DM_hw2_MS

2024 年 4 月 26 日

0.0.1 数据获取与预处理

数据 Microsoft 资讯推荐:https://mind201910small.blob.core.windows.net/release/MINDlarge_train.zip

[1]:
```
import os
import shutil
import urllib
import pandas as pd
import requests
import matplotlib.pyplot as plt
import seaborn as sns



temp_dir = os.path.join(os.getcwd(), 'data/MINDlarge_train')
os.makedirs(temp_dir, exist_ok=True)
base_url = 'https://mind201910small.blob.core.windows.net/release'
training_large_url = f'{base_url}/MINDlarge_train.zip'
```

利用 pandas 解析数据，查看 news_df，保留三列—id、category 和 subcategory，去除其他列。

[2]:
```
behaviors_path = os.path.join(temp_dir, 'behaviors.tsv')
behaviors_df = pd.read_table(
    behaviors_path,
    header=None,
    names=['impression_id', 'user_id', 'time', 'history', 'impressions'])
news_path = os.path.join(temp_dir, 'news.tsv')
news_df = pd.read_table(news_path,
              header=None,
              names=[
```

```
                'id', 'category', 'subcategory', 'title', 'abstract', 'url',
                'title_entities', 'abstract_entities'
            ])
news_df = news_df.drop(columns=['title', 'abstract', 'url', 'title_entities',␣
 ↪'abstract_entities'])
news_df.head(10)
```

[2]:

	id	category	subcategory
0	N88753	lifestyle	lifestyleroyals
1	N45436	news	newsscienceandtechnology
2	N23144	health	weightloss
3	N86255	health	medical
4	N93187	news	newsworld
5	N75236	health	voices
6	N99744	health	medical
7	N5771	health	cardio
8	N124534	sports	football_nfl
9	N51947	news	newsscienceandtechnology

同上操作，保留 impressions 中被点击的数据，去除其他列。

[3]:
```
def extract_positive_ids(cell):
    parts = cell.split()
    return " ".join([part.split('-')[0] for part in parts if part.
 ↪endswith('-1')])

behaviors_df["impressions"] = behaviors_df["impressions"].
 ↪apply(extract_positive_ids)
behaviors_df = behaviors_df.drop(columns=['impression_id', 'user_id', 'time'])
behaviors_df.head(10)
```

[3]:

	history \
0	N8668 N39081 N65259 N79529 N73408 N43615 N2937…
1	N56056 N8726 N70353 N67998 N83823 N111108 N107…
2	N128643 N87446 N122948 N9375 N82348 N129412 N5…
3	N31043 N39592 N4104 N8223 N114581 N92747 N1207…
4	N65250 N122359 N71723 N53796 N41663 N41484 N11…

```
5  N8668 N29136 N128643 N9740 N9375 N52911 N12090…
6                       N91810 N96438 N104027 N53650
7  N9740 N59820 N18389 N23320 N12322 N9375 N11563…
8  N67770 N65823 N35599 N8753 N126368 N32221 N844…
9  N14678 N71340 N65259 N92085 N31043 N70385 N123…
```

```
                   impressions
0  N94157 N78699 N71090 N31174
1                N25587 N36266
2                       N47925
3                      N114935
4                       N86258
5                       N98178
6                       N94572
7                       N98178
8                      N123077
9                       N28902
```

查看两个表的数据规模以更好的分析数据。

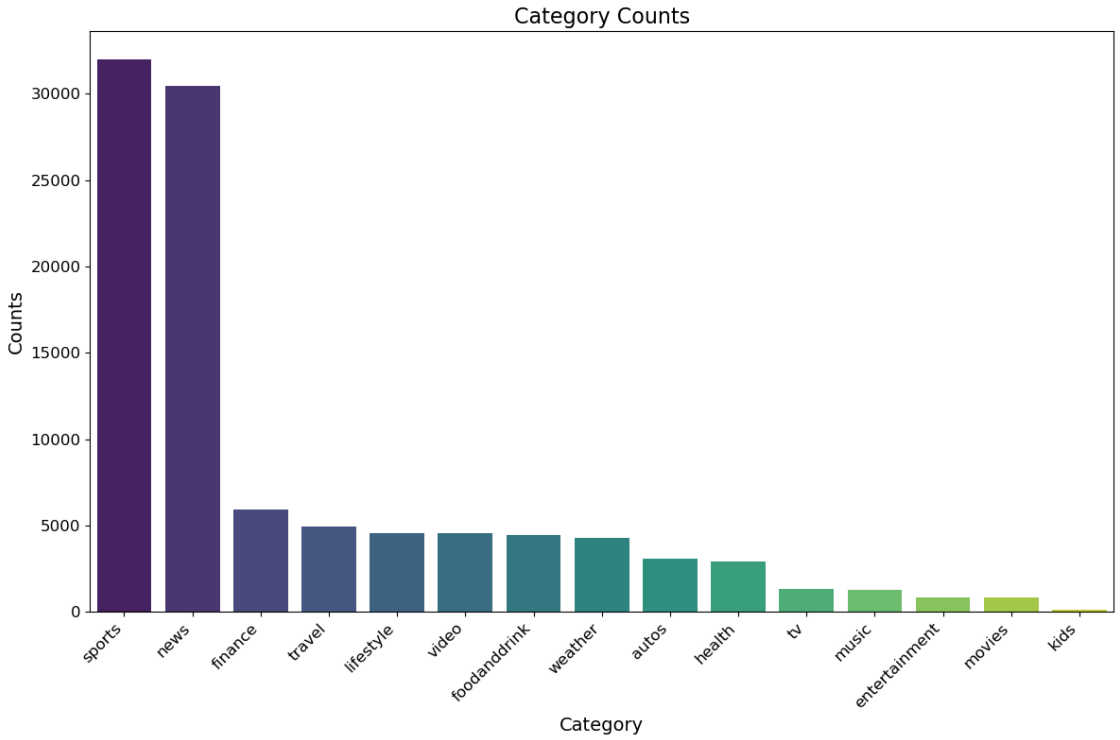
[4]:
```
news_df.dropna(subset=['category', 'subcategory'], inplace=True)
behaviors_df.dropna(subset=['history', 'impressions'], inplace=True)
print('len news_df:', len(news_df))
print('len behaviors_df:', len(behaviors_df))
```

