6819个样本，但只有220个为破产样本，导致预测结果偏向非破产

基本思想：利用已知数据构造合理的破产样本(over-sampling)；减少非破产样本个数(under-sampling)结合；前面两种结合

方法：imblanced-learning库

初始结果（6819个样本未经过任何裁剪）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 97.07% | 95.45% | 97.07% | 97.14% | 97.14% | 96.99% |
| F1 score | 0.2593 | 0.3111 | 0.1667 | 0.3607 | 0.0488 | 0.3279 |

**Over-Sampling:**

1 随机过程采样

从少数类的样本中进行随机采样来增加新的样本，使得所有类样本数相同

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 86.51% | 96.11% | 97.07% | 92.08% | 87.61% | 96.04% |
| F1 score | 0.2459 | 0.3291 | 0.3333 | 0.3333 | 0.2489 | 0.2703 |

效果较差。可能原因：这个随机采样等价于给破产的样本赋予了很高的权重进行分类，导致这些破产的样本对模型影响太大，进而产生误差的积累。

2 Synthetic Minority Oversampling Technique (SMOTE)

通过插值产生新的样本。对于少数类样本a, 随机选择一个近邻的少数类样本b, 然后对每一对a与b，从其连线上随机选取一个点c作为新的少数类样本，对每个样本a重复N次。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 87.98% | 93.99% | 95.75% | 93.33% | 88.64% | 96.41% |
| F1 score | 0.2743 | 0.2931 | 0.3830 | 0.3546 | 0.2654 | 0.3288 |

SMOTE对少数样本一视同仁，未考虑近邻样本的类别信息，往往出现样本混叠现象，导致分类效果不佳。

3 ADASYN:

通过插值产生新的样本。与SMOTE中插值方法类似，但不同少数类样本根据其周围的多数类样本数目而进行不同数目的插值过程。K近邻中多数类样本数目越多，插值的次数越多。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 86.07% | 93.55% | 95.53% | 92.30% | 89.22% | 96.41% |
| F1 score | 0.2460 | 0.2903 | 0.3297 | 0.3046 | 0.2613 | 0.3467 |

4 BorderlineSMOTE

对少数类样本进行了分类。仅针对那些K近邻中多数样本占一半以上但又有少数样本的少数样本进行SMOTE插值。Borderline-1插值过程与SMOTE相同，而Borederline-2则并不限制b的类别。（上面是l1的数据，下面是l2的）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 91.13% | 93.99% | 96.48% | 94.57% | 92.23% | 96.63% |
| F1 score | 0.3086 | 0.2807 | 0.4286 | 0.3729 | 0.3117 | 0.3611 |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 87.24% | 93.11% | 96.41% | 94.21% | 90.47% | 96.19% |
| F1 score | 0.2564 | 0.2295 | 0.4235 | 0.3876 | 0.2697 | 0.3333 |

<https://blog.csdn.net/u010654299/article/details/103980964>

还有其它类型SMOTE，有待研究https://zhuanlan.zhihu.com/p/137826761

**under-Sampling**

1 原型生成——从原始数据生成若干数据

ClusterCentroids函数实现了上述功能: 每一个类别的样本都会用K-Means算法的中心点来进行合成, 而不是随机从原始样本进行抽取.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 79.55% | 59.16% | 79.25% | 64.59% | 75.73% | 81.23% |
| F1 score | 0.2051 | 0.1059 | 0.1891 | 0.1329 | 0.1704 | 0.1950 |

该方法要求原始数据集最好能聚类成簇，然而很明显这个数据并不适合聚类。

2 原型选择——从原始数据集抽取若干数据

2.1 直接随机下采样（不重复采样）。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 81.01% | 77.86% | 83.94% | 81.16% | 78.37% | 80.06% |
| F1 score | 0.2031 | 0.1793 | 0.2422 | 0.2043 | 0.1918 | 0.1905 |

效果一般。猜测是此时数据量太少（220\*2个样本）。

实际上直接从原始数据集删除前5000条数据的效果都比这个好。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 93.68% | 88.74% | 92.31% | 94.78% | 92.86% | 93.13% |
| F1 score | 0.6349 | 0.4533 | 0.4615 | 0.6984 | 0.5000 | 0.6269 |

2.2 清洗数据

本质上是通过移除原始数据中可能是噪声的数据来降低样本量。但最终得到的样本数目不可控。由于下述都在某种程度上采用了K近邻法，而事实上大部分多数类样本周围也都是多数类，因而能剔除的多数类样本比较有限。

2.2.1 Tomeks links

样本x与样本y来自于不同的类别, 满足以下条件, 它们之间被称之为TomekLinks：不存在另外一个样本z, 使得d(x,z) < d(x,y) 或者 d(y,z) < d(x,y)成立. 其中d(.)表示两个样本之间的距离, 也就是说两个样本之间互为近邻关系. 这个时候, 样本x或样本y很有可能是噪声数据, 或者两个样本在边界的位置附近。

选取数据中所有的TomeLinks，删除这些点对中属于多数类的点

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 97.14% | 95.60% | 96.99% | 97.14% | 97.14% | 96.77% |
| F1 score | 0.2909 | 0.3517 | 0.1961 | 0.3810 | 0.0488 | 0.3333 |

得到结果和未经处理的情况几乎相同。可能原因：删除的样本数有限

2.2.2 EditedNearestNeighbours

对于属于多数类的一个样本，如果其K个近邻点有超过一半都不属于多数类，则这个样本会被剔除（对应kind\_sel='mode'）。这个方法的另一个变种是所有的K个近邻点都不属于多数类，则这个样本会被剔除（对应kind\_sel='all'）。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 97.21% | 95.09% | 97.21% | 97.14% | 97.14% | 96.70% |
| F1 score | 0.3667 | 0.3093 | 0.2692 | 0.4179 | 0.0488 | 0.3478 |

准确率较好，F1 score较未处理也有了一定的提升。

<https://blog.csdn.net/weixin_43329700/article/details/107325026>

**过采样与下采样结合**

在之前的SMOTE方法中, 当由边界的样本与其他样本进行过采样差值时, 很容易生成一些噪音数据. 因此, 在过采样之后需要对样本进行清洗. 而TomekLink 与 EditedNearestNeighbours方法都能实现上述的要求

SMOTETomek：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 87.98% | 93.55% | 96.33% | 93.48% | 88.71% | 96.11% |
| F1 score | 0.2743 | 0.2281 | 0.4186 | 0.3776 | 0.2667 | 0.2933 |

比纯粹的SMOTE方法略好，但改进效果有限。

SMOTEENN：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | L-regression | D-tree | R-forest | G-boost | SVM | MLP |
| accuracy | 85.12% | 91.86% | 94.21% | 90.25% | 86.66% | 93.70% |
| F1 score | 0.2454 | 0.3106 | 0.4060 | 0.3179 | 0.2353 | 0.3485 |

不如SMOTETomek，具体原因有待研究。

<https://blog.csdn.net/kizgel/article/details/78553009>

总体来讲，Over-Sampling以及Over-Sampling+Under-Sampling的方法都牺牲了一部分准确率从而获得了更高的F1 score。而Under-Sampling中清洗数据的方法由于删除的数据量过少与不做任何处理时几乎没有区别。效果最好的反而是直接删除前5000个样本。