Team Project 2022

H&M PERSONALIZED FASHION RECOMMENDATIONS

Chenlu Jiang (chenluj@andrew.cmu.edu)
Jingyi Wu (jingyiw2@andrew.cmu.edu)
Shengyi Yan (syan2@ andrew.cmu.edu)
Haorong Zhu (haorong2@andrew.cmu.edu)

Content

1. Business Understanding

1.1 Background

H&M is a fast fashion brand that was first established in Sweden. The brand offers the latest style and the best prices to customers. H&M consists of inexpensive clothing necessities, perfect appearance accessories, and sportswear for women, men, teenagers, and children. It also includes living items from H&M Home. H&M operates 4420 stores, 72 store markets, and 49 online markets worldwide. H&M was considered as the second most valuable clothing brand in the world after Nike in 2018.

1.2 Situation Assessment

Every day, H&M handles large amounts of orders from its multiple sales channels, and product volume is tremendous. By giving accurate predictions on what items are best sellers and offering personalized recommendations, H&M increases sales and save inventory costs by optimizing supply chain management. It is a waste of money and resources to have inventories piled up in stores or warehouses and became outdated or deadstock. Providing satisfactory personalized recommendations is also important for H&M to retain and attract more customers and improve benefits.

1.3 Models Objectives

Here, we want to first analyze the datasets to develop models in helping H&M identify valuable customers and then use text analysis technique and association rules to provide personalized product recommendation which customers are most likely to buy in the future.

1.4 Project Plan

In general, our project will address two problems.
Problem1: Segment customers and define high-value customers; use machine learning models to find out important features for identifying potential high-value customers. Here various classification models will be used.
Problem2: Develop algorithms to provide personalized article recommendations based on association rules and text analysis.

1.4.1 Classification Models

In first problem, we will define high-value customers to do customer segmentation and build classification models to help H&M to identify important predictors.

We calculate customers' annual consumption in 2020 and define high-value customers as those with top 25% consumption, medium-value customers as those with middle 50% consumption and low-value customers as bottom 25%. (This segmentation is based on long-tail theory -- 20% of the customers contribute to 80% of total sales). Then we will use these customers' transactions history data in 2019 to get attributes such as age, buying channels, and different amounts of items a person bought, and other customers features and behaviors as predictors. By building a classification model on these, we could use the customer's previous purchase characteristics and buying behaviors to predict his/her value in the future.

H&M can use this list to do customers analysis. By analyzing attributes of customers, such as age, buying channel, etc., H&M can find out common characteristics of high-value customers and better target them in future marketing and strategy. For example, it could assign more marketing budget for high-value customers' favorable buying channel; it can replenish its stock with more clothes high-value customers prefer; it can also get a high-value customers' portrait.

1.4.2 Text analysis

The next model we use for giving predictions is text mining techniques TF-IDF. Article attributes in articles table tend to give a detailed description of articles and the information combined together can be viewed as a "document" describing this article. TF-IDF stands for term-frequency-inverse document frequency and quantifies the importance of relevance of string representations in the "document". We use a document's TF-IDF values for all words to represent this document and calculate similarity between document pairs. In this way, we can find out similar products to what customers already bought and recommend these to them.

1.4.3 Association Models

In personalized recommendation part, due to the large amount of data and our limited computer resources, we will use a subset of data in training our model, but models in this part are capable of being expanded to the whole dataset.

We will choose high-value customers' transaction data in May 2020 and recommend similar products to what they bought in May as predictions of possible items they will buy in the near future. We use their real purchases in the next three months to test our predictions' accuracy.

Association Rules Model treats customer's purchases as the articles bundles and find out highly-related articles. By applying association rules, we sort pairs of items that are usually bought together for all the customers in HM and give top 10 highly related articles as recommendations for each customer in May 2020 cohort.

2. Data Understanding

H&M provides us with 3 datasets, the articles.csv, the customers.csv, and the transaction.csv[1].

The Articles Dataset

The articles Dataset includes all the items that H&M currently has. It consists of 105, 542 articles with 25 attributes.

---

[1] https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations/data

```
articles.head()
```

	article_id	product_code	prod_name	product_type_no	product_type_name	product_group_name	graphical_appearance_no	graphical_appearance_name	colour_group_code	colour_group_name	...	department_name	index_code	index_name	index_group_no	index_group_name	section_no
0	108775015	108775	Strap top	253	Vest top	Garment Upper body	1010016	Solid	9	Black	...	Jersey Basic	A	Ladieswear	1	Ladieswear	16
1	108775044	108775	Strap top	253	Vest top	Garment Upper body	1010016	Solid	10	White	...	Jersey Basic	A	Ladieswear	1	Ladieswear	16
2	108775051	108775	Strap top (1)	253	Vest top	Garment Upper body	1010017	Stripe	11	Off White	...	Jersey Basic	A	Ladieswear	1	Ladieswear	16
3	110065001	110065	OP T-shirt (Idro)	306	Bra	Underwear	1010016	Solid	9	Black	...	Clean Lingerie	B	Lingeries/Tights	1	Ladieswear	61
4	110065002	110065	OP T-shirt (Idro)	306	Bra	Underwear	1010016	Solid	10	White	...	Clean Lingerie	B	Lingeries/Tights	1	Ladieswear	61

5 rows × 25 columns

```
articles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105542 entries, 0 to 105541
Data columns (total 25 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   article_id                    105542 non-null  int64
 1   product_code                  105542 non-null  int64
 2   prod_name                     105542 non-null  object
 3   product_type_no               105542 non-null  int64
 4   product_type_name             105542 non-null  object
 5   product_group_name            105542 non-null  object
 6   graphical_appearance_no       105542 non-null  int64
 7   graphical_appearance_name     105542 non-null  object
 8   colour_group_code             105542 non-null  int64
 9   colour_group_name             105542 non-null  object
 10  perceived_colour_value_id     105542 non-null  int64
 11  perceived_colour_value_name   105542 non-null  object
 12  perceived_colour_master_id    105542 non-null  int64
 13  perceived_colour_master_name  105542 non-null  object
 14  department_no                 105542 non-null  int64
 15  department_name               105542 non-null  object
 16  index_code                    105542 non-null  object
 17  index_name                    105542 non-null  object
 18  index_group_no                105542 non-null  int64
 19  index_group_name              105542 non-null  object
 20  section_no                    105542 non-null  int64
 21  section_name                  105542 non-null  object
 22  garment_group_no              105542 non-null  int64
 23  garment_group_name            105542 non-null  object
 24  detail_desc                   105126 non-null  object
dtypes: int64(11), object(14)
memory usage: 20.1+ MB
```

```
articles.nunique()
```

```
article_id                      105542
product_code                     47224
prod_name                        45875
product_type_no                    132
product_type_name                  131
product_group_name                  19
graphical_appearance_no             30
graphical_appearance_name           30
colour_group_code                   50
colour_group_name                   50
perceived_colour_value_id            8
perceived_colour_value_name          8
perceived_colour_master_id          20
perceived_colour_master_name        20
department_no                      299
department_name                    250
index_code                          10
index_name                          10
index_group_no                       5
index_group_name                     5
section_no                          57
section_name                        56
garment_group_no                    21
garment_group_name                  21
detail_desc                      43404
dtype: int64
```

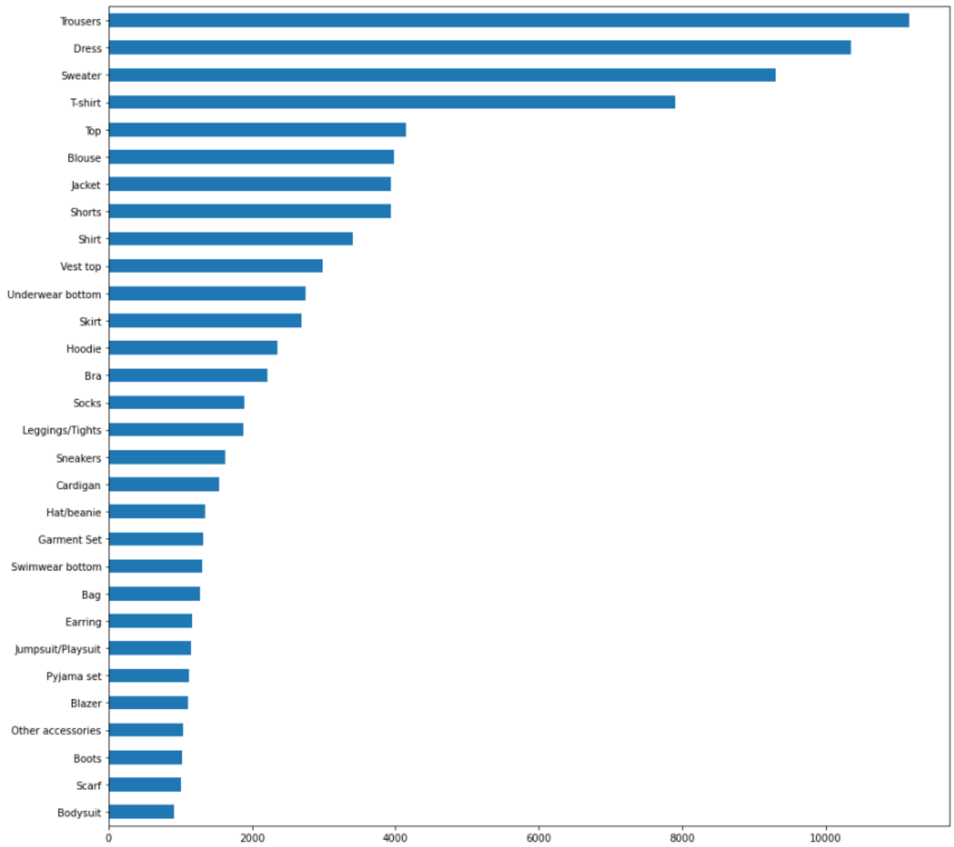
Apart from some common attributes such as **article_id**, **prod_name** and **product_code**, the dataset also provides us with some useful attributes such as the color, the perceived color, and the description of the article.

Among these attributes, most of them are categorical features. Their categories are divided based on H&M's methodology, for example types, departments, sections, garment groups, and so on.

