Q2 Report: Imbalanced Data Classification

Author: YANG Rongfeng

Date: 2020-05-31

1. The Algorithms Details

In this question, I implement two sampling algorithms in bi-class classification task: SMOTE and SMOTEENN. Then I use random forest to do classification on the resampled dataset.

For classifying the Multi-class datasets, I implement four classifiers which can handle the imbalanced datasets: balanced random forest classifier, RUS boost classifier, easy ensemble classifier and balanced bagging classifier. I will try these four classifiers for each dataset and select the one with best performance for final prediction on the testing set.

Now I will discuss them in detail.

1.1 SMOTE

SMOTE algorithm which is known as *Synthetic Minority Over-sampling Technique* is a technology to creates artificial data based on similarities in feature space between existing minority classes through the introduction of minority classes that are not replicated that does not lead to loss of information. Introducing new examples is an effective way of changing learner bias and creating more general bias, chiefly in terms of minority classes. The K-NN algorithm is employed for the extrapolation and creation of new minority examples from present imbalances in the minority classes. The K-NN neighbors are selected at random based on the quantity of oversampling that is needed. Adding synthetically generated minority class examples creates more balance within the class distribution.

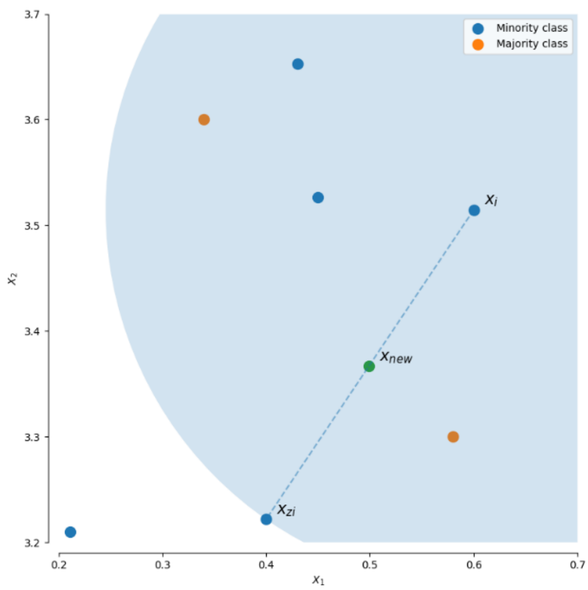

Figure.1 An example of SMOTE generation

Specifically, for a minority sample x_i using K-NN to calculate k closest samples from x_i by Euclidean distance. Then randomly choose a point from its K nearest neighbors, use the equation to generate new samples:

$$x_{new} = x_i + (\hat{x}_i - x_i) \times \delta$$

where x_i is the point that we randomly choose in x_i's K nearest neighbors. $\delta \in [0, 1]$ is a random number. In Figure 1, it shows a point generated by SMOTE algorithm.

1.2 SMOTEENN

SMOTEENN is a kind of hybrid methodology that combines SMOTE and ENN (Edited Nearest Neighbors). SMOTE is used to over-sampling and ENN is used for cleaning data. The purpose of SMOTEENN is to balance training data as well as to take out noisy instances. ENN tends to remove a greater number of instances than Tomek links. ENN is employed for the removal of instances in all classes and so any instance which undergoes misclassification from all three Nearest Neighbors will be taken out of the training set.

In Figure 2, we compare different resampling methods: Original, SMOTE and SMOTEEEN. We can clearly see that the SMOTE oversampling cause data overlapping which will make classification more difficult. The SMOTEEEN do data cleaning after SMOTE and its classification boundary is more clearly.