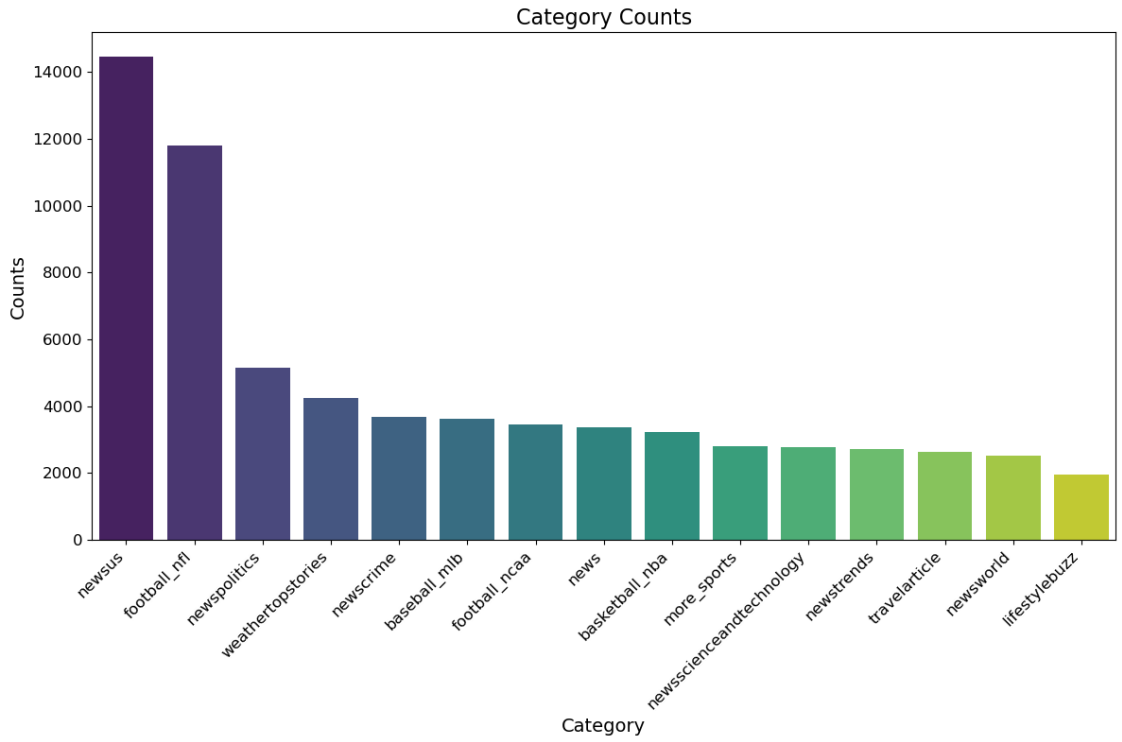
```
len news_df: 101527
len behaviors_df: 2186683
```

查看 news_df 中不同类比/子类别（这里只展示出现次数前 15 的子类）的新闻数量。

[6]:
```
def plot_category_counts(df, column_name, top_n=15):
    plt.figure(figsize=(12, 8))
    sns.barplot(x=df[column_name].value_counts().index[:top_n],␣
 ↪y=df[column_name].value_counts().values[:top_n], palette='viridis')
    plt.title('Category Counts', fontsize=16)
    plt.xlabel('Category', fontsize=14)
    plt.ylabel('Counts', fontsize=14)
    plt.xticks(rotation=45, ha='right', fontsize=12)
```

```
    plt.yticks(fontsize=12)
    plt.tight_layout()
    plt.show()

plot_category_counts(news_df, 'category')
plot_category_counts(news_df, 'subcategory')
```





将 behaviors_df 的 history 和 impressions 字段中的新闻 ID，替换其在 news_df 中对应的子类别。

[7]:
```
id_to_subcategory = dict(zip(news_df['id'], news_df['subcategory']))

def ids_to_subcategories(ids):
    return ' '.join(id_to_subcategory.get(news_id, '') for news_id in ids.
 ↪split())

behaviors_df['history'] = behaviors_df['history'].apply(
    ids_to_subcategories)
behaviors_df['impressions'] = behaviors_df['impressions'].apply(
    ids_to_subcategories)
behaviors_df.head(10)
```

[7]:
```
                                              history  \
0  tv-celebrity newspolitics musicnews nutrition …
1  travelnews football_ncaa_videos news traveltri…
```

```
2  tv-celebrity movies-celebrity animals tv-celeb…
3  newsus lifestylebuzz movies-celebrity entertai…
4  markets movienews movies-celebrity newsworld n…
5  tv-celebrity lifestylebuzz tv-celebrity newscr…
6  music-celebrity lifestylebuzz finance-real-est…
7  newscrime football_nfl foodnews movies-gallery…
8  baseball_mlb newscrime newsus lifestyle celebr…
9  baseball_mlb football_ncaa musicnews health-ne…
```

	impressions
0	movies-celebrity newspolitics newsus musicnews
1	newsus music-celebrity
2	newscrime
3	foodnews
4	animals
5	football_nfl
6	newsus
7	football_nfl
8	musicnews
9	football_ncaa

