Web Scraping Assignment.

Lenskart Product Sales,Rating and things.

Dayashankar Mehta PRN - 19030141013 MBA-IT Sem 3rd

In [1]:
```
import requests

from bs4 import BeautifulSoup
```

In [2]:
```
url = "https://www.lenskart.com/"
```

In [3]:
```
headers = {'User-Agent':"Mozilla/5.0 (Windows NT 5.1; rv:7.0.1) Gecko/20100101 F
```

In [4]:
```
import re
Brand = []
Bought = []
Selling_Price = []
Market_Price = []
Rating = []
Product_links = []
for x in range(1,20):
    r = requests.get('https://www.lenskart.com/eyeglasses/collections/all-compute

    soup = BeautifulSoup(r.text, 'html.parser')

    for item in soup.find_all(class_="productWidgetBox border"):
        brand = item.find('div', class_="col-md-12 no-padding text-color-black fw
        Brand.append(brand)
        try:
            bought = item.find('span', class_="display-ib text-color-grey border-
            Bought.append(bought)
        except:
            bought = 0
            Bought.append(bought)


        sellPrice = item.find('span', class_="text-color-black fw700").get_text()
        Selling_Price.append(sellPrice)
        try:
            marketPrice = item.find('span', class_="text-color-grey strike-throug
            Market_Price.append(marketPrice)
        except:
            marketPrice = None
            Market_Price.append(marketPrice)
        try:
            rating = item.find('span', class_="text-color-white bg-color-green di
            Rating.append(rating)
        except:
            rating = 0
            Rating.append(rating)
        for link in item.find_all('a', href=True):
            Product_links.append(url + link['href'])
        #print(item)
        #print(Productlinks)
        #print(Bought)
```

In [5]:
```
len(Brand)
```

Out[5]: 171

In [6]:
```
#Bought = re.findall('\d+', ' '.join(Bought))
#Bought
```

In [7]:
```
#removing None value and extract only Number and clean string"bought"
Bought
try:
    while True:
        Bought[Bought.index(0)] = '0 bought'
except ValueError:
    pass

Bought = re.findall('\d+', ' '.join(Bought))
Bought
```

```
 '110',
 '137',
 '121',
 '106',
 '203',
 '106',
 '81',
 '88',
 '682',
 '195',
 '94',
 '66',
 '98',
 '72',
 '1048',
 '931',
 '16',
 '0',
 '23',
```

In [8]:
```
Selling_Price
```

Out[8]:
```
['₹999',
 '₹999',
 '₹999',
 '₹999',
 '₹999',
 '₹999',
 '₹1500',
 '₹999',
 '₹1500',
 '₹999',
 '₹999',
 '₹999',
 '₹999',
 '₹1500',
 '₹999',
 '₹999',
 '₹999',
 '₹999',
 '₹999',
```

In [9]:
```
#cleaning preceding term Rs.
Selling_Price = re.findall('\d+', ' '.join(Selling_Price))
Selling_Price
```

Out[9]:
```
['999',
 '999',
 '999',
 '999',
 '999',
 '999',
 '1500',
 '999',
 '1500',
 '999',
 '999',
 '999',
 '999',
 '1500',
 '999',
 '999',
 '999',
 '999',
 '999',
```

In [10]:
```
#cleaning preceding term Rs.
Market_Price = re.findall('\d+', ' '.join(Market_Price))
Market_Price
#len(Market_Price)
```

Out[10]:
```
['3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
 '3500',
```

In [11]:
```
Rating
```

```
 '4.6 ',
 '5 ',
 '4.6 ',
 '3 ',
 '4.6 ',
 '1 ',
 '4.9 ',
 '5 ',
 '5 ',
 '5 ',
 '4 ',
 0,
 '4.5 ',
 '1 ',
 '4.5 ',
 '5 ',
 '4.6 ',
 0,
 '4 ',
 '4.4 ',
```

In [12]:
```
Product_links = set(Product_links)
```

In [26]:
```
Name = []
Product_Id = []
for link in Product_links:
    r = requests.get(link, headers = headers)
    soup = BeautifulSoup(r.content, 'html.parser')
    try:
        name = soup.find('h1', class_='product-fullName padding-t10').get_text()
        Name.append(name)
    except:
        name = None
        Name.append(name)
    try:
        productId = soup.find('span', class_='color-green').get_text()
        Product_Id.append(productId)
    except:
        productId = None
        Product_Id.append(productId)
```

