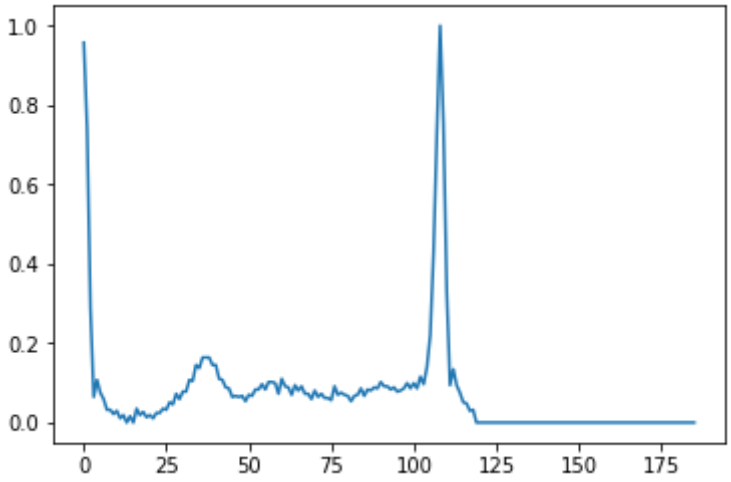
A way to deal with imbalanced dataset

By: ZhongLexing 18124722, ChenWenlong 18124723

Github:https://github.com/lbcsimon/ProjectMT01.git

1. Introduction

In this project, we are trying to deal with an imbalanced classification task by comparing the performance of the classifiers we learn in the course. The main idea of this project is how the sampling method improves the performance of each classifier. The dataset we use is the **Physionet's MIT-BIH Arrhythmia Dataset**. Each observation of this dataset contains 186 records of the heart signal during 12 seconds of a person. The following figure shows one example.



2. Dataset

The basic information of our dataset as followed:

```
Number of Samples: 109446
Number of Categories: 5
Sampling Frequency: 125Hz
Data Source: Physionet's MIT-BIH Arrhythmia Dataset
Classes: ['N': 0, 'S': 1, 'V': 2, 'F': 3, 'Q': 4]

class proportion in the train dataset
    0 - Non-ecotic beats (normal beat)         72471   82.7%
    1 - Supraventricular ectopic beats         2223    2.54%
    2 - Ventricular ectopic beats              5788    6.61%
    3 - Fusion Beats                           641     0.7%
    4 - Unknown Beats                          6431    7.34%
```

We can see that the observations of **the fusion beats** and **the supraventricular ectopic beats** are far small than the other and the normal beat is bigger more than 10 times than the other. This might be the normal case in the Medical field. The negative samples are far small than the normal case. So how to deal with the imbalanced classification dataset is a significant part when we apply machine learning in the medical field.

In the following part, we are going to use **oversampling** and the **SMOTE sampling** method to do the comparison. Before that, Let's have a look at the performance of the classifiers we learned in the course without any sampling.

3. Performance without sampling

We have trained KNN, LDA, Logistic Regression, decision tree, Random Forest, Naïve Bayes, and a simple neural network with 5 hidden layer models to predict the test dataset. Since the number of observations of normal beat is quite large so the accuracy score metric is not that meaningful, so we use the f1 score and the roc area of each class as the main metric to compare the model performance. We can see the result as follows:

classifier\metric	F1 score	accuracy score	auc 0	auc 1	auc 2	auc 3	auc 4	micro auc
KNN	0.974	0.974	0.978	0.906	0.984	0.925	0.990	0.994
LDA	0.892	0.892	0.904	0.780	0.852	0.947	0.966	0.974
Logistic Regression	0.906	0.906	0.914	0.815	0.897	0.963	0.983	0.984
Decision Tree	0.956	0.956	0.939	0.831	0.939	0.816	0.973	0.978
Random Forest	0.973	0.973	0.994	0.978	0.998	0.983	0.999	0.999
neural network	0.974	0.974	0.986	0.947	0.994	0.976	0.999	0.997
Naïve Bayes	0.20	0.19	0.77	0.62	0.63	0.92	0.69	0.62

as we can see in the table:

1. The prediction of class 1 and class 3 is not promising because of the gap in the number of observations.
2. It seems that the KNN and neural network and the random forest model do a great job but they still have problems with class 1 and class 3.
3. The performance of the Naïve Bayes is the worse. the reason for that is because the Naïve Bayes model assumes all predictors are independent but in our case is not.

Let's have a look at the confusion matrix of the KNN, Random Forest, and the neural network model to see how they predict the class that has small train data.

KNN

```
confusion_matrix:
[[18021    38    45     9     5]
 [  186   357    13     0     0]
 [  113     3  1305    23     4]
 [   43     0    16   103     0]
 [   69     0    10     0  1529]]
```

Random Forest

```
confusion_matrix:
[[18101     4    11     0     2]
 [  230   323     2     0     1]
 [  160     0  1270    14     4]
 [   51     0    13    98     0]
 [   97     0     4     0  1507]]
```

Neural Network

```
confusion_matrix:
[[17944   111    21    11    31]
 [  147   394     9     3     3]
 [  107    11  1292    27    11]
 [   34     0     9   119     0]
 [   29     0     8     1  1570]]
```

We can see that the KNN and the random Forest can assure the accuracy of the normal case but perform worse than the neural network when dealing with the other class. Although the neural network does perform a little bit better than the other, it lost the quality of the normal case.

4. Sample Data

How can we deal with this situation? There are two ways to solve. One is to increase the weight of the small class when training the data. The other one is to preprocess the dataset to make it balanced. Since the second one is easier to get and the idea of it is also increase the weight of small class. In the following part, We are going to use two methods of sampling the data and test

these three models to see how balanced data help to improve our task.

4.1 Oversample

The first method we use is oversample. The main idea of this method is to randomly sample the minority until its number reaches the number of the majority. In this part, since the normal case have almost 72000 observations, We have just sampled a new dataset by randomly repeated the other one to 7000. So here we are. We have generated a dataset of 350000 observations. Let's have a look at the result of this dataset.

classifier\metric	F1 score	accuracy score	auc 0	auc 1	auc 2	auc 3	auc 4	micro auc
KNN_oversample	0.971	0.971	0.976	0.899	0.979	0.924	0.990	0.992
KNN	0.974	0.974	0.978	0.906	0.984	0.925	0.990	0.994
Random Forest_oversample	0.974	0.974	0.994	0.981	0.997	0.981	0.999	0.999
Random Forest	0.973	0.973	0.994	0.978	0.998	0.983	0.999	0.999
neural network_oversample	0.952	0.952	0.982	0.929	0.991	0.975	0.997	0.995
neural network	0.974	0.974	0.986	0.947	0.994	0.976	0.999	0.997

So as the table shows, Increase the number of observations of the small class does not seem to have a very good improvement of our task. Let's have a look at the confusion matrix to see the detail.

Shape (87554, 187)

KNN

```
confusion_matrix:
[[18021    38    45     9     5]
 [  186   357    13     0     0]
 [  113     3  1305    23     4]
 [   43     0    16   103     0]
 [   69     0    10     0  1529]]
```

Random Forest

```
confusion_matrix:
[[18101     4    11     0     2]
 [  230   323     2     0     1]
 [  160     0  1270    14     4]
 [   51     0    13    98     0]
 [   97     0     4     0  1507]]
```

Neural Network

```
confusion_matrix:
[[17944   111    21    11    31]
 [  147   394     9     3     3]
 [  107    11  1292    27    11]
 [   34     0     9   119     0]
 [   29     0     8     1  1570]]
```

Shape (350000, 187)

KNN oversample

```
confusion_matrix:
[[17906   128    46    33     5]
 [  138   402    11     5     0]
 [  105    10  1289    40     4]
 [   21     0    10   131     0]
 [   67     2     9     1  1529]]
```

Random Forest oversample

```
confusion_matrix:
[[18098    10     6     3     1]
 [  206   349     0     0     1]
 [  175     2  1250    17     4]
 [   42     0     8   112     0]
 [   94     0     5     0  1509]]
```

Neural Network oversample

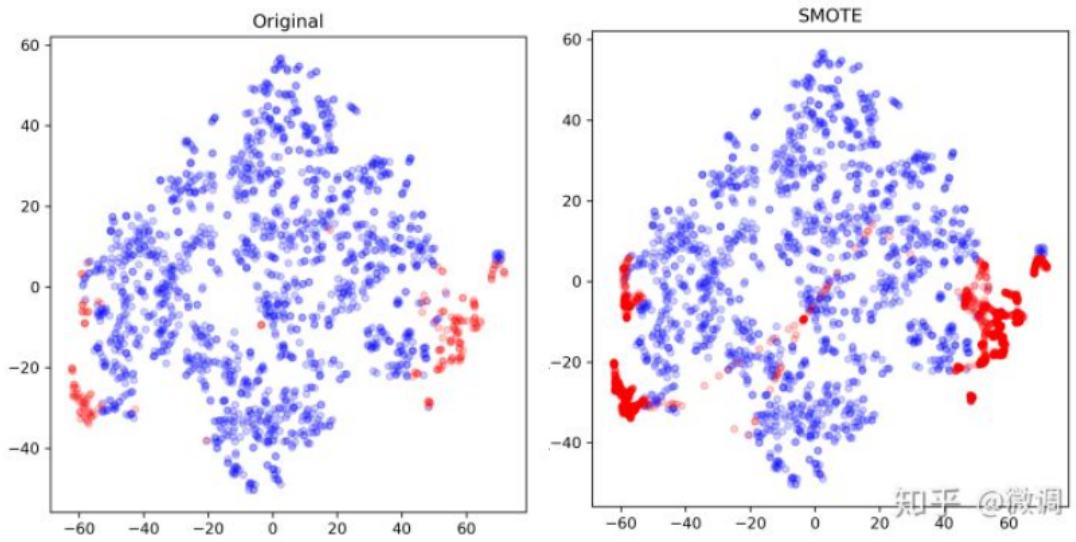
```
confusion_matrix:
[[17399   342   269    38    70]
 [  124   413    14     1     4]
 [   62     6  1358    17     5]
 [   22     1    14   125     0]
 [   31     6    16     2  1553]]
```

We can see that in the real prediction, all models have an improved performance when dealing with class 1 and class 3 comparing to the without sampling version, but they also may lose the quality of the normal case.

We can conclude that balance the data did help to improve the performance. But the result is not that pleasing. It indeed increases the weight of the small class and it also potentially decrease the weight of the normal case. So in the following part, we are going to use a cleverer algorithm to balance the dataset.

4.2 SOMTE

the second method we use is SOMTE. It's a sample algorithm which means to find the data neighbor as the new data. The biggest difference to the basic sample method is that it generates data base on the original one. Here is an example.



We have applied this method to our dataset and generated a new dataset with 350000 instances as the oversampling datasets. The confusion matrix is showed as follow:

Shape (350000, 187)

KNN smote

```
confusion_matrix:
[[17393   411   158   105    51]
 [   76   455    18     5     2]
 [   25    18  1366    31     8]
 [   12     1    14   135     0]
 [   19     5    10     5  1569]]
```

Random Forest smote

```
confusion_matrix:
[[17997    68    32    13     8]
 [  127   424     2     2     1]
 [   71     3  1352    16     6]
 [   29     0    12   121     0]
 [   43     0     5     0  1560]]
```

Neural Network smote

```
confusion_matrix:
[[16769   826   248   131   144]
 [  106   431    11     5     3]
 [   85    10  1315    28    10]
 [   14     3    10   135     0]
 [   25     6    10     1  1566]]
```

Shape (350000, 187)

KNN oversample

```
confusion_matrix:
[[17906   128    46    33     5]
 [  138   402    11     5     0]
 [  105    10  1289    40     4]
 [   21     0    10   131     0]
 [   67     2     9     1  1529]]
```

Random Forest oversample

```
confusion_matrix:
[[18098    10     6     3     1]
 [  206   349     0     0     1]
 [  175     2  1250    17     4]
 [   42     0     8   112     0]
 [   94     0     5     0  1509]]
```

Neural Network oversample

```
confusion_matrix:
[[17399   342   269    38    70]
 [  124   413    14     1     4]
 [   62     6  1358    17     5]
 [   22     1    14   125     0]
 [   31     6    16     2  1553]]
```

As we can see, with the same number of observations, SMOTE dataset perform quite same as the oversample version. And they share the same conclusion: with the balanced data, we can have a better prediction when dealing with the imbalanced classification task.

Conclusion

In this project, we are facing a regular situation in the Medical field: with small irregular data, it's hard to predict a person is health or not. And in our experiment, balance the data can help us to solve the situation. There is still another way to deal with it, like control the threshold when training the data or combine sampling dataset method with the regularization. Sample data may not be the best way to do but we think it's the easier and interpretable one.

We have implemented our project with Python and R. And there is an interesting thing: with python sci-kit-learn packages, the KNN model is not that well, but in R, the KNN model did a perfect job and its accuracy score reach almost 1 and the prediction of each class are also unbelievable. The result is shown as follow:

	pred_smote_knn1				
	0	1	2	3	4
0	18113	5	0	0	0
1	17	536	3	0	0
2	1	7	1435	5	0
3	0	0	2	160	0
4	0	0	1	4	1603

We think maybe this is the most interesting part of Machine Learning, We humans, in this powerful, interesting, and far-reaching field, still have a lot to do.