0.0.2 频繁模式挖掘

由于 FP-growth 算法比 Apriori 算法执行速度快，能更高效地发现频繁项集，因此使用 `fpgrowth` 挖掘频繁项集。

[8]:
```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import fpgrowth



def find_frequent_patterns(behaviors_df):
    transactions = behaviors_df['history'].apply(
        lambda x: x.split()) + behaviors_df['impressions'].apply(lambda x: x.
 ↪split())
```

```
    te = TransactionEncoder()
    te_ary = te.fit(transactions).transform(transactions)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    frequent_itemsets = fpgrowth(df, min_support=0.1, use_colnames=True)
    return frequent_itemsets


frequent_itemsets = find_frequent_patterns(behaviors_df)
print(frequent_itemsets)
```

```
       support                          itemsets
0     0.767892                          (newsus)
1     0.557694                    (tv-celebrity)
2     0.533670                    (newspolitics)
3     0.523931                   (lifestylebuzz)
4     0.412645                (movies-celebrity)
…        …                                 …
2950  0.106111           (tv-celebrity, voices)
2951  0.105330              (newscrime, voices)
2952  0.100047              (newsworld, voices)
2953  0.102809  (newsus, lifestylebuzz, voices)
2954  0.100188      (newsus, newscrime, voices)

[2955 rows x 2 columns]
```

0.0.3 模式命名与分析

最频繁的项目是 “newsus”，支持度为 0.898。其他频繁项集包括 tv-celebritye”、newspoliticss 和、lifestylebuzzv 的支持度也在 0.5 以上，这些都是较为热门的项目。别通过对频繁项集的分析，新闻平台可以更好地了解用户需求，优化内容推荐和分类系统，提高用户体验和内容吸引力，从而增强平台竞争力和持续发展能力。

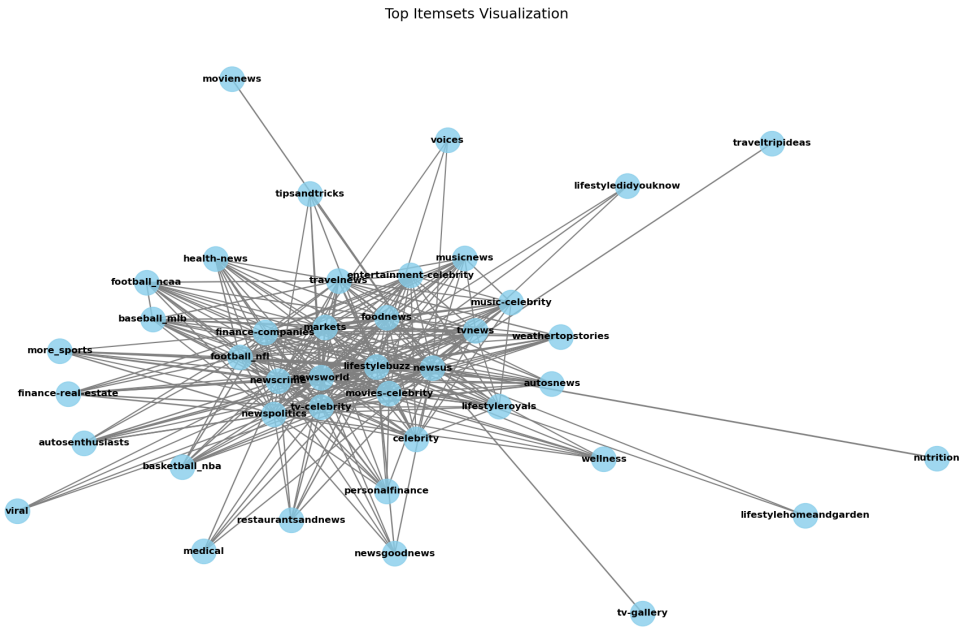
模式命名：新闻类别模式

0.0.4 可视化展示

利用 `networkx` 可视化新闻子类别之间的关系。

[10]:
```
import networkx as nx

def visualize_top_itemsets(frequent_itemsets, top_n=1):
    plt.figure(figsize=(20, 12))
    frequent_itemsets_sorted = frequent_itemsets.sort_values(by='support',␣
 ↪ascending=False)

    selected_items = set()

    # 遍历排序后的项集，直到收集到足够的独特项目
    for items in frequent_itemsets_sorted['itemsets']:
        selected_items.update(items)
        if len(selected_items) >= top_n:
            break

    filtered_itemsets = frequent_itemsets[frequent_itemsets['itemsets'].
 ↪apply(lambda x: any(item in x for item in selected_items))]

    G = nx.Graph()
    for index, row in filtered_itemsets.iterrows():
        items = list(row['itemsets'])
        support = row['support']

        if len(items) > 1:
            for i in range(len(items)):
                for j in range(i + 1, len(items)):
                    G.add_edge(items[i], items[j], weight=support)
        else:
            G.add_node(items[0])

    pos = nx.spring_layout(G, k=0.5, iterations=50)

    nx.draw_networkx_nodes(G, pos, node_size=1000, node_color='skyblue',␣
 ↪alpha=0.8)

    weights = [G[u][v]['weight']*15 for u, v in G.edges()]
```

```
    nx.draw_networkx_edges(G, pos, width=weights, edge_color='gray')
    nx.draw_networkx_labels(G, pos, font_size=12, font_color='black',␣
 ↪font_weight='bold')

    plt.title('Top Itemsets Visualization', fontsize=18)
    plt.axis('off')
    plt.tight_layout()
    plt.show()

visualize_top_itemsets(frequent_itemsets)
```



DM_hw2_SNAP

2024 年 4 月 26 日

0.0.1 数据获取与预处理

SNAP(Stanford Large Network Dataset Collection): https://snap.stanford.edu/data/amazon-meta.html