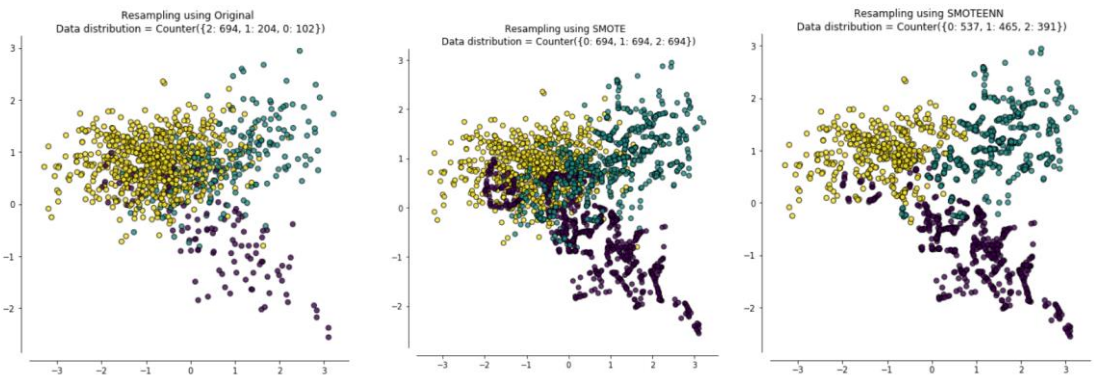

Figure2. Resampling using (a) Original (b) SMOTE (c) SMOTEENN

1.3 Random Oversampling (ROS)

ROS technique operates through the replication of a randomly chosen collection of examples in the minority class to prevent the majority class disproportionately affecting training processes. As random sampling is used, the decision function finds it problematic in distinguishing borders between classes. Thus, although it is a standard method, random oversampling may not be effective in causing large improvements to minority class recognition. Possible problems with over-sampling are that the classifier training time may be increased and overfitting may appear as class imbalance ratios worsen with the duplication of minority class examples.

1.4 Random Forest

Random Forest is an ensemble method which combines predictions from a multiplicity of Decision Tree algorithms to produce predictions that have a higher accuracy than any individual model. RF employs the classification from each tree to make an overall classification. The RF will choose the class of instances presented to it that has the most votes.

1.5 BalancedRandomForestClassifier

BalancedRandomForestClassifier is another ensemble method in which each tree of the forest will be provided a balanced bootstrap sample.

1.6 RUSBoostClassifier

RUSBoostClassifier randomly under-sample the dataset before to perform a boosting iteration.

1.7 EasyEnsembleClassifier

A specific method which uses AdaBoost as learners in the bagging classifier is called EasyEnsemble. The EasyEnsembleClassifier allows to bag AdaBoost learners which are trained on balanced bootstrap samples.

1.8 BalancedBaggingClassifier

In ensemble classifiers, bagging methods build several estimators on different randomly selected subset of data. In scikit-learn, this classifier is named BaggingClassifier. However, this classifier does not allow to balance each subset of data. Therefore, when training on imbalanced data set, this classifier will favor the majority classes.

BalancedBaggingClassifier allows to resample each subset of data before to train each estimator of the ensemble. In short, it combines the output of an EasyEnsemble sampler with an ensemble of classifiers (i.e. BaggingClassifier). Therefore, BalancedBaggingClassifier takes the same parameters than the scikit-learn BaggingClassifier. Additionally, there is two additional parameters, sampling_strategy and replacement to control the behaviour of the random under-sampler.

2. Bi-class Datasets

2.1 v_train.csv

I use **SMOTE** algorithm to oversample the original data.

```
smote = SMOTE(random_state=0)
X_smote, y_smote = smote.fit_sample(X, y)
```

Then the Random Forest Classifier is used to do classification.

```
rf = RandomForestClassifier(n_estimators=5, random_state=0, max_depth=2)
```

Split the training set by 20% for validation and 80% for training. The evaluation result on the validation set is shown below:

```
Random Forest evalute on validation set
              precision    recall  f1-score   support

    negative       0.94      0.92      0.93       161
    positive       0.93      0.95      0.94       186

    accuracy                           0.94       347
   macro avg       0.94      0.94      0.94       347
weighted avg       0.94      0.94      0.94       347
```

2.2 p_train.csv

I use a **SMOTEEEN** to resample the dataset.

```
sme = SMOTEENN(random_state=27)
X_sme, y_sme = sme.fit_resample(X, y)
```

Also split validation for evaluation and use Random Forest Classifier to predict as Section 1.1. The evaluation result:

```
Random Forest evalute on validation set
              precision    recall  f1-score   support

    negative       0.90      0.85      0.88        41
    positive       0.89      0.93      0.91        55

    accuracy                           0.90        96
   macro avg       0.90      0.89      0.89        96
weighted avg       0.90      0.90      0.90        96
```

3. Multi-class Datasets

To classify three multi-class datasets, I try four different ensemble methods provided by `imblearn` for each dataset and select the one with best performance for final prediction on the testing set.

- BalancedRandomForestClassifier
- RUSBoostClassifier
- EasyEnsembleClassifier
- BalancedBaggingClassifier

3.1 y_train.csv

The categories distribution of y_train.csv

```
Counter({'MIT': 234,
          'NUC': 410,
          'CYT': 450,
          'ME1': 36,
          'EXC': 33,
          'ME2': 46,
          'ME3': 140,
          'VAC': 30,
          'POX': 18,
          'ERL': 5})
```

Try four different imbalanced classifier models

```
brf = BalancedRandomForestClassifier(n_estimators=15, max_depth=2,
                                     random_state=0)
rusb = RUSBoostClassifier(n_estimators=50, learning_rate=1e-3, random_state=0)
eec = EasyEnsembleClassifier(n_estimators=10, random_state=0)
bbc = BalancedBaggingClassifier(n_estimators=10, random_state=0)
```

The performance comparison. Use RUSBoost Classifier to predict on testing set.

