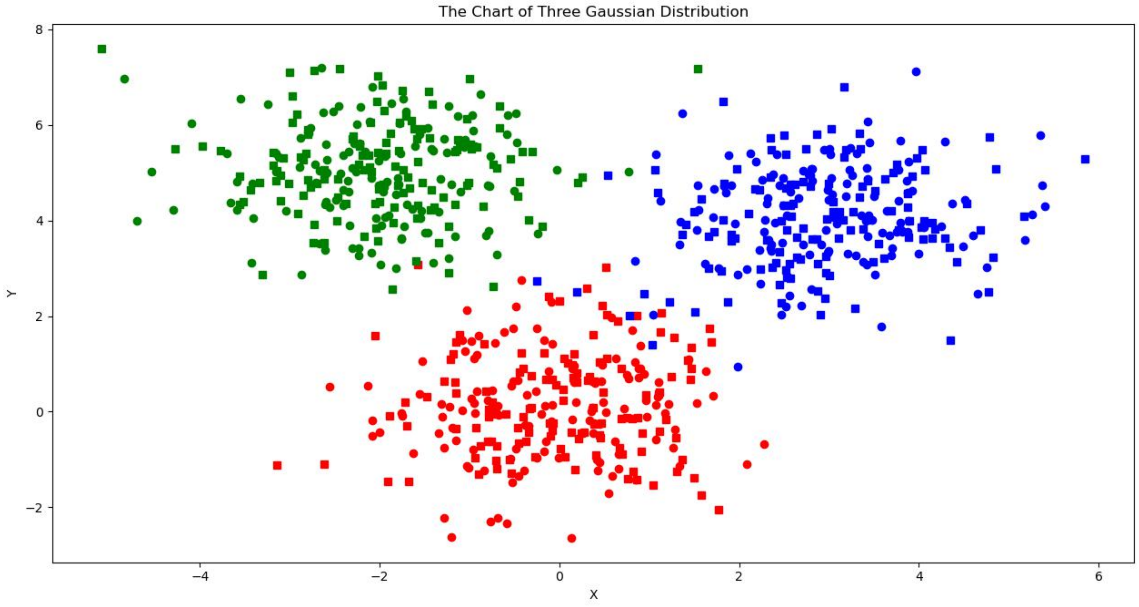
# Assignment 1 的报告

## Part I

高斯分布模型建立的部分用 datacreate( )函数实现，通过 numpy 库中 numpy.random.multivariate_normal( )函数，生成了三组六份均值为 mean1, mean2, mean3，协方差为 cov，数据大小分别为 size 的高斯分布数据。其中每一组中两份数据，一份放置于训练集，一份放置于测试集，数据标签分别为 A(0), B(1), C(2)。

运用 matplotlib.pyplot 库中的函数进行画图，每组数据中（x, y, label）分别代表数据在图中的 x, y 坐标和数据的标签。图中 A、B、C 三组数据的点分别对应红、蓝、绿色，训练集、测试集中的点用圆形和正方形分别标记。

所得到的散点图如下所示：




得到的数据被存入与 source.py 同目录下的 data.data 文件保存。

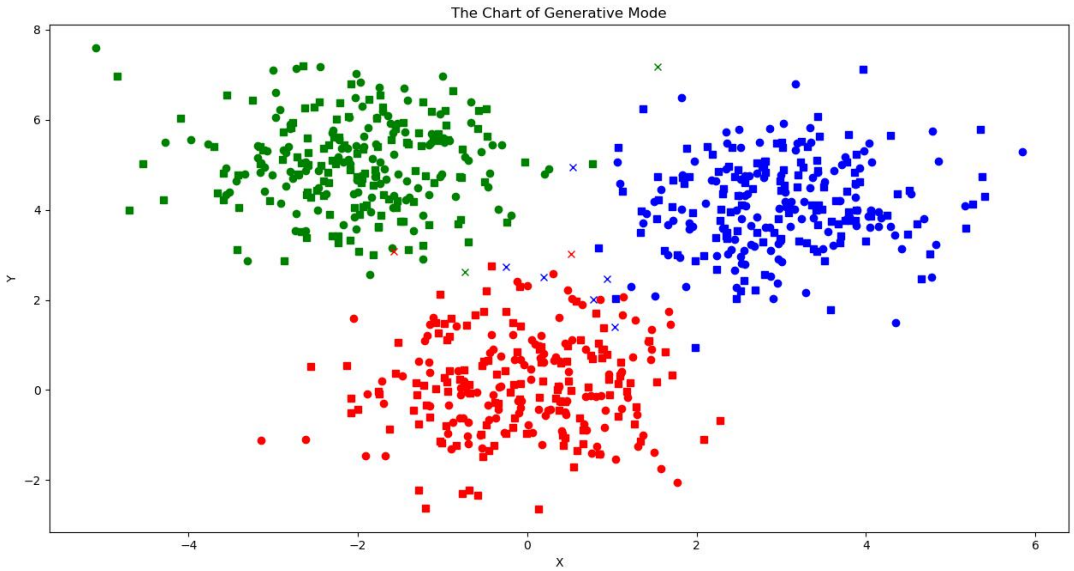
## Part II

程序中使用 readdata( )函数，从 data.data 文件中读取数据。读取到的点与标签分别存入 data 和 type 中。

生成模型模型的部分通过 init_generative( ), maxitem( ), predict_generative( ),

test_benerative( )函数完成模型的生成和对测试集的测试。init_generative( )计算出训练集中每个标签对应的分布数据对应的概率 prob，均值 mean，协方差 cov。再用 predict_generative( )对每个测试集中的数据，使用 multivariate_normal( ).pdf( )得到当前数据在训练集中的三组标签对应高斯分布数据集可能出现的概率，加权求和后，使用 maxitem( )选出概率最大的作为预测标签。最后在 test_benerative( )中完成标签的比对以及散点图的描绘。估计错误的点用 X 表示。

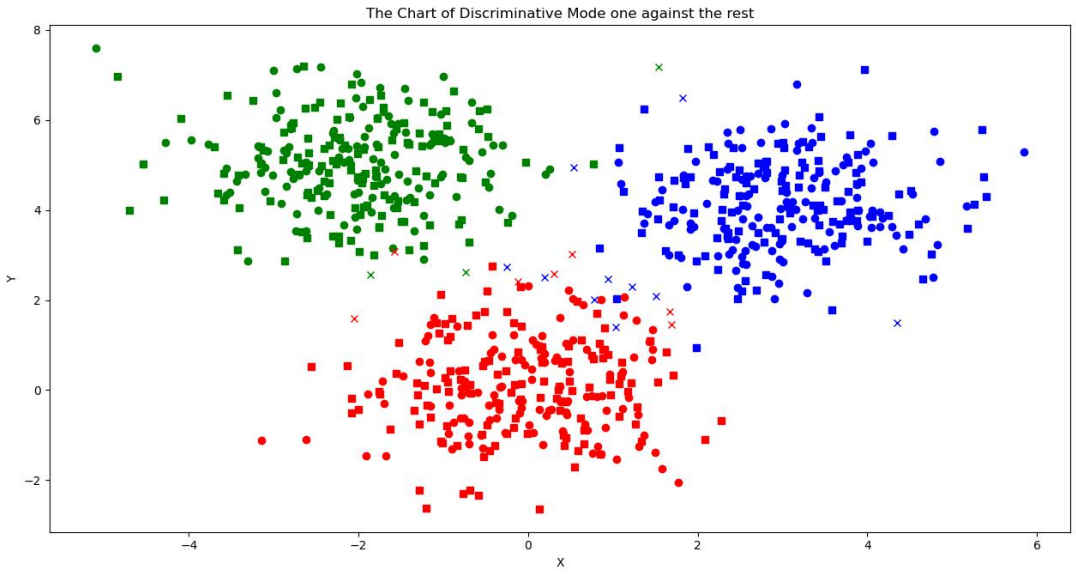
所得到的散点图如下所示：（估计的准确率为 0.974）

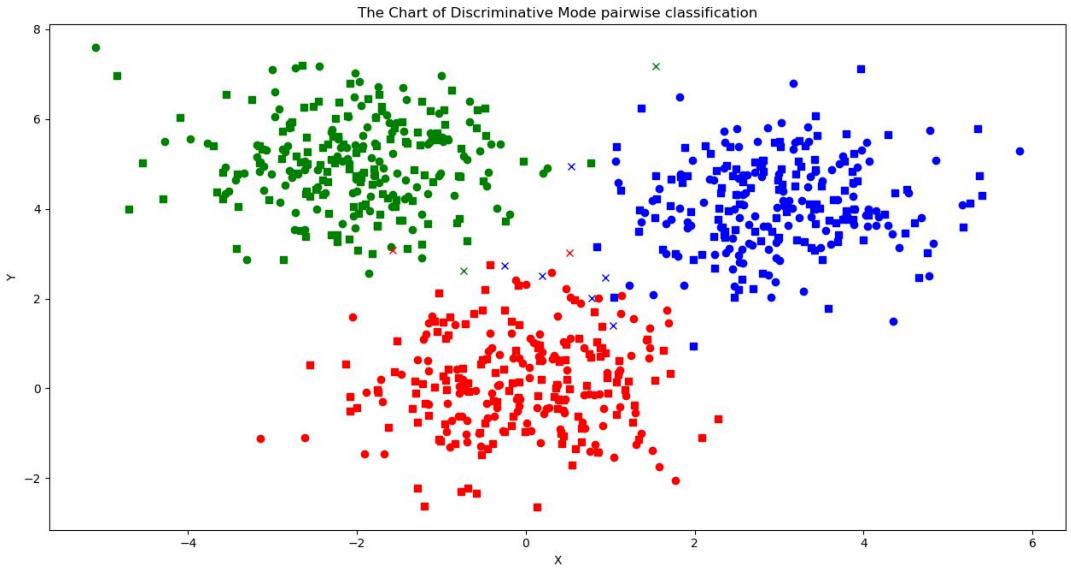



判别模型的部分通过 init_discriminative( ), predict_discriminative_one_against_the_rest( ), predict_discriminative_pairwise_classification( )和 test_discriminative( )函数完成模型的生成和对测试集的测试。程序使用 Fisher 线性判别分析（Linear Discriminant Analysis， LDA），使用 $y = w^T x$ 作为预测值，目标函数，$J(w) = \frac{\text{类间离散度}}{\text{类内离散度}} = \frac{S_B}{S_w} = \frac{|\overline{y_1} - \overline{y_2}| \times |\overline{y_1} - \overline{y_2}|}{inner_1^2 + inner_2^2}$，其中 $y_i = \frac{1}{N_i} \sum_{x \in X_i} w^T x$，Xi 为同一类的数据集，其中包含数据个数为 Ni，$inner_i^2 = \sum_{y \in y_i} (y - y_i)^2$。对 J(w)改写可得到 $J(w) = \frac{w^T S_B w}{w^T S_w w}$，由 Lagrange 乘子法及可以设 $S_B w = \gamma(\overline{x_1} - \overline{x_2}), \gamma := (\overline{x_1} - \overline{x_2})^T w$，得到 $w = S_w^{-1}(\overline{x_1} - \overline{x_2})$。通过 init_discriminative( )函数得到传递的数据集 a, b 对应的 w 和数据均值对应的 y。下面的 predict_discriminative_one_against_the_rest( )和 predict_discriminative_

pairwise_classification( )函数分别采用“一对剩余”和“一对一”的方法的多分类器。“一对剩余”是将数据分为(A, BC), (B, AC), (C, AB)三组，确认数据点对于某个标签对应集合（如 A）或者包含当前标签的集合（如 AC）的均值点的距离。如果在三组测试中均成立对某个标签的距离更短，则估计其属于该标签。“一对一”是将数据分为(A, B), (A, C), (B, C)三组，分别判断该点距离某个数据集的均值点与另两个数据集的均值点的距离的大小，若都较小则估计该点属于当前判断的数据集。最后在 test_benerative( )中完成标签的比对以及散点图的描绘。估计错误的点用 X 表示。

所得到的散点图如下所示：（估计的准确率分别为 0.948 和 0.974）



**两个模型的比较：**

多组数据比较两种三类模型准确率的统计如下图所示：

| | 生成模型 | 一对剩余 | 一对一 |
|---|---|---|---|
| 1 | 0.974 | 0.948 | 0.974 |
| 2 | 0.99 | 0.979 | 0.99 |
| 3 | 0.99 | 0.964 | 0.987 |
| 4 | 0.995 | 0.953 | 0.992 |
| 5 | 0.992 | 0.958 | 0.992 |
| 均值 | 0.9882 | 0.9604 | 0.987 |

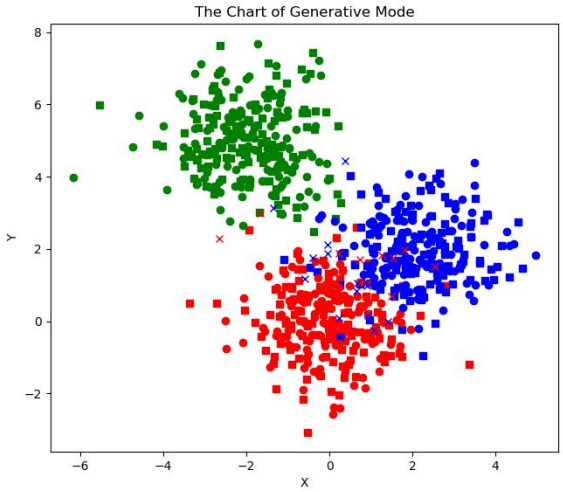
从随机生成的多组数据的结果看来，生成模型准确率为 0.9882%，“一对剩余”判别模型为 0.9604%，“一对多”判别模型为 0.987%，生成模型略优但三者相差不大。因为模型比较简单，并且数据的分布相对分散，只有少数点会分布在大量的其他数据中，造成预测错误。

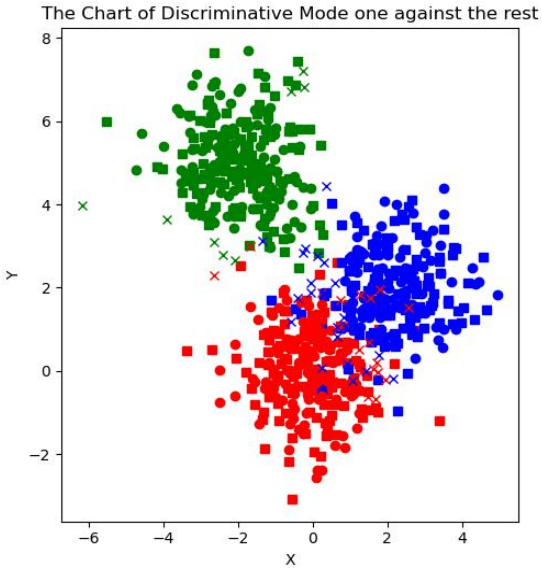
生成模型的收敛速度更快，在样本较多的情况下生成模型能更快的收敛于真实模型。但生成模型的计算需要大量的样本和更多的计算，为了更准确估计类别条件分布，需要增加样本的数目。且当属于的样本边缘分布很小的情况下，学习出的模型分类效果并不理想。

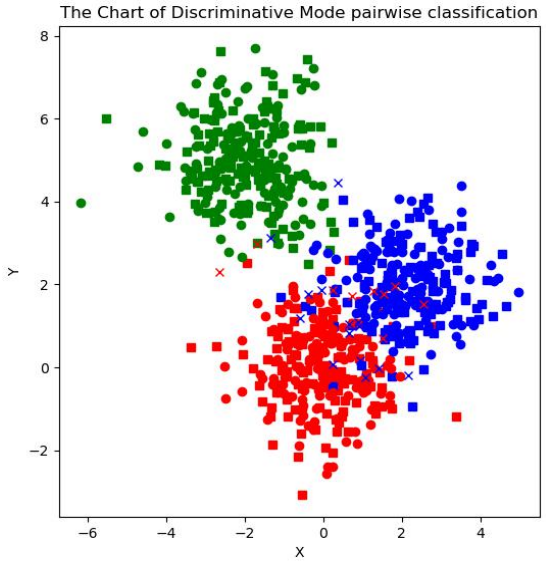
判别模型与生成模型相比更节省计算资源，需要的样本数量也少于生成模型。由于不需要求类别条件概率，所以可以对输入进行抽象构造，能够简化学习问题。

## Part III

1 .调整不同的高斯分布之间的重叠：构造一组数据（下表中数据 2），其原始数据图和三个模型的估计图如下：



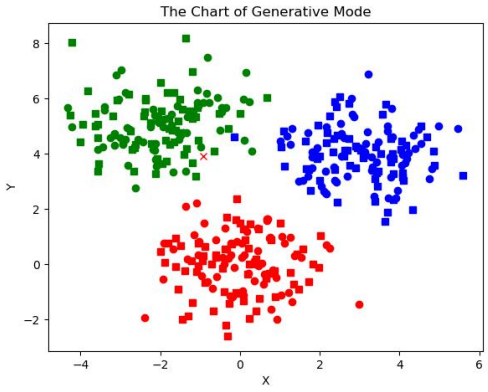



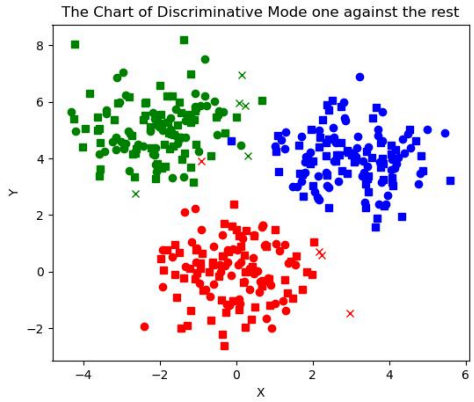

其中生成模型、“一对剩余”判别模型和“一对多”判别模型准确率分别为 0.94%、0.9867%、0.94%。多次改变数据中心的值，得到的结果如下表所示：

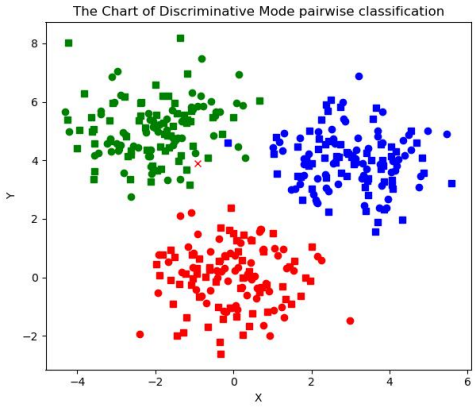
| | 数据 A 均值 | 数据 B 均值 | 数据 C 均值 | 生成模型 | 一对剩余 | 一对一 |
|---|---|---|---|---|---|---|
| 1 | 0，0 | 3，4 | -2,5 | 0.974 | 0.948 | 0.974 |
| 2 | 0，0 | 2，2 | -2,5 | 0.94 | 0.867 | 0.94 |
| 3 | 0，0 | 1.5,1.5 | -1.5,4 | 0.904 | 0.776 | 0.906 |
| 4 | 0.5,0.5 | 1.5，1.5 | -1,3 | 0.771 | 0.599 | 0.766 |
| 5 | 0.5,0.5 | 1,1 | 0,1 | 0.5 | 0.292 | 0.505 |
| cov=[[1,0],[0,1]],size=128 | | | | | | |

可以推断出，当数据分布越分散时，模型的估计效果越好，数据分布越集中，估计效果越差。

2 .同时调整三个的高斯分布的数据个数：构造一组数据（下左表中数据 2），其原始数据图和三个模型的估计图如下：



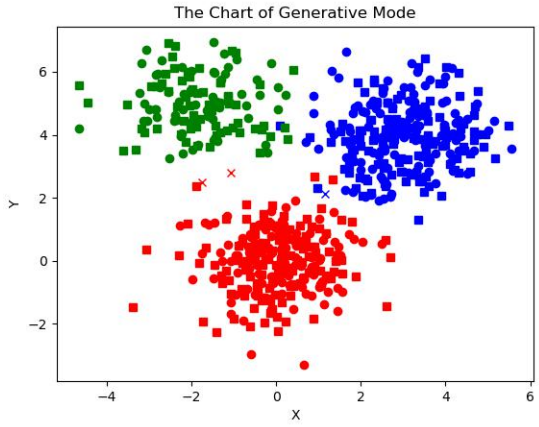

其中生成模型、“一对剩余”判别模型和“一对多”判别模型准确率分别为 0.995%、0.953%、0.974%。多次改变数据的数量，得到的结果如下表所示：

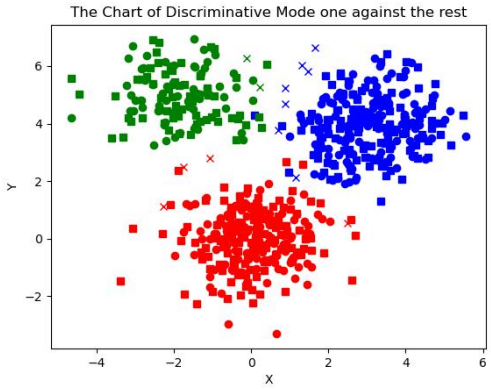
| | size | 生成模型 | 一对剩余 | 一对一 |
|---|---|---|---|---|
| 1 | 128 | 0.974 | 0.948 | 0.974 |
| 2 | 64 | 0.995 | 0.953 | 0.95 |
| 3 | 32 | 0.969 | 0.927 | 0.969 |
| 4 | 16 | 1 | 0.958 | 1 |
| 5 | 8 | 0.958 | 0.958 | 0.958 |
| mean=[[0,0],[3,4],[-2,5]]cov=[[1,0],[0,1]] | | | | |

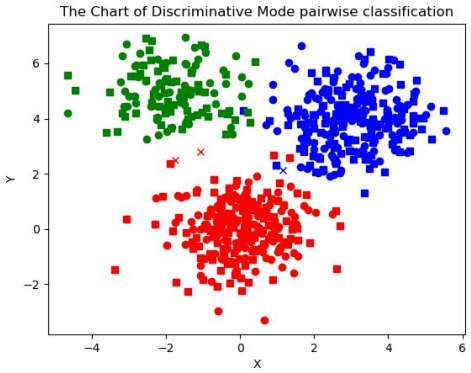
| | size | 生成模型 | 一对剩余 | 一对一 |
|---|---|---|---|---|
| 1 | 128 | 0.5 | 0.292 | 0.505 |
| 2 | 64 | 0.505 | 0.333 | 0.505 |
| 3 | 32 | 0.531 | 0.323 | 0.521 |
| 4 | 16 | 0.396 | 0.271 | 0.354 |
| 5 | 8 | 0.548 | 0.125 | 0.542 |
| mean=[[0.5,0.5],[1,1],[0,1]],cov=[[1,0],[0,1]] | | | | |

可以推断出，整体数据的多少对估计结果影响的规律并不显著。

3 .调整一个的高斯分布的数据个数。修改源程序，将第三组数据的个数减少，构造一组数据（下表 1 中数据 2），其原始数据图和三个模型的估计图如下：




其中生成模型、“一对剩余”判别模型和“一对多”判别模型准确率分别为 0.991%、0.959%、0.991%。多次改变第三组数据的数量，得到的结果如下表所示：

| | size3 | 生成模型 | | 一对剩余 | | 一对一 | |
|---|---|---|---|---|---|---|---|
| | | 第三组错误数目 | 估计正确概率 | 第三组错误数目 | 估计正确概率 | 第三组错误数目 | 估计正确概率 |
| 1 | 128 | 2 | 0.974 | 3 | 0.948 | 2 | 0.974 |
| 2 | 64 | 0 | 0.991 | 2 | 0.959 | 0 | 0.991 |
| 3 | 32 | 0 | 0.997 | 1 | 0.965 | 0 | 0.993 |
| 4 | 16 | 0 | 0.993 | 2 | 0.96 | 0 | 0.989 |
| 5 | 8 | 0 | 0.992 | 1 | 0.977 | 0 | 0.992 |

| mean=[[0,0],[3,4],[-2,5]],cov=[[1,0],[0,1]],size1=size2=128 |
|---|

| | size3 | 生成模型 | | 一对剩余 | | 一对一 | |
|---|---|---|---|---|---|---|---|
| | | 第三组错误数目 | 估计正确概率 | 第三组错误数目 | 估计正确概率 | 第三组错误数目 | 估计正确概率 |
| 1 | 128 | 29 | 0.888 | 64 | 0.5625 | 21 | 0.88 |
| 2 | 64 | 12 | 0.919 | 42 | 0.553 | 7 | 0.894 |
| 3 | 32 | 13 | 0.903 | 24 | 0.6 | 7 | 0.896 |
| 4 | 16 | 10 | 0.963 | 16 | 0.585 | 2 | 0.912 |
| 5 | 8 | 7 | 0.966 | 8 | 0.674 | 1 | 0.905 |
| mean=[[0,0],[4,4],[2,3]],cov=[[1,0],[0,1]],size1=size2=128 | | | | | | | |

可以推断出，当数量变化的数据与另两组数据重叠部分越多时，数据数量减少对生成模型和 LDA 一对剩余模型的估计效果负相关，LDA 一对一模型受影响较小。

## 代码运行命令

python source.py

## 参考网页


[1] https://blog.csdn.net/weixin_42782150/article/details/96283330 #python 生成高斯分布数据

[2] https://www.cnblogs.com/dudududu/p/9149762.html #python 可视化

[3] https://blog.csdn.net/qq_28618765/article/details/78085457 #python 创建数组

[4] https://blog.csdn.net/u010358304/article/details/79748153 #生成模型与判别模型区别

[5] https://blog.csdn.net/wuxintdrh/article/details/89647956 #线性回归 python 实现

[6] https://www.cnblogs.com/wanghui-garcia/p/10763418.html #多元正态随机变量获取

[7] https://blog.csdn.net/qiurisiyu2016/article/details/80187177 #plt 运用

[8] https://blog.csdn.net/code_caq/article/details/70194488 #Fisher 线性判别分析（Linear Discriminant Analysis，LDA）
[9] https://www.jianshu.com/p/a68a3c0a547c #LDA 多分类
[10] https://blog.csdn.net/itplus/article/details/12038357 #LDA 二分类
[11] https://www.cnblogs.com/ysdu/p/6003077.html #LDA 多分类
[12] http://www.cmlab.csie.ntu.edu.tw/~cyy/learning/tutorials/LDA.pdf #LDA 介绍
[13] https://www.cnblogs.com/abella/p/10207945.html #矩阵用法
[14] https://www.cnblogs.com/hezhiyao/p/8490762.html #python 合并数组