For example, **product_type_no** and **product_type_name** are two attributes that H&M uses to categorize the articles. H&M put the articles into different types and assign them a type series number. From the chart below, we can see that most of the articles are trousers and dress.

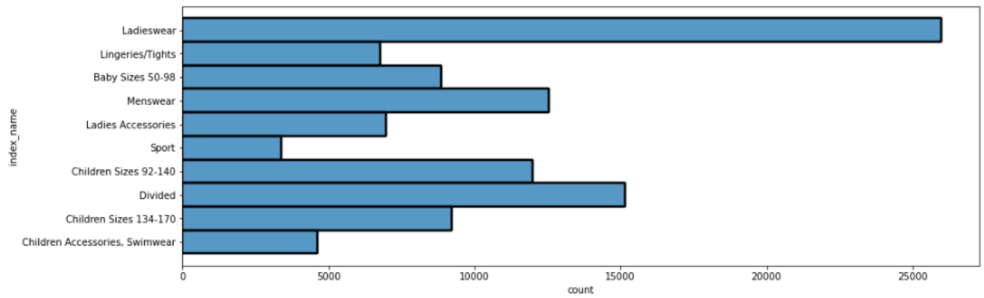
```
# The amount of different product type of articles
serie = articles.loc[:, "product_type_name"].value_counts(ascending=True)[-30:]
serie.plot(kind='barh', figsize=(15, 15))
```

```
<AxesSubplot:>
```



For another attribute, **index_name**, is also similar.

```
# The amount of different type of articles
f, ax = plt.subplots(figsize=(15, 5))
ax = sns.histplot(data=articles, y='index_name')
ax.set_xlabel('count')
ax.set_ylabel('index_name')
plt.show()
```



The Customers Dataset

The Customers Dataset includes all the customer information, consisting of 1,371,980 customers with 7 attributes.

```
customers.head()
```

	customer_id	FN	Active	club_member_status	fashion_news_frequency	age	postal_code
0	00000dbacae5abe5e23885899a1fa44253a17956c6d1c3...	NaN	NaN	ACTIVE	NONE	49.0	52043ee2162cf5aa7ee79974281641c6f11a68d276429a...
1	0000423b00ade91418cceaf3b26c6af3dd342b51fd051e...	NaN	NaN	ACTIVE	NONE	25.0	2973abc54daa8a5f8ccfe9362140c63247c5eee03f1d93...
2	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	NaN	NaN	ACTIVE	NONE	24.0	64f17e6a330a85798e4998f62d0930d14db8db1c054af6...
3	00005ca1c9ed5f5146b52ac8639a40ca9d57aeff4d1bd2...	NaN	NaN	ACTIVE	NONE	54.0	5d36574f52495e81f019b680c843c443bd343d5ca5b1c2...
4	00006413d8573cd20ed7128e53b7b13819fe5cfc2d801f...	1.0	1.0	ACTIVE	Regularly	52.0	25fa5ddee9aac01b35208d01736e57942317d756b32ddd...

```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1371980 entries, 0 to 1371979
Data columns (total 7 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   customer_id             1371980 non-null  object
 1   FN                      476930 non-null   float64
 2   Active                  464404 non-null   float64
 3   club_member_status      1365918 non-null  object
 4   fashion_news_frequency  1355971 non-null  object
 5   age                     1356119 non-null  float64
 6   postal_code             1371980 non-null  object
dtypes: float64(3), object(4)
memory usage: 73.3+ MB
```

```
customers.nunique()
```

```
customer_id               1371980
FN                              1
Active                          1
club_member_status              3
fashion_news_frequency          4
age                            84
postal_code                352899
dtype: int64
```

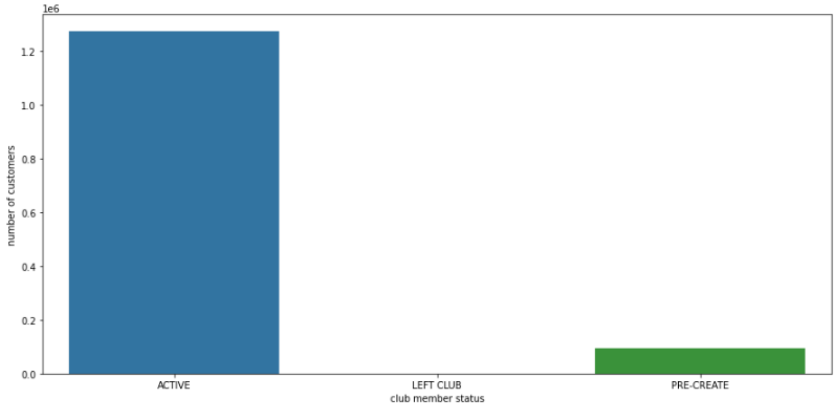
Also, there are some missing values for some attributes. Note that over 50% of values in FN and Active columns are missing here, which implies bad data quality, and we decide not to use them in our models.

```
# Missing Value
customers.isnull().sum()
```

```
customer_id                    0
FN                        895050
Active                    907576
club_member_status          6062
fashion_news_frequency     16009
age                        15861
postal_code                    0
dtype: int64
```

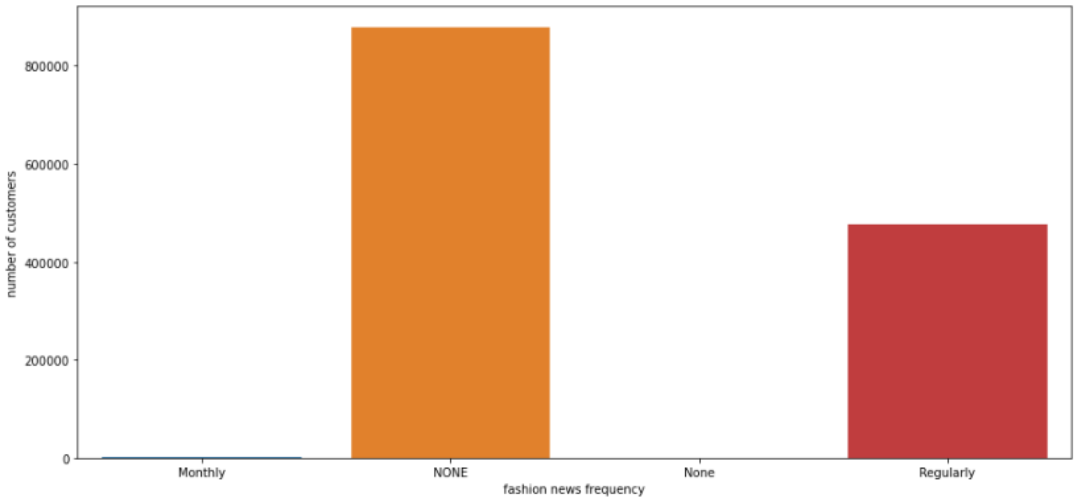
For **club_member_status**, this is an attribute to show the status of the customers. It has 3 unique values, ACTIVE, LEFT CLUB, and PRE-CREATED. We can see most of these customers are active members and a few of them are at other statuses. We believe this attribute has little effect on the model, so we decide to drop it.

```python
# Customers' Membership Distribution
fig, ax = plt.subplots(1, 1, figsize=(15, 7))
club_member_statuses = customers.groupby('club_member_status', as_index=False)['customer_id'].count()
ax = sns.barplot(data=club_member_statuses, x='club_member_status', y='customer_id')
plt.xlabel("club member status")
plt.ylabel("number of customers")
plt.show()
```



The **fashion_news_frequency** describes the frequency of the customers receiving the fashion news. We can see the distribution as the following:

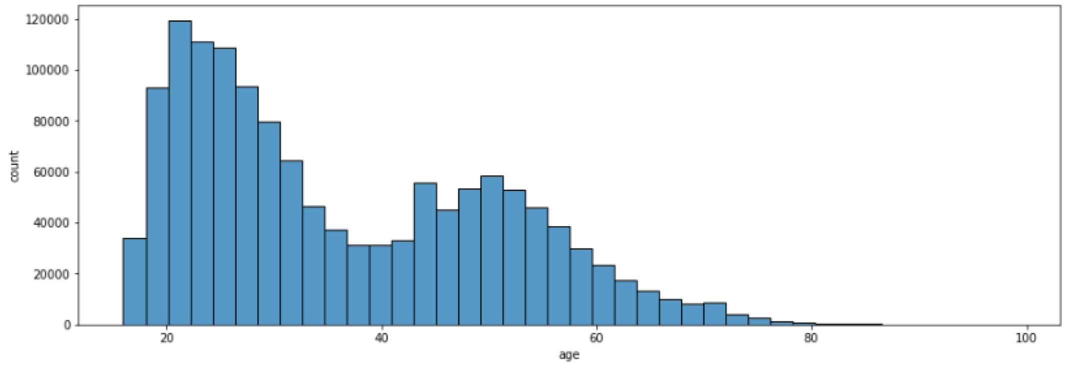
```python
# Fashion News Frequency
fig, ax = plt.subplots(1, 1, figsize=(15, 7))
club_member_statuses = customers.groupby('fashion_news_frequency', as_index=False)['customer_id'].count()
ax = sns.barplot(data=club_member_statuses, x='fashion_news_frequency', y='customer_id')
plt.xlabel("fashion news frequency")
plt.ylabel("number of customers")
plt.show()
```



Most of customers have no habit of receiving fashion news.

