Predict Ratings of Sephora Products

Siyu Shen

October 2020

1 Introduction

Curating positive ratings on Sephora listing is one of the easiest ways to increase consumer trust, brand equity, and sales. That's why it is important to find out what kind of product will have a good rating which is over 4.0. Therefore, in Sephora website data file, we use 'rating' as target variable. Since rating is range from 0.0 to 5.0, we have a regression problem.

There are 9168 data points and 12 features in 'sephora_website_data.csv' data. To be more specific, 'love' is the number of people loving the product. For categorical features, 'online_only' is 1 if the product is sold online, 0 otherwise. 'MarketingFlags' is a boolean feature which determines if the product from the website has limited edition, sells exclusive or only online.

This data set has been used by Raghad Alharbi's project[1] in Data Science Immersive Course with General Assembly and Misk academy. The project was about using web scraping methods to collect more than 1,000 useful records from any website of their choice. She analyzed the distributions of features. Also, she finds correlations between each columns.

2 Exploratory Data Analysis

The very first step we took is to visualize features and explore the relationship of different features. To maximize insight into the target variable 'rating', we use bar plot.

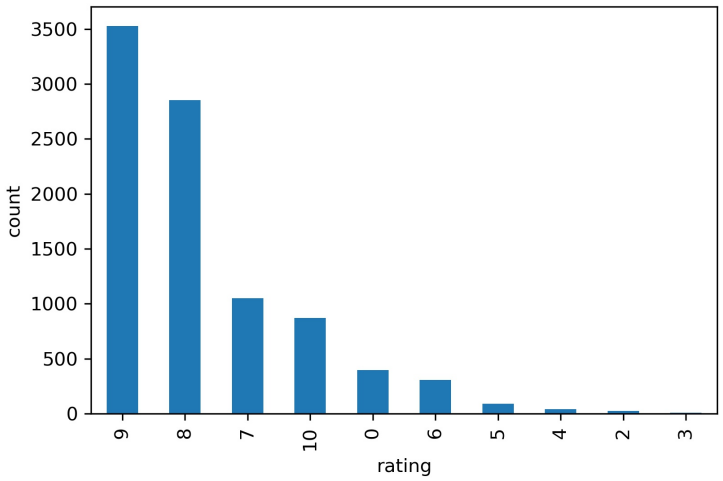


Figure 1: Bar plot of rating

Figure 1 is right skewed and most rating is range from 4.0 to 4.5. Also, seldom customers give a low rate which is range from 0.0 to 2.5.

Also, we want to examine the relationship between different categories of products and their prices. Since there are 143 categories, we first take a look at the categories sells more than 200 products. Figure 2 shows that Face Serum and Perfume have higher prices which are higher than or equal to 50 dollars. Face Wash &

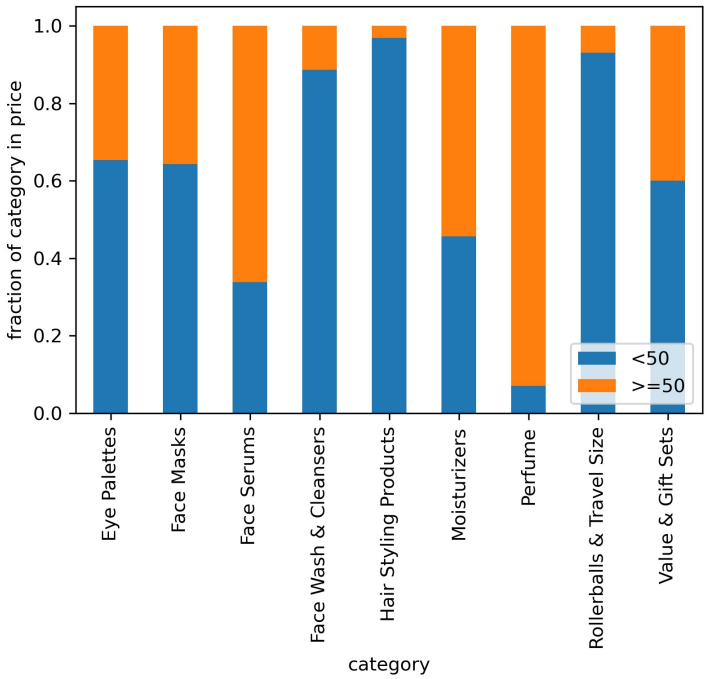


Figure 2: Stacked bar plot of category and fraction in price

Cleansers, Hair Styling Products & Roller ball and Travel size have comparable lower prices which are less than 50 dollars.

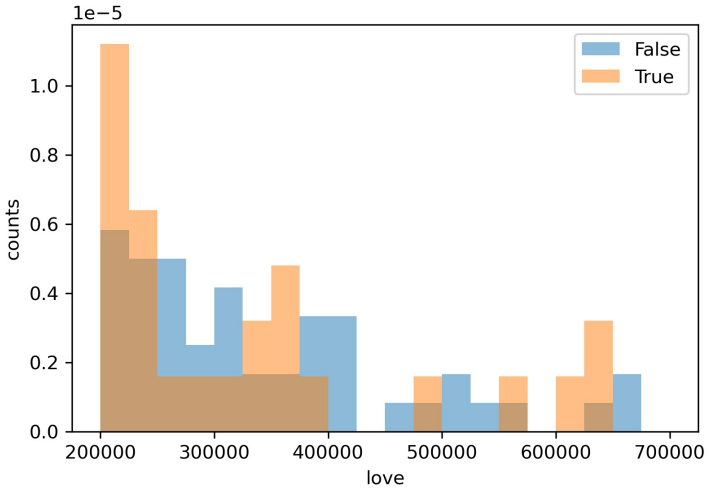


Figure 3: Category-specific histograms of love and MarketingFlags

Furthermore, we want to see that if a products is MarketingFlags, which means if it has limited edition, sells exclusive or only online, will affect the numbers of love. According to Figure 3, the product without MarketingFlags has a smoother distribution which indicates that products with MarketingFlags tend to have less love. However, the number of 'love' have similar right skewed distributions for both.

Then we take a look at the relationship between numbers of love customers give for a product and the number of actual reviews. Figure 4 shows that less 'love' tend to have less reviews since the data points are concentrated in the bottom left corner. There seems to have a weak positive relationship between 'love' and number of reviews.