Balanced Random Forest evalute on validation set

	precision	recall	f1-score	support
CYT	0.49	0.57	0.53	109
ERL	0.50	1.00	0.67	2
EXC	0.33	0.57	0.42	7
ME1	0.50	0.44	0.47	9
ME2	0.21	0.33	0.26	9
ME3	0.44	0.84	0.58	25
MIT	0.47	0.16	0.24	44
NUC	0.42	0.12	0.18	68
POX	0.14	0.75	0.24	4
VAC	0.08	0.25	0.12	4
accuracy			0.41	281
macro avg	0.36	0.50	0.37	281
weighted avg	0.44	0.41	0.38	281

RUSBoost Classifier evalute on validation set

	precision	recall	f1-score	support
CYT	0.44	0.35	0.39	88
ERL	0.50	1.00	0.67	1
EXC	0.42	0.56	0.48	9
ME1	0.50	0.50	0.50	8
ME2	0.00	0.00	0.00	6
ME3	0.76	0.76	0.76	29
MIT	0.31	0.42	0.36	43
NUC	0.59	0.39	0.47	89
POX	0.67	0.40	0.50	5
VAC	0.03	0.33	0.05	3
accuracy			0.42	281
macro avg	0.42	0.47	0.42	281
weighted avg	0.49	0.42	0.45	281

Easy Ensemble Classifier evalute on validation set

	precision	recall	f1-score	support
CYT	0.55	0.43	0.48	92
ERL	0.33	1.00	0.50	1
EXC	0.42	1.00	0.59	5
ME1	0.25	0.33	0.29	3
ME2	0.43	0.50	0.46	12
ME3	0.68	0.62	0.65	24
MIT	0.40	0.44	0.42	54
NUC	0.61	0.32	0.42	79
POX	0.33	0.17	0.22	6
VAC	0.02	0.20	0.04	5
accuracy			0.42	281
macro avg	0.40	0.50	0.41	281
weighted avg	0.52	0.42	0.45	281

Balanced Bagging Classifier evalute on validation set

	precision	recall	f1-score	support
CYT	0.44	0.36	0.40	83
ERL	0.67	1.00	0.80	2
EXC	0.41	0.70	0.52	10
ME1	0.62	0.83	0.71	6
ME2	0.17	0.38	0.23	8
ME3	0.68	0.83	0.75	30
MIT	0.59	0.47	0.52	47
NUC	0.47	0.29	0.36	82
POX	0.20	0.75	0.32	4
VAC	0.04	0.11	0.06	9
accuracy			0.43	281
macro avg	0.43	0.57	0.47	281
weighted avg	0.48	0.43	0.44	281

3.2 e_train.csv

The categories distribution of e_train.csv

```
Counter({'cp': 143,
         'im': 77,
         'imS': 2,
         'imL': 2,
         'imU': 35,
         'om': 20,
         'omL': 5,
         'pp': 19})
```

Try four different imbalanced classifier models

```
brf = BalancedRandomForestClassifier(n_estimators=20, max_depth=2,
                                      random_state=9)
rusb = RUSBoostClassifier(n_estimators=50, learning_rate=1e-2, random_state=0)
eec = EasyEnsembleClassifier(n_estimators=10, random_state=0)
bbc = BalancedBaggingClassifier(n_estimators=10, random_state=0)
```

Compare the performance of four classifier and use Balanced Bagging Classifier to predect on testing set.

Balanced Random Forest evalute on validation set

	precision	recall	f1-score	support
cp	1.00	0.14	0.25	76
im	0.82	0.84	0.83	38
imL	0.00	0.00	0.00	0
imS	0.00	0.00	0.00	1
imU	0.75	0.35	0.48	17
om	0.36	1.00	0.53	8
omL	0.00	0.00	0.00	4
pp	0.06	0.50	0.11	8
accuracy			0.40	152
macro avg	0.37	0.35	0.28	152
weighted avg	0.81	0.40	0.42	152