Moreover, the dataset also provides the age of the customers. Most of these customers are between 20 and 30 years old. The maximum age could reach over 80. The median age is 32 years old.

```python
# Age Distribution
f, ax = plt.subplots(figsize=(15, 5))
ax = sns.histplot(data=customers, x='age', bins=40)
ax.set_xlabel('age')
ax.set_ylabel('count')
plt.show()

median_age = customers['age'].median(skipna=True)
print(f"The median of age is {median_age}")
```




```
The median of age is 32.0
```

The Transaction Dataset

The Transaction Dataset includes the transaction history from 2018-09-20 to 2020-09-20. It consists of 31,788,324 transactions with 5 attributes. Each instance represents a transaction of a customer for a single article.

```python
transactions.head()
```

	t_dat	customer_id	article_id	price	sales_channel_id
0	2018-09-20	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	663713001	0.050831	2
1	2018-09-20	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	541518023	0.030492	2
2	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	505221004	0.015237	2
3	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	685687003	0.016932	2
4	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	685687004	0.016932	2

```python
transactions.info()
```

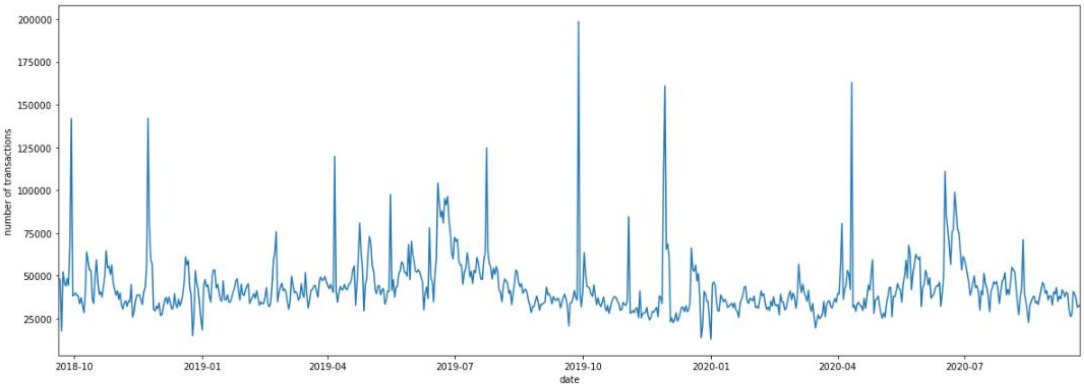
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31788324 entries, 0 to 31788323
Data columns (total 5 columns):
 #   Column            Dtype
---  ------            -----
 0   t_dat             datetime64[ns]
 1   customer_id       object
 2   article_id        int64
 3   price             float64
 4   sales_channel_id  int64
dtypes: datetime64[ns](1), float64(1), int64(2), object(1)
memory usage: 1.2+ GB
```

There is no missing value for this dataset, which is good.

```python
# Missing Value
transactions.isnull().sum()
```

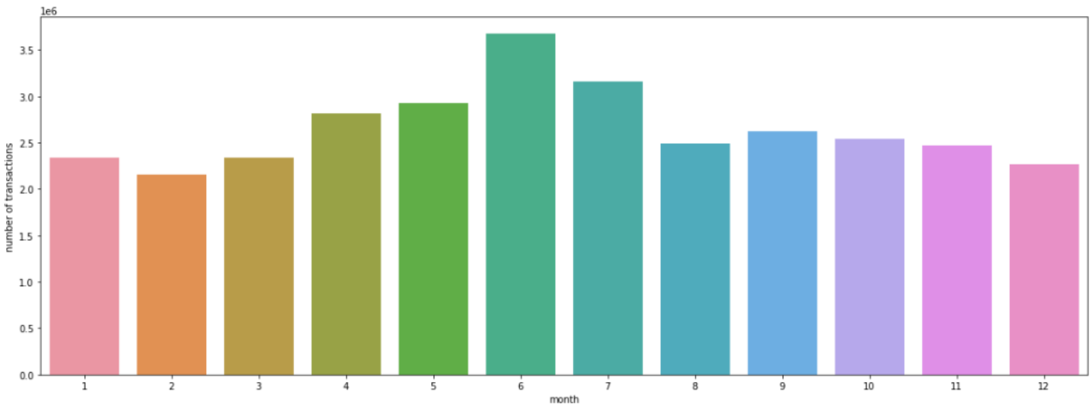
```
t_dat               0
customer_id         0
article_id          0
price               0
sales_channel_id    0
dtype: int64
```

```python
# Daily Transaction
transactions_per_day = transactions.groupby('t_dat', as_index=False)['article_id'].count()

fig, ax = plt.subplots(1, 1, figsize=(20, 7))
plt.plot(transactions_per_day['t_dat'], transactions_per_day['article_id'])
plt.xlabel("date")
plt.ylabel("number of transactions")
ax.set_xlim(transactions_per_day['t_dat'].min(), transactions_per_day['t_dat'].max())

plt.show()
```



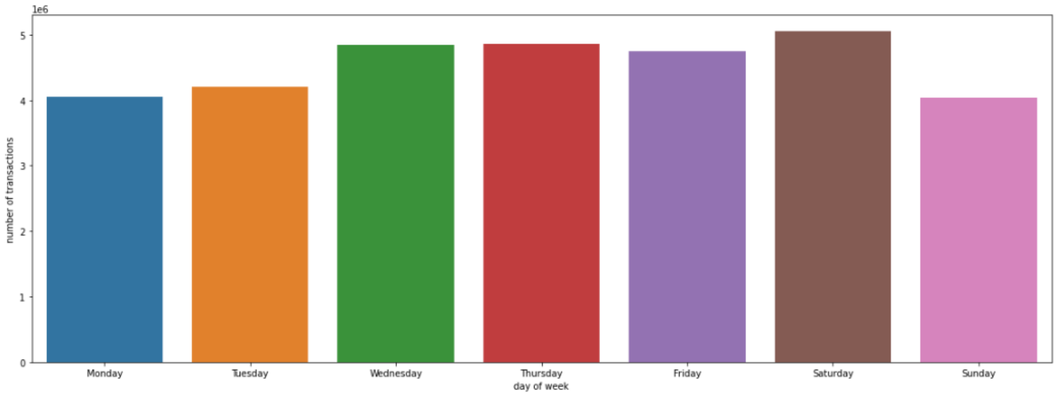
We can see some patterns in the line chart. We have a huge volume of transactions happening in October 2019.

```python
# Monthy Sales
transactions['t_dat_month'] = transactions['t_dat'].apply(lambda date: date.month)

fig, ax = plt.subplots(1, 1, figsize=(20, 7))
transactions_by_month = transactions.groupby('t_dat_month', as_index=False)['article_id'].count()
ax = sns.barplot(data=transactions_by_month, x='t_dat_month', y='article_id')
plt.xlabel("month")
plt.ylabel("number of transactions")
plt.show()
```



If we just look at the monthly distribution, most transactions occurred in June and July and few in winter and spring.

```
# Weekdays Distribution
import calendar
transactions['t_dat_weekday'] = transactions['t_dat'].apply(lambda date: date.weekday())
fig, ax = plt.subplots(1, 1, figsize=(20, 7))
transactions_by_weekday = transactions.groupby('t_dat_weekday', as_index=False)['article_id'].count()
transactions_by_weekday['t_dat_weekday'] = transactions_by_weekday['t_dat_weekday'].apply(lambda x: calendar.day_name[x])
ax = sns.barplot(data=transactions_by_weekday, x='t_dat_weekday', y='article_id')
plt.xlabel("day of week")
plt.ylabel("number of transactions")
plt.show()
```



We have the highest volume of transactions on Saturday. For Monday, Tuesday and Sunday, the sales are lower. We can arrange our employee schedule based on this. Also, for Sunday, the management team should go to discover the reason why Sunday, a non-working day, has this few numbers of transactions.

3. Data Preparation

3.1 Divide the Transaction into 2 period

We are going to build a model to predict the future importance of a customer. If we want to use many X to predict Y. We need to collect our X, the customers' buying behavior, and feature attributes from 2019. We get the Y, our target variable, from the data for 2020. Thus, first, we need to divide the transaction dataset into two time periods.

```
STARTING_DATE = '2018-9-20'
split_point = '2019-9-20'
ENDING_DATE = '2020-9-20'

transactions_2019_2020 = transactions[transactions['t_dat'] >= STARTING_DATE]
transactions_2019 = transactions_2019_2020[transactions_2019_2020['t_dat'] < split_point]
transactions_2019.shape
```

```
(16803901, 5)
```

```
transactions_2020 = transactions_2019_2020[transactions_2019_2020['t_dat'] >= split_point]
transactions_2020 = transactions_2020[transactions_2019_2020['t_dat'] < ENDING_DATE]
transactions_2020.shape
```

```
<ipython-input-60-e24a486244df>:2: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  transactions_2020 = transactions_2020[transactions_2019_2020['t_dat'] < ENDING_DATE]
```

```
(14887938, 5)
```

We split the transaction in the point of 2019-9-20.

3.2 Merge the Transactions and Articles Dataset

We are going to combine the transaction data with articles data and keep the attributes that are useful for our models.

There are many attributes describing the category of the article, eventually, we pick the most distinct and meaningful attributes as shown below.

```
transactions_articles_joined = transactions_2019.merge(articles, on='article_id')
reserved_columns = ['customer_id', 'article_id','price', 'sales_channel_id', 'product_group_name',
                    'perceived_colour_master_name', 'department_name', 'index_name']
transactions_articles_joined = transactions_articles_joined[reserved_columns]
transactions_articles_joined.head()
```

	customer_id	article_id	price	sales_channel_id	product_group_name	perceived_colour_master_name	departmen
0	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	663713001	0.050831	2	Underwear	Black	Exp
1	3681748607f3287d2c3a65e00bb5fb153de30e9becf158...	663713001	0.049475	2	Underwear	Black	Exp
2	4ef5967ff17bf474bffebe5b16bd54878e1d4105f7b4ed...	663713001	0.050831	2	Underwear	Black	Exp
3	6b7b10d2d47516c82a6f97332478dab748070f09693f09...	663713001	0.050831	1	Underwear	Black	Exp
4	8ac137752bbe914aa4ae6ad007a9a0c5b67a1ab2b2d474...	663713001	0.050831	2	Underwear	Black	Exp

3.3 Modify the fashinon_news_frequency NONE value

To improve the classification models' performance, we decide to turn the string description into numbers. 0 represents NONE, 1 represents Monthly, and 2 represents the other circumstances.

```
# Modify fashion_news_frequency None Value
customers['fashion_news_frequency'] = customers['fashion_news_frequency'].fillna('NONE')
customers.loc[customers['fashion_news_frequency'] == 'None', 'fashion_news_frequency'] = 'NONE'

def frequency_type_to_code(type):
    if type == 'NONE':
        return 0
    elif type == 'Monthly':
        return 1
    else:
        return 2

customers['fashion_news_frequency'] = customers['fashion_news_frequency'].apply(lambda x: frequency_type_to_code(x))
```

3.4 Drop useless attributes

Some attributes, because they have many null values and some of them are hard to analyze and interpret, such as the encrypted postal code. We decided to drop them.