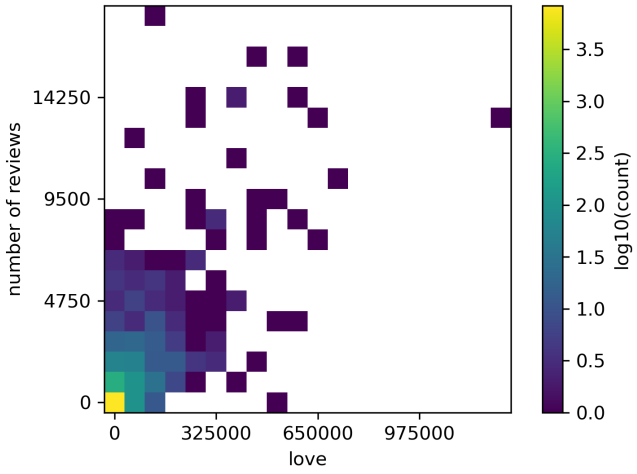


Figure 4: heatmap of love and number of reviews

3 Data Prepossessing

The data set is IID because all data were obtained from one-time purchase on Sephora website. Every data point is generated by different products and the customers are independent individuals. Besides, all data are obtained under same conditions and have same features. Also, the generative process has no memory of past generated samples. So the data has no group structure or time structure.

First, we split the the data set into other and test with ratio 8:2. Since there are 9168 data points in the Sephora data set, it is a small data set. Then applied Kfold for splitting data because we want to random splitting the train and validation set for our regression problem. When we change the random state and repeat the whole ML pipeline, we will have similar uncertainty when the model is deployed.

There are 11 features we are going to use in the prepossessed data from the original data set. We did not applied any encoder for the target variable 'rating'. We applied One-Hot encoder to unordered categorical data like 'MarketingFlags', 'category', 'exclusive', 'limited_edition', 'limited_time_offer' and 'online_only'. Then standardized all other continuous features like 'love', 'number_of_reviews', 'price' and 'liquid_size' using StandardScaler encoder. Now we have 154 columns after fit transforming data.

Since 'size' variable cannot be use directly, we first converted the unit to 'oz' for every data points. However, there are 52% of 'size' data has missing value in the original data set. We cannot deal with them since they are missing not at random. For example, for the gift sets, there is always no size information because it include different products. Therefore, we finally decide to drop the 'size' features during the predicting process.

4 Github repository

For further information go to the url: https://github.com/ssy00603/DATA-1030-project.git

References

[1] Raghad Alharbi. *Sephora Website Data: More than 9,000 Products with their Ratings and Ingredient Lists*, 2020 (accessed October 10, 2020).

Appendix

October 13, 2020

[81]:
```
import numpy as np
import pandas as pd
import statistics as stats
import matplotlib
from matplotlib import pylab as plt

# Classification - target variable: rating
df = pd.read_csv('../data/sephora_website_dataset.csv')
# Drop 'MarketingFlags_content' becasue it has redundant information
df = df[df.columns.difference(['options', 'details', 'how_to_use',␣
↪'ingredients', 'URL','MarketingFlags_content','id','name','brand'])]
print(df.shape)
df.head()
```

```
(9168, 12)
```

[81]:

	MarketingFlags	category	exclusive	limited_edition	limited_time_offer	love	number_of_reviews	online_only	price	rating	size	value_price
0	True	Fragrance	0	0	0	3002	4	1	66.0	4.0	5 x 0.16oz/5mL	75.0
1	True	Cologne	0	0	0	2700	76	1	66.0	4.5	0.7 oz/ 20 mL	66.0
2	True	Perfume	0	0	0	2600	26	1	180.0	4.5	5 oz/ 148 mL	180.0
3	True	Perfume	0	0	0	2900	23	1	120.0	4.5	2.5 oz/ 74 mL	120.0
4	True	Fragrance	0	0	0	943	2	1	72.0	3.5	5 x 0.16oz/5mL	80.0

[38]:
```
len(df['category'].unique())
```

[38]: 143

[71]:
```
# Convert 'size' into usable information
def convert_size(size):
    if 'x' in size:
        return np.nan
    elif 'fl' in size or 'g' in size or 'Vials' in size or 'Mini' in size or␣
 ↪'Glitter' in size or 'Jumbo' in size:
        return np.nan
    elif ' oz' in size:
        return float(size.strip().split('/')[0].split(' ')[0])
    elif 'oz' in size:
        return float(size.strip().split('oz')[0])
    else:
        return np.nan

df['liquid_size'] = [convert_size(df['size'][i]) for i in range(df.shape[0])]
df['unit_price'] = df['price']/df['liquid_size']
```

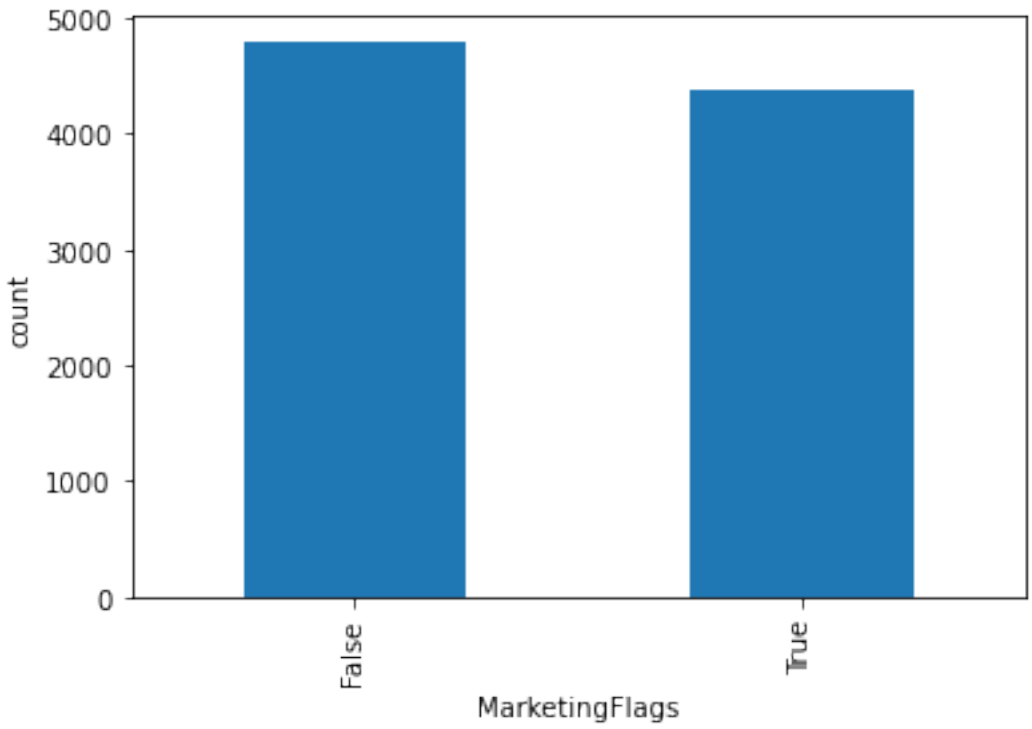
[40]:
```
# Extraxt categorical columns
def cat_cols(df, col = df.columns):
    cat = []
    for name in col:
        if len(df[name].value_counts()) < 3:
            cat.append(name)
    return cat
```

[41]:
```
categorical_cols = cat_cols(df)
categorical_cols.append('rating')
```

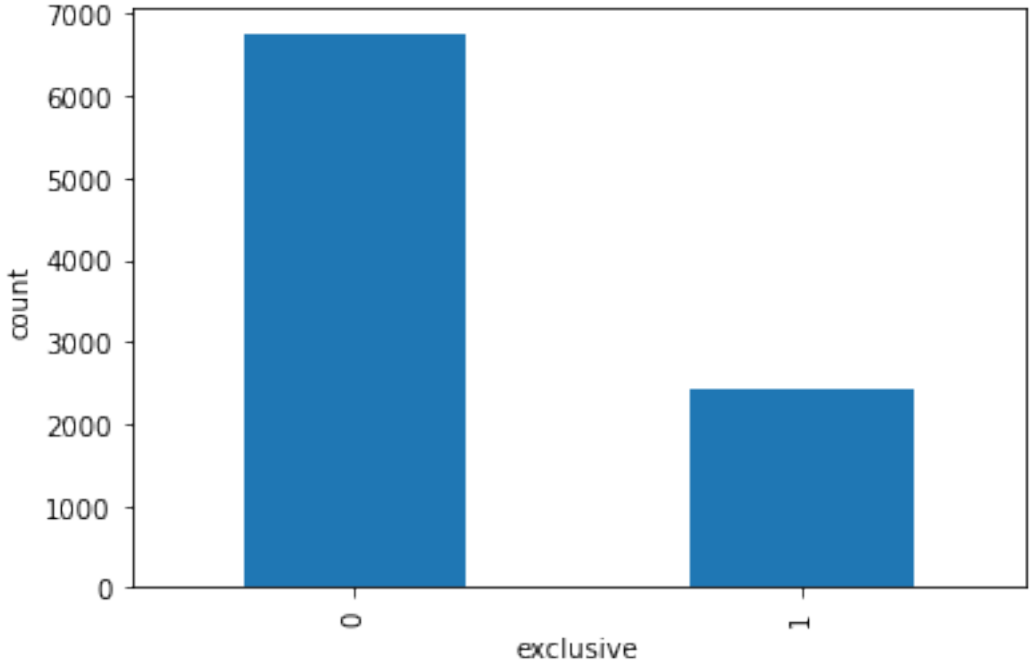
[42]:
```
# Extraxt continuous columns
continuous_cols = list(df[df.columns.difference(categorical_cols)].columns)
```

[43]:
```
# categorical
from matplotlib.pyplot import figure

for col in categorical_cols:
    print(df[col].value_counts())
    pd.value_counts(df[col]).plot.bar()
    plt.ylabel('count')
    plt.xlabel(col)
    plt.savefig('../figures/'+str(col)+'.jpg', dpi=300)
    plt.show()
```

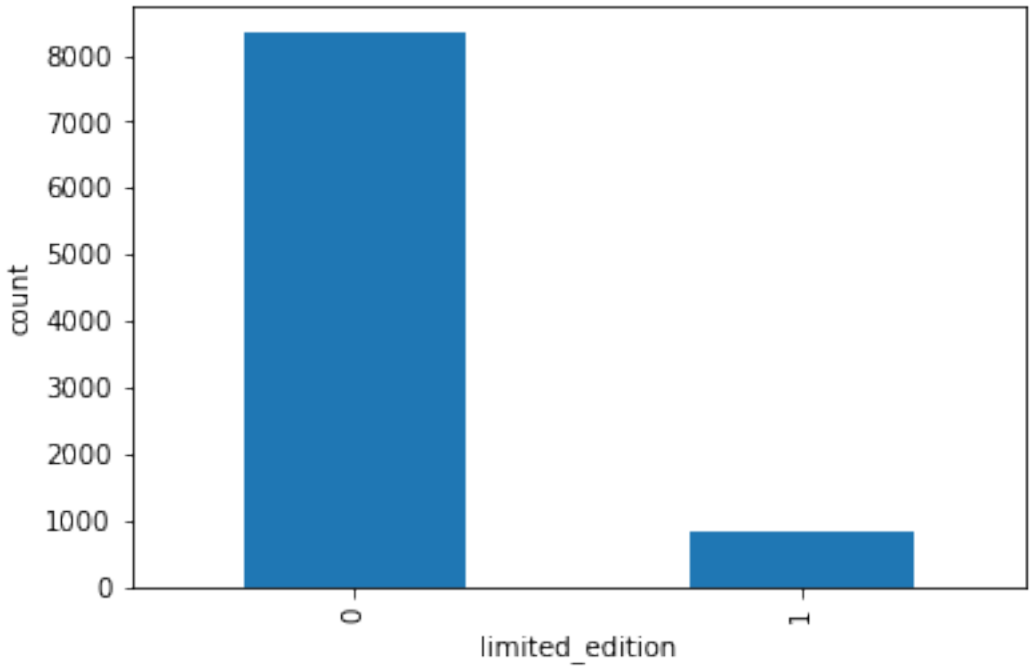
```
False    4786
True     4382
Name: MarketingFlags, dtype: int64
```