RUSBoost Classifier evalute on validation set

	precision	recall	f1-score	support
cp	0.66	0.95	0.78	22
im	1.00	0.11	0.19	19
imL	0.00	0.00	0.00	0
imS	0.07	1.00	0.12	1
imU	0.50	0.12	0.20	8
om	1.00	0.20	0.33	5
omL	1.00	1.00	1.00	1
pp	0.71	1.00	0.83	5
accuracy			0.52	61
macro avg	0.62	0.55	0.43	61
weighted avg	0.77	0.52	0.48	61

Easy Ensemble Classifier evalute on validation set

	precision	recall	f1-score	support
cp	0.96	0.77	0.85	30
im	0.57	0.53	0.55	15
imL	0.00	0.00	0.00	0
imS	0.00	0.00	0.00	0
imU	0.12	0.10	0.11	10
om	0.25	0.50	0.33	2
omL	1.00	0.50	0.67	2
pp	0.00	0.00	0.00	2
accuracy			0.56	61
macro avg	0.36	0.30	0.31	61
weighted avg	0.67	0.56	0.61	61

Balanced Bagging Classifier evalute on validation set

	precision	recall	f1-score	support
cp	0.88	0.96	0.92	23
im	1.00	0.41	0.58	22
imL	0.00	0.00	0.00	1
imS	0.25	1.00	0.40	1
imU	0.40	0.50	0.44	4
om	1.00	0.75	0.86	4
omL	0.50	1.00	0.67	1
pp	0.50	1.00	0.67	5
accuracy			0.70	61
macro avg	0.57	0.70	0.57	61
weighted avg	0.84	0.70	0.71	61

3.3 a_train.csv

The categories distribution of a_train.csv

```
Counter({15: 103,
         7: 383,
         9: 669,
         10: 617,
         8: 553,
         20: 26,
         16: 67,
         19: 32,
         14: 126,
         11: 464,
         12: 264,
         18: 42,
         13: 200,
         5: 115,
         4: 56,
         6: 255,
         21: 14,
         17: 58,
         22: 6,
         1: 1,
         3: 15,
         26: 1,
         23: 9,
         29: 1,
         2: 1,
         27: 2,
         25: 1,
         24: 2})
```

Because the data has some outliers like `label= 1, 26, 2`, I have tried some outlier detection method like isolation tree. However, the prediction result is still not good and many normal data are recognized as outliers. Hence, I just use random oversampling method to balance data and then use four classifiers that have been mentioned in the front.

```
ros = RandomOverSampler(random_state=0)
X_ros, y_ros = ros.fit_sample(X, y)
```

```
brf = BalancedRandomForestClassifier(n_estimators=20, max_depth=2,
                                     random_state=9)
rusb = RUSBoostClassifier(n_estimators=50, learning_rate=1e-2, random_state=0)
eec = EasyEnsembleClassifier(n_estimators=10, random_state=0)
bbc = BalancedBaggingClassifier(n_estimators=10, random_state=0)
```

The evaluation results on validation set and we find that the Balanced Bagging Classifier performs the best. Hence, we choose it to predict on testing dataset.

Balanced Random Forest evalute on validation set

	precision	recall	f1-score	support
1	1.00	1.00	1.00	123
2	0.81	1.00	0.89	141
3	0.35	0.87	0.50	127
4	0.18	0.08	0.11	137
5	0.20	0.02	0.04	135
6	0.00	0.00	0.00	129
7	0.00	0.00	0.00	145
8	0.00	0.00	0.00	142
9	0.00	0.00	0.00	121
10	0.05	0.01	0.01	136
11	0.00	0.00	0.00	125
12	0.00	0.00	0.00	144
13	0.00	0.00	0.00	126
14	0.00	0.00	0.00	143
15	0.00	0.00	0.00	134
16	0.00	0.00	0.00	146
17	0.00	0.00	0.00	133
18	0.00	0.00	0.00	110
19	0.00	0.00	0.00	131
20	0.00	0.00	0.00	131
21	0.04	1.00	0.08	126
22	0.00	0.00	0.00	137
23	0.00	0.00	0.00	125
24	0.00	0.00	0.00	148
25	0.00	0.00	0.00	136
26	0.00	0.00	0.00	146
27	0.00	0.00	0.00	134
29	0.00	0.00	0.00	136
accuracy			0.14	3747
macro avg	0.09	0.14	0.09	3747
weighted avg	0.09	0.14	0.09	3747

