西安電子科技大學
XIDIAN UNIVERSITY

人工智能学院

智能数据挖掘课程作业报告

UCI Musk 数据集特征提取分析报告

姓名：杨文韬

学号：18020100245

班级：1920012

2022 年 03 月 27 日

目录

UCI Musk 数据集特征提取分析报告

1 Musk 数据集简介

Musk (麝香) 数据集 (版本 1) 有 476 行 169 列，共 166 个属性 2 类标签。该数据集描述了一组 92 个分子，其中 47 个被人类专家判断为麝香，其余 45 个分子被判断为非麝香。目标是学习预测新分子是麝香还是非麝香。然而，描述这些分子的 166 个特征取决于分子的确切形状或构象。因为键可以旋转，单个分子可以采用许多不同的形状。为了生成这个数据集，生成了分子的低能构象，然后过滤以去除高度相似的构象。这留下了 476 个构象。然后，提取描述每个构象的特征向量。

特征向量和分子之间的这种多对一关系称为“多实例问题”。在为该数据学习分类器时，如果分子的任何构象被分类为麝香，分类器应将其分类为“麝香”。如果分子的构象中没有一个被归类为麝香，则该分子应归类为“非麝香”。各列说明如下：

- molecule_name (列 1)：每个分子的符号名称。麝香有诸如 MUSK-188 之类的名称。非麝香具有诸如 NON-MUSK-jp13 之类的名称。
- onformation_name (列 2)：每个构象的符号名称。它们的格式为 MOL_ISO+CONF，其中 MOL 是分子编号，ISO 是立体异构体编号（通常为 1），CONF 是构象编号。
- f1 到 f162：这些是沿射线的“距离特征”。距离以百分之几埃为单位。距离可以是负数或正数，因为它们实际上是相对于沿每条射线放置的原点测量的。起源由不再使用的“共识麝香”表面定义。因此，任何对数据的实验都应将这些特征值视为位于任意连续尺度上。特别是，算法不应使用零点或每个特征值的符号。
- f163：这是分子中的氧原子到 3 空间中指定点的距离。这也称为 OXY-DIS。
- f164：OXY-X：从指定点开始的 X 位移。
- f165：OXY-Y：从指定点开始的 Y 轴位移。
- f166：OXY-Z：从指定点开始的 Z 位移。
- class: 0 ⇒ non-musk, 1 ⇒ musk。

2 原理分析

2.1 数理统计知识

设随机变量 X 期望为 $\mathrm{E}[X] = \mu$，计算方差如下

$$\mathrm{Var}(X) = \sigma^2 = E[(X - \mu)^2] \tag{1}$$

上式计算需要知道 X 具体分布，实际采样用 S^2 近似 σ^2

$$S^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2 \tag{2}$$

但现实中往往 μ 也不清楚，只知道样本均值 $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$，总体样本估计值为

$$\sigma_X^2 = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2 \tag{3}$$

利用结论 $\mathrm{Var}(X) = \mathrm{E}[X^2] - \mathrm{E}[X]^2$，计算期望如下

$$\begin{aligned}
\mathrm{E}[\sigma_X^2] &= \mathrm{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathrm{E}\left[X_i^2 - \frac{2}{n}X_i\sum_{j=1}^{n}X_j + \frac{1}{n^2}\sum_{j=1}^{n}X_j\sum_{k=1}^{n}X_k\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{n-2}{n}\mathrm{E}[X_i^2] - \frac{2}{n}\sum_{j\neq i}\mathrm{E}[X_iX_j] + \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k\neq j}E[X_jX_k] + \frac{1}{n^2}\sum_{j=1}^{n}E[X_j^2]\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\left[\frac{n-2}{n}(\sigma^2+\mu^2) - \frac{2}{n}(n-1)\mu^2 + \frac{1}{n^2}n(n-1)\mu^2 + \frac{1}{n}(\sigma^2+\mu^2)\right] \\
&= \frac{n-1}{n}\sigma^2
\end{aligned} \tag{4}$$

为了避免偏差，采用修正值

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2 \tag{5}$$

2.2 特征值分解

2.2.1 特征值与特征向量

若向量 v 是矩阵 $\boldsymbol{A}$ 的特征向量，一定可以表示成如下的形式：

$$\boldsymbol{A}v = \lambda v \Rightarrow (\lambda\boldsymbol{E} - \boldsymbol{A})v = 0 \tag{6}$$

其中，λ 是特征向量 v 对应的特征值，矩阵的一组特征向量是一组正交向量。

2.2.2 特征值分解

对于矩阵 $\boldsymbol{A}$，有一组特征向量 v，将这组向量进行正交化单位化，就能得到一组正交单位向量。特征值分解，就是将矩阵 $\boldsymbol{A}$ 分解为如下式：

$$\boldsymbol{A} = Q\Lambda Q^{-1} \tag{7}$$

其中，Q 是矩阵 $\boldsymbol{A}$ 的特征向量组成的矩阵，Λ 是对角阵，对角线上的元素就是特征值。

2.3 PCA 降维

主成分分析 (Principal components analysis，PCA) 是一种常用的降维方法，主要思想是将 n 维特征映射到 k 维上，k 维正交新特征被称为主成分。PCA 在原始空间中找一组相互正交的坐标轴，其中，第一个新坐标轴选择是原始数据中方差最大的方向，第二个新坐标轴选取是与第一个坐标轴正交的平面中使得方差最大的，第三个轴是与第 $1,2$ 个轴正交的平面中方差最大的，以此类推得到 n 个这样的坐标轴，部分方差都包含在前面 k 个坐标轴中，后面的坐标轴所含的方差几乎为 0，故可以忽略余下的坐标轴，只保留前面 k 个含有绝大部分方差的坐标轴，实现对数据特征的降维处理。

Algorithm 1 PCA 算法

Input: 样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$，低维空间维数 k

Output: 投影矩阵 P 和新空间中的数据表示

1: 所有样本中心化: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$

2: 计算样本协方差矩阵 $\frac{1}{n}XX^T$

3: 对协方差矩阵 $\frac{1}{n}XX^T$ 做特征值分解得到特征值与特征向量

4: 特征值从大到小排序，取最大的 k 个，对应的 k 个特征向量作为行向量组成投影矩阵 P

5: 将数据据转换到新空间中 $Y = PX$

2.4 SVD 降维

奇异值分解 (Singular value decomposition，SVD) 是线性代数中一种重要的矩阵分解，适用于任意矩阵。对于任意矩阵 $\boldsymbol{A}$ 总是存在一个奇异值分解：

$$\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^T \tag{8}$$

假设 $\boldsymbol{A}$ 是一个 $m \times n$ 阶矩阵，那么得到的 $\boldsymbol{U}$ 是一个 $m \times m$ 阶方阵，$\boldsymbol{U}$ 里面的正交向量被称为左奇异向量。Σ 是一个 $m \times n$ 阶矩阵，Σ 除了对角线其它元素都为 0，对角线上的元素称为奇异值。$\boldsymbol{V}$ 是 $\boldsymbol{V}$ 的转置矩阵，是一个 $n \times n$ 阶矩阵，它里面的正交向量被称为右奇异值向量。一般而言，我们将 Σ 上的值按从大到小的顺序排列。对矩阵 $\boldsymbol{A}$ 进行 SVD 分解的步骤如下：

***Step* 1.** 计算 $\boldsymbol{A}\boldsymbol{A}^T$ 的特征值和特征向量，用单位化的特征向量构成 $\boldsymbol{U}$。

***Step* 2.** 计算 $\boldsymbol{A}^T\boldsymbol{A}$ 的特征值和特征向量，用单位化的特征向量构成 $\boldsymbol{V}$。

***Step* 3.** 计算 $\boldsymbol{A}\boldsymbol{A}^T$(or $\boldsymbol{A}^T\boldsymbol{A}$) 的特征值求平方根，然后构成 Σ。

经过 SVD 分解后，可以用前 k 个非零奇异值对应的奇异向量表示矩阵 A 的主要特征，这样就把矩阵 A 进行了降维。

3 实验过程

3.1 PCA 降维

导入数据集和预处理后，为了合理选择主成分个数，我利用特征值分解的 PCA 对主成分的贡献率和贡献增长率进行分析，如图 1 所示，红色曲线表示总贡献率，蓝色曲线表示贡献增长率，可见 10 个主成分后贡献增长率接近于 0，且 10 个主成分共享率为 83%，故可选择 10 个主成分进行后续实验。

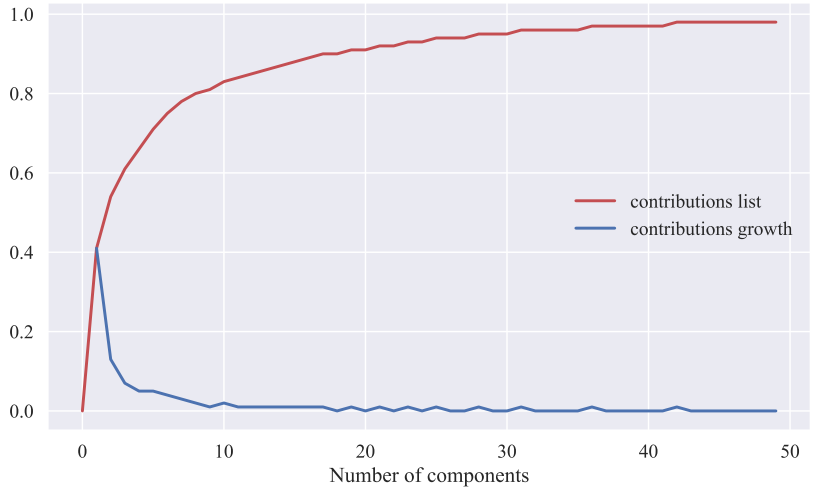


图 1: 主成分贡献率和贡献增长率曲线

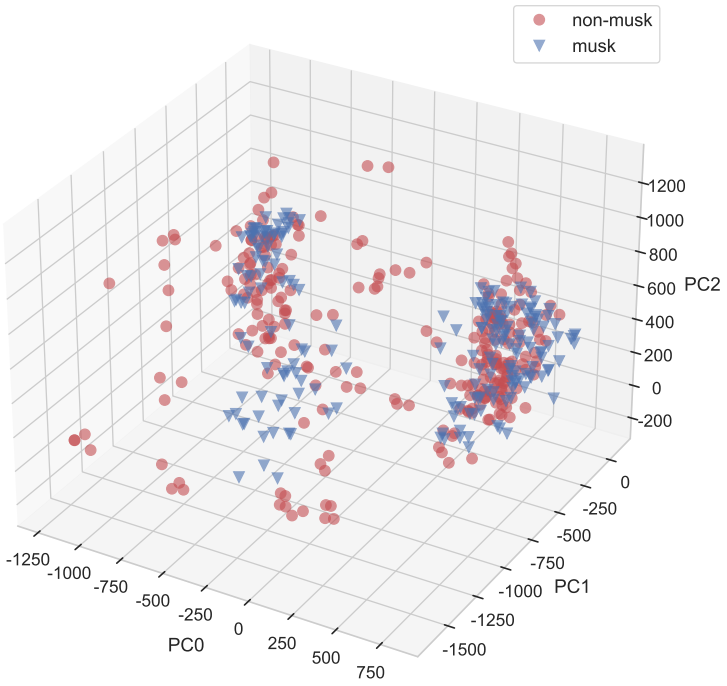


图 2: 前 3 个主成分散点图

10 个主成分降维获得的特征值为 [453240 144754 77811 62336 49078 46722 30678 23609 18173 17244]，由于特征向量和降维后的数据太长，具体结果见附录后的代码和结果。之后取前 3 个主成分绘制散点图进行可视化观察分析，如图 2 所示，由之前分析知前三个主成分只有不到 50% 贡献率，故不能很好分类。

现在绘制获得新数据的盒图 (boxplot)，一个盒图的示例如图 3 所示，可以告诉我们异常值有哪些、数据是否对称、数据分组的紧密程度以及数据是否和怎样倾斜。它由五个数值点组成：最小值 (min)，下四分位数 (Q1)，中位数 (median)，上四分位数 (Q3)，最大值 (max)。也可以往盒图里面加入平均值 (mean)。

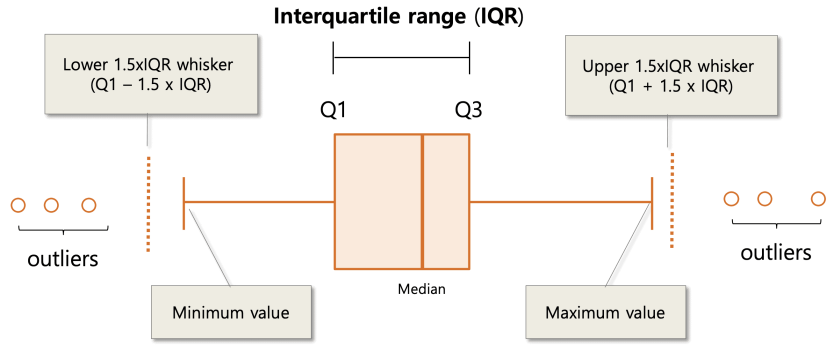


图 3: 盒图示例

利用 PCA 降维获得的新数据绘制的盒图如图 4 所示，绿色虚线表示平均值。

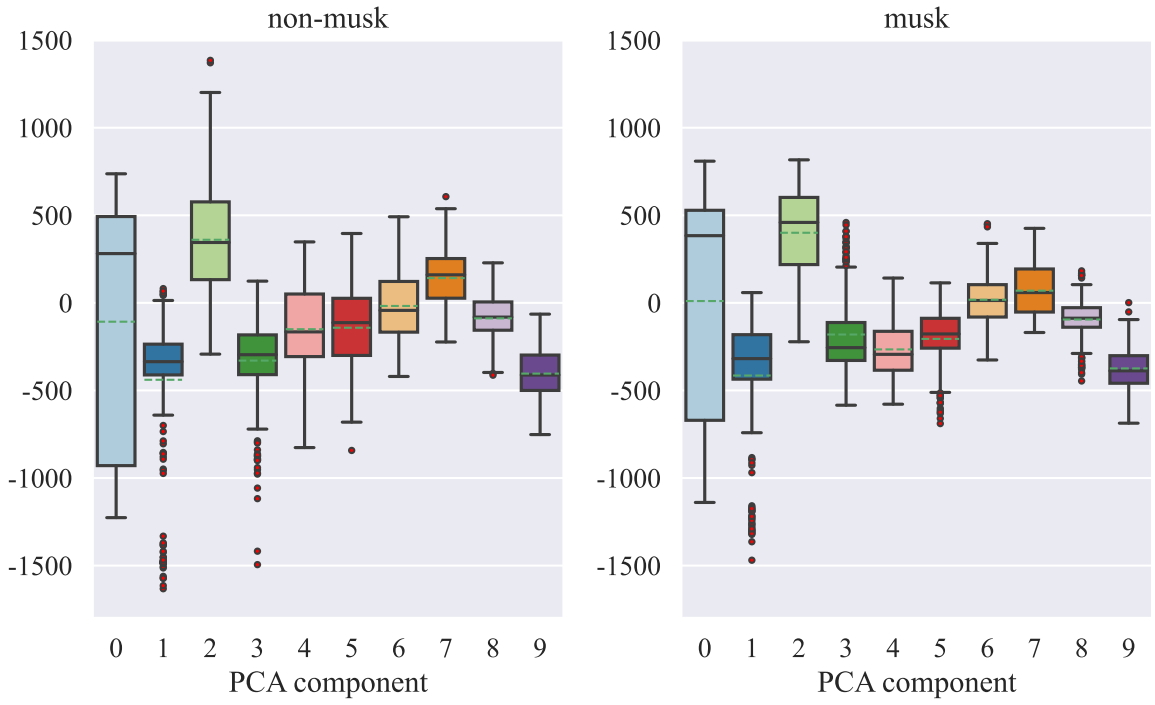


图 4: PCA 降维后新数据盒图

3.2 SVD 降维

取 k 个奇异值使得占总奇异值的 90%，可以进行压缩存储，计算得 $k = 18$。利用 SVD 降维获得的新数据绘制的盒图如图 5 所示，绿色虚线表示平均值。

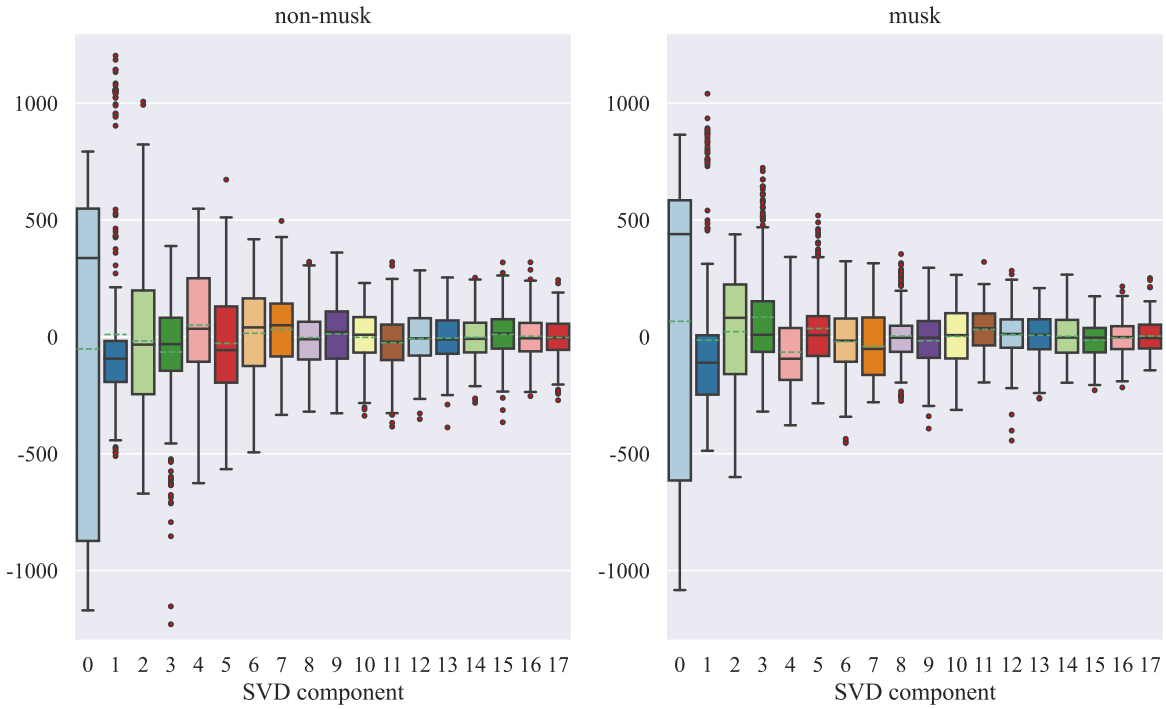


图 5: SVD 降维后新数据盒图

4 总结

通过此次实验，我回顾了数理统计中样本方差以及 PCA、SVD 降维的相关知识，掌握了盒图的概念和 Python 中盒图的画法。具体 Jupyter Notebook 代码和结果见附录。

5 附录

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
sns.set_style("darkgrid", {"grid.color": ".6", "grid.linestyle": ":"})
sns.set_theme(font='Times New Roman', font_scale=1.2)
plt.rc("figure", autolayout=True)
# Chinese support
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

In [3]:
```
from sklearn import preprocessing
def norm(X):
    min_max_scaler = preprocessing.MinMaxScaler()
    X = min_max_scaler.fit_transform(X)
    return X
def standartize(X):
    scaler = preprocessing.StandardScaler().fit(X)
    X = scaler.transform(X)
    return X
```

In [4]:
```
X = pd.read_csv('./clean1.data', header=None)
print(X.shape)
X.head()
```

```
(476, 169)
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9	...	159	160	\
0	MUSK-188	188_1+1	42	-198	-109	-75	-117	11	23	-88	...	-74	-129	
1	MUSK-188	188_1+2	42	-191	-142	-65	-117	55	49	-170	...	-302	60	
2	MUSK-188	188_1+3	42	-191	-142	-75	-117	11	49	-161	...	-73	-127	
3	MUSK-188	188_1+4	42	-198	-110	-65	-117	55	23	-95	...	-302	60	
4	MUSK-190	190_1+1	42	-198	-102	-75	-117	10	24	-87	...	-73	-127	

```
   161  162  163  164  165  166  167  168
0 -120  -38   30   48  -37    6   30  1.0
1 -120  -39   31   48  -37    5   30  1.0
2 -120  -38   30   48  -37    5   31  1.0
3 -120  -39   30   48  -37    6   30  1.0
4   51  128  144   43  -30   14   26  1.0

[5 rows x 169 columns]
```

```
In [5]: y = X[168].values
        X.drop([0, 1, 168], axis=1, inplace=True)
        data = X.values
        print(data.shape)
        data
```

```
(476, 166)
```

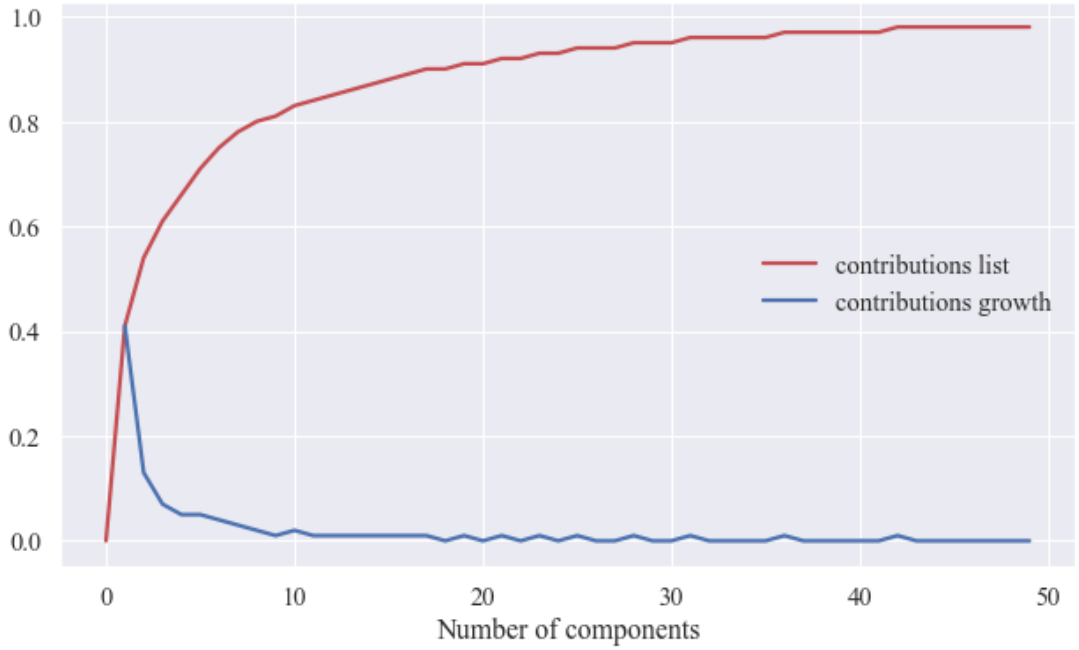
```
Out[5]: array([[  42, -198, -109, ...,  -37,    6,   30],
               [  42, -191, -142, ...,  -37,    5,   30],
               [  42, -191, -142, ...,  -37,    5,   31],
               ...,
               [  43, -102,  -20, ...,  -37,  -19,  -36],
               [  39,  -58,   27, ...,  -28,    3,   74],
               [  52, -121,  -24, ...,  -14,   12,   96]], dtype=int64)
```

5.1 PCA 降维

```
In [6]: def mypca(X, k):
            X = X - np.mean(X, 0)
            cov_mat = np.cov(X, rowvar=False)
            # column: variable, row: observations (num is N, div N-1)
            eigval, eigvec = np.linalg.eig(cov_mat)
            indices = np.argsort(eigval)
            eigval_k = eigval[indices[:-k-1:-1]]
            eigvec_k = eigvec[:, indices[:-k-1:-1]]
            contri = round(float(sum(eigval_k)/sum(eigval)), 2)
            return eigval_k, eigvec_k, contri
```

```
def contri_plot(X, maxk):
    contri_list = []
    for n in range(maxk):
        eigval_k, eigvec_k, contri = mypca(X, n)
        contri_list.append(contri)
    contri_growth = [round(contri_list[i] - contri_list[i-1], 2)
                     for i in range(1, maxk)]
    plt.figure(figsize=(8, 5), dpi=80)
    plt.plot(range(maxk), contri_list, 'r',
             lw=2.0, label='contributions list')
    plt.plot(range(1, maxk), contri_growth, 'b',
             lw=2.0, label='contributions growth')
    plt.legend(loc='best', frameon=False)
    plt.xlabel('Number of components')
    plt.savefig('./document/figure/contributions.pdf')
    plt.show()
    return contri_list, contri_growth
```

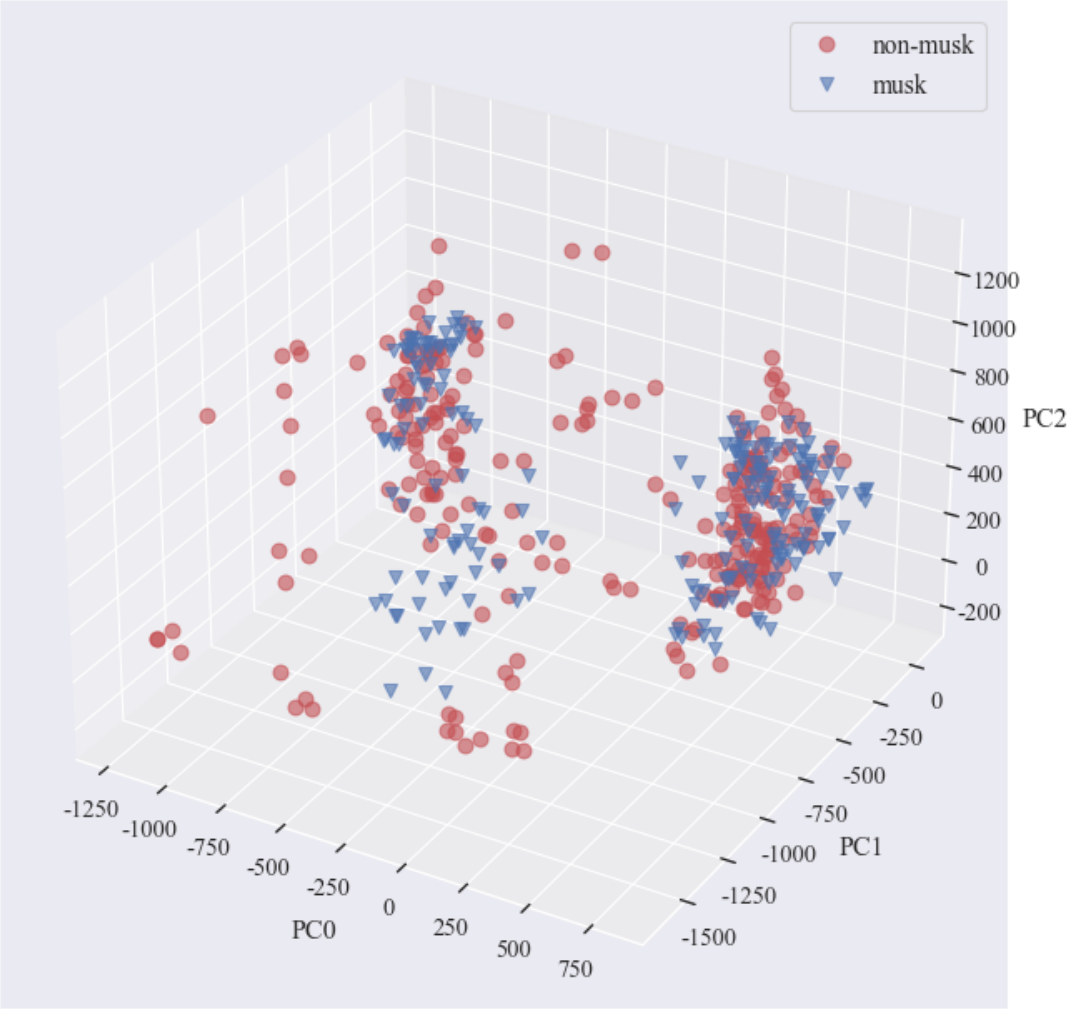
In [7]:
```
contri_list, contri_growth = contri_plot(data, 50)
```



```
In [8]: eigval_k, eigvec_k, contri = mypca(data, 10)
        print("dimension k:", len(eigval_k))
        print("eigenvalues:\n", eigval_k,
              "\neigenvectors:\n", eigvec_k,
              "\ncontributions\n", contri)
        newX = np.dot(data, eigvec_k)
        print(newX)

dimension k: 10
eigenvalues:
 [453239.58247622 144753.81184437  77810.6275262   62335.68439338
  49078.05359872  46721.80964996  30678.16920381  23608.91285056
  18173.24933061  17244.43944881]
eigenvectors:
 [[-6.07485616e-03 -5.92762401e-04 -2.15462256e-02 ...  4.79464045e-02
   6.02582645e-04 -6.07472613e-03]
 [-5.91429272e-02 -2.37599858e-02 -2.07846004e-01 ... -8.09154155e-02
  -2.62452795e-02  9.31827209e-02]
 [-6.62868505e-02 -1.79219018e-03 -1.30939223e-01 ... -7.71129850e-02
  -5.88707902e-03  4.99696908e-02]
 ...
 [-8.49513583e-06  5.05151914e-03 -3.66054356e-04 ... -2.79065762e-02
   1.85485732e-02 -5.89761968e-03]
 [-1.59190184e-03  1.69781919e-02 -1.94359625e-02 ...  7.22524663e-03
   3.58845327e-02 -5.20493901e-02]
 [ 7.30081669e-03 -7.01984258e-03 -5.28802497e-02 ... -9.06630591e-02
   5.94946628e-02  3.38688688e-02]]
contributions
 0.83
[[ -174.76730063 -1178.51423646   592.11244167 ...   -92.3832692
    -49.03502241  -236.22487273]
 [ -189.64669851 -1185.56277833   565.23276113 ...   -96.31026639
    -21.19092515  -609.77025951]
 [ -108.96473757 -1158.56331151   544.50822986 ...   -77.69383271
    -57.8003762   -342.94097366]
 ...
 [-1071.14439159  -222.4524703    641.07032159 ...   304.95168691
```

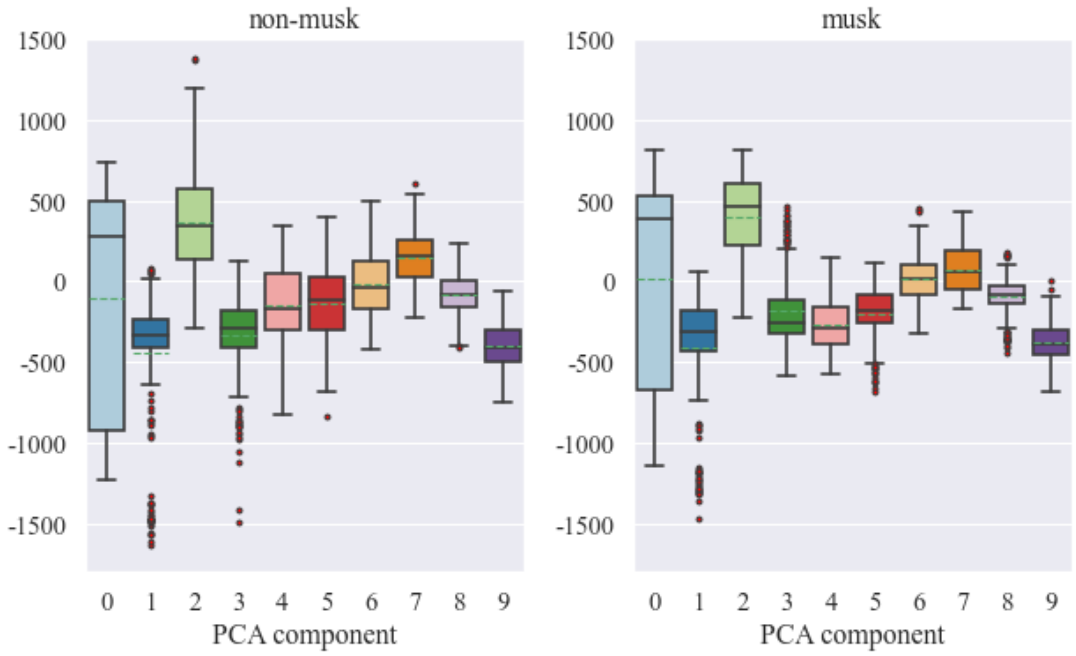
```
    32.17841203  -203.86071292]
 [  404.98082914  -422.4923334   -222.27143215 ...    17.70346026
   -18.59353185  -431.15173206]
 [-1068.17978578  -237.73340742   417.19897434 ...   186.43770452
   185.84667101  -166.18510337]]
```

```
In [9]: def plot_3D(X, y):
            rx, ry, rz=[], [], []
            bx, by, bz=[], [], []
            for i in range(len(X)):
                if y[i] ==0:
                    rx.append(X[i][0])
                    ry.append(X[i][1])
                    rz.append(X[i][2])
                else:
                    bx.append(X[i][0])
                    by.append(X[i][1])
                    bz.append(X[i][2])
            fig = plt.figure(figsize=(9, 8), dpi=80)
            ax = fig.add_subplot(111, projection='3d')
            ax.plot(rx, ry, rz, 'ro', markersize=8, alpha=0.6, label='non-musk')
            ax.plot(bx, by, bz, 'bv', markersize=8, alpha=0.6, label='musk')
            ax.set_xlabel('PC0', labelpad=10)
            ax.set_ylabel('PC1', labelpad=10)
            ax.set_zlabel('PC2', labelpad=10)
            plt.legend(loc='best', prop={'size':14},
                       bbox_to_anchor=(0.99, 0.99))
            #plt.savefig('./document/figure/test.pdf')
            plt.show()

        plot_3D(newX, y)
```




```
In [10]: df = pd.DataFrame(newX)
         df['tag'] = y
         df1 = df[df['tag']==1]
         df0 = df[df['tag']==0]
         df0 = df0.drop(['tag'], axis=1)
         df1 = df1.drop(['tag'], axis=1)

         plt.figure(figsize=(8, 5), dpi=80)
         plt.subplot(1,2,1)
         sns.boxplot(data=df0, palette="Paired",
                     meanline=True, showmeans=True,
```

```
            flierprops={'marker':'.',
                        'markerfacecolor':'red',
                        'color':'black',})
plt.ylim(-1800, 1500)
plt.title('non-musk')
plt.xlabel('PCA component')
plt.subplot(1,2,2)
sns.boxplot(data=df1, palette="Paired",
            meanline=True, showmeans=True,
            flierprops={'marker':'.',
                        'markerfacecolor':'red',
                        'color':'black',})
plt.ylim(-1800, 1500)
plt.title('musk')
plt.xlabel('PCA component')
plt.tight_layout()
#plt.savefig('./document/figure/boxplot_pca.pdf')
plt.show()
```



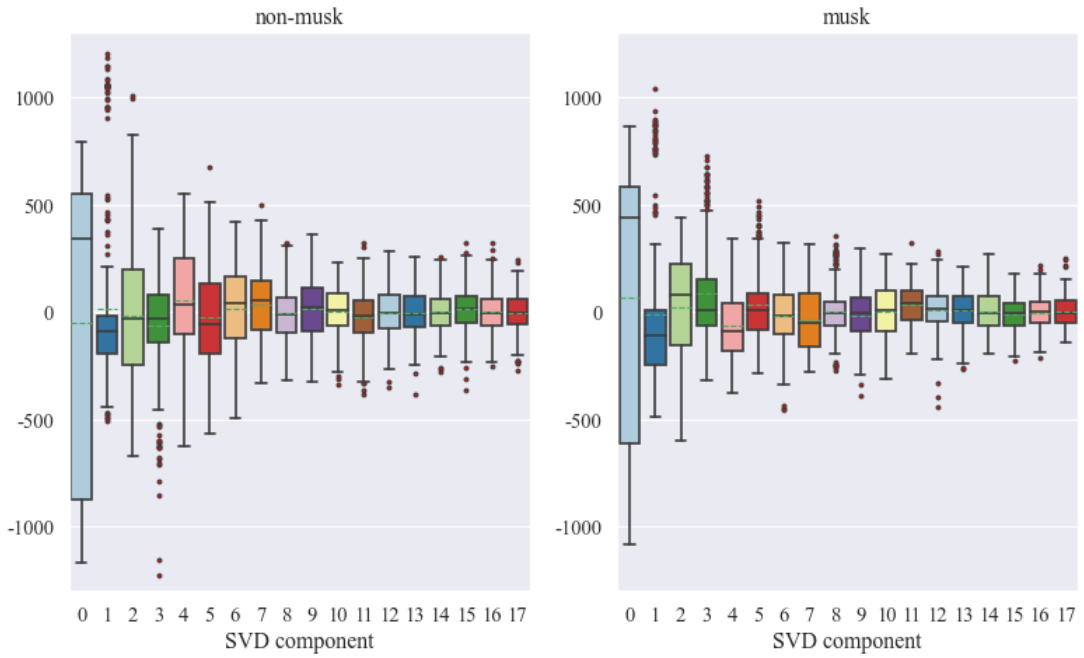
5.2 SVD 降维

```
In [11]: def calk(Sigma, percent):
             k = 0
             total = sum(np.square(Sigma))
             svss = 0 # singular values square sum
             for i in range(np.shape(Sigma)[0]):
                 svss += np.square(Sigma[i])
                 if (svss/total) >= percent:
                     k = i + 1
                     break
             return k

         def SVD_DR(X):
             X = X - np.mean(X, 0)
             u, sigma, v = np.linalg.svd(X[:, :])
             k = calk(sigma, 0.9)
             newX = np.dot(u[:,:k], np.diag(sigma[:k]))
             return k, sigma, newX

         k, sigma, newX = SVD_DR(data)
         newX
```

```
Out[11]: array([[ -118.66637432,   750.09743105,   214.33802756, ...,
                    19.44669189,  -145.26012079,   102.44151991],
                [ -133.54577219,   757.14597293,   187.45834702, ...,
                    21.07132898,     4.07866674,  -132.80492299],
                [  -52.86381125,   730.1465061 ,   166.73381575, ...,
                   -70.62167466,   -71.64807003,    61.13252226],
                ...,
                [-1015.04346527,  -205.96433511,   263.29590748, ...,
                  -113.43285357,  -126.83625301,  -230.18256042],
                [  461.08175546,    -5.924472  ,  -600.04584626, ...,
                   160.97093809,     1.40666757,   -21.51082042],
                [-1012.07885946,  -190.68339799,    39.42456023, ...,
                   -63.4478099 ,   -84.43434068,  -163.80082521]])
```

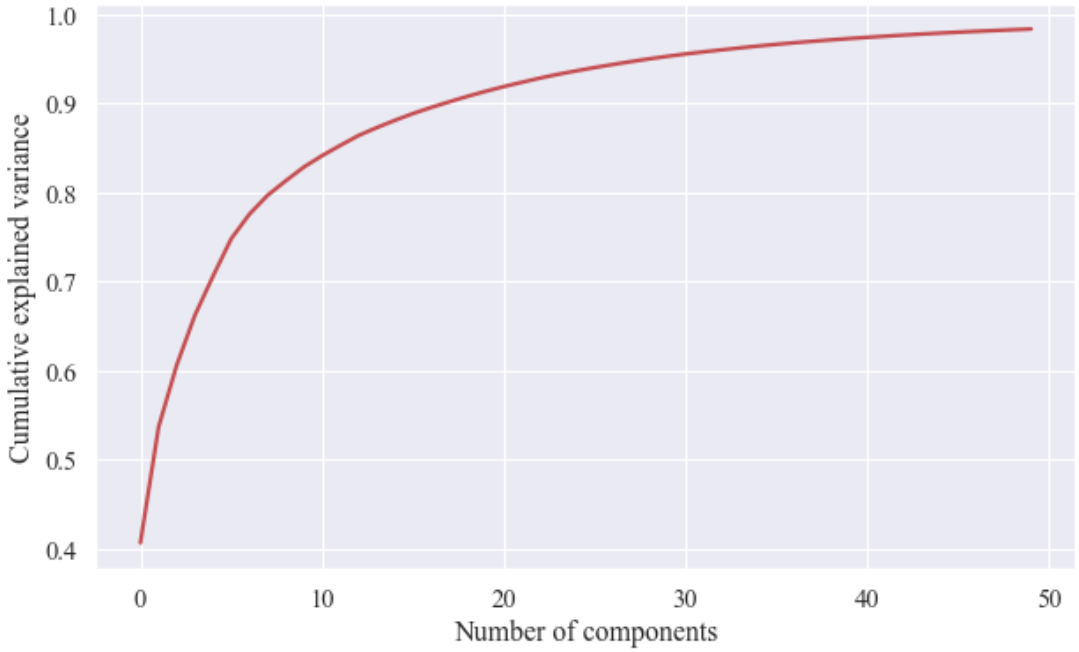
```
In [12]: df = pd.DataFrame(newX)
         df['tag'] = y
```

```
df1 = df[df['tag']==1]
df0 = df[df['tag']==0]
df0 = df0.drop(['tag'], axis=1)
df1 = df1.drop(['tag'], axis=1)

plt.figure(figsize=(10, 6.2), dpi=80)
plt.subplot(1,2,1)
sns.boxplot(data=df0, palette="Paired",
            meanline=True, showmeans=True,
            flierprops={'marker':'.',
                        'markerfacecolor':'red',
                        'color':'black',})
plt.ylim(-1300, 1300)
plt.title('non-musk')
plt.xlabel('SVD component')
plt.subplot(1,2,2)
sns.boxplot(data=df1, palette="Paired",
            meanline=True, showmeans=True,
            flierprops={'marker':'.',
                        'markerfacecolor':'red',
                        'color':'black',})
plt.ylim(-1300, 1300)
plt.title('musk')
plt.xlabel('SVD component')
plt.tight_layout()
#plt.savefig('./document/figure/boxplot_svd.pdf')
plt.show()
```



5.3 sklearn 实现 PCA

```
In [13]: from sklearn.decomposition import PCA
         def PCA_DR(X, k):
             X = X - np.mean(X, 0)
             pca = PCA(n_components=k, whiten=True)
             X = pca.fit_transform(X)
             ev = pca.explained_variance_ratio_
             return newX, ev

         newX, ev = PCA_DR(data, 50)

         plt.figure(figsize=(8, 5), dpi=80)
         plt.plot(np.cumsum(ev), 'r', lw=2.0)
         plt.xlabel('Number of components')
         plt.ylabel('Cumulative explained variance')
         plt.show()
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
Cumulative explained variance
0
10
20
30
40
50
Number of components