In [27]:
```
#Name
```

In [28]:
```
len(Name)
```

Out[28]: 251

In [29]:
```
#Product_Id
```

In [30]:
```
len(Product_Id )
```

Out[30]: 251

In [31]:
```
Lenskart = {
        'Brand':Brand,
        'Name':Name,
        'Product_Id':Product_Id,
        'Selling_Price':Selling_Price,
        'Market_Price':Market_Price,
        'Rating':Rating,
        'Bought':Bought,
}
```

In [32]:
```
Lenskart
```

```
  'Lenskart Air Computer Glasses',
  'Lenskart Blu',
  'Lenskart Air Computer Glasses',
  'Lenskart Air Computer Glasses',
  'Lenskart Air Computer Glasses',
  'Lenskart Air Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Lenskart Air Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Lenskart Air Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Lenskart Air Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Lenskart Air Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Vincent Chase Computer Glasses',
  'Vincent Chase Computer Glasses',
```

In [33]:
```
import pandas as pd
```

In [34]:
```
df = pd.DataFrame.from_dict(Lenskart, orient = 'index')
df = df.transpose()
dfs = df.head(171)
dfs
```

Out[34]:

	Brand	Name	Product_Id	Selling_Price	Market_Price	Rating	Bought
0	Vincent Chase Computer Glasses	Blue Block Phone & Computer Glasses: Golden Bl...	143656	999	3500	4.6	896
1	Vincent Chase Computer Glasses	Blue Block Phone & Computer Glasses: Tortoise ...	133212	999	3500	4.4	756
2	Lenskart Blu	Blue Block Phone & Computer Glasses: Transpare...	138877	999	3500	4.4	1849
3	Lenskart Air Computer Glasses	Blue Block Phone & Computer Glasses: Gunmetal ...	142708	999	3500	4.8	765
4	Lenskart Air Computer Glasses	Blue Block Phone & Computer Glasses: Matte Bla...	141855	999	3500	4.5	455
...	...	...	...	...	...	...	...
166	Lenskart Air Computer Glasses	Blue Block Phone & Computer Glasses: Matte Gol...	144479	999	3500	5	4
167	Lenskart Air Computer Glasses	Blue Block Phone & Computer Glasses: Matte Bla...	144473	999	3500	5	3
168	Lenskart Air Computer Glasses	None	None	999	3500	2	3
169	Lenskart Air Computer Glasses	Blue Block Phone & Computer Glasses: Matte Gre...	144304	999	3500	5	0
170	Lenskart Air Computer Glasses	Blue Block Phone & Computer Glasses: Matte Ger...	144217	999	3500	4.8	13

171 rows × 7 columns

In [35]:
```
dfs.to_csv(r'C:\Users\win\Desktop\Lenskart.csv', index=False)
```

In [36]:
```
dfs = pd.read_csv('C:\\Users\\win\\Desktop\\Lenskart.csv')
```

In [37]:
```
dfs
```

	Brand	Name	Product_id	Selling_Price	Market_Price	Rating	Bought
0	Vincent Chase Computer Glasses	Blue Block Phone & Computer Glasses: Golden Bl...	143656.0	999	3500	4.6	896
1	Vincent Chase Computer Glasses	Blue Block Phone & Computer Glasses: Tortoise ...	133212.0	999	3500	4.4	756
2	Lenskart Blu	Blue Block Phone & Computer Glasses: Transpare...	138877.0	999	3500	4.4	1849
3	Lenskart Air Computer Glasses	Blue Block Phone & Computer Glasses: Gunmetal ...	142708.0	999	3500	4.8	765
4	Lenskart Air Computer Glasses	Blue Block Phone & Computer Glasses: Matte Bla...	141855.0	999	3500	4.5	455
...	...	...	...	...	...	...	...

Implementing Multiple Linear Regression on Lenskart Dataset

In [71]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

In [72]:
```
#Extracting useful columns to predict bought item
x=dfs.iloc[:,[3,4,5]].values
x
```

Out[72]:
```
array([[9.990e+02, 3.500e+03, 4.600e+00],
       [9.990e+02, 3.500e+03, 4.400e+00],
       [9.990e+02, 3.500e+03, 4.400e+00],
       [9.990e+02, 3.500e+03, 4.800e+00],
       [9.990e+02, 3.500e+03, 4.500e+00],
       [9.990e+02, 3.500e+03, 4.700e+00],
       [1.500e+03, 3.500e+03, 4.300e+00],
       [9.990e+02, 3.500e+03, 4.400e+00],
       [1.500e+03, 3.500e+03, 4.600e+00],
       [9.990e+02, 3.500e+03, 4.500e+00],
       [9.990e+02, 3.500e+03, 4.500e+00],
       [9.990e+02, 3.500e+03, 4.500e+00],
       [9.990e+02, 3.500e+03, 4.800e+00],
       [1.500e+03, 3.500e+03, 4.500e+00],
       [9.990e+02, 3.500e+03, 4.600e+00],
       [9.990e+02, 3.500e+03, 4.600e+00],
       [9.990e+02, 3.500e+03, 4.500e+00],
       [9.990e+02, 3.500e+03, 4.400e+00],
       [9.990e+02, 3.500e+03, 4.700e+00],
```