RUSBoost Classifier evalute on validation set

	precision	recall	f1-score	support
1	1.00	1.00	1.00	135
2	0.00	0.00	0.00	147
3	0.00	0.00	0.00	153
4	0.00	0.00	0.00	122
5	0.00	0.00	0.00	121
6	0.00	0.00	0.00	145
7	0.06	0.43	0.11	118
8	0.20	0.25	0.23	130
9	0.00	0.00	0.00	138
10	0.00	0.00	0.00	120
11	0.00	0.00	0.00	133
12	0.00	0.00	0.00	141
13	0.00	0.00	0.00	124
14	0.00	0.00	0.00	131
15	0.07	0.85	0.14	135
16	0.00	0.00	0.00	128
17	0.00	0.00	0.00	137
18	0.00	0.00	0.00	142
19	0.00	0.00	0.00	156
20	0.00	0.00	0.00	134
21	0.18	0.33	0.23	129
22	0.00	0.00	0.00	116
23	0.00	0.00	0.00	119
24	0.00	0.00	0.00	129
25	0.33	1.00	0.49	144
26	0.44	1.00	0.61	130
27	0.00	0.00	0.00	147
29	0.00	0.00	0.00	143
accuracy			0.17	3747
macro avg	0.08	0.17	0.10	3747
weighted avg	0.08	0.17	0.10	3747

Easy Ensemble Classifier evalute on validation set

	precision	recall	f1-score	support
1	0.00	0.00	0.00	119
2	0.00	0.00	0.00	144
3	0.07	0.13	0.09	149
4	0.34	0.98	0.50	139
5	0.71	0.11	0.19	139
6	0.41	0.06	0.10	124
7	0.00	0.00	0.00	119
8	0.00	0.00	0.00	151
9	0.00	0.00	0.00	141
10	0.00	0.00	0.00	135
11	0.00	0.00	0.00	139
12	0.00	0.00	0.00	136
13	0.00	0.00	0.00	139
14	0.00	0.00	0.00	118
15	0.00	0.00	0.00	141
16	0.00	0.00	0.00	139
17	0.00	0.00	0.00	125
18	0.00	0.00	0.00	140
19	0.00	0.00	0.00	114
20	0.00	0.00	0.00	129
21	0.00	0.00	0.00	134
22	0.00	0.00	0.00	142
23	0.09	0.12	0.10	139
24	0.00	0.00	0.00	130
25	0.00	0.00	0.00	124
26	0.05	1.00	0.09	139
27	0.00	0.00	0.00	132
29	0.00	0.00	0.00	127
accuracy			0.09	3747
macro avg	0.06	0.09	0.04	3747
weighted avg	0.06	0.09	0.04	3747

Balanced Bagging Classifier evalute on validation set

	precision	recall	f1-score	support
1	1.00	1.00	1.00	145
2	1.00	1.00	1.00	143
3	1.00	1.00	1.00	134
4	0.99	1.00	1.00	131
5	0.92	0.99	0.95	131
6	0.75	0.96	0.84	129
7	0.78	0.74	0.76	160
8	0.52	0.46	0.49	134
9	0.25	0.24	0.25	125
10	0.45	0.29	0.35	138
11	0.64	0.51	0.57	150
12	0.78	0.80	0.79	131
13	0.84	0.94	0.89	121
14	0.92	1.00	0.96	139
15	0.95	0.96	0.96	136
16	0.95	1.00	0.97	129
17	0.97	1.00	0.98	131
18	0.95	1.00	0.97	124
19	0.94	1.00	0.97	136
20	0.98	1.00	0.99	120
21	0.98	1.00	0.99	128
22	1.00	1.00	1.00	132
23	1.00	1.00	1.00	141
24	1.00	1.00	1.00	120
25	1.00	1.00	1.00	137
26	1.00	1.00	1.00	129
27	1.00	1.00	1.00	138
29	1.00	1.00	1.00	135
accuracy			0.89	3747
macro avg	0.88	0.89	0.88	3747
weighted avg	0.88	0.89	0.88	3747

Reference

[1] 机器学习之类别不平衡问题 (3) —— 采样方法: https://www.cnblogs.com/massquantity/p/9382710.html

[2] 不平衡数据集的处理: https://www.cnblogs.com/kamekin/p/9824294.html

[3] imblearn document: https://imbalanced-learn.org/stable/index.html

[4] https://books.google.com.ph/books?id=GvKrDwAAQBAJ&pg=PA452&lpg=PA452&dq=smote+SMOTEENN&source=bl&ots=uD6yhIQ_cZ&sig=ACfU3U01CuiKc1bOPmQgsOdaMUekdiozRA&hl=en&sa=X&ved=2ahUKEwiEsIL_mt3pAhXoy4sBHfVDC38Q6AEwD3oECAoQAg#v=onepage&q=smote%20SMOTEENN&f=false