به نام خدا

پروژه‌ی درس مدیریت ارتباط با مشتریان - تحلیل اطلاعات یک فروشگاه کتاب صوتی

۷ بهمن ۱۴۰۰

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from kneed import KneeLocator
from scipy import stats
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn import preprocessing
from sklearn import svm
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB, BernoulliNB
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import silhouette_score
```

[2]:
```
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
sns.set_theme(style="white", context="talk")
```

۱ پیش‌پردازش

ابتدا باید با کتابخانه‌ی پانداس اطلاعات دیتاست ساختاردهی شوند.

۱.۱ بارگزاری داده و بررسی آن

[4]:
```
df = pd.read_csv('CRM-Project-Noorbehbahani-Dataset.csv')
df.head()
```

[4]:

	id	Book_length(mins)_overall	Book_length(mins)_avg	Price_overall	Price_avg	Review	Review10/10	Completion	Minutes_listened	Support_Request	Last_Visited_mins_Purchase_date	Target
0	994	1620.0	1620	19.73	19.73	1	10.0	0.99	1603.8	5	92	0
1	1143	2160.0	2160	5.33	5.33	0	NaN	0.00	0.0	0	0	0
2	2059	2160.0	2160	5.33	5.33	0	NaN	0.00	0.0	0	388	0
3	2882	1620.0	1620	5.96	5.96	0	NaN	0.42	680.4	1	129	0
4	3342	2160.0	2160	5.33	5.33	0	NaN	0.22	475.2	0	361	0

۲.۱ حذف id به دلیل بی‌اهمیت‌بودن آن

[5]:
```
df = df.drop('id', axis=1)
```

۳.۱ مشکل تداخل دو ستون Review و Review10/10

به این خاطر که بر اساس ستون Review10/10 می‌توان به ستون Review رسید و البته این دو ستون با هم تداخل دارند، چهار کار انجام میشود.

۱. ستون Review چون اطلاعات مفیدی ندارد حذف میشود.

۲. مقدارهای ناموجود در Review10/10 با Unknown جایگذاری می‌شود.

۳. مقدارهای Review10/10 به شکل گسسته و بر اساس معیار NPS تبدیل می‌شود. البته چون در مورد ماتریس همبستگی و عملیات‌های دیگر نمی‌توان از مقدارهای گسسته استفاده کرد، هر مقدار گسسته با یک عدد جایگزین می‌شود. به این شکل که:

- مقدار ۱ جایگزین هاDetractor است.
- مقدار ۲ جایگزین هاPassive است.
- مقدار ۳ جایگزین هاPromoter است.
- مقدار ۰ جایگزین مقادیر گمشده است.

۴. اسم ستون Review10/10 به Net Promoter Score تغییر پیدا می‌کند.

[6]:
```
df = df.drop(labels='Review', axis=1)
df['Review10/10'] = df['Review10/10'].fillna(11)
df['Review10/10'].value_counts()
```

[6]:
```
11.00    11616
10.00     1284
8.00       404
9.00       381
7.00       157
6.00       104
5.00        43
9.50        21
4.00        18
8.50        11
1.00        10
3.00         9
2.00         7
6.50         5
```

```
8.67         2
7.50         2
5.50         2
8.33         2
4.50         1
9.67         1
9.40         1
1.50         1
6.67         1
7.75         1
Name: Review10/10, dtype: int64
```

[7]:
```
nps_labels = [1, 2, 3, 0] # 'Detractor', 'Passive', 'Promoter', 'Unknown'
nps_values = [0, 6, 8, 10, 11]
df['Net Promoter Score'] = pd.cut(
    df['Review10/10'],
    bins=nps_values,
    labels=nps_labels
).astype('int64')
df = df.drop(labels='Review10/10', axis=1)
df['Net Promoter Score'].value_counts()
```

[7]:
```
0    11616
3     1703
2      570
1      195
Name: Net Promoter Score, dtype: int64
```

۴.۱ مشکل بزرگتر بودن میانگین خرید از مجموع خرید و به همین شکل در مورد طول هر کتاب و میانگین آن

همانطور که در کلاس اشاره شد، لیبل ستون‌های Price_Overall و Price_avg .باید جابه‌جا شود. در مورد «طول زمان کتاب‌ها» هم همین مورد وجود دارد

[8]:
```
```

```
df = df.rename(columns={'Price_overall': 'Price_avg', 'Price_avg':␣
↪'Price_overall'})
df = df.rename(columns={'Book_length(mins)_overall': 'Book_length(mins)_avg',␣
↪'Book_length(mins)_avg': 'Book_length(mins)_overall'})
df.head(5)
```

[8]:

	Book_length(mins)_avg	Book_length(mins)_overall	Price_avg	Price_overall	\
0	1620.0	1620	19.73	19.73	
1	2160.0	2160	5.33	5.33	
2	2160.0	2160	5.33	5.33	
3	1620.0	1620	5.96	5.96	
4	2160.0	2160	5.33	5.33	

	Completion	Minutes_listened	Support_Request	\
0	0.99	1603.8	5	
1	0.00	0.0	0	
2	0.00	0.0	0	
3	0.42	680.4	1	
4	0.22	475.2	0	

	Last_Visited_mins_Purchase_date	Target	Net Promoter Score
0	92	0	3
1	0	0	0
2	388	0	0
3	129	0	0
4	361	0	0

۵.۱ بهبود عنوان‌های هر ستون

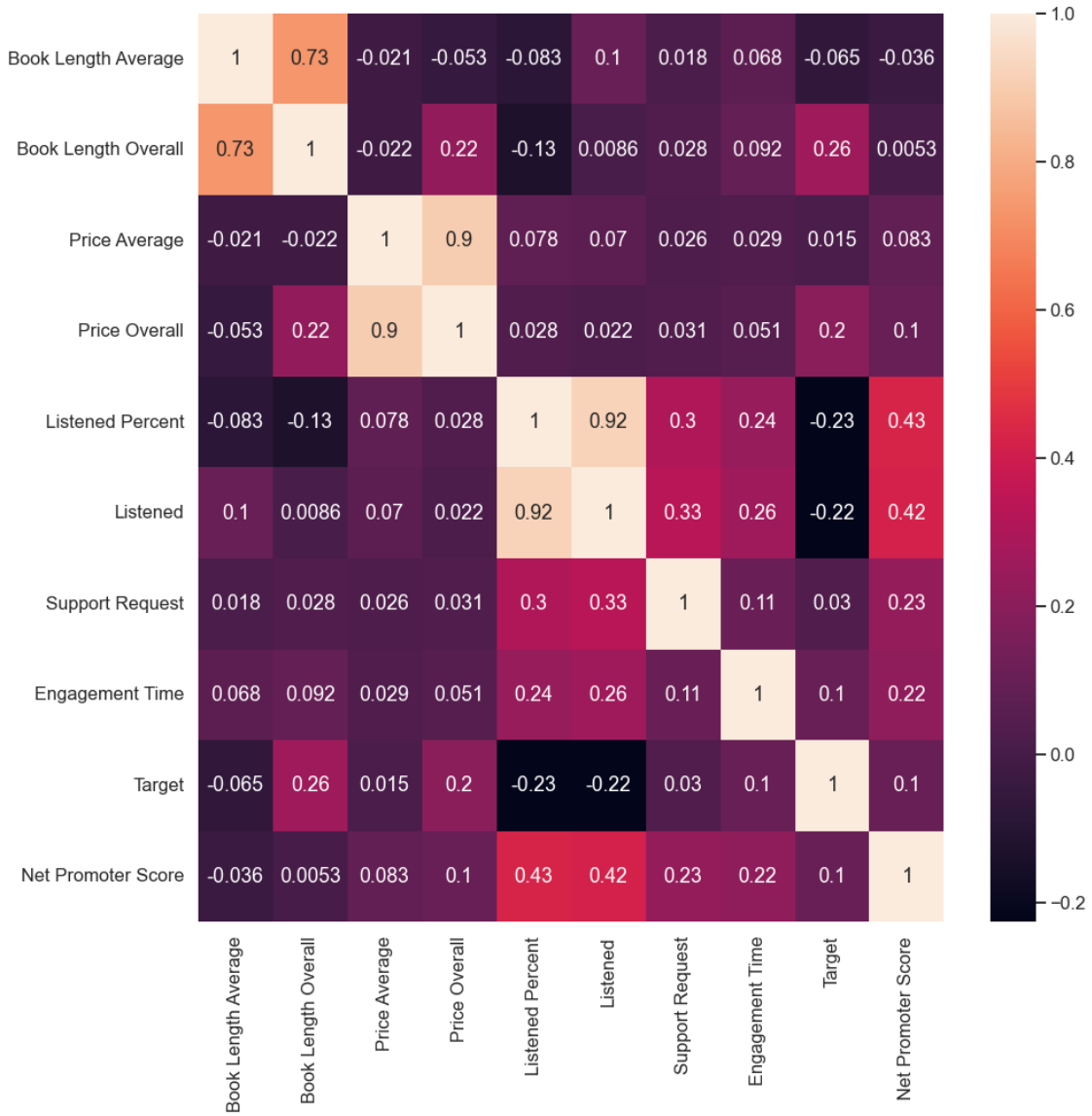
برای هر ستون لیبل‌های بهتری قرار می‌دهم تا کارکردن با این دیتاست راحت‌تر باشد.

[9]:
```
df = df.rename(columns={
    'Book_length(mins)_overall': 'Book Length Overall',
    'Book_length(mins)_avg': 'Book Length Average',
    'Price_overall' : 'Price Overall',
```

```
    'Price_avg': 'Price Average',
    'Completion': 'Listened Percent',
    'Minutes_listened': 'Listened',
    'Support_Request': 'Support Request',
    'Last_Visited_mins_Purchase_date': 'Engagement Time'
})
```

۶.۱ بررسی همبستگی بین ویژگی‌ها

[10]:
```
plt.figure(figsize=(15,15))
corrMatrix = df.corr()
sns.heatmap(corrMatrix, annot=True)
plt.show()
```



همانطور که از ماتریس همبستگی بین ویژگی‌های مختلف می‌توان متوجه شد، بین ویژگی‌های «میانگین قیمت» و «قیمت کل»، «زمانی که به کتاب صوتی گوش داده‌شده» و «درصد تکمیل آن» و همچنین بین «کل زمان‌های کتاب‌هایی که خریده» و «میانگین آن» وابستگی زیادی وجود دارد که قابل پیش‌بینی بود. در مورد ویژگی «هدف» با سایر ویژگی‌ها، چیزی که به چشم می‌آید ارتباط نسبتا معکوس بین ویژگی «هدف» با «میزانی که به کتاب گوش داده‌اند» و «درصد تکمیل آن» است و از سمت دیگر، وابستگی کمی بین «کل زمان کتاب‌هایی که خریده‌اند» و ویژگی «هدف» وجود دارد.

در مورد ویژگی «هدف»، چیزی که دریافت می‌شود این است که اگر کاربران کتاب‌های طولانی‌تری خریداری کنند، احتمال خرید مجدد آن‌ها بیشتر است ولی هر چقدر به کتاب‌ها بیشتر گوش بدهند، احتمال خرید مجدد کمی دارند. شاید این مورد به خاطر این است که نیاز کسانی که کتاب‌ها را گوش کرده‌اند، کاهش یافته‌است.

از سمت دیگر، کسانی که احتمال خرید دارند، هم بیشتر نظر داده‌اند و هم نظر بهتری داده‌اند. یعنی صرفا بر اساس ویژگی NPS نیز می‌توان احتمال خوبی از خرید مجدد افراد به دست آورد.

۷.۱ نهایی‌کردن پیش‌پردازش

[11]:
```
clean_df = df.copy()
clean_df.head(10)
```

[11]:

	Book Length Average	Book Length Overall	Price Average	Price Overall	\
0	1620.0	1620	19.73	19.73	
1	2160.0	2160	5.33	5.33	
2	2160.0	2160	5.33	5.33	
3	1620.0	1620	5.96	5.96	
4	2160.0	2160	5.33	5.33	
5	2160.0	2160	4.61	4.61	
6	2160.0	2160	5.33	5.33	
7	648.0	648	5.33	5.33	
8	2160.0	2160	5.33	5.33	
9	2160.0	2160	5.33	5.33	

	Listened Percent	Listened	Support Request	Engagement Time	Target	\
0	0.99	1603.8	5	92	0	
1	0.00	0.0	0	0	0	
2	0.00	0.0	0	388	0	
3	0.42	680.4	1	129	0	
4	0.22	475.2	0	361	0	
5	0.00	0.0	0	0	0	
6	0.04	86.4	0	366	0	
7	0.00	0.0	0	0	1	
8	0.26	561.6	0	33	0	
9	0.27	583.2	0	366	0	

	Net Promoter Score
0	3
1	0
2	0

3	0
4	0
5	0
6	0
7	0
8	0
9	3

[12]:
```
clean_df.describe()
```

[12]:

	Book Length Average	Book Length Overall	Price Average	Price Overall
count	14084.000000	14084.000000	14084.000000	14084.000000
mean	1591.281685	1678.608634	7.103791	7.543805
std	504.340663	654.838599	4.931673	5.560129
min	216.000000	216.000000	3.860000	3.860000
25%	1188.000000	1188.000000	5.330000	5.330000
50%	1620.000000	1620.000000	5.950000	6.070000
75%	2160.000000	2160.000000	8.000000	8.000000
max	2160.000000	7020.000000	130.940000	130.940000

	Listened Percent	Listened	Support Request	Engagement Time
count	14084.000000	14084.000000	14084.000000	14084.000000
mean	0.125659	189.888983	0.070222	61.935033
std	0.241206	371.084010	0.472157	88.207634
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	11.000000
75%	0.130000	194.400000	0.000000	105.000000
max	1.000000	2160.000000	30.000000	464.000000

	Target	Net Promoter Score
count	14084.000000	14084.000000
mean	0.158833	0.457540
std	0.365533	1.026995
min	0.000000	0.000000

25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	1.000000	3.000000

۲ بخش‌بندی کاربران

برای اطمینان از اینکه دیتافریم اصلی دچار تغییر نمی‌شود، یک دیتافریم جدید می‌سازیم.

[13]:
```
rfm_df = clean_df.copy()
```

۱.۲ بخش‌بندی بر اساس RFM

برای بخش‌بندی بر اساس RFM به سه پارامتر زیر نیاز داریم:

- Recency یا تازگی
- Frequency یا تعداد خرید
- Monetary یا مبلغ پولی خرید

برای مورد اول میتوان از معیار Engagement Time استفاده کرد. برای مورد سوم از Price Overall و در مورد «تعداد خرید» کمی محاسبه نیاز است.

۱.۱.۲ محاسبه‌ی Purchases Count

دو راه برای محاسبه‌ی «تعداد خرید» وجود دارد.

$$PurchasesCount = \frac{PriceOverall}{PriceAverage}$$

$$PurchasesCount = \frac{BookLengthOverall}{BookLengthAverage}$$

منطقا خروجی حاصل از این دو روش نباید با یکدیگر مغایر باشد.

برای بررسی، مقدار هر دو محاسبه می‌شود و با یکدیگر مقایسه می‌شوند.

[14]:
```
counter = 0
for index, row in rfm_df.iterrows():
    purchases_count_based_on_price = row['Price Overall'] / row['Price Average']
```

```
    purchases_count_based_on_book_length = row['Book Length Overall'] /␣
 ↪row['Book Length Average']
    if(purchases_count_based_on_price != purchases_count_based_on_book_length):
        counter = counter + 1
        if(counter < 10):
            print(row['Price Overall'], row['Price Average'], row['Book Length␣
 ↪Overall'], row['Book Length Average'], purchases_count_based_on_price,␣
 ↪purchases_count_based_on_book_length)
print(counter)
```

```
58.67 29.33 1188.0 594.0 2.0003409478349816 2.0
23.99 6.0 4752.0 1188.0 3.998333333333333 4.0
55.99 27.99 2808.0 1404.0 2.000357270453734 2.0
17.47 8.73 1512.0 756.0 2.001145475372279 2.0
38.4 12.8 3564.0 1188.0 2.9999999999999996 3.0
20.27 10.13 3780.0 1890.0 2.0009871668311945 2.0
10.67 5.33 2484.0 1242.0 2.0018761726078798 2.0
13.33 6.66 2808.0 1404.0 2.0015015015015014 2.0
28.69 9.56 3348.0 1116.0 3.0010460251046025 3.0
469
```

مشاهده می‌شود که در ۴۶۹ مورد این اتفاق افتاده‌است. اگر کل تضادها بررسی شوند متوجه می‌شویم که در حالت مبتنی بر قیمت، عددها در حد اعشار با حالت مبتنی بر طول زمان کتاب‌ها اختلاف دارند و این نتیجه گرفته می‌شود که در داده‌های مرتبط با قیمت بی‌دقتی رخ داده یا به هر حال داده‌های صحیحی نیستند. این می‌تواند به خاطر تخفیف یا محاسبات اشتباه باشد.

به همین خاطر از روش مبتنی بر طول هر کتاب استفاده می‌شود تا تعداد خریدهای فرد بررسی شود.

[15]:
```
rfm_df['Purchases Count'] = rfm_df['Book Length Overall'] / rfm_df['Book Length␣
 ↪Average']
```

مورد بعدی این است که Engagetime به طور خام قابل استفاده نیست. از روی این مقدار، مقدار Recency محاسبه می‌شود.

[16]:
```
rfm_df['Recency'] = rfm_df['Engagement Time'].max() - rfm_df['Engagement Time']
rfm_df.describe()
```

[16]:

	Book Length Average	Book Length Overall	Price Average	Price Overall
count	14084.000000	14084.000000	14084.000000	14084.000000
mean	1591.281685	1678.608634	7.103791	7.543805
std	504.340663	654.838599	4.931673	5.560129
min	216.000000	216.000000	3.860000	3.860000
25%	1188.000000	1188.000000	5.330000	5.330000
50%	1620.000000	1620.000000	5.950000	6.070000
75%	2160.000000	2160.000000	8.000000	8.000000
max	2160.000000	7020.000000	130.940000	130.940000

	Listened Percent	Listened	Support Request	Engagement Time
count	14084.000000	14084.000000	14084.000000	14084.000000
mean	0.125659	189.888983	0.070222	61.935033
std	0.241206	371.084010	0.472157	88.207634
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	11.000000
75%	0.130000	194.400000	0.000000	105.000000
max	1.000000	2160.000000	30.000000	464.000000

	Target	Net Promoter Score	Purchases Count	Recency
count	14084.000000	14084.000000	14084.000000	14084.000000
mean	0.158833	0.457540	1.063689	402.064967
std	0.365533	1.026995	0.330884	88.207634
min	0.000000	0.000000	1.000000	0.000000
25%	0.000000	0.000000	1.000000	359.000000
50%	0.000000	0.000000	1.000000	453.000000
75%	0.000000	0.000000	1.000000	464.000000
max	1.000000	3.000000	7.000000	464.000000

۲.۱.۲ استخراج هر بخش

[17]:
```
r = 'Recency'
f = 'Purchases Count'
m = 'Price Overall'
r_rank = 'Recency Rank'
```

```
f_rank = 'Purchases Count Rank'
m_rank = 'Price Overall Rank'
```

[18]:
```
rfm_df[r].value_counts()
```

[18]:
```
464    5493
463     357
462     198
461     165
459     140
       ...
95        1
97        1
125       1
85        1
101       1
Name: Recency, Length: 371, dtype: int64
```

[19]:
```
rfm_df[f].value_counts()
```

[19]:
```
1.0    13440
2.0      463
3.0      127
4.0       43
5.0        5
6.0        5
7.0        1
Name: Purchases Count, dtype: int64
```

[20]:
```
rfm_df[m].value_counts()
```

[20]:
```
5.33     5074
8.00     2115
10.13     782
7.99      339
5.87      313
          …
33.07       1
10.05       1
15.89       1
18.65       1
16.87       1
Name: Price Overall, Length: 476, dtype: int64
```

[21]:
```
rfm_df[r_rank] = rfm_df[r].rank(ascending=True, method='first')
rfm_df[f_rank] = rfm_df[f].rank(ascending=False, method='first')
rfm_df[m_rank] = rfm_df[m].rank(ascending=False, method='first')
```

[22]:
```
# copy of data frame for future use
sorted_rfm_df = rfm_df.copy()

labels = [1, 2, 3]
rfm_df['R Segment'] = pd.qcut(rfm_df[r_rank], 3, labels=labels)
rfm_df['F Segment'] = pd.qcut(rfm_df[f_rank], 3, labels=labels)
rfm_df['M Segment'] = pd.qcut(rfm_df[m_rank], 3, labels=labels)
rfm_df.head(10)
```

[22]:

	Book Length Average	Book Length Overall	Price Average	Price Overall	\
0	1620.0	1620	19.73	19.73	
1	2160.0	2160	5.33	5.33	
2	2160.0	2160	5.33	5.33	
3	1620.0	1620	5.96	5.96	
4	2160.0	2160	5.33	5.33	
5	2160.0	2160	4.61	4.61	
6	2160.0	2160	5.33	5.33	

7	648.0	648	5.33	5.33
8	2160.0	2160	5.33	5.33
9	2160.0	2160	5.33	5.33

	Listened Percent	Listened	Support Request	Engagement Time	Target	\
0	0.99	1603.8	5	92	0	
1	0.00	0.0	0	0	0	
2	0.00	0.0	0	388	0	
3	0.42	680.4	1	129	0	
4	0.22	475.2	0	361	0	
5	0.00	0.0	0	0	0	
6	0.04	86.4	0	366	0	
7	0.00	0.0	0	0	1	
8	0.26	561.6	0	33	0	
9	0.27	583.2	0	366	0	

	Net Promoter Score	Purchases Count	Recency	Recency Rank	\
0	3	1.0	372	3786.0	
1	0	1.0	464	8592.0	
2	0	1.0	76	2.0	
3	0	1.0	335	3002.0	
4	0	1.0	103	20.0	
5	0	1.0	464	8593.0	
6	0	1.0	98	13.0	
7	0	1.0	464	8594.0	
8	0	1.0	431	5567.0	
9	3	1.0	98	14.0	

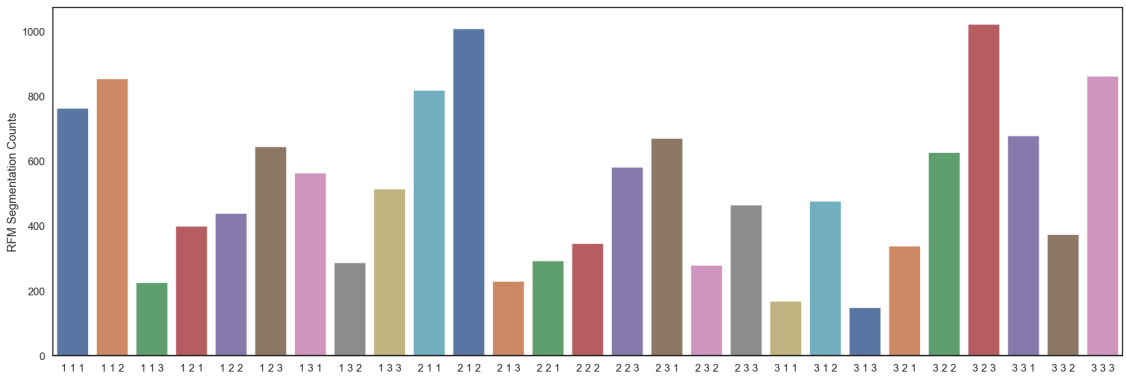
	Purchases Count Rank	Price Overall Rank	R Segment	F Segment	M Segment
0	645.0	277.0	1	1	1
1	646.0	8633.0	2	1	2
2	647.0	8634.0	1	1	2
3	648.0	7262.0	1	1	2
4	649.0	8635.0	1	1	2
5	650.0	13846.0	2	1	3
6	651.0	8636.0	1	1	2

7	652.0	8637.0	2	1	2
8	653.0	8638.0	2	1	2
9	654.0	8639.0	1	1	2

[23]:
```
segments_labels = []
segments_counts = []
for i in range(1, 4):
    for j in range(1, 4):
        for k in range(1, 4):
            count = len(rfm_df[(rfm_df['R Segment'] == i) & (rfm_df['F␣
 ↪Segment'] == j) & (rfm_df['M Segment'] == k)])
            label = f'{i} {j} {k}'
            segments_labels.append(label)
            segments_counts.append(count)
```

[24]:
```
_, ax = plt.subplots(1, 1, figsize=(30, 10), sharex=True)
sns.barplot(x=segments_labels, y=segments_counts, palette="deep", ax=ax)
ax.axhline(0, color="k", clip_on=False)
ax.set_ylabel("RFM Segmentation Counts")
```

[24]: Text(0, 0.5, 'RFM Segmentation Counts')



این نمودار نشان می‌دهد که بخش مطلوب ما یعنی ۱۱۱ تعداد خوبی از مشتریان را در خود جای داده‌است.

با توجه به اینکه ۴ بخش از ۶ بخش‌بندی پر تعداد مربوط به F=1 هستند، می‌توان گفت که معیار Frequency یا «تعداد خریدهای

مشتریان» بیشترین تاثیر در بخش‌بندی را دارد.

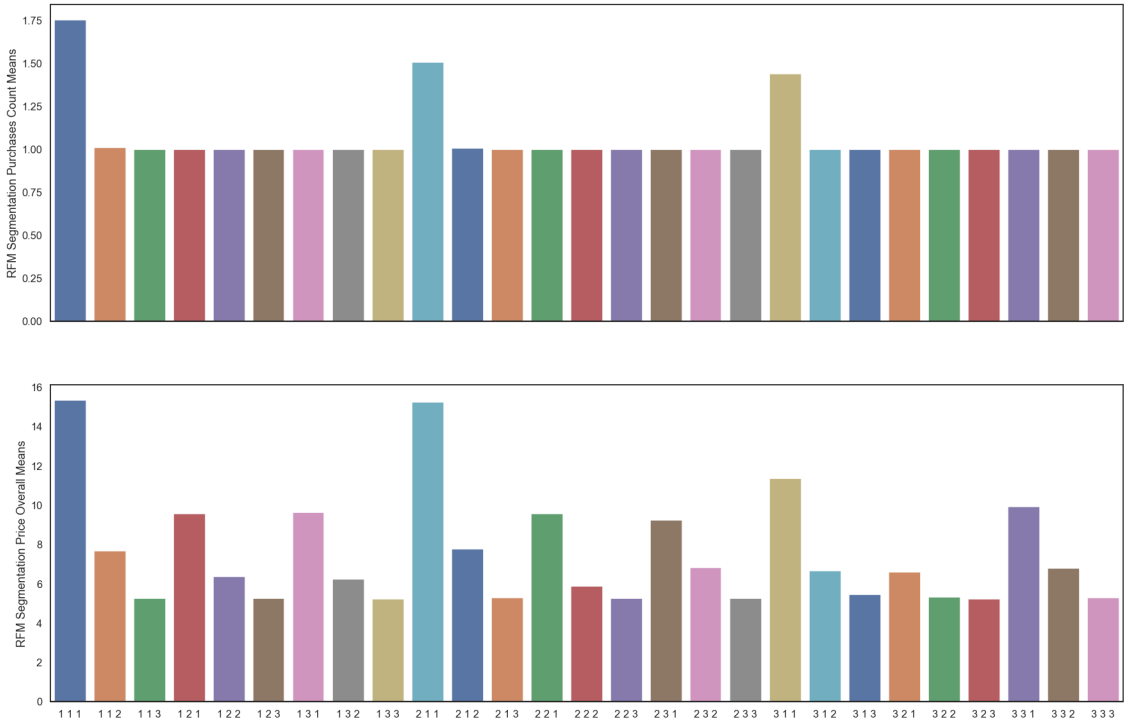
۳.۱.۲ بخش‌بندی به حالت مرتب‌سازی

[25]:
```
segments_labels = []
f_means = []
m_means = []

# labels had been defined before
sorted_rfm_df['R Segment'] = pd.qcut(sorted_rfm_df[r_rank], 3, labels=labels)

for i in range(1, 4):
    sr_df = sorted_rfm_df[(sorted_rfm_df['R Segment'] == i)].copy()
    sr_df['F Segment'] = pd.qcut(sr_df[f_rank], 3, labels=labels)
    for j in range(1, 4):
        srf_df = sr_df[(sr_df['F Segment'] == j)].copy()
        srf_df['M Segment'] = pd.qcut(srf_df[m_rank], 3, labels=labels)
        for k in range(1, 4):
            srfm_df = srf_df[(srf_df['M Segment'] == k)].copy()

            label = f'{i} {j} {k}'
            segments_labels.append(label)

            f_mean = srfm_df['Purchases Count'].copy().mean()
            m_mean = srfm_df['Price Overall'].copy().mean()

            f_means.append(f_mean)
            m_means.append(m_mean)
```

[26]:
```
f, (ax1, ax2) = plt.subplots(2, 1, figsize=(30, 20), sharex=True)

sns.barplot(x=segments_labels, y=f_means, palette="deep", ax=ax1)
ax1.axhline(0, color="k", clip_on=False)
ax1.set_ylabel(f"RFM Segmentation Purchases Count Means")
```

```
sns.barplot(x=segments_labels, y=m_means, palette="deep", ax=ax2)
ax2.axhline(0, color="k", clip_on=False)
ax2.set_ylabel(f"RFM Segmentation Price Overall Means")
```

[26]: Text(0, 0.5, 'RFM Segmentation Price Overall Means')



در اینجا دو چیز مشاهده می‌شود:

- مشتریان بخش‌های ۱۱۱ و ۲۱۱ و ۳۱۱ میانگین تعداد خرید بالاتری را نسبت به دیگر بخش‌ها کسب کرده‌اند. این یعنی که برای دستیابی به مشتریان با تکرار خرید زیاد، معیارهای Frquency و Monetary Value مهم هستند و تازگی خرید آن‌ها چندان مهم نیست.
- مجددا در مورد میانگین مقدار خرج نیز، دسته‌های ۱۱۱ و ۲۱۱ و ۳۱۱ میانگین بیشتری دارند و همانند مورد قبل، در اینجا نیز معیار تازگی چندان نقشی ایفا نمی‌کند.

پس؛ برای دستیابی به تعداد خرید بیشتر و میزان پول پرداختی، باید ابتدا به بخش‌های ۱۱۱ و بعد ۲۱۱ و ۳۱۱ توجه شود.

۲.۲ بخش‌بندی با کلاسترینگ

[27]:
```
clustering_df = clean_df.copy()
```

ابتدا باید ویژگی‌هایی که وابستگی زیادی به هم دارند را حذف کنیم. به همین خاطر ویژگی‌های زیر حذف می‌شوند:

- Book Length Overall
- Price Overall
- Listened

چرا این موارد و چرا ویژگی‌های دیگر نه؟ به این خاطر که میانگین در مورد قیمت، اطلاعات بیشتری از رفتار خرید مشتری به ما می‌دهد، همچنین در مورد نیاز مشتری نیز با میانگین طول کتاب‌هایی که خریده به نتایج بهتری نسبت به مجموع طول کتاب‌هایی که خریده دست پیدا می‌کنیم. از سمت دیگر میزان دقایقی که فرد به کتاب‌ها گوش داده چندان مهم نیست و به طور مستقیم، اطلاعات زیادی به ما نمی‌دهد.

ویژگی ID نیز به خاطر بی‌اهمیت‌بودن در این فرایند، حذف می‌شود.

یک ویژگی جدید اما، به این مجموعه داده اضافه می‌شود و آن ویژگی تعداد خرید است. علت افزودن این ویژگی این است که بر اساس این پارامتر بتوان تفاوت میانگین‌های یکسان را در نظر گرفت. فرض کنیم دو مشتری داریم که یکی ۵ خرید ۲۰۰ تومانی داشته و یکی یک خرید ۲۰۰ تومانی دارد. میانگین خرید این دو برابر با ۲۰۰ می‌شود. برای ایجاد تمایز بین این دو فرد، نیاز به ویژگی تعداد خرید داریم که قبلا در بخش‌بندی بر اساس RFM آن را به دست آوردیم.

[28]:
```
labels = ['Book Length Overall', 'Price Overall', 'Listened']
clustering_df = clustering_df.drop(labels=labels, axis=1)
clustering_df['Purchases Count'] = rfm_df['Purchases Count']
clustering_df.head()
```

[28]:

	Book Length Average	Price Average	Listened Percent	Support Request	Engagement Time	Target	Net Promoter Score	Purchases Count
0	1620.0	19.73	0.99	5	92	0	3	1.0
1	2160.0	5.33	0.00	0	0	0	0	1.0
2	2160.0	5.33	0.00	0				
3	1620.0	5.96	0.42	1				
4	2160.0	5.33	0.22	0				

2	388	0	0	1.0
3	129	0	0	1.0
4	361	0	0	1.0

۱.۲.۲ بررسی خوشه‌بندی با دندروگرام

خوب است قبل از خوشه‌بندی، کمی داده‌ی خود را با دندروگرام بررسی کنیم.

برای بررسی متریک‌ها و متدهای مختلف، من تصمیم گرفتم صرفا دندروگرام را بر اساس داده‌های عددی رسم کنم و از سمت دیگر، نتیجه‌ی خوشه‌بندی را طبق داده‌های کتگوریکال بررسی کنم. بر همین اساس به مقدارهای هر ویژگی کتگوریکال مثل NPS و Target یک رنگ اختصاص داده‌شد که رنگ سفید برای داده‌های گم‌شده، رنگ قرمز برای افراد ناراضی و ضدترویج، آبی برای منفعلان و سبز برای راضی‌ها و مروجان است. دو رنگ خاکستری و زرد نیز مقدارهای تارگت را بررسی می‌کنند.

[29]:
```
numerical_and_useful_cols = [
    'Book Length Average',
    'Price Average',
    'Listened Percent',
    'Support Request',
    'Engagement Time',
    'Purchases Count'
]

nps_colors = clustering_df['Net Promoter Score'].map({
    0: 'white',
    1: 'red',
    2: 'blue',
    3: 'green'
})

target_colors = clustering_df['Target'].map({
    0: 'gray',
    1: 'yellow'
})
```

پس از بررسی انواع مختلف متریک و متد، مشاهده کردم که متریک canberra بهترین نوع خوشه‌بندی را برای تارگت‌ها دارد و به خوبی تاثیر هر ویژگی را در تارگت نشان می‌دهد.

همانطور که مشاهده می‌شود، در دو مقطع رنگ زرد تجمیع شده است:

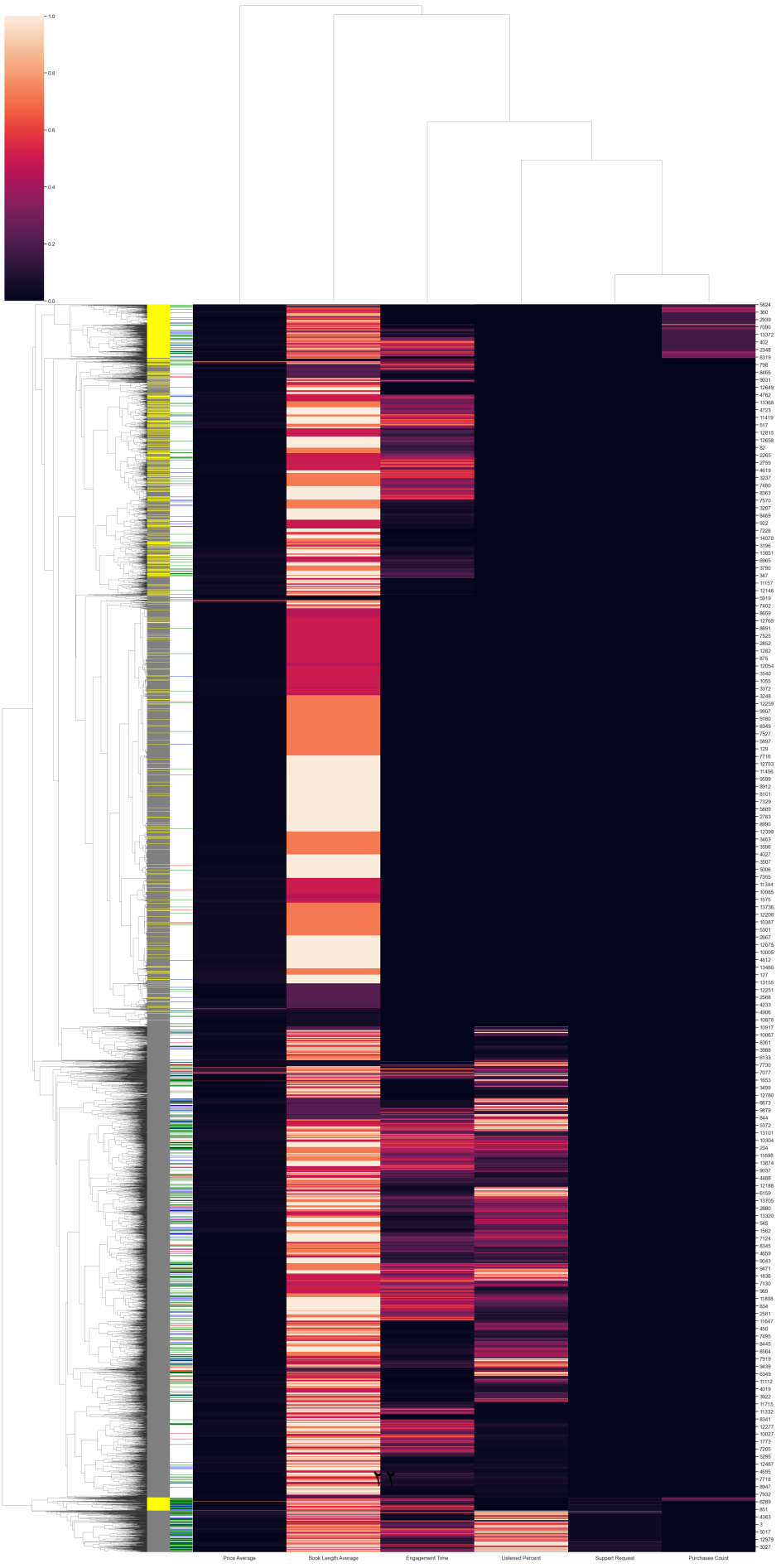
- مقطع بالایی که در اینجا افراد از نظر طول کتاب‌هایی که خریده‌اند وضعیت نسبتا متوسطی دارند و پراکندگی در این مورد به چشم می‌خورد، متوسط هزینه‌ای که کرده‌اند و درصد شنیده‌هایشان و تقاضاهای پشتیبانی نیز به شدت کم است. از سمت دیگر این مقطع تنها جایی است که در آن تعداد خریدها بیشتر است و کاربران با تعداد خرید بالا در اینجا حضور دارند. در مورد میزان انگیجمنت هم پراکندگی به چشم می‌خورد.
- در مقطع پایینی، همه‌چیز به جز مقدار تقاضاهای پشتیبانی مشابه مقطع بالایی است که در اینجا مقدار تقاضاهای پشتیبانی نسبت به مقطع بالایی بیشتر شده.

در مورد NPS نیز همبستگی زیادی بین تارگت‌های ۱ و کسانی که امتیاز داده‌اند وجود دارد.

خوشه‌بندی‌ای که این دندروگرام ارائه می‌دهد را می‌توان در سه خوشه خلاصه کرد:

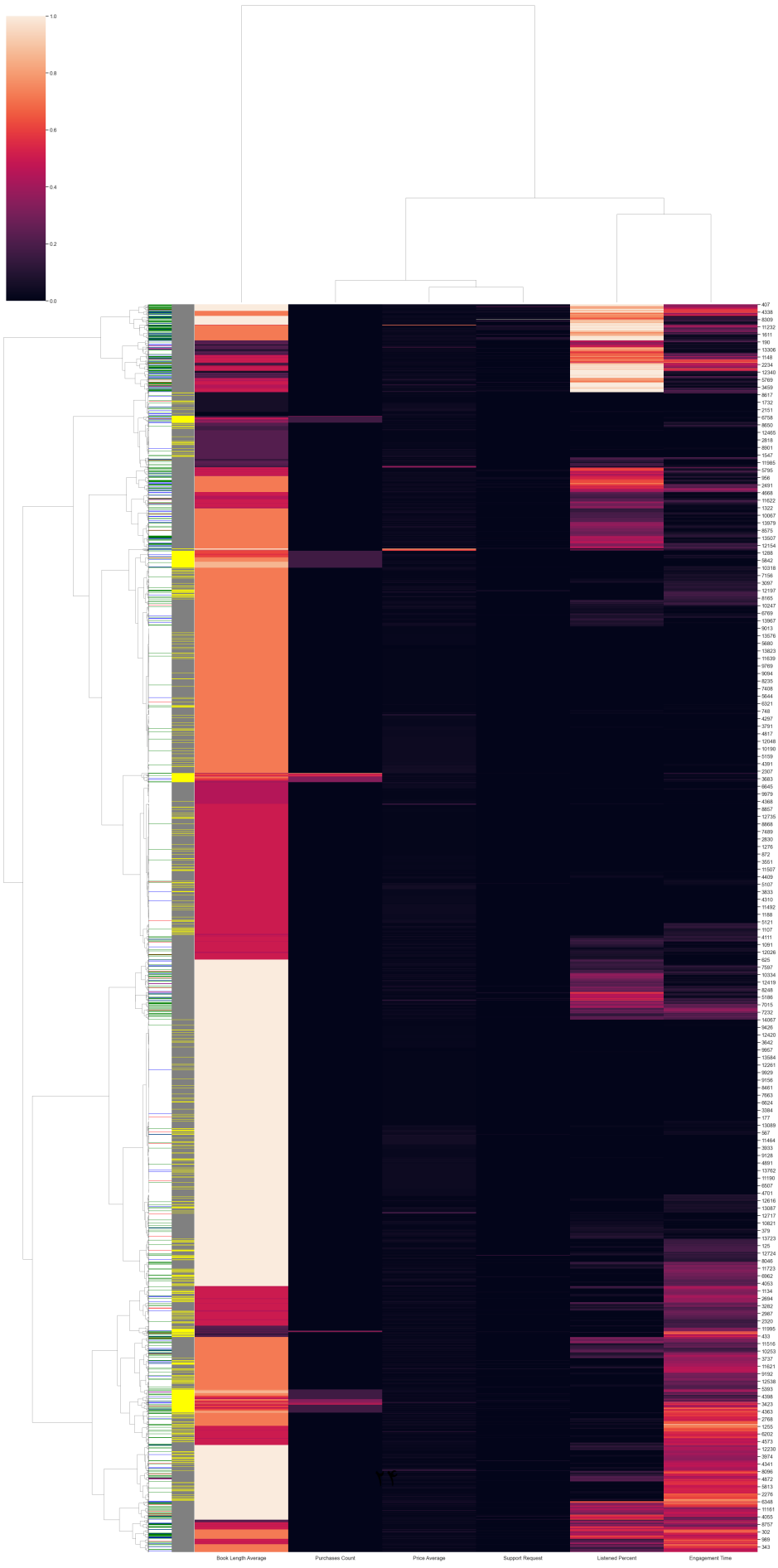
۱. از ابتدای شکل تا اواسط آن که تارگت‌های زرد ناپدید می‌شوند: در این خوشه بیشتر کسانی قرار می‌گیرند که درصد شنیده‌ی کمی دارند.
۲. از خوشه‌ی قبلی تا جایی که تارگت‌های زرد پدیدار می‌شوند: در این خوشه کسانی قرار می‌گیرند که درصد شنیده‌ی زیادی دارند.
۳. باقی‌مانده‌ی داده‌ها: در این خوشه کسانی که تقاضای پشتیبانی داشته‌اند قرار می‌گیرند.

[30]:
```
g = sns.clustermap(
    clustering_df[numerical_and_useful_cols],
    metric='canberra',
    standard_scale=1,
    row_colors=[target_colors, nps_colors],
    figsize=(35,70))
g.savefig('dendrogram_canberra.png', dpi=150)
```


1.0
0.8
0.6
0.4
0.2
0.0
5924
360
2939
7090
13372
402
2348
8319
798
9465
9031
12649
4762
13368
4723
11419
517
12815
12658
82
2265
2759
4619
3237
7480
8363
7570
3207
9469
922
7226
14070
3196
13851
8965
3790
347
11157
12146
5919
7402
8659
12765
8891
7525
2652
1282
875
12054
3540
1055
3372
3248
12259
9907
9180
8349
7527
9897
129
7716
12793
11456
9599
8912
8101
7529
5889
2783
8990
12399
3463
3596
4027
3507
9006
7355
11344
10985
1575
13736
12208
10387
5301
2667
12075
10005
4612
13460
127
13155
12251
2568
4233
4906
10878
10917
10067
8361
3988
6133
7730
7077
9553
3499
12780
9673
9879
944
5372
13101
10304
234
11698
13874
9037
4488
12188
6159
13705
2680
13320
545
1562
7124
8345
4659
9043
9471
1636
7130
969
11808
834
2561
11647
450
7495
8445
8564
7919
9439
6349
11112
4019
3922
11715
11332
8341
12277
10027
1773
7205
5295
12487
4695
7718
8947
7932
6289
851
4383
5
5017
12979
3027
Price Average
Book Length Average
Engagement Time
Listened Percent
Support Request
Purchases Count

معیار اقلیدسی خوشه‌بندی جذاب‌تری را در صورتی که از متد ward استفاده کنیم به ما می‌دهد ولی این خوشه‌بندی اطلاعات جامعی در مورد تارگت به ما ارائه نمی‌دهد.

[31]:
```
g = sns.clustermap(
    clustering_df[numerical_and_useful_cols],
    method='ward',
    metric='euclidean',
    standard_scale=1,
    row_colors=[nps_colors, target_colors],
    figsize=(35,70))
g.savefig('dendrogram_euclidean.png', dpi=150)
```


1.0
0.8
0.6
0.4
0.2
0.0
Book Length Average
Purchases Count
Price Average
Support Request
Listened Percent
Engagement Time

بر اساس این دندروگرام به اطلاعات جدیدی می‌رسیم. اگر خوشه‌بندی را بر اساس ۴ خوشه در نظر بگیریم (چون اگر دندروگرام را به سمت راست بچرخانیم، در ناحیه‌ای که فاصله‌ی عمودی بین گره‌ها زیاد شده، ۴ خط افقی وجود دارد)؛ به این ۴ خوشه می‌رسیم:

۱. چند داده‌ی بالایی نمودار که NPS غیر صفر دارند. در این خوشه کسانی قرار می‌گیرند که هم امتیاز داده‌اند، هم زمان خیلی زیادی به کتاب‌ها گوش داده‌اند و خرید مجدد نداشته‌اند. این دسته تارگت ۰ را ثبت کرده‌اند.
۲. ادامه‌ی خوشه‌ی اول تا اواسط داده‌ها که تغییر در رنگ معیار «میانگین طول کتاب‌ها» از قرمز به سفید وجود دارد: این خوشه شامل کسانی است که از هر نظر به جز میانگین طول کتاب‌هایی که خریده‌اند، مقدار کمی دارند.
۳. از ادامه‌ی خوشه‌ی دوم تا جایی که معیار «میانگین طول کتاب‌ها» مجددا تغییر رنگ داده: مشخص‌ترین خصیصه‌ی این خوشه کاربرانی هستند که بیشترین میزان میانگین کتاب‌های خریداری‌شده را داشته‌اند.
۴. خوشه‌ی چهارم که از خوشه‌ی سوم تا آخر نمودار ادامه دارد در مورد کسانی است که انگیجمنت زیادی را با محصول داشته‌اند.

پس بنابر این خوشه‌بندی‌ها، می‌توانیم این چهار دسته کاربر را داشته باشیم:

۱. کسانی که به محصول امتیاز داده‌اند.
۲. کسانی که انگیجمنت کمی دارند.
۳. کسانی که میانگین طول کتاب‌هایشان زیاد است.
۴. کسانی که انگیجمنت زیادی با محصول دارند.

اکنون دیدی تحلیل در مورد خوشه‌ها داریم. برای خوشه‌بندی بهتر، سراغ الگوریتم K-Means می‌رویم.

۲.۲.۲ نرمال‌سازی

برای خوشه‌بندی و تشخیص داده‌های پرت بهتر است داده‌ها نرمال شوند. برای این کار از نرمال‌سازی مین‌ماکس استفاده می‌شود.

[32]:
```
clustering_df.dtypes
```

[32]:
```
Book Length Average    float64
Price Average          float64
Listened Percent       float64
Support Request          int64
Engagement Time          int64
Target                   int64
Net Promoter Score       int64
Purchases Count        float64
dtype: object
```

برای نرمال‌سازی نمی‌توان از داده‌های کتگوریکال استفاده کرد، که مشاهده می‌شود چنین داده‌ای وجود ندارد.

[33]:
```
values = clustering_df.values
columns = clustering_df.columns.values
min_max_scaler = preprocessing.MinMaxScaler()
normalized_array = min_max_scaler.fit_transform(values)
normalized_df = pd.DataFrame(normalized_array, columns=columns)
normalized_df.describe()
```

[33]:

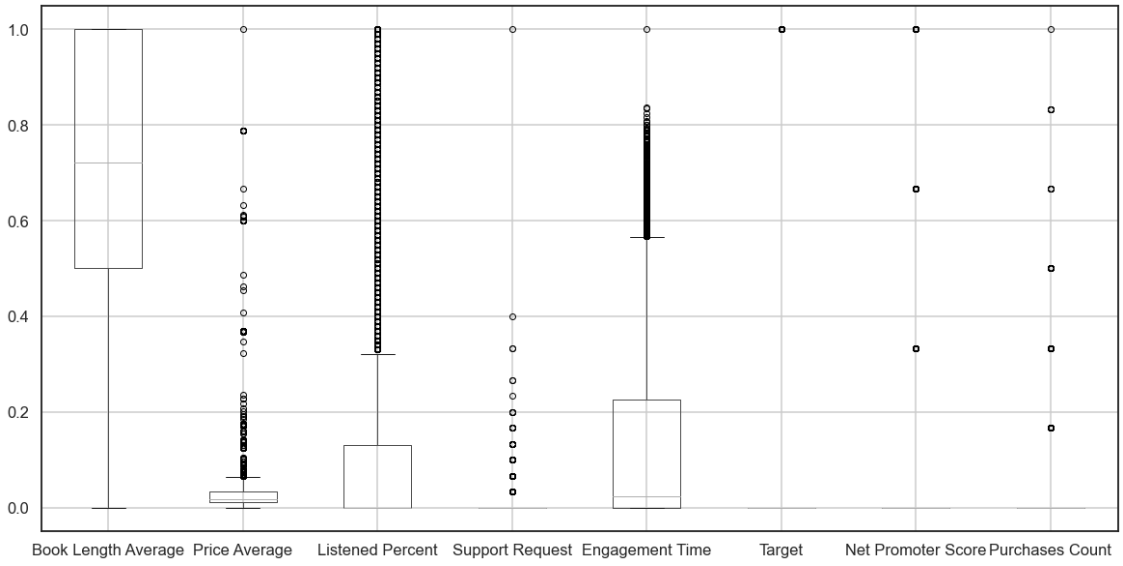
	Book Length Average	Price Average	Listened Percent	Support Request	\
count	14084.000000	14084.000000	14084.000000	14084.000000	
mean	0.707449	0.025526	0.125659	0.002341	
std	0.259434	0.038808	0.241206	0.015739	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.500000	0.011568	0.000000	0.000000	
50%	0.722222	0.016446	0.000000	0.000000	
75%	1.000000	0.032578	0.130000	0.000000	
max	1.000000	1.000000	1.000000	1.000000	

	Engagement Time	Target	Net Promoter Score	Purchases Count
count	14084.000000	14084.000000	14084.000000	14084.000000
mean	0.133481	0.158833	0.152513	0.010615
std	0.190103	0.365533	0.342332	0.055147
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.023707	0.000000	0.000000	0.000000
75%	0.226293	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000

۳.۲.۲ بررسی داده‌های پرت

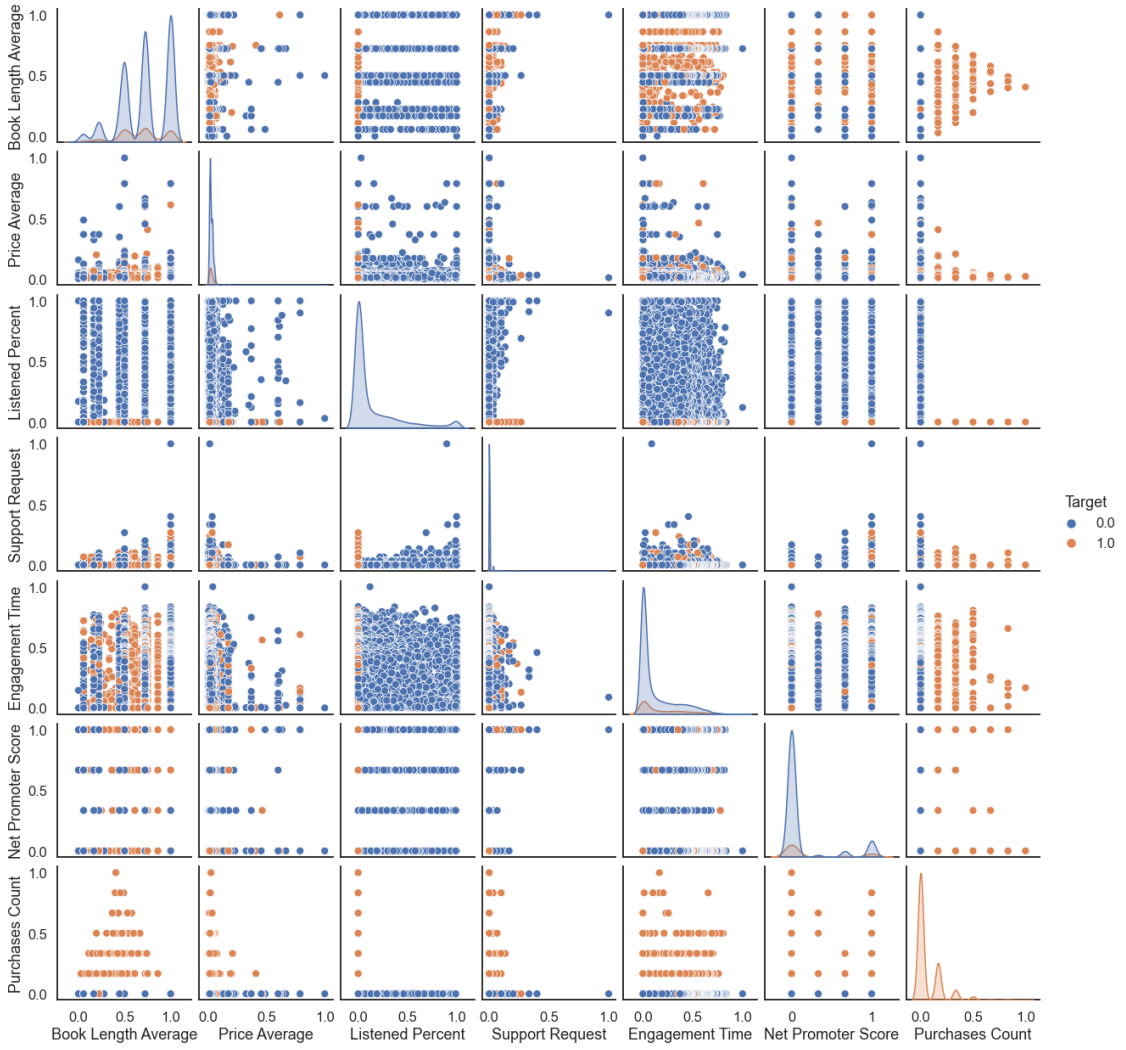
چون در مورد کلاسترینگ، من از متد K-Means استفاده می‌کنم، بررسی داده‌های پرت و حذف یا اصلاح آن‌ها اهمیت زیادی دارد.

[34]:
```
normalized_df.boxplot(figsize=(20,10))
plt.show()
```



از طریق این باکس‌پلات متوجه می‌شویم که در مورد «میانگین طول کتاب‌ها»، داده‌ی پرتی وجود ندارد ولی سایر ویژگی‌ها به شدت نویزی هستند. با اینحال برای نتیجه‌گیری زود است و نیاز به بررسی بیشتری وجود دارد.

[35]:
```
plot = sns.pairplot(normalized_df, hue="Target")
plot.savefig('overview.png', dpi=150)
```



بر اساس این پیرپلات، دیدگاه کلی جالبی در مورد داده‌ها می‌توان به دست آورد.

یکی در مورد ویژگی «میانگین زمان کتاب‌ها» است که هر چقدر بیشتر می‌شود، آمار افراد با «تارگت» ۰ هم بیشتر می‌شود، در حالی که افراد با «تارگت» ۱ تا قسمتی زیاد می‌شوند ولی از مقطعی به بعد ثابت می‌مانند. همچنین غیر از این مورد، سایر ویژگی‌ها نموداری به شدت غیر متوازن دارند.

دیگر اینکه «میانگین خرید مشتریان» تا حد زیادی در مقدارهای کم خلاصه شده‌است. این مورد را از طریق باکس‌پلات هم می‌توان متوجه شد و به همین خاطر داده‌های نویزی زیادی وجود دارند.

در مورد «میزان درخواست»، نمودار به شدت نامتوازن است و دچار چردگی شده‌است. در مورد این ویژگی و ویژگی «میانگین قیمت پرداختی»، نیاز به اصلاح داریم چون نویز شدید است.

در مورد «تعداد خرید»، کسانی که تعداد خرید بیشتری را ثبت کرده‌اند، «تارگت» داشته‌اند؛ یعنی، می‌توان بر اساس تعداد خرید پیش‌بینی کرد که چه کسانی در آینده به ما مجددا مراجعه می‌کنند. کسانی که فقط یکبار خرید کرده‌اند، مشتریان آینده‌ی ما نیستند.

یک نکته‌ی جالب دیگر در مورد تعداد خرید، وقتی است که با میانگین طول کتاب‌ها تریب می‌شود. هرچقدر میانگین طول کتاب‌ها از ۰ به ۵۰ درصد نزدیک می‌شود، تعداد خریدهای مجدد بیشتر می‌شود ولی اگر از ۵۰ درصد دور شویم، تعداد خریدهای مجدد کم می‌شود.

[36]:
```
df['Book Length Average'].quantile(0.5)
```

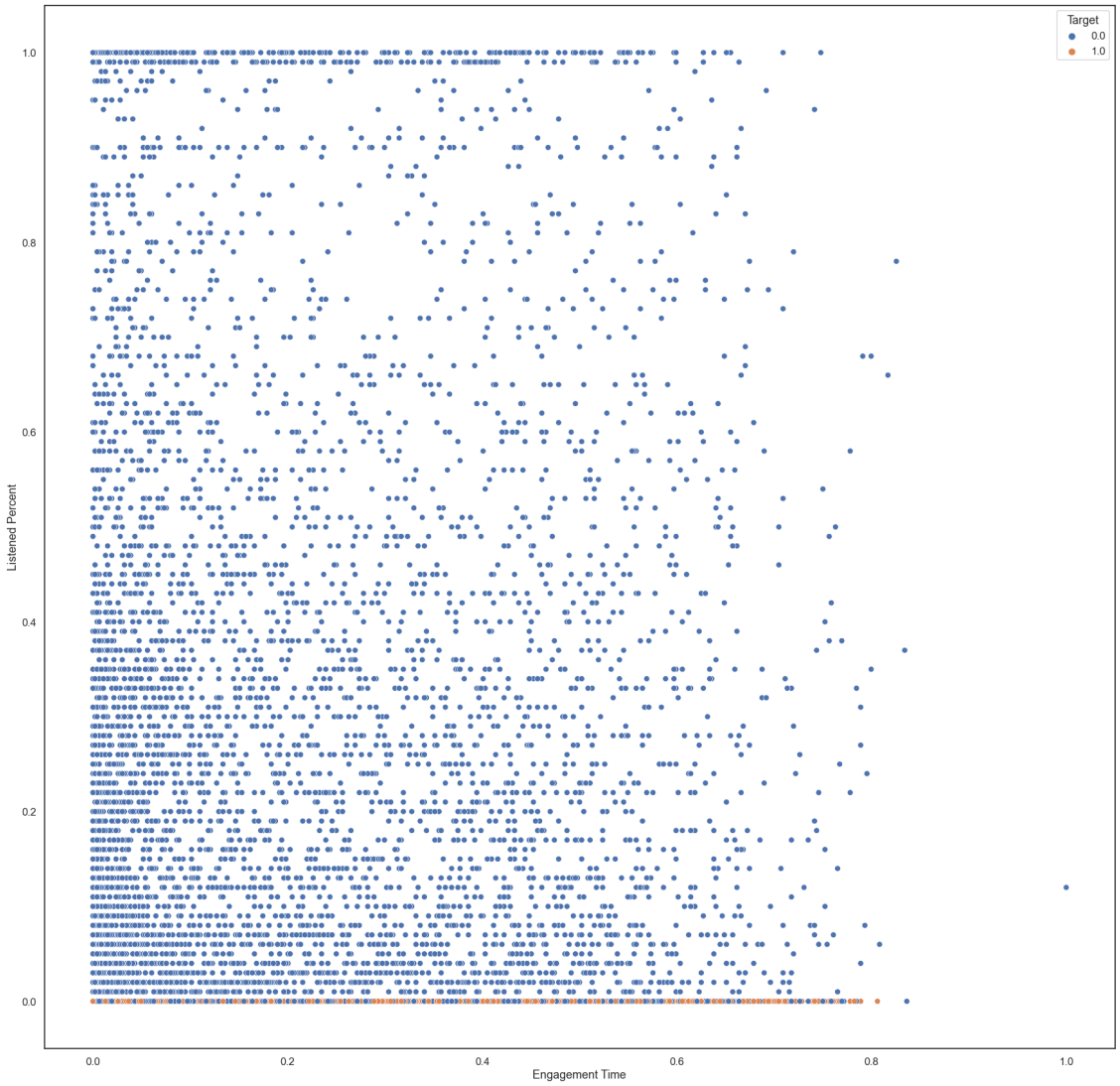
[36]:
```
1620.0
```

۱۶۲۰ میانگینی است که در ۵۰ درصدم دیتافریم مخصوص به Book Length Average وجود دارد. پس باید سعی کنیم میانگین کتاب‌هایی که کاربران می‌خرند را به ۱۶۲۰ دقیقه نزدیک کنیم.

بعضی موارد نیاز به جزئی‌شدن دارند که با اسکترپلات بررسی می‌شوند.

[37]:
```
_, ax = plt.subplots(1, 1, figsize=(30, 30), sharex=True)
sns.scatterplot(data=normalized_df, x="Engagement Time", y="Listened Percent",␣
↪hue="Target", ax=ax)
```

[37]:
```
<AxesSubplot:xlabel='Engagement Time', ylabel='Listened Percent'>
```



این اسکترپلات اطلاعات عجیبی را نشان می‌دهد. کسانی که مقداری از طول کتاب‌ها را گوش داده‌اند، ممکن نیست که «تارگت» ۱ داشته باشند!

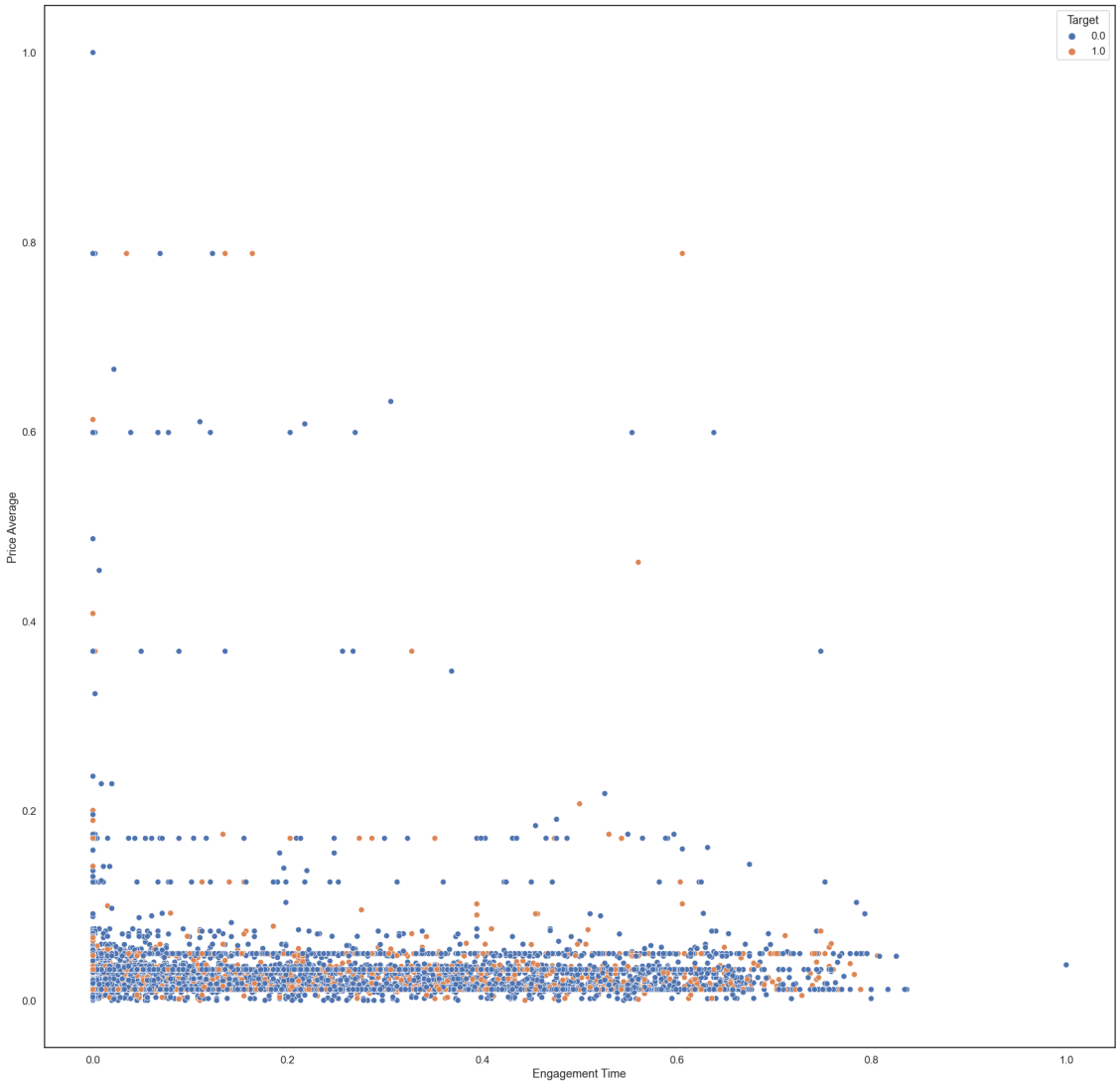
شاید این به خاطر کیفیت بد کتاب‌ها باشد؛ به هر حال خیلی عجیب است.

این مورد همانطور که با نگاه به پیرپلات در مورد ویژگی «درصد شنیده‌شده» قابل مشاهده است، نسبت به ویژگی‌های دیگر نیز وجود دارد. و اگر به خود نمودار بررسی ویژگی «درصد شنیده‌شده»، صرفا طبق تارگت‌ها در پیرپلات نگاه کنیم، کلا رنگ نارنجی به چشم نمی‌خورد. علت این است که کسانی که درصدی از کتاب‌ها را گوش داده‌اند، دیگر هرگز برای خرید باز نگشته‌اند!

در مورد ویژگی «درصد شنیده‌شده»، با اینکه قسمت بالای نمودار نسبت به پایین آن تنک‌تر است ولی به طور کلی، داده‌های بالا به اندازه‌ای زیاد هستند که بتوان از نویزی بودن آن‌ها صرف نظر کرد. پس ویژگی درصد شنیده‌شده هم به خاطر وضعیت خاص آن در شناسایی تارگت و هم به خاطر پراکندگی نسبتا متوازن، نویزی در نظر گرفته نمی‌شود.

[38]:
```
_, ax = plt.subplots(1, 1, figsize=(30, 30), sharex=True)
sns.scatterplot(data=normalized_df, x="Engagement Time", y="Price Average",␣
↪hue="Target", ax=ax)
```

[38]: <AxesSubplot:xlabel='Engagement Time', ylabel='Price Average'>



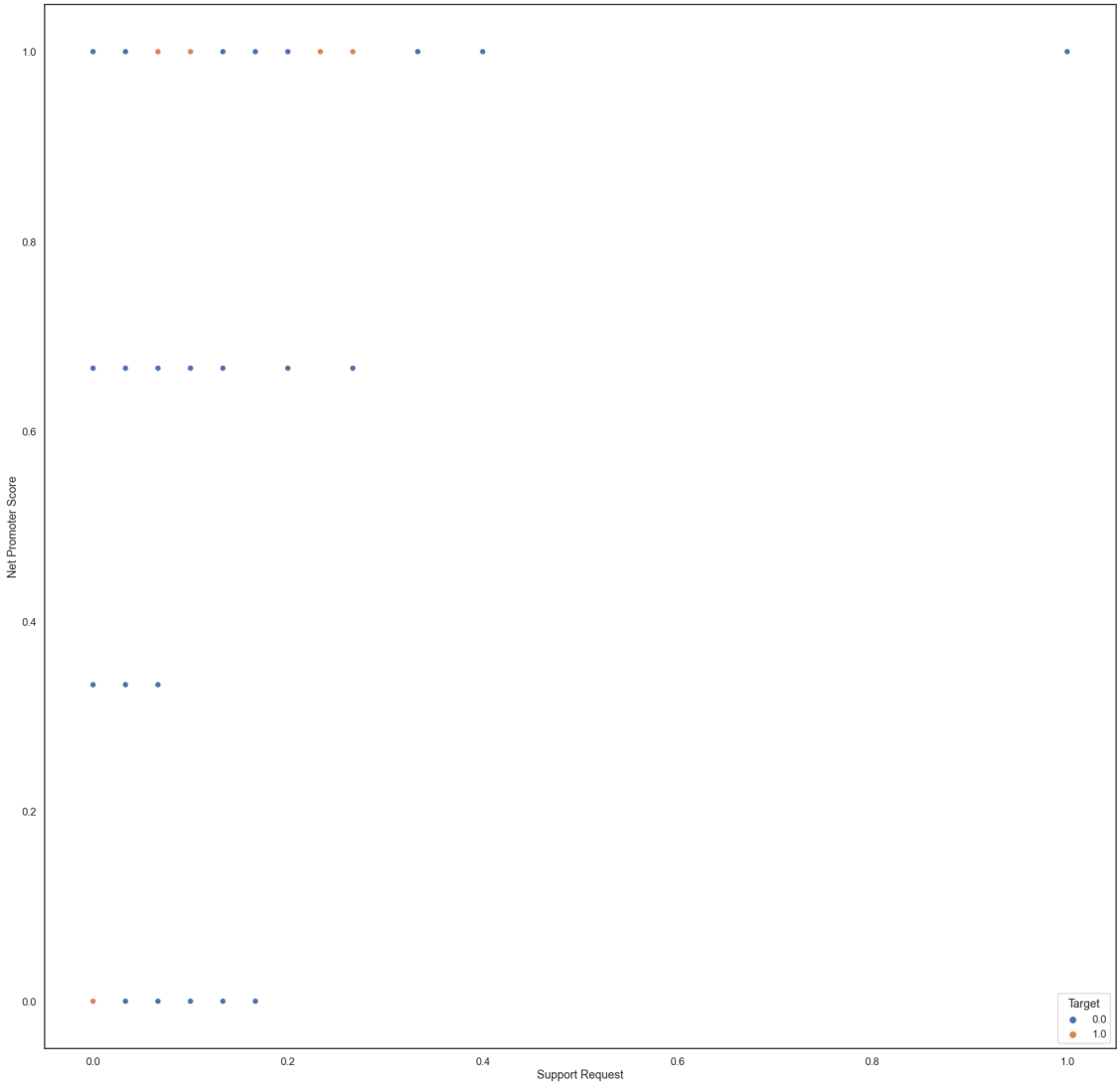
بر اساس این اسکاتر پلات می‌توان تمرکز داده‌ها در مورد ویژگی «میانگین قیمت» را در محدوده‌ی ۰ تا ۰/۲ مشاهده کرد و از سمت دیگر، در مورد Engagement Time نیز از ۰/۸ به بالا، داده‌ها تنک شده‌اند.

بر همین اساس به شکل عینی، نتیجه گرفته می‌شود که موارد زیر داده‌های پرت هستند:

- Engagement Time > 0.8
- Price Average > 0.2

[39]:
```
_, ax = plt.subplots(1, 1, figsize=(30, 30), sharex=True)
sns.scatterplot(data=normalized_df, x="Support Request", y="Net Promoter␣
↪Score", hue="Target", ax=ax)
```

[39]: <AxesSubplot:xlabel='Support Request', ylabel='Net Promoter Score'>



در مورد این دو ویژگی نیز معیار NPS تا حد خوبی متوازن است ولی در مورد معیار Support Request مقادیر خاصی وجود

دارند که مقدار ۱ دارند. این مقادیر به عنوان داده‌ی پرت در نظر گرفته می‌شود؛ چراکه، همانطور که مشاهده می‌شود از باقی داده‌ها فاصله‌ی زیادی دارند.

ویژگی «تارگت» نیز چون ویژگی هدف است، به عنوان ویژگی نویزی در نظر گرفته نمی‌شود.

۴.۲.۲ سیاست متناسب با داده‌های پرت

تشخیص داده‌های پرت، راه‌های زیادی دارد و یکی از آن‌ها که مشاهده‌ی عینی است، بررسی شد. راه دیگر استفاده از معیار Z است که بر اساس انحراف معیار مشخص می‌کند چه داده‌هایی پرت هستند.

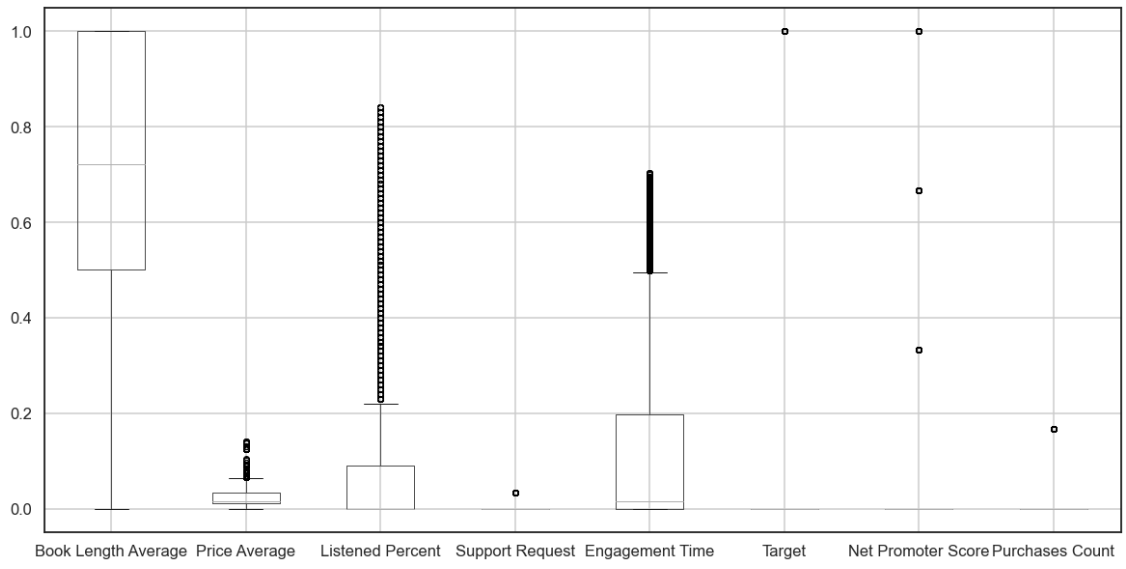
چیزی که من آن را مناسب برای این پروژه می‌بینم همین راه معیار Z است؛ زیرا، راه مشاهده‌ی عینی در جایی که باکس‌پلات آن تا این حد نویز را نشان می‌دهد، اصولی نیست و دارای خطای زیادی است. به علاوه طبق پیرپلات می‌توان تشخیص داد که داده‌ها چند بیشینه یا کمینه‌ی محلی ندارند و صرفا به یک سمت خاص کشیده شده‌اند و دچار چردگی شده‌اند. در این حالت، استفاده از انحراف معیار می‌تواند بسیار کمک‌کننده باشد.

به دلیل بزرگی دیتاست، می‌توان داده‌های پرت را مستقیما حذف کرد و نیازی به اصلاح نیست.

بیشینه‌ی معیار Z برابر با ۳ در نظر گرفته شده‌است که مقدار متداولی در یادگیری ماشین است.

[40]:
```
mining_df = normalized_df[(np.abs(stats.zscore(normalized_df)) < 3).all(axis=1)]
```

[41]:
```
mining_df.boxplot(figsize=(20,10))
plt.show()
```



مشاهده می‌شود که داده‌های پرت تا حد خوبی حذف شده‌اند.

۵.۲.۲ عملیات خوشه‌بندی

چون خوشه‌بندی یک روش نظارت‌نشده است و نیازی به لیبل ندارد، ویژگی Target حذف می‌شود.

[42]:
```
ready_df = mining_df.copy().drop(labels='Target', axis=1)
```

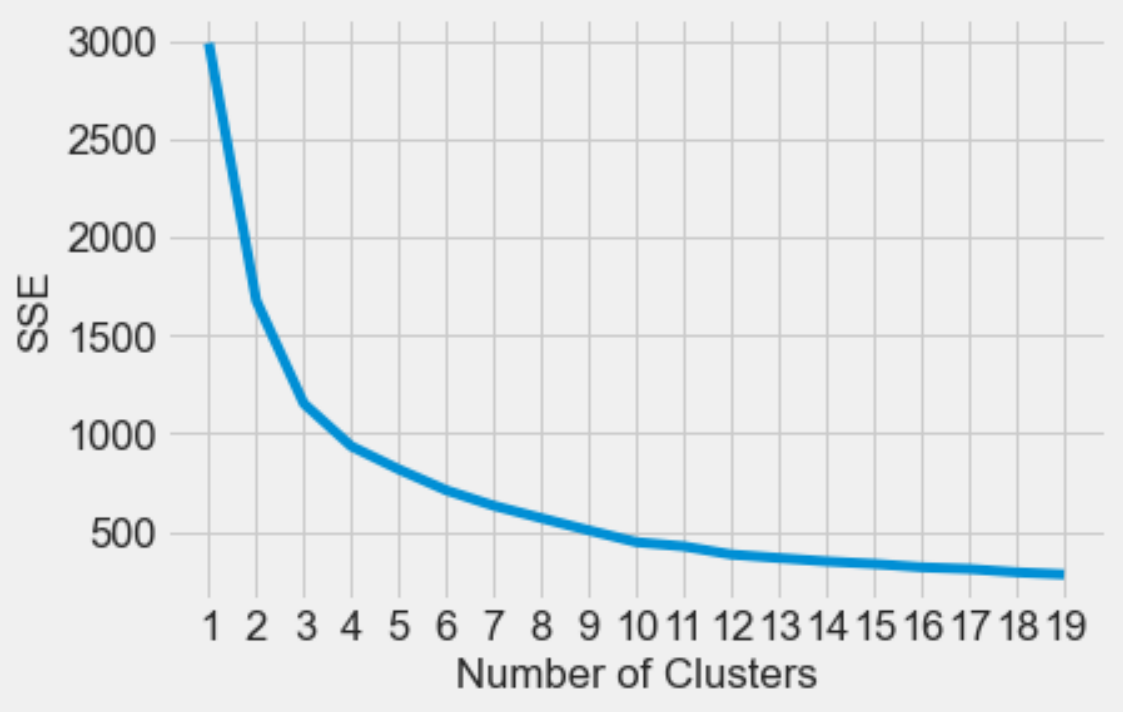
برای خوشه‌بندی از متد K-Means استفاده شده‌است. برای انتخاب گره‌های ابتدایی نیز از روش k-means++

[43]:
```
kmeans_kwargs = {
    "init": "k-means++",
    "n_init": 10,
    "max_iter": 300,
    "random_state": 42,
}
```

برای انتخاب مقدار k بر اساس روش نمودار elbow و ضریب silhouette، بررسی می‌شود که کدام مقدار برای k مناسب است.

[44]:
```
sse = []
for k in range(1, 20):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(ready_df)
    sse.append(kmeans.inertia_)
```

[45]:
```
plt.style.use("fivethirtyeight")
plt.plot(range(1, 20), sse)
plt.xticks(range(1, 20))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.show()
```



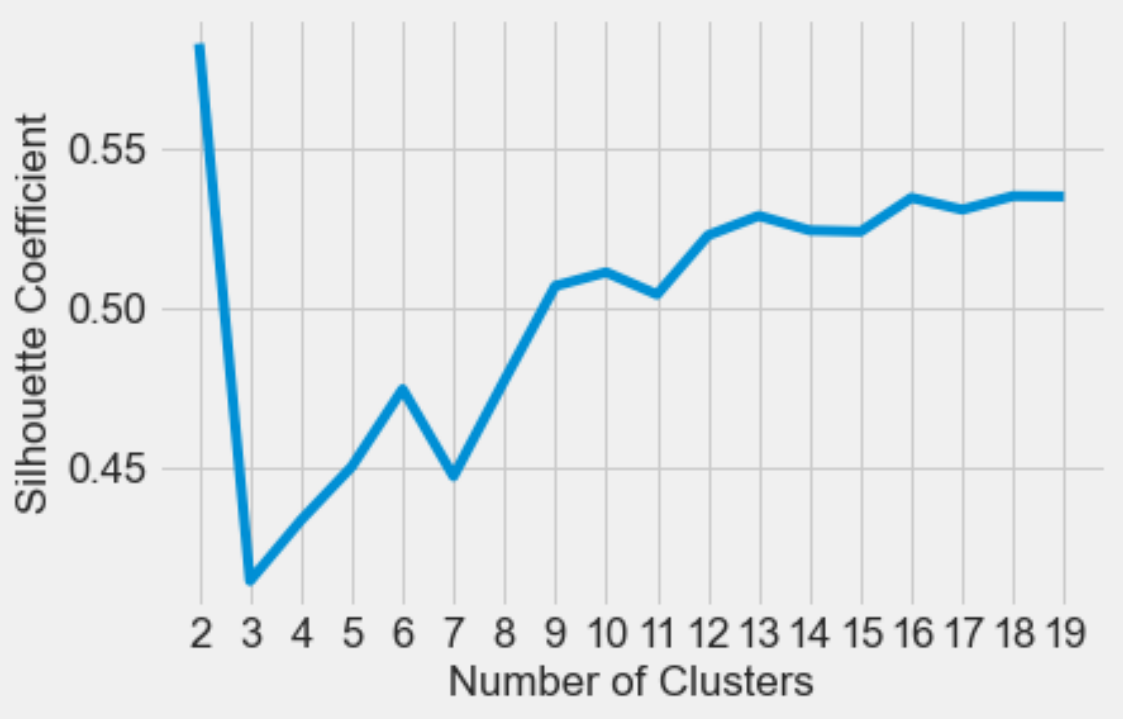
[46]:
```
kl = KneeLocator(range(1, 20), sse, curve="convex", direction="decreasing")
kl.elbow
```

[46]: 4

مقدار مناسب از نظر elbow برابر با ۴ است.

[47]:
```
silhouette_coefficients = []

# Notice you start at 2 clusters for silhouette coefficient
for k in range(2, 20):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(ready_df)
    score = silhouette_score(ready_df, kmeans.labels_)
    silhouette_coefficients.append(score)
```

[48]:
```
plt.style.use("fivethirtyeight")
plt.plot(range(2, 20), silhouette_coefficients)
plt.xticks(range(2, 20))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
plt.show()
```



اما بر اساس معیار ضریب silhouette مقدار مناسب برای k برابر با ۱۸ است، چراکه بیشینه‌ی این نمودار برای مقدار ۱۸ است.

البته بیشینه‌ی اصلی برای ۲ کلاستر است که مقدار مطلوبی نیست؛ به همین خاطر، بیشینه‌ی کلاستر سوم به بعد در نظر گرفته می‌شود.

به این ترتیب، k باید چه مقدار باشد؟

پاسخ من به این سوال این است که با توجه به عامل زیر، خروجی نمودار elbow پذیرفته می‌شود:

۱۸ در بخش‌بندی کاربران یک پلتفرم کتاب صوتی، تعداد زیاد بخش‌بندی برای کاربران چندان مطلوب نیست. بخش، مقدار خیلی زیادی است ولی ۴ بخش، نه.

بر این اساس، خروجی نمودار elbow با k=4 پذیرفته می‌شود.

[49]:
```
kmeans = KMeans(n_clusters=4, **kmeans_kwargs)
clusters = kmeans.fit_predict(ready_df)
kmeans.cluster_centers_.shape
```

[49]:
```
(4, 7)
```

و در نهایت مقدار تارگت و خوشه‌ای که به هر کاربر تعلق گرفته‌است را به دیتافریم متصل می‌کنم.

[50]:
```
data_with_clusters = ready_df.copy()
data_with_clusters['Cluster'] = clusters
data_with_clusters['Target'] = mining_df['Target'].copy()
data_with_clusters.head()
```

[50]:

	Book Length Average	Price Average	Listened Percent	Support Request	\
1	1.000000	0.011568	0.00	0.000000	
3	0.722222	0.016525	0.42	0.033333	
5	1.000000	0.005902	0.00	0.000000	
7	0.222222	0.011568	0.00	0.000000	
8	1.000000	0.011568	0.26	0.000000	

	Engagement Time	Net Promoter Score	Purchases Count	Cluster	Target
1	0.000000	0.0	0.0	0	0.0
3	0.278017	0.0	0.0	3	0.0
5	0.000000	0.0	0.0	0	0.0
7	0.000000	0.0	0.0	2	1.0
8	0.071121	0.0	0.0	0	0.0

تحلیل و بررسی نتایج در این قسمت، من ابتدا سعی کردم با بررسی خوشه‌هایی که توسط الگوریتم K-Means ایجاد شده‌است شروع کنم و محتوای هر خوشه را مقایسه کنم تا الگویی بین آن‌ها پیدا کنم.

[51]:
```
data_with_clusters[data_with_clusters['Cluster']==0].describe()
```

[51]:

	Book Length Average	Price Average	Listened Percent	Support Request	\
count	5766.000000	5766.000000	5766.000000	5766.000000	

mean	0.872567	0.021931	0.041191	0.000185
std	0.137466	0.014199	0.102212	0.002477
min	0.722222	0.000000	0.000000	0.000000
25%	0.722222	0.011568	0.000000	0.000000
50%	1.000000	0.015030	0.000000	0.000000
75%	1.000000	0.032578	0.020000	0.000000
max	1.000000	0.141328	0.820000	0.033333

	Engagement Time	Net Promoter Score	Purchases Count	Cluster	\
count	5766.000000	5766.000000	5766.000000	5766.0	
mean	0.031083	0.003815	0.003266	0.0	
std	0.057234	0.035461	0.023104	0.0	
min	0.000000	0.000000	0.000000	0.0	
25%	0.000000	0.000000	0.000000	0.0	
50%	0.000000	0.000000	0.000000	0.0	
75%	0.036638	0.000000	0.000000	0.0	
max	0.280172	0.333333	0.166667	0.0	

	Target
count	5766.000000
mean	0.108221
std	0.310686
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

در خوشه‌ی اول ۵۷۶۶ عضو وجود دارد. از این خوشه موارد زیر مشاهده می‌شود:

- بیشتر اعضای خوشه تارگت ۰ دارند.
- از نظر تعداد خرید، بیشتر کسانی که فقط یک خرید داشته‌اند در این خوشه بیشتر به چشم می‌خورند.
- بیشتر اعضای خوشه کسانی هستند که وضعیت NPS آن‌ها مشخص نیست و در دیتاست داده‌ی آن‌ها گم شده‌بود.
- بیشتر اعضای خوشه کسانی هستند که انگیجمنت کمی دارند.
- بیشتر اعضای خوشه تقاضای پشتیبانی کمی داشته‌اند. اعضای این خوشه حتی از خوشه‌ی اول هم کمتر پشتیبانی خواسته‌اند.
- در مورد درصد شنیده‌شده، انحراف معیار زیادی داریم و بیشتر اعضای خوشه کم شنیده‌اند ولی داده‌های بسیار پرتی هم در اینجا وجود دارند.

- در مورد میانگین قیمتی هم انحراف معیار زیادی داریم ولی اکثر افراد پول کمی پرداخت کرده‌اند.
- در مورد میانگین طول کتاب‌ها، باز هم انحراف معیار زیاد است ولی برعکس سایر موارد، اعضای این خوشه کتاب‌های طولانی‌ای را خریداری کرده‌اند.

[52]:
```
data_with_clusters[data_with_clusters['Cluster']==1].describe()
```

[52]:

	Book Length Average	Price Average	Listened Percent	Support Request
count	1738.000000	1738.000000	1738.000000	1738.000000
mean	0.706623	0.026179	0.251203	0.003069
std	0.254631	0.017124	0.241388	0.009640
min	0.000000	0.000000	0.000000	0.000000
25%	0.500000	0.011568	0.000000	0.000000
50%	0.722222	0.022427	0.210000	0.000000
75%	1.000000	0.032578	0.430000	0.000000
max	1.000000	0.139518	0.840000	0.033333

	Engagement Time	Net Promoter Score	Purchases Count	Cluster
count	1738.000000	1738.000000	1738.000000	1738.0
mean	0.214913	0.913694	0.010549	1.0
std	0.198784	0.146493	0.040593	0.0
min	0.000000	0.333333	0.000000	1.0
25%	0.034483	0.666667	0.000000	1.0
50%	0.154095	1.000000	0.000000	1.0
75%	0.375000	1.000000	0.000000	1.0
max	0.702586	1.000000	0.166667	1.0

	Target
count	1738.000000
mean	0.257768
std	0.437531
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

در خوشه‌ی دوم ۱۷۳۸ عضو وجود دارد. از این خوشه موارد زیر مشاهده می‌شود:

- در این خوشه هم تارگت ۱ داریم و هم تارگت ۰ ولی بیشتر اعضای این خوشه تارگت ۰ دارند.
- از نظر تعداد خرید، اعضای این خوشه مشابه خوشه‌ی اول هستند ولی تعداد خرید بیشتری را ثبت کرده‌اند.
- کسانی که به برنامه رای و امتیاز داده‌اند و بیشتر مروجان برنامه بوده‌اند، اینجا وجود دارند.
- در مورد میزان انگیجمنت، افراد با انگیجمنت کم در اینجا مشاهده می‌شوند. در حقیقت طبق باکس‌پلات (بلاک کد ۳۶) می‌توان در نظر گرفت که کسانی که داخل باکس انگیجمنت وجود دارند (از حد پایینی تا حد بالایی آن) در این خوشه قرار گرفته‌اند.
- کسانی که تقاضای پشتیبانی خیلی کمی دارند، در این قسمت قرار گرفته‌اند. البته این افراد داده‌ی پرت حساب نمی‌شوند.
- در مورد درصد شنیده‌شده، همه نوع کاربری وجود دارد ولی بیشتر کسانی که کم گوش کرده‌اند، مشاهده می‌شود.
- در مورد میانگین قیمت، همه‌ی افراد در نظر گرفته شده‌اند.
- در مورد طول میانگین کتاب‌های خریده‌شده، کسانی که زمان بیشتری گوش داده‌اند در اینجا قرار گرفته‌اند.

[53]:
```
data_with_clusters[data_with_clusters['Cluster']==2].describe()
```

[53]:

	Book Length Average	Price Average	Listened Percent	Support Request	\
count	3520.000000	3520.000000	3520.000000	3520.000000	
mean	0.408728	0.021356	0.058832	0.000284	
std	0.149759	0.013749	0.146716	0.003065	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.222222	0.011568	0.000000	0.000000	
50%	0.500000	0.015817	0.000000	0.000000	
75%	0.500000	0.032499	0.020000	0.000000	
max	0.611111	0.141564	0.840000	0.033333	

	Engagement Time	Net Promoter Score	Purchases Count	Cluster	\
count	3520.000000	3520.000000	3520.000000	3520.0	
mean	0.054806	0.007197	0.006629	2.0	
std	0.104929	0.056292	0.032575	0.0	
min	0.000000	0.000000	0.000000	2.0	
25%	0.000000	0.000000	0.000000	2.0	
50%	0.000000	0.000000	0.000000	2.0	
75%	0.056034	0.000000	0.000000	2.0	
max	0.674569	0.666667	0.166667	2.0	

	Target

count	3520.000000
mean	0.156534
std	0.363413
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

در خوشه‌ی سوم ۳۵۲۰ عضو وجود دارد که موارد زیر از این خوشه مشاهده می‌گردد:

- در این خوشه هم نظر تارگت، بیشتر افراد ۰ هستند.
- از نظر تعداد خرید، مشابه خوشه‌های قبلی است.
- از نظر NPS همه نوع فردی اینجا وجود دارد.
- در این خوشه افراد انگیجمنت کمی دارند.
- افراد این خوشه کمتر به پشتیبانی نیاز پیدا کرده‌اند.
- افراد این خوشه کم کتاب‌های خود را شنیده‌اند.
- از نظر قیمتی، همه‌نوع کاربری در این خوشه وجود دارد.
- از نظر طول کتاب‌های خریده‌شده، افراد این خوشه حد وسطی در کتاب‌های خریداری‌شده دارند.
- اعضای این خوشه کتاب‌های طولانی‌ای را خریداری کرده‌اند.

[54]:
```
data_with_clusters[data_with_clusters['Cluster']==3].describe()
```

[54]:

	Book Length Average	Price Average	Listened Percent	Support Request	\
count	2027.000000	2027.000000	2027.000000	2027.000000	
mean	0.801609	0.023088	0.148777	0.001052	
std	0.191920	0.014694	0.203890	0.005830	
min	0.166667	0.000000	0.000000	0.000000	
25%	0.722222	0.011568	0.000000	0.000000	
50%	0.722222	0.016761	0.040000	0.000000	
75%	1.000000	0.032578	0.240000	0.000000	
max	1.000000	0.136843	0.840000	0.033333	

	Engagement Time	Net Promoter Score	Purchases Count	Cluster	\
count	2027.000000	2027.000000	2027.000000	2027.0	
mean	0.420054	0.008716	0.006578	3.0	

std	0.132609	0.053204	0.032459	0.0
min	0.032328	0.000000	0.000000	3.0
25%	0.318966	0.000000	0.000000	3.0
50%	0.420259	0.000000	0.000000	3.0
75%	0.515086	0.000000	0.000000	3.0
max	0.702586	0.333333	0.166667	3.0

	Target
count	2027.000000
mean	0.165269
std	0.371515
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

در خوشه‌ی چهارم ۲۰۲۷ عضو وجود دارد. از این خوشه موارد زیر مشاهده می‌شود:

- اعضای خوشه معمولا تارگت ۰ دارند.
- تعداد خرید کمی دارند.
- از نظر NPS معمولا کسانی هستند که نظر آن‌ها مشخص نیست.
- از نظر انگیجمنت در اینجا کسانی که هیچ انگیجمنتی با فروشگاه ندارند، وجود ندارد. در حقیقت اکثر کسانی که با فروشگاه انگیجمنت زیادی دارند، در این خوشه قرار گرفته‌اند.
- در اینجا نیز افراد نیاز کمی به پشتیبانی داشته‌اند.
- این خوشه نیز مشابه خوشه‌های قبلی چندان از نظر قیمتی وضعیت خاصی ندارد.
- این خوشه مشابه خوشه‌ی اول شامل کسانی می‌شود که طول کتاب‌های خریداری شده‌شان زیاد است.

نتیجه‌گیری کلی بر اساس داده‌های آماری یک سری نتیجه‌گیری کلی قابل برداشت است:

- افرادی که کتاب‌های بلندی خریداری کرده‌اند یا به طور کلی میانگین طول کتاب‌هایی که خریداند زیاد باشد، یا در خوشه‌ی اول قرار می‌گیرند یا در خوشه‌ی چهارم.
- مروجان و کسانی که به برنامه امتیاز داده‌اند، در خوشه‌ي دوم قرار می‌گیرند.
- سایر افراد در خوشه‌ی سوم قرار می‌گیرند.
- خوشه‌ی اول و چهارم به شدت به هم شبیهند ولی خوشه‌ی اول از نظر درصد شنیده‌شده، کاربرانی که بیشتر کتاب‌ها را گوش داده‌اند در خود نگه داشته‌است. البته کاربران خوشه چهارم تعامل بسیار بیشتری نسبت به خوشه‌ی اول دارند.

۶.۲.۲ نتیجه‌گیری نهایی

بر اساس آنچه که در قسمت خوشه‌بندی گذشت، ما می‌توانیم مشتریان را به چهار بخش زیر تقسیم کنیم و نحوه‌ی تعامل با این چهار بخش را به این شکل ترسیم کنیم:

۱. کسانی که به برنامه امتیاز داده‌اند. این افراد معمولا جزو مروجان برنامه هستند و نه فقط امتیاز داده‌اند بلکه از برنامه راضی بوده‌اند. باید با این دسته به بهترین شکل برخورد شود چراکه مروجان تاثیر بسیار زیادی در گسترش دامنه‌ی مشتریان این برنامه و افزایش سودآوری ما دارند. همچنین از کسانی که امتیاز می‌دهند ولی امتیاز بالایی نمی‌دهند باید تقدیر کرد و اشکالاتی که گوشزد کرده‌اند را رفع کرد چراکه این افراد از روی دلسوزی این کار را کرده‌اند و می‌خواسته‌اند که در برنامه تغییرات مثبتی به نفعشان ایجاد گردد.

۲. افرادی که تمایل زیادی به کتاب‌های طولانی دارند. این افراد ظرفیت تبدیل‌شدن به مروجان را دارند، اگر اشکالی که در کتاب‌ها وجود دارد برطرف شود. تعداد زیادی از کسانی که کتاب‌ها را خریده‌اند و به مقداری از آن‌ها گوش داده‌اند، دیگر هرگز علاقه‌ای به خرید مجدد نداشته‌اند. با احتمال بسیار بالایی کیفیت کتاب‌های صوتی به قدری پایین است که آن‌ها را از خرید پشمیان می‌کند. کسانی که کتاب‌های طولانی می‌خرند وقتی از کیفیت کتاب‌ها ناامید می‌شوند، ضرر زیادی را به کمپانی می‌توانند متحمل کنند چراکه مشتریان بالقوه‌ی پر سودی بودند که از دست رفته‌اند.

۳. کسانی که انگیجمنت زیادی با محصول دارند. این افراد کسانی هستند که به خوبی می‌توانند اشکالات نرم‌افزاری ما را گوش‌زد کنند. باید بر این ظرفیت تمرکز شود تا ایرادات نرم‌افزاری که خوشه‌ی دوم به شدت در مقابل آن‌ها آسب‌پذیر هستند، برطرف شوند و از این طریق، هم کسانی که انگیجمنت زیادی با برنامه دارند، هم کسانی که NPS پایینی دارند و هم کسانی که به کتاب‌های صوتی بلند علاقه دارند، خوشنود می‌شوند و می‌تواند آن‌ها را به خوشه‌ی اول منتقل کند و سودآوری زیادی داشته باشد.

۴. سایر مشتریان که به نظر می‌رسد با اهداف تفریحی یا برای آزمایش‌کردن از این برنامه خرید کرده‌اند و به طور کلی این برنامه برای آن‌ها اهمیت خاصی ندارد. می‌توان در کمپین‌ها و موارد مرتبط با ترویج، روی این دسته تمرکز کمی داشت و امکانات کمی را به آن‌ها اختصاص داد.

۳ پیش‌بینی خرید مجدد مشتری

چون در کلاسترینگ و قسمت پیش‌پردازش، کارهای لازم برای عملیات‌های مختلف داده‌کاوی و یادگیری ماشین انجام شده‌است، بعد از بررسی توازن داده‌ها، مستقیم به سراغ متد طبقه‌بند می‌روم.

[55]:
```
customers = mining_df.copy()
customers['Target'].value_counts()
```

[55]:
```
0.0    11093
1.0     1958
Name: Target, dtype: int64
```

همانطور که مشاهده می‌شود، داده‌ها با نسبت ۱۱ به ۲، متوازن نیستند. برای حل این مشکل می‌توان متوازن‌سازی را با کتاب‌خانه‌ای

مثل imblearn انجام داد ولی من در اینجا به این خاطر که داده‌های مفید زیادی در این بین آسیب می‌بینند، ترجیح دادم از الگوریتم‌هایی که به این مسئله حساس‌نیستند و همچنین از معیارهایی به جز معیار دقت استفاده کنم تا حساسیت روش طبقه‌بندی به متوازن‌نبودن داده‌ها کم شود.

[56]:
```
test = customers['Target']
train = customers.drop('Target', axis=1)
train.head()
```

[56]:

	Book Length Average	Price Average	Listened Percent	Support Request	\
1	1.000000	0.011568	0.00	0.000000	
3	0.722222	0.016525	0.42	0.033333	
5	1.000000	0.005902	0.00	0.000000	
7	0.222222	0.011568	0.00	0.000000	
8	1.000000	0.011568	0.26	0.000000	

	Engagement Time	Net Promoter Score	Purchases Count
1	0.000000	0.0	0.0
3	0.278017	0.0	0.0
5	0.000000	0.0	0.0
7	0.000000	0.0	0.0
8	0.071121	0.0	0.0

مسئله‌ای که وجود دارد این است که من نمی‌دانم کدام الگوریتم برای این طبقه‌بندی مناسب است، پس ابتدا چند مدل را انتخاب کردم و سپس از هر کدام از آن‌ها یک خروجی اعتبارسنجی به همراه چهار معیار Accuracy ، F1 Score ، Roc AUC و Recall را گرفتم.

[57]:
```
validation_size = 0.30
num_folds = 10
X_train, X_validation, Y_train, Y_validation = train_test_split(train, test,␣
 ↪test_size=validation_size)

models = []
models.append(('Decision Tree Gini', DecisionTreeClassifier(criterion='gini')))
models.append(('Bernoulli NB', BernoulliNB()))
models.append(('Gaussian NB', GaussianNB()))
```

```
models.append(('Random Forest', RandomForestClassifier()))
models.append(('Gradient Boosting Classifier', GradientBoostingClassifier()))
models.append(('SVM', svm.SVC(class_weight='balanced', probability=True,␣
 ↪kernel='linear')))
```

مدل‌های Gradient Boosting Classifier و Random Forest که بر اساس گروه‌های درختی هستند، تا حد خوبی در برابر داده‌های نامتوازن خوب عمل می‌کنند. همچنین Decision Tree Gini از الگوریتم‌های کلاسیکی است که در برابر داده‌های نامتوازن می‌تواند مفید باشد. الگوریتم SVM در حالت عادی چندان در برابر داده‌های نامتوازن خوب عمل نمی‌کند، برای همین من قسمت‌های class_weight و probability را طوری تنظیم کردم که از طریق Penalized-SVM به بهبود SVM در برابر داده‌های متوازن بپردازم. دو الگوریتم دیگر یعنی Bernoulli NB و Gaussian NB که بر اساس نیو بیز و احتمالات هستند، مشخصا در برابر داده‌های نامتوازن عملکرد خوبی ندارند ولی خوب است که بررسی کنیم که چه میزان تفاوت بین این الگوریتم‌ها وجود دارد.

[58]:
```
results = []
for name, model in models:
    kfold = KFold(n_splits=num_folds)
    acc_results = cross_val_score(model, X_train, Y_train, cv=kfold,␣
 ↪scoring='accuracy')
    f1_results = cross_val_score(model, X_train, Y_train, cv=kfold,␣
 ↪scoring='f1')
    roc_results = cross_val_score(model, X_train, Y_train, cv=kfold,␣
 ↪scoring='roc_auc')
    recall_results = cross_val_score(model, X_train, Y_train, cv=kfold,␣
 ↪scoring='recall')
    results.append(
        pd.DataFrame({
            'Name': name,
            'Accuracy': acc_results.mean(),
            'F1': f1_results.mean(),
            'Roc_auc': roc_results.mean(),
            'Recall': recall_results.mean()
        }, index=[0]),
    )
```

[59]:
```
results_df = pd.concat(results, ignore_index=True)
names = list(results_df['Name'])
results_df = results_df.set_index('Name')
results_df
```

[59]:

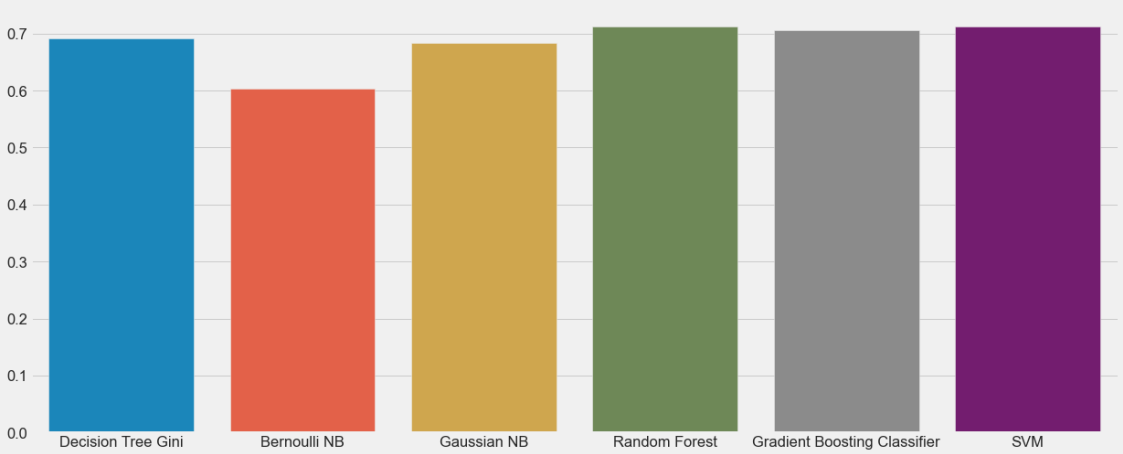
Name	Accuracy	F1	Roc_auc	Recall
Decision Tree Gini	0.888669	0.589433	0.767673	0.519080
Bernoulli NB	0.886261	0.390465	0.890114	0.243868
Gaussian NB	0.901586	0.547382	0.886939	0.396847
Random Forest	0.895456	0.586927	0.865248	0.502194
Gradient Boosting Classifier	0.909797	0.586552	0.901421	0.426888
SVM	0.855717	0.549414	0.859742	0.585728

[60]:
```
results_df.mean(axis=1)
```

[60]:
```
Name
Decision Tree Gini              0.691214
Bernoulli NB                    0.602677
Gaussian NB                     0.683189
Random Forest                   0.712456
Gradient Boosting Classifier    0.706164
SVM                             0.712650
dtype: float64
```

[61]:
```
_, ax = plt.subplots(1, 1, figsize=(19, 8), sharex=True)
sns.barplot(x=names, y= list(results_df.mean(axis=1)), ax=ax)
```

[61]:
```
<AxesSubplot:>
```



همانطور که مشاهده می‌شود. الگوریتم Penalized-SVM بهترین عملکرد را در بین دیگران دارد. بر همین اساس، من طبق همین الگوریتم، طبقه‌بندی خود را به پیش می‌برم.

برای استخراج ماتریس درهم‌ریختگی مدل، از تابع کلسیفایری که نوشته شده‌است، استفاده می‌کنم.

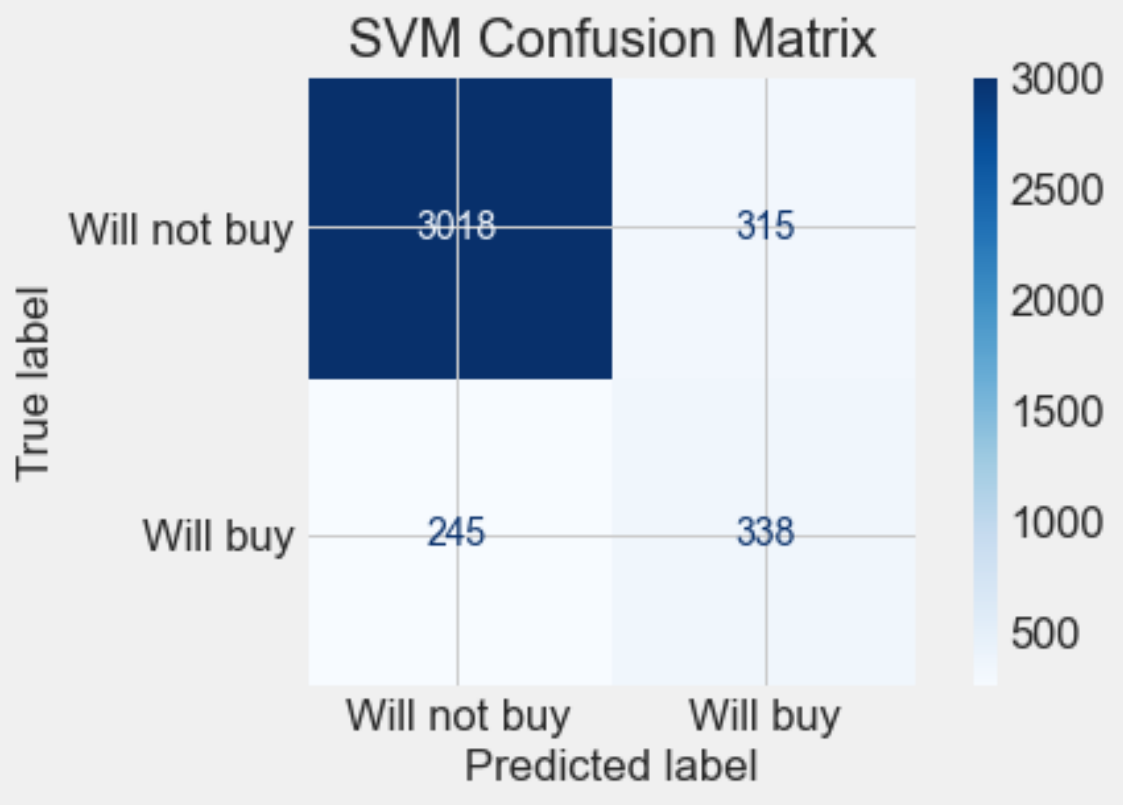
[62]:
```
target_names=['Will not buy', 'Will buy']
name, clsfr = models[5]
clsfr.fit(X_train, Y_train)
predications = clsfr.predict(X_validation)
print(classification_report(Y_validation, predications))
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.91      0.92      3333
         1.0       0.52      0.58      0.55       583

    accuracy                           0.86      3916
   macro avg       0.72      0.74      0.73      3916
weighted avg       0.86      0.86      0.86      3916
```

[63]:
```
disp = ConfusionMatrixDisplay.from_estimator(
    clsfr,
    X_validation,
```

```
    Y_validation,
    display_labels=target_names,
    cmap=plt.cm.Blues,
    normalize=None,
)
disp.ax_.set_title(f'{name} Confusion Matrix')
print(disp.confusion_matrix)
```

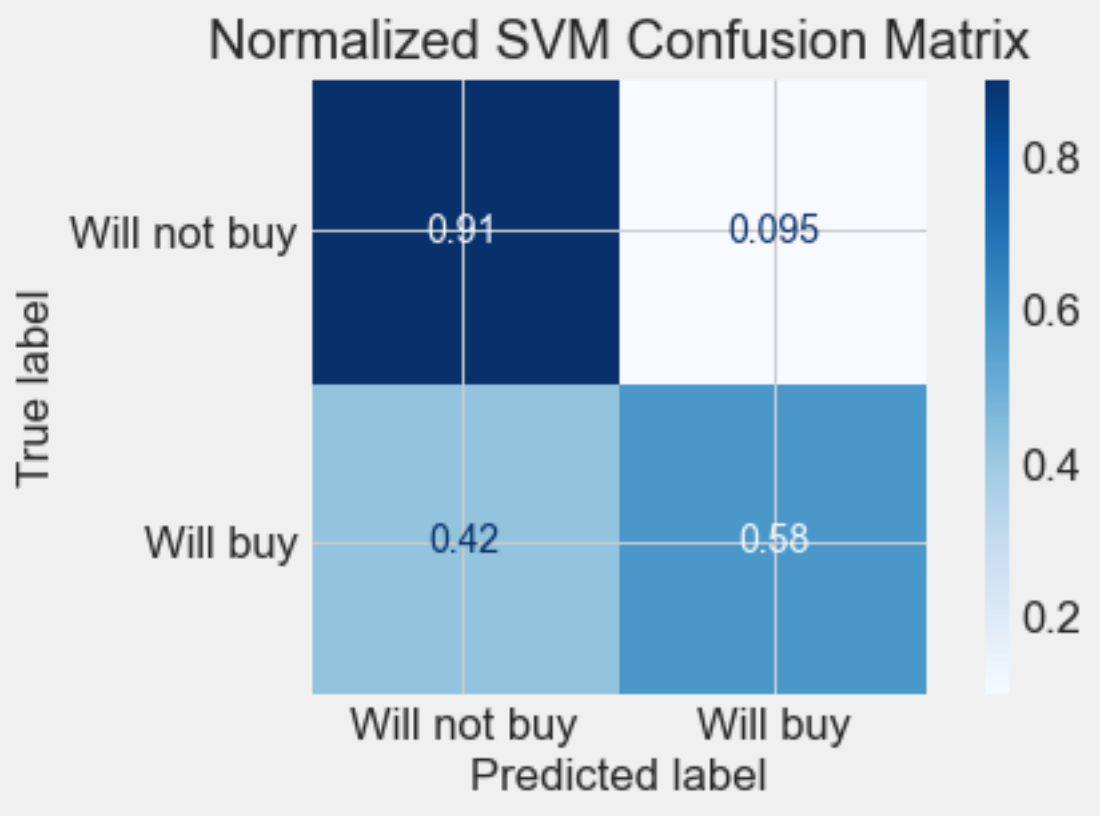
```
[[3018  315]
 [ 245  338]]
```



[64]:
```
disp = ConfusionMatrixDisplay.from_estimator(
    clsfr,
    X_validation,
    Y_validation,
    display_labels=target_names,
```

```
    cmap=plt.cm.Blues,
    normalize='true',
)
disp.ax_.set_title(f'Normalized {name} Confusion Matrix')
print(disp.confusion_matrix)
```

```
[[0.90549055 0.09450945]
 [0.42024014 0.57975986]]
```



از طریق اطلاعاتی که به دست می‌آید، متوجه عملکرد این نوع از الگوریتم SVM در مورد پیش‌بینی خرید مجدد می‌شویم:

- ۳۳۵۶ پیش‌بینی درست
- دقت ۹۳ درصدی در مورد اهداف ۰ و تشخیص مطلوب احتمال نخریدن در آینده
- دقت وزن‌دار متوسط ۸۶ درصدی
- تشخیص اهداف ۱ در حد بالاتر از ۵۰ درصد.

۴ جمع‌بندی

بعد از مقداری پیش‌پردازش و مناسسازی مجموعه‌ی داده، عملیات بخش‌بندی بر اساس RFM انجام شد که مشخص شد در این نوع از کاربرد، مشتریانی با تعداد خرید بالا و مقدار پول پرداختی بیشتر، اهمیت بیشتری برای ما دارند، هر چند که از تاریخ خرید آن‌ها مقداری گذشته باشد. سپس با استفاده از خوشه‌بندی به این نتیجه رسیدیم که این برنامه ۴ خوشه‌ی اصلی را در برمی‌گیرد که این خوشه‌ها شامل مروجان، کتاب بلند خواهان، تعامل‌گرایان و تفریح‌کنندگان می‌شود و می‌توان بر اساس این خشوه‌ها به افزایش درآمد و سود و افزایش میزان رضایت مشتریان رسید. در گام آخر با استفاده از متدهای طبقه‌بندی، مدلی مطلوب برای پیش‌بینی نخریدن یک کاربر در آینده فراهم شد که تا حدی می‌تواند خریدن مجدد را نیز پیش‌بینی درستی بکند.