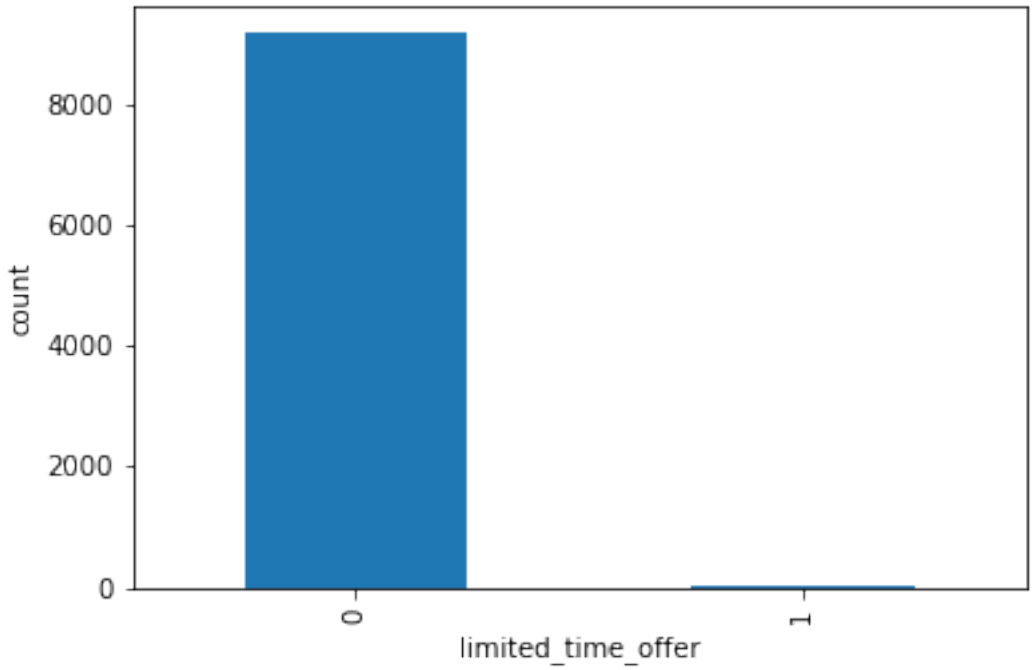

```
0    6741
1    2427
Name: exclusive, dtype: int64
```



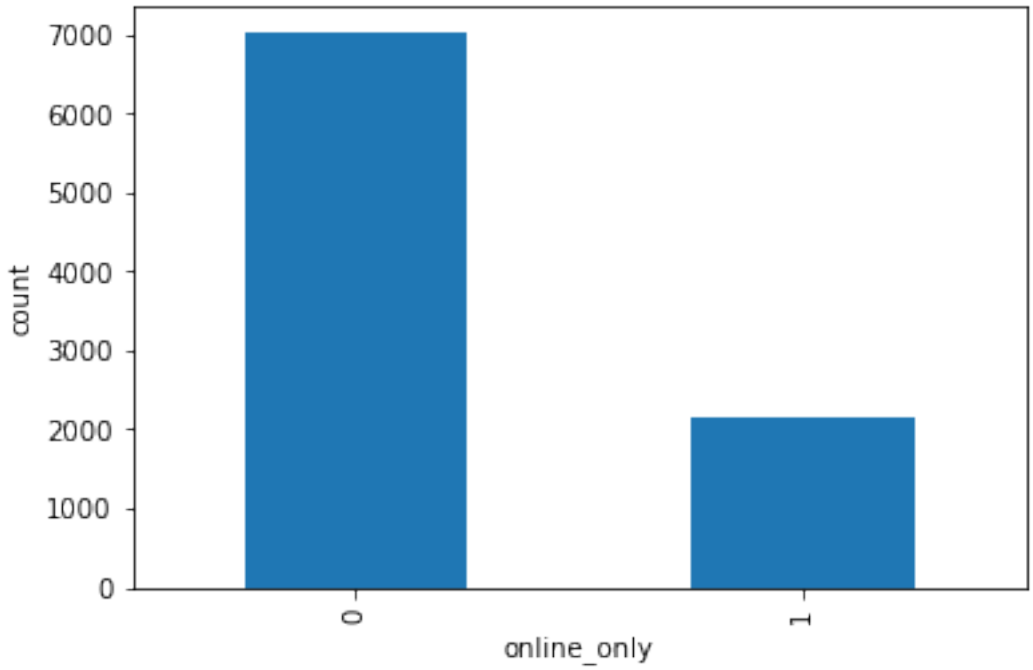
```
0    8326
1     842
Name: limited_edition, dtype: int64
```



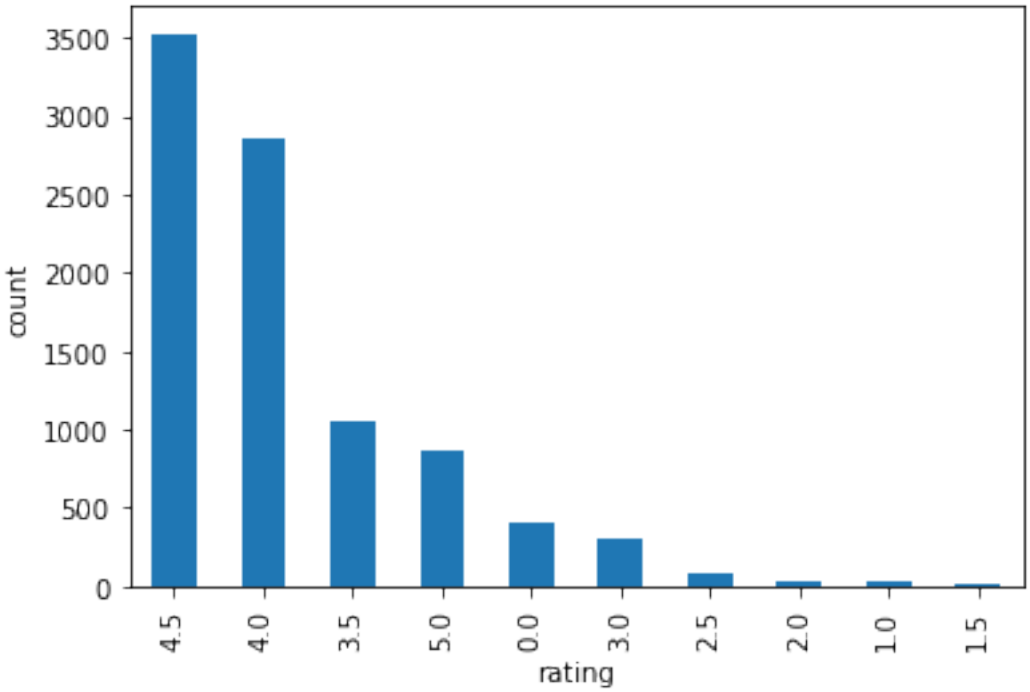

```
0    9165
1       3
Name: limited_time_offer, dtype: int64
```




```
0    7015
1    2153
Name: online_only, dtype: int64
```

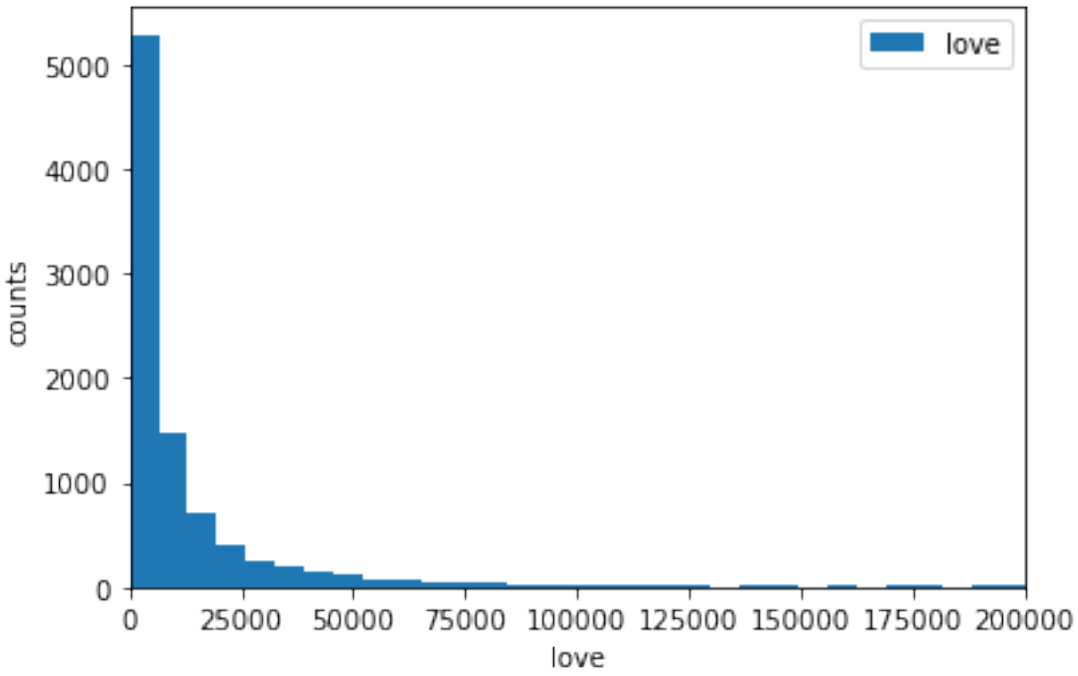


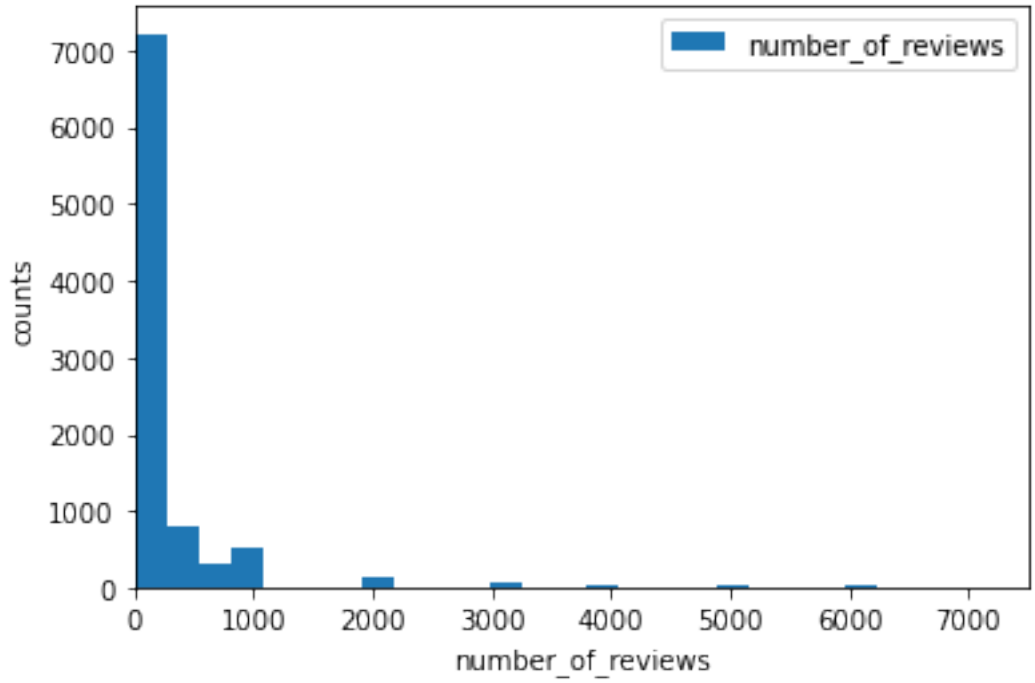
```
4.5    3527
4.0    2853
3.5    1051
5.0     872
0.0     398
3.0     308
2.5      88
2.0      39
1.0      23
1.5       9
Name: rating, dtype: int64
```

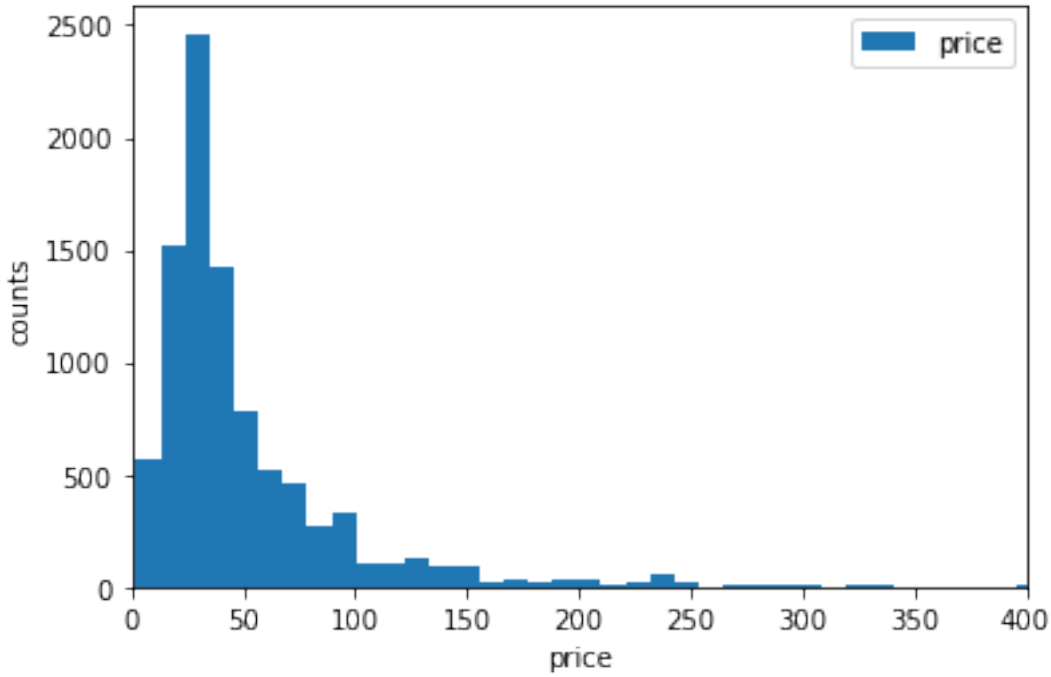


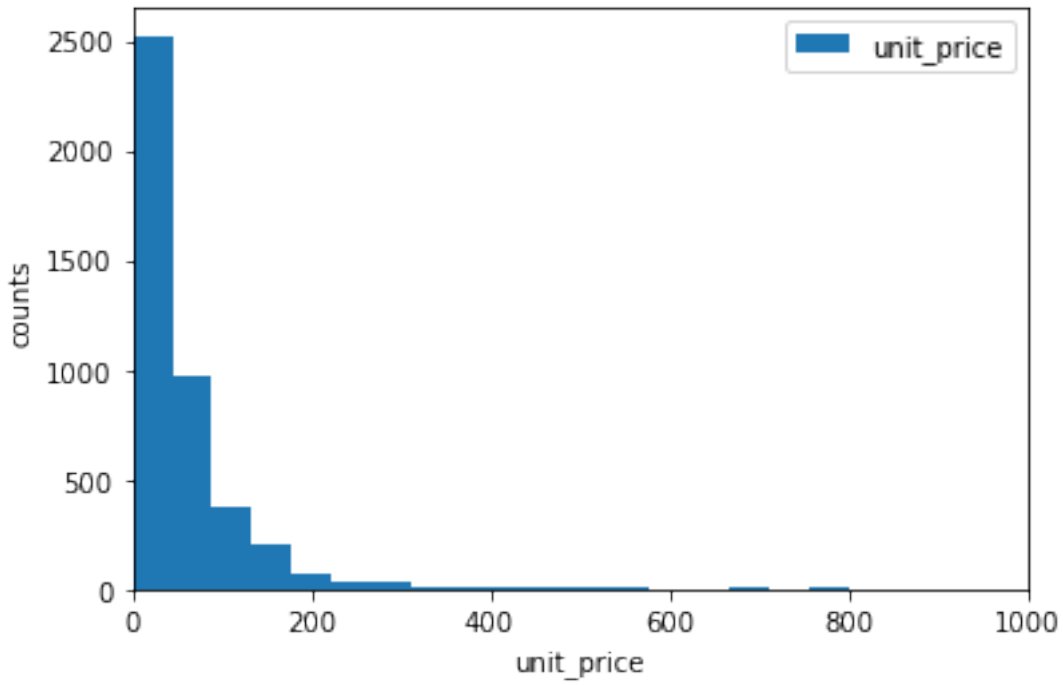
[44]:
```
continuous_cols = [ 'love', 'number_of_reviews', 'price', 'unit_price']
df['love'].plot.hist(bins = 200, xlim=(0.0, 200000))
plt.legend()
plt.ylabel('counts')
plt.xlabel('love')
plt.savefig('../figures/love.jpg', dpi=300)
plt.show()

df['number_of_reviews'].plot.hist(bins = 70, xlim=(0, 7500))
plt.legend()
plt.ylabel('counts')
plt.xlabel('number_of_reviews')
```

```
plt.savefig('../figures/number_of_reviews.jpg', dpi=300)
plt.show()

df['price'].plot.hist(bins = 50, xlim=(0, 400))
plt.legend()
plt.ylabel('counts')
plt.xlabel('price')
plt.savefig('../figures/price.jpg', dpi=300)
plt.show()


df['unit_price'].plot.hist(bins = 100, xlim=(0, 1000))
plt.legend()
plt.ylabel('counts')
plt.xlabel('unit_price')
plt.savefig('../figures/unit_price.jpg', dpi=300)
plt.show()
```




number_of_reviews
counts
7000
6000
5000
4000
3000
2000
1000
0
0
1000
2000
3000
4000
5000
6000
7000
number_of_reviews


price
counts
2500
2000
1500
1000
500
0
0
50
100
150
200
250
300
350
400
price

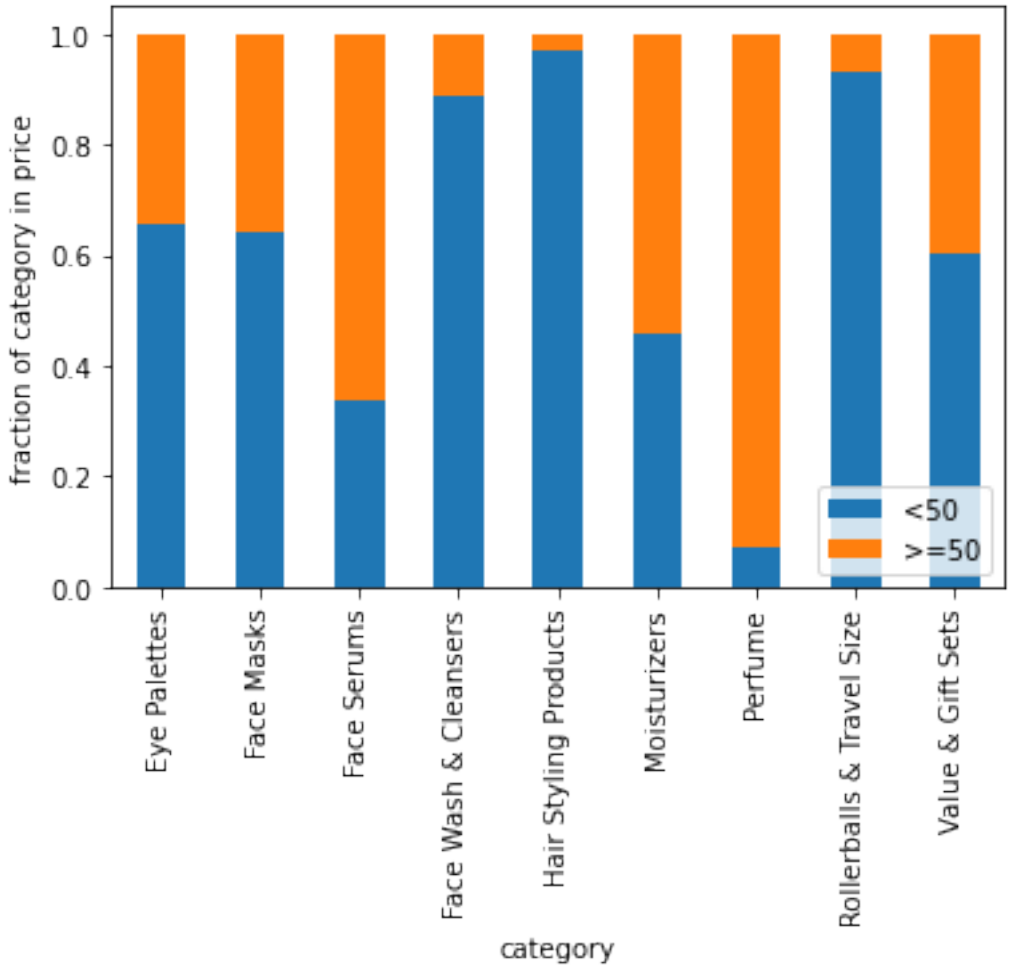


[45]:
```
# Since the data is highly imbalanced, we cannot observe any useful information␣
 ↪from scatter matrix
# pd.plotting.scatter_matrix(df.select_dtypes(int), figsize=(9, 9),␣
 ↪marker='o',hist_kwds={'bins': 50}, s=30, alpha=.1)
# plt.savefig('../figures/scatter_matrix.jpg', dpi=300)
# plt.show()
```

[46]:
```
# Categorical vs Categorical
# Stacked bar plot: price vs category
mean_price = stats.mean(df['price'])
price = df['price']
price_level = []
for i in price:
    if i<50:
        price_level.append('<50')
    else:
        price_level.append('>=50')
new_df = df
new_df['price_level']= price_level
```

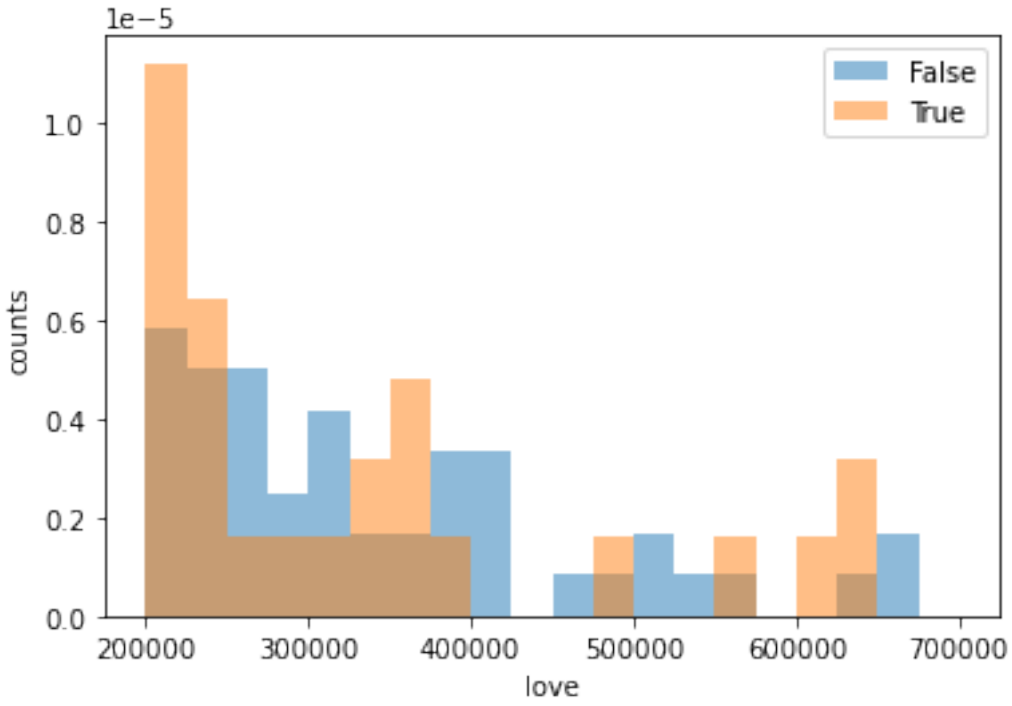
[47]:
```
category = new_df['category'].value_counts()
df_cat = new_df.loc[df['category'].isin(list(category[category>200].index))]
count_matrix = df_cat.groupby(['category', 'price_level']).size().unstack()
count_matrix_norm = count_matrix.div(count_matrix.sum(axis=1),axis=0)
```

```
count_matrix_norm.plot(kind='bar', stacked=True)
plt.ylabel('fraction of category in price')
plt.legend(loc=4)
plt.savefig('../figures/stacked_bar_plot.jpg', dpi=300, bbox_inches = 'tight')
plt.show()
```

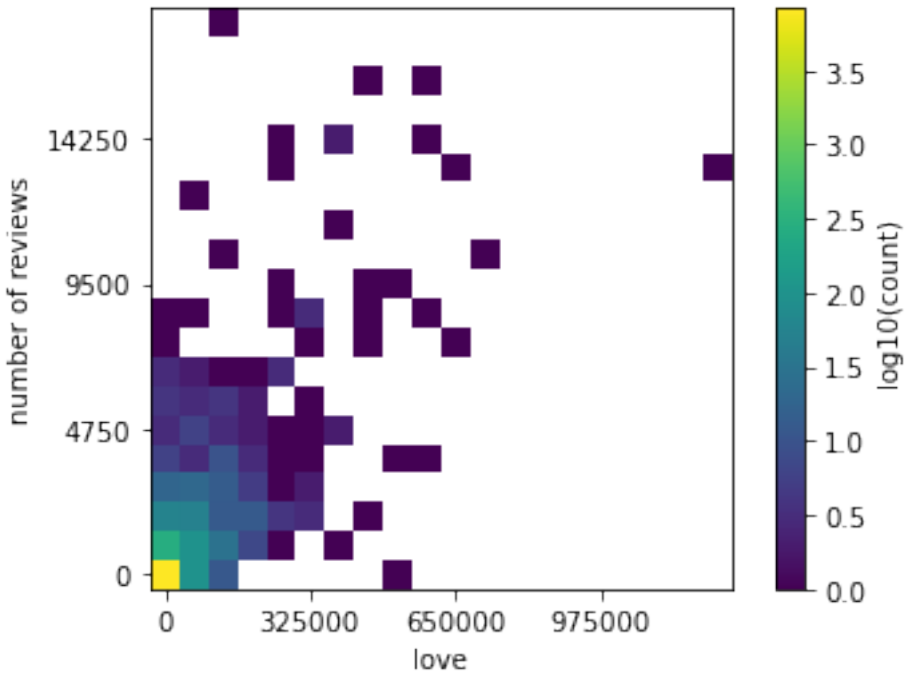


[86]:
```
new_df = df[df['love']>100000]
categories = new_df['MarketingFlags'].unique()
bin_range = (200000,700000)

for c in categories:
    plt.hist(df[df['MarketingFlags']==c]['love'],alpha=0.
 ↪5,label=c,range=bin_range,bins=20,density=True)
plt.legend()
plt.ylabel('counts')
plt.xlabel('love')
```

```
plt.savefig('../figures/compared_histogram.jpg', dpi=300)
plt.show()
```



[49]:
```
nbins = 20
heatmap, xedges, yedges = np.histogram2d(df['love'], df['number_of_reviews'],␣
↪bins=nbins)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]

np.seterr(divide = 'ignore')
plt.imshow(np.log10(heatmap).T, origin='lower') # use log count
plt.xlabel('love')
plt.ylabel('number of reviews')
plt.xticks(np.arange(nbins)[::int(nbins/4)],xedges[::int(nbins/4)].astype(int))
plt.yticks(np.arange(nbins)[::int(nbins/4)],yedges[::int(nbins/4)].astype(int))
plt.colorbar(label='log10(count)')
plt.savefig('../figures/heatmap.png', dpi=300)
plt.show()
```



[78]:
```
import pandas as pd
import numpy as np

from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder, OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold, KFold

# let's separate the feature matrix X, and target variable y
df2 = df[df_cat.columns.difference(['brand', 'id', 'name', 'size',␣
 ↪'value_price', 'price_level', 'unit_price'])]
X = df2.loc[:,df2.columns!='rating']
y = df2.loc[:,df2.columns=='rating']
random_state = 63

# collect which encoder to use on each feature
std_ftrs = ['love', 'number_of_reviews', 'price', 'liquid_size']
onehot_ftrs = ['MarketingFlags', 'category', 'exclusive', 'limited_edition',␣
 ↪'limited_time_offer','online_only']

# collect all the encoders
preprocessor = ColumnTransformer(transformers=[('onehot',␣
 ↪OneHotEncoder(sparse=False,handle_unknown='ignore'), onehot_ftrs),
```

```
                                                ('std', StandardScaler(),␣
 ↪std_ftrs)])
clf = Pipeline(steps=[('preprocessor', preprocessor)])

# split to separate out the training, validation and testing set
X_other, X_test, y_other, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state=random_state)
print('test balance:',y_test['rating'].value_counts(normalize=True))

kf = KFold(n_splits=5,shuffle=True,random_state=random_state)

for train_index, val_index in kf.split(X_other,y_other):
    X_train = X_other.iloc[train_index]
    y_train = y_other.iloc[train_index]
    X_val = X_other.iloc[val_index]
    y_val = y_other.iloc[val_index]

    X_train_prep = clf.fit_transform(X_train)
    X_val_prep = clf.transform(X_val)
    X_test_prep = clf.transform(X_test)

    print("training set:", X_train_prep.shape)
    print("validation set:", X_val_prep.shape)
    print("test set:", X_test_prep.shape)
```

```
test balance: 4.5    0.354417
4.0    0.309160
3.5    0.121592
5.0    0.103053
0.0    0.053980
3.0    0.037077
2.5    0.012541
2.0    0.005998
1.0    0.002181
Name: rating, dtype: float64
training set: (5867, 153)
validation set: (1467, 153)
test set: (1834, 153)
training set: (5867, 152)
validation set: (1467, 152)
test set: (1834, 152)
training set: (5867, 151)
validation set: (1467, 151)
test set: (1834, 151)
training set: (5867, 151)
validation set: (1467, 151)
test set: (1834, 151)
```

```
training set: (5868, 155)
validation set: (1466, 155)
test set: (1834, 155)
```

[65]:
```
print('data dimensions:',df.shape)
perc_missing_per_ftr = df.isnull().sum(axis=0)/df.shape[0]
perc_missing_per_ftr
```

```
data dimensions: (9168, 14)
```

[65]:
```
MarketingFlags        0.00000
category              0.00000
exclusive             0.00000
limited_edition       0.00000
limited_time_offer    0.00000
love                  0.00000
number_of_reviews     0.00000
online_only           0.00000
price                 0.00000
rating                0.00000
size                  0.00000
value_price           0.00000
liquid_size           0.52836
unit_price            0.52836
dtype: float64
```

[ ]:
```

```