

TONGJI UNIVERSITY

《机器学习》

实验报告（大作业）

|  |  |
| --- | --- |
| 实验名称 | Lab3 聚类实验 |
| 小组成员 | 2152871 王颢瑾 |
|  | 2151569 明添识 |
|  | 2152203 叶哲轩 |
| 学院（系） | 电子与信息工程学院 |
| 专 业 | 计算机科学与技术 |
| 任课教师 | 李洁 |
| 日 期 | 2023年 12 月 7 日 |

机器学习聚类模型报告

本次任务基于“Facebook Live Sellers in Thailand”数据集，开展机器学习的聚类任务。主要聚类模型试验的开展分为了五个阶段，分别是数据预处理（检查并处理掉数据集中的空值和异常值,选择分析目标,可能需要进行特征工程来构造更有意义的特征）、选择聚类算法（常用的聚类算法有K-Means,DBSCAN等。根据研究目标和数据集特点选择合适的算法。比如可以使用K-Means对主播类型进行聚类）、确定聚类数目（聚类结果很大程度上取决于聚类数目k的选择。可以通过观察聚类评价指标来确定最优的k值。常用指标有轮廓系数、Calinski-Harabasz指数等）、训练模型与评估（将选定的聚类算法应用于数据集并生成聚类结果。计算评价指标来评估结果的质量。还可以进行可视化来更直观地判断聚类效果）和结果分析（分析并描述不同的聚类组所具有的特点,这些特点可以是主播的属性、商品类别、销售数据的统计特征等。这些发现的 customer/seller personas 可以提供商业决策支持）。通过聚类分析,可以更深入地理解不同类型的Facebook直播销售主播及其客户群特征,为电商和投资决策提供支持。

1. **任务目标与数据集说明**

本次任务基于“Facebook Live Sellers in Thailand”数据集，这份数据集记录了泰国Facebook平台上的实时产品销售情况。它收集了7050组具有12个属性的数据。每个数据项都对应一条Facebook帖子,这些帖子均与Facebook上的实时销售功能相关。数据包括销售的实时信息,发帖时间,以及衡量用户参与度的各种指标。这些指标记录了用户的分享、评论和表情符号反应数量,并区分出“爱”、“哇”、“哈哈”、“悲伤”和“生气”这5种反应类型。通过与其他形式的内容(文本、视频、直播和图片)的比较研究,以及对Facebook Live销售的季节性统计分析等方式,这份数据集能成为研究客户与Facebook Live销售渠道互动的基础。

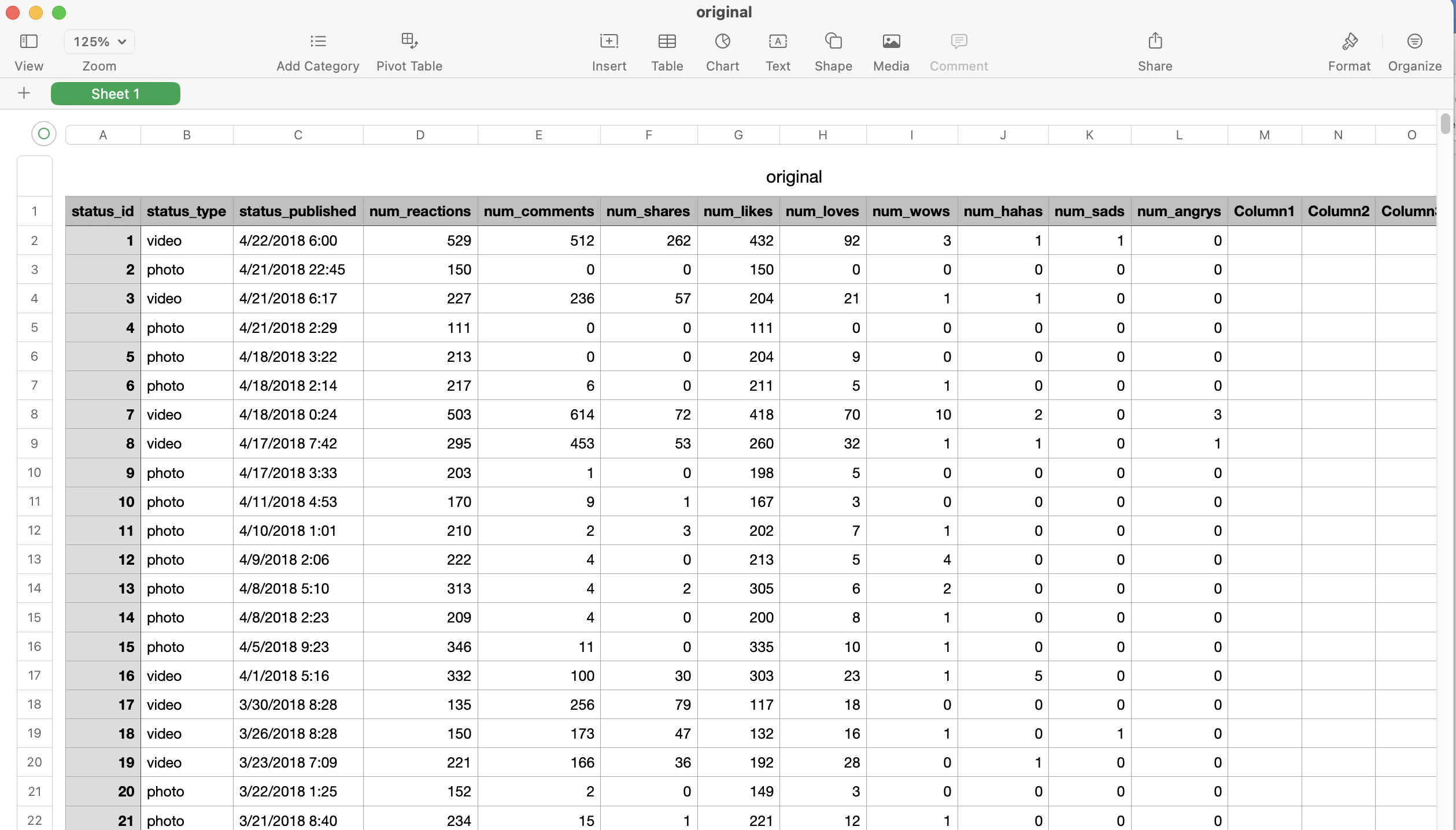


图 1 数据集样态

1. **数据清洗**

这一部分主要给出数据预处理以及清洗后的数据特征展示，代码使用pandas进行调用如下：

|  |
| --- |
| original\_data = pd.read\_csv('./dataset/original.csv',index\_col=0)  original\_data.info() |

表 1 数据特征展示代码

这一部分将展示数据的具体特征，其中发现数据的最后四行是全空的无效信息，所以选择删除这四个信息。同时注意到数据的特点中有两列数据不是可以直接用于计算的数据特征，所以使用有效的方法解决数据的特性问题。并且最后根据筛选出的数据特点做一次特征整合，以方便进行后续的特征处理。删除信息之后，我们筛选出的特征有status\_type和status\_published两个非数值特征。下面展示status\_published的分布图。

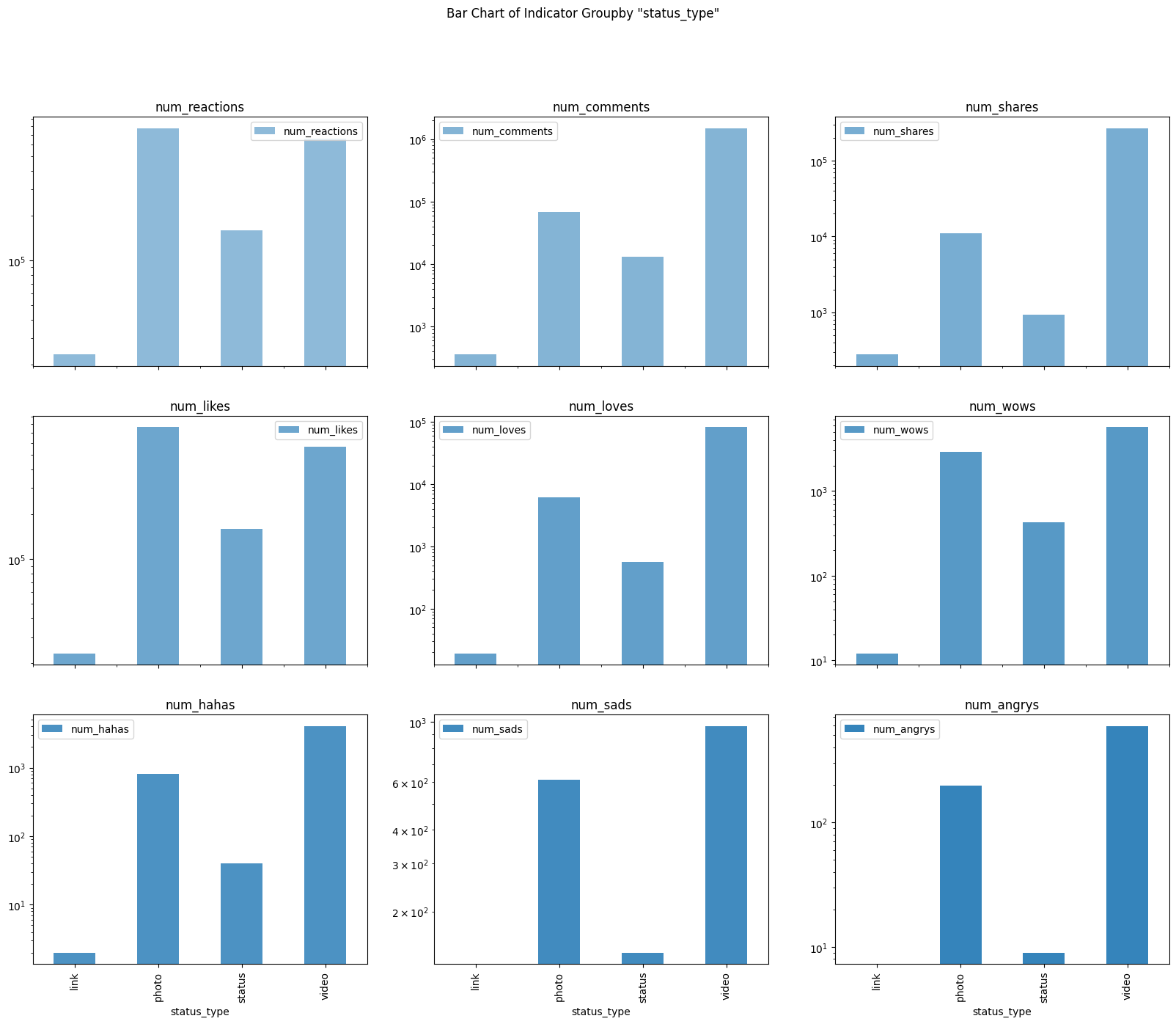


图 2 status\_type分布图

这里使用了sklearn库中的LabelEncoder进行运算。LabelEncoder是sklearn中用于数据预处理的类,其工作原理是将类别型特征的值映射为从0开始的整数索引。比如一个特征有三种可能的取值A,B,C,LabelEncoder会将它们分别映射为0,1,2。 LabelEncoder可以通过fit\_transform方法先学习数据集中的所有类别,然后将其转化为整数编码。这样做的好处是许多机器学习算法需要输入数字特征,而类别型特征无法直接输入到模型中。 使用LabelEncoder编码后可以将类别型特征数字化,同时也会减少特征的基数,降低特征空间的维度。需要注意,LabelEncoder编码后的值并没有大小关系,0,1,2并不意味着有某种排序,仅是简单的索引。因此,使用LabelEncoder需结合算法特点和业务意义来判断。

而在时间字符串上，我们使用time生成时间戳再归一化的方法用数值信息替代原始的文本信息。具体的时间戳归一化代码参考下图：

|  |
| --- |
| times = clean\_data['time'].to\_list()  for time in times:  time = float(time)  min\_ts = min(times)  max\_ts = max(times)  new\_time = []  for time in times:  time = (time-min\_ts)/(max\_ts-min\_ts)  new\_time.append(time)  clean\_data['final\_time'] = pd.DataFrame(new\_time)  clean\_data.drop(['time'],axis=1,inplace=True) |

表 2 归一化时间戳

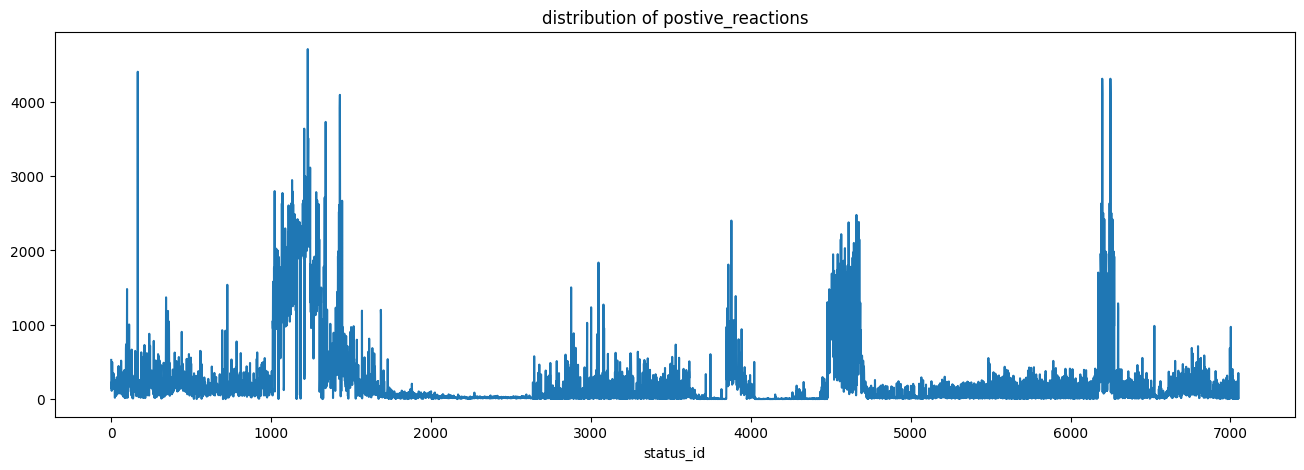
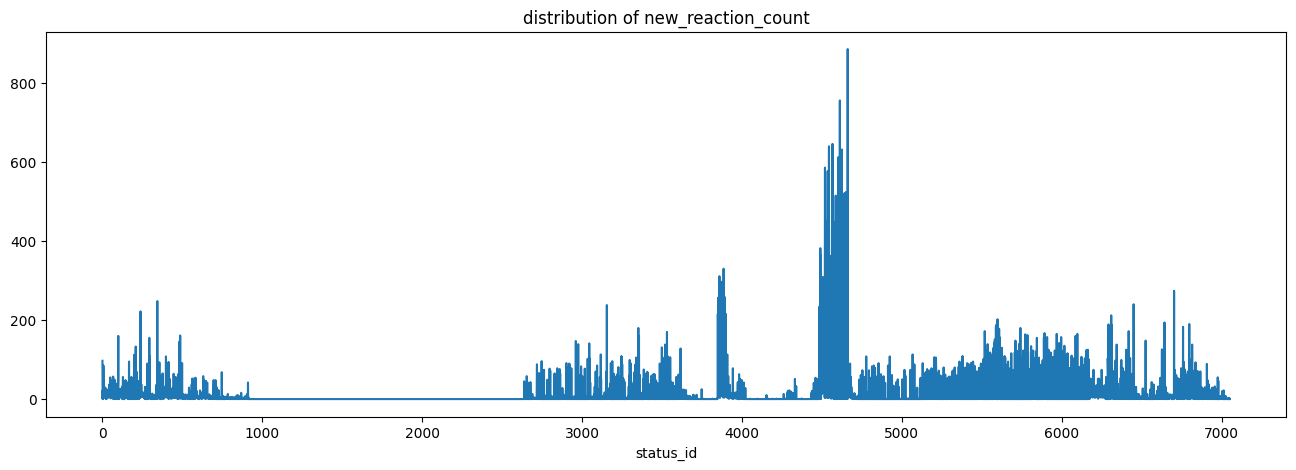
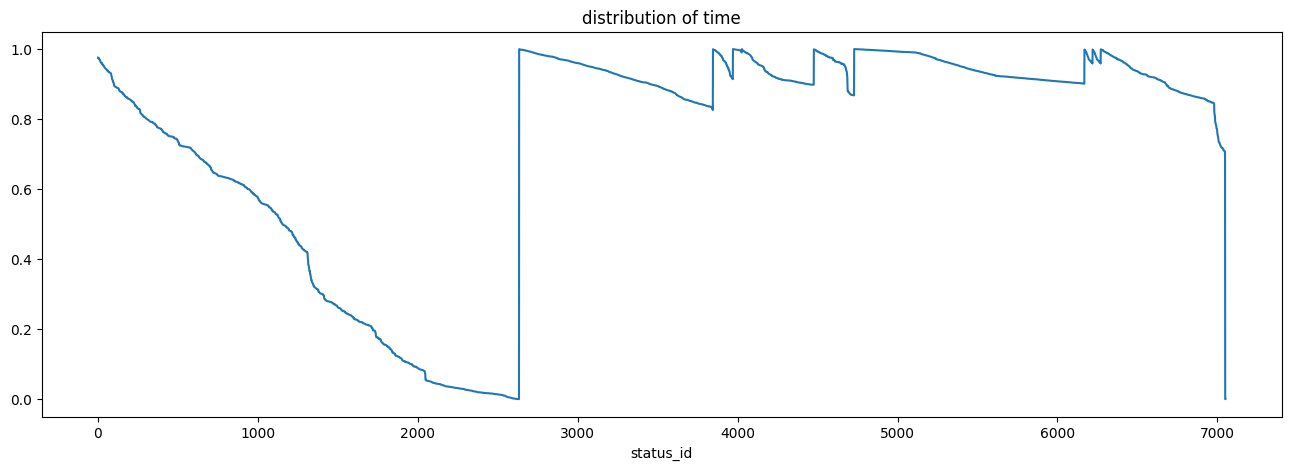
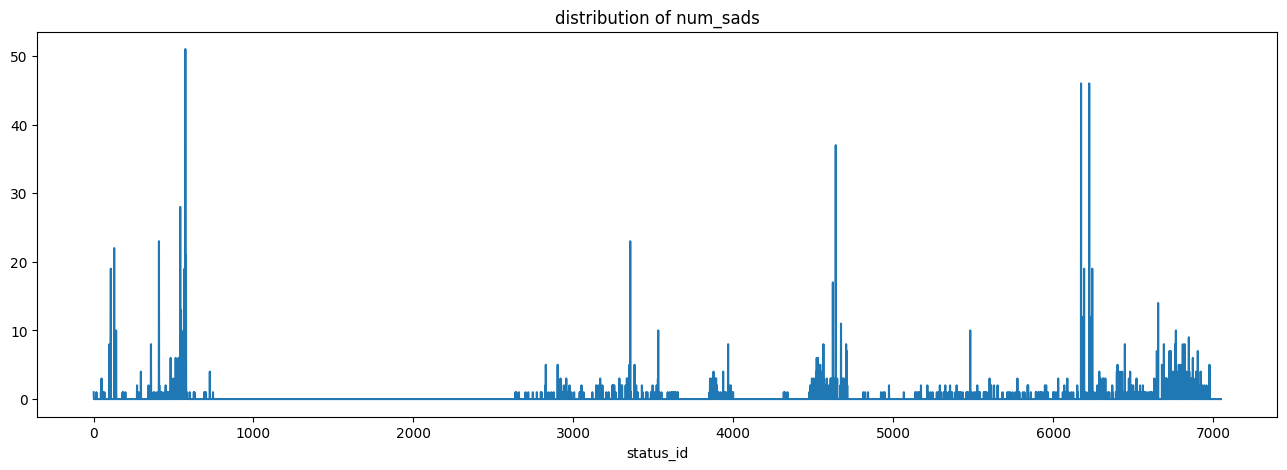
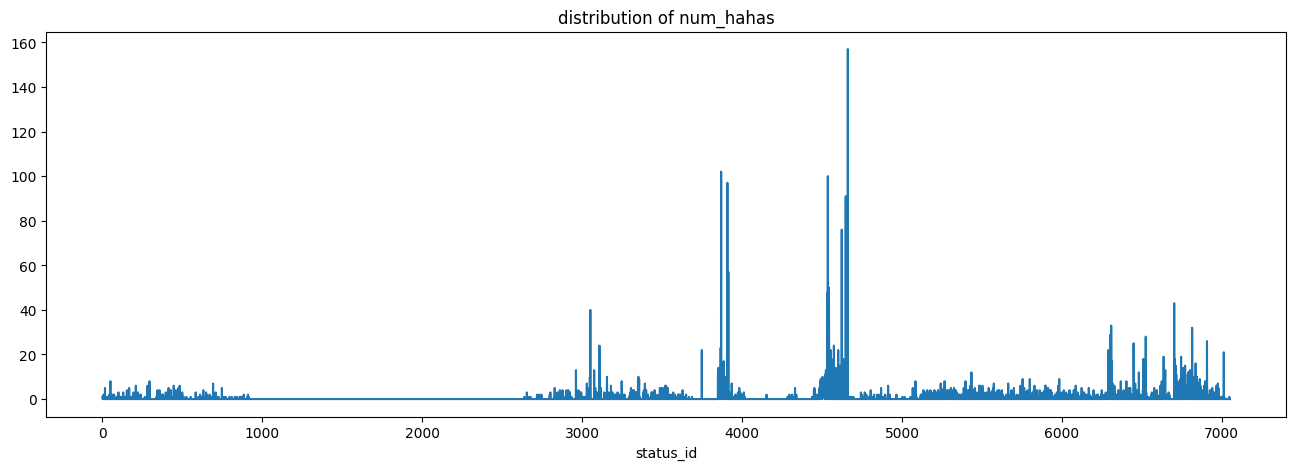
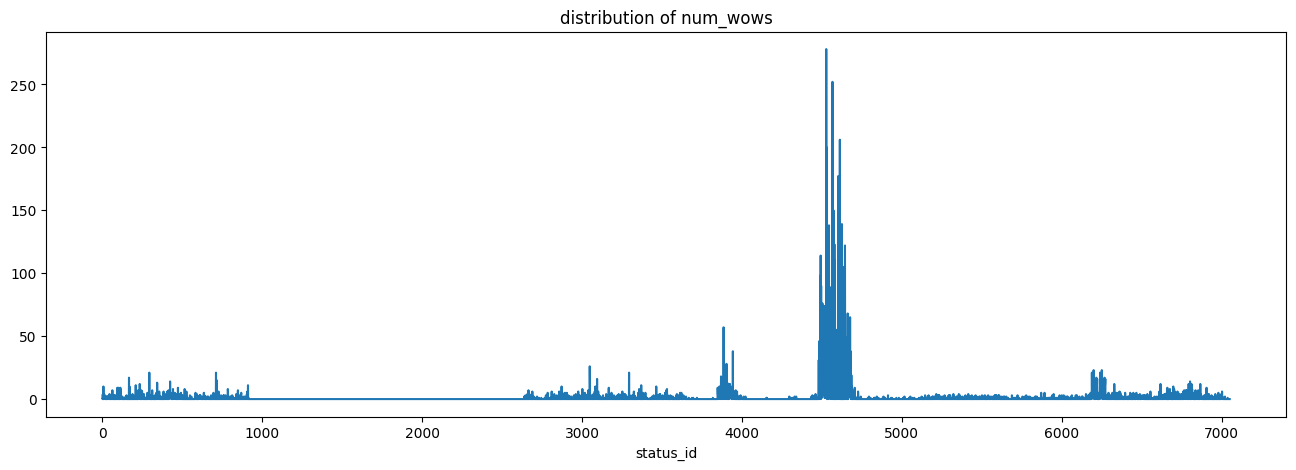
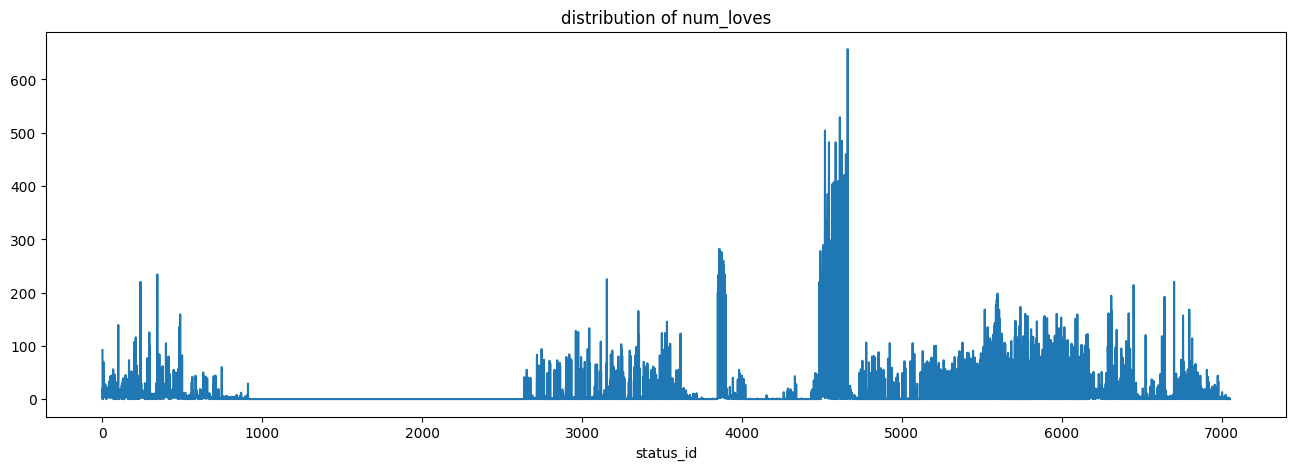
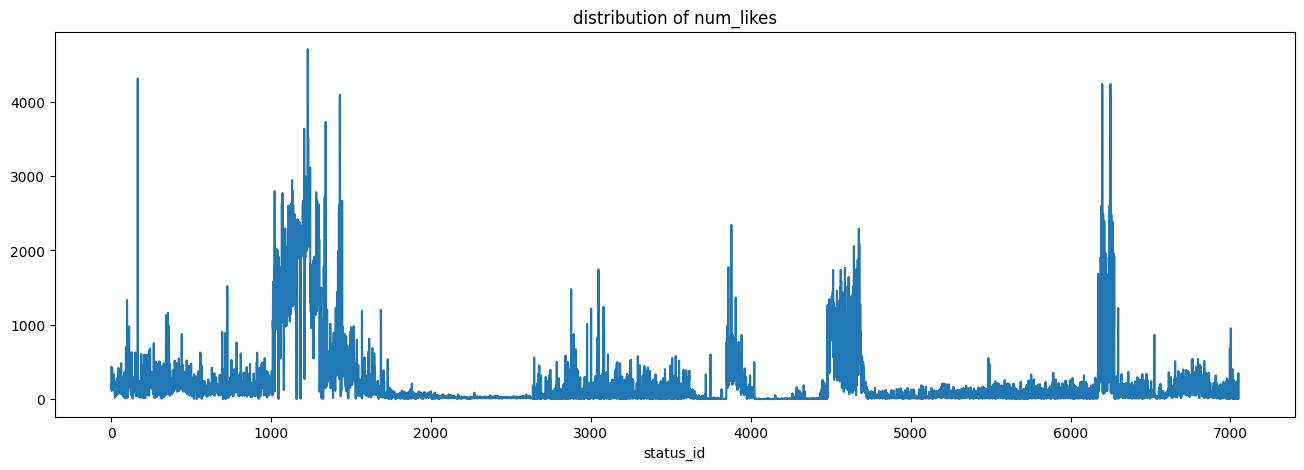
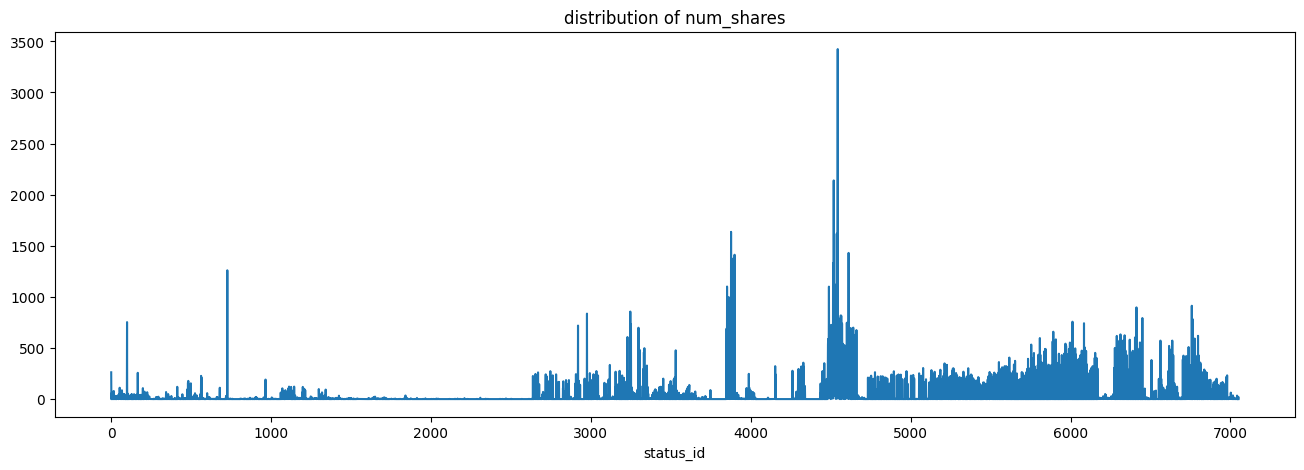
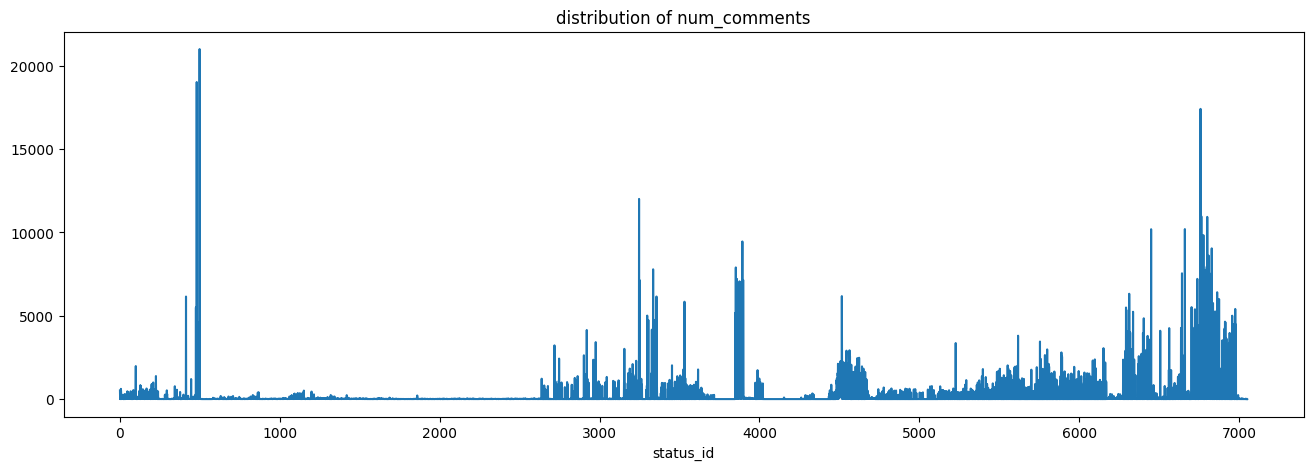
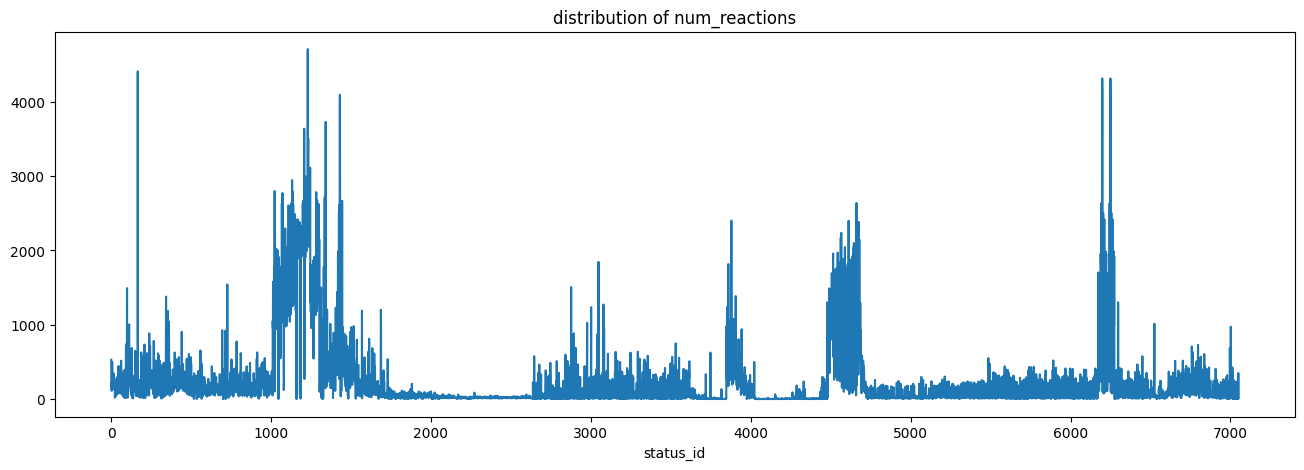
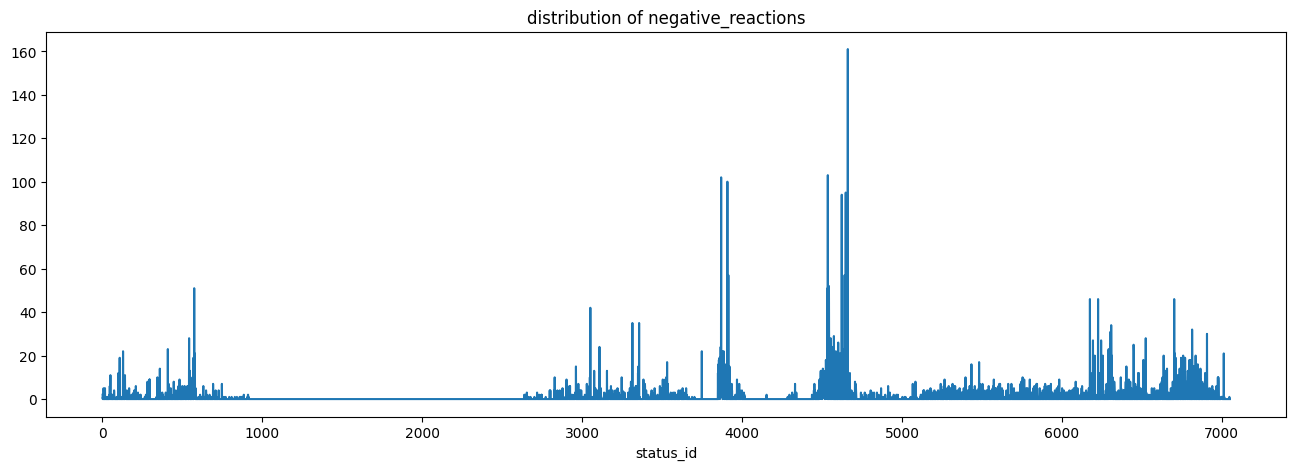
根据上面的处理得到的纯数据特征，我们还进行了一次数据特征的组合。构造了postive\_reactions和negative\_reactions去记录用户的两种反应态度，构造了'interest'去记录用户对于此条动态的感兴趣程度。最终的预处理数据特征参考下表：

|  |  |  |  |
| --- | --- | --- | --- |
| # | Column | Non-Null Count | Dtype |
| 0 | status\_type | 7050 non-null | int64 |
| 1 | num\_reactions | 7050 non-null | int64 |
| 2 | num\_comments | 7050 non-null | int64 |
| 3 | num\_shares | 7050 non-null | int64 |
| 4 | num\_likes | 7050 non-null | int64 |
| 5 | num\_loves | 7050 non-null | int64 |
| 6 | num\_wows | 7050 non-null | int64 |
| 7 | num\_hahas | 7050 non-null | int64 |
| 8 | num\_sads | 7050 non-null | int64 |
| 9 | num\_angrys | 7050 non-null | int64 |
| 10 | final\_time | 7050 non-null | float64 |
| 11 | new\_reaction\_count | 7050 non-null | int64 |
| 12 | positive\_reactions | 7050 non-null | int64 |
| 13 | negative\_reactions | 7050 non-null | int64 |
| 14 | interest | 7050 non-null | float64 |

表 3 处理后数据特征

1. **数据特征可视化**

根据处理得到的数据，在这里展示数据的分布情况。具体如下：

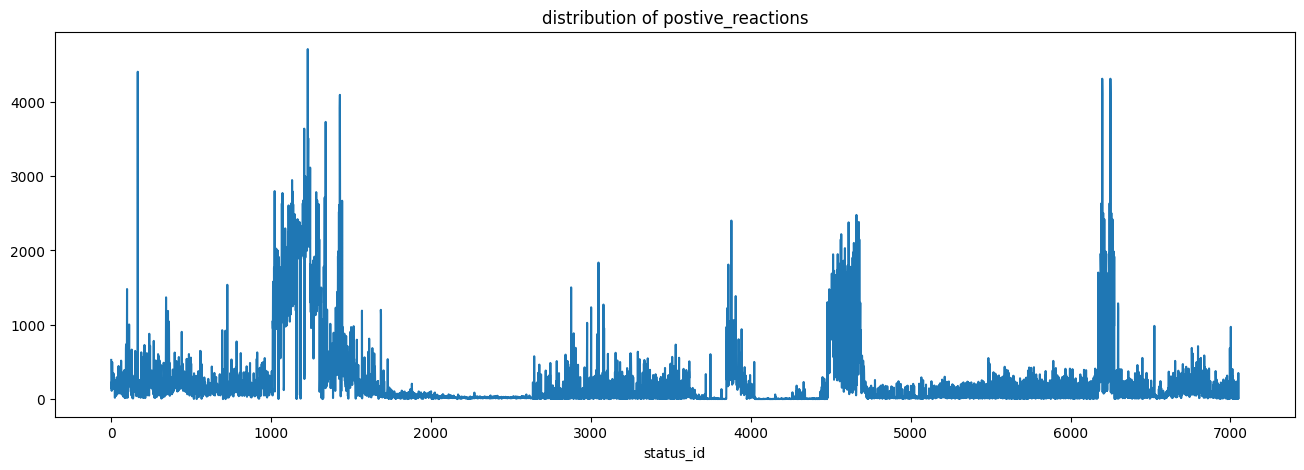


图 3 数据分布情况

相关矩阵是展示多个数值特征之间线性相关关系的矩阵,矩阵中的每个元素表示对应的两个特征间的皮尔逊相关系数,系数范围介于-1到1之间,1表示正相关,-1表示负相关,0表示不相关。相关矩阵对角线元素为1,且矩阵中心对称。通过相关矩阵可以清晰直观地查看多个数值特征相互之间的相关性和程度,是特征选择、降维等过程常用的工具。相关矩阵越密集说明FEATURES间相关性越大,反之则相关性越小。 sklearn中提供了相关矩阵的可视化函数方便分析。

下面给出处理后数据的相关性系数：

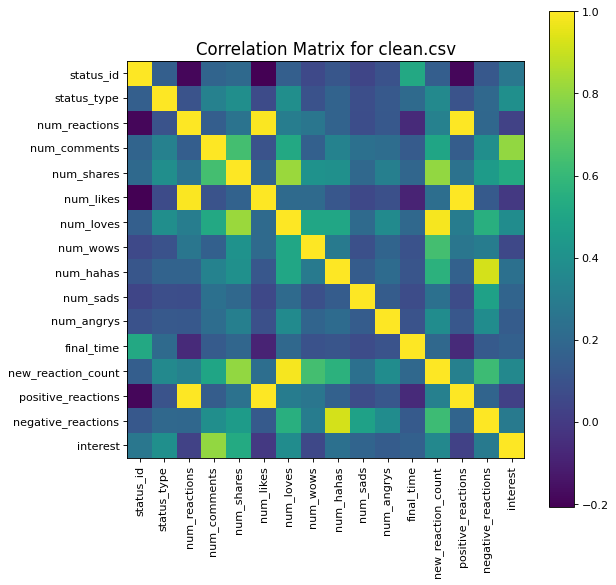


图 4 相关矩阵

整体散点密度图(Overall Scatter Plot Density)是一种基于散点图(Scatter Plot)的可视化方法,用于显示两个连续变量或分类变量之间的关系及其分布情况。与传统的散点图不同,整体散点密度图将大量离散的点用颜色代表点密度的变化,点的密集区域颜色越深,而颜色渐变代表点密度从高到低的变化,通过颜色变化直观显示变量分布的整体密度情况。这种可视化方法溢出显示了所有数据点的共性分布,克服了传统散点图在数据量大时会遮挡数据的缺点,更好地展示变量分布的整体聚集景象及变化趋势。这种图常用于显示存在复杂聚类或异质分布的数据集,或希望呈现数据整体分布而非局部细节的场景。它既保留了散点图直观显示变量关系的优点,又利用颜色对密度变化进行了形象的可视编码。下面给出我们数据分布的整体散点密度图：

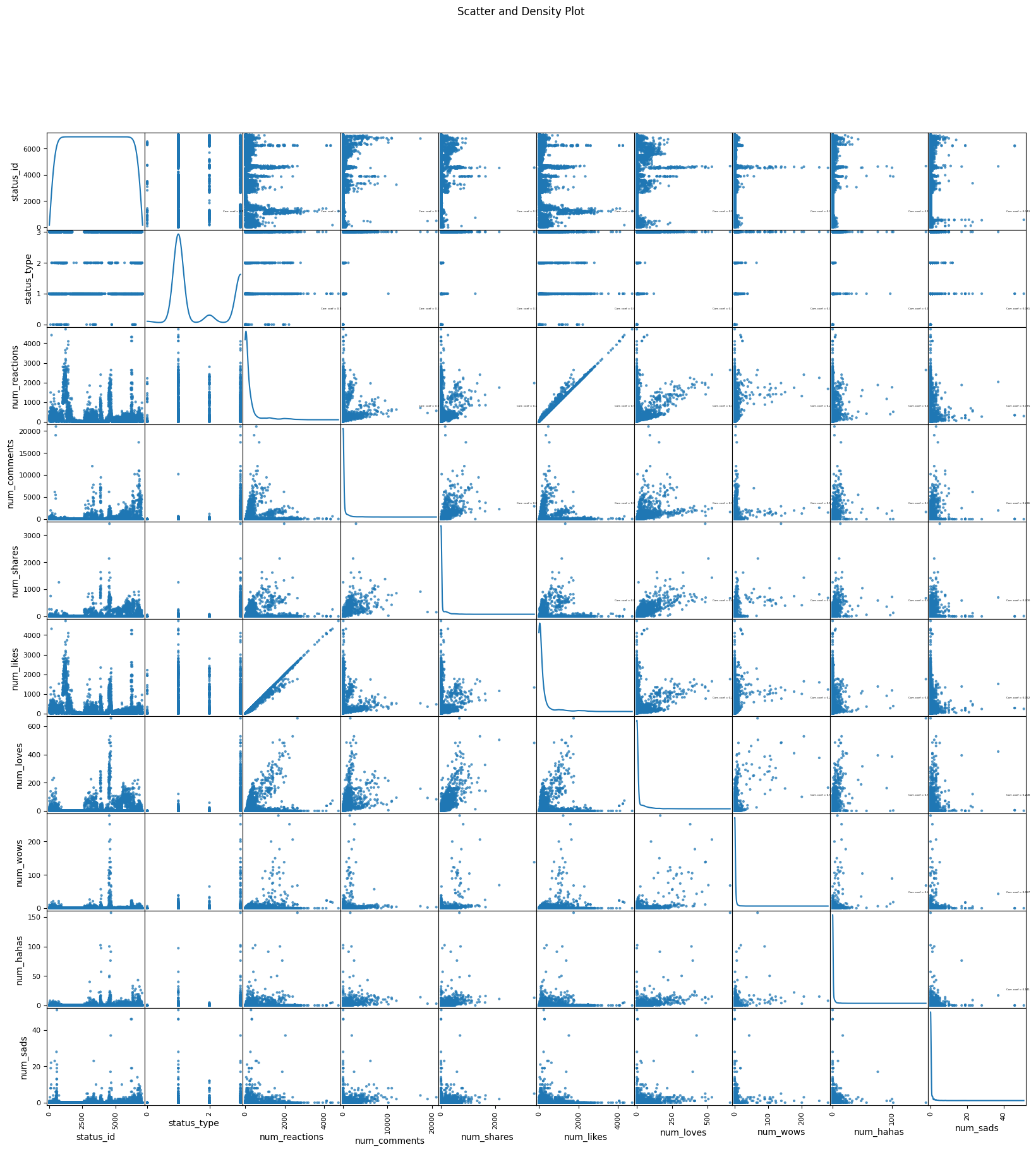


图 5 整体散点密度图

1. **聚类过程**

**A.KMeans**

K-Means是一个著名的聚类算法,其基本原理是将给定的数据集按照一定规则分为K个簇,使得每个簇内的数据点之间的距离尽可能小,而不同簇的数据点之间的距离尽可能大。其中最关键的规则是选取能最小化各簇内距离平方和的聚类中心点。

具体来说,K-Means聚类分为以下几个步骤:

1) 给定数据集,人工指定参数K值,代表划分K个簇

2) 从数据集中随机选取K个数据点作为初始聚类中心

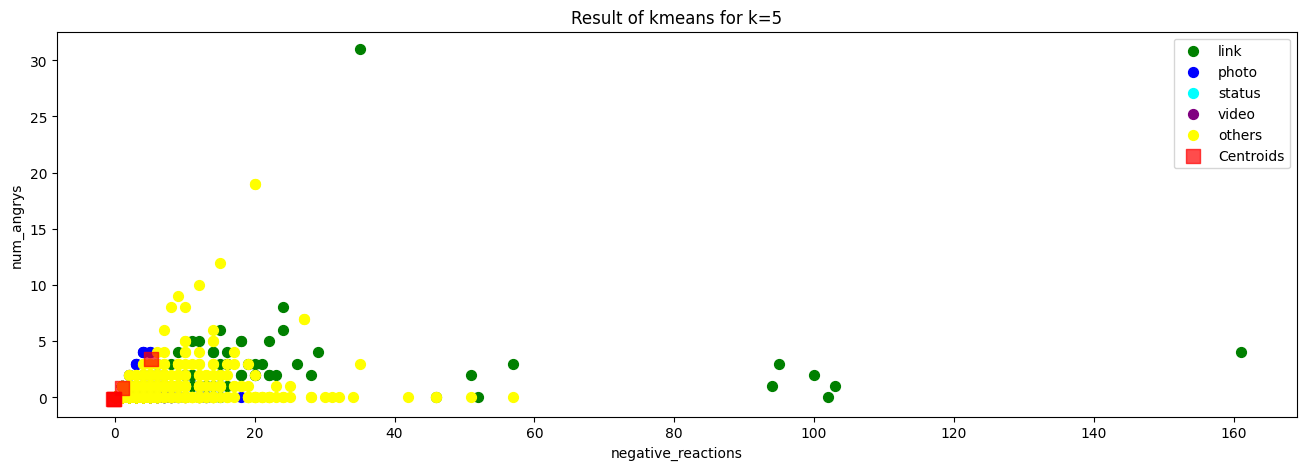
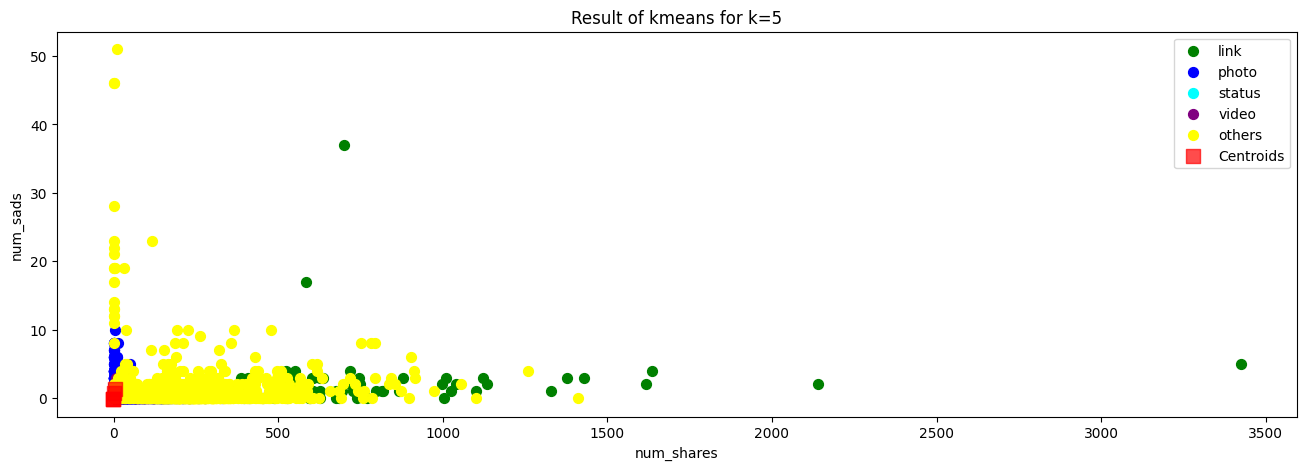
3) 计算所有数据点与K个聚类中心的距离,按照距聚类中心最近的原则将每个数据点分配给某个簇

4)对每个簇内数据点重新计算均值以更新该簇的聚类中心

5)迭代上述计算重心和分配数据点的过程,直到聚类中心稳定即相邻两次迭代结果变化较小为止

通过不断迭代优化,K-Means能够使聚类结果趋于最优,找到数据集的天然聚类结构。这种简单但有效的聚类思想被广泛应用到许多领域。

下面给出sklearn.cluster中的模型进行训练，并且给出最终结果的可视化：



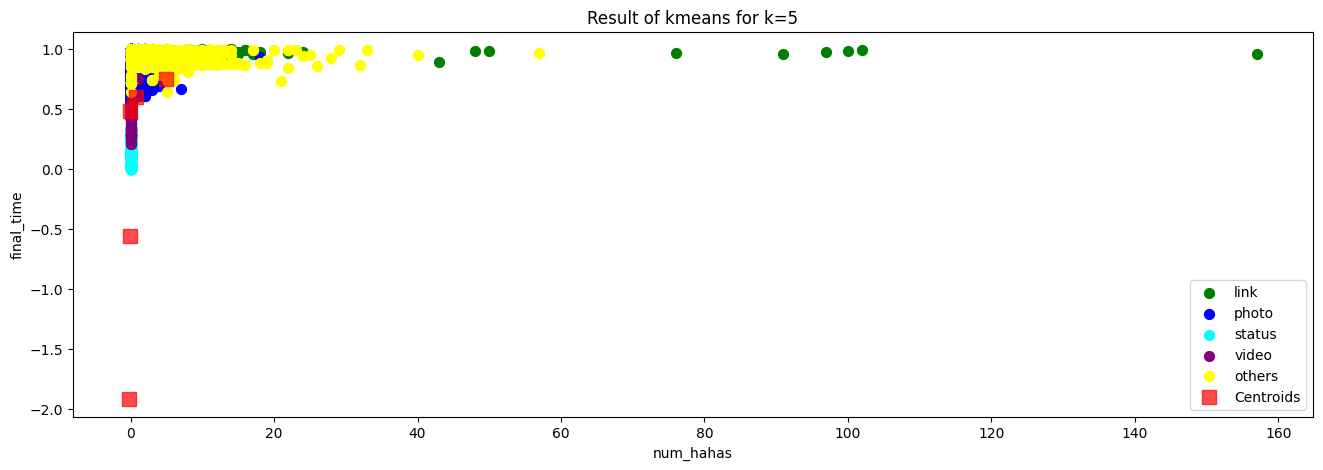
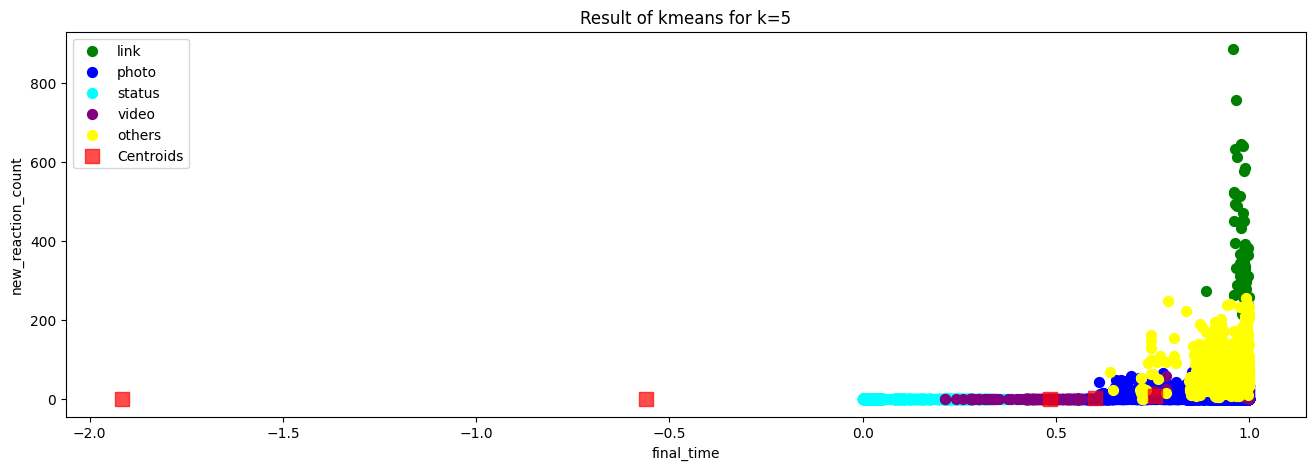


图 6 KMeans聚集结果

**B.高斯混合模型聚类**

高斯混合模型聚类首先假设数据是由多个高斯分布组成的混合分布生成的。其聚类过程主要包含以下步骤:

1. 随机初始化高斯分布的均值、协方差矩阵和混合系数。即为每个聚类中心初始化一个高斯分布模型。

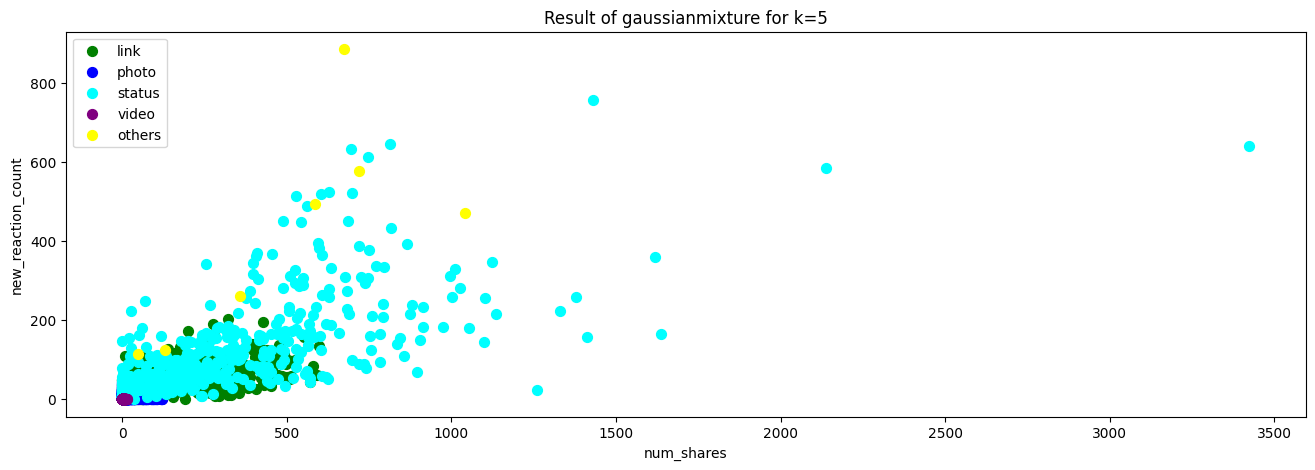
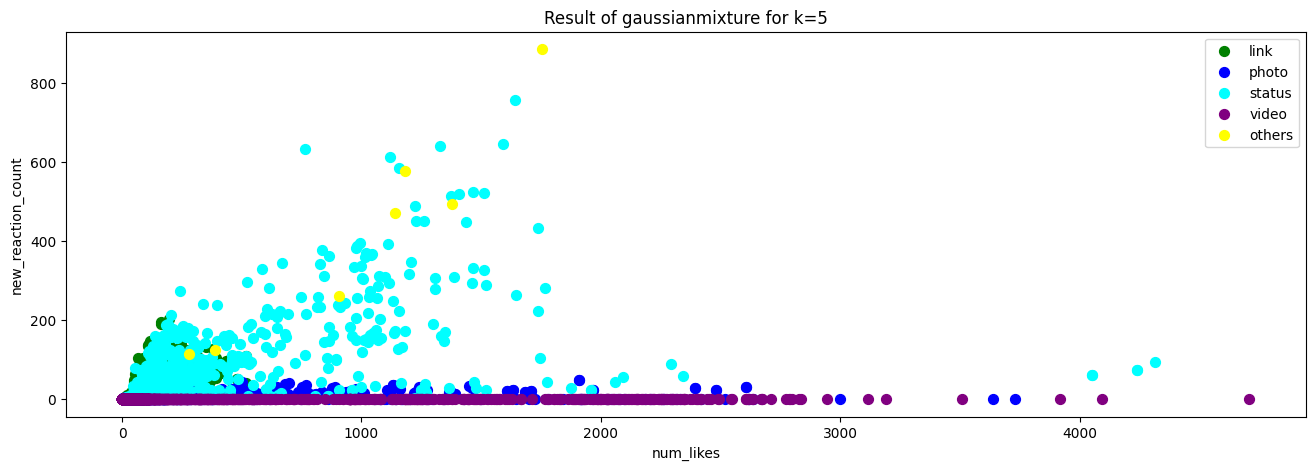
2. E步:计算每个数据点属于每个高斯分布的后验概率。这表示每个数据点属于每个聚类的概率。这需要用到贝叶斯规则来计算数据点来自每个高斯分布的概率,并结合混合系数得到后验概率。

3. M步:最大化参数的似然函数。具体为重新估计每个高斯分布的均值、协方差矩阵和混合系数,使似然函数最大化。均值可以更新为按posterior概率加权的数据点平均值。协方差矩阵同理。混合系数更新为每个高斯分布对应的所有数据点posterior概率之和。

4. 迭代E步和M步直到收敛。即参数估计结果稳定时停止迭代。

5. 根据每个数据最后的posterior概率将其硬性分类到posterior概率最大的高斯分布对应的聚类中。

这样通过迭代E步和M步,不断更新模型参数和posterior概率,实现了软聚类,将原数据按posterior概率最大划分到相应的聚类中,完成了聚类建模。下面给出这个模型的聚类结果：

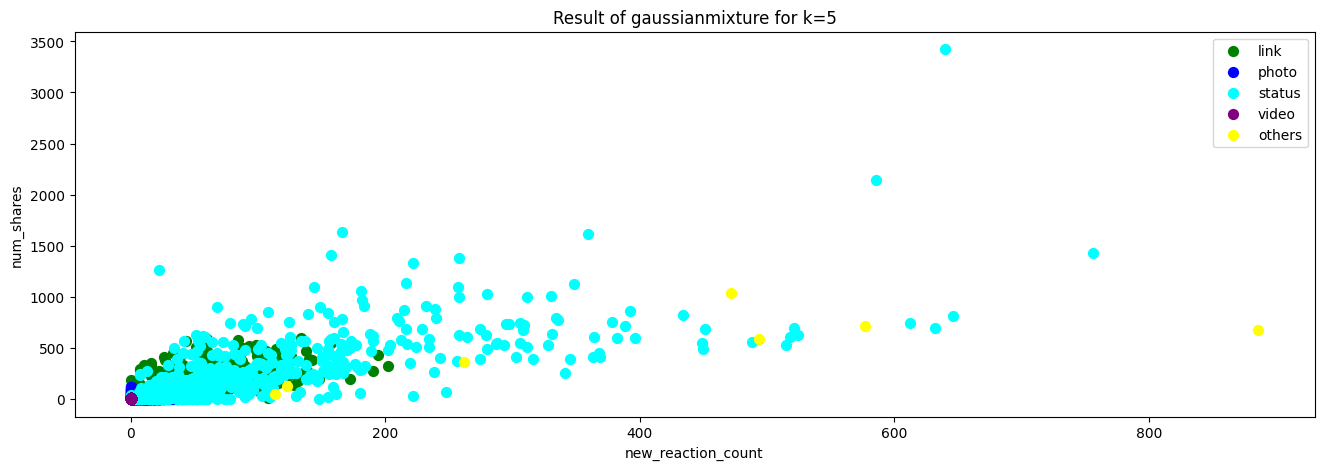
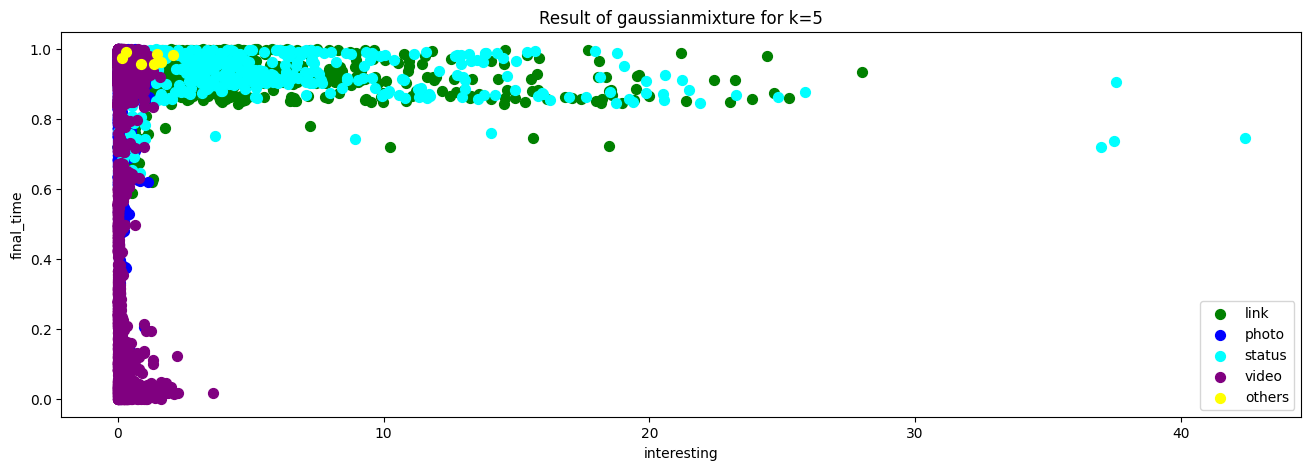
 

图 7 高斯混合模型分布

**C.DBSCAN模型聚类**

DBSCAN(Density-Based Spatial Clustering of Applications with Noise)是一种基于密度的聚类算法,其聚类过程如下:

1. 选择一个半径参数Eps和密度阈值MinPts。

2. 从任意未访问点开始,找出Eps领域内的所有点,如果点的个数大于等于MinPts,则构成一个簇。这些点标记为已访问。

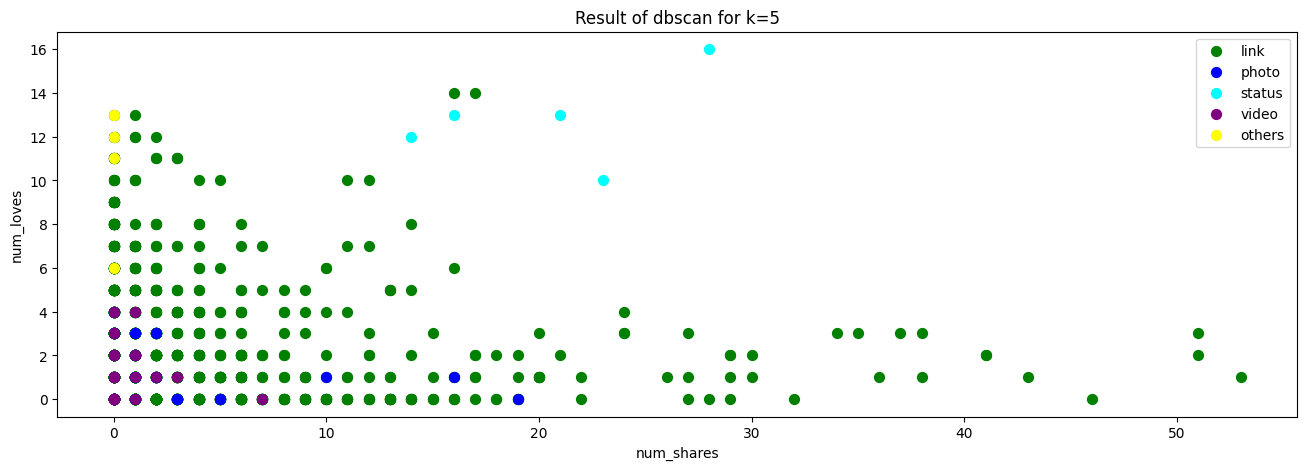
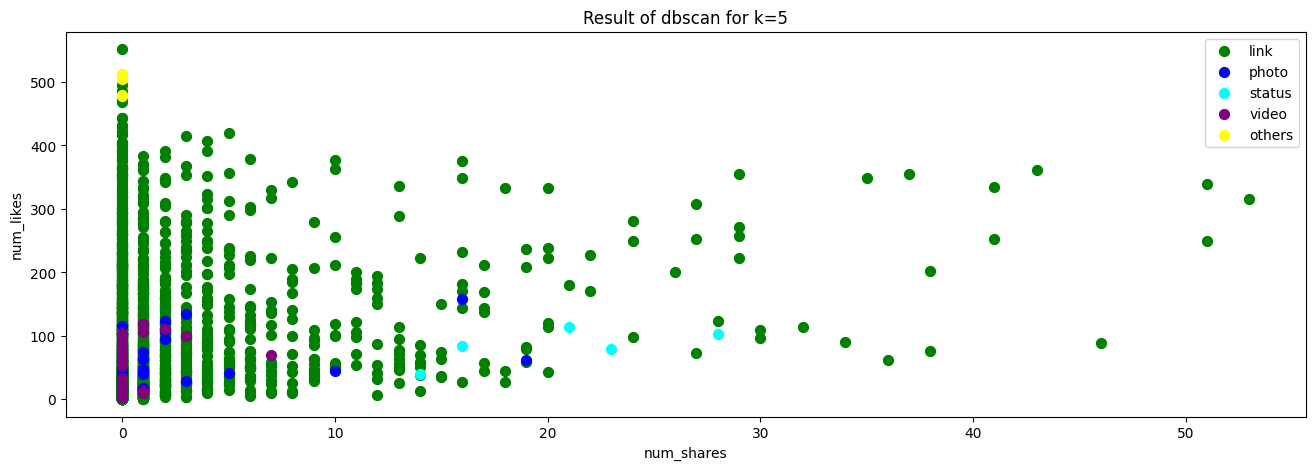
3. 对新找到的点重复步骤2,直到没有点可以再添加到当前簇中。

4. 选择一个未访问的点,重复步骤2、3形成新簇。

5. 重复步骤4,直到所有点都标记为已访问。未被任何簇吸收的点会标记为噪声点。

6. 所有形成的簇产生最终的聚类结果。

DBSCAN聚类的主要思想是以高密度区域中的点为核心构建簇,通过密度相连扩张区域来聚集点。它可以发现任意形状的聚类,同时检测噪声点。选择合适的参数Eps和MinPts对其结果影响很大。

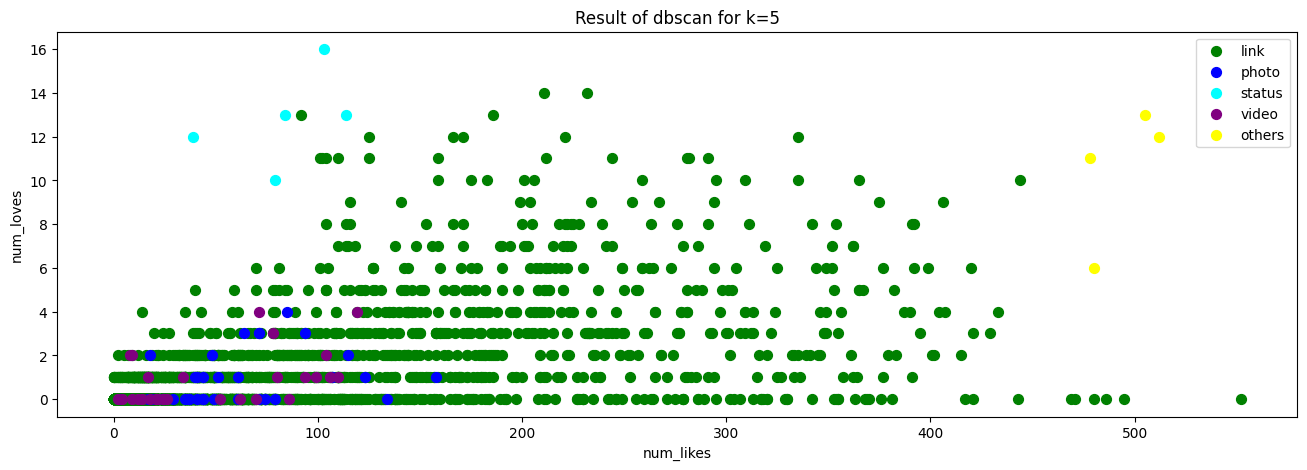
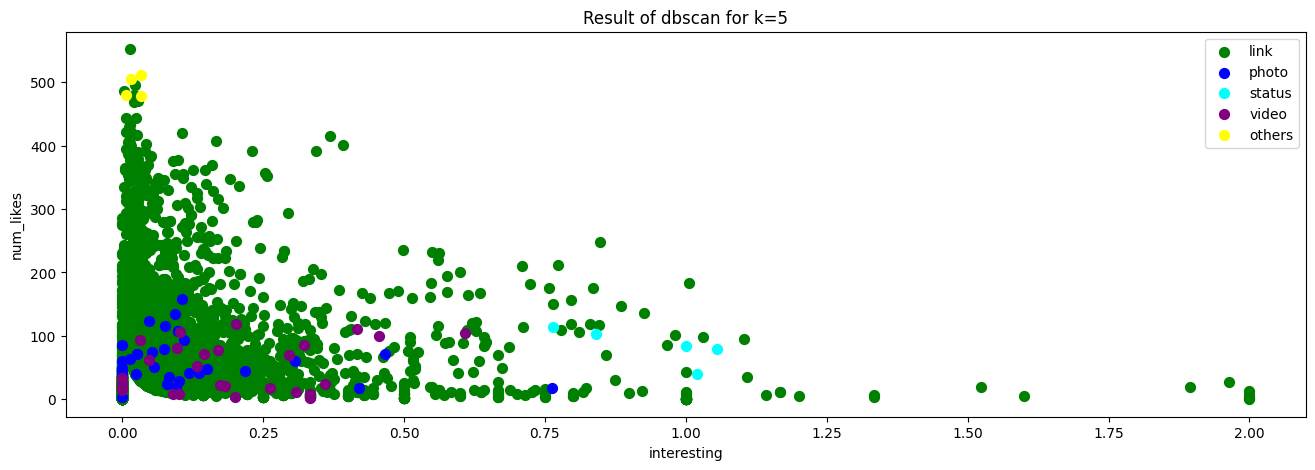
 

图 8 DBSCAN聚类结果

**D.Agglomerative模型聚类**

Agglomerative聚类(层次聚类)是一种自下而上的层次聚类方法,其聚类过程如下:

1. 开始时每一个数据点是一个单独的聚类;

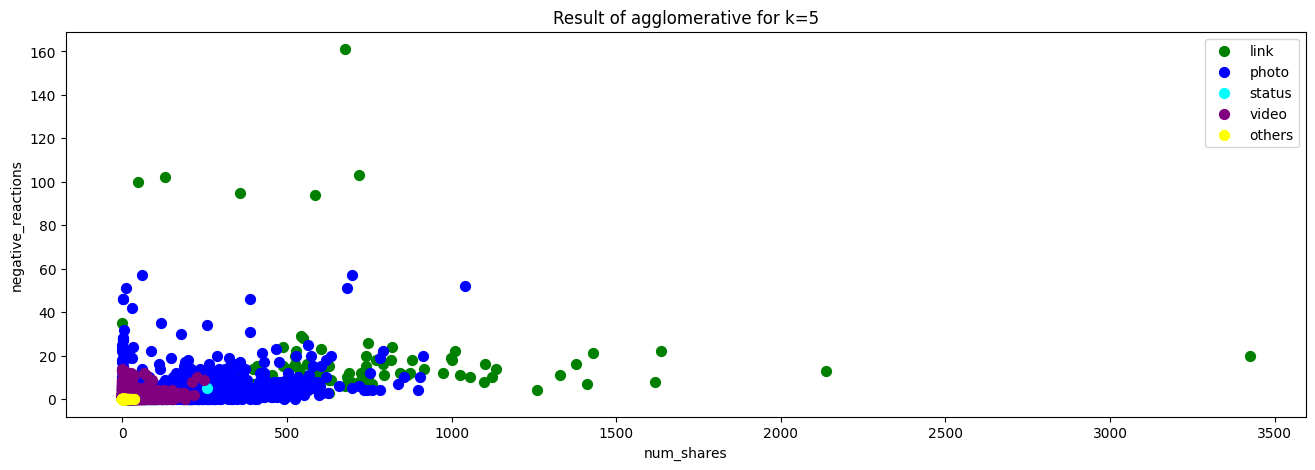
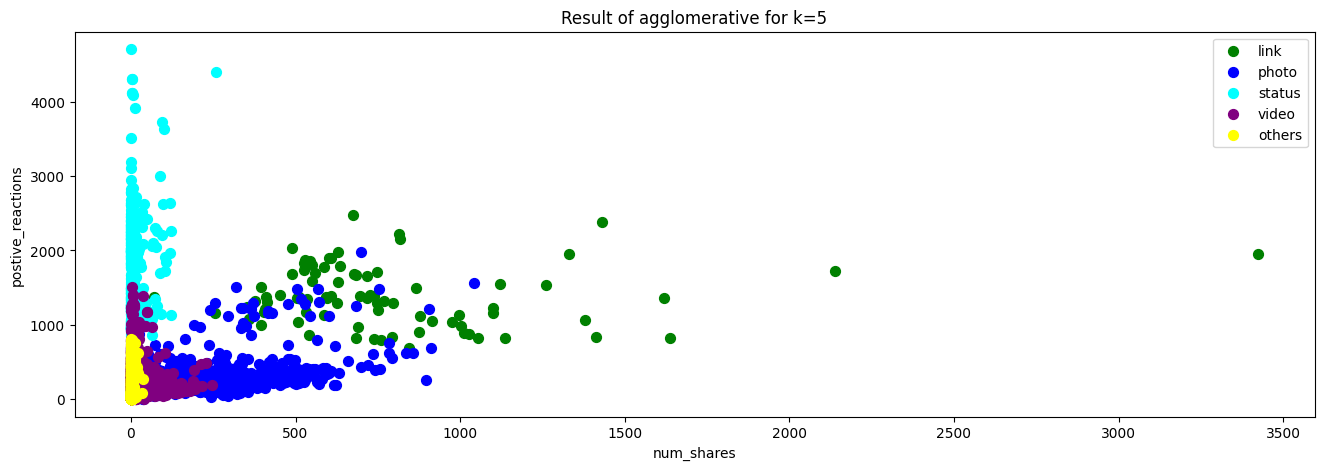
2. 根据聚类间距离的相似性准则,计算并选择最相似(距离最近)的聚类对进行合并;

3. 更新距离矩阵中的数据,以反映聚类间距离值的变化;

4. 重复步骤 2 & 3,直到满足停止条件。最常用的停止条件有聚类数量减少到某个值。

根据不同的相似性度量,常用的聚类距离包括欧式距离、曼哈顿距离或余弦相似性系数等。常见的相似性准则包括单链准则、完全链准则、平均链准则等。

Agglomerative 聚类通过不断合并相邻最近的聚类实现从单一数据点聚合为层次聚类的过程,可以通过树形图展示聚类的合并过程。其聚类结果如下：



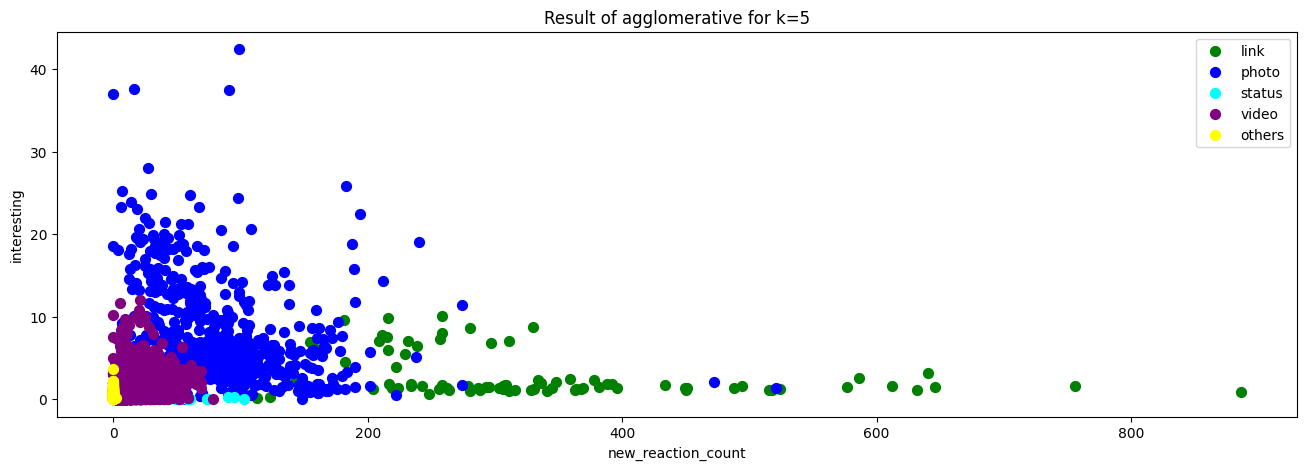
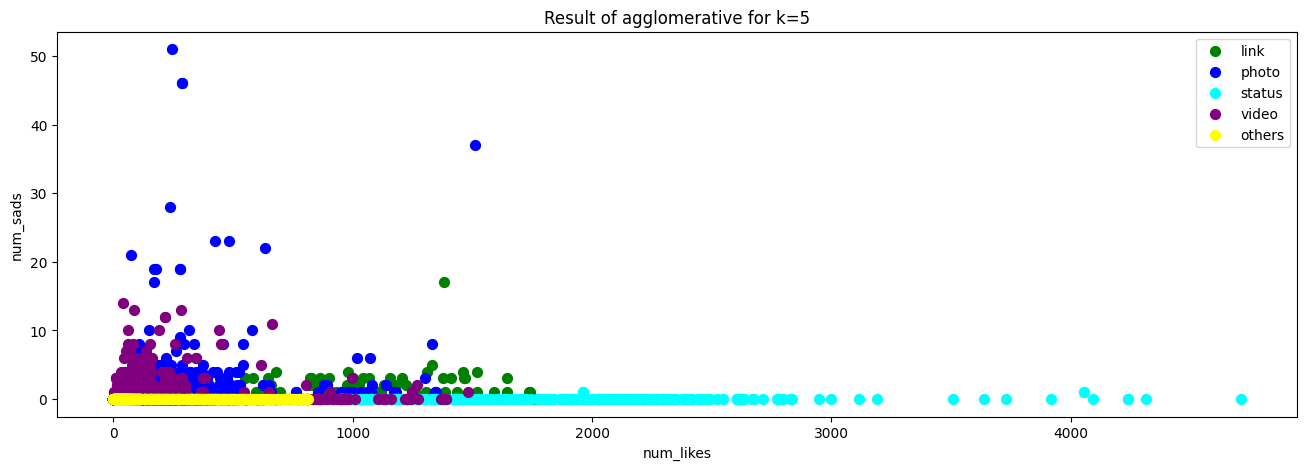


图 9 Agglomerative模型聚类结果

最终给出多个聚类算法结果的对比效果图如下：

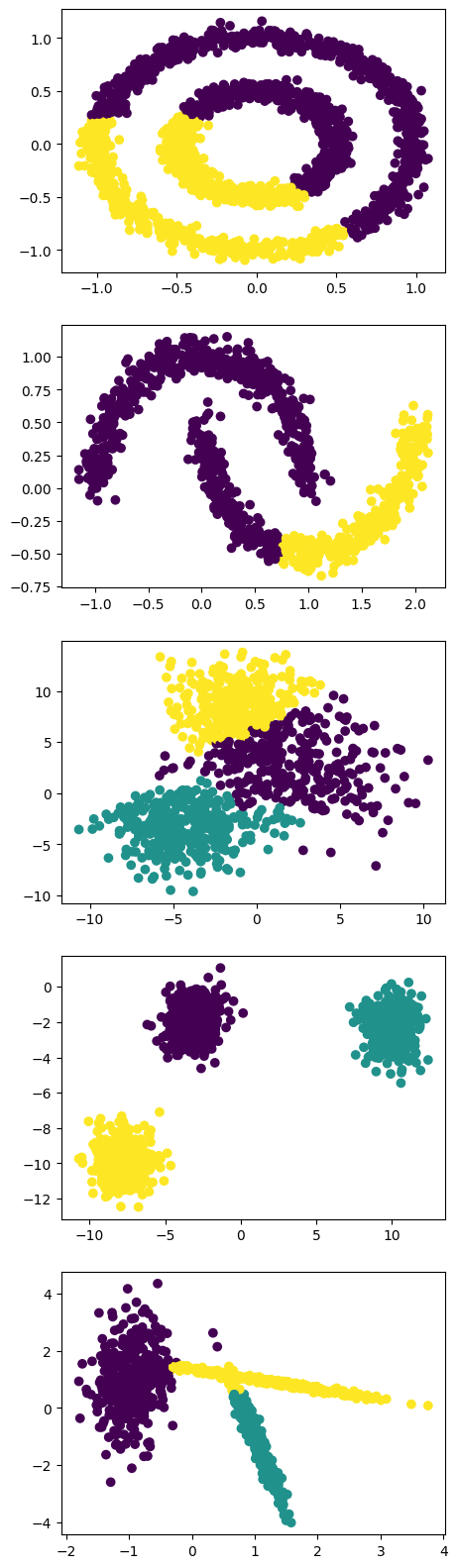
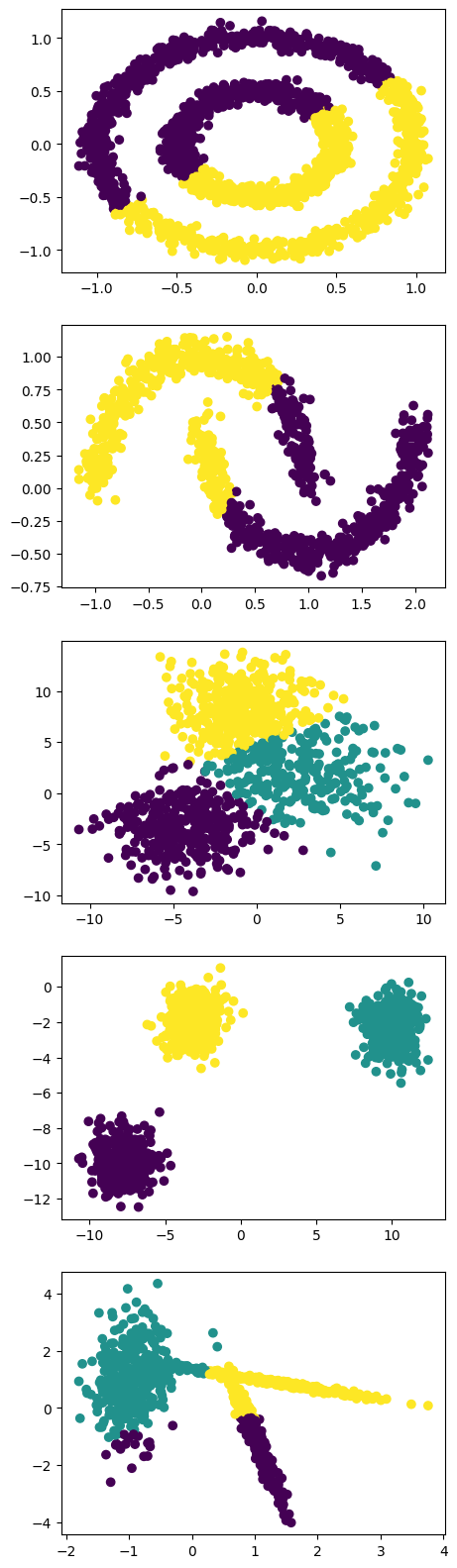
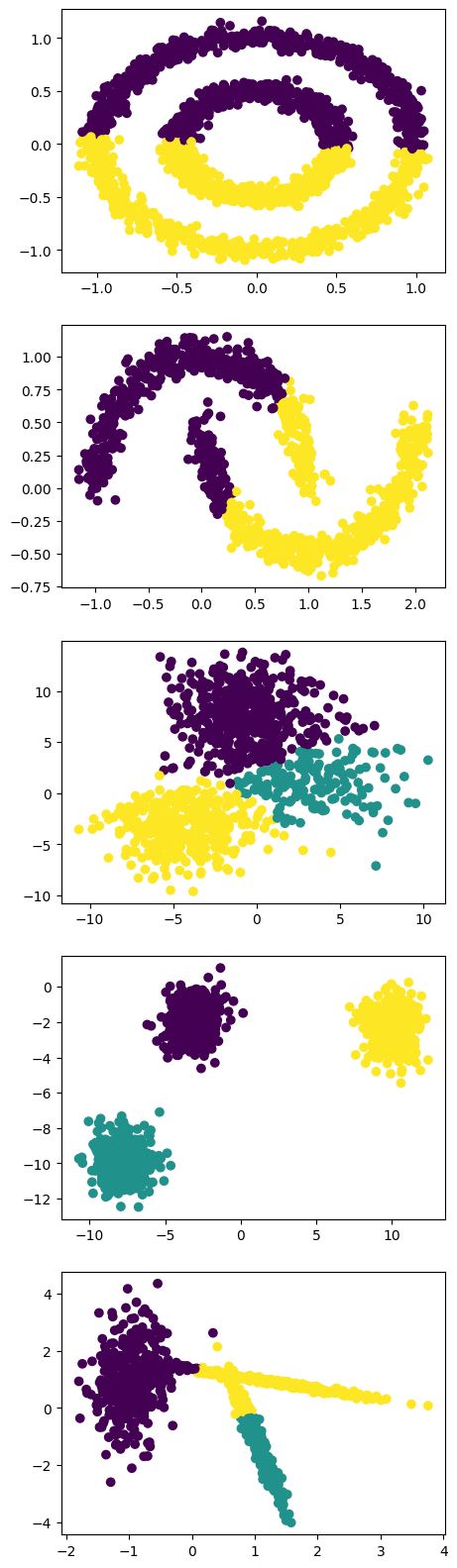
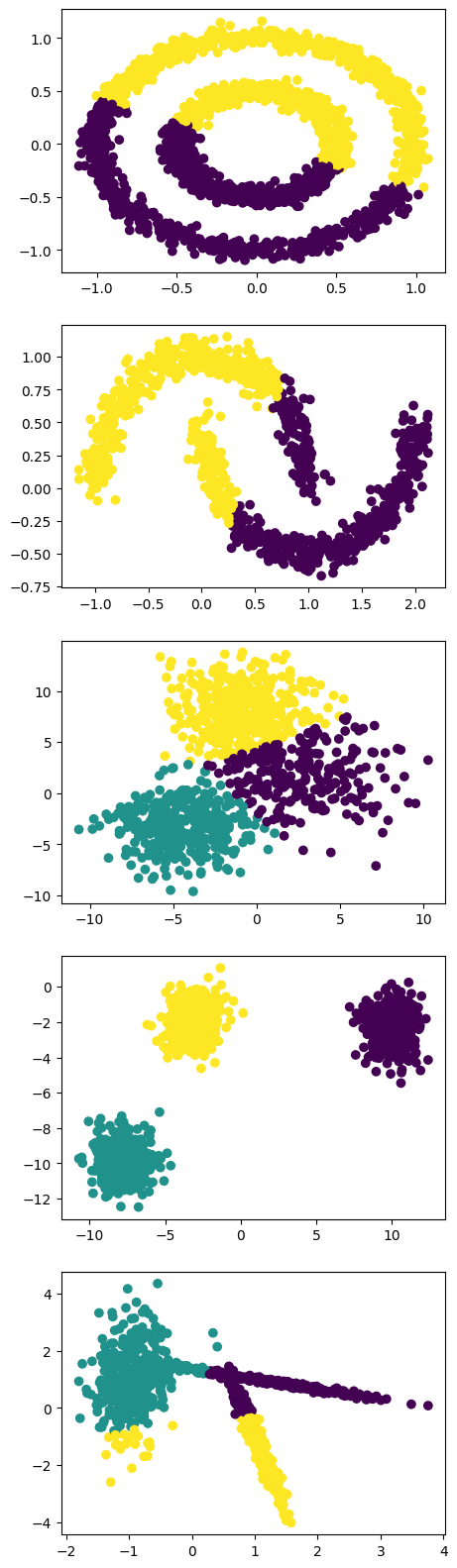


图 10 KMeans、spectral clustering、minibatch KMeans、agglomerative聚类结果

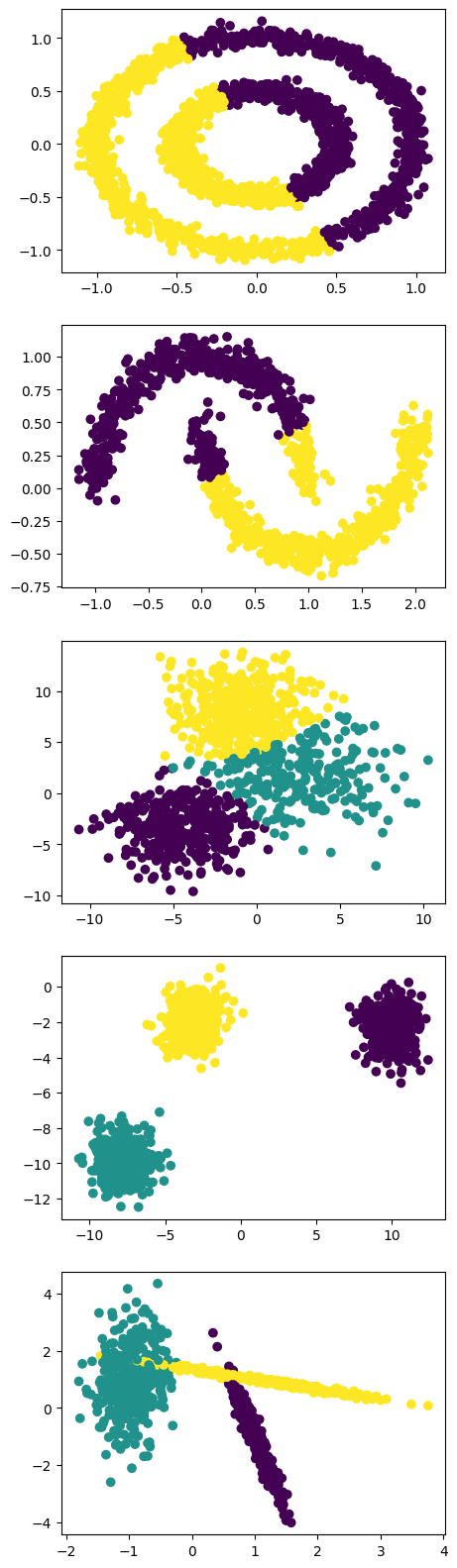
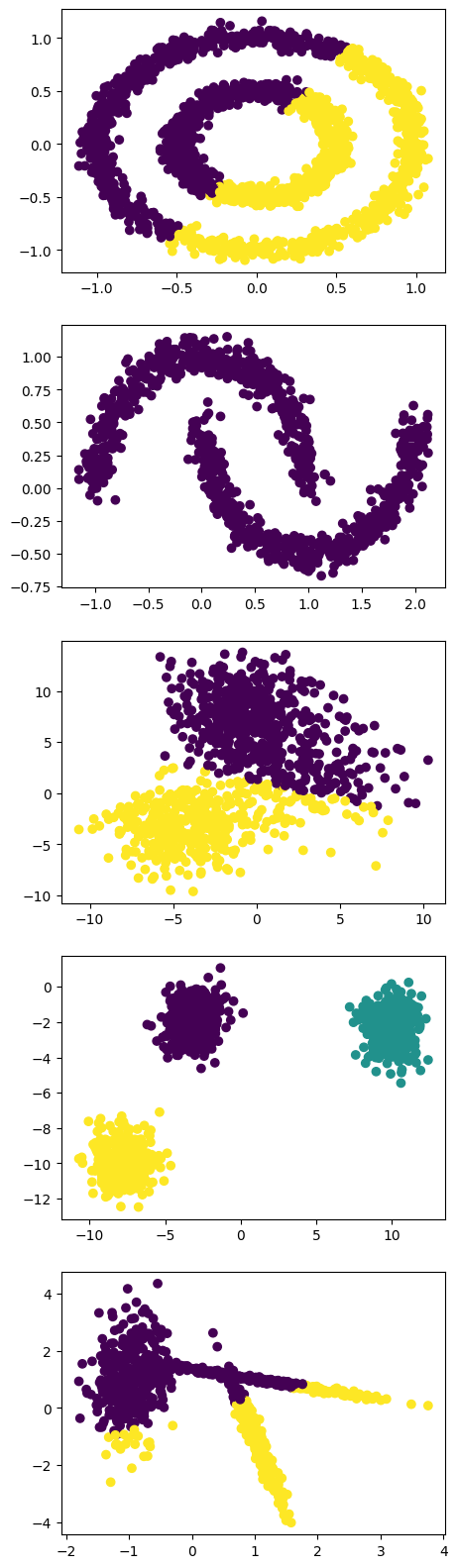
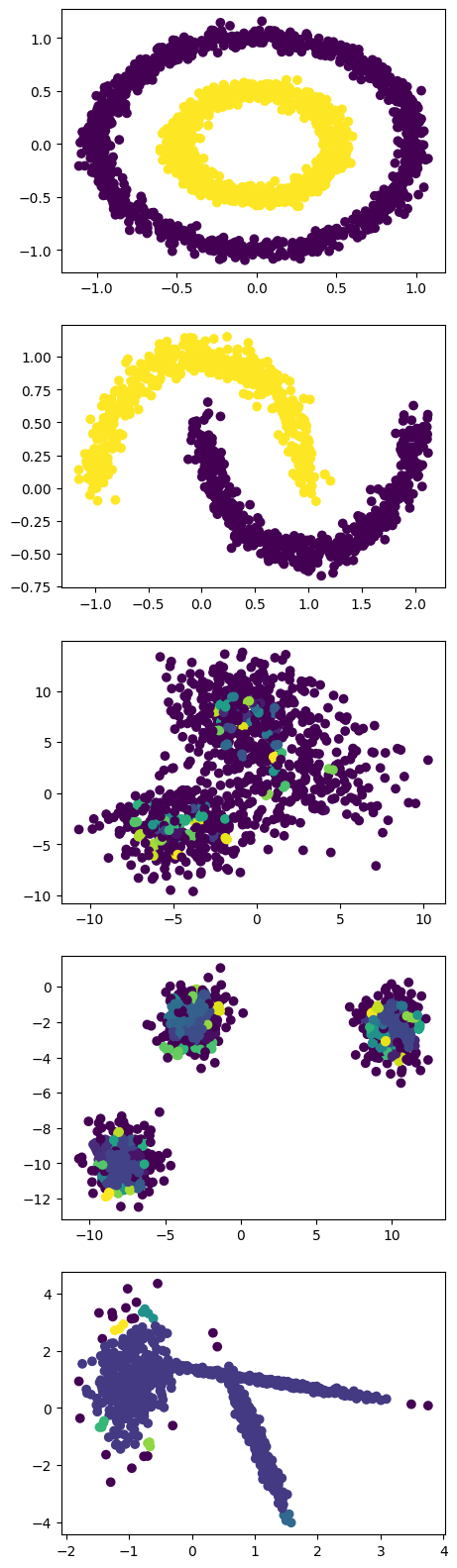
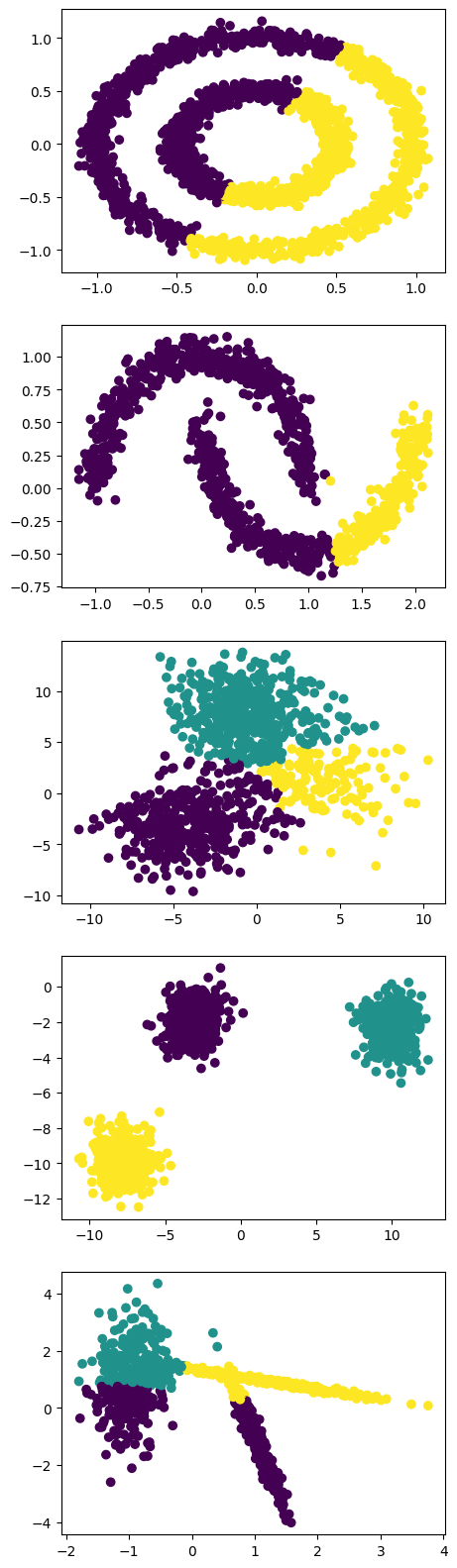


图 birch、dbscan、meanshift、gaussianmixture聚类结果

1. **模型优化**

Agglomerative聚类(层次聚类)是一种自下而上的层次聚类方法,其聚类过程如下: