数据挖掘导论大作业报告

选题

本次课程综合实验的选题为“淘宝母婴购物数据可视化分析”，本题为天池大数据竞赛学习赛的一道选题（[淘宝母婴购物数据可视化分析_学习赛_天池大赛-阿里云天池 (aliyun.com)](#)）。

背景

母婴用品是淘宝的热门购物类目，随着国家鼓励二胎、三胎政策的推进，会进一步促进了母婴类目商品的销量。与此年轻一代父母的育儿观念也发生了较大的变化，因此中国母婴电商市场发展形态也越来越多样化。随之引起各大母婴品牌更加激烈的争夺，越来越多的母婴品牌管窥到行业潜在的商机，纷纷加入母婴电商，行业竞争越来越激烈。本项目会基于“淘宝母婴购物”数据集进行可视化分析，帮助开发者更好地做出数据洞察。

环境

本次实验同样使用了前几次实验中新构建的环境，python版本为3.8，主要安装的packages为numpy、pandas、scikit-learn、matplotlib.pyplot、scipy及其依赖项。.ipynb文件使用jupyter notebook进行编写和运行。

数据集解释

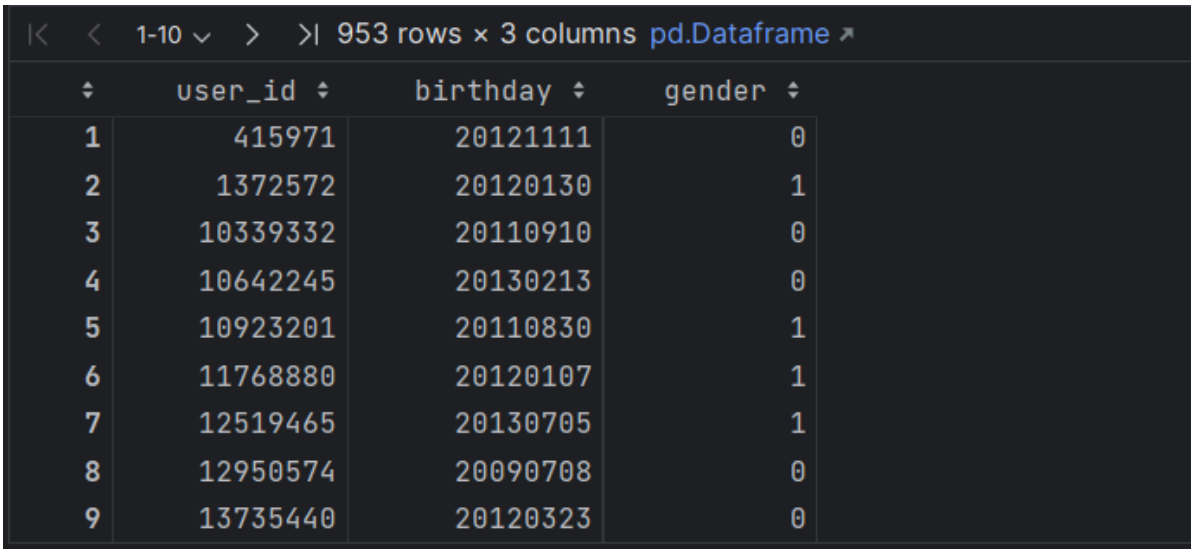
本次学习赛提供了两个数据集，分别为“tianchi_mum_baby_trade_history.csv” 和“tianchi_mum_baby.csv“，两个数据集分别包含不同信息：

数据集	内容
'tianchi_mum_baby_trade_history.csv'	购物行为表
'tianchi_mum_baby.csv'	母婴信息表

每个数据集中的数据及特征值分别如下：

母婴信息表（'tianchi_mum_baby.csv'）：

字段	字段说明
user_id	用户id
birthday	出生年月日
gender	性别(0-男孩 1-女孩 2-不明)



	user_id	birthday	gender
1	415971	20121111	0
2	1372572	20120130	1
3	10339332	20110910	0
4	10642245	20130213	0
5	10923201	20110830	1
6	11768880	20120107	1
7	12519465	20130705	1
8	12950574	20090708	0
9	13735440	20120323	0

购物行为表（'tianchi_mum_baby.csv'）：

字段	字段说明
user_id	用户id
auction_id	交易id
category_1	商品一级目录
category_2	商品二级目录
buy_mount	购买数量
day	交易时间（年月日）

1-10 29971 rows × 6 columns pd.Dataframe

	user_id	auction_id	category_2	category_1	buy_mount	day
0	786295544	41098319944	50014866	50022520	2	20140919
1	532110457	17916191097	50011993	28	1	20131011
2	249013725	21896936223	50012461	50014815	1	20131011
3	917056007	12515996043	50018831	50014815	2	20141023
4	444069173	20487688075	50013636	50008168	1	20141103
5	152298847	41840167463	121394024	50008168	1	20141103
6	513441334	19909384116	50010557	50008168	1	20121212
7	297411659	13540124907	50010542	50008168	1	20121212
8	82830661	19948600790	50013874	28	1	20121101

实验步骤

数据预处理

首先读入两个数据集

```
data0 = pd.read_csv('tianchi_mum_baby_trade_history.csv')
data1 = pd.read_csv('tianchi_mum_baby.csv')
```

data0中存在的项为['user_id', 'auction_id', 'category_2', 'category_1', 'buy_mount','day']，其中'user_id', 'auction_id', 'category_2', 'category_1'都属于id类型，不需要进行预处理，对 'buy_mount','day'进行预处理。

首先查看各变量类型：

```
In  [2]: data0.dtypes
Out[2]: user_id       int64
        auction_id    int64
        category_2    int64
        category_1    int64
        buy_mount     int64
        day           int64
        dtype: object
```

我们发现所有变量都识别为int64类型，但是为了方便我们之后的处理看，最好将day变为datetime类型：

In [3]:
```
data0['day'] = pd.to_datetime(data0.day.astype('str'))
data0
```

Out[3]:

	user_id	auction_id	category_2	category_1	buy_mount	day
0	786295544	41098319944	50014866	50022520	2	2014-09-19
1	532110457	17916191097	50011993	28	1	2013-10-11
2	249013725	21896936223	50012461	50014815	1	2013-10-11
3	917056007	12515996043	50018831	50014815	2	2014-10-23
4	444069173	20487688075	50013636	50008168	1	2014-11-03
...	...	...	...	...	...	...
29966	57747284	35169635909	50010549	50008168	1	2014-01-09
29967	287541325	19778523000	50007011	50008168	2	2014-01-09
29968	82915321	12766532512	50011993	28	1	2013-10-08
29969	78259523	18309305134	50013711	50008168	1	2013-10-08
29970	758305789	20177445814	50018860	28	1	2013-10-08

29971 rows × 6 columns

之后我们检查day的范围是否正常，是否存在空缺或明显异常数据：

In [5]:
```
data0.day.describe(datetime_is_numeric=True)
```

Out[5]:
```
count                            29971
mean     2014-01-16 17:25:38.086817536
min                2012-07-02 00:00:00
25%                2013-06-20 00:00:00
50%                2014-03-06 00:00:00
75%                2014-09-09 00:00:00
max                2015-02-05 00:00:00
Name: day, dtype: object
```

day分布在2012-07-02到2015-02-05之间，符合数据统计范围，之后可以继续将day分为year、month、quarter等项来方便之后的作图分析和聚类分类操作：

In [4]:
```
data0['year'] = data0.day.dt.year
data0['quarter'] = data0.day.dt.quarter
data0['month'] = data0.day.dt.month
data0.head()
```

Out[4]:

	user_id	auction_id	category_2	category_1	buy_mount	day	year	quarter	month
0	786295544	41098319944	50014866	50022520	2	2014-09-19	2014	3	9
1	532110457	17916191097	50011993	28	1	2013-10-11	2013	4	10
2	249013725	21896936223	50012461	50014815	1	2013-10-11	2013	4	10
3	917056007	12515996043	50018831	50014815	2	2014-10-23	2014	4	10
4	444069173	20487688075	50013636	50008168	1	2014-11-03	2014	4	11

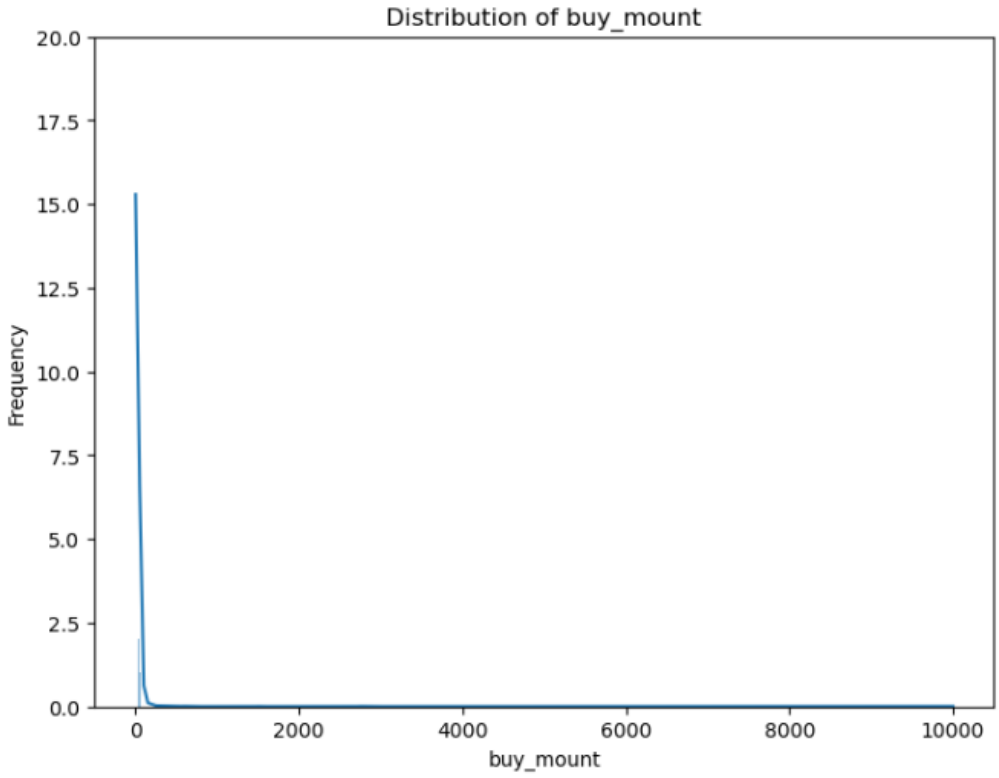
接下来我们处理buy_mount数据类型：

```
In [6]: data0.buy_mount.describe()
```

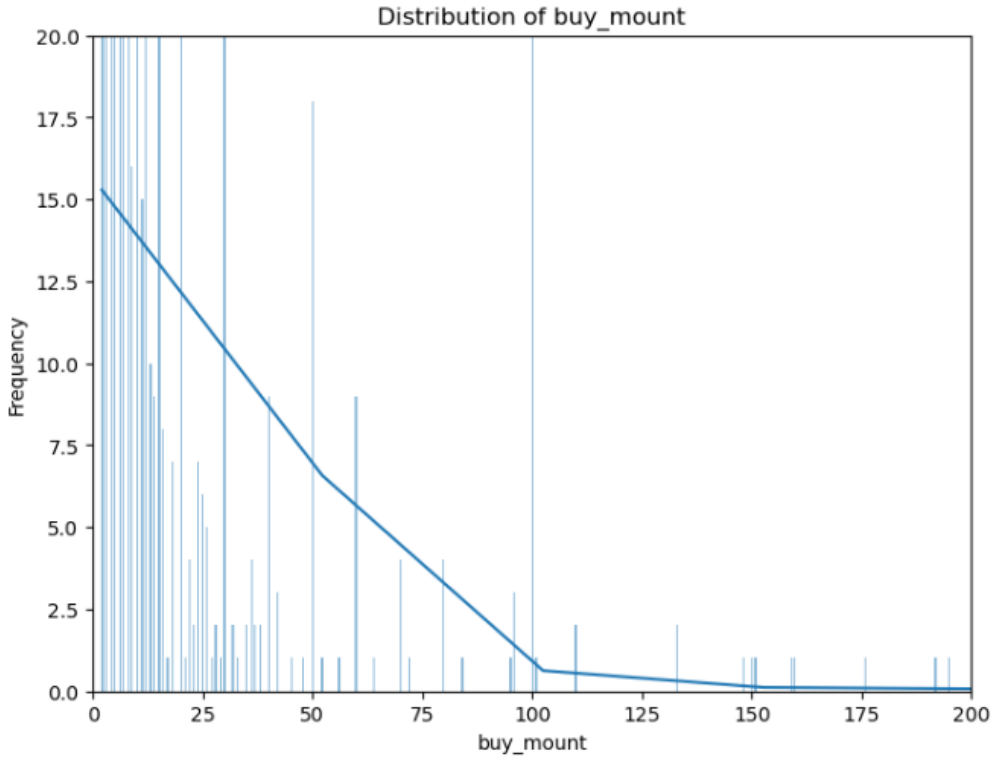
```
Out[6]: count    29971.000000
        mean         2.544126
        std         63.986879
        min          1.000000
        25%          1.000000
        50%          1.000000
        75%          1.000000
        max      10000.000000
        Name: buy_mount, dtype: float64
```

平均值为2.54，1的占比最大，超过了75%。但是最大值达到10000，明显不合理（可能是错误数据或者二道贩子），我们现将数据进行可视化，发现175之后的购买量基本为0：

```
In [7]: filtered_data0 = data0[data0['buy_mount'] != 1]
        plt.figure(figsize=(8, 6))
        sns.histplot(filtered_data0['buy_mount'], kde=True)
        plt.xlabel('buy_mount')
        plt.ylabel('Frequency')
        plt.title('Distribution of buy_mount')
        plt.ylim(0, 20)
        plt.show()
```



In [8]:
```
plt.figure(figsize=(8, 6))
sns.histplot(filtered_data0['buy_mount'], kde=True)
plt.xlabel('buy_mount')
plt.ylabel('Frequency')
plt.title('Distribution of buy_mount')
plt.ylim(0, 20)
plt.xlim(0, 200)
plt.show()
```



因此我们筛选删除大于175和小于0（从上面的分析可以看出来负的购买量不存在）

In [9]:
```
data0 = data0[data0.buy_mount <= 175]
data0.buy_mount.describe()
#至此，data0中的数据基本预处理完毕，userid/auction_id/category_1/category_2属于id类型，经检查无负数后无需处理
```

Out[9]:
```
count    29939.000000
mean         1.650356
std          4.924949
min          1.000000
25%          1.000000
50%          1.000000
75%          1.000000
max        160.000000
Name: buy_mount, dtype: float64
```

之后我们用类似的手段对data1进行处理：

首先查看数据类型，然后对birthday进行和data0中的day相同的操作，之后将gender=2（性别未知）的数据删除：

In [11]:
```
data1 = data1[data1.gender != 2]
data1['birthday'] = pd.to_datetime(data1.birthday.astype('str'))
data1 = data1[data1.birthday > '2000-01-01']
data2 = data1
data1['birthyear'] = data1.birthday.dt.year
data1['birthquarter'] = data1.birthday.dt.quarter
data1['birthmonth'] = data1.birthday.dt.month
data1.head()
```

Out[11]:

	user_id	birthday	gender	birthyear	birthquarter	birthmonth
0	2757	2013-03-11	1	2013	1	3
1	415971	2012-11-11	0	2012	4	11
2	1372572	2012-01-30	1	2012	1	1
3	10339332	2011-09-10	0	2011	3	9
4	10642245	2013-02-13	0	2013	1	2

In [12]:
```
data1.birthday.describe(datetime_is_numeric=True)
```

Out[12]:
```
count                            926
mean     2012-03-15 15:00:23.326133760
min                2002-12-05 00:00:00
25%                2011-02-12 00:00:00
50%                2012-08-07 12:00:00
75%                2013-09-26 18:00:00
max                2015-08-15 00:00:00
Name: birthday, dtype: object
```

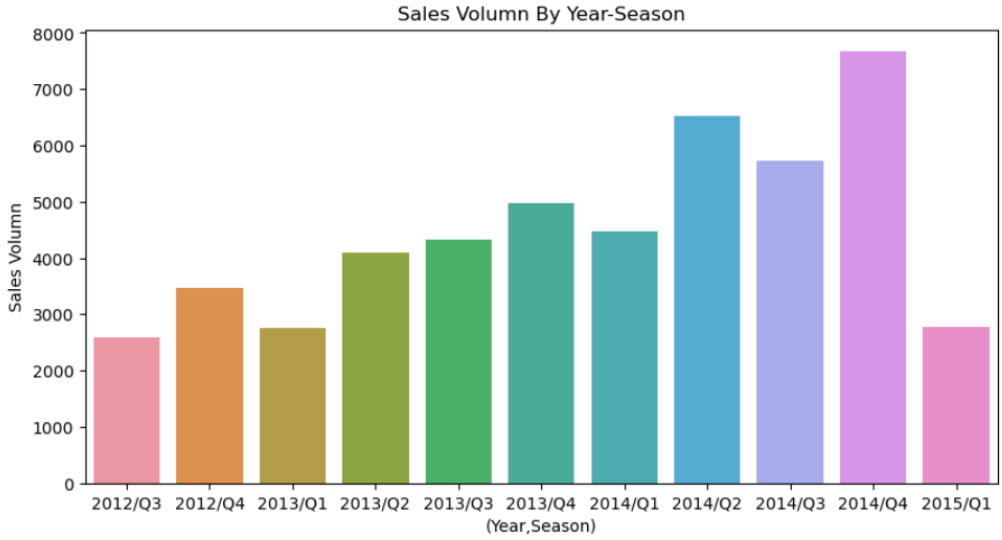
在处理过程中发现birthday中存在1997年这样明显不合理的存在（相当于至少15岁还在购买母婴用品，推测为数据错误或者二胎但是采用了一孩的出生日期），此处将该数据初步限制为出生日期为2000-01-01之后，在merge并计算age之后会根据age进行进一步的筛查。

可视化

通过数据可视化分析，我们能够直观地得到一些较为笼统的结论。

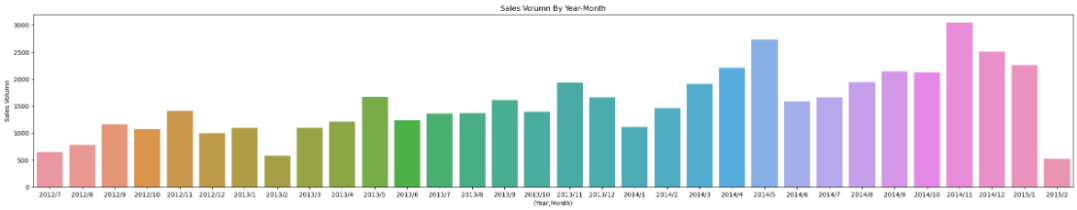
首先我们根据季度分类，绘制直方图：

In [13]:
```
year_quarter_stats = data0.groupby(by=['year', 'quarter'])['buy_mount'].sum()
plt.figure(figsize=(10, 5))
x_list = [str(idx[0]) + "/Q" + str(idx[1]) for idx in year_quarter_stats.index]
y_list = [int(value) for value in year_quarter_stats.values]
sns.barplot(x=x_list, y=y_list)
plt.title("Sales Volumn By Year-Season")
plt.xlabel("(Year,Season)")
plt.ylabel("Sales Volumn")
plt.show()
```



之后按照月份绘制直方图：

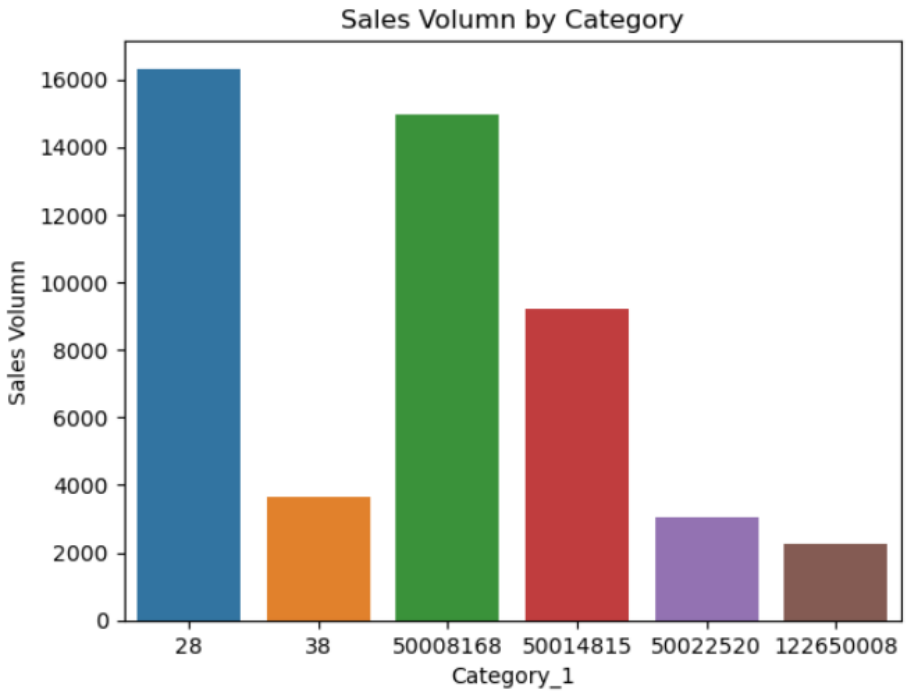
In [14]:
```
year_month_stats = data0.groupby(by=['year', 'month'])['buy_mount'].sum()

plt.figure(figsize=(30, 5))
x_list = [str(idx[0]) + "/" + str(idx[1]) for idx in year_month_stats.index]
y_list = [int(value) for value in year_month_stats.values]
sns.barplot(x=x_list, y=y_list)

plt.title("Sales Volumn By Year-Month")
plt.xlabel("(Year,Month)")
plt.ylabel("Sales Volumn")
plt.show()
```



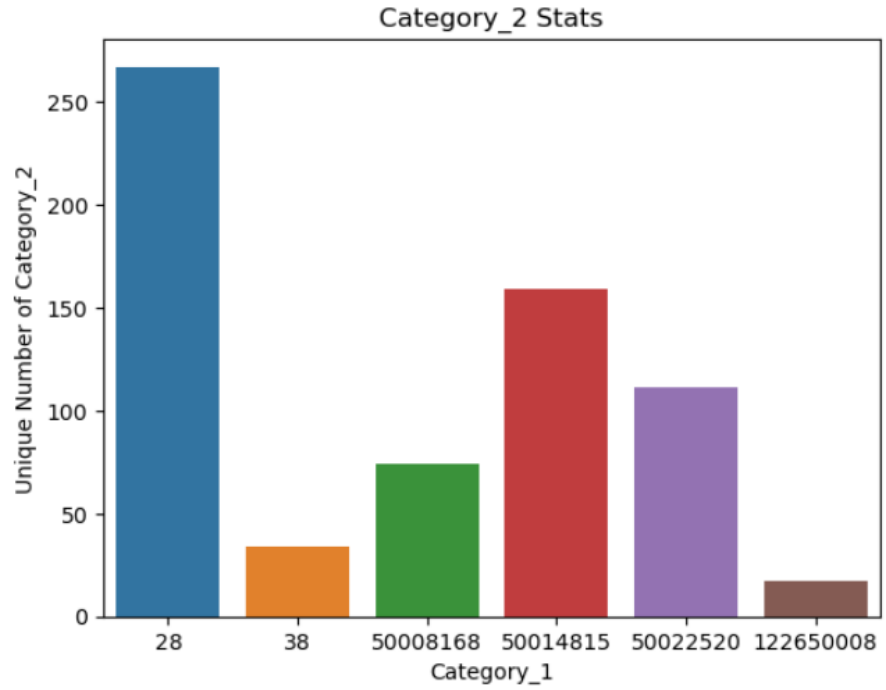
可以可看到2014/11明显销售量最大，推测是双十一影响。

之后我们还可以根据商品大类绘制销量和小类数量图：

In [15]:
```
category_1_stats = data0.groupby(by="category_1")['buy_mount'].sum()
sns.barplot(x=category_1_stats.index, y=category_1_stats.values)
plt.title("Sales Volumn by Category")
plt.xlabel("Category_1")
plt.ylabel('Sales Volumn')
plt.show()
```

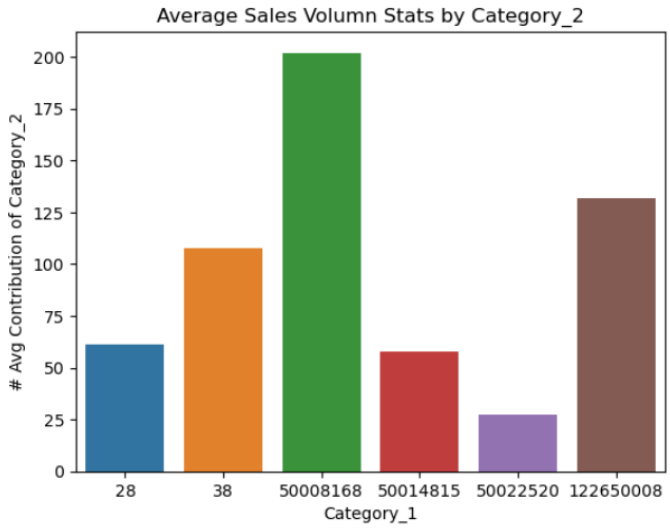


In [16]:
```
category_2_stats = data0.groupby('category_1')['category_2'].nunique()
sns.barplot(x=category_2_stats.index, y=category_2_stats.values)
plt.title("Category_2 Stats")
plt.xlabel("Category_1")
plt.ylabel("Unique Number of Category_2")
plt.show()
```



之后可以得出平均二级类销量比较直方图：

In [17]:
```
ave_category_stats = (data0.groupby("category_1")['buy_mount'].sum() / data0.groupby("category_1")['category_2'].nunique())

sns.barplot(x=ave_category_stats.index, y=ave_category_stats.values)
plt.title("Average Sales Volumn Stats by Category_2")
plt.xlabel("Category_1")
plt.ylabel("# Avg Contribution of Category_2")
plt.show()
```



接下来构建merge后的dataframe：

```
data = pd.merge(data0, data2)
data = data[data.day > data.birthday]
data['age'] = round(( data['day'] - data['birthday'] ) /
pd.Timedelta(days = 365) , 2)
data = data[data.age < 5]
data.age.describe()
```

计算age并排除age小于0和大于5的数据

聚类

此处直接调用KMeans聚类算法：

```
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split


features = ['category_2', 'category_1', 'buy_mount', 'year', 'quarter', 'gender', 'birthyear', 'birthquarter', 'age']
X = data[features]

# 创建KMeans模型并训练
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)

# 获取聚类结果标签
labels = kmeans.labels_

# 将聚类结果添加到原始DataFrame
data['cluster_label'] = labels

data
```

分类

在聚类的基础上进行分类，将数据首先分为训练集和测试集，之后训练模型并计算分类准确率：

In [42]:
```
from sklearn.metrics import accuracy_score


# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, data['cluster_label

# 创建分类器模型并训练
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

# 对所有样本进行预测
data['predicted_target'] = clf.predict(X)
y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("分类准确率：", accuracy)
```

```
分类准确率： 1.0
```

参考资料

本次实验中预处理和可视化部分参考了天池比赛论坛中的官方baseline：[淘宝母婴购物数据可视化分析baseline_天池notebook-阿里云天池 (aliyun.com)]