下载数据，将 txt 文件转为 csv 文件

[42]:
```
import csv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

with open('amazon-meta.txt') as f:
    data = f.readlines()
data = [x.strip() for x in data]

target_file = open('amazon_preprocessed.txt', 'w', encoding = 'utf-8')
all_row = ['Id', 'title', 'group', 'salesrank', 'categories', 'totalreviews',␣
 ↪'avgrating']

for line in data:
    lines = line.split(':')
    if lines[0] == 'Id':
        if len(all_row) == 7:
            for comp in all_row[:6]:
                target_file.write(comp)
                target_file.write(',')
```

```
            target_file.write(all_row[6])
            target_file.write('\n')
            all_row = []
            all_row.append(lines[1].strip())
    if lines[0] == 'title':
        title = ':'.join(lines[1:]).strip().replace(',', ' ').replace('\n', '␣
↪').strip()
        all_row.append(title)
    if lines[0] == 'group' or lines[0] == 'salesrank' or lines[0] ==␣
↪'categories':
        all_row.append(lines[1].strip())
    elif lines[0] == 'reviews' and lines[1].strip() == 'total':
        all_row.append(lines[2].split(' ')[1])
        all_row.append(lines[4].strip())

target_file.close()

meta = pd.read_csv('amazon_preprocessed.txt', sep = ',')
meta['Id'].iloc[0] = 1
meta.to_csv('amazon_meta.csv', index = False)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_9508\1077412725.py:37:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  meta['Id'].iloc[0] = 1
```

读取数据，打印数据前十列，观察数据形式，便于后续分析

[43]:
```
df = pd.read_csv('amazon_meta.csv')
print(df.head(10))
```

```
   Id                                          title  group  salesrank  \
0   1           Patterns of Preaching: A Sermon Sampler   Book     396585
1   2                     Candlemas: Feast of Flames   Book     168596
```

```
2   3   World War II Allied Fighter Planes Trading Cards   Book    1270652
3   4  Life Application Bible Commentary: 1 and 2 Tim…   Book     631289
4   5   Prayers That Avail Much for Business: Executive   Book     455160
5   6  How the Other Half Lives: Studies Among the Te…   Book     188784
6   7                                             Batik  Music       5392
7   8                               Losing Matt Shepard   Book     277409
8   9  Making Bread: The Taste of Traditional Home-Ba…   Book     949166
9  10                           The Edward Said Reader   Book     220379
```

```
   categories  totalreviews  avgrating
0           2             2        5.0
1           2            12        4.5
2           1             1        5.0
3           5             1        4.0
4           2             0        0.0
5           5            17        4.0
6           3             3        4.5
7           4            15        4.5
8           1             0        0.0
9           3             6        4.0
```

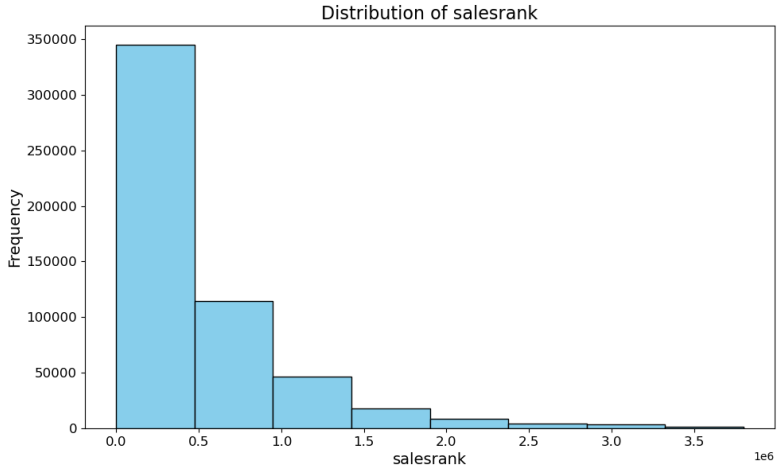
检查缺失值并删除重复值, 五值分析

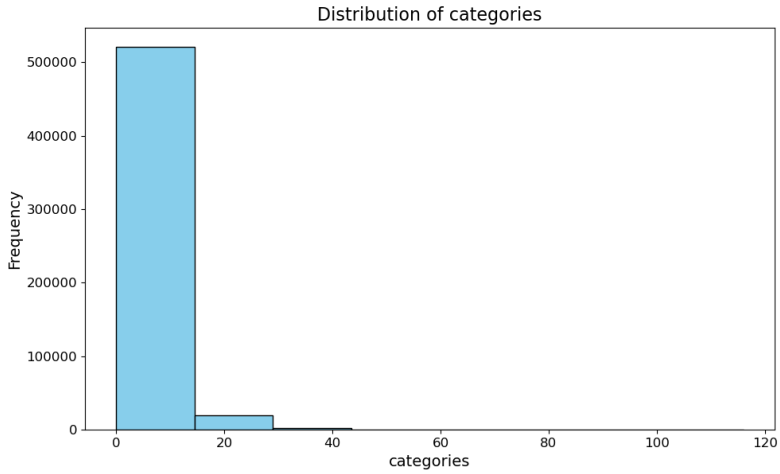
[44]:
```
print(df.isnull().any())

df_nodup = df.drop_duplicates()

numeric = ['salesrank', 'categories', 'totalreviews', 'avgrating']
print(df[numeric].describe().loc[['max', '75%', '50%', '25%', 'min']])
```

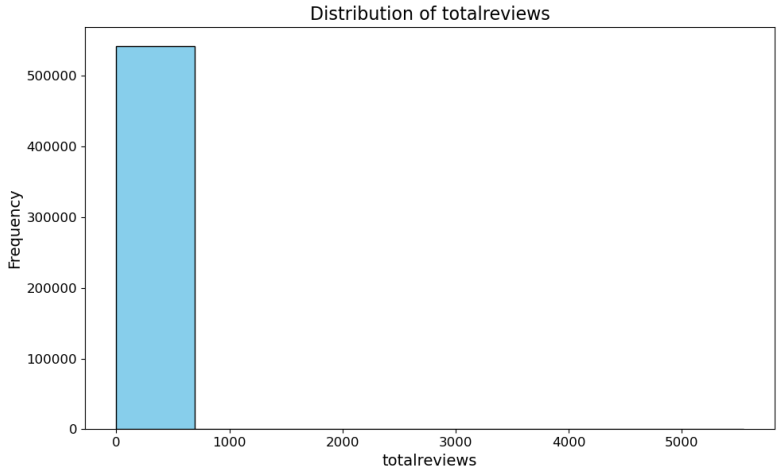
```
Id              False
title           False
group           False
salesrank       False
categories      False
totalreviews    False
avgrating       False
dtype: bool
```