In [73]:
```
y=dfs.iloc[:,6].values
y
```

Out[73]:
```
array([ 896,  756, 1849,  765,  455,  397,  421,  887,  105,  430,  244,
        301,  253,  484,  409,  419,  261,  205,  254, 1170,  634, 1918,
        158,  588, 2776,  330,  233,  147,  169,  183,  189,  168,  283,
        375,  291,  664,  491,  174,  145,  153,  185,  140,  139,  114,
        184,  218,  195,  155,  187,  111,  109,  110,  110,  137,  121,
        106,  203,  106,   81,   88,  682,  195,   94,   66,   98,   72,
       1048,  931,   16,    0,   23,    0,   14,  140,   39,   84,   13,
          2,    9,   31,    3,    2,   68,    8,    9,    8,   15,    0,
          6,    3,    0,    2,    6,    0,   11,   47,    6,   16,   64,
         74,  143,  151,   82,  199,   28,   57,   22,   16,   47,  111,
         44,   42,   11,   74,  143,  151,   82,   74,   22,   61,  167,
         54,   11,   64,   77,   64,   31,   51,   16,    5,   46,   81,
         44,   21,  105,   39,    1,   23,   21,    0,    8,    0,    0,
          0,   17,   12,   56,    8,   32,    0,    0,    0,    0,    1,
         12,    4,    7,    4,    3,    3,    0,   13,    1,   12,    4,
          7,    4,    3,    3,    0,   13], dtype=int64)
```

In [76]:
```
#split data 25% test data and 75% train data
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4,
                                                    random_state=1)
```

In [77]:
```
reg = linear_model.LinearRegression()
reg.fit(x_train, y_train)
```

Out[77]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [78]:
```
reg.coef_
```

Out[78]:
```
array([-1.40502094e-01,  6.44001059e-03,  2.74521647e+01])
```

In [79]:
```
reg.score(x_test, y_test)
```

Out[79]:
```
-0.013217899546207335
```

In [82]:
```
#predict the Bought of item with our model
prd = reg.predict([[1000,5000,4]])
int(prd)
```

Out[82]:
```
233
```

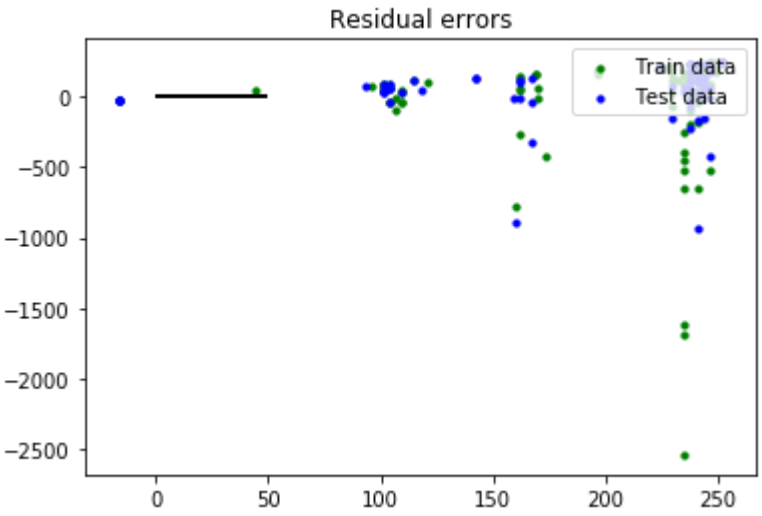
In [83]:
```
prd = reg.predict([[1000,5000,0]])
int(prd)
```

Out[83]:
```
123
```

In [84]:
```
prd = reg.predict([[999,3000,4]])
int(prd)
```

Out[84]:
```
220
```

In [85]:
```
## plotting residual errors in training data
plt.scatter(reg.predict(x_train), reg.predict(x_train) - y_train,
            color = "green", s = 10, label = 'Train data')

## plotting residual errors in test data
plt.scatter(reg.predict(x_test), reg.predict(x_test) - y_test,
            color = "blue", s = 10, label = 'Test data')

## plotting line for zero residual error
plt.hlines(y = 0, xmin = 0, xmax = 50, linewidth = 2)

## plotting legend
plt.legend(loc = 'upper right')

## plot title
plt.title("Residual errors")

## function to show plot
plt.show()
```



In [ ]: