combine them in dataset called df
df.shape #show df shape
```

```
(1573, 34)
```

SOME EXPLORATIONS ON THE DF:

```
df.info()  #show some details on the df using method info()
```

```
 16  shipping_option_price         1573 non-null   int64
 17  shipping_is_express           1573 non-null   int64
 18  countries_shipped_to          1573 non-null   int64
 19  inventory_total               1573 non-null   int64
 20  has_urgency_banner            473 non-null    float64
 21  urgency_text                  473 non-null    object
 22  origin_country                1556 non-null   object
 23  merchant_title                1573 non-null   object
 24  merchant_name                 1569 non-null   object
 25  merchant_info_subtitle        1572 non-null   object
 26  merchant_rating_count         1573 non-null   int64
 27  merchant_rating               1573 non-null   float64
 28  merchant_id                   1573 non-null   object
 29  merchant_has_profile_picture  1573 non-null   int64
 30  merchant_profile_picture      226 non-null    object
 31  theme                         1573 non-null   object
 32  crawl_month                   1573 non-null   object
 33  id                            1573 non-null   int64
dtypes: float64(3), int64(17), object(14)
memory usage: 418.0+ KB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1573 entries, 0 to 1572
Data columns (total 34 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   price                         1573 non-null   float64
 1   retail_price                  1573 non-null   int64
 2   currency_buyer                1573 non-null   object
 3   units_sold                    1573 non-null   int64
 4   uses_ad_boosts                1573 non-null   int64
 5   rating                        1573 non-null   int64
 6   rating_count                  1573 non-null   int64
 7   badges_count                  1573 non-null   int64
 8   badge_local_product           1573 non-null   int64
 9   badge_product_quality         1573 non-null   int64
 10  badge_fast_shipping           1573 non-null   int64
 11  tags                          1573 non-null   object
 12  product_color                 1532 non-null   object
 13  product_variation_size_id     1559 non-null   object
 14  product_variation_inventory   1573 non-null   int64
 15  shipping_option_name          1573 non-null   object
 16  shipping_option_price         1573 non-null   int64
 17  shipping_is_express           1573 non-null   int64
```

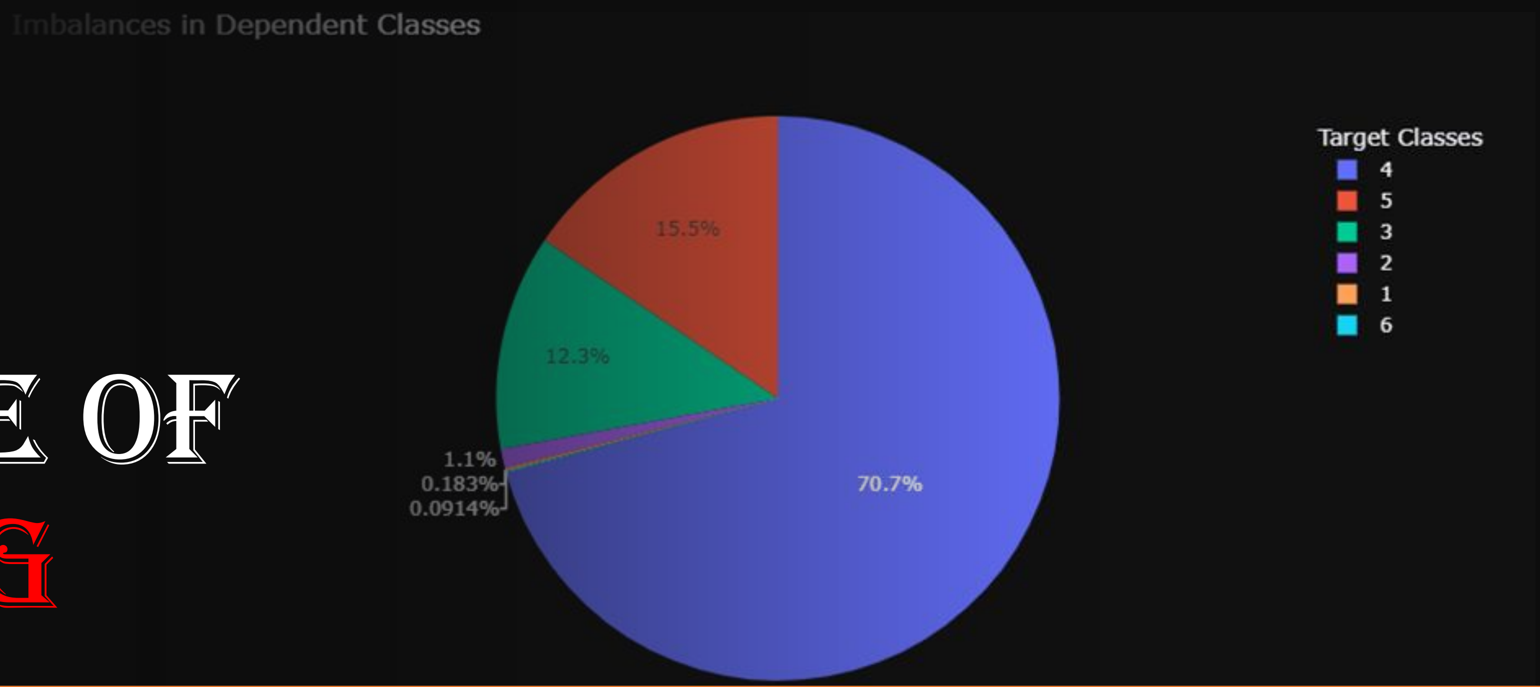
SOME EXPLORATIONS ON THE DF:

```
print('\n*****************check the values of each colomun are unique or not and how many number for each one
print(df.nunique())  #check the values of each colomun are unique or not and how many number for each one
```

```
price                           129
retail_price                    104
currency_buyer                    1
units_sold                       15
uses_ad_boosts                    2
rating                            7
rating_count                    761
badges_count                      4
badge_local_product               2
badge_product_quality             2
badge_fast_shipping               2
tags                           1230
product_color                   101
product_variation_size_id       106
product_variation_inventory      48
shipping_option_name             16
shipping_option_price             8
shipping_is_express               2
countries_shipped_to             94
inventory_total                  10
has_urgency_banner                1
```

```
countries_shipped_to             94
inventory_total                  10
has_urgency_banner                1
urgency_text                      2
origin_country                    6
merchant_title                  958
merchant_name                   957
merchant_info_subtitle         1058
merchant_rating_count           917
merchant_rating                1137
merchant_id                     958
merchant_has_profile_picture      2
merchant_profile_picture        125
theme                             1
crawl_month                       1
id                             1573
dtype: int64
```

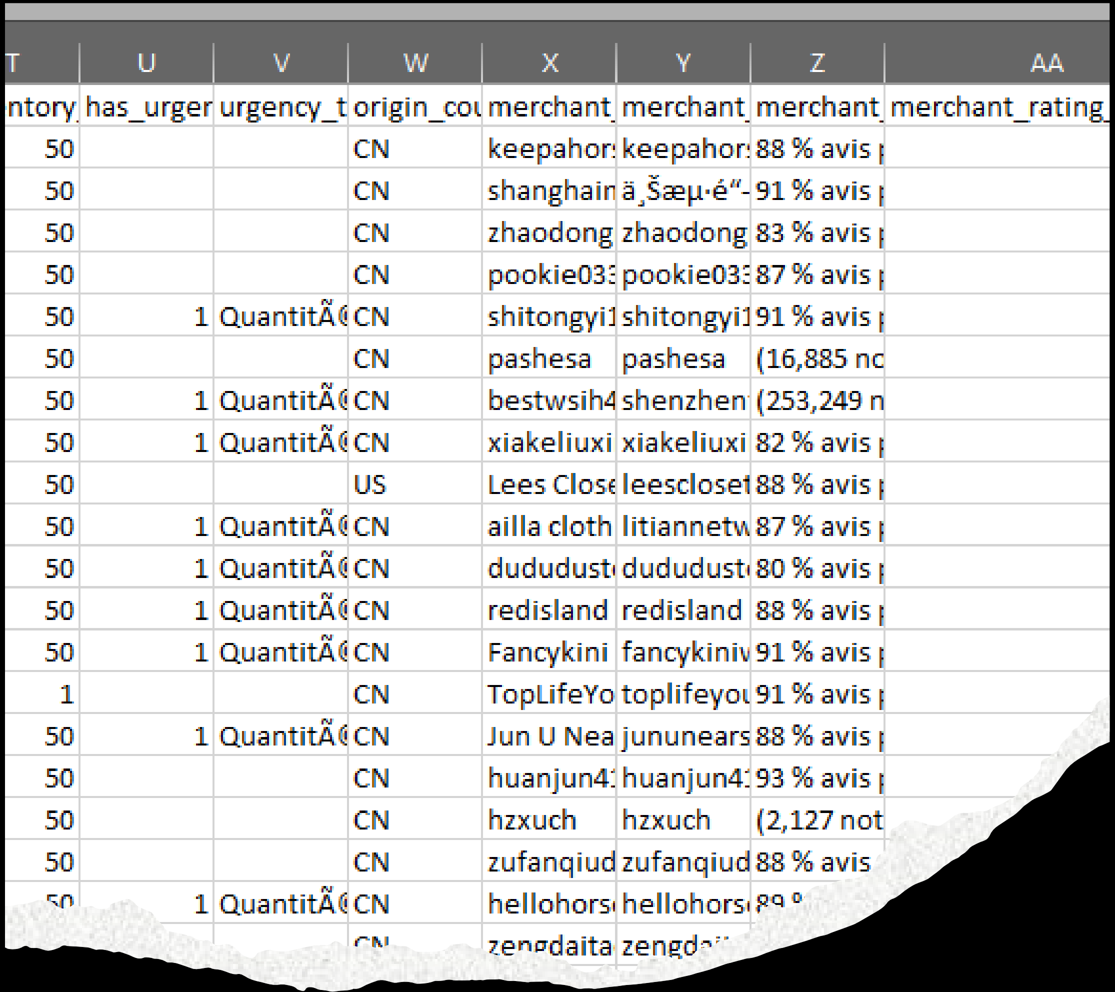
THE PERCENTAGE OF EACH RATING




```
#this show the rating of train dataset so it stell has value equal 6 "this file without any  update"
dependent_classes_labels= train_data.rating.value_counts().index.values
dependent_classes_values = train_data.rating.value_counts().values
fig = go.Figure()
fig.add_trace(go.Pie(labels=dependent_classes_labels, values=dependent_classes_values))
fig.update_layout(title="Imbalances in Dependent Classes", legend_title="Target Classes", template="plotly_dark")
```

NOW WE CAN WORK ON EACH COLUMN:

- For each column has only one value.
- For each column has null values.
- For each column that will be drop.
- For each column that will make preprocessing for it
- And we apply some methods for each column



T	U	V	W	X	Y	Z	AA
ntory_	has_urger	urgency_t	origin_cou	merchant_	merchant_	merchant_	merchant_rating_
50			CN	keepahor	keepahor	88 % avis	
50			CN	shanghair	ä¸Šæµ·é“-	91 % avis	
50			CN	zhaodong	zhaodong	83 % avis	
50			CN	pookie03	pookie03	87 % avis	
50	1	QuantitÃ	CN	shitongyi1	shitongyi1	91 % avis	
50			CN	pashesa	pashesa	(16,885 nc	
50	1	QuantitÃ	CN	bestwsih4	shenzhen	(253,249 n	
50	1	QuantitÃ	CN	xiakeliuxi	xiakeliuxi	82 % avis	
50			US	Lees Close	leescloset	88 % avis	
50	1	QuantitÃ	CN	ailla cloth	litiannetw	87 % avis	
50	1	QuantitÃ	CN	dududust	dududust	80 % avis	
50	1	QuantitÃ	CN	redisland	redisland	88 % avis	
50	1	QuantitÃ	CN	Fancykini	fancykiniv	91 % avis	
1			CN	TopLifeYo	toplifeyou	91 % avis	
50	1	QuantitÃ	CN	Jun U Nea	jununears	88 % avis	
50			CN	huanjun4	huanjun4	93 % avis	
50			CN	hzxuch	hzxuch	(2,127 not	
50			CN	zufanqiud	zufanqiud	88 % avis	
50	1	QuantitÃ	CN	hellohors	hellohors	89 %	
			CN	zengdaita	zengd		

THEME COLUMN:

drop the theme column because it contain only one value

```
df.theme.value_counts() #show the valuse of the column and the number of

summer    1573
Name: theme, dtype: int64

df.drop('theme', axis=1, inplace=True) #in this line i dropped the theme
```

URGENCY TEXT AND URGENCY BANNER:

Both columns have null values, and in very large number so I dropped them.

```
#drop two columns urgency_text','has_urgency_banner

df.drop(['urgency_text','has_urgency_banner'], inplace=True,axis=1)
```

BADGES COLUMNS :

convert 'badge_local_product', 'badge_product_quality', 'badge_fast_shipping' into categorical values.

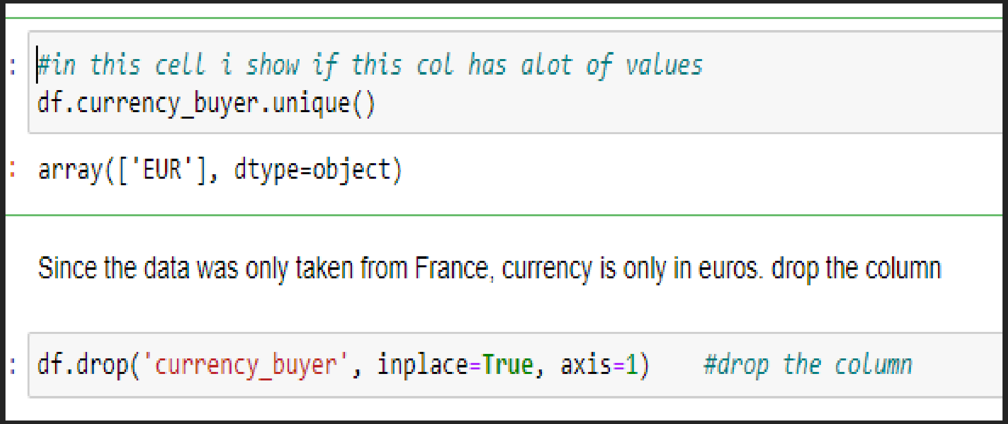
```
#this bring all columns that start with badge
df.loc[:,df.columns.str.startswith('badge')].columns
```

```
Index(['badges_count', 'badge_local_product', 'badge_product_quality',
       'badge_fast_shipping'],
      dtype='object')
```

```
#in this cell i convert the valuse of this columns to
#Categorical values to show to the model that it is categorical
df[['badge_local_product', 'badge_product_quality',
    'badge_fast_shipping']] = df[['badge_local_product',
                                  'badge_product_quality','badge_fast_shipping']].astype(str)
```

CURRENCY_BUYER COLUMN:

currency is only in euros drop the column



```
#in this cell i show if this col has alot of values
df.currency_buyer.unique()
```

```
array(['EUR'], dtype=object)
```

Since the data was only taken from France, currency is only in euros. drop the column

```
df.drop('currency_buyer', inplace=True, axis=1)     #drop the column
```

CRAWL MONTH:
crawl month is only from August, drop this column.

```python
df.crawl_month.unique()     #check the values of crawl_month
```

```
array(['2020-08'], dtype=object)
```

crawl month is only from August,drop this column.

```python
df.drop('crawl_month', inplace=True, axis=1) #drop column.
```

RATING COLUMN:

this is our target we check its values

```
df.rating.value_counts()  #check the values of rating should be from 1 to 5 only
```

```
4    774
0    479
5    170
3    135
2     12
1      2
6      1
Name: rating, dtype: int64
```

```
#check if we have any rating more than 5  and detrmine the index of it
df.loc[df['rating'] > 5]
```

RATING COLUMN:

after I checked if any rating more than 5 i found that only one row has rating equal to 6 and the index is 971 so I replaced it by near value which it 5

```python
#after i check if any rating more than 5 i found that only
#one row has rating equal to 6 and the index is 971
#and i used at[] to update the value of the column rating at this index by 5
df.at[971,'rating']=5
```

```python
# this for print row index 971 to check if the value updated or not
print(df.loc[971])
```

```
price                 49.0
retail_price            42
units_sold             100
uses_ad_boosts           0
rating                   5
```

RATING COLUMN:

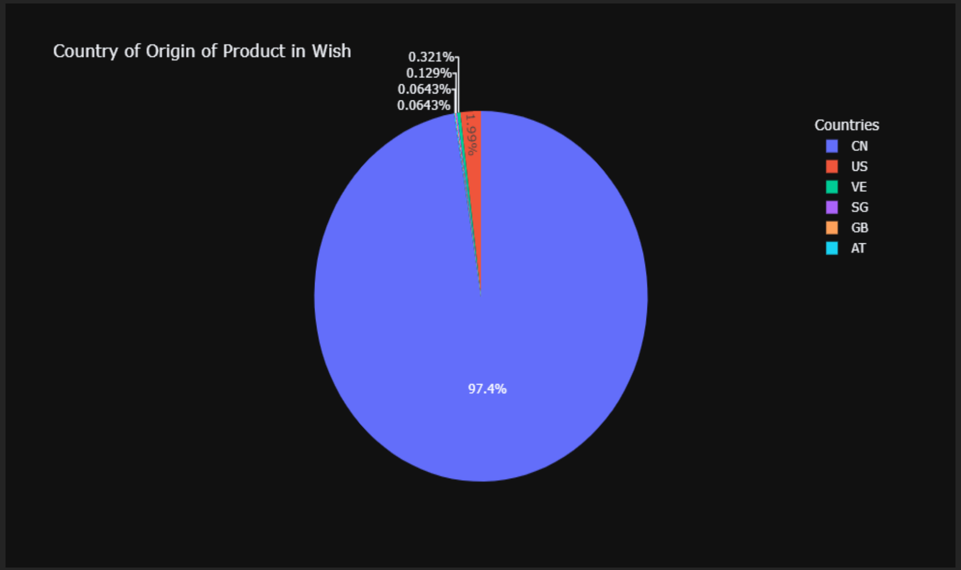
for more checking if the value updated or not
and zeros values come from test rating column so no problem about it

```
#this for more checking if the value updated or not
#and i know that rating has values 0 this from test_new dataset
df.rating.value_counts()
```

```
4    774
0    479
5    171
3    135
2     12
1      2
Name: rating, dtype: int64
```

ORIGIN COUNTRY COLUMN:

show country of origin of Product in Wish
the products mostly originate from China



SHIPPING OPTIONS AND PRICES:

Livraison standard is quite popular option for shipping

```
#this return each columns that contain word 'shipping'
df.loc[:,df.columns.str.startswith("shipping")].columns
```

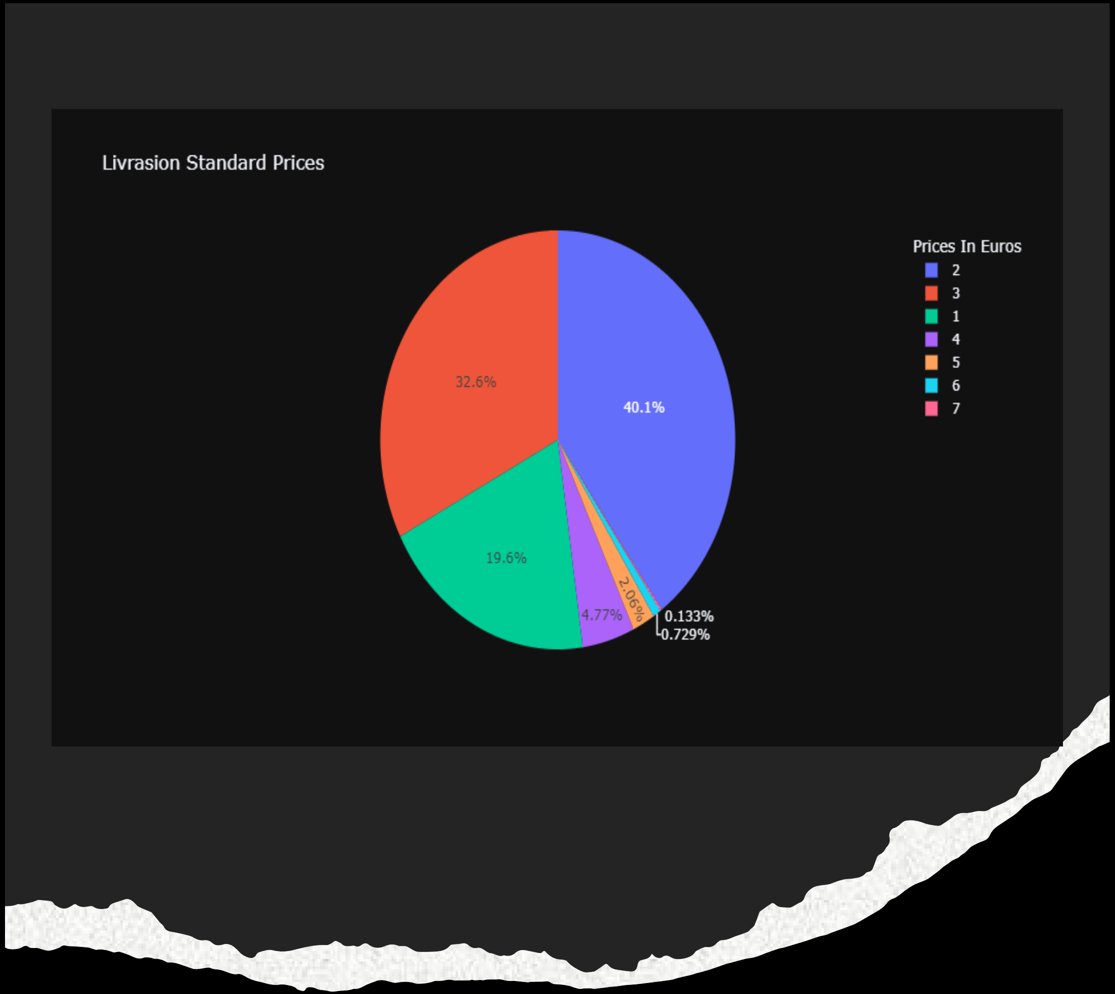
```
Index(['shipping_option_name', 'shipping_option_price', 'shipping_is_express'], dtype='object')
```

```
#this retur the short ddescription about shipping_option_name vlues
df['shipping_option_name'].value_counts()
```

```
Livraison standard                          1508
Standard Shipping                             20
Envio Padrão                                   9
Expediere Standard                             6
Envío normal                                   5
4                               الشحن القياسي
Standardowa wysyłka                            3
Standardversand                                3
Livraison Express                              3
Стандартная доставка                           3
Spedizione standard                            2
Standart Gönderi                               2
การส่งสินค้ามาตรฐาน                        2
currency_buyer' , 'theme' , 'crawl_month'      1
ការដឹកជញ្ជូនតាមស្តង់ដារ                      1
Ekspresowa wysyłka                             1
Name: shipping_option_name, dtype: int64
```

SHIPPING OPTIONS AND PRICES:

Most customers choose shipping options from 1-3 euros.



Origin Country, Shipping Names:

not important in make rating and as I was show in last slides about these columns

```
#in this cell i drop some columns
#not important in make rating
df.drop(['origin_country', 'shipping_option_name'  ,'merchant_name',
        'merchant_title' ,],axis=1, inplace=True)
```

SHIPPING_IS_EXPRESS:

Almost all the shipping is not express so we can drop it.

```
df['shipping_is_express'].value_counts()   #shipping_is_express valuses count
```

```
0    1569
1       4
Name: shipping_is_express, dtype: int64
```

Almost all the shipping is not express so we can drop it

```
df.drop('shipping_is_express', inplace=True, axis=1)
```

MERCHANT_HAS_PROFILE_PICTURE:

I think this col not important because it contains URL of the pictures and has a lot of null values so we can drop it

```
df.merchant_has_profile_picture.value_counts()
```

```
0    1347
1     226
Name: merchant_has_profile_picture, dtype: int64
```

```
df.drop('merchant_profile_picture' ,  inplace=True, axis=1) #drop the column
```

IDS:

Columns with ids will mislead our algorithms so I dropped them

```
df.drop(['merchant_id' , 'id'],axis=1, inplace=True)    #drop the columns
```

TAGS: tags set by the seller which customer can do search using it

```python
bag_of_words= ['summer',                                    #this is more words customer use it for seach
               "women's fashion",
               'fashion',
               'women',
               'casual',
               'plus size',
               'sleeveless',
               'dress',
               'shorts',
               'tops',
               'sexy',
               'beach',
               'sleeve',
               'short sleeves',
               'print',
               'shirt',
               'tank',
               'necks',
               'v-neck',
               'printed']
```

1 First replace uppercases with lowercases

2 Create separate columns with top 20 tags 'bag_of_words'

```python
for word in bag_of_words:
    # First check if str contains the word
    #If yes converto to 1 , if no convert to 0
    # Again convert 1 and 0 into strings for dummy variables later.
    df["tag_"+word] = df.tags.str.lower().str.contains(word).astype(int).astype(str)
```

```python
df.drop(['tags'],axis=1,inplace=True)   #now we can drop tags column
```

```python
df['tags'].head()
```

```
0    Summer,soildcolor,Plus Size,Tank,camisole,Tops...
1    bathing suit,Plus Size,bikini set,sexy swimsui...
2    Summer,Vest,momshirt,Get,summer t-shirts,funny...
3    Summer,Shorts,pants,Beach,Plus Size,beachpant,...
4    Summer,Floral print,women dresses,fashion dres...
Name: tags, dtype: object
```

PRODUCT COLOR:

on the color column we will make some updates

```
#on the color column we will make some updates
df['product_color']=df['product_color'].str.lower()  #convert all values of this column to lowercase

df['product_color'].replace('gray', 'grey', inplace=True)  #convet the value of gray to grey
df['product_color'].replace(np.nan, 'black', inplace=True) #to fill the nan values by the most frequent color

# df['product_color'].replace('RED', 'red', inplace=True)
# df['product_color'].replace('White', 'white', inplace=True)
# df['product_color'].replace('Blue', 'blue', inplace=True)

print(df['product_color'].unique())
```

PRODUCT COLOR:

this image show the unique values and the count of each one

```
print(df['product_color'].unique())  #from this method we determine the values of this column
print(df['product_color'].value_counts())
```

```
['yellow' 'black' 'white' 'lakeblue' 'apricot' 'brown' 'winered' 'blue'
 'red' 'navyblue' 'green' 'khaki' 'White' 'white & green' 'multicolor'
 'lightpink' 'pink' 'RED' 'armygreen' 'lightblue' nan 'coffee' 'grey'
 'skyblue' 'watermelonred' 'pink & black' 'whitefloral' 'purple' 'navy'
 'pink & white' 'rosered' 'orange' 'Black' 'mintgreen' 'leopardprint'
 'gray' 'navy blue' 'star' 'rose' 'lightyellow' 'camouflage'
 'black & yellow' 'whitestripe' 'navyblue & white' 'black & blue'
 'lightred' 'violet' 'gold' 'black & green' 'white & black' 'burgundy'
 'black & white' 'lightgrey' 'coolblack' 'lightgreen' 'beige' 'darkblue'
 'darkgreen' 'silver' 'wine red' 'Army green' 'pink & blue' 'rainbow'
 'claret' 'floral' 'brown & yellow' 'light green' 'Pink' 'blue & pink'
 'dustypink' 'camel' 'orange-red' 'rosegold' 'ivory' 'fluorescentgreen'
 'winered & yellow' 'offwhite' 'lightgray' 'wine' 'army' 'applegreen'
 'nude' 'pink & grey' 'Rose red' 'denimblue' 'blackwhite' 'Blue' 'leopard'
 'coralred' 'tan' 'orange & camouflage' 'army green' 'offblack' 'jasper'
 'white & red' 'red & blue' 'greysnakeskinprint' 'lightpurple'
 'black & stripe' 'lightkhaki' 'prussianblue' 'gray & white']
black              302
white              254
yellow             105
blue                99
pink                99
                  ...
```

PRODUCT_VARIATION_SIZE_ID:

this col show the different sizes but contains a lot of errors, so I make some updates on it

```
#show the values count of Products size
print(df['product_variation_size_id'].value_counts())
```

```
S                  647
XS                 357
M                  204
XXS                102
L                   50
                   ...
2                    1
20PCS-10PAIRS        1
Size-5XL             1
Size/S               1
36                   1
Name: product_variation_size_id, Length: 106, dtype: int64
```

PRODUCT_VARIATION_SIZE_ID:

show the values of the product size "different size"

```python
#show the the values of the product size "different size"
df['product_variation_size_id'].unique()
```

```
array(['M', 'L', 'XS', 'S', 'XL', '26(Waist 72cm 28inch)', 'S.',
       'S(bust 88cm)', 'XXS', 's', '29', 'choose a size', 'XXXS',
       'Base Coat', 'Size M', 'XXL', 'M.', 'XS.',
       '100 x 100cm(39.3 x 39.3inch)', '2pcs', '4XL', '1', '25-S',
       'Size-XXS', '5PAIRS', '35', 'Pack of 1', 'Size S', 'Size-S', '6XL',
       '25', 'S/M(child)', '60', 'Size-XS', 'S (waist58-62cm)',
       'SIZE XXS', '10 ml', 'X   L', 'Women Size 36', '04-3XL',
       'Size -XXS', '1 pc.', 'Floating Chair for Kid', 'S Pink', '34',
       'US-S', 'Size XXS', 'pants-S', 'XXXXL', 'SIZE-XXS', 'SIZE XS',
       '1pc', 'Size S.', '100 cm', 'S..', 'Round', '4-5 Years', '5', '33',
       '30 cm', '2', 'XXXXXL', '20PCS-10PAIRS', '2XL', 'Size-5XL',
       'Size4XL', 'One Size', 'size S', 'Size/S', 'B', 'SizeL', '20pcs',
       '1 PC - XL', 'Suit-S', 'Base & Top & Matte Top Coat',
       'Baby Float Boat', '1m by 3m', 'SIZE S', 'White', '40 cm', '5XL',
       '10pcs', 'H01', 'S(Pink & Black)', '32/L', 'daughter 24M', 'XXXL',
       '4', '3XL', '80 X 200 CM', 'EU 35', '100pcs', 'first  generation',
       'Size--S', 'SIZE-4XL', 'L.', 'Women Size 37', 'S Diameter 30cm',
       'Size-L', 'AU plug Low quality', '3 layered anklet', '17',
       'US 6.5 (EU 37)', 'US5.5-EU35', 'EU39(US8)', '36'], dtype=object)
```

PRODUCT_VARIATION_SIZE_ID:

replace some of sizes to main sizes as below

```
#in below cell i replace some of sizes to main sizes as below
```

```
df['product_variation_size_id'].replace(['S', 'S.', 's', 'Size S', 'Size-S', 'Size S.', 'Suit-S',
                                         'size S','S Pink', 'pants-S', 'US-S', 'SIZE S', 'S (waist58-62cm)',
                                         'Size--S', '25-S', 'Size/S', 'S Diameter 30cm', 'S..',
                                         'S(Pink & Black)'], 'S', inplace=True)
df['product_variation_size_id'].replace(['XS', 'XS.', 'SIZE XS', 'Size-XS'], 'XS', inplace=True)
df['product_variation_size_id'].replace(['XXS', 'XXXS', 'SIZE-XXS', 'Size -XXS', 'Size XXS',
                                         'Size-XXS', 'SIZE XXS'], 'XXS+', inplace=True)
df['product_variation_size_id'].replace(['M', 'M.', 'Size M'], 'M', inplace=True)
df['product_variation_size_id'].replace(['L', 'SizeL', '32/L', 'L.', 'Size-L'], 'L', inplace=True)
df['product_variation_size_id'].replace(['XL', '2XL', '1 PC - XL', 'X   L'], 'XL', inplace=True)
df['product_variation_size_id'].replace(['XXL', '4XL', '2XL', 'Size4XL', '3XL',
                                         'XXXXXL', '1 PC - XL', 'SIZE-4XL', '04-3XL',
                                         'Size-5XL', 'XXXXL', '5XL', 'XXXL'], 'XXL+', inplace=True)
size_val_counts = df['product_variation_size_id'].value_counts()
# Select the values where the count is less than 5
to_change = size_val_counts[size_val_counts <= 5].index
df.loc[df['product_variation_size_id'].isin(to_change), 'product_variation_size_id'] = "Other"
df['product_variation_size_id'] = df['product_variation_size_id'].replace(np.nan, "Other")
```

PRODUCT_VARIATION_SIZE_ID :

show the values of the product size after make updates

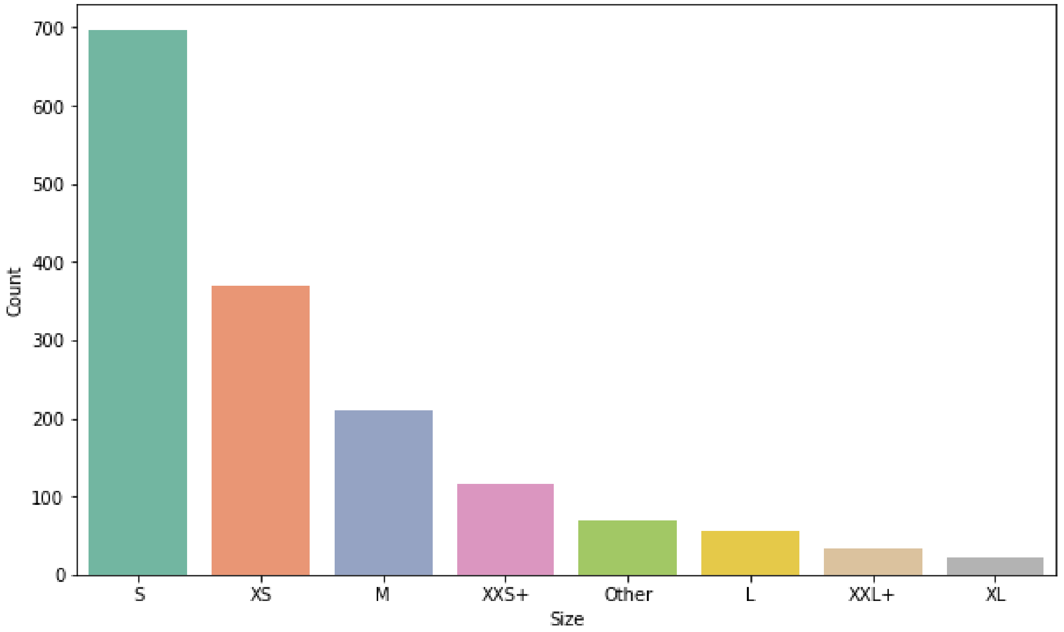
```
df['product_variation_size_id'].value_counts()
```

```
S        696
XS       370
M        210
XXS+     115
Other     69
L         56
XXL+      34
XL        23
Name: product_variation_size_id, dtype: int64
```

THIS SHOW THE MOST REPEATED SIZES AND THE NUMBER OF REPEATS:

```python
#this show the most repeated sizes and the number of repeats
plt.figure(figsize=(10, 6))
ax = sns.countplot(x = 'product_variation_size_id',
                   order = df['product_variation_size_id'].value_counts().index,
                   palette= "Set2",
                   data=df)
ax.set(xlabel='Size', ylabel='Count')

plt.show()
```



We convert the categorical data to numeric values:

Using LabelEncoder

product_color

product_variation_size_id

```
from sklearn import preprocessing

#make label encoding to product_color

pc_fit = df['product_color'].unique()
le = preprocessing.LabelEncoder()
le.fit(pc_fit)
df['product_color'] = le.transform(df['product_color'])

#make label encoding to product_variation_size_id

pc_fit1 = df['product_variation_size_id'].unique()
le1 = preprocessing.LabelEncoder()
le1.fit(pc_fit1)
df['product_variation_size_id'] = le1.transform(df['product_variation_size_id'])
```

MERCHANT_INFO_SUBTITLE COLUMN:

The website shows this to the user to give an overview of the seller's stats to the user.

```
]: df['merchant_info_subtitle'].value_counts()

]: 83 % avis positifs (32,168 notes)    14
   86 % avis positifs (12,309 notes)    11
   87 % avis positifs (42,919 notes)     8
   85 % avis positifs (80,093 notes)     7
   84 % avis positifs (5,654 notes)      6
                                        ..
   77 % avis positifs (849 notes)        1
   82 % avis positifs (870 notes)        1
   90 % avis positifs (6,033 notes)      1
   (12,913 notes)                        1
   87 % avis positifs (1,086 notes)      1
   Name: merchant_info_subtitle, Length: 1058, dtype: int64
```

WE GET THE PERCENTAGE FOR EACH ROW IN merchant_info_subtitle COL

```python
#here we get the percentage for each row in merchant_info_subtitle col
import re
def getPercentage(x):
    match = re.search(r'\d+%', str(x))
    if match is None:
        return None
    else:
        return float(match.group().rstrip("%"))
```

```python
df['merchant_info_subtitle'] = df['merchant_info_subtitle'].str.replace(' ', '')
df['merchant_positive_pct'] = df['merchant_info_subtitle'].apply(getPercentage)
df['merchant_positive_pct'].head()
```

```
0    88.0
1    91.0
2    83.0
3    87.0
4    91.0
Name: merchant_positive_pct, dtype: float64
```

FILL NULL VALUES IN merchant_positive_pct AND DROP merchant_info_subtitle

```python
#fill any null values by the mean of total
df['merchant_positive_pct'].fillna((df['merchant_positive_pct'].mean()), inplace=True)
```

```python
# after i take the percentage for each row and
# generate new col i will drop merchant_info_subtitle
df.drop('merchant_info_subtitle' , axis=1 , inplace=True)
```

BY THIS FUNCTION WE CAN CHECK IF OUR DATA HAS ANY MISSING VALUES

```
# df=df.fillna(0)                          #missing_values_table in dataset
def missing_values_table(df):
        mis_val = df.isnull().sum()
        mis_val_percent = 100 * df.isnull().sum() / len(df)
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
        print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")
        return mis_val_table_ren_columns

df_qa=missing_values_table(df)
df_qa
```

```
Your selected dataframe has 39 columns.
There are 0 columns that have missing values.
```

Missing Values	% of Total Values

NOW WE CAN SPLIT DATA AFTER WE MADE PREPROCESSING

```
train_data_rows = train_data.shape[0]                    #number of rows train_new dataset
train_data_cleaned = df.iloc[:train_data_rows]           #split df after preprocessing until train_data_rows
print(train_data_cleaned.shape)                          #to show the train_data_cleaned shape
test_data_cleaned = df.iloc[train_data_rows :]           #split df after preprocessing from train_data_rows to last row
print(test_data_cleaned.shape)
test_data_cleaned= test_data_cleaned.drop('rating', axis=1) #drop the rating column from test_data_cleaned
```

SOME IMPORTANT LIBRARIES TO HELP ME

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score, roc_curve, auc
from sklearn import linear_model
from sklearn.linear_model import LassoCV
from sklearn.feature_selection import SelectKBest, chi2, f_classif, SelectFromModel, RFECV, VarianceThreshold
# from fuzzywuzzy import fuzz,process
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from plotly.offline import iplot, init_notebook_mode
import cufflinks as cf
import plotly.graph_objs as go
# import chart_studio.plotly as py
init_notebook_mode(connected=True)
cf.go_offline(connected=True)
# Set global theme
cf.set_config_file(world_readable=True, theme='ggplot')
import warnings
```

```python
#import some model to use it
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn import decomposition, datasets
from sklearn import tree
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
```

MAKE SPLIT TO X_TRAIN AND Y_TRAIN

```
#make varible called "X" "input" contains all columns without rating column
X=train_data_cleaned.drop(columns ='rating' )
#make varaible called "y" "target" contain only rating column
y= train_data_cleaned['rating']
print(X.shape)
print(y.shape)
```

BY USING STANDARDSCALER WE MADE SCALE TO OUR DATA "TRAIN AND TEST"

```
]:  #in this cell we make scale to our train data
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    scaler.fit(X)
    X_scale = scaler.transform(X)
```

```
#in this cell we make scale to our test data
from sklearn.preprocessing import StandardScaler
scaler1 = StandardScaler()
scaler1.fit(test_data_cleaned)
test_data_cleaned_scal = scaler1.transform(test_data_cleaned)
```

MAKE SPLIT TO TRAIN OUR MODEL AND TEST IT

```
#make X_train, X_test, y_train, y_test on train data to train our model and test it
X_train, X_test, y_train, y_test = train_test_split(X_scale,
                                                   y, test_size = .2,random_state=10 , shuffle=True)
```

MODELS ARE:-

- DecisionTreeClassifier mode
- RandomForestClassifier model
- Support vector machines model
- GaussianNB model
- CategoricalNB model

DecisionTree Classifier mode

```
# ds=DecisionTreeClassifier(random_state=10 ,criterion='entropy',max_depth = 20 )#this when we use different Parameters
ds=DecisionTreeClassifier()  #this model without hyperparameter2
ds.fit(X , y) #fit the model  on all train dataset
pred=ds.predict(X) #test our model
f1_score(y, pred,average = 'weighted')  #check the model f1 score
```

```
0.9927469454945528
```

```
y_pred_test=ds.predict(test_data_cleaned)  #test the trained model on the test data
y_pred_test= y_pred_test.astype(float) #convert the out put to float datatype
```

```
id = test_data['id']                                                    #in this line i take id from sample data
pred_df = pd.DataFrame(data={'id': np.asarray(id), 'rating': y_pred_test})   #take prediction values and id to make df and
pred_df.to_csv('pred_TES.csv', index=False)                            #save them in csv file
```

Random_Forest Classifier model

```
# rfc= RandomForestClassifier(n_estimators =50, random_state=10)    #this when we use different Parameters
rfc= RandomForestClassifier() #make model randomforestclassifier     "this give us best f1 score on kaggle"
rfc.fit(X , y) #train the model
pred=rfc.predict(X)   #test the model
# f1_score(y_test, pred,average = 'weighted')
f1_score(y, pred,average = 'weighted')   #check the f1 score of the model
```

```
0.992610514673952
```

```
y_pred=rfc.predict(test_data_cleaned)                                #make predictions for the test data
y_pred= y_pred.astype(float)                                         #convert the output to float datatype
```

```
id = test_data['id']                                                 #in this line i take id from sample data
pred_df = pd.DataFrame(data={'id': np.asarray(id), 'rating': y_pred}) #take prediction values and id to make df and
pred_df.to_csv('pred_TEST_final.csv', index=False)                   #save them in csv file
```

Support vector machines model

```
]: model_svc = SVC(gamma = 0.01 , C=1 , degree= 10) #create object from SVM
   model_svc.fit(X, y)                 #train the model
   svc_pred = model_svc.predict(X)     #predict the train data to check the model
   #this check the accurracy and f1 score of svc model
   svc_accuracy = accuracy_score(y, svc_pred)
   svc_f1 = f1_score(y, svc_pred, average='weighted')
   print('Accuracy : ', "%.2f" % (svc_accuracy*100))
   print('F1 : ', "%.2f" % (svc_f1*100))
```

```
Accuracy :  99.18
F1 :  99.17
```

```
]: y_predsvc=model_svc.predict(test_data_cleaned) #make predictions for the test data
   y_predsvc= y_predsvc.astype(float)             #convert the output to float datatype
```

```
]: id = test_data['id']                                                       #in this line i take id from sample data
   pred_df = pd.DataFrame(data={'id': np.asarray(id), 'rating': y_predsvc})   #take prediction values and id to make df and
   pred_df.to_csv('predcsv.csv', index=False)                                 #save them in csv file
```

GaussianNB model

```
model_GNB=GaussianNB()                            #genetrate model GaussianNB() without perparameters
# model_GNB=GaussianNB(var_smoothing=1e-2) #genetrate model GaussianNB() with    perparameters
model_GNB.fit(X , y)                              #train the model on train data
predGNB=model_GNB.predict(X)                  #test the model on train data
f1_score(y, predGNB,average = 'weighted')  #check f1 score for our model
```

```
0.5360514059431413
```

```
y_predGNB=model_GNB.predict(test_data_cleaned)  #predict the test data
predGNB= y_predGNB.astype(float)                 #convert the output to float datatype
```

```
id = test_data['id']                                                    #in this line i take id from sample data
pred_df = pd.DataFrame(data={'id': np.asarray(id), 'rating': y_predsvc})  #take prediction values and id to make df and
pred_df.to_csv('predGaussianNB.csv', index=False)                              #save them in csv file
```

CategoricalNB model

```
# from sklearn.preprocessing import MinMaxScaler #using pipleine to make apply this steps
                                                  # "Normalizing" ana "CategoricalNB"
# p = Pipeline([('Normalizing',MinMaxScaler()),('MultinomialNB',CategoricalNB())])
p = CategoricalNB() # this model without scaling
p.fit(X,y)                     #train the model on train data
pre_CategoricalNB=p.predict(X) #test the model on train data
f1_score(y, pre_CategoricalNB,average = 'weighted') #check the value of f1 score
```

```
0.7881267153083404
```

```
y_CategoricalNB=p.predict(test_data_cleaned.astype(float))      #predict the test data
# predCategoricalNB= y_CategoricalNB.astype(float)  #convert the output to float datatype
```

```
id = test_data['id']                                                          #in this line i take id from sample data
pred_df = pd.DataFrame(data={'id': np.asarray(id), 'rating': y_predsvc})   #take prediction values and id to make df and
pred_df.to_csv('predCategoricalNB.csv', index=False)                                  #save them in csv file
```

1. Why Data Mining is a misnomer? What is another preferred name?

- Data mining is a misnomer, because the goal is the extraction of patterns and knowledge from large amounts of data, not the extraction (mining) of data itself.
- Data mining is also known as Knowledge Discovery in Data (KDD).

2. What is the general knowledge discovery process? What is the difference between a data engineer and data scientist/AI engineer?

- knowledge discovery can be defined as the process of identifying interesting and new patterns in data. These patterns can include relations, events or trends, and they can reveal both exceptions and regularities. In the core of the process, data mining methods are utilized for extracting and verifying patterns.
- Data engineers create and maintain key data infrastructures like databases, data warehouses, and data pipelines. Data engineers also prepare data for production by converting raw, unstructured data into a structured format that can be analyzed and interpreted.
- Data scientists analyze and interpret data to solve business problems. Initially, data scientists explore data and conduct market research in order to formulate business questions around a specific trend or pain point. Data scientists must then frame business questions as data analytics problems.

3. In data mining, what is the difference between prediction and categorization?

- \- Functionality:

  Classification is about determining a (categorial) class (or label) for an element in a dataset

  Prediction is about predicting a missing/unknown element(continuous value) of a dataset

- \- Working Strategy:

  In classification, data is grouped into categories based on a training dataset.

  In prediction, a classification/regression model is built to predict the outcome(continuous value)

4. Why data science/machine learning is a bad idea in the context of information security?

- Machine learning was not designed with security in mind and as such is prone to adversarial manipulation and reverse engineering. While most data-based learning models rely on a static assumption of the world, the security landscape is one that is especially dynamic, with an ongoing never-ending arms race between the system designer and the attackers.
- Any solution designed for such a domain needs to consider an active adversary and needs to evolve over time, in the face of emerging threats. We term this as the "Dynamic Adversarial Mining" problem

5. What is CIA principle and how can we use it to access the security/privacy aspect of the AI system/pipelines?

- A simple but widely-applicable security model is the CIA triad; standing for Confidentiality, Integrity and Availability; three key principles which should be guaranteed in any kind of secure system. This principle is applicable across the whole subject of Security Analysis, from access to a user's internet history to security of encrypted data across the internet. If any one of the three can be breached, it can have serious consequences for the parties concerned
- In the information security (InfoSec) community, "CIA" has nothing to do with a certain well-recognized US intelligence agency.