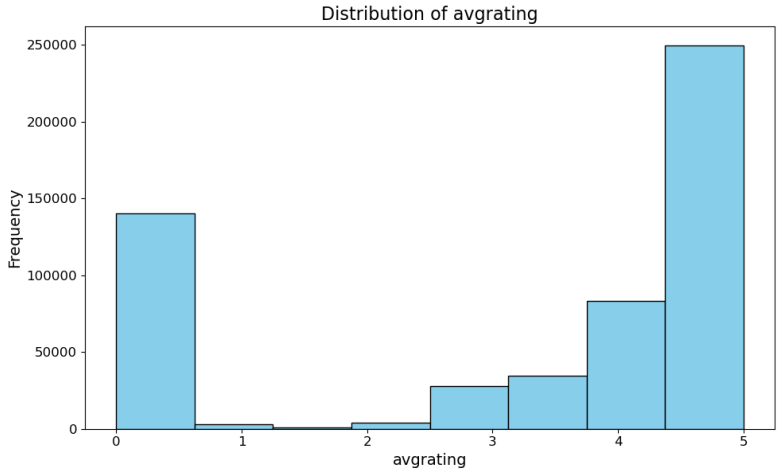
	salesrank	categories	totalreviews	avgrating
max	3798351.0	116.0	5545.0	5.0
75%	672069.5	6.0	8.0	5.0
50%	300493.0	4.0	2.0	4.0
25%	90744.0	2.0	0.0	0.0
min	-1.0	0.0	0.0	0.0

[45]:
```
def hist_plot(data, features):
    num_plots = len(features)
    fig, axes = plt.subplots(nrows=num_plots, ncols=1, figsize=(10,␣
 ↪6*num_plots))

    for i, col in enumerate(features):
        ax = axes[i]
        ax.hist(data[col], bins=8, histtype='bar', color='skyblue',␣
 ↪edgecolor='black')  # 添加黑色边框
        ax.set_title(f'Distribution of {col}', fontsize=16)
        ax.set_xlabel(col, fontsize=14)
        ax.set_ylabel('Frequency', fontsize=14)
        ax.tick_params(axis='both', labelsize=12)

    plt.tight_layout()
    plt.show()

hist_plot(df, numeric)
```

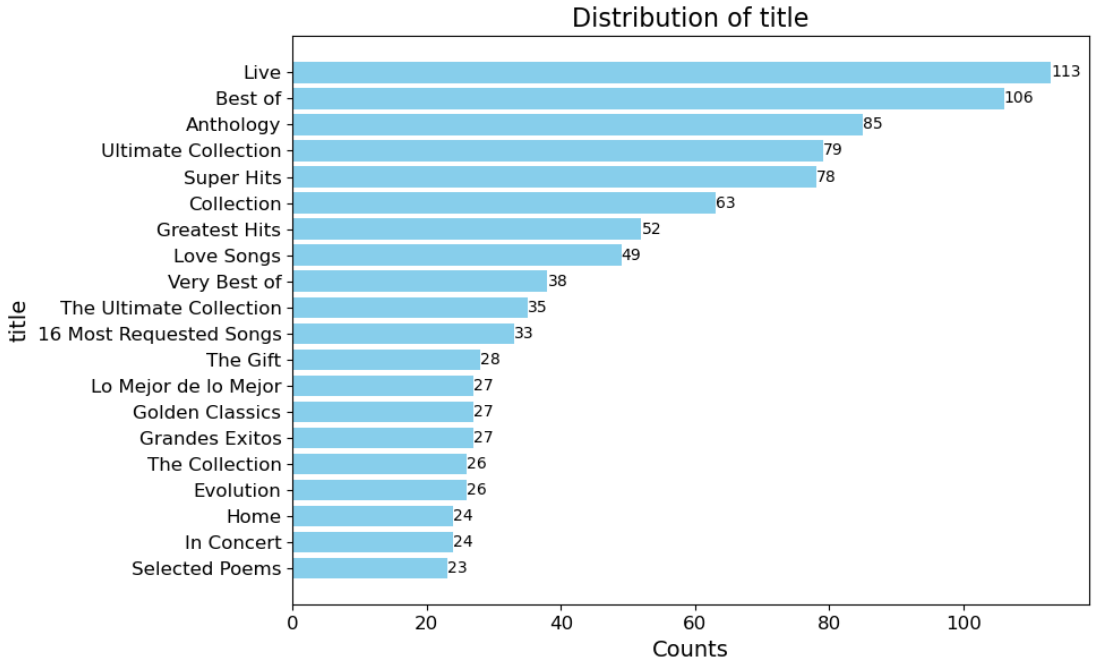

Distribution of salesrank
Frequency
salesrank
1e6

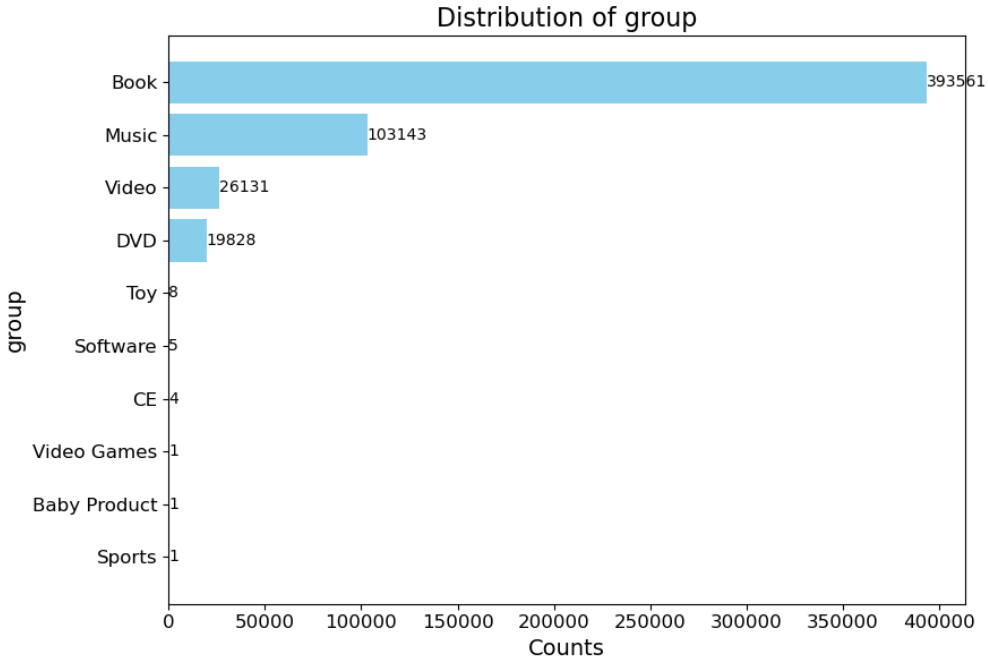

Distribution of categories
Frequency
categories


Distribution of totalreviews
Frequency
totalreviews


Distribution of avgrating
Frequency
avgrating

[46]:
```
def bar_plot(data, features):
    num_plots = len(features)
    fig, axes = plt.subplots(nrows=num_plots, ncols=1, figsize=(10,␣
 ↪6*num_plots))

    for i, col in enumerate(features):
        ax = axes[i]
        counts = data[col].value_counts().head(20)
        bars = ax.barh(counts.index, counts.values, color='skyblue')
        ax.set_title(f'Distribution of {col}', fontsize=16)
        ax.set_xlabel('Counts', fontsize=14)
        ax.set_ylabel(col, fontsize=14)
        ax.tick_params(axis='both', labelsize=12)
        ax.invert_yaxis()  # 反转 y 轴，让最高的条形显示在顶部

        # 在每个柱形上标注数字
        for bar, count in zip(bars, counts.values):
            ax.text(bar.get_width(), bar.get_y() + bar.get_height() / 2, count,
                    va='center', ha='left', fontsize=10, color='black')

    plt.tight_layout()
    plt.show()

bar_plot(df, ['title', 'group'])
```





将数值型属性根据数值划分范围，能更方便的挖掘关联规则

[47]:
```
def trans_avg(row):
    if 0 <= row['avgrating'] < 1:
        new = '[0,1)'
```

```
    elif 1 <= row['avgrating'] < 2:
        new = '[1,2)'
    elif 2 <= row['avgrating'] < 3:
        new = '[2,3)'
    elif 3 <= row['avgrating'] < 4:
        new = '[3,4)'
    elif 4 <= row['avgrating']:
        new = '[4,5]'
    return new

def trans_reviews(row):
    if 0 <= row['totalreviews'] < 5:
        new = '[0,5)'
    elif 5 <= row['totalreviews'] < 10:
        new = '[5,10)'
    elif 10 <= row['totalreviews'] < 1000:
        new = '[10,1000)'
    elif 1000 <= row['totalreviews'] < 3000:
        new = '[1000,3000)'
    elif 3000 <= row['totalreviews']:
        new = '[3000,-)'
    return new

def trans_categories(row):
    if 0 <= row['categories'] < 2:
        new = '[0,2)'
    elif 2 <= row['categories'] < 5:
        new = '[2,5)'
    elif 5 <= row['categories'] < 10:
        new = '[5,10)'
    elif 10 <= row['categories'] < 50:
        new = '[10,50)'
    elif 50 <= row['categories']:
        new = '[50,-)'
    return new
```

```
def trans_salesrank(row):
    if row['salesrank'] < 500000:
        new = '[-,500000)'
    elif 500000 <= row['salesrank'] < 1000000:
        new = '[500000,1000000)'
    elif 1000000 <= row['salesrank'] < 1500000:
        new = '[1000000,1500000)'
    elif 1500000 <= row['salesrank'] < 2000000:
        new = '[1500000,2000000)'
    elif 2000000 <= row['salesrank']:
        new = '[2000000,-)'
    return new
```

[48]:
```
new_rating = []
new_reviews = []
new_rank = []
new_cat = []
for i, row in df.iterrows():
    new_rating.append(trans_avg(row))
    new_reviews.append(trans_reviews(row))
    new_rank.append(trans_salesrank(row))
    new_cat.append(trans_categories(row))
df['avgrating'] = new_rating
df['totalreviews'] = new_reviews
df['categories'] = new_cat
df['salesrank'] = new_rank
df.head(5)
```

[48]:

	Id	title	group	\
0	1	Patterns of Preaching: A Sermon Sampler	Book	
1	2	Candlemas: Feast of Flames	Book	
2	3	World War II Allied Fighter Planes Trading Cards	Book	
3	4	Life Application Bible Commentary: 1 and 2 Tim…	Book	
4	5	Prayers That Avail Much for Business: Executive	Book	

	salesrank	categories	totalreviews	avgrating
0	[-,500000)	[2,5)	[0,5)	[4,5]

1	[-,500000)	[2,5)	[10,1000)	[4,5]
2	[1000000,1500000)	[0,2)	[0,5)	[4,5]
3	[500000,1000000)	[5,10)	[0,5)	[4,5]
4	[-,500000)	[2,5)	[0,5)	[0,1)

