机器学习 上机2 乳腺癌聚类

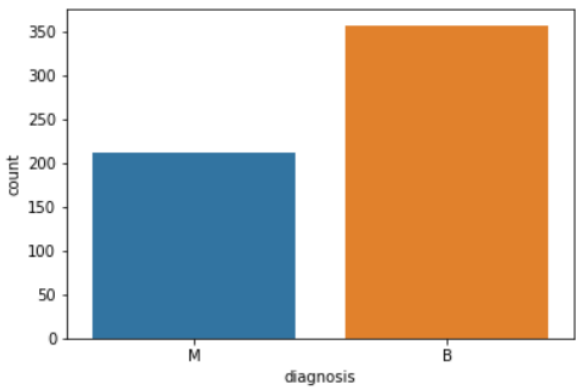
姓名： 张泽群 学号： 19049100002 班级：2班

1. 数据集分析与预处理

1.1. 数据集分析

所用数据集是威斯康辛乳腺癌数据集，具有32维度，数据集除了ID以及癌细胞属性信息，具有**半径、纹理、周界、区域、平滑度、压实度、凹度、凹面点、对称性、分维**，10×3个数据特征。癌细胞属性信息包括恶性和良性，也代表了两个类别，这两个类别在样本数据集中的分布如下图所示： M 恶性 B 良性，可见样本分布并不均衡。

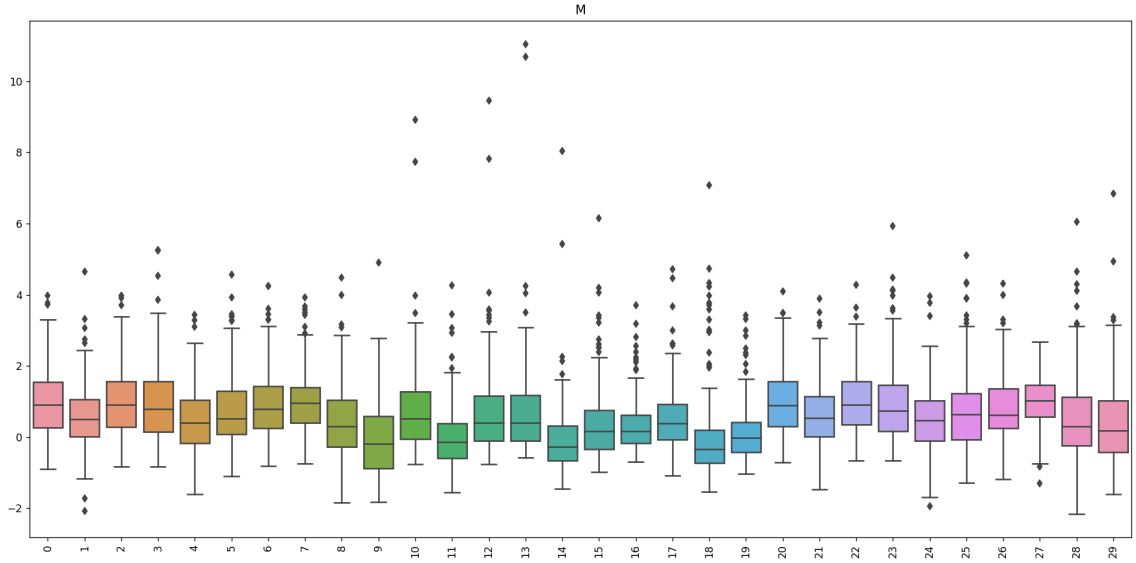
```
Number of Benign       :       357
Number of Malignant    :       212
Percentage Benign      :       62.74 %
Percentage Malignant   :       37.26 %
```

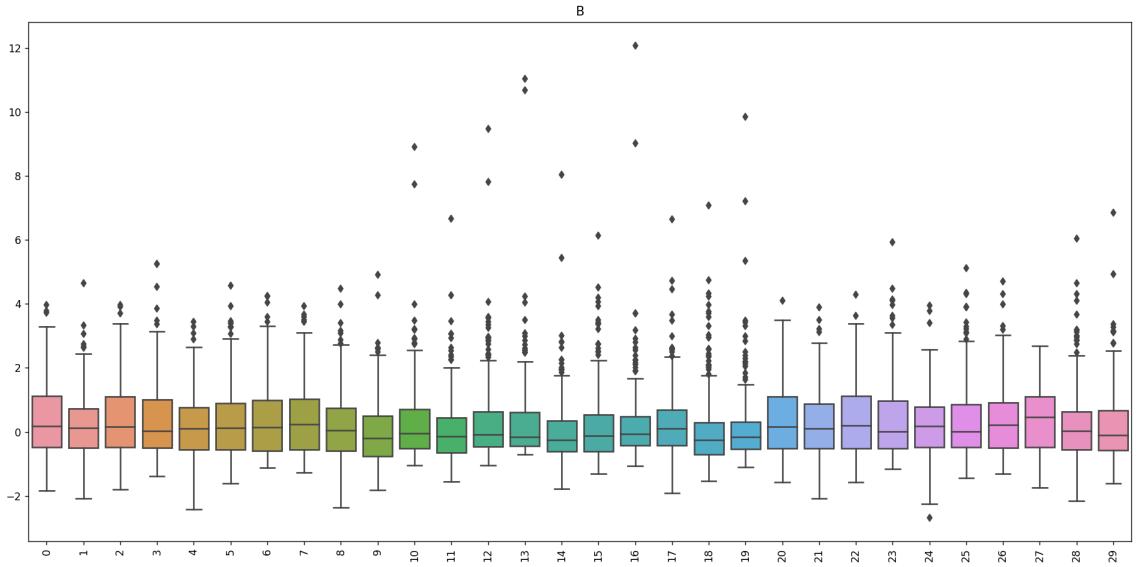


此外，数据集划分为训练、验证、测试集，样本数量分别为341、114、114。

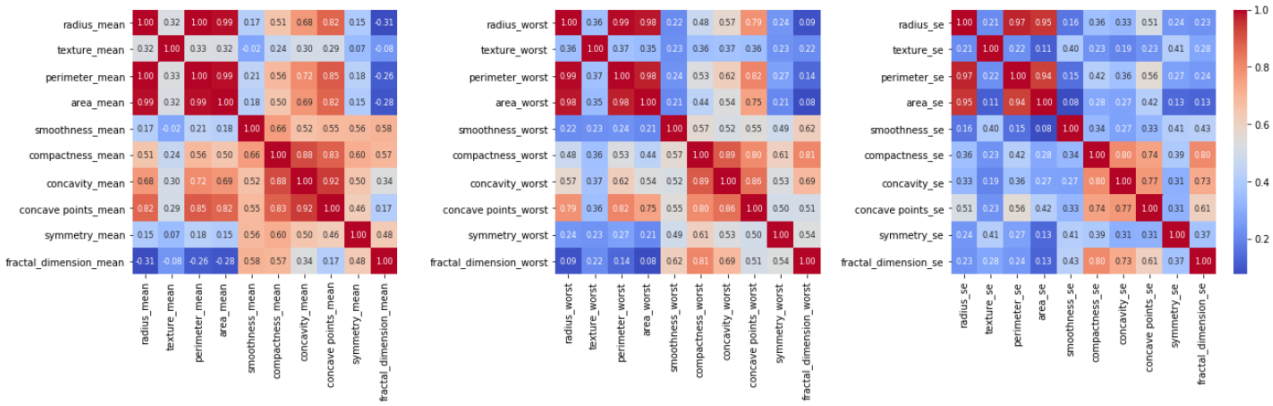
对于样本分布不均衡的数据，一般的模型在训练过程中会更多地关注“多类”，这样容易导致模型在预测过程中偏向于将样本划分在“多类”中。此时，应当对数据进行处理，包括上采样、下采样、加权、数据生成等方法。

对于数据特征，总共有10×3个维度，我们对这三十个特征进行标准化缩放，再绘制箱图，可以看到数据中异常值的分布状况，这些异常值可能表示不正确的数据，可能由于多种原因导致了这些异常，这些异常值或许需要进行处理。





再对数据特征进行相关性热力图分析：



可以看出半径、面积和周长（平均值、最差和误差标准）彼此密切相关，紧凑性、凹点和凹度的特征也是如此，可能可以藉此进行特征选择。

1.2. 数据预处理

1.2.1 处理数据集样本不均匀的问题

对于数据集样本分布不均衡的问题，可以有上采样、下采样、加权、数据生成等方法。

在这里我尝试使用了三种方式，分别为SMOTE方法，RandomUnderSampler方法，集成方法EasyEnsemble处理，分别为过抽样，欠抽样，欠抽样。

```
    # 法1 SMOTE方法进行过采样处理
    model_smote = SMOTE()  # 建立SMOTE模型对象
    train_data_resampled1, train_label_resampled1 =
model_smote.fit_resample(train_data, train_label)  # 输入数据做过抽样处理

    # 法2 RandomUnderSampler方法进行欠抽样处理
    model_RandomUnderSample = RandomUnderSampler()  # 建立RandomUnderSampler模型对象
    train_data_resampled2, train_label_resampled2 =
model_RandomUnderSample.fit_resample(train_data, train_label)  # 输入数据做欠抽样处理


    # 法3 使用集成方法EasyEnsemble处理不均衡样本
    model_EasyEnsemble = EasyEnsembleClassifier()  # 建立EasyEnsemble模型对象
    train_data_resampled3, train_label_resampled3 =
model_RandomUnderSample.fit_resample(train_data, train_label)  # 输入数据并应用集成方法
处理
```

其运行出的新数据集规模和正负样本比例如下所示，实现了样本类别的均衡。

```
Size after SMOTE :  (442, 2)
% : 0.5

Size after RandomUnderSampler :  (240, 2)
% : 0.5

Size after EasyEnsemble :  (240, 2)
% : 0.5
```

此外，通过实验结果验证，我们发现抽样方式对于影响NMI不大，因此任取一种抽样方式都可以解决样本分布不均衡的问题。

1.2.2 特征预处理

对于同一个特征，不同的样本中的取值可能会相差非常大，一些异常小或异常大的数据会误导模型的正确训练；另外，如果数据的分布很分散也会影响训练结果。以上两种方式都体现在方差会非常大。此时，我们可以将特征中的值进行标准差标准化，即转换为均值为0，方差为1的正态分布或者将特征转换到同一个限定范围的区间。

(1) StandardScaler：

标准化缩放的数学原理为：

$$z = \frac{(x - \mu)}{\sigma}$$

```
sclar = StandardScaler()
all_data = sclar.fit_tra
```

(2) MinMaxScaler：

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}}, X_{max}/X_{min}：该column的最大/最小值$$

$$X_{scaled} = X_{std} * (max - min) + min, max/min：指定缩放的上界/下界$$

数学原理：将特征缩放到给定的最小值和最大值之间，这里选择了（0-1）区间

```
all_data = preprocessing.minmax_scale(all_data, feature_range=(0, 1))
```

对于这两种特征预处理的方法，我利用多个模型训练得出NMI得分进行对比，发现第二种方法可以获得更高的NMI，因此在数据预处理阶段选择MinMaxScaler作为特征预处理的手段。

1.2.3 数据降维

太多的变量可能会导致很多问题，包括提高计算机吞吐量，太复杂的可视化问题，由于分析没有影响的变量从而降低效率，使数据解释困难。因此我们需要通过数据降维来简化数据集，同时又不丢失相关信息，本实验中采用了以下技术尝试降维：**PCA、T-SNE、Isomap**

主成分分析（PCA） 基于两个基本考虑：（1）.变量之间的高度相关性表明数据中存在冗余,(2).最重要的变量表示更高的方差。基于这些考虑，该模型简化了变量的复杂性。

T-SNE（T-分布式随机邻居嵌入） 是一种非线性降维技术，特别适合于降低二维或三维模型中多维数据集的复杂性。

Isomap 全称为isometric mapping，称之为等距映射，该算法的本质是通过流形中的测地距离来表示高维空间的距离，然后通过MDS算法进行降维

```
# 法1 T-SNE降维
    tsne = manifold.TSNE(n_components=3, init="pca")
    all_data = pca.fit_transform(all_data)

    # 法2 PCA降维
    pca = decomposition.TruncatedSVD(n_components=2)
    all_data = pca.fit_transform(all_data)

    # 法3 Isomap降维
    isomap = manifold.Isomap(n_neighbors=20, n_components=3)
    all_data = pca.fit_transform(all_data)
```

经过实验验证发现，使用PCA降维，将数据维度降到2维的情况能获得最好的聚类效果。

2. 模型训练

对数据处理情况的调参过程：

情况 ①： 欠抽样（240）+ PCA（3）+ Kmeans，NMI = 0.7158 AC = 0.9385 （107/114）

情况 ②： 欠抽样（240）+ PCA（2）+ Kmeans，NMI = 0.7762 AC = 0.9561 （109/114）

情况 ③： 过抽样（442）+ PCA（2）+ Kmeans，NMI = 0.7762 AC = 0.9561 （109/114）

情况 ④： 欠抽样（442）+ isomap（2-n），NMI = 0.7450 AC = 0.9473 （108/114）

- 最佳参数(局部)： PCA(2) + Kmeans，但tsne（2，pca）也能获得同样效果并且同时提升GMM和DBSCAN的NMI
- 从结果上看抽样方式并不影响NMI。

模型的训练：

此次实验尝试了多种模型，包括Kmeans聚类，GMM聚类，DBSCAN聚类，Agg聚类，由于我们已经知道了需要聚类的簇数，因此没有很多需要调整的超参数，可以直接创建并调用模型。

```python
# Kmeans聚类
model_kmeans = KMeans(n_clusters=2, init="k-means++", n_init=10)
model_kmeans.fit(train_data_resampled2)

# GMM聚类
model_gmm = GaussianMixture(n_components=2)
model_gmm.fit_predict(train_data_resampled2)

# DBSCAN聚类
model_dbscan = DBSCAN(eps=0.1, min_samples=10)
model_dbscan.fit_predict(train_data_resampled2)

# Agg聚类
model_agg = AgglomerativeClustering(n_clusters=2, linkage="ward")
model_agg.fit_predict(train_data_resampled2)
```

3. 实验结果

```
D:\Python37\python.exe F:/python/ML2/cluster.py
Size after SMOTE :  (442, 2)
% : 0.5

Size after RandomUnderSampler :  (240, 2)
% : 0.5

Size after EasyEnsemble :  (240, 2)
% : 0.5

Kmeans NMI 1 =  0.776162136910614
Gmm NMI 2 =  0.5664597570480034
DBSCAN NMI 3 =  0.5664597570480034

Kmeans Purity =  0.956140350877193
Kmeans ARI =  0.5664597570480034

Process finished with exit code 0
```

其中，测试集上的聚类精度为0.9561（109/114），NMI=0.7761，ARI=0.5664。

同时利用joblib导出模型为model_2.model。

```python
# 导出模型
joblib.dump(model_kmeans, 'model_2.model')
```

4. 讨论与结论

4.1 NMI

标准互信息(NMI)是信息论里一种有用的信息度量，它可以看成是一个随机变量中包含的关于另一个随机变量的信息量，或者说是一个随机变量由于已知另一个随机变量而减少的不肯定性。

$$\mathrm{NMI}(U,V)=2\frac{\mathrm{MI}(U,V)}{\mathrm{H}(U)+\mathrm{H}(V)}$$

撇开数学公式的推导，我们利用一个程序来更直观的展现NMI提升的途径。

```
# -*- coding: utf-8 -*-
from sklearn import metrics
import math
import numpy as np
A = np.array([0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,
 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1])
B = np.array([1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1,
 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,
 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1,
 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1])

result_NMI=metrics.normalized_mutual_info_score(A, B)
result_AC=metrics.accuracy_score(A, B)
print("result_NMI:", result_NMI)
print("result_AC:", result_AC)
```

```
113/114
result_NMI: 0.9347885684978764
result_AC: 0.9912280701754386
112/114
result_NMI: 0.870084924635688
result AC: 0.9824561403508771
111/114
result_NMI: 0.822829189942911
result_AC: 0.9736842105263158
110/114
result_NMI: 0.7765360581302789
result_AC: 0.9649122807017544
106/114
result_NMI: 0.6414062531643422
result_AC: 0.9298245614035088
```

结论： 以上是在测试集中模拟计算NMI（标签-预测）的情况，从结果可见聚类的精度和NMI是成非线性正相关的，因此提升NMI的途径即为提升聚类的精度。

4.2 改进的方向

由于我们的模型在测试集上的NMI为0.7761，精度为0.9561（109/114），仍有5个样本的空间可以提升。在经历模型训练包括数据集处理的过程后，我认为能够改进的方向有：

1. 数据集特征选择 2. 选用混合聚类模型，或者其他的利用神经网络的例如SOM模型。