```
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1371980 entries, 0 to 1371979
Data columns (total 7 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   customer_id             1371980 non-null  object
 1   FN                      476930 non-null   float64
 2   Active                  464404 non-null   float64
 3   club_member_status      1365918 non-null  object
 4   fashion_news_frequency  1371980 non-null  int64
 5   age                     1356119 non-null  float64
 6   postal_code             1371980 non-null  object
dtypes: float64(3), int64(1), object(3)
memory usage: 73.3+ MB
```

```
# Drop useless columns
customers = customers.drop(['FN', 'Active', 'club_member_status', 'postal_code'], axis=1)
```

After dropping, the Customers Dataset is shown below:

```
customers.head()
```

	customer_id	fashion_news_frequency	age
0	00000dbacae5abe5e23885899a1fa44253a17956c6d1c3...	0	49.0
1	0000423b00ade91418cceaf3b26c6af3dd342b51fd051e...	0	25.0
2	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	0	24.0
3	00005ca1c9ed5f5146b52ac8639a40ca9d57aeff4d1bd2...	0	54.0
4	00006413d8573cd20ed7128e53b7b13819fe5cfc2d801f...	2	52.0

3.5 Fill null value in Age with the median

Age has many null values, we decided to fill it with the median.

```
# Fill Age null value
median_age = customers['age'].median(skipna=True)
customers['age'] = customers['age'].fillna(median_age)
```

3.6 Merge Customers Dataset with the merged Transactions_Artibles Dataset

```
# Join cumstomers dataset to transactions_articles
all_joined = transactions_articles_joined.merge(customers, on='customer_id')
reserved_columns.extend(['age', 'fashion_news_frequency'])
all_joined = all_joined[reserved_columns]
all_joined.head()
```

	customer_id	article_id	price	sales_channel_id	product_group_name	perceived_colour_master_name	department_
0	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	663713001	0.050831	2	Underwear	Black	Expre Li
1	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	663713001	0.050831	2	Underwear	Black	Expre Li
2	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	541518023	0.030492	2	Underwear	Pink	Casual Li
3	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	578020002	0.013542	2	Garment Upper body	Blue	E
4	3681748607f3287d2c3a65e00bb5fb153de30e9becf158...	663713001	0.049475	2	Underwear	Black	Expre Li

3.7 Turn nominal attributes into dummy variables.

As mentioned above, we have some attributes about the type and the color and some other nominal attributes in the current dataset. We decide to make them into dummy variables for our next step.

```
# OneHotEncoding
all_joined_ohe = pd.get_dummies(all_joined, columns=all_joined.columns[4:-2])
all_joined_ohe.shape
```

```
(16803901, 293)
```

```
all_joined_ohe.columns
```

```
Index(['customer_id', 'article_id', 'price', 'sales_channel_id', 'age',
       'fashion_news_frequency', 'product_group_name_Accessories',
       'product_group_name_Bags', 'product_group_name_Cosmetic',
       'product_group_name_Furniture',
       ...
       'index_name_Baby Sizes 50-98',
       'index_name_Children Accessories, Swimwear',
       'index_name_Children Sizes 134-170', 'index_name_Children Sizes 92-140',
       'index_name_Divided', 'index_name_Ladies Accessories',
       'index_name_Ladieswear', 'index_name_Lingeries/Tights',
       'index_name_Menswear', 'index_name_Sport'],
      dtype='object', length=293)
```

3.8 Select the customers with more than 5 transactions

In order to get meaningful insight, we decided to concentrate on the customers who had more than 5 transactions. We don't want some outlier or noise in our dataset.

```
# Only select customers who has more than 5 transcations
CUSTOMER_MIN_TRANSACTIONS = 5

customers_num_purchases = all_joined_ohe.groupby('customer_id').size().reset_index(name='count')
customers_min_purchases = customers_num_purchases[customers_num_purchases['count'] >= CUSTOMER_MIN_TRANSACTIONS]['customer_id']

all_joined_ohe = all_joined_ohe[all_joined_ohe['customer_id'].isin(customers_min_purchases)]
all_joined_ohe['customer_id'].nunique()
```

```
667213
```

3.9 Group the dataset

In order to calculate the difference and preference of each customer, we can sum up the 1 in a column. If a customer often buys a certain type or a certain color of a product, we will know.

```
all_joined_ohe.head()
```

	customer_id	article_id	price	sales_channel_id	age	fashion_news_frequency	product_group_name_Accessories
4	3681748607f3287d2c3a65e00bb5fb153de30e9becf158...	663713001	0.049475	2	30.0	2	0
5	3681748607f3287d2c3a65e00bb5fb153de30e9becf158...	598859003	0.059305	2	30.0	2	0
6	3681748607f3287d2c3a65e00bb5fb153de30e9becf158...	640539001	0.042356	2	30.0	2	0
7	3681748607f3287d2c3a65e00bb5fb153de30e9becf158...	399136009	0.067780	1	30.0	2	0
8	3681748607f3287d2c3a65e00bb5fb153de30e9becf158...	554450001	0.033881	2	30.0	2	0

5 rows × 293 columns

Group some attributes that do not need to sum up.

```
df1 = all_joined_ohe.groupby('customer_id').first()
```

```
df1.head()
```

customer_id	article_id	price	sales_channel_id	age	fashion_news_frequency	product_group_na
00000dbacae5abe5e23885899a1fa44253a17956c6d1c3d25f88aa139fdfc657	656719005	0.044051	2	49.0	0	
0000423b00ade91418cceaf3b26c6af3dd342b51fd051eec9c12fb36984420fa	673677002	0.016932	2	25.0	0	
00007d2de826758b65a93dd24ce629ed66842531df6699338c5570910a014cc2	505221004	0.015237	2	32.0	2	
000097d91384a0c14893c09ed047a963c4fc6a5c021044eec603b323e8c82d1d	624121005	0.012186	1	31.0	0	
00009d946eec3ea54add5ba56d5210ea898def4b46c68570cf0096d962cacc75	610776024	0.007610	2	56.0	2	

5 rows × 292 columns

Group some attributes that need to sum up.

```
df2 = all_joined_ohe.groupby('customer_id').sum()
df2.head()
```

	article_id	price	sales_channel_id	age	fashion_news_frequency	product_grou
customer_id						
00000dbacae5abe5e23885899a1fa44253a17956c6d1c3d25f88aa139fdfc657	5273449066	0.299847	13	441.0	0	
0000423b00ade91418cceaf3b26c6af3dd342b51fd051eec9c12fb36984420fa	33784262244	1.714390	99	1275.0	0	
00007d2de826758b65a93dd24ce629ed66842531df6699338c5570910a014cc2	61518883238	2.137136	195	3168.0	198	
000097d91384a0c14893c09ed047a963c4fc6a5c021044eec603b323e8c82d1d	8135208155	0.222492	15	465.0	0	
00009d946eec3ea54add5ba56d5210ea898def4b46c68570cf0096d962cacc75	49673974179	2.394322	154	4312.0	154	

5 rows × 292 columns

```
df2.drop('article_id',axis=1)
df2.iloc[:,5:]
```

	product_group_name_Accessories	product_group_name_Bags	product_group_name_Cos
customer_id			
00000dbacae5abe5e23885899a1fa44253a17956c6d1c3d25f88aa139fdfc657	0.0	0.0	
0000423b00ade91418cceaf3b26c6af3dd342b51fd051eec9c12fb36984420fa	3.0	0.0	
00007d2de826758b65a93dd24ce629ed66842531df6699338c5570910a014cc2	4.0	0.0	
000097d91384a0c14893c09ed047a963c4fc6a5c021044eec603b323e8c82d1d	0.0	0.0	
00009d946eec3ea54add5ba56d5210ea898def4b46c68570cf0096d962cacc75	1.0	0.0	
...	...	...	
ffff64f7850d4268016db8db3d48bf5433db2a926ba71bcf0b17dc4e15f1f223	0.0	0.0	
ffff8f9ecdce722b5bab97fff68a6d1866492209bfe5242c50d2a10a652fb5ef	0.0	0.0	
ffffbbf78b6eaac697a8a5dfbfd2bfa8113ee5b403e4747568cac33e8c541831	0.0	0.0	

Combine the sum and the non-sum-up data frame.

```
df_final = df1[['sales_channel_id','age','fashion_news_frequency']].merge(df2.iloc[:,5:], on='customer_id')
df_final = df2[['price']].merge(df_final, on='customer_id')

df_final.head()
```

	price	sales_channel_id	age	fashion_news_frequency	product_group_name_Accesso
customer_id					
00000dbacae5abe5e23885899a1fa44253a17956c6d1c3d25f88aa139fdfc657	0.299847	2	49.0	0	
0000423b00ade91418cceaf3b26c6af3dd342b51fd051eec9c12fb36984420fa	1.714390	2	25.0	0	
00007d2de826758b65a93dd24ce629ed66842531df6699338c5570910a014cc2	2.137136	2	32.0	2	
000097d91384a0c14893c09ed047a963c4fc6a5c021044eec603b323e8c82d1d	0.222492	1	31.0	0	
00009d946eec3ea54add5ba56d5210ea898def4b46c68570cf0096d962cacc75	2.394322	2	56.0	2	

5 rows × 291 columns

3.10 Create the target variable based on total consumption.

We are going to create our target variable, the importance of a customer based on his/her next year's total consumption at H&M.

First, we sum the price of all the transactions for customers as total consumption.

```
transactions_2020.head()
```

	t_dat	customer_id	article_id	price	sales_channel_id
16803901	2019-09-20	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	745014003	0.020322	2
16803902	2019-09-20	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	812484002	0.054220	2
16803903	2019-09-20	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	733098013	0.003373	2
16803904	2019-09-20	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	802842001	0.028797	2
16803905	2019-09-20	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	802842001	0.028797	2

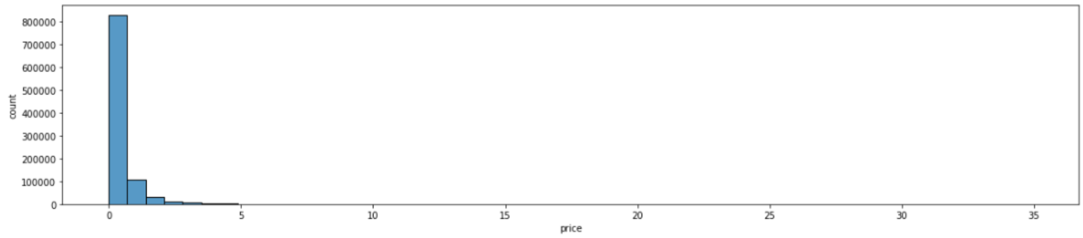
```
transactions_2020 = transactions_2020[['customer_id', 'price']]
transactions_2020.head()
```

	customer_id	price
16803901	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	0.020322
16803902	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	0.054220
16803903	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	0.003373
16803904	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	0.028797
16803905	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	0.028797

```
transactions_2020 = transactions_2020.groupby('customer_id').sum()
```

Then, we could see the total consumption (the sum of price in 2020) distribution.

```
# Total Contribution Distribution
f, ax = plt.subplots(figsize=(20, 4))
ax = sns.histplot(data=transactions_2020, x='price', bins=50)
ax.set_xlabel('price')
ax.set_ylabel('count')
plt.show()
median_total_sales = transactions_2020['price'].median(skipna=True)
print(f"The median of total sales of each customer is {median_total_sales}")
```




```
The median of total sales of each customer is 0.2027288135593218
```

```
transactions_2020.describe()
```

	price
count	993045.000000
mean	0.422690
std	0.677007
min	0.000763
25%	0.083000
50%	0.202729
75%	0.486153
max	34.954610

We decide to divide them into 3 groups by using the 25% and the 75% percentile as our split points. Therefore, the top 25% of customers will be labeled as 2 - the valuable customer. The last 25% will be labeled as 0 – less valuable, and the rest will be 1 – normal customer,

```
transactions_2020.loc[(transactions_2020.price >= 0.486) ,'Y'] = 2
transactions_2020.loc[(transactions_2020.price < 0.083) ,'Y'] = 0

transactions_2020.loc[((transactions_2020.price >= 0.083) & (transactions_2020.price < 0.486)), 'Y'] = 1
```

```
transactions_2020
```

customer_id	price	Y
00000dbacae5abe5e23885899a1fa44253a17956c6d1c3d25f88aa139fdfc657	0.349136	1.0
0000423b00ade91418cceaf3b26c6af3dd342b51fd051eec9c12fb36984420fa	0.887542	2.0
000058a12d5b43e67d225668fa1f8d618c13dc232df0cad8ffe7ad4a1091e318	0.559085	2.0
00006413d8573cd20ed7128e53b7b13819fe5cfc2d801fe7fc0f26dd8d65a85a	0.359593	1.0
000064249685c11552da43ef22a5030f35a147f723d5b02ddd9fd22452b1f5a6	0.101644	1.0

Finally, we merge this column into our previous dataset.

```
df_final = df_final.merge(transactions_2020, on='customer_id')
```

```
df_final.head()
```

customer_id	price_x	sales_channel_id	age	fashion_news_frequency	product_group_name_Accessories	product_group
e5e23885899a1fa44253a17956c6d1c3d25f88aa139fdfc657	0.299847	2	49.0	0	0.0	
1418cceaf3b26c6af3dd342b51fd051eec9c12fb36984420fa	1.714390	2	25.0	0	3.0	
›65a93dd24ce629ed66842531df6699338c5570910a014cc2	2.137136	2	32.0	2	4.0	
54add5ba56d5210ea898def4b46c68570cf0096d962cacc75	2.394322	2	56.0	2	1.0	
›b24feec422ef13df3ccedaedc85368e6664d04ca30b2f8daff	0.460712	1	54.0	2	0.0	

mns

4. Data mining and Evaluation

4.1 Classification Modeling

After the data cleaning and exploration, we have finalized the dataset and firstly we decide to use the classification modeling to find the most and least valuable customers. In this dataset, we have used the data of 2020 to classify our customers into 3 groups in which the value 2 corresponds to the most valuable customers and the value 2 corresponds to the least valuable ones. And we use the data of 2019 to do the classification modeling.

Considering that we have a large number of independent variables and a huge dataset, we decide not to use the support vector machine and KNN algorithms. Instead, we decide to try 3 classification algorithms: naïve bayes, decision tree and random forest, and select the best result one for our client. And before coding the algorithms in Python, we decide to drop the customer_id column out of the independent variables and preprocess both the dependent and independent variables. Then we split the data into 70% train and 30% test sets.

```
df1 = df.drop(["customer_id"], axis = 1)
df1.head()
```

	price_x	sales_channel_id	age	fashion_news_frequency	product_group_name_Accessories	product_group_name_Bags
0	0.299847	2	49.0	0	0.0	0.0
1	1.714390	2	25.0	0	3.0	0.0
2	2.137136	2	32.0	2	4.0	0.0
3	2.394322	2	56.0	2	1.0	0.0
4	0.460712	1	54.0	2	0.0	0.0

5 rows × 292 columns

```
from sklearn.model_selection import train_test_split

x = df1.drop(["Y"], axis = 1)
y = df1["Y"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 100)
```

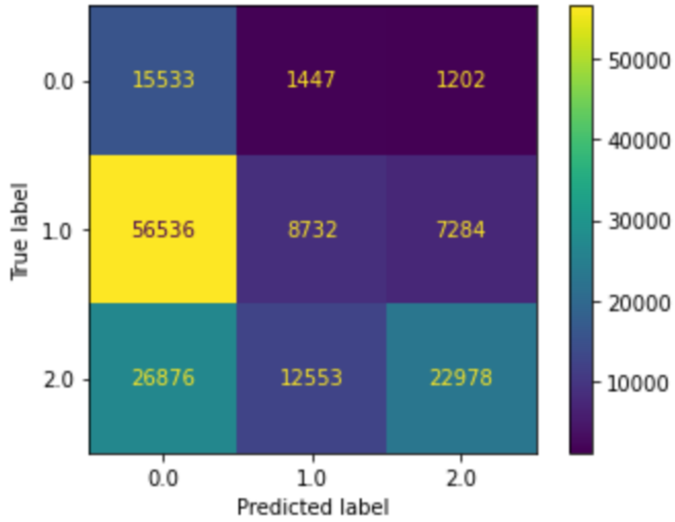
4.1.1 Gaussian Naive Bayes

We first try the Gaussian Naïve Bayes model using scikit-learn package. After fitting the model and generating the prediction, we then calculate its accuracy and generate the confusion matrix for the model.

```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
gnb.fit(x_train, y_train)

y_pred = gnb.predict(x_test)
```

```
from sklearn import metrics
import matplotlib.pyplot as plt

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

cm_gnb = metrics.confusion_matrix(y_test, y_pred, labels = gnb.classes_)
cmd_gnb = metrics.ConfusionMatrixDisplay(confusion_matrix=cm_gnb, display_labels=gnb.classes_)

cmd_gnb.plot()
plt.show()
```

```
Accuracy: 0.30849347986496106
```



As the result showing above, we only get around 31% accuracy for the gaussion naïve bayes models.

4.1.2 Decision Tree

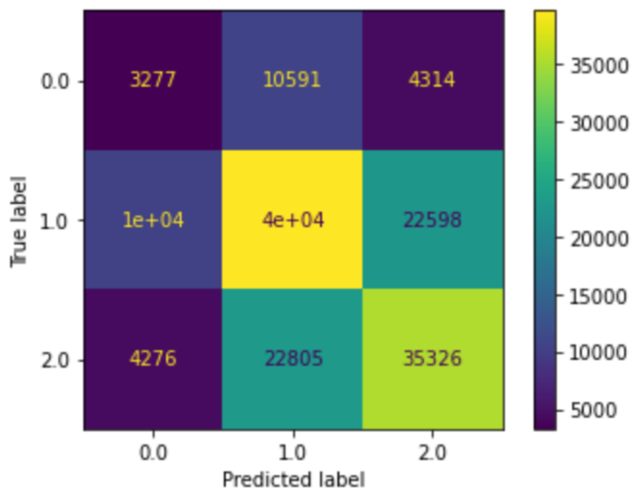
Next, we try the decision tree model. After fitting the model and generating the prediction, we then calculate its accuracy and generate the confusion matrix for the model.

```python
from sklearn.tree import DecisionTreeClassifier

dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

y_pred = dt.predict(x_test)
```

```python
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

cm_dt = metrics.confusion_matrix(y_test, y_pred, labels = dt.classes_)
cmd_dt = metrics.ConfusionMatrixDisplay(confusion_matrix=cm_dt, display_labels=dt.classes_)

cmd_dt.plot()
plt.show()
```

As the result showing above, we get a better accuracy than naïve bayes model which is around 51%.

```
Accuracy: 0.5112282145212582
```



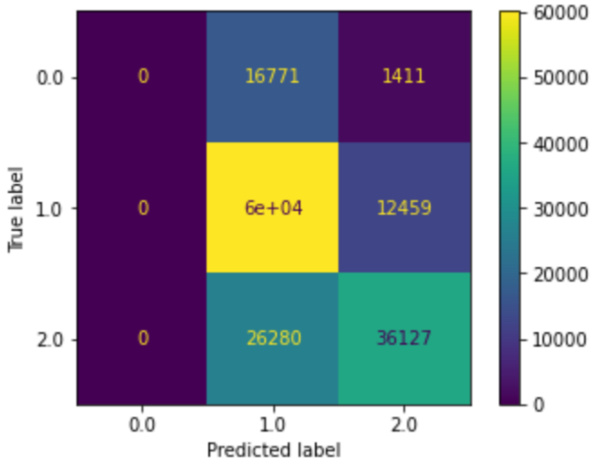
4.1.1 Random Forest

Finally, we try the random forest model. We find the max_depth = 5 yields to the great result. Like always, after fitting the model and generating the prediction, we then calculate its accuracy and generate the confusion matrix for the model.

```
from sklearn.ensemble import RandomForestClassifier

rf2 = RandomForestClassifier(max_depth=5, random_state=0)
rf2.fit(x_train, y_train)

y_pred = rf2.predict(x_test)
```

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

cm_rf2 = metrics.confusion_matrix(y_test, y_pred, labels = rf.classes_)
cmd_rf2 = metrics.ConfusionMatrixDisplay(confusion_matrix=cm_rf2, display_labels=rf.classes_)

cmd_rf2.plot()
plt.show()
```

```
Accuracy: 0.6283098582352211
```



It turns out that the random forest model performs the best among the 3 models. It has around 63% overall accuracy.

Because the main goal of the classification modeling is to help the client find the most valuable customers, compared to the rest 2 groups, the value 2 group is the “important” group that we care about most.

To focus on the value 2 group, we deeply analyze the confusion matrices of the 3 models.

For Gaussian Naïve Bayes:

Sensitivity = 22978 / (26876 + 12553 +22978) = 37%

Specificity = (8732+15533) / (15533+1447+1202+56536+8732+7284) = 27%

For decision tree model:

Sensitivity = 35326 / (4276 + 22805 + 35326) = 57%

Specificity = (39577+3279) / (3279+10592+4311+10359+39577+22616) = 47%

For random forest model:

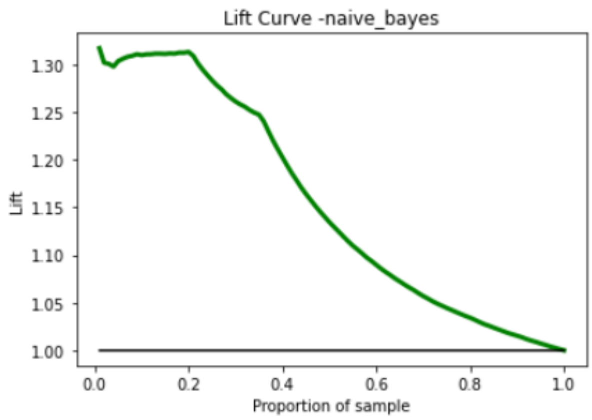
Sensitivity = 36127 / (36127+26280) = 58%
Specificity = 60296 / (16762 + 60296 + 1420 + 12256) = 67%

In this perspective, we find it that decision tree model and the random forest has the similar sensitivity to detect the important group members. But the random forest model is more likely to rule out the other 2 group members.

Another aspect for model evaluation is to check how fast the model could capture the essential information, in our case is to find the most valuable customers. We decide to use the lift chart to compare among the 3 models.

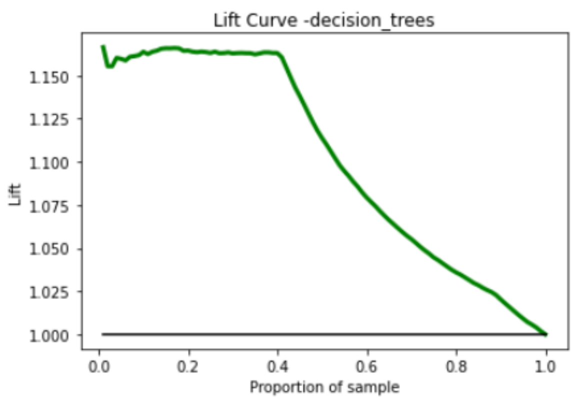
For Gaussian Naïve Bayes:

```
plot_Lift_curve(y_test, y_pred, step=0.01, model_name='naive_bayes')
```



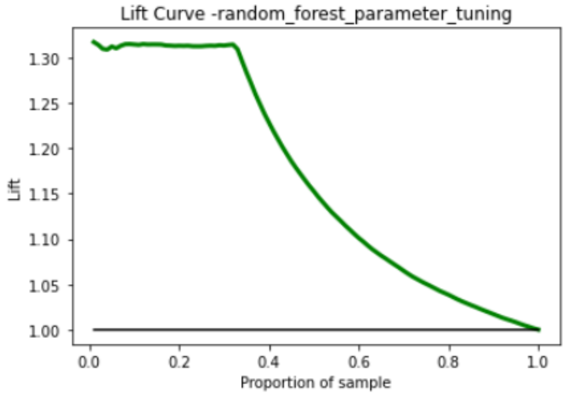
For decision tree model:

```
# Draw the lift curve
plot_Lift_curve(y_test, y_pred, step=0.01, model_name='decision_trees')
```



For random forest model:

```
plot_Lift_curve(y_test, y_pred, step=0.01, model_name='random_forest_parameter_tuning')
```



Based on the results of lift charts, we find it that the random forest model is the one capture the most valuable customers quickest.

To conclude, after the model evaluation, we decide to choose the random forest model to classify the customers to find the most valuable ones.

4.2 Text Mining Technique – TF-IDF

To give recommendations, we have two approaches: first is to recommend similar products to what a customer already bought, based on the assumption that the customer has a certain purchase preference pattern; second is to recommend what similar customers bought to a customer, based on the assumption that similar customers share common purchase preference pattern. TF-IDF here is the first approach. We consider articles' attributes as the "document" representing articles, and get TF-IDF values for all words in article "documents" to calculate similarity between articles. In this way, we successfully recommend similar products to customers.

4.2.1 Data Preparation

Due to large volume of articles and transaction data and limited computer resource, we select customers identified as high-value in 2020 as our customer list and select their transaction data in 2020-05 to make predictions and test the prediction accuracy from June to August.

```
pred_set = transaction_data[(transaction_data['t_dat']>='2020-05-01') & (transaction_data['t_dat']<='2020-05-31')]
test_set = transaction_data[(transaction_data['t_dat']>='2020-06-01') & (transaction_data['t_dat']<='2020-08-31')]
```

```
#print(len(pred_set))  1361815
pred_set = pred_set[pred_set['customer_id'].isin(HV_customer_list)]
print(len(pred_set)) #785634
test_set = test_set[test_set['customer_id'].isin(HV_customer_list)]
print(len(test_set)) #2467377
```

```
785634
2467377
```

As for articles table, there are 105542 articles in total with 25 attributes. It is quite difficult for our computer to calculate similarity between all articles pair. Here we will only select 8000 of them and treat these 8000 articles as the article basket. To create the document for these 8000 articles, we combine its prod_name, product_type_name, product_group_name, graphical_appearance_name, colour_group_name, perceived_colour_value_name, perceived_colour_master_name, department_name, index_name, index_group_name, section_name, garment_group_name and detail_desc as a string. The result is as below.

```
articles_df.head()
```

	article_id	info
0	108775015	Straptop Vesttop GarmentUpperbody Solid Black ...
1	108775044	Straptop Vesttop GarmentUpperbody Solid White ...
2	108775051	Straptop(1) Vesttop GarmentUpperbody Stripe Of...
3	110065001	OPT-shirt(Idro) Bra Underwear Solid Black Dark...
4	110065002	OPT-shirt(Idro) Bra Underwear Solid White Ligh...

4.2.2 Model Training

Then we calculate TF-IDF values for these articles and get the similarities for each article pairs. We find out the five most similar articles for each article and define this as a function to give recommendations. To check if we successfully get the similar articles for each article, we used images in the H&M datasets and check some articles.

We selected three articles in the list and used the algorithm to find its most similar 5 products and the results are as below. We can see that it works pretty well.

Based on the results below, we recommend similar products in this 8000-article basket to what they bought in May and form a prediction list of articles.

4.2.3 Model Accuracy

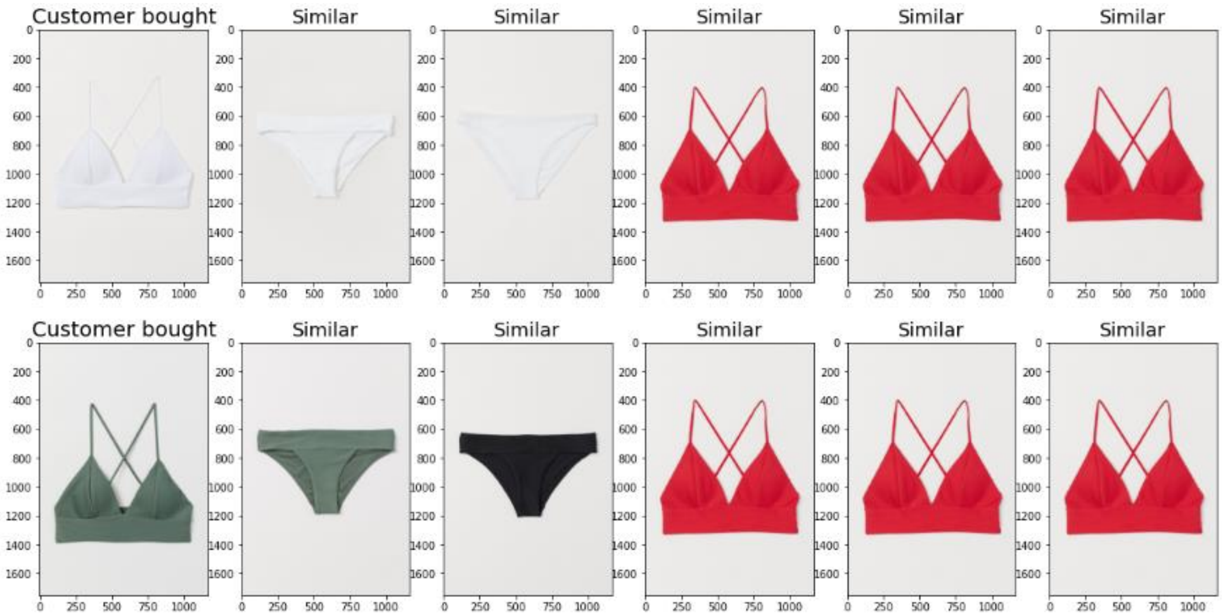
By comparing our predictions to what customers actually bought in the next three months, the accuracy in the articles side: we give 123322 recommendations in total. Among them, 815 are correct, 122507 are incorrect. The accuracy rate is 0.66%.
From the customer side: we give recommendations for 19477 customers. Among them, 719 are correct and 18758 are incorrect. The accuracy rate is 3.69%.

4.3 Association Rules

We applied association rules to the dataset and give recommendations for customers.

4.3.1 Reason for applying association rules

By exploring the data, we found that many customers bought more than 1 articles at a time and we believe that there may be some relations between articles.
The simplest example is the association between a holiday dress and a straw hat. We may find that when a customer buys a holiday dress, he/she is likely to buy a straw hat through the purchase records of a large number of customers in the past. Therefore, it would be a meaningful recommendation to recommend a straw hat to a customer who just bought a white holiday dress recently.
By applying association rules, we can figure out the relation between articles and recommend articles to customers that they are most likely to buy.
Also use images to check the similar articles we find. We can see from the below pictures that the results are pretty good.



4.3.2 Choice of parameters

Customer id and article id are the main parameters to analyze the problem.
One traditional case for the association rule is in 1990s, Walmart found that beer and diapers, two completely insignificant items, have a high probability of being purchased together. Therefore, to analyze the problem, we cannot presuppose possible outcomes from experience, sometimes unexpected results occur.
In this case, we choose customer id to help filter out some low-quality users and only keep the customers who have high potentials to buy articles in HM. Articles ids are grouped by customer id and date parameters to find out what articles are bought together by a customer at a time.

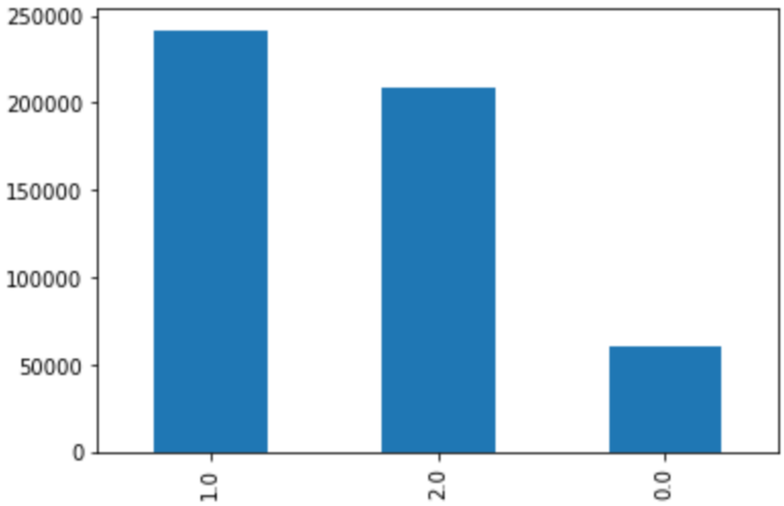
4.3.3 Model Accuracy

Analyze the problem from the total recommended articles side: we give 542155 recommendations in total. Among them, 9669 are correct, 532486 are incorrect. The accuracy rate is 1.8%.
Analyze the problem from the customer side: we give recommendations for 72515 people, among them, 7228 have bought at least one articles from the recommendations we gave them, the accuracy rate is 10.0%.
For a recommender system, we have a considerable prediction accuracy by applying association rule.

5. Analysis of Models:

5.1 Classification Models

Based on the modeling results, we finally decide to use the decision tree model to classify our customers into 3 groups with different purchasing powers. The results of the model indicate that around 60% customers follow the same pattern on spending money for H&M products.



And around 40% customers are classified as the most valuable customers (labeled with value 2). For the group of the most valuable customers, H&M decision makers may consider make the marketing team to push ads of top gear products, send new products emails in the first places and may design customized products or gifts for certain shopping seasons. For those customers, it's important to keep their loyalty and generate more profits from them. On the other hand, for those one-time or new customers (labeled with value 0), they should consider attracting them to place at least a new order through sending promotion codes for instance.

And another thought is to expand our modeling by make models to recommend the "right" products for "right" customers. Therefore, our next step is to use text mining technique and association rules to recommend the customized preferable products to the high value customers.

5.2 Text Mining Technique – TF-IDF

5.2.1 General Description

We use a subset of 8000 articles as the article basket to calculate article pair's similarity. As for transaction data, we select transactions data in May 2020 as the training data and provide similar articles to what they bought in May. Here we only select the high-value customers, those tagged as 2. By sorting the similarity score for each article, top 5 similar products become recommendations to customers. The testing dataset are the same group of people who have transactions in the following 3 months (June, July, August 2020).

5.2.2 Results in Detail

The articles and their "documents" are like below.

	article_id	info
0	108775015	Straptop Vesttop GarmentUpperbody Solid Black ...
1	108775044	Straptop Vesttop GarmentUpperbody Solid White ...
2	108775051	Straptop(1) Vesttop GarmentUpperbody Stripe Of...
3	110065001	OPT-shirt(Idro) Bra Underwear Solid Black Dark...
4	110065002	OPT-shirt(Idro) Bra Underwear Solid White Ligh...

The recommendations we provided for each customer are as below.

```
pred_df2.head()
```

	customer_id	article_id	similar_articles
25275165	00ff2cefd0f11593213fac85099062d6491fec5d840bda...	349301001	[349301045, 349301046, 349301028, 349301029, 3...
25275209	0130fa846141d8d21422187219af10b41cbb0b8daba887...	408875001	[408875030, 408875016, 408875009, 408875028, 4...
25275269	01abe31ce31dd51c04076e2fe562c67ea39d2921355de1...	351484039	[351484026, 351484009, 351484002, 351484013, 3...
25275270	01abe31ce31dd51c04076e2fe562c67ea39d2921355de1...	351484039	[351484026, 351484009, 351484002, 351484013, 3...
25275297	01d4772ffa5aff2629f0c03d2a25647b91188eb2ca99ff...	470789030	[470789031, 470789001, 470789028, 470789019, 4...

Since our recommendation is provided per customer, then we aggregate the result by customer.

	customer_id	predictions
0	00023e3dd8618bc63ccad995a5ac62e21177338d642d66...	[452818021, 452818029, 452818031, 452818034, 4...
1	00080403a669b3b89d1bef1ec73ea466d95e39698d6dde...	[496923008, 451744001, 451744005, 451744006, 4...
2	0008d644deb96bdc0ca262f161cf6d5e9a4e619bb75faa...	[486639009, 486639011, 486639012, 486639013, 4...
3	000989f72a2b8e5da2f4abafc86c2e213816fa2ff2a060...	[488561027, 488561032, 488561036, 488561015, 4...
4	000a74a8ce6e616c1d26d186d8ffae487878b89223b60f...	[408875009, 408875016, 408875028, 408875030, 4...

After that, we compare our predictions with actual articles they bought and calculate the accuracy for each customer.

customer_id	predictions	articles_list	accuracy
00b6efd6ae233e446805ced7ad1f6003afac91e4b42c37...	[253448001, 253448003, 253448056, 253448059, 2...	[689389048, 596740012, 253448059, 253448003, 5...	0.400000
012ae734fb97c1f44025243d4b5a33ebfad7c222a80cf9...	[372860001, 373506018, 373506019, 372860069, 3...	[372860001, 857572001, 464297007, 891655001, 8...	0.066667
013c3aadf29d43aae7c100766aa1f20b2730fb897faa64...	[372860001, 253448001, 253448003, 253448002, 2...	[253448003, 762846007, 856617003, 824764004, 5...	0.090909
028a493371e362fa06b6a446244e46801c9123379a861b...	[372860002, 372860043, 160442007, 160442042, 1...	[821886001, 860632003, 160442043, 841383003, 2...	0.200000
02aae8afcbc7874b1457056ee3d7999776363967a79564...	[372860002, 372860043, 160442007, 160442042, 1...	[806388009, 783346001, 156224001, 349301001, 8...	0.200000

5.2.3 Future Improvements

We can see from above results that the accuracy is not quite satisfying. From my perspective, I think this could due to the following reasons:

- Here we only select a subset of 8000 articles in our basket to calculate article pair similarity. If we select the whole basket, the top 5 similar articles may be different; In addition, these customers did not buy articles in this basket for the next three months, but they may have bought other similar products to what they bought in May and we did not cover those in our basket. If the similarity algorithm is expanded to the entire dataset, the accuracy will be much likely to lift.

- Maybe one-month data is not enough to capture the purchase preference and three months is not enough for test the accuracy. In real life, H&M may use the past several months purchase history for giving recommendations and we need to try different test period to find out the most proper test period.

5.3 Association Rules

5.3.1 General Description

We use transactions data in May 2020 as the training data to figure out the association rules between pairs of articles. By sorting the confidence score by pairs of articles, recommendations are given for the customers. The testing dataset are the people who have transaction history in May 2020 and in either the following 3 months (June, July, August 2020). People who have low intends tag in HM are also filtered out.

5.3.2 Results in Detail

Use data from "transactions_train.csv" where date range from 2020-05-01 to 2020-05-31 as training data for the association rules.
Extract all transactions where people bought at least two items a time as the original table. Descriptive statistics for origin table：

```
count    264011.000000
mean          4.291011
std           3.117811
min           2.000000
25%           2.000000
50%           3.000000
75%           5.000000
max          86.000000
```

Construct a frequency table for pairs of articles bought in May 2020. Calculate the joint frequency and confidence to evaluate the association between articles. The frequency table is as following:

	item1	item2	joint_freq	item_freq1	item_freq2	confidence
3106	0160442007	0160442010	95	366	637	0.259563
3107	0160442007	0160442043	35	366	147	0.095628
3188	0160442010	0160442007	95	637	366	0.149137
3190	0160442010	0160442043	45	637	147	0.070644
3189	0160442010	0372860001	21	637	1195	0.032967

Descriptive statistics for confidence in origin table：

```
mean        0.154674
std         0.164152
min         0.015075
25%         0.061224
50%         0.098991
75%         0.172107
max         0.955224
```

For the prediction part, we first construct a transaction table where date range from 2020-06-01 to 2020-08-31.

	customer_id	article_id
0	0000f1c71aafe5963c3d195cf273f7bfd50bbf17761c91...	[[0864716001, 0889714001, 0841383002, 08327320...
1	0000f2ea26b7f0a9175f428c8cf7743e9e10e193465ecd...	[[0808840004, 0858640004]]
2	0001177027259b455f979d85a278e4b280205d4de5cce4...	[[0863456003, 0570002090, 0863456005], [082046...
3	00012315fd38859ff2c446876ca507abbcbcf582d0e266...	[[0842607004, 0842607002, 0690936001]]
4	00015c1a121e08bbd2552c15fbbb6e6b19d3bf8f7b6a3d...	[[0842605015, 0762846007]]

Then we inner join the transaction table with original table to get the customers who bought article in May 2020 and also in either the following 3 months of 2020. We also use the tags we get when clustering the people to filter out the low potential customers. Thus, we get a new customer list for prediction.
For these customers, we get their articles bought in May and extract all the association articles from the frequency table. In this step, we limit the association articles by setting the joint frequency > 20 (items1 and items2 are bought more than 20 times together by a same customer in May 2020). Finally, we sort the related articles by sorting confidence value from the highest to the lowest. For each customer, we give at most 10 recommendations.
We then get a prediction table as following:

	customer_id	prediction	article_id
0	0000f1c71aafe5963c3d195cf273f7bfd50bbf17761c91...	0749699002 0749699008 0821163008 0716672001 07...	[0917434002, 0685814048, 0895418003, 087546900...
1	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	0825720005 0852746007 0825720003 0825720004 08...	[0842792001, 0859118002, 0886540001, 086942400...
2	00080403a669b3b89d1bef1ec73ea466d95e39698d6dde...	0372860001 0723469001 0704754001 0748355002 07...	[0832505001, 0902992001, 0556260001, 057954107...
3	0008d644deb96bdc0ca262f161cf6d5e9a4e619bb75faa...	0861712001 0857272001 0865533001 0875951002 08...	[0869424001, 0855262002, 0908729002, 0237347060]
4	000934651054f08396856cd83fad3b36b97ab95a0baf79...	0610776001 0610776083 0610776103 0610776072 05...	[0750480004, 0790686006, 0841674002, 076327500...
...	...	...	...
72510	fffae8eb3a282d8c43c77dd2ca0621703b71e90904dfde...	0506098007 0723469001 0253448003 0652924004 08...	[0652924041, 0877599001, 0877599001, 055459806...
72511	fffb68e203e88449a1dc7173e938b1b3e91b0c93ff4e1d...	0751551001 0751551004 0874465005 0874320001 07...	[0723469002, 0717490058, 0834217009, 081642300...
72512	fffb834e3b357155d4f72274f3621f68db9c4bac221851...	0841960003 0841960004	[0706268031, 0783707041, 0783707037, 084808200...
72513	fffe7116f9f68e8ad287fd7b6e33aad4871d7080e77d2d...	0717490059 0717490081 0610776072 0610776002 07...	[0783346020, 0783346024, 0808698001, 087703700...
72514	ffffcd5046a6143d29a04fb8c424ce494a76e5cdf4fab5...	0841699003 0901955001 0776237020 0811835005 08...	[0805370005, 0668012013, 0685816044, 085365401...

72515 rows × 3 columns

For each customer, we give at most 10 predictions and extract what they actually buy in June, July, August 2020 for comparison (article_id in the table). There are 72515 people in total that we give recommendations.
To evaluate the model, we calculate the accuracy of each predictions we give. In total, we give 542155 predictions and 1.8% of them are correct.
We also calculate the accuracy for each customer:

	customer_id	prediction	article_id	accuracy
0	0000f1c71aafe5963c3d195cf273f7bfd50bbf17761c91...	0749699002 0749699008 0821163008 0716672001 07...	[0917434002, 0685814048, 0895418003, 087546900...	0.000000
1	0001d44dbe7f6c4b35200abdb052c77a87596fe1bdcc37...	0825720005 0852746007 0825720003 0825720004 08...	[0842792001, 0859118002, 0886540001, 086942400...	0.000000
2	00080403a669b3b89d1bef1ec73ea466d95e39698d6dde...	0372860001 0723469001 0704754001 0748355002 07...	[0832505001, 0902992001, 0556260001, 057954107...	0.000000
3	0008d644deb96bdc0ca262f161cf6d5e9a4e619bb75faa...	0861712001 0857272001 0865533001 0875951002 08...	[0869424001, 0855262002, 0908729002, 0237347060]	0.000000
4	000934651054f08396856cd83fad3b36b97ab95a0baf79...	0610776001 0610776083 0610776103 0610776072 05...	[0750480004, 0790686006, 0841674002, 076327500...	0.100000
...	...	...	...	...
72510	fffae8eb3a282d8c43c77dd2ca0621703b71e90904dfde...	0506098007 0723469001 0253448003 0652924004 08...	[0652924041, 0877599001, 0877599001, 055459806...	0.111111
72511	fffb68e203e88449a1dc7173e938b1b3e91b0c93ff4e1d...	0751551001 0751551004 0874465005 0874320001 07...	[0723469002, 0717490058, 0834217009, 081642300...	0.100000
72512	fffb834e3b357155d4f72274f3621f68db9c4bac221851...	0841960003 0841960004	[0706268031, 0783707041, 0783707037, 084808200...	0.000000
72513	fffe7116f9f68e8ad287fd7b6e33aad4871d7080e77d2d...	0717490059 0717490081 0610776072 0610776002 07...	[0783346020, 0783346024, 0808698001, 087703700...	0.000000
72514	ffffcd5046a6143d29a04fb8c424ce494a76e5cdf4fab5...	0841699003 0901955001 0776237020 0811835005 08...	[0805370005, 0668012013, 0685816044, 085365401...	0.000000

72515 rows × 4 columns

For all the 72515 customers whom we give recommendations, 7228 (10%) of them bought at least 1 article we recommended to them.

Construct a <u>predict correct table</u> for all the customers whom we have at least 1 correct prediction.

	customer_id	prediction	article_id	accuracy
4	000934651054f08396856cd83fad3b36b97ab95a0baf79...	0610776001 0610776083 0610776103 0610776072 05...	[0750480004, 0790686006, 0841674002, 076327500...	0.100000
8	000c5c714aefd0d5ed1205e2781070167826ffc117ab9e...	0399256037 0399256001 0636323002 0399256023 05...	[0579541072, 0869691001, 0579541001, 090195500...	0.100000
11	000fb6e772c5d0023892065e659963da90b1866035558e...	0572797041 0572797001 0841383003 0841383002 05...	[0865470002, 0832732001, 0900382001, 091435100...	0.100000
28	001ddeb8fb74fec5693116da83b488e05ee9a9e179f3fd...	0706016015 0706016002 0706016019 0706016006 07...	[0706016002, 0706016053, 0842000001, 084579000...	0.100000
30	001f5299820c00df306221ff581abf9d18507c2e35ecb3...	0832361001 0832361003 0832361007 0832362002 08...	[0821336004, 0872453002, 0818031002, 088492000...	0.100000
...	...	...	...	...
72484	ffeb041f188b71de1b7354e8fa0369c14c22a1b4d5f55e...	0749699002 0749699008 0821163008 0716672001 07...	[0870970001, 0870970001, 0887830002, 085816100...	0.100000
72489	ffedd10bbc166ed253113951a1c028389064df97a48198...	0810172002 0814817002 0838900002 0838900003 08...	[0814817001, 0857812003, 0850906001, 086209200...	0.400000
72497	fff2c4204fac63f93aec10ed657958d372efe948de1492...	0470789031 0863515004	[0826211001, 0470789031, 0253448062, 080069100...	0.500000
72510	fffae8eb3a282d8c43c77dd2ca0621703b71e90904dfde...	0506098007 0723469001 0253448003 0652924004 08...	[0652924041, 0877599001, 0877599001, 055459806...	0.111111
72511	fffb68e203e88449a1dc7173e938b1b3e91b0c93ff4e1d...	0751551001 0751551004 0874465005 0874320001 07...	[0723469002, 0717490058, 0834217009, 081642300...	0.100000

7228 rows × 4 columns

Descriptive statistics for accuracy in predict correct table:

```
count    7228.000000
mean        0.187178
std         0.158164
min         0.100000
25%         0.100000
50%         0.100000
75%         0.200000
max         1.000000
```

5.3.3 Business meaning of the result

The business meaning behind the result is if we predict 10 items to each high-potential customers, around 10% of the customers will actually buy at least 1 item from the recommendations, which is a really high prediction accuracy rate. The result can give HM the confidence that reasonable personalized recommendations are useful Therefore, in reality, if we actually push the 10 items' product information to the customers by front page or advertisements, we may probably get a more than 10% accuracy rate. In this case, HM customers can benefit from reducing searching time to get what they want and have a better customer experience. HM can also make front pages meaningful and earn profits from directly giving good recommendations to customer.

6. Recommendations:

For the group of the most valuable customers, H&M decision makers may consider make the marketing team to push ads of top gear products, send new products emails in the first places and may design customized products or gifts for certain shopping seasons. For those customers, it's important to keep their loyalty and generate more profits from them. On the other hand, for those one-time or new customers (labeled with value 0), they should consider attracting them to place at least a new order through sending promotion codes for instance.

As for personalized recommendation, according to our previous results, association rule performs better. In fact, there are several methods for us to provide personalized recommendations and here we only provide two of them. To offer more accurate predictions, we can ensemble these algorithms using majority voting and find out articles that appear frequently in our predictions. By dynamically scrolling every week or month, we get a dynamic personalized recommendation for each customer based on their previous purchases.

After getting accurate predictions, the recommendation presenting methods may include：

1. Pushing the predicted product on the homepage；
2. Using the predicted product as an opening advertisement on the website;
3. Display our recommendations after customers finished a transaction and made the payment.

At the same time, by analyzing the most related products, HM can also use business strategies such as bundling sales to promote customer purchases (For example, if customers purchase certain types of goods at the same time, they can have discounts).

7. Conclusions:

After using the classification to capture the high value customers, we build models to predict what those high value customers will buy in the next following 3 months with two methods: 1.text mining technique – TF-IDF; 2. association rules. Comparing the accuracy result on these two models and we find out association rules has a better accuracy rate. Possible reasons may be TF-IDF mainly compare the similarity of the articles and give predictions based on it, where association rules take people's personal aesthetic taste into consideration, and there may be some unexpected pairs of articles.

However, association rules have a strict limitation on the joint frequencies of pairs articles, which may ignore some low-selling items with great potential (which may occur on new items). Therefore, if we could do more analysis, we would work on combining both TF-IDF and association rules for prediction.