将 group、salesrank、categories、totalreviews 和 avgrating 提取出来，转化为 list

[49]:
```
data = df[['avgrating', 'totalreviews', 'categories', 'salesrank', 'group']]
print(data.head(5))
arr = np.array(data)
d = arr.tolist()
```

	avgrating	totalreviews	categories	salesrank	group
0	[4,5]	[0,5)	[2,5)	[-,500000)	Book
1	[4,5]	[10,1000)	[2,5)	[-,500000)	Book
2	[4,5]	[0,5)	[0,2)	[1000000,1500000)	Book
3	[4,5]	[0,5)	[5,10)	[500000,1000000)	Book
4	[0,1)	[0,5)	[2,5)	[-,500000)	Book

0.0.2 频繁模式挖掘

利用 apyori 挖掘数据集中的频繁项集和关联规则，设置最小支持度为 0.1

[53]:
```
from apyori import apriori
result = list(apriori(transactions = d, min_support = 0.1, min_confidence = 0.
↪5))

def show_result(result):
    lastitems = []
    for i in result:
        tempitem = ':'.join(i.items)
        sup = round(i.support, 4)
        for j in i.ordered_statistics:
            col = []
            tempitembase = ':'.join(j.items_base)
            tempitemadd = ':'.join(j.items_add)
            col.append(tempitem)
```

```
            col.append(sup)
            col.append(tempitembase)
            col.append(tempitemadd)
            col.append(round(j.confidence, 4))
            col.append(round(j.lift, 2))
            lastitems.append(col)
    lastitems = pd.DataFrame(lastitems)
    lastitems.columns = ['LastItem', 'Support', 'ItemBase', 'ItemAdd',␣
↪'Confidence', 'Lift']
    lastitems.index = range(len(lastitems))
    print(len(lastitems))
    lastitems = lastitems.sort_values('Confidence', ascending = False).
↪reset_index()
    print(lastitems.head(10))

    return lastitems

lastitems = show_result(result)
```

```
128
   index                      LastItem  Support                ItemBase  \
0     54             [0,1):[0,5):Book   0.2125              [0,1):Book
1    101           [0,1):[2,5):[0,5)   0.1197            [0,1):[2,5)
2     28                 [0,1):[0,5)   0.2579                 [0,1)
3     86       [0,1):[0,5):[-,500000)   0.1005       [0,1):[-,500000)
4    119       [0,1):[0,5):[2,5):Book   0.1037       [0,1):[2,5):Book
5     85       Music:[4,5]:[-,500000)   0.1375            Music:[4,5]
6     79  [500000,1000000):[4,5]:Book   0.1086  [500000,1000000):[4,5]
7     74  [500000,1000000):[2,5):Book   0.1131  [500000,1000000):[2,5)
8     17        [500000,1000000):Book   0.2020       [500000,1000000)
9     18             Music:[-,500000)   0.1845                 Music

      ItemAdd  Confidence  Lift
0       [0,5)      1.0000  1.55
1       [0,5)      1.0000  1.55
2       [0,5)      1.0000  1.55
3       [0,5)      1.0000  1.55
```

```
4        [0,5)      1.0000  1.55
5   [-,500000)      0.9890  1.52
6         Book      0.9861  1.36
7         Book      0.9805  1.35
8         Book      0.9734  1.34
9   [-,500000)      0.9709  1.49
```

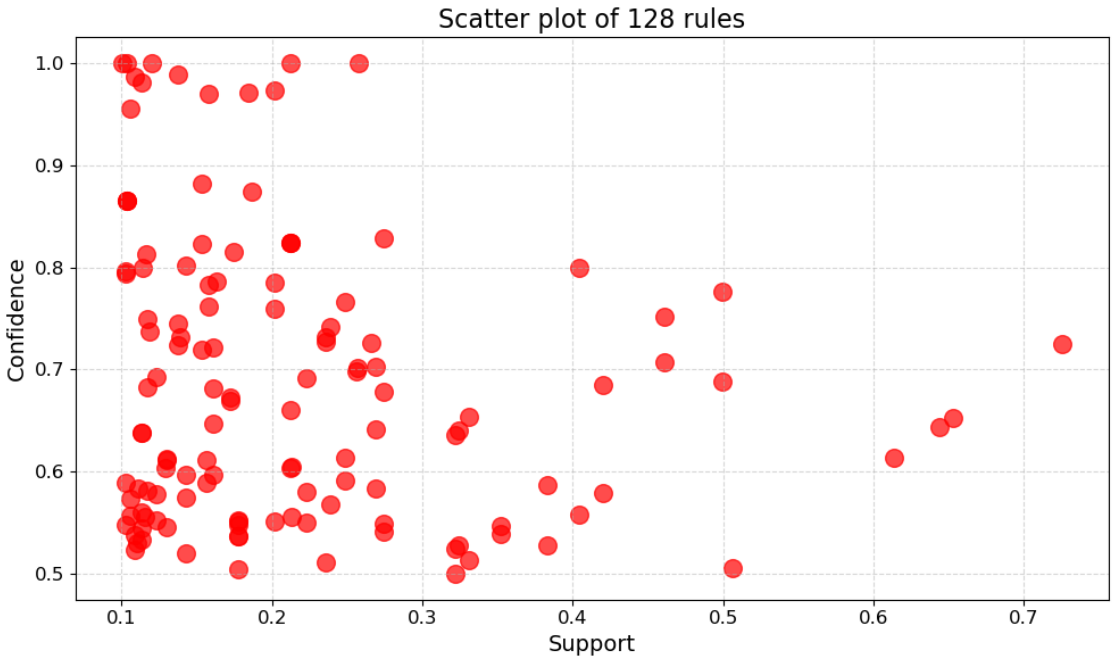
0.0.3 模式命名与分析

分析：以 index 为 18 的数据为例，存在 “Music->[-,500000)”，即 “音乐类别的产品-> 销售排名高于 500000”，可以得出，音乐类别的产品的销售排名约有 97% 的概率高于 500000，且这种情况的发生比例约为 18.35%，其余数据同理可进行解释。

模式命名：音乐产品销售模式

0.0.4 可视化

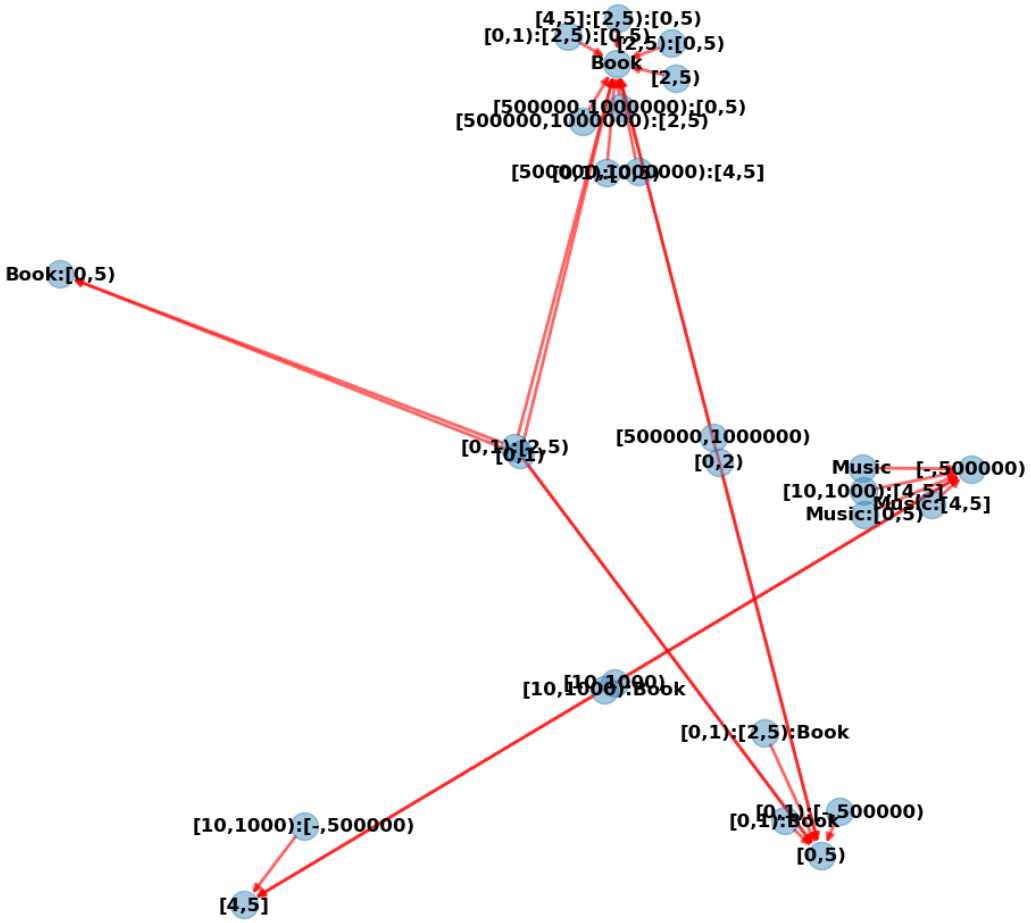
[59]:
```python
def scatter(items, size):
    plt.figure(figsize=(10, 6))
    plt.scatter(items['Support'], items['Confidence'], c='r', s=size, alpha=0.
 ↪7)  # 设置透明度为 0.7
    plt.grid(True, linestyle='--', alpha=0.5)  # 设置虚线网格，透明度为 0.5
    plt.xlabel('Support', fontsize=14)
    plt.ylabel('Confidence', fontsize=14)
    plt.title(f'Scatter plot of {size} rules', fontsize=16)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.tight_layout()  # 自动调整子图参数，使之填充整个图像区域
    plt.show()

scatter(lastitems, 128)
```



[61]:
```
import networkx as nx

def network(items, size):
    plt.figure(figsize=(10, 10))
    # 生成社交网络图
    G = nx.DiGraph()

    draw_df = items.head(size)

    # 为图像添加边
    for idx, row in draw_df.iterrows():
        G.add_edge(row['ItemBase'], row['ItemAdd'], weight=row['Confidence'])

    # 定义不同权重的边
    elarge = [(u, v) for (u, v, d) in G.edges(data=True) if d['weight'] > 0.6]
    emidle = [(u, v) for (u, v, d) in G.edges(data=True) if 0.45 <= d['weight']␣
 ↪<= 0.6]
    esmall = [(u, v) for (u, v, d) in G.edges(data=True) if d['weight'] < 0.45]
```

```
    # 图的布局方式
    pos = nx.spring_layout(G)

    # 根据规则的置信度节点的大小
    nx.draw_networkx_nodes(G, pos, node_size=300, alpha=0.4)

    # 设置边的形式
    nx.draw_networkx_edges(G, pos, edgelist=elarge, width=2, alpha=0.6,␣
↪edge_color='r', style='solid')
    nx.draw_networkx_edges(G, pos, edgelist=emidle, width=2, alpha=0.6,␣
↪edge_color='g', style='dashdot')
    nx.draw_networkx_edges(G, pos, edgelist=esmall, width=2, alpha=0.6,␣
↪edge_color='b', style='dashed')

    # 为节点添加标签
    nx.draw_networkx_labels(G, pos, font_size=12, font_weight='bold')

    plt.axis('off')
    plt.title(f'Sample network of {size} rules', fontsize=16)
    plt.tight_layout()  # 自动调整子图参数，使之填充整个图像区域
    plt.show()

network(lastitems, 30)
```

Sample network of 30 